

Computer-Aided Design and Simulation

of Chemical Plants.

PEETPACK,

A Non-Proprietary Flowsheeting Program.

By

Norman Peters

THESIS  
660-121  
PET

174592

A Thesis Presented For The Award Of The  
Degree Of DOCTOR OF PHILOSOPHY, At  
The University Of Aston In Birmingham.

Department of Chemical Engineering.

June 1974.

## Summary

There are over thirty flowsheeting programs presently in use or at various stages of development in North America and in Europe, however very few of them are non-proprietary and of these none contains a sufficiently wide or general library of unit models and physical property data, or contains an efficient calculation order finder. Many programs suffer also from a poor data interface with the users which makes teaching or learning process difficult and time-consuming.

PEETPACK (Process Engineering Evaluation Techniques Package) described in this thesis is a contribution towards fulfilling the need for a more flexible, complete and above all non-proprietary program. This program has been developed based on the author's experience with a wide variety of other flowsheeting programs, and it attempts to meet a number of criteria of acceptability through its careful programming on local resources to ensure its non-proprietary nature.

PEETPACK is constructed in a highly modular form to facilitate its installation, further development and its ease of understanding by students and industrial users alike. It is composed of six sets of semi-independent programs: the data communicators, the calculation program, a library of general unit models, a library of physical property routines and a set of job control and command programs to link all the segments.

Its data communicators are very flexible and allow the mixing of interactive and batch-mode data input. The library of models contains fifteen models of basic and essential unit operations and these make extensive use of thermodynamic and physical property evaluation routines for hydrocarbons based on well-documented theoretical and empirical methods. Its error tracing and trapping facilities should make its running smooth and help the user diagnose the error sources.

The thesis presents a detailed analysis of three flowsheeting programs (PACER, GEMCS and CONCEPT) which helped in preparing this new program then describes in detail each of the six sections making up the PEETPACK package. The thesis concludes by presenting a new application of flowsheeting programs, that of actual plant-data reconciliation through an analysis of the error distribution in the data.

### Acknowledgements

The author gratefully acknowledges the assistance of the following people in the successful completion of his thesis.

Professor G.V. Jeffreys and the Department of Chemical Engineering, for providing the facilities for the work to be carried out in the department.

Professor P.E. Barker, for his supervision and continuous encouragements.

Dr. J.D. Jenkins, for fruitful discussions on thermodynamic topics.

Mr. P. Abbott, manager of the Computer Centre, the programmers and the operators, for their help and cooperation in learning to use the computer efficiently.

Dean A.I. Johnson of the University of Western Ontario, Canada, for a three-month period spent in his department studying the advances in flowsheeting.

Drs. M. Leesley and P. Winter of the Computer-Aided Design Centre, Cambridge, for a long and fruitful association in using and improving their flowsheeting program CONCEPT, and for valuable discussions on flowsheeting.

Messrs. A.C. Kilikas and S.G. Payne, for providing computer software used in conjunction with the author's own programs.

---

## Table of Contents.

	<u>Page</u>
Summary	
Acknowledgements	
Introduction	i
<u>Chapter 1</u> : The evolution of flowsheeting programs	1
1.1 : The initial phase in computer applications	3
1.1.1 Flowsheeting program organisation	
1.2 : The early years of flowsheeting	5
1.3 : The development years	10
1.4 : The establishment of the large programs	21
1.5 : The revival of academic interest	24
1.6 : Discussion	27
1.7 : Conclusion	29
<u>Chapter 2</u> : A detailed comparison of PACER, GEMCS and CONCEPT	 32
2.1 : The data input	35
2.1.1 The PACER program	35
2.1.2 The GEMCS program	39
2.1.3 The CONCEPT program	42
2.2 : The calculation phase	47
2.2.1 PACER	48
2.2.2 GEMCS	50
2.2.3 CONCEPT	52
2.3 : Unit models and thermodynamic data	57
2.3.1 The transmission of variables	58
2.3.2 The thermo-physical property routines	59
2.4 : Initial, intermediate and final printout	60
2.4.1 The data echo	61
2.4.2 The intermediate printout	62
2.4.3 The final printout	64
2.5 : Conclusion	65

<u>Chapter 3</u>	: Application of the simulation programs	66
3.1	: Example One	66
	3.1.1 The plant description	68
	3.1.2 Proces modifications	69
	3.1.3 Modelling considerations in PACER and GEMCS	71
	3.1.4 Modelling considerations in CONCEPT	75
	3.1.5 Process simulation conditions	77
	3.1.6 The data input	78
	3.1.7 The calculation order	80
	3.1.8 The results	82
	3.1.9 General comments	88
3.2	: Example Two	89
	3.2.1 The application of the optimiser	92
	3.2.2 The results	95
3.3	: Conclusion	96
<u>Chapter 4</u>	: A critical appraisal of the programs and the need for a new one	97
4.1	: Case 1, The undergraduate student	98
4.2	: Case 2, The postgraduate researcher	102
4.3	: Case 3, The practicing engineer	104
4.4	: The importance of choosing a good program	106
4.5	: The need for a new flowsheeting program	106
	4.5.1 PEETPACK	110
<u>Chapter 5</u>	: The general outline of PEETPACK	111
5.1	: Meeting the objectives	112
5.2	: The structure of the package	113
5.3	: The package sections sequencing	115
5.4	: The two types of executive programs	123
5.5	: Initiating a run on PEETPACK	126
5.6	: Major implementation drawbacks in PEETPACK	128
	5.6.1 Size problems	128
	5.6.2 Extended Fortran facilities	129
	5.6.3 Compiler library routines	129

5.6.4	The MACRO programs	130
5.6.5	LIBRARY's of routines	131
5.7	: Conclusion	131
 <u>Chapter 6</u> : The data input and output facilities		134
6.1	: The batch-mode data input	139
6.1.1	Set-A commands	140
6.1.2	Set-B commands	143
6.1.3	Set-C commands	143
6.2	: The interactive-mode data input	144
6.3	: The order of data input	147
6.4	: The CALCULATE stage	149
6.4.1	The gathering of property data	149
6.4.2	The routine calling sequence	150
6.4.3	The data dump	151
6.5	: Data recognition	154
6.5.1	Level one of data recognition	154
6.5.2	Level two	155
6.6	: The Free-Alphanumeric reading Format	156
6.6.1	The reading procedure	157
6.7	: Conclusion	160
 <u>Chapter 7</u> : The calculation order finder		161
7.1	: The initial ordering attempts	163
7.2	: The location of recycle loops	165
7.2.1	The adjacency matrix method	165
7.2.2	Exhaustive path searching	167
7.2.3	The method used	169
7.3	: The search for iterate streams	173
7.3.1	The method used	176
7.4	: The calculation order	179
7.4.1	The method used	181
7.5	: Conclusion	187



9.11	:	Function VISC	263
9.12	:	Function CNDVTY	268
9.13	:	Function SURFTENS	275
9.14	:	Conclusion	278
 <u>Chapter 10</u> : The EXECUTIVE program			279
10.1	:	The MASTER segment	280
10.2	:	Routine DREAD	283
10.3	:	Routine DPRINT	283
10.4	:	Routine COMPUT	284
10.5	:	Routine LOADST	286
10.6	:	Routine EQCALL	292
		10.6.1 Version 1	292
		10.6.2 Version 2	293
10.7	:	Routine TEST	294
10.8	:	Routine PUTSTM	294
10.9	:	The convergence of recycle loops	295
		10.9.1 Unaided convergence	295
		10.9.2 Accelerated convergence	295
		10.9.3 Analytical methods for convergence	300
10.10	:	Routine PROMOT1	302
10.11	:	Routine PROMOT2	304
10.12	:	Routine RESULT	305
10.13	:	Conclusion	305
 <u>Chapter 11</u> : Practical applications of PEETPACK			306
11.1	:	Example One	306
		11.1.1 The plant instability	309
11.2	:	Example Two	329
11.3	:	Example Three	338
11.4	:	Conclusion	349
 <u>Chapter 12</u> : "Beyond conventional flowsheeting - Discussions and ideas"			350
12.1	:	What can still be done ?	351

12.2	:	"Beyond conventional flowsheeting"	354
		12.2.1 Problem statement	355
12.3	:	The methods of data reconciliation	356
		12.3.1 Their advantages and drawbacks	358
		12.3.2 The Kuehn and Davidson reconciliation method	360
		12.3.3 The advantage of a flowsheeting context	363
12.4	:	The implementation of the Kuehn and Davidson method in PEETPACK	364
		12.4.1 The use of the program	364
		12.4.2 The program description	365
12.5	:	Examples	368
		12.5.1 Example One	368
		12.5.2 Example Two	371
12.6	:	Conclusion	373
 <u>Chapter 13</u> :			374
Conclusion and recommendations			
13.1	:	Recommendations for future work	379
		13.1.1 Further improvements	380
		13.1.2 Long-term additions and improvements	381
13.2	:	The future of flowsheeting programs	383
		13.2.1 Industry	383
		13.2.2 University	384
 <u>Bibliography</u>			385

Volume II :

Supporting Publications

Public presentations in symposia

Appendix 1.:

Code sheets for data input to PACER.

Input data to PACER for the ethane-ethylene separation plant.

Input data to GEMCS for the ethane-ethylene separation plant.

Appendix 2 :

Description of the Hooke and Jeeves optimisation method.

Flowchart of the optimisation routine.

Listing of the optimisation routine.

Appendix 3 :

Commands to compile and save the programs making up the  
PEETPACK and PROPACK programs.

Listings of these programs.

---

List of figures and tables.

	<u>Page</u>
<u>Chapter 1 :</u>	
Figure 1-1 : Typical flowsheeting program organisation	31
Table 1-1 : Some academic flowsheeting programs	19
Table 1-2 : Some industrial flowsheeting programs	20
 <u>Chapter 2 :</u>	
Figure 2-1 : Examples of data input to CONCEPT	44
Figure 2-2 : CONCEPT's data input and updating commands	46
Figure 2-3 : The simplified PACER flowchart	54
Figure 2-4 : The GEMCS flowchart	55
Figure 2-5 : The CONCEPT Mark 3 system	56
Table 2-1 : Part of PACER's process matrix	37
Table 2-2 : Feed streams and initial guesses	37
Table 2-3 : Some of GEMCS' equipment arrays	40
 <u>Chapter 3 :</u>	
Figure 3-1 : The ethane-ethylene separation plant	67
Figure 3-2 : The modified plant diagram	70
Figure 3-3 : Three-stage cross-current extraction plant	93
Figure 3-4 : The equilibrium relationships	93
Table 3-1 : Survey data and simulation results	83
Table 3-2 : Unit parameters for the ethane-ethylene plant	87
 <u>Chapter 4 :</u>	
Table 4-1 : Requirements in simulation programs for different users	99

## Chapter 5 :

Figure 5-1	:	The structure of PEETPACK	114
Figure 5-2	:	The flowchart of the PEETREADER MACRO	116
Figure 5-3	:	The "MODELSUBS" LIBRARY creation program	132
Figure 5-4	:	The "THERMOSUBS" LIBRARY creation program	132
Figure 5-5	:	The "PHYSUBS" LIBRARY creation program	133
Figure 5-6	:	The "EXECUTIVESUB" LIBRARY creation program	133
Table 5-1	:	Commands to initiate PEETPACK	119
Table 5-2	:	The segments sizes	124

## Chapter 6 :

Figure 6-1	:	Batch-mode data input commands	137
Figure 6-2	:	Interactive-mode data input commands	138
Figure 6-3	:	The COPYER routine flowchart	159

## Chapter 7 :

Figure 7-1	:	Flowchart of the loops finder routine	171
Figure 7-2	:	Flowchart of the iterate stream finder routine	178
Figure 7-3	:	Flowchart of the calculation order finder	183
Figure 7-4	:	Flowchart of routine FLAGGER	185

## Chapter 8 :

Figure 8-1	:	Flowchart of routine COMPRESR	201
Figure 8-2	:	Flowchart of routine HEATEXCH	205
Figure 8-3	:	Flowchart of routine DISCOL	214
Table 8-1	:	The parameters for the PEETPACK models	191

## Chapter 9 :

Figure 9-1	:	Flowchart of routine KVALUE	226
Figure 9-2	:	Flowchart of routine ENTHALPY	233
Figure 9-3	:	Flowchart of routine FLASH	238
Figure 9-4	:	Flowchart of routine DEWBUB	242
Figure 9-5	:	Flowchart of routine TEMP	246

Figure 9-6	:	Flowchart of routine DENSTY	250
Figure 9-7	:	Flowchart of routine SPHEAT	261
Figure 9-8	:	Flowchart of routine VISC	266
Figure 9-9	:	Flowchart of routine CNDVTY	273
Figure 9-10	:	Flowchart of routine SURFTENS	277
Table 9-1	:	Chemical compounds catered for in PEETPACK's property files	220
Table 9-2	:	Data set in the PROPTS data file	222
Table 9-3	:	Data set in the PPTEXTRA data file	251
Table 9-4	:	Equations for gaseous thermal conductivity	271

#### Chapter 10 :

Figure 10-1	:	Organisation of the Executive	281
Figure 10-2	:	Flowchart of routine COMPUT	287
Figure 10-3	:	Flowchart of routine LOADST	290

#### Chapter 11 :

Figure 11-1	:	Batch-mode data input to PEETPACK	310
Figure 11-2	:	Interactive-mode data input to PEETPACK	312
Figure 11-3	:	Dumpfile for the ethane-ethylene separation plant problem	317
Figure 11-4	:	Example of TRACE 0 mode output	320
Figure 11-5	:	Example of TRACE 1 mode output	321
Figure 11-6	:	Data echo	
Figure 11-7	:	Updating commands in batch-mode for the ethane-ethylene plant	328
Figure 11-8	:	The mixer plant diagram	333
Figure 11-9	:	First iterate stream convergence (no initial estimates)	334
Figure 11-10	:	First iterate stream convergence (with initial estimates)	335
Figure 11-11	:	First iterate stream convergence (no initial estimates)	336
Figure 11-12	:	First iterate stream convergence (with initial estimates)	337

Figure 11-13 :	The fractionation plant diagram	341
Figure 11-14 :	Data echo for the flash problem	342
Table 11-1 :	Results for the initial simulation	346
Table 11-2 :	Results for the improved simulation	347
Table 11-3 :	Flows for the mixer plant example	348

Chapter 12 :

Figure 12-1 :	Diagram for Example One	370
Table 12-1 :	Reconciled flowrates for Example One	370
Table 12-2 :	Measurements and reconciled flows in Example Two	372

## INTRODUCTION

The evolution of complex and continuously changing technology has forced the engineer to turn to more sophisticated computing methods to ease and accelerate his work, and this search for more efficient mathematical tools had led to the development of computer programs, that solve the heat and mass balance equations describing large sections of plants in both their steady- and their unsteady- or dynamic- operating states.

These programs first appeared in the late 1950's when digital computers had become large enough to accommodate a few simple unit model routines residing in core simultaneously. However the high cost and narrow-spread of these computers and the novelty of this application slowed its acceptance until the 1960's. The development of the larger and faster computers and their increased use in engineering work in the early 1960's prompted the creation of more standard-types of simulation programs incorporating more elaborate models and their associated property data evaluation routines. These were soon called "Flowsheeting Programs" as the logic followed in their fairly-standardised format and the way in which units in a plant were represented by auxiliary routines controlled by a central executive, both resembled the logical approach in drawing program flowsheets; similarly the transfer of flow in the pipes connecting the plant units was represented by the transfer of information between the various segments of a program, thus strengthening the analogy.

In recent years the advent of the very large and extremely fast computers has caused flowsheeting program developers to direct their improvements along two different lines. The first one is a continuation of the established trend of increasing the accuracy of the calculations by refining the unit models and the data estimation methods, the other is the incorporation of new features and facilities such as mechanical design, cost evaluation and plant performance optimisation.

From an earlier Ph.D. thesis by A.J. Leather entitled "Computer-Aided Flowsheet Design and Simulation of Chemical Plant" and from a search of the literature, it became evident that most of the flowsheeting programs were inaccessible to Universities and much of the British chemical Industry due to the proprietary nature and/or cost of the programs that had been written. From a preliminary study of a few of the accessible programs, it was considered that a superior flowsheeting program could be written. This then became the prime objective of this research, namely to attempt to write a more efficient and more easily usable flowsheeting program, and after suitable testing, to make such a program generally available to the chemical Industry and to institutions of advanced study.

A secondary objective was to seek new fields of application for flowsheeting programs.

## CHAPTER 1

### THE EVOLUTION OF MODULAR FLOWSHEETING PROGRAMS

The evolution of increasingly complex technology and the fast rate of product, plant and idea obsolescence has forced the process engineer to turn to more sophisticated computing methods to aid him in his task and accelerate his work. The evaluation of alternative processing schemes for new plant design and the investigation of operating plant behaviour under different running conditions are two fields in which new computing techniques can assist the engineer. Programs written for plant simulation can be used to recommend optimum operating conditions without the necessity of actual plant experiments.

Flowsheeting programs perform the mass and heat balances associated with large chemical processes consisting of many interconnected units often with recycle streams. The size and complexity of the plants under consideration are reflected in the work needed to generate the programs for their simulation, and if a new program had to be created for each new application the improvements in the design or operating performance so produced could be outweighed by the effort and time taken in preparing the new programs. However flowsheeting programs may be composed from readily available unit models to simulate new plants, and models would need to be prepared only for the new process units in the plant.

The work involved in reaching the degree of sophistication

many modern flowsheeting programs exhibit has spanned many years of continuous research and development since the electronic calculators were first commercialised in the late 1940's. This chapter attempts to present a chronological survey of computer-aided design of chemical plants and classifies the developments over four, time periods.

1. The initial period in computer applications, when computers were still novelties and their use rather narrow.
2. The early years of flowsheeting which saw the linking together of various models to simulate plant sections.
3. The development of these programs into large flowsheeting programs requiring minimum effort in their use on the part of the engineer.
4. The present-day large flowsheeting programs incorporating design and cost estimation features.

## 1.1 The initial phase in computer applications (Up to 1960)

The analog computer was developed earlier than the digital computer. It was used as early as 1948 and one of its first recorded applications was to study the thermal problems in fixed bed catalytic convertors (2), and two years later phase equilibria for flash vapourisation were also calculated using analog circuits (3). This type of computer was however very slow and could not be used to study steady state heat and mass balances. Its use was retained for the study of single unit operations where transient or unsteady-state chemical and physical operations were present.

With the advent of card-programmed digital computers in the mid-1950's more receptive to steady-state repetitive design techniques, the simulation activity increased rapidly, first in building-up fairly sophisticated unit models then in tying them together to obtain the response of a train of units subjected to known operating conditions.

One of the earliest published attempts at unit simulation on a digital computer dates back to 1956 when an advanced heat exchanger program was discussed at the 21st Midyear meeting of the American Petroleum Institute (4, 5). Another proposition dating back to 1958 was that of a new approach to the modelling of distillation columns by computers (6). These programs were followed by other ones proposing improved methods for heat exchangers simulation and design in 1959 (7), 1962 (8) and even later.

These programs were thorough, accurate and general and the latter one would even estimate the cost of the optimum heat exchanger configuration chosen by the program. Although these programs led to the development of numerous other programs of the same kind (9) these models were never, then, tied-up together to simulate a complete section of plant. A more general program for steady state simulation of a set of units was however postulated in 1958 (10) but did not materialise until a few years later as the forerunner of the "Flexible Flowsheet" by the M.W. Kellogg Company (11, 12).

The first article on the computer solution of material balances in a set of units operating together was published by Rosen in 1962 (13). The proposed method was applied to the solution of simultaneous non-linear equations. The equations were first linearised by hand then solved iteratively "en bloc" rather than by representing the equations describing each unit by a separate routine.

The evolution of increasingly powerful and sophisticated computer programs for the simulation and design of chemical plants, since the two attempts mentioned above were publicised, has followed two different paths.

1. The simultaneous-equation approach, where all the equations describing all the units in the plant are solved simultaneously, in a large program possibly broken down into sets of routines. This method is represented by the

Rosen approach (13).

2. The modular approach method where each unit is represented by a semi-independent routine called in a program- or user-specified sequence by a main program which supplies it with its feed and parameter conditions and does the book keeping of the flows in and out of it. The units calculation sequence is repeated until convergence is achieved. This method was postulated by Kesler and Griffiths (11).

Both methods have evolved simultaneously since the early 1960's and are still being improved today, but the modular approach has gained a quicker and more wide-spread acceptance for three reasons.

- 1 - Its apparent flexibility and ease in piecing together a new problem from existing or new unit model routines.
- 2 - The nearly one-to-one correspondence between the models used and the plant configuration.
- 3 - The information flow which is usually in the same direction as the material flow and thus easy to visualise.

The simultaneous-equation approach tends to solve the equations describing the plant much faster because of the more advanced mathematical techniques applied to the solution of large sets of simultaneous equations, but because of the difficulty in visualising the steps in the solution, it is more difficult to trace the causes of errors in faulty runs, and to understand and easily apply the modelling concepts.

The chronological development of the modular approach programs, the type related to the author's research, will be

followed below but to understand the general development of flowsheeting programs, a short description of the standard program organisation is helpful.

#### 1.1.1 Flowsheeting programs organisation

A flowsheeting program starts by reading the plant information data such as the units interconnections, the feed conditions, the units operating conditions and maybe the order in which to calculate the units. If this sequence is not provided, the program may analyse the plant diagram to locate the sequential and the recycle sets of units. The executive program then calls each unit model in turn according to the calculation order and transmits to it its operating parameters and its input streams descriptions. The model calculates the heat and mass balances and outputs the output streams conditions. Sets of units in recycle loops are calculated repetitively until convergence on the steady-state behaviour of the units is obtained. Certain programs may accelerate the convergence of the calculations by applying special mathematical manipulations to the streams values at every iteration. At convergence the final results are printed out and in some programs they may be stored on magnetic tapes for further runs. This program organisation is illustrated in figure 1.1.

Each of the five sections making-up a typical program as described above (the data input, the calculation order finder, the main executive, the models and physical property packages and the

data print out) has been developed and written in a different way by the various program developers, but none has received as much attention as the problem of ordering the units in the optimum calculable sequence. This section which is at the heart of every complete program will be dealt with separately in chapter 7. The other sections have usually been treated together in the literature, except for the convergence accelerators, and a survey of their development is presented below. The convergence promoters are usually treated as specific mathematical topics and only occasionally mentioned in flowsheeting program descriptions. Their development is summarised in chapter 10 which deals with the development of the new flowsheet executive program.

## 1.2 The early years of flowsheeting. (1960-1964)

The M.W. Kellogg "Flexible Flowsheet" program is one of the earliest successful simulators. Developed around 1960, it was only described in 1963 (11). It consisted of a variety of unit models called by a supervisory program which did the book-keeping and interpretation of engineering data. Technical and thermodynamic data were also included. A straightforward substitute method was used to solve recycle loop calculations, but recycle streams were initially assumed to bear no flow which would suggest in the author's experience that very simple models were used. No ordering procedure was incorporated.

Of the early industrial programs developed at about the same time as the Flexible Flowsheet, the few known ones are the "Bonner and Moore" program (around 1960) and the "Generalised Interrelated Flow Simulator" (GIFS, 1962). Little has been published about them (17) except that they consisted of a master program calling unit models from a library in a user-specified order.

Sargent and Westerberg (14) were the first academic researchers in Britain to tackle the modular approach. They announced the development of a new flowsheeting program "SPEED-UP" which never materialised (although a new version of SPEED-UP is presently in preparation at Imperial College, London). In their paper they described the scope of their new program which was more advanced than that of any other publicised program. New requirements were laid down emphasizing the ease and speed of data preparation and input and consequently the need for an automatic calculation order. Their approach to data input did away with coded input cards in rigid formats, in favour of an English language type of input which made data preparation very understandable but at the expense of more complex programming. They also provided basic algorithms for ordering the plant units in a calculable sequence (discussed in chapter 7), but their paper failed to give evidence of the usefulness of their work and the feasibility of the rather complicated data interpretation and unit ordering methods.

Ravicz and Norman also developed a flowsheeting program in 1964 (15), but in a way that seemed to include both the modular and the simultaneous-equation approaches. Their program is set up to sense non-recycle parts of the flow diagram and compute them sequentially, reserving the simultaneous method for those parts that contain recycles. Units are calculated only if their inputs are known. Recycle units are treated in a special compiler that "senses" their mass balance equations and sets them aside to be solved by a multi-dimensional Newton-Raphson technique after being compiled, linearised and reduced in number to the smallest set possible to save computing time. This compilation of recycle equations which is the base of their program was not described, nor do the authors elaborate on the methods of locating their equations. It does seem though that their models must be written according to a fixed and rigid format to ease the compilation. The paper avoids also commenting on the ease of use or generality of the program and no solved examples are presented.

These early flowsheeting years helped promote an ever-increasing interest in the field of computer-aided design but their direct influence on the practical development of flowsheeting programs was not very strong. They were years during which developers experimented with various approaches and formats for these programs but no single approach was really accepted. The emphasis remained on the simultaneous-equation approach developed by Nagiev (16) and promoted by Rosen (13) chiefly because of the known mathematical tools employed but also because of the smaller

computer core needed. The small size of computers must have required a great deal of user intervention in each simulation to sequence the necessary programs so that additional intervention to arrange the sets of equations describing each simulation was not as unfeasible as it is at present.

The modular-approach concept which slowly evolved from papers such as Kesler and Griffiths's (11) was probably finally established as a feasible method when Sargent and Westerberg treated for the first time the problem of locating and tearing recycle loops as an integral part of the modular flowsheeting program. Unfortunately, the aura of secrecy that surrounded the development of the better programs of this period is such that these were only known by name or through sales literature (17, 1). It is also quite possible that other programs may have been written during this period but no published evidence exists as to their sources or contents.

### 1.3 The development years (1964-1969)

The application of the programme, disclosed until 1965, to any practical problem was never demonstrated in the literature and thus must have left little impression on the readers. The first program to have a real impact was PACER. Developed originally as an M.Sc.Thesis in 1964 (1), PACER was first referred to in an article (18) describing very briefly its features, its logic and its potential uses. It was developed further at

McMaster University (Hamilton, Canada) and later described in more detail in an article by Johnson et al (19). This article complemented the first one as it led the reader through all the phases involved in the preparation and running of a typical plant simulation problem, that of a complete sulphuric acid plant based on real industrial data. It also showed how models could be written to simulate various new units and it compared the simulation results with actual plant data. It assessed the significance of the program from both its teaching and its industrial benefits. Overall it was the most thorough and convincing article to be published to that date on the uses and advantages of computer simulation programs.

These articles and the partial non-proprietary nature of PACER (only to academic institutions) had a great influence in spreading the knowledge and acceptance of flowsheeting and promoting the interest of other researchers in associated fields such as generalised property estimation methods and graph theory applied to unit calculation sequencing.

At about the same time as PACER was developed at University, a few new industrial programs were also appearing but about which very little was being disclosed, either because of their proprietary nature or because they were not released at that time for general use. Some were eventually offered to paying users but little more than their user manual was ever published. These were:

CHEOPS, developed by Shell over many years, was applied mainly to the simulation and design of refineries and petro-chemical plants. Its existence was barely mentioned (17, 20).

CHIPS, written by I.B.M.'s Service Bureau Corp. (17, 28) as an improved version of GIFS, contained many unit models and a number of sets of physical and thermodynamic property data entered in tabular form (17). CHIPS was claimed to be of use in four areas of applications (21, 28):

Planning, designing or alternative equipment use; computing operating conditions for changes in feed; production relationships for changes in output as function of operating conditions, and equipment performance and decline studies. Though of sales-promotion type this information does hint to its extensive library of models for simulation and design purposes which also included cost estimation methods. However the scope and differences between these four applications of a similar nature was not discussed.

The Chevron System was developed by Chevron Research Co. from the old Ravicz and Norman program (15) mainly for organic compound plants, but allowed the user to input additional data for compounds not catered for already (17). It contained a special program to read the data which consisted of a mixture of words and numbers.

MAEBE, although not of industrial origin but written at the University of Tennessee, has also received little publicity (17). It is only known that it used Rubin's approach to reduce the number

of unknown recycle parameters (21) (see chapter 7) then applied an unaccelerated successive substitution to converge the answers, whereas CHIPS and CHEOPS used convergence promoters.

The advantages of these programs and their novel features, if any, were not disclosed nor described in any convincing manner.

A fairly thorough but condensed survey of most of the programs mentioned above was published in 1968 (17). It used not only literature material but also unpublished communications from numerous Companies and Universities, analyses of industrial programs and their user manuals, and a large set of references not normally collected in University libraries. In it the authors re-present and clarify the basic framework of flow-sheeting and of the presented programs and they summarize all the published attempts at the solution of the unit ordering problem and the convergence of simulation runs. The benefit of this publication was in surveying the mathematical tools employed by all known programs and in presenting information otherwise difficult to obtain. Its contents were used extensively in presenting some of the above-mentioned information on industrial programs and as a base for other more recent surveys (1, 24, 25, 26, 27).

A second survey (24) soon followed the one above, except that being written as a course refresher, it concentrated on the general aspects and goals of flowsheeting and presented the various

sections making up simulation programs in a simplified manner to engineers unaware of this powerful tool. It did not however throw any fresh light on flowsheeting concepts nor did it offer any information on any one program, but merely repeated and condensed published information (11, 15, 17, 19).

Up to this time, the development of flowsheeting programs was mainly the concern of Industry, save for the successful improvement and use of PACER at McMaster University. With the acceptance of flowsheeting programs as potential teaching tools, the attention was again focused on PACER but more from the aspect of training students in the science of mathematical modelling and in reconciling models with actual plant data. This was achieved through the simulation of an actual synthetic rubber plant, by A.I. Johnson and co-workers (29). The paper furnished a detailed modelling sequence and parametric study and gave comparisons with actual runs, but no additional information regarding the program itself was included. The main recommendation in this paper was for closer collaboration between Industry and University to make actual plant data available to students and to help Industry in its modernisation and acceptance of this new tool.

This academic development work on PACER led to the publication of the articles mentioned above and, more important, to the publication of one of the rare books on the whole subject of steady-state chemical plant simulation by computers (23). This excellent book by the six co-workers on the PACER project

elaborates in sufficient details on all phases of simulation and classifies all its aspects which had been only mentioned or vaguely discussed previously. It used PACER as the executive program to which all examples and models were tailored, and gave detailed simulation studies of a number of industrial plants. This book is unique of its kind in that it presents the strategy of simulation not just as a case history but also attempts to answer technical questions about real non-linear processes with realistic objectives and industrial constraints. Some of the mathematical methods presented are slightly outdated, but the strategies proposed are not time-sensitive.

CHES, developed about 1968 at the University of Houston, Texas, is a more recent successful academic flowsheeting program to benefit from many years of development by a small group of researchers before being well publicised in 1971 (40) and 1972 (41). Derived from PACER, it was designed primarily for simulating hydrocarbon systems where emphasis is laid on phase equilibria and enthalpy calculation in addition to mass balancing (40). Its comprehensive physical property package supplying information on 65 compounds was its original major feature, whereas PACER had no data package. CHES has since been extended to include equipment design and sizing, costing, plant investment analysis and economic return computations (41) thus classifying it amongst the larger programs described in the next section. Though it is now one of the large established programs, it had the most academic impact after PACER because of

its partial non-proprietary nature and its large property package.

The first program of the newer generation of industrial programs to be partially described in the literature is NETWORK 67 of Imperial Chemical Industries. In a series of articles (30, 31, 32) Andrew demonstrated its features through a number of simulation and design calculation examples. In the first two articles some of its particularly highlighted features emphasised more its deviations from what seems to be the accepted conventional approach to flowsheeting programs. These deviations are as follows:

1. The data input is far more complicated than for other programs as the user must write programs to control the simulation, set the unit connections and input the data, thus requiring a more tedious run preparation.
2. The plant streams contain not only their flow description but also the unit parameters, all set in one array for each unit. The stream descriptors also vary from unit to unit and no set pattern is proposed.
3. The program does not contain property data banks.
4. It does not order the units in a calculable sequence.

Overall NETWORK 67 is described as a once-off program (i.e. completely re-tailored for each new application) useful for quick (?) calculations of relatively simple problems where computation efficiency is not important. But at the cost of how much user work?

The third paper (32) discussed the simulation, design and optimisation of a complete process starting with the development of the mass-balance program and gradually increasing its complexity to incorporate detailed process models, cost routines and optimisation algorithms. This seems to diminish the value of NETWORK 67 as Andrew states that emphasis was placed on making the control processor as efficient as possible to speed-up the multi-variable optimisation (rather than improving NETWORK 67 to meet the requirements!).

At about the same time, around 1967, another flowsheeting program was being developed at Cambridge University. The Chemical Plant Calculating system (C.P.C.) later renamed CONCEPT. CONCEPT is probably the only working program which had its calculation ordering program extensively discussed (33), though not clearly enough to facilitate its reproduction, long before the program's scope and facilities were presented at various symposia in the early 1970's. CONCEPT has been developed and handled with as much confidentiality as most other proprietary programs, its development having been commissioned and paid for by the Department of Trade and Industry for use in its own consulting bureau (The Cambridge Computer-Aided Design Centre). CONCEPT is another program that took advantage of previous work both in its calculation order finder (see chapter 7), and in its overall design based on a PACER type approach. The major innovation developed for CONCEPT was an interactive drawing program for the

visual display and input of the simulation data (34), NETDES 20. NETDES 20 allows an easy construction of pictures which are sufficiently like a flowsheet to be a useful visual aid. It is connected to CONCEPT via a Transfer Zone from which CONCEPT reads the interpreted data. (At the time of the joint project between the author and the C.A.D. Centre (see chapters 2 and 3) this facility was not operative and no practical details about it could be obtained).

CONCEPT represents on the British scene the established large and general simulation program of the accepted standard type of program. In the United States this state of the art had been reached much earlier with programs such as CHIPS and CHEOPS.

It is obvious that with the advent of larger and more powerful computers, flowsheeting programs were endowed with more features and larger banks of unit models and physical properties. Similarly the data input was improved from being merely sets of unintelligible numbers such as in PACER (19) to more comprehensive sets of names and figures as initially heralded by Sargent and Westerberg (14) and later used in the Chevron System and other programs.

These were the main characteristics of this flowsheeting period which also saw the development of good calculation order finder methods and strengthened the position of these programs as indispensable tools though still chiefly simulation orientated.

Program	Source of Origin	Year of Release
PACER	Purdue University McMaster University	1964 1965
SPEED-UP	Imperial College, London	Not completed
MAEBE	University of Tennessee	Not released
CONCEPT	Cambridge University	1968
CHESS	University of Houston	1969
GEMCS	McMaster University	1969
MACSIM	McMaster University	Not released
SLED	University of Michigan	1971
ASCEND	University of Florida	1971
GPD	University of Western Ontario	1971
GEPDS	University of Oklahoma	?
ESSPROS	Edinburgh University	1972
PEETPACK	University of Aston	1974

Table 1-1 : Some academic flowsheeting programs

Program	Source of release	Year of Release
Flexible Flowsheet	M. W. Kellogg Co.	1962
GFS	M. W. Kellogg Co.	?
GIFS	Service Bureau Corp.	1962
CHEOPS	Shell Development Corp.	1963
Chevron System	Chevron Research Co.	1964
CHIPS	Service Bureau Corp.	1966
Network 67	I.C.I. Ltd	1967
Network 68	I.C.I. Ltd	1968
PROVES	Diamond Shamrock Corp.	1969
GPS	Phillips	?
PDA	Phillips	?
FLOWPACK	I.C.I. Ltd	1970
FLOWTRAN	Monsanto	1970
Exploratory Evaluation Package	I.C.I. Ltd	1971
BASYS	I.C.I. Ltd	1971
CAPS	Chemshare Corp.	1971
FLAWSIM	Interdisciplinary Technical Systems Corp.	1971
Cyanamid Simulation System	American Cyanamid Co.	?

Table 1-2 : Some industrial flowsheeting programs

#### 1.4 The establishment of the large programs (since 1969)

This third phase in the development of flowsheeting programs represents the establishment of the very large and general programs, which include most of the tedious mechanical and economic evaluations, as the standard type of programs. They have now come to be accepted as essential tools in design work and little new basic research work is necessary any longer in the purely modular-type package.

A typical example of this new kind of integrated systems is PROVES (Project Valuation and Estimation system) (35, 36, 37) prepared by Diamond Shamrock Corp. This system is written in four individual but complementary segments.

MODEL calculates material balances based on a simplified process simulation.

SCOPE yields equipment specifications and requirements for utilities.

INVEST computes detailed investment costs, usage rates and the required number of operators for the processes.

EFFECT estimates approximate total investment costs, detailed manufacturing costs, forecast sales volumes and selling prices and determines profitability and sensitivity towards various technical and economic parameters.

This very versatile package can be used either as an integrated system, each program feeding its results as data to the next one, or as four independent programs requesting their

data directly from the user; however, because of its complexity and the size of the whole set of programs written to fit a 32 K computer or a 16 K computer with disc backing store, the accuracy and generality of its models and the thorough investigation by its economic package is limited. Details about the executive program were not given but the different segments were well described, showing some of the models provided for the mass balances and the equations used to achieve the economic evaluations.

FLOWTRAN, developed by Morsanto in the U.S.A., is another design orientated package of the new era (38). Like PROVES it may be assembled from a number of program blocks to offer any combination of features the engineer needs to design and cost-evaluate his process. But unlike PROVES it yields less cost estimates and sales volume forecasts, and its more detailed models produce more accurate results since the program size is not a constraint. Its control blocks can also adjust the values of the process parameters and variables to yield specified plant performances, but to the author's knowledge no information has been revealed concerning the methods used to affect these adjustments. FLOWTRAN has its own library of physical property evaluation routines covering over 200 organic and inorganic compounds over wide ranges of temperature and pressure, and its physical and chemical procedures handle ideal, regular and non-ideal solutions, mixtures and two-phase flows. It has also a very large library of unit models of different complexity and

accuracy. These extensive features make FLOWTRAN today's best and most complete package. It was offered for use to paying customers in the U.S.A. between 1970 and 1972 but because of its low economic profitability, it was withdrawn from the industrial scene and made available to Universities in 1973 at a minimal cost, in an unusual form of aid to education (39).

Such large all-encompassing programs have not yet appeared on the British market. CONCEPT is one of the two most advanced packages here, the other one being FLOWPACK of I.C.I., an improved version of Network 68 (the successor to Network 67) (42). This proprietary package is also mainly a simulation orientated program but with a very large thermo-physical property package and unit models library.

The reason is not clear whether this situation arises because of the conservative attitude of British Industry towards computers, the little demand for these large packages or because British firms prefer to segregate the various tasks performed in these packages into numerous small programs each adaptable to the case in hand. However, this trend towards larger and more general programs will soon be followed and the Computer-Aided Design Centre is already preparing an economic evaluation package to incorporate into CONCEPT Mark III and its physical property package is growing steadily in size and range.

Since about 1971 the interest in flowsheeting in both industry and academic institutions has risen to such a high level

that a voluminous number of publications and presentations are being made every year at innumerable regional and international symposia on the subject of flowsheeting. There is also a tendency for certain authors to repeat themselves at different meetings. These repeat performances were omitted from this survey for brevity.

#### 1.5 The revival of academic interest.

By the end of the 1960's academic interest in flowsheeting had fallen well behind the growing industrial developments and innovations. Besides a few Universities where programs such as PACER and CHESS were still being used the majority of academic institutions showed little interest towards this expanding specialisation. Obviously the demands and scopes of application of flowsheeting programs are very limited in Universities thus reducing the areas of productive research, but even the mere interest in the subject seemed restricted to certain centres.

This lack of interest may well have motivated Professor Johnson in publishing a new series of articles on the subject, treating it again from an academic point of view to promote new interest (43, 44, 45, 46). The presentation of two new flowsheeting programs, GEMCS and DYNYSYS, of very simple structure and quick appeal to students, highlighted this series. DYNYSYS is a transient- or dynamic- state simulation program of interest in the study of plant start-ups, shut-downs or transient behaviour

due to feed changes. It follows the modular-approach concept of simulation but makes different demands on the writing of the unit models which usually incorporate differential equations. The major difference between this type of program and a steady-state simulation program is in the presence of a general ordinary differential equation solver in DYNSSYS which in a way replaces the convergence promoters in steady-state programs. This type of program has yet received little attention and cannot be generalised upon. GEMCS is a very simplified steady-state modular simulation program. Its chief attraction is in its absolute compactness and simplicity making it ideal for teaching purposes where students are expected to provide their own models and data routines. GEMCS is described in detail in chapter 2.

The effect of these articles and more particularly that of the general availability of GEMCS, as the first non-proprietary program, has been remarkable. GEMCS has become one of the most popular flowsheeting programs in Universities throughout North-America and Britain, and a number of new academic programs ESSPROSS (47), UNICORN (48) have been modelled directly on it. Its other major advantage is its limited computer core demand, putting it well within the computing facilities of most Universities.

The revival of academic interest since the publication of Johnson's programs was demonstrated in two recent surveys of academic and industrial flowsheeting developments. The first one (49, 1) presented a classification of the new terminology

that has appeared in connection with flowsheeting and listed all known flowsheeting programs, tabulating the sources of information available about each one and categorising them from the point of view of the fundamental approach they use and their present state of development or obsolescence.

The second survey (25, 26, 27) complemented the first one by classifying and briefly commenting on the various approaches to the solution of the different phases in flowsheeting programs (such as the calculation ordering, convergence promotion, general approaches to simulation etc.,) rather than on the programs directly. Together these two surveys cover all aspects of flowsheeting and constitute an excellent though brief introduction and guide to the whole subject.

The kind of research accomplished in Universities is normally more fundamental or theoretical in orientation than industrial research and large industrial programs are usually out of the range of interest of researchers. New academic achievements should be expected more in the sphere of new application such as dynamic-state simulation (49, 50) and batch-mode operation plant simulation (51) where, in both cases, academic research still has the lead, or in the sphere of better and more generalised property generation methods or mathematical convergence controllers for large simulators.

## 1.6 Discussion.

The general trend in flowsheeting programs observable in this survey is clearly related to the evolution of the computers on which they were developed. There too the same three phases of evolutions are discernible. The first phase in flowsheeting program development coincided with that period in time when computers were merely fast calculating machines but small in size and much slower than present ones. During this time, the computers could only accommodate models of unit operations one at a time. The models were quite advanced in concept but took a relatively long time to solve. The very high cost of computers and their relatively narrow spread was also a major factor in the limited interest in flowsheeting.

With the advent of faster and larger computers the idea of connecting the units became more practical during the second phase in flowsheeting development. Executive programs were prepared to connect the models, call them in a specified order and keep an account of the plant flows. The next step was the inclusion of data and model banks. This occurred when the basic structure of the executive was improved and generalised and the use of the programs became more accessible thus raising the demand and the standards of the models and the data. At the same time recycle loop solvers and unit ordering routines were introduced to ease and reduce the data preparation work imposed on the user. By 1965 the standard format of the simulators

was set (and best illustrated in PACER).

The appearance of the very large and extremely fast American computers lead to the appearance of two different attitudes towards flowsheeting. The first one was a continuation of the established trend that is increasing the accuracy of the final results by data and model improvement; the other, representative of the third phase in computer-aided design, being the incorporation of new features and facilities in the programs such as mechanical designs, cost evaluations and optimisation.

This strong dependence of flowsheeting programs on computers is easily demonstrated by the fact that in Britain where British computers are lagging in speed and size many years behind their American counterparts, the flowsheeting scene resembles the American one of the second period in flowsheeting.

Much work has gone into automating the simulation and partial design of chemical plants to reduce the repetitive calculation work for the engineer, but more fundamental research is necessary to free him completely from choosing the most profitable plant synthesis and design among the many alternatives he may be faced with. The kind of program which will certainly emerge during the third phase is still in its infancy and little can be recorded about it at present.

## 1.7 Conclusion

Throughout this survey the author has attempted to present a short history of computer-aided steady-state chemical plant simulation and design by modular-type programs. The success of this presentation has been hindered by two important factors.

1. The almost complete secrecy surrounding most good simulation and design packages, particularly those well developed though continuous application and research namely the industrial packages.
2. The lack of continuity in the various academic attempts at presenting a comparably good and competitive package and the repetition of the same type of work over the last ten years both in Industry and at University, because of the lack of communication in this field.

In a survey of Chemical Engineering programs available for sale, lease or licensing (9), nearly 200 programs were listed. Admittedly only a few of them were flowsheeting packages, but more than 100 programs dealt with some aspect of simulation, design or plant optimisation. This partly proves the point that industrial secrecy is the primary factor in the relatively retarded state of the art when accounting for all the lost effort put through duplicated work.

This is also one of the main reasons that encouraged the

author and his superior in presenting here a new non-proprietary package which would be made publicly available to all interested users in the hope that using this relatively large program, further non-proprietary research and development would help raise the general standards of flowsheeting.

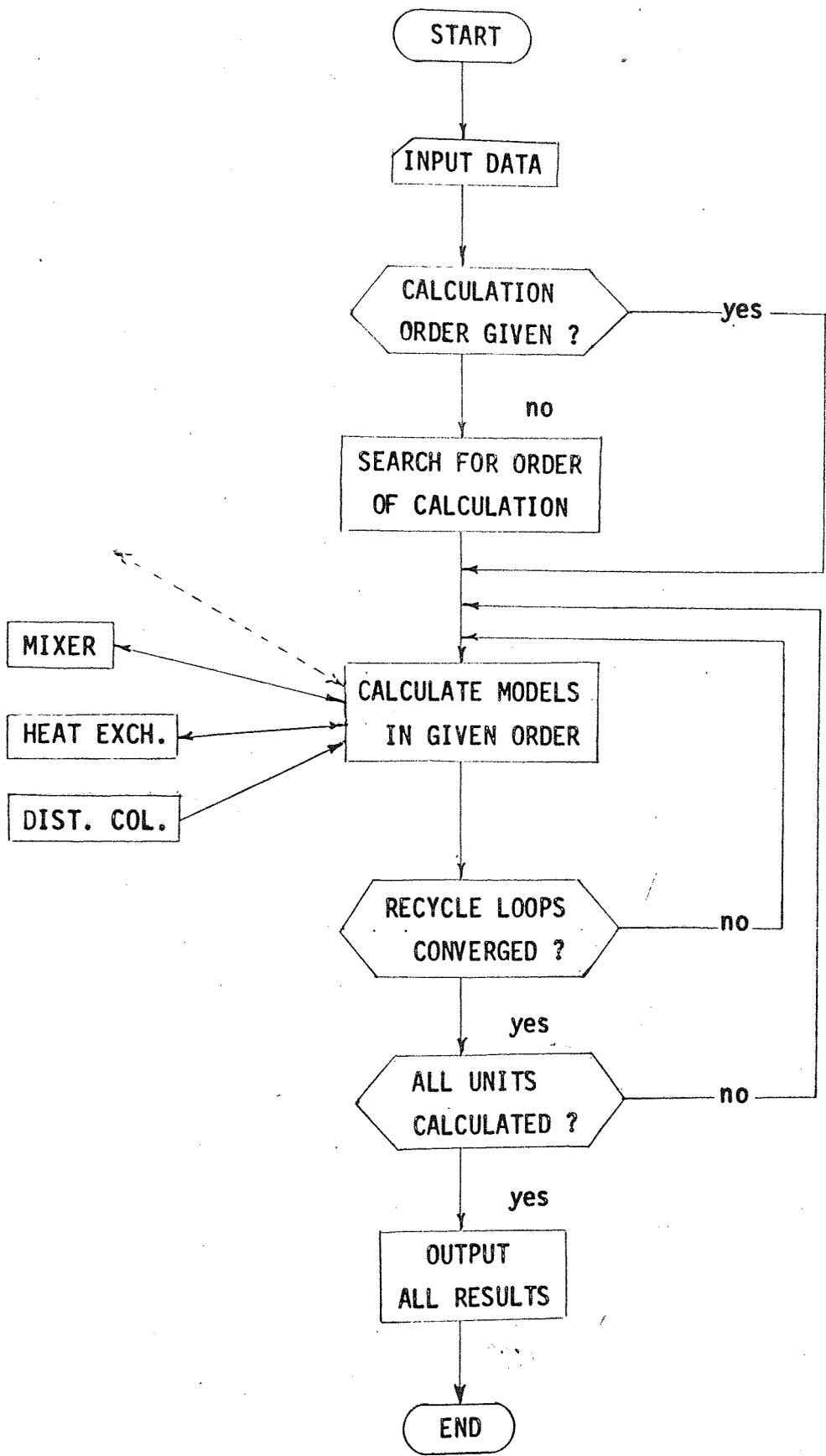


Figure 1 - 1 : Usual Flowsheeting Program Organisation.

## CHAPTER 2

### A DETAILED COMPARISON OF PACER, GEMCS AND CONCEPT

A flowsheet of a chemical plant consists of processing items or units joined by lines indicating transfer of materials. These lines or streams may represent a physical flow or may be fictitious and represent the transfer of financial or other non-physical information. Interest is usually in the condition of these streams since they determine the performance of the plant. Flowsheeting programs reflect this approach and deal with two concepts, processing units and stream connections.

Most flowsheeting programs use a simulation approach to the problem of finding a steady-state heat and mass balance in a plant. Given a mathematical model of a processing unit and the relevant parameters describing it, the conditions and flow rates of its feed streams, its output conditions can then be calculated. Execution of the flowsheet calculation usually follows four phases.

1. The Input Phase reads and checks to a certain extent the data describing the plant flow diagram and its feed streams, and stores the information in a form usable by the executive.
2. The Calculation Order Finder analyses the interconnections of the processing units to locate the presence of recycle loops and the units they include, and specifies the order in which the units must be calculated to provide the

solution in a minimum time or with the minimum of additional user specified information.

3. The Calculation Phase solves the equations describing the units using the input data.
4. Throughout and at the end of the calculation phase the print out facilities print the streams conditions and any other information specified by the models to allow the user to follow the progress of the simulation.

The three programs chosen here for comparison are some of the best known or more easily available programs in Britain. PACER, the best known flowsheeting program, was developed initially by H.A. Mosler at Purdue University in 1964. It was then improved by P.T. Shannon at Dartmouth College, New Hampshire, U.S.A. and by a team of professors and P.T. Shannon at McMaster University, Canada, and from whom the McMaster 1966 version was acquired by Professor P.E. Barker and later implemented at Birmingham and Aston Universities. It should be mentioned that a new commercial version, PACER 245, now exists in the U.S.A. (52) but about which little is known. The 1966 version available at Aston is a batch process executive with few unit models and no physical data package. It occupies about 18000 words on an ICL 1904 computer. When the unit models were added its total size exceeded 32000 words but even better programming could not reduce the size below 28000 words, this is mainly due to the large number of matrices used for data storage.

GEMCS is one of the most compact executives and is presently enjoying a wide academic use. It was developed by Professor A.J. Johnson and associates in 1968 at McMaster University, Canada from whom it is available (63). It can be used either batchwise or interactively (54). Like PACER from which it was derived, but to which it bears little resemblance, it is an executive program but does not incorporate a calculation order finder or any general unit models and property routines. It occupies less than 5000 words of core and plants of fair size (10-15 units) with simple physical property generation routines have been run in overlay-mode on an IBM 1130 (8K) computer with disc backing.

CONCEPT was initially developed in the late 1960's by a number of postgraduate students under the supervision of Mr.H.P. Hutchison at Cambridge University (U.K.) and it is now being improved and offered for use to paying users by the "Computer-Aided Design Centre" (C.A.D.) in Cambridge. This complete package comprises general and accurate unit models and a large thermodynamic data bank and it may be accessed, interactively only, either by teletype terminal or cathode ray tube and light pen (34, 55). It has extra design and costing facilities in addition to its simulation capability but these are attached to the program only if expressly requested. It is designed to run in a number of interactive and overlaid segments occupying 12,000 to 20,000 words on an Atlas computer or 24,000 words on an ICL 1906A computer.

The four sections making up the programs will now be

described for the three programs, but because of the complexity and length of these programs, they will be discussed briefly and with an emphasis on their differences from the user's interest aspect.

## 2.1 The data input

PACER, GEMCS and CONCEPT require the same basic data but each one reads and rearranges them in its own way. The common sets of data to all three programs are:

1. The process flowsheet, i.e. the interconnection of units and streams.
2. The equipment parameters characterising each unit.
3. The feed streams descriptions.
4. The preferred iterate streams to obtain the calculation order.

Other information is also required to completely specify the problem and tailor the program to their running sizes, but these are of lesser importance and are specific to each program.

### 2.1.1 The PACER program:

PACER demands the most input data because of its less efficient programming. It is not interactive but many simulation cases may be run consecutively in one batch as long as all the models are provided but the complete set of data must be provided for each case. In PACER, each unit is numbered and its model name given in its process matrix. To recognise the model to be called, the data set must contain at the beginning the list of

models available in the package and in the order the CALL statements are arranged. By comparing these names with the models requested, a list of the CALL statements to be accessed, for each unit, is formed. The other data required are as follows.

The process flowsheet. This is represented by a matrix bearing in each row the unit number, its model name, plant identification tag and input then output stream numbers attached to it. Output streams are differentiated from input streams by a negative sign preceding their numbers. Streams and units do not have to be numbered in an uninterrupted sequence but omitted numbers represent blank arrays in the data storage matrices. This matrix shows clearly the plant connections and is used to produce the calculation order. It is illustrated in table 2.1.

The equipment parameters. These are stored in another matrix. Each row begins with the unit number (identical to the row number in the storage matrix) and contains all the parameters needed to characterise the unit, ordered according to the model's requirements. The maximum array length defines the matrix width and unused portions of other arrays must be read in as zeroes. Extra long arrays may be broken into two parts and the second part introduced in an "Additional Parameter Array", where the different arrays are accumulated internally head to tail.

The feed conditions. PACER requires the input of the description of all streams in the plant. Generally only plant feeds will be

Unit Number	Model Name	Plant Identifier	Associated Streams					
1	THRØTL	-	1	-2	0	0	0	0
2	DISCØL	-	2	17	7	-3	-8	-19
3	HEXCH	C-113	3	14	-4	-15	0	0
4	CØMFRS	J-44	4	-5	0	0	0	0
5	THRØTL	-	5	-6	0	0	0	0
6	SPLITR	-	6	-7	-9	0	0	0

Table 2 - 1 : PART OF PACER'S PROCESS MATRIX

Stream Number	Flag	Flow lb moles/hr	Temperature °R	Pressure psia	Enthalpy btu/lb mole	Ethylene fraction mole fractions	Ethane fraction mole fractions
1	1.	511.	440.	334.	2400.	.45	.55
4	0.	1325.	452.	215.	6400.	.95	.05
17	0.	1085.	430.	215.	2400.	.95	.05

Table 2 - 2 : FEED STREAM AND INITIAL GUESSES

known and their values set. These streams are flagged as such (flag = 1, where the flag is in position 2 of the stream array). The values of all unknown streams are entered as zeroes and their flag is zero. Stream values may be guessed to insure or accelerate convergences. The number of guessed streams is not limited and unnecessary guesses will be overwritten at execution.

The preferred streams and the calculation order. Before solving the heat and mass balances the program attempts to find an order in which to calculate the units. This long trial and error procedure (section 2.3.1) may be shortened by proposing a set of streams to cut in order to solve the recycle problem. In this case the user indicates these numbers explicitly in a specified array. He may choose up to three preferred streams. The tedious order finder may be by-passed completely if the user provides his own order. This is done by inputting the unit numbers in the required order, each number followed by a flag indicating whether it is part of a sequential set (flag = 1), a recycle set (flag = 2) or the last unit in the loop (flag = 3). In this case a preferred streams list is not required but recycle stream guesses are still recommended.

Coded data input sheets were prepared by Professor P.E. Barker for the Birmingham University version of PACER to facilitate the complex data input. They also define clearly the additional information required to consolidate the run. The sheets are shown in Appendix 1.

### 2.1.2 The GEMCS program.

GEMCS's structure is completely different to PACER's even though it was derived from it. Its data requirements are much simpler because it lacks the ordering section. The sets of data mentioned above are still required but in a different way. Many simulation cases can be run consecutively even with different plant configurations as long as all the models are provided.

The flowsheet. This matrix used in PACER to find the calculation order and define the streams attached to each unit, is not required explicitly in GEMCS. The stream connections are specified in the unit parameter arrays. There, the number of inputs is defined followed by their stream numbers, then the number of output streams followed by their numbers. Inputs and outputs are defined by their specific locations in the array rather than by signs. This reduction in storage requirement reduces also the clarity of presentation.

The equipment parameters. The list of parameters for each unit (illustrated in table 2.2), which includes the stream connections, are stored in a long cumulative array, head to tail. A special routine stacks them up in the new array, records their starting positions and lengths in a two-column matrix and retrieves them when needed in the calculation phase. Missing unit numbers do not cause gaps in the cumulative array, and unit parameter arrays may be of various lengths. To extract an array from the cumulative one, the same routine checks the monitoring matrix

1. (unit no.)	1. (type)	16. (array length)	0.	0.
1. (no. of inputs)	1. (stream no.)	0.	0.	0.
1. (no. of outputs)	2. (stream no.)	0.	0.	0.
215. (outlet pressure)				
2. (unit no.)	2. (type)	26. (array length)	0.	0.
3. (no. of inputs)	2. ( stream	7. numbers	17. )	0.
3. (no. of outputs)	3. ( stream	8. numbers	19. )	0.
40. (no. of plates)	20. (feed plate)	215. (colm.pressure)	235. (tops fl.r)	235. (vap.fl.r. on plate 1)
235. (vap.fl.r. on last plate)	.95 ( tops composition )	.05	1. ( key components	2. )
4.60 (reflux ratio)				

Table 2 - 3 : Some of GEMCS's Equipment Arrays.

and copies the array into another vector accessible to the models. Considerable memory space is saved but at the expense of increased run times.

The Calculation Order. An auxiliary program "ORDER", based on the PACER method (section 2.2.1) may be used to obtain only a set of iterate streams which must then be followed by eye to obtain the order. The user-generated calculation order is input as an array. Recycle loops are not shown explicitly but each set of nested loops is terminated by a special controller unit which is introduced in the order as an additional unit. Its function is to check for convergence and return the calculation to the first unit in the loop until convergence is attained.

The feed streams. GEMCS accepts the description of any number of stream descriptions and builds up the stream matrix as it proceeds with the plant calculations, thus only feed and guessed streams need be input. Unlike PACER, not all stream descriptors are kept in memory at all times. Only those flagged "permanent" by an array of stream flags remain permanently, the others are erased after use. This allows the matrix of constant size to accommodate a larger number of streams. The position of temporary streams is always changing as new ones are introduced in the first vacant array in the matrix soon after its calculation and until it is erased again. Another special routine is required here to locate, store and erase streams. For this reason each stream array begins with its stream number for recognition. This variable content matrix reduces the storage requirements for small computers

but at the expense of longer run times. The simple data requirements of GEMCS do not require special coded sheets.

### 2.1.3 The CONCEPT program.

CONCEPT is a complete package with thermodynamic data and unit models and is completely interactive, but because of its programming it requires less data than PACER. The data may be stored in advance on file or input at run time. The execution is performed in two steps: Finding the calculation order (not optional) then calculating the heat and mass balances. Data not used in the first step may be input before the second step. Consecutive runs with fairly similar plant conditions may be effected simply by changing some of the data already stored on file. The models are called internally according to their plant order by a small program created at run time by the data reader.

The flowchart. The program asks about the number of units and streams in the plan, then for the name of each unit and that of its input and output streams (alphanumeric names are allowed). The streams and units are remembered internally for easier manipulation but are always referred to by their original names when the program communicates with the user. The data is stored internally in matrix form as in PACER. Since the data input is entirely interactive, a mistake in the flowsheet may be easily corrected by retyping the unit name and modifying its streams connections.

The equipment parameters. By commanding INPUT PARAMETERS, the

program asks for the unit name, the number of parameters needed then for the list of parameters. This command must be repeated many times, once for each unit, however by typing ALL after INPUT, the program repeats the input command for all units. The parameters are stored internally in a cumulative array and model routines have access to them through a FUNCTION routine by calling PARAMETER (i), where i refers to relevant parameter in the unit's parameter array. The parameter may be prepared and stored in advance in a file and the file name given to the program thus saving time during the interactive data input. In both cases, the program will inform the user of missing unit parameter arrays and will not proceed to the next data set until they are input.

The stream matrix. The user may input the description of any stream he wishes by commanding INPUT STREAMS, but CONCEPT will ask automatically for the feed and iterate streams descriptions. The program asks for the stream name, then prints the name of each descriptor or stream variable (flow, temperature, etc.) to which the user inputs the required data. The streams are stacked internally in a cumulative array monitored by a two-column matrix (as with the parameters in GEMCS). Streams and input are zeroed.

The calculation order. CONCEPT finds its own calculation order based on either its own preferred iterate streams or those specified by the user which it checks for sufficiency to cut all loops. Insufficient sets of streams are completed by the program and the new streams are presented to the user for acceptance or rejection in which case the user is asked to input a new set of

Figure 2 - 1 : Examples of data input into CONCEPT.

INTERACTIVE FLOWSHEET SYSTEM ENTERED - WAIT FOR '--> '

--> TV1

NO. OF I/PS = 1

NO. OF O/PS = 1

I/P NO. 1 : 1

O/P NO. 2 : 2

--> DC2

NO. OF I/PS = 3

NO. OF O/PS = 3

I/P NO. 1 : 2

I/P NO. 2 : 17

I/P NO. 3 : 7

O/P NO. 1 : 3

O/P NO. 2 : 19

O/P NO. 3 : 8

Part of CONCEPT's Interactive flowsheet input.

? INPUT STREAM 1

STREAM 1

FLOWRATE = 511

TEMPERATURE DEGF = -20

PRESSURE PSIA = 345

VAPOUR FRACTION = 0

ENTHALPY BTU/LBM = 0

WATER = 0

ETHYLENE = .45

ETHANE = .55

? INPUT ALL GUESSES

GUESSED STREAMS

LDB NO. = 7

STREAM 17

FLOWRATE = 1085  
TEMPERATURE DEGF = -40  
PRESSURE PSIA = 215  
VAPOUR FRACTION = 0  
ENTHALPY BTU/LBM = 0  
WATER = 0  
ETHYLENE = .95  
ETHANE = .05

Example of stream data input.

? INPUT ALL PARAMETERS

PARAMETERS FOR DC2

HOW MANY ? 8

- 1 . 40.
- 2 . 20.
- 3 . 235.
- 4 . 235.
- 5 . 4.6
- 6 . .95
- 7 . .05
- 8 . 1.

Example of parameter input.

? INPUT ALL S/RS

TYPE THE S/R NAME AFTER THE UNIT NAME

IN1 - SOURCE  
TV1 - TVALVE  
DC2 - DISCOL

Example of model routines allocation.

Figure 2 - 1 : Examples of data input into CONCEPT.  
(Cont'd)

Class One Commands : General organisation and data use.

FLWSHEET return to Flowsheet phase.  
ITFIND return to ITFINDER.  
CHECK check all the data.  
LOG record all the information for a unit or a stream.  
UNLOG reverse of LOG.  
DUMP create a record of all the data present in CONCEPT.  
UNDUMP replace the present data with that from an other file.  
OFFLINE create a file suitable for batch mode running.  
UNITS set the thermodynamic units.  
CALCULATE initiate a simulation calculation.  
EXIT leave the CONCEPT system.  
ECONOMIST transfer data to the ECONOMIST.

Class Two Commands : Data access and alteration.

Possible first part	Possible second part	Possible third part
COLLECT	PARAMETERS	Unit name(s)
INPUT	STREAMS	Stream name(s)
TYPE	GUESSES	Component code(s)
PRINT	ITSTMS	File title
CHANGE	S/RS	
EXAMINE	COMPONENTS	
DELETE	COMPSEQ	
	CONTROLS	
	THERMODATA	
	INPUTS	

The word ALL can be inserted before the second or third part.

Figure 2 - 2 : CONCEPT's data input and running Commands.

preferred streams, but, he is not allowed to input his own calculation order. Examples of data input are illustrated in figure 2-1.

Besides these four sets of data, CONCEPT requires the name of the components involved in the plant, since it has its own set of physical property generation routines. These are input following the command INPUT ALL COMPONENTS. CONCEPT then asks for the components names and extracts their property constants from its own files to transmit them to the executive.

A number of data inputting, checking and updating facilities are available to the user. They are all affected by simple commands illustrated in figure 2-2. Contrary to GEMCS and PACER, the order of data input in CONCEPT is not rigid but it is set by the commands input by the user, although a logical input sequence should be followed.

CONCEPT offers great flexibility and many data input facilities because of its interactiveness and their form is much more understandable than PACER's or GEMCS's. PACER's data set is unnecessarily long and tedious to prepare. GEMCS's data set is concise and simple because the program offers few facilities and services.

## 2.2 The calculation phase

After providing the models and inputting the data, the

executives take over the command for the calculation phase which consists of three parts.

1. The ordering of the units in a calculable sequence.
2. The calculation of the units' heat and mass balances.
3. The checking and acceleration of the convergence of these calculations.

The calculation order finder is the most significant difference between the three programs, but the calculation section is the most important part in all three.

#### 2.2.1 PACER:

PACER builds up initially from the process matrix a connection matrix showing the source and destination of each stream in the plant (feeds emanate from, and products sink into unit zero). This matrix shows also the stream numbers not used and locates plant feeds. Two arrays are also created to hold the stream and unit flags denoting calculated or unknown items. The program selects the first unit in the process matrix. If it is flagged unknown (flag = 0), its input streams are checked while if they are all known (flag  $\neq$  0), the program calculates the unit and flags it and its output streams as known (flag = 1), then the next unit is checked. The procedure is repeated until all units are checked. Units with one or more unknown inputs are not flagged. The scanning of the matrix is repeated until, during a complete scan, no more units are calculated. The remaining units, if any, form one or more recycle loops. In a

serial problem all units would have been calculated by this stage in a manner shown below, and the printout stage accessed. Otherwise a list is made up of all uncalculated streams and any preferred streams are moved to the top of the list. From the list, the first stream is picked and assumed known (flagged 3). The process matrix is scanned again as above in the hope of flagging the remaining streams. Every time a stream can be calculated as a result of this guess, it is flagged differently (flag = 3) and placed in a new list. The scan is repeated until no more units are flagged. If all units are not flagged, the newly flagged units and streams are deflagged and the next unknown stream on the list is chosen as known and the scanning repeated. All combinations of one then two then three streams may be tried. Problems requiring more than three iterate streams are abandoned. As the program finds the iterate streams it also builds up a list of the units involved in the recycle loops. Starting with each iterate stream, labelled known, the program sets the unit it feeds in a new array and examines its output streams. Other units are added to the list only if they are fed by these outputs and if they are in a recycle loop. Units fed by such streams but not partaking in loops are not added. The repetition of this process yields an ordered list of calculation and the remaining units are then calculated. Next comes the calculation phase of the recycle loops (initial sequential sets would have been calculated before entering the order finder).

Starting with the first unit in the list, or the first

unit in the list, or the first non-recycle unit that can be solved before finding the order (these are calculated one by one until the first recycle unit is met), its input streams are extracted from the stream matrix and set in the STRMI (streams in) matrix, the output streams are set in STRMO (the stream numbers are obtained from the process matrix). The unit model is called according to its plant sequence number and calculated. The input and output streams are then recopied back into the stream matrix. The other units are calculated similarly. The calculation order finder is accessed only when the first recycle unit is met. Convergence checks on the recycle streams are performed by a separate routine which compares the relative change in each descriptor of each stream in the loops over every two successive iterations against a preset tolerance and sets a flag to 1 if any variable has not converged. Their convergence is never accelerated but all stream descriptors must converge for the calculation to end. At the end of a successful or unsuccessful run, the final printout stage is accessed and the next case started.

A simplified flowchart of PACER's organisation is shown in figure 2-3.

### 2.2.2 GEMCS.

The calculation order is input into GEMCS by the user, based on a visual analysis of the plant diagram. This reduces the program's size and running time at the expense of more work

for the user. Recycle loops are not represented explicitly, and units do not bear flags, instead a controller unit terminates each recycle set. This routine checks the convergence of one user-specified stream in the loop it controls and checks the relative change in all its elements against a present tolerance for convergence. In GEMCS a pointer points at successive positions in the calculation order and initiates the calculation of the unit whose number is given in the position pointed at. The controller subtracts from this pointer the number of units involved in the loop thus set using it to point at the first recycle unit to repeat the sequence until convergence is achieved. Then the next sequence of units is tackled. This calculation method eliminates the need for additional storage for lists or other routines since many commands and checks are grouped in one routine, and the controller is introduced only when required. This benefit is partly offset by the additional work imposed on the user in deciding on the optimum order, preparing the controller's parameter list and choosing a sensitive stream to check. Since only one stream is controlled, convergence might be wrongly attained if the stream controlled is not very sensitive to changes elsewhere in the loop.

The procedure for solving the units is similar to PACER's except that two special routines are used to extract the streams and the parameters from their respective arrays, since the parameters are stacked cumulatively in one array and they are of variable lengths and because the streams positions in the

matrix are variable. After copying the input data to their appropriate locations, the unit model is called according to its model type (a parameter) and calculated. The output streams only and the parameters are recopied into their cumulative arrays by the same two routines then the temporary input streams are erased.

The unit calculations are repeated until convergence is achieved or the maximum number of loop calculations is exceeded. Then the whole unit sequence calculation is repeated once more to print the final stream values before the simulation ends.

A simplified flowchart of the GEMCS executive is shown in figure 2-4.

### 2.2.3 CONCEPT.

Little is known about the calculation order finder in the new version of CONCEPT, which has been changed since the work published by Forder and Hutchison (33, 56), and a sketchy description only can be given here because of the unavailability of the proprietary program listings.

The order finder operates in three parts and is part of the data input section because of the option offered to the user of refusing its iterate stream choice. In the first part (the LOOPFINDER) all possible "elementary" loops are detected by following each input to each unit backwards through all the units its input streams come from, until either a plant feed is met thus

producing a sequential set or until the starting stream is met again thus locating a closed loop. All inputs to all units must be traced to locate all elementary loops (an elementary loop sequence contains no unit more than once). In the second step (the ITFINDER) the minimum number of iterate streams is found. When all loops are located, a table is prepared where rows represent the loops and the columns represent the streams. Positions representing streams in loops are set to 1 and the other zeroed. An extra column shows the number of streams in each loop and an extra row shows the number of loops each stream is involved in. The program then presents the user with the description of the loops and asks for any preferred streams. These are flagged differently and the remaining streams analysed to find any additional streams needed to fully cut the system open. There is no limit on the number of iterate streams. To obtain these streams, the program analyses each stream starting with those contained in the least number of loops and works upwards. A stream present in a loop, which is still represented by other non-eliminated streams, is erased. Once all such streams are eliminated, the remaining streams represent the additional iterate streams required. They are presented to the user for acceptance or rejection, in which case he is requested to input new iterate streams and then the procedure is repeated. The third step consists in obtaining the calculation order by tracing forward each iterate stream through the units, taking care not to add a unit to the order before all its inputs are

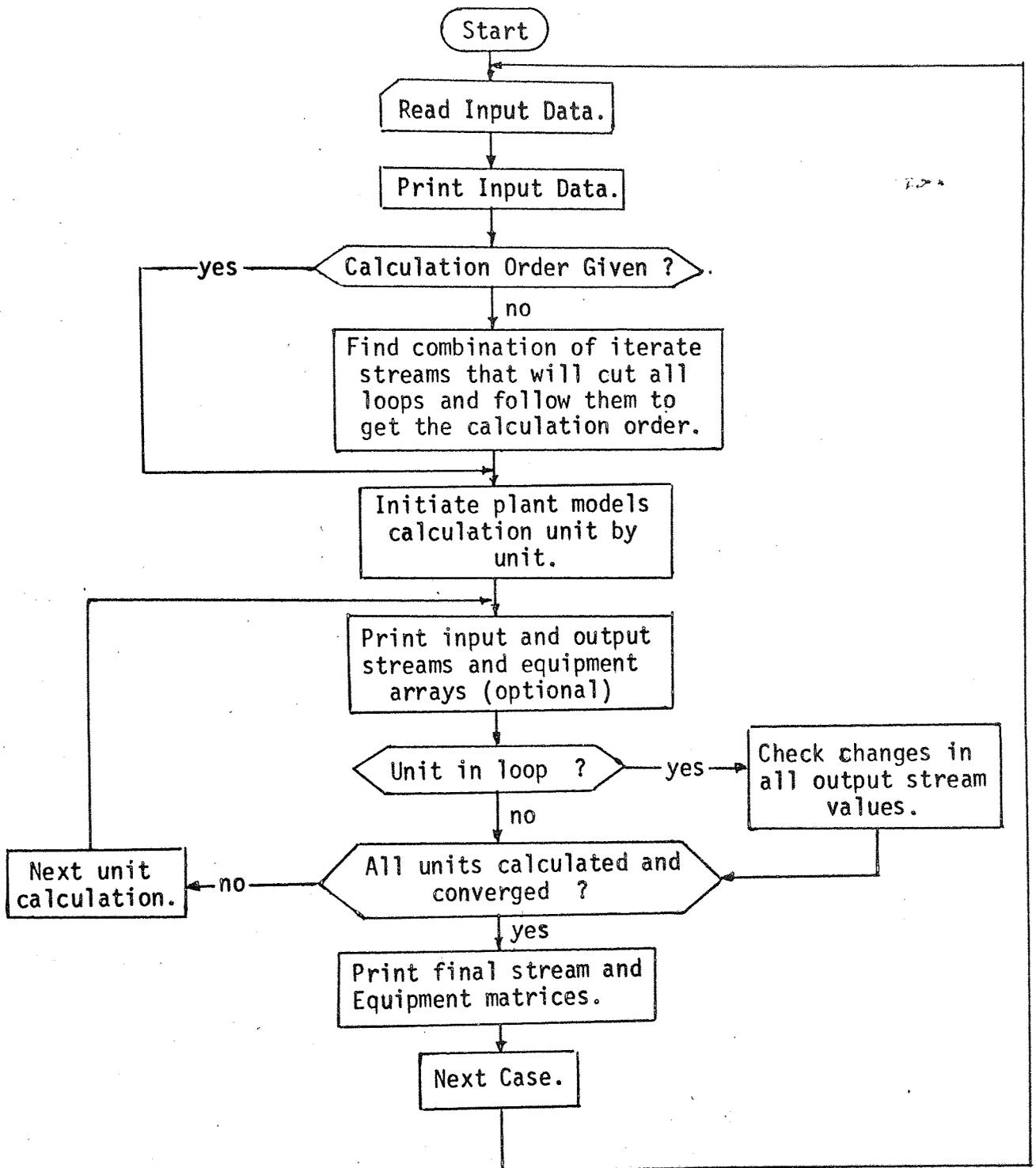


Figure 2 - 3 : The simplified PACER Flowchart.

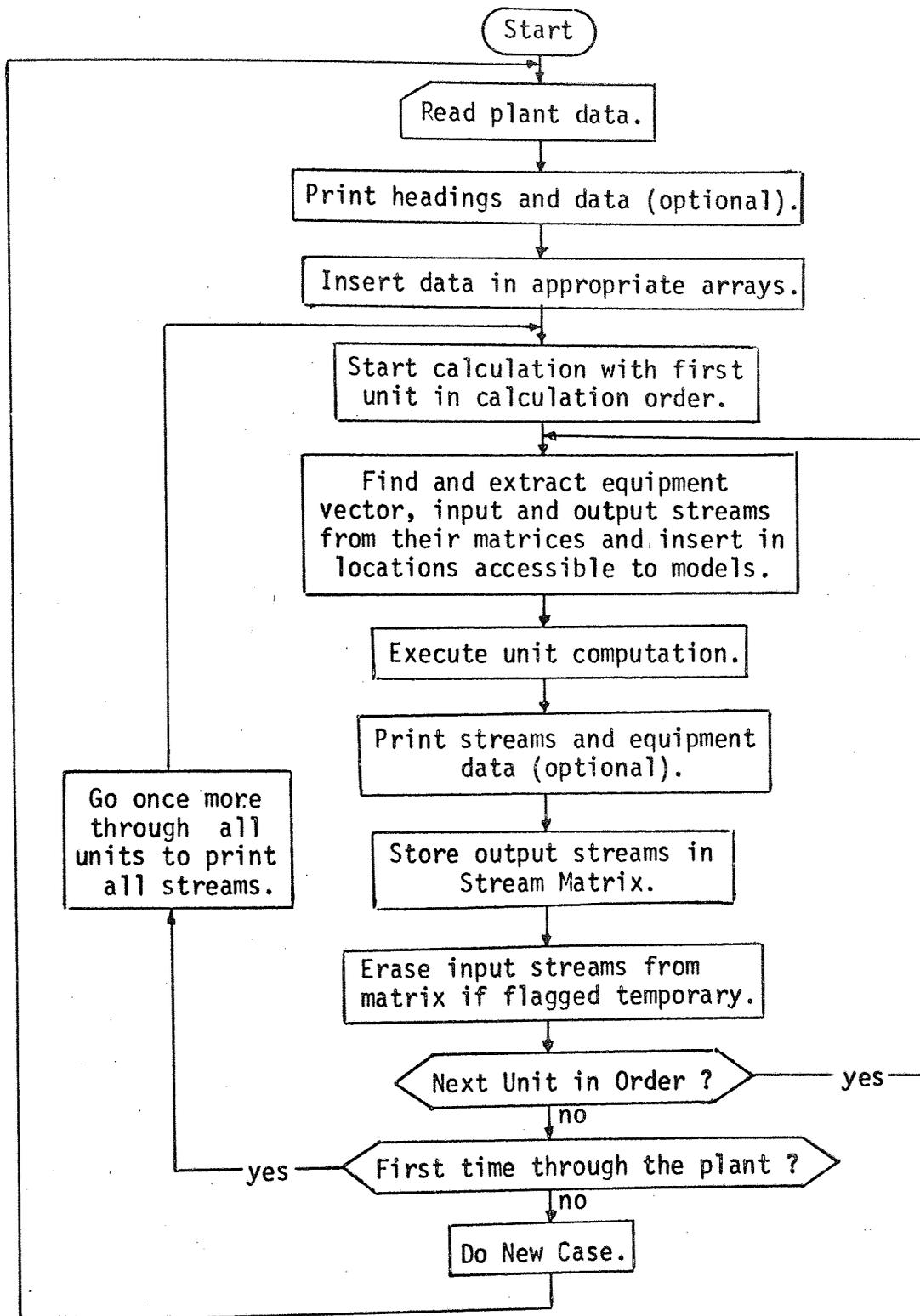
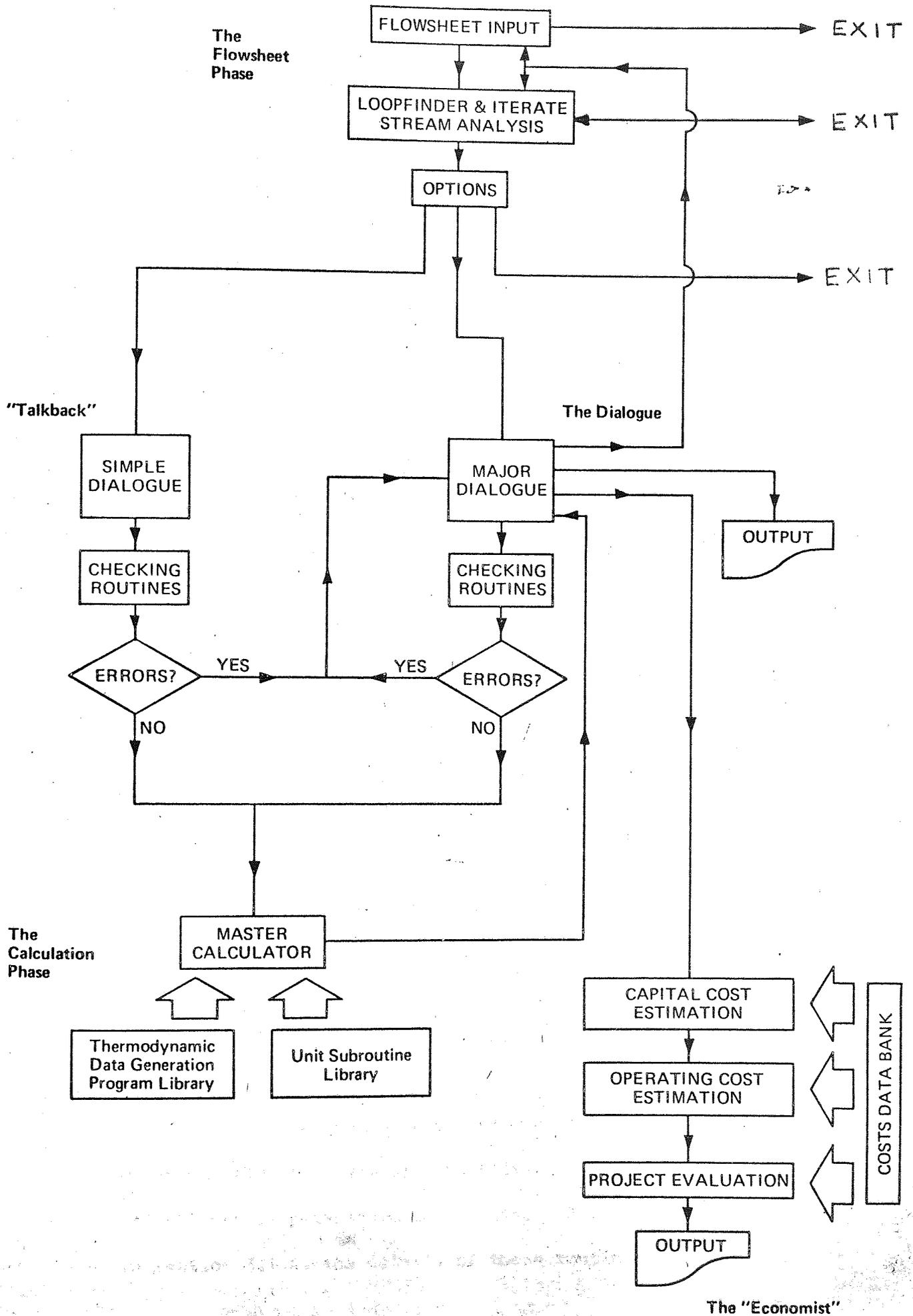


Figure 2 - 4 : The GEMCS Flowchart.

Figure 2 - 5 : The Concept Mark 3 System



known, guessed or are outputs to units already in the order.

In the calculation of the units, CONCEPT copies the input streams attached to each unit calculated from the cumulative stream matrix to the input stream array accessible to the models. There the streams are stacked head to tail in one cumulative array. The model is called by the specially prepared routine and calculated. Upon return, the output streams only are copied back to the stream matrix. Following the calculation of all the units in a set of nested recycle loop, the loop's iterate streams descriptors are improved by a Wegstein-type (57) accelerator. Since the parameters cannot be updated internally, they are not recopied to their cumulative array after each unit calculation. At convergence all the units in the plant are called a final time to printout any special messages, and then the printout stage is accessed.

The CONCEPT Mark 3 system is illustrated in figure 2-5.

### 2.3 Unit models and thermodynamic data.

The main practical difference between CONCEPT on the one hand and PACER and GEMCS on the other is the availability in CONCEPT of an extensive thermodynamic data bank covering the properties of the most widely used hydrocarbons, light gases and water and also of a set of unit models describing the common units found in petro-chemical plants. For reasons explained in section 3.1.4, the details of these routines cannot be

divulged, but they are thought to be similar to those described in chapters 8 and 9.

A complete PACER package is available, but only to paying users in the U.S.A. (52), but the versions of PACER and GEMCS offered to Universities contain only the essential executive routines and very few elementary models, as these programs are merely used as learning tools for model writing.

The difficulty in writing models depends on the degree of accuracy or generality required, but the requirements for all three programs are the same. The two main areas of difference are as follows.

1. The variation in the data transmitted through COMMON statements and the relative position of the variables in the transmitted arrays.
2. The type of thermodynamic and physical property routines available.

#### 2.3.1 The transmission of variables.

In all three programs, argument lists in routine names are not used. All variables are transferred through sets of unlabelled COMMON statements, however the number of variables transmitted in each program is different and so are the contents, locations and dimensions of the arrays. PACER and GEMCS use the same variables because of their common origin however the parameter arrays in GEMCS use only one subscript referring to

the variable location whereas in PACER a second subscript denotes the array row in the matrix. In CONCEPT, the names of the arrays are different and the parameter array is only accessible as a FUNCTION statement by specifying PARAMETER (i) on the right-hand side of an expression, thus parameters cannot be updated internally. The stream vectors are doubly-subscripted arrays in PACER and GEMCS, the first subscript referring to the input or out stream order and the second to the variable in the array. In CONCEPT, the inputs and the outputs are stacked head to tail in two cumulative arrays and one subscript only is used, thus requiring a more careful setting of the subscript value.

Another compatibility problem is the relative position of the variables in the stream and parameter arrays. The stream arrays in PACER and GEMCS start with the stream number, then its flag (a redundant variable), followed by the variables in the order required by the user's models. In CONCEPT, the stream name is omitted and the flow rate occupies the first position followed by the other variables in a strict order (flow, temperature, pressure, vapour fraction, enthalpy, composition). The equipment parameters, in PACER, start in position 2 following the unit number. In GEMCS they start in position 16 after the unit number, its type number, array length and the input and output streams numbers. In CONCEPT, they start in position 1.

### 2.3.2 The thermo-physical property routines

PACER and GEMCS are not equipped with such routines.

The user must prepare his own methods and routines and specify his arrays completely in the light of his thermo-physical package.

CONCEPT offers routines to calculate the dew point, bubble point, enthalpy, temperature or phase composition of otherwise defined streams. It will even calculate the enthalpy of guessed streams and plant feeds without special request from the user. The thermo-physical package will accept data in either British or S.I. units systems.

Model writing for CONCEPT is fairly simple since the available routines enable the user to delegate to them most of the property calculations. Thus his models are relatively unit-system independent whereas in PACER and GEMCS, unless a generalised property package is introduced, each model may require its own property routines.

Even though the incompatibility in data transfer may be overcome by heading each model routine with a variable-redefinition section to be changed for each program, the presence or absence of property packages governs to a larger extent the exchange and compatibility of models.

#### 2.4 Initial, intermediate and final printout.

In most programs there are three phases of printout during a simulation.

1. A data-echo phase which prints out the input data with

appropriate identifying comments. This phase may also include a data check.

2. An intermediate unit-by-unit printout of streams and parameters attached to each unit simulated.
3. A final printout stage listing the streams final condition and any relevant message or energy requirements from the various units.

These phases exist in all three programs in some way or other and may be set or suppressed by setting appropriate switches or ordering certain listings.

#### 2.4.1 The data echo.

Data-echo is a built-in feature in PACER and GEMCS. In the available version of PACER it cannot be suppressed, and it produces a double-echo. In the first one the data are echoed immediately as they are read, this helps check the limits on the array sizes and provides the user with a direct card image. The second echo presents the process matrix, the stream connection matrix, some dimensions set in the data and the stream and parameter matrices. It also checks that the matrices maximum sizes as requested in the data do not exceed the DIMENSION sizes.

In GEMCS, among the data read in are four switches to control the nature, amount and frequency of printout. The first three switches control directly the data echo. The first switch causes or suppresses the printing of the number of components, the number of units, the calculation order and the stream flags.

The second switch controls the printing of the initial streams and the third one controls the printing of the equipment array. No data check is effected.

In CONCEPT there is no real need for a data echo as the program checks the data on input for consistency, and the user retains a printed copy of the checked data input. However at any time before or after the simulation calculations, he may obtain a printout of any necessary data by using a TYPE command.

In the case of a faulty run, the echo helps trace the error back to one of many reasons, an error in the input data, a poor initial guess for an iterate stream, or a wrong unit parameter, unless the error is in the models. PACER's echo is longer than GEMCS's because of its longer data input. CONCEPT's listing is long, because each piece of data is asked for separately.

#### 2.4.2 The intermediate printout.

The intermediate printout is of two kinds: Executive-generated and model-generated. In the first case the printout is usually controlled by print switches. In the second it depends on the models, this kind will not be discussed here as it depends on the user writing his own models.

PACER has two switches which can take on many values. The first switch (KSETS) controls the nature of the printout, and the second one (KPRINT) its frequency. KSETS may assume one of four values.

- 0 : Initial conditions, intermediate printout of the stream matrix every KPRINT loops, and final printout.
- 1 : To control printouts in the unit models and for a trace of the unit calculations (for the user's use).
- 2 : For additional printing of the streams associated with each unit calculated.
- 3 : Gives also the unit flag list when searching for a calculation order and gives a detailed trace of the search for the order.

In GEMCS when the fourth print switch is set to 1 the whole stream matrix, the parameter array for the unit being calculated and its input streams are printed for each unit calculated. Each set of arrays is preceded by a general title but the user must know the representation of each location in each array (the same applies to PACER). The output streams are not printed out but may be obtained from the stream matrix the next time it is printed, however they may be printed explicitly if the parameter "largest stream number" is given a negative sign. Another source of printout is the loop convergence controller which prints out at every specified number of loops (in the parameter array) the controlled stream values and their fractional change over the last two iterations. The printout is generally adequate if somewhat lengthy for large or slow converging problems.

In CONCEPT, the input streams to each unit calculated

are printed in a log file in the computer, where all the printing is done. No printing is sent to the teletype until the problem is solved or fails, then the user may ask for either a teletype or a line-printer printout of this file. The trace is very clear since besides printing the name of the routine called to compute each unit, it also prints a complete text description of each stream variable.

### 2.4.3 The final printout.

The final printout is different in each program. PACER merely prints out the final contents of the stream matrix and the equipment parameter matrix, but as is the case with the intermediate printout, it does not identify the various variables by any special comment. In GEMCS, the program repeats once only the calculation of each unit in the plant and prints their final output stream values as they are calculated. This time-consuming final run cannot be suppressed except by altering the program. CONCEPT follows GEMCS's method of re-entering each unit a final time to print any extra results or messages to be output at convergence, but a special switch which changes value before this final run is available and may be tested within the models and used to by-pass their calculation section. At this stage, a new file, RESULTS, is created to contain the detailed description of all the streams, and it may be listed at the user's command. The program also offers the facility of examining the values of certain chosen streams interactively by commanding TYPE STREAM.

## 2.5 Conclusion

This short description compares the main features of the four similar phases in PACER, GEMCS and CONCEPT. A critical appraisal and assessment as to the type of user they suit most will be given in chapter 4 following a practical demonstration in chapter 3 of their features and differences.

## CHAPTER 3

### APPLICATIONS OF THE SIMULATION PROGRAMS

PACER, GEMCS and to a certain extent CONCEPT have all been used to solve various types of problems such as plant simulation, optimisation of plant productivity, dynamic programming and plant design.

Two of the more relevant problems to this comparative study are now presented to illustrate these programs' applications and demonstrate their main differences.

The first example deals with the simulation of the flows of heat and mass and the energy requirements of an ethane-ethylene separation plant. The objective there was to obtain results as close as possible to the plant survey data.

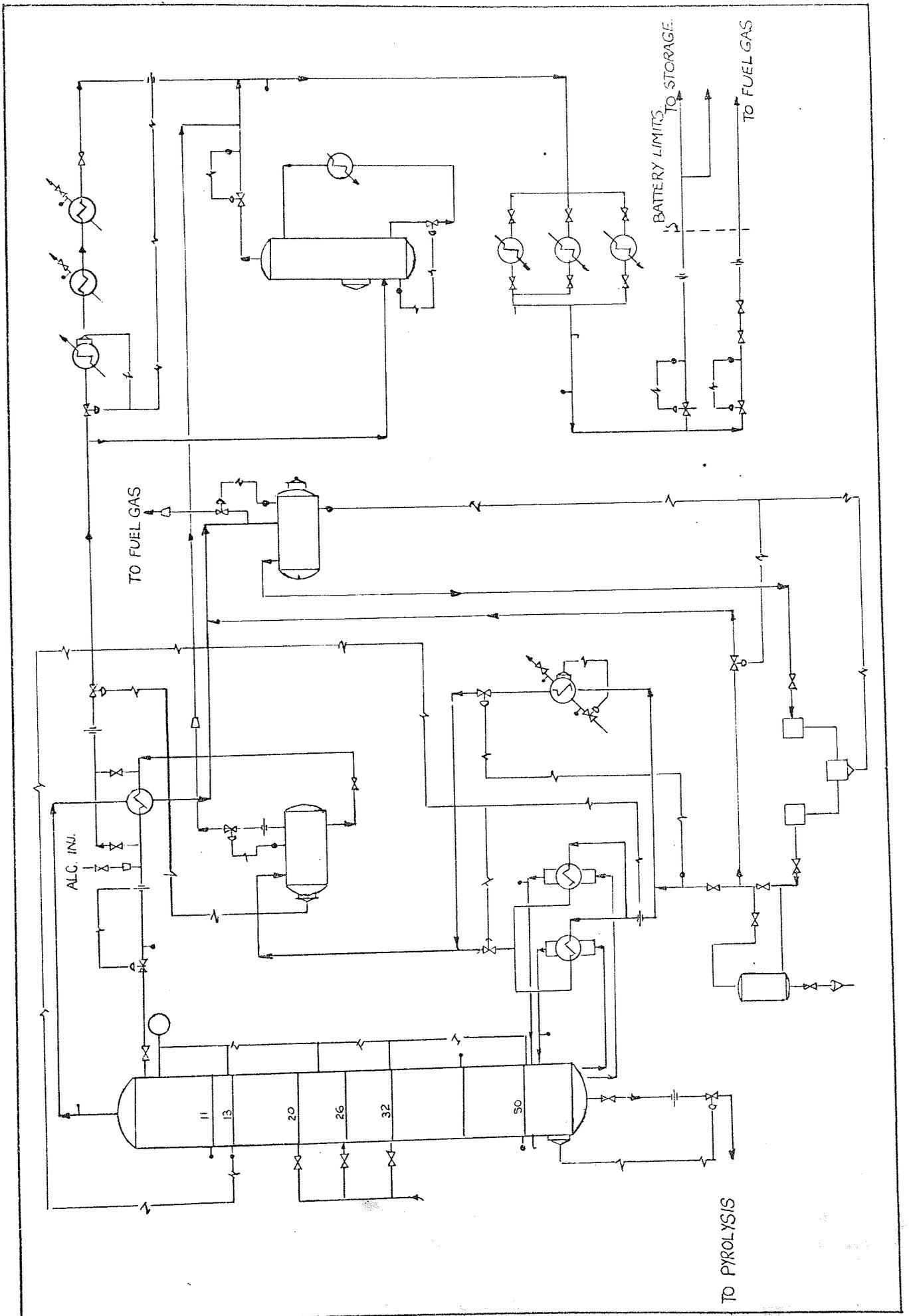
The second example is the implementation of a general optimisation routine adaptable to any plant simulation. The objective here is to describe the way of introducing the optimiser in a simulation without changing the executive or the unit models.

An example of plant design has been solved in a joint project with an M.Sc. student and is described elsewhere (58).

#### 3.1 Example One:

The first example describes the simulation of mass and heat balances and the flows and compositions in an ethane-ethylene

Figure 3 - 1 : The Ethane - Ethylene Splitter Plant



separation plant. It provides also estimates of the energy requirements of some of the units involved. This plant, illustrated in figure 3-1, is part of a large fractionation complex that was in operation at Polymer Corporation, Sarnia, Canada in the mid 1960's. Actual plant operation data were made available by Prof. A.I. Johnson formerly of McMaster University, Canada.

The main features in this plant that make it an interesting simulation exercise are:

1. Consistent data from a real plant survey are available for most units.
2. The plant contains many of the common hydrocarbon processing units.
3. It contains a number of nested recycle loops.

Certain important steps had to be taken before results could be obtained.

1. Process Modifications.
2. Modelling considerations in PACER and GEMCS.
3. Modelling considerations in CONCEPT.
4. Process Data study.
5. Writing the Models.

### 3.1.1 The plant description

A liquid mixture of ethylene ( 44 mole %), ethane (54%) and propane (2%) at 345 p.s.i.a is throttled down to 215 p.s.i.a. and is fed to a distillation column of 55 actual trays. The

top vapour product containing 95% ethylene and 5% ethane is superheated in a heat exchanger by the effluent of the heating-side of the column reboiler, then compressed to 585 p.s.i.a. in a reciprocating pump, thus heating it up further to about 115°F. This stream is partly used to reboil the column bottoms and the other part of the compressed vapour is sub-cooled and added back to the reboiler effluent to control the stability of the plant. After heating the top vapour the stream is divided into a recycle stream which returns to the top of the column and a product stream. The column bottom product contains most of the ethane and all the propane and is not treated.

All thermal and physical property data in this example are in British units since these were the survey data units and because property data in the literature were easier to find in the unit system.

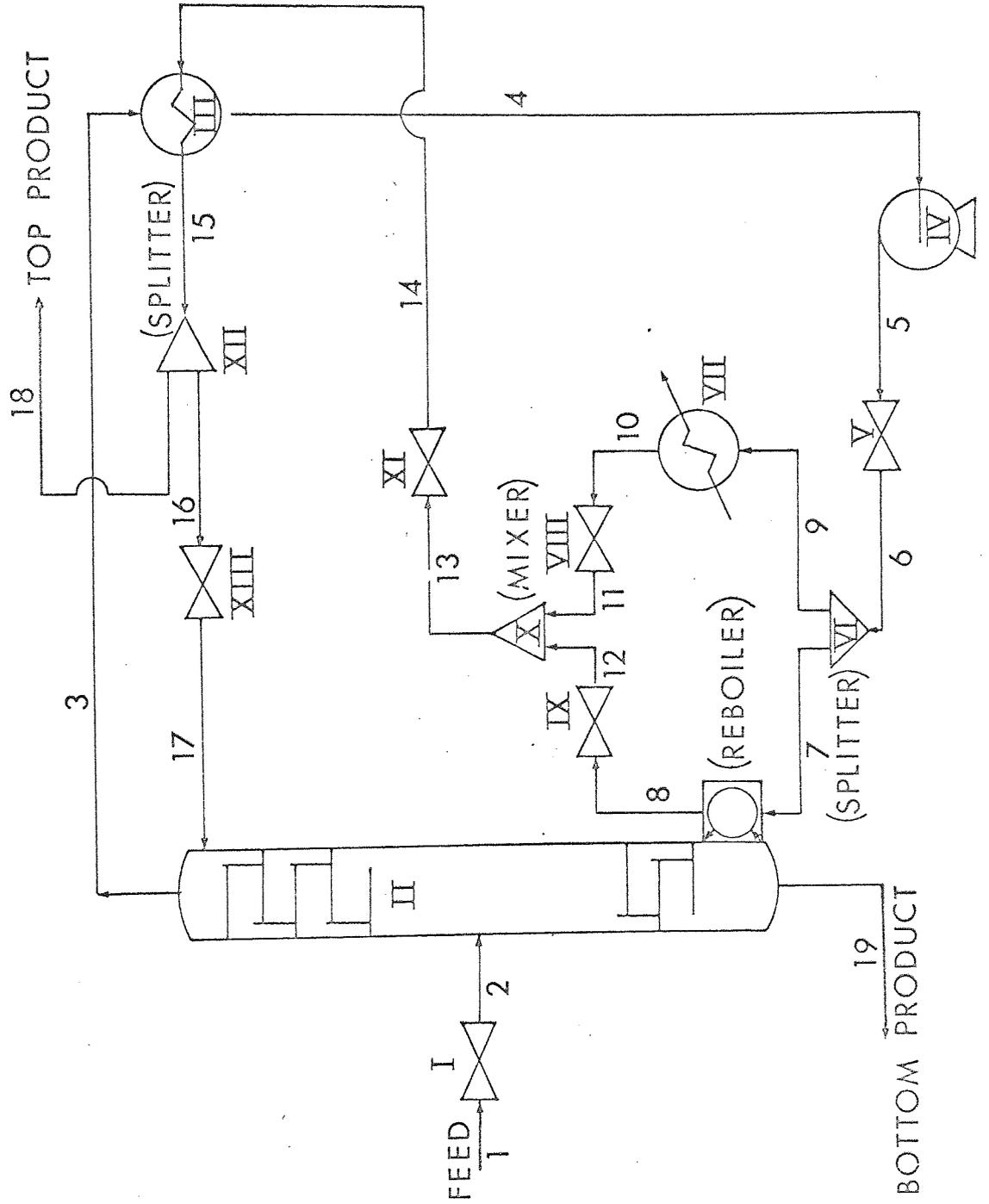
### 3.1.2 Process modifications

The plant diagram shown in figure 3-1 had to be modified to simplify its simulation as some of its units had little or no effect on the final results and thus were irrelevant to the simulation. Exact models of some of the units could not be written in a short time and the units had to be modified. The important modifications introduced were:

1. The exact temperature and vapour fraction of the feed were not given, it was assumed to be liquid at 345 p.s.i.a and its temperature set at 440°R.

Figure 3 - 2 :

### Flow diagram of the splitter plant



2. The propane content of the feed (44 mole % ethylene, 54% ethane, 2% propane) was ignored for simplicity. The ethylene and ethane compositions were rounded up to 45% and 55% respectively.
3. The number of trays in the distillation column was reduced from 55 to 40 to give results comparable to the plant data and the trays were considered ideal.
4. The pressure drop in the heat exchangers, the pipes and the control valves could not be modelled easily nor their effect on the heat transfer easily implemented. They were then modelled by a number of throttle valves introduced at convenient spots.
5. The intermediate settling tanks, where no important changes took place, were eliminated.
6. The column reboiler model was designed as part of the column model to avoid mixing heat-carrying streams (reboiler heat load to model) and matter-carrying streams.
7. The sets of exchangers on the right hand side of figure 3-1 were ignored to shorten the simulation runs.

The modified flow diagram is shown in figure 3-2.

### 3.1.3 Modelling considerations in PACER and GEMCS

PACER and GEMCS were both provided with a few simple models of no use in this simulation, but since both programs follow similar flowsheeting concepts, new models and physical property data prepared for one were easily adapted to the other. The lack

of experience in flowsheeting and model writing proved a major obstacle to fast and efficient model writing and to the development of concise thermo-physical data evaluation routines. The models written for this exercise were a long and less efficient version of those presented later (chapters 8 and 9) but were based on the same theoretical considerations. Details of these models and data evaluations routines are not included here since improved versions of the same routines are proposed in chapters 8 and 9.

A number of slight variations in the models were introduced for simplicity:

The fairly constant temperature difference between the first output (stream 4) from the heat exchanger (unit 3) and the second output (stream 14) suggested using this difference as a model parameter rather than setting an absolute outlet temperature as parameter.

The polytropic compressor (unit 4) was modelled according to the supplier design equations (59) relating absolute temperature and absolute pressure ratios:

$$\frac{T_{out}^*}{T_{in}} = \left[ \frac{P_{out}}{P_{in}} \right]^{(k-1)/k} \quad (3.1-1)$$

where  $k = C_p/C_v$  (the specific heats ratio) (3.1-2)

An isentropic efficiency accounts for the non-ideality of the compressor

$$\eta = \frac{T_{out}^* - T_{in}}{T_{out} - T_{in}} \quad (3.1-3)$$

where  $T^*_{out}$  = Ideal outlet temperature  
 $T_{out}$  = Real outlet temperature.

The value of 1.20 for k given by the manufacturers had to be changed to 1.23 for better results, and the efficiency factor, not provided was set at 75%. The mechanical efficiency of the pump was given as 94% and used to obtain the compressor's energy requirements calculated as:

$$\text{Energy} = \frac{(\text{Enthalpy out} - \text{Enthalpy in}) * \text{Flow Rate}}{\text{Mechanical Efficiency}} \quad (3.1-4)$$

The other units were modelled by the following models described in chapter 8:

<u>Unit</u>	<u>Unit Number</u>	<u>Model Name</u>
Throttle valve	1	VALVE
Distillation Column	2	DISCOL
Heat Exchanger	3	HEATEXCH
Compressor	4	COMPRESR
Pressure drop	5	VALVE
Stream divider	6	SPLITTER
Cooler	7	HEATER
Control valves	8-9	VALVE
Mixer	10	MIXER
Pressure drop	11	VALVE
Stream divider	12	SPLITTER
Control valve	13	VALVE

The thermo-physical data evaluation package comprised three more sections.

1. The vapour liquid equilibrium constants data (K-values) were generated by the general Chao-Seader method (60) based on the Reddlich-Kwong equation of state (61) for the vapour fugacities and the Hildebrand solubility parameters for the liquid phase fugacities.
2. The dew-point and bubble-point routine used a Newton-Raphson search technique to generate the temperatures at which the liquid compositions in equilibrium with the given vapour compositions would sum up to unity (dew point) or the vapour compositions in equilibrium with the given liquid compositions would sum up to unit (bubble point).  
Details of the above two methods are described in chapter 9.
3. The enthalpy data generation routines differed from those described in chapter 9. Vapour phase enthalpies were calculated from the Reddlich-Kwong equation of state and required time consuming iterative calculations to obtain Pressure-Volume-Temperature values at every iteration. This equation was not suitable for the liquid phase, so the pure components saturated liquid enthalpies were curve-fitted from tables (62, 63) and used throughout the liquid phase. These routines proved laborious in computing time and were later replaced by the Yen and Alexander generalised enthalpy method (64) (section 9.2).
4. To shorten the computation time for the distillation column, the components' saturated liquid and vapour enthalpies (62, 63) were curve fitted as functions of temperature to an excellent

accuracy (about 99%) and used instead of the generalised routines. The K-values were also curve-fitted as functions of temperature (65) thus obviating the need for the iterative Chao-Seader method.

5. To obtain the temperature of a stream specified by its enthalpy, pressure and composition, a Newton-Raphson iteration technique was applied using the enthalpy functions as dependent variables of temperature.

The routines used here gave good results but at the expense of long calculation times.

#### 3.1.4 Modelling considerations in CONCEPT

The plant simulation was attempted on CONCEPT Mark II and Mark III. Modelling on CONCEPT Mk.II proved to be impossible. Little information was revealed about its internal structure and the theories used in its unit models; no listing of the models were made available, and no models existed to represent the compressor and the distillation column. The latter had to be provided by the author whereas the former was made-up of a combination of the pump and the heater model, however it was soon realised that the exercise was becoming one of piecing-up new plant-flowsheets and forcing the plant data onto the simulator rather than simulating the available data on CONCEPT. The physical property generation package was an even greater stumbling block as it was later found out that the enthalpy routines provided data only for the saturated vapour and liquid phases and no

consideration was given to the superheated or subcooled regions, which are the more important phases in this plant. The difficulty in performing real simulations on CONCEPT Mk. II led to another joint project between the C.A.D. Centre and the author. The latter was asked to help implement a newly acquired thermo-physical property package on a revised version of CONCEPT, and to rewrite the models in a form compatible with the new property package. The author also added some missing models, the throttle valve and the polytropic compressor models.

This new version of CONCEPT, CONCEPT Mk. III, was much easier to use, both from a data input/output aspect and because of the better and more accurate properties and models offered. Routines were made available to the users to compute the following information from a knowledge of the remaining stream description: Stream enthalpy, temperature, and two-phase equilibrium compositions. With these powerful tools and a better understanding of the models, the simulation was simpler. Details about CONCEPT's new physical property package and its models' details is still confidential (although similar to the routines presented in chapters 8 and 9), and the author's silence is compulsory under the National Secrets Acts signed at the C.A.D Centre.

The simple question-and-answer data input section was very adequate to input the plant data although more guidance from the program in ordering the input would have made clearer the data preparation and sequencing. Less data was required here about

the feed and guessed streams as their enthalpy was automatically calculated by the executive, and the bubble point temperature of the feed stream was set by routine SETBP which received the feed stream as input and outputted the fully defined stream to the first unit. The distillation column and compressor models were those used in PACER/GEMCS and the other models were similar to those used there. Similarly the physical and thermodynamic data evaluation methods in CONCEPT gave results in close agreement to those of PACER/GEMCS. The filing system used in CONCEPT facilitated the storage and updating of the input data. Reruns required the modification of only the changing data whereas in PACER and GEMCS all the data had to be re-input for each run.

### 3.1.5 Process simulation study.

The results of a plant survey performed between the 8th and the 13th of June 1966 are presented in table 3-1. According to an accompanying report not included here, flow rates are accurate to within 5% of their true values, temperatures to within 2°F and pressures within 1%. The compressor outlet pressure was set at 585 p.s.i.a. The column's tops product purity is 95% ethylene and 5% ethane.

The simulation of only two cases was undertaken because of financial restrictions on computer time allocation and because of the slowness and poor turn round time provided by the Aston University computer. The first case studied was that of the 13th of June, the second one was that of the averaged data for the

six days. Since the consistency of the data could not be checked and because of the relative independence of the data from each other, their averaging should not have introduced much incompatibility between them. The important data for each case are shown in table 3-1.

#### 3.1.6 The data input.

The data input by PACER and GEMCS is shown in Appendix 1. Sections of the long print-out of interactive data input to CONCEPT are reproduced in figure 2.1. Data preparation for CONCEPT required no prior preparation since most data could be read off the flow-diagram, the survey data sheets or the user manual, but some thought had to be given to the correct sequencing of the data. Input was fairly quick (not accounting for the computer slow response time) and the free format for data input simplified the input further. Updating and correcting most input errors was straight forward due to the large number of updating commands available in the dialogue section. Unfortunately errors created by typing-in letters rather than numbers could not be corrected and caused runs to fail.

Data preparation for PACER was time-consuming. Coded sheets had to be filled in the correct format, all unknown data had to be input as zeroes and all matrices sized up to set the relevant tailoring control data. Card punching was also long because of all the large amount of unnecessary data to punch. The coded sheets however simplified the data gathering and sequencing task. Many data and punching errors could not be spotted until

The plant survey data.

SPLITTER OPERATING DATA

Date	June/66	8	9	10	11	12	13
<u>Flow Rates (M lb/day)</u>							
Distillate Product		145.0	153.3	173.1	173.1	168.7	139.8
Ethane to Pyrolysis		145.8	167.7	248.2	245.8	222.4	154.8
Ethane to Fuel Gas		-	-	-	-	-	-
Bottom Product		145.8	167.7	248.2	245.8	222.4	154.8
Feed Rate		290.8	321.0	421.3	418.9	391.1	294.6
Reflux Rate		730.5	731.3	731.3	730.5	730.5	730.5
Overhead Vapour Rate		875.5	884.6	904.4	903.6	899.2	870.3
Ethylene to Reboilers		740.0	757.6	757.6	757.6	757.6	735.0
Ethylene to C114 Condenser		135.5	127.0	146.8	146.0	141.6	135.3
<u>Temperatures (°F)</u>							
1. Feed at Condenser C-13-1		-5.8	-3.7	-0.7	-1.8	-1.9	-3.0
C-13-2		-5.6	-6.3	-3.6	-4.2	-4.0	-4.4
2. C <sub>2</sub> Splitter - Tray 10		-31.3	-30.0	-29.3	-31.2	-31.7	-32.4
Tray 14		-25.9	-26.5	-25.6	-24.2	-23.8	-24.9
Tray 19		-21.9	-21.5	-22.5	-24.0	-24.5	-22.0
(Tray above feed inlet) Tray 25		-21.1	-20.8	-21.8	-22.5	-22.5	-20.1
Tray 31		-10.6	-12.5	-18.3	-21.7	-20.0	-13.8
Tray 40		-3.0	-2.7	-8.1	-10.8	-6.8	-3.5
Reboiler Outlet C-44		-1.1	-1.2	-0.9	-1.9	-1.8	-2.5
(Ethane) C-44A		-1.1	-1.2	-0.9	-1.9	-1.8	-2.5
3. Overhead to C113		-39.8	-40.0	-39.5	-39.8	-40.0	-40
4. Compressor Inlet		-8.1	-6.8	-8.1	-8.3	-7.6	-8.0
5. Compressor Discharge		+112	+112	+113	+112	+113	+113
6. Ethylene at Retailer Inlet		+100	+100	+101	+100	+101	+101
7. Ethylene at reboiler outlet		+5.8	+6.5	+5.1	+4.4	+5.3	+4.8
8. Ethylene at C114 outlet		-24.4	-24.4	-23.4	-18.7	-17.8	-17.1

9. NA										
10. Ethylene F103 to C113	-2.0	-0.0	-0.8	-1.1	-1.4	-1.7				
11. Reflux and C113 outlet	-16.4	-16.0	-15.6	-15.6	-15.2	-16.3				
Ammonia Liquid to C114	-28	-29	-25	-26	-25	-26				
Ammonia Vapour from C114	-20	-19	-19	-11	-13	-12				
<u>Pressures (psig)</u>										
Barometric Pressure	-14.34	-14.29	-14.38	-14.39	-14.33	-14.29				
1. Feed Upstream of Controller	330	330	330	330	330	330				
2. Column Bottom	204	204	203	203	203	202				
Differential Pressure	2.88	3.01	3.01	3.05	3.07	3.03				
3. Top of Column (by difference)	201	201	200	200	200	199				
4. Compressor Inlet	200	200	200	200	200	200				
5. Compressor Discharge (log)	(496)	(495)	(494)	(495)	(495)	(495)				
"	497	497	497	497	497	497				
6. NA										
7. Ethylene at Reboiler	400	400	400	400	400	400				
8. Ethylene at C114 Outlet	450	450	450	440	430	430				
9. Pressure in P103	388	391	391	387	385	385				
10. NA										
11. Reflux and C113 Outlet	280	280.5	280.5 <sup>o</sup>	281.5	280	279.5				

SPLITTER OPERATING DATA

(Cont'd)

runs were attempted and failed. The data echo produced by the data reader was helpful but not always sufficient to trace all the errors as most of the echo was merely a copy of the input cards and not a check on their entire consistency. The major source of errors was in setting the size of the various matrices and omitting or adding extra cards with zeroes because of their large numbers.

Data preparation for GEMCS involved more work than for PACER. The order of calculation had to be traced by hand and the lack of executive-checking for convergence necessitated the addition of the control routine as part of the flowdiagram and the preparation of its particular parameters. Data coding following the GEMCS manual was straight-forward and card punching was shorter because of the minimal amount of punched data. A possible source of error was the data preparation for the convergence controller, however as was the case with PACER, data preparation was very much that of following printed instructions and abiding to a fixed rigid format. The completed set of data cards was in both cases quite unintelligible, and misplaced cards were difficult to spot except by rechecking each card.

### 3.1.7 The calculation order.

The plant diagram shows a number of interconnected and nested loops. No single stream can cut them all, but there are eleven possible sets of two streams that could tear them open: Stream 3 and Stream (3 or 14) (2 sets) and Stream (4 or 5 or 6)

and any one of Streams (15, 16 or 17) (9 sets). Following PACER's reasoning which is to start with streams with low numbers and work up to higher numbered streams, the set (3-13) would have been chosen as iterate streams, had no preferred streams been suggested. For practical reasons streams 4 and 17 were preferred. Stream 4 is of known pressure, its temperature is given in the survey sheets and its enthalpy is not needed initially required by the compressor model, whereas the enthalpy of streams 3, 5 or 6 is required. Stream 17, the recycle stream does not affect the column model which was made insensitive to the recycle conditions. Thus good guesses are unnecessary. Had the flow diagram not been studied beforehand, guesses for all streams should have been supplied to avoid the possibility of using zero-streams hence risking the high possibility of arithmetic errors (numerical over- or underflow) in the models.

GEMCS requires the input of the calculation order found by hand with or without the use of the programs "ORDER" to suggest initial starting streams (the same as those given by PACER were proposed since the method is identical). The order in this plant was easy to follow by eye but more complex plants would be very difficult to sequence by hand and the PACER and ORDER method fail on plants requiring more than three iterate streams.

CONCEPT accepted the preferred streams 4 and 17 and the sequence generated was identical to PACER's.

In all three cases the calculation order chosen was (in

unit numbers):

1 - (4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 3 - 12 - 13)

with the units in brackets forming one set of nested loops the calculation of which was repeated until convergence occurred

### 3.1.8 The results

The results of the two simulation cases are presented in table 3-1.

A large number of parameters had to be specified in the data input (see table 3-2), but the remaining variables were calculated satisfactorily. Some parameters had to be altered to obtain a consistent simulation, the major difference between the survey data and the simulation results was the fractional split at unit 6. The column reboiler required an increase in the heating medium flow rate of about 7%. The split had thus to be altered from 0.85/0.15 to 0.91/0.09. Accounting for a maximum deviation of 5% in the flow-meter readings still does not justify the difference, but the discrepancy could not be investigated further for the lack of more plant data. A few inconsistencies in pressure readings are also noticeable. These are due to the difficulty of modelling them exactly at the locations they were read at during the plant survey. They may have affected consequently certain temperature readings. The remaining simulation results were so good that the vapour content of the recycle stream to the column matched exactly that in the survey report, i.e. 10% molar vapour flow rate.

Table 3 - 1-a : Survey data and simulation results.

(average data)

Property	Location	Data	
		From survey	From simulation
Flows (lb mole/hr)	Feed	511.	511.
	Tops	232.	232.
	Bottoms	280.	280.
	Reflux	1085.	1057.
	Vapour Ov'head	1325.	1288.
	C <sub>2</sub> H <sub>4</sub> to reboiler	1120.	1185.
	C <sub>2</sub> H <sub>4</sub> to cooler	205.	103.
Compositions (C <sub>2</sub> H <sub>4</sub> mole %)	Feed	44.	45.
	Tops	95.	94.
	Bottoms	2.5	3.
	Throughout plant	95.	94.
Temperatures (Degrees F)	To plant	?	-20.
	Column top	-40.	-40.
	Tray No. 10	-31.	-31.5
	Feed plate	-21.5	-22.5
	C <sub>2</sub> H <sub>6</sub> at reboiler	-1.6	-6.
	C <sub>2</sub> H <sub>4</sub> to reboiler	100.5	102.5
	C <sub>2</sub> H <sub>4</sub> out of reboiler	5.	3.
	Inlet 2 to heat exchanger	-1.	0.
	Outlet 2 from heat exchanger	-16.	-17.
	Compressor in.	-8.	-6.
	Compressor out.	112.5	118.5
	Condenser out.	-21.	-21.
Pressures (p.s.i.a.)	To plant	344.	344.
	Column	215.	215.
	Compressor in.	214.	215.
	Compressor out.	585.	585.
	Reboiler out.	414.	457.
	Condenser out.	457.	457.
	Inlet 2 to heat exchanger	402.	402.

	Outlet 2 from heat exchanger	294.	402.
Reflux ratio	To column	4.6	4.6
Vapour fraction	In reflux	.10	.10
Heat loads (M btu/hr)	Condenser	-	550.
	Compressor	-	1200.
	Reboiler	-	5500.

Table 3 - 1-a : Survey data and simulation results.  
(average data)  
(Cont'd)

Table 3 - 1-b : Survey data and simulation results.

(6th survey day).

Property	Location	Data	
		From survey	From simulation
Flows (lb mole/hr)	Feed	425.	425.
	Tops	205.	205.
	Bottoms	215.	215.
	Reflux	1095.	1076.
	Vapour Ov'head	1290.	1282.
	C <sub>2</sub> H <sub>4</sub> to reboiler	1090.	1166.
	C <sub>2</sub> H <sub>4</sub> to condenser	200.	115.
Compositions (C <sub>2</sub> H <sub>4</sub> mole %)	Feed	44.	45.
	Tops	95.	94.
	Bottoms	2.5	3.
	Throughout plant	95.	94.
Temperatures (Degrees F)	To plant	?	-20.
	Column top	-40.	-37.5
	Tray No. 10	-32.5	-24.
	Feed plate	-20.	-19.
	C <sub>2</sub> H <sub>6</sub> at reboiler	-2.5	-5.
	C <sub>2</sub> H <sub>4</sub> to reboiler	101.	97.
	C <sub>2</sub> H <sub>4</sub> out of reboiler	5.	0.
	Inlet 2 to heat exchanger	-2.	-2.
	Outlet 2 from heat exchanger	-16.	-18.
	Compressor in.	-8.	-9.
	Compressor out.	113.	115.
	Condenser out.	-17.	-17.
Pressures (p.s.i.a.)	To plant	345.	345.
	Column	215.	215.
	Compressor in.	214.	215.
	Compressor out.	585.	585.
	Reboiler out.	414.	444.
	Condenser out.	444.	444.
	Inlet 2 to heat exchanger	399.	399.

	Outlet 2 from heat exchanger	294.	399.
Reflux ratio	To column	5.25	5.25
Vapour fraction	In reflux	.10	.09
Heat loads	Condenser	-	610.
(M btu/hr)	Compressor	-	1202.
	Reboiler	-	5540.

Table 3 - 1-b : Survey data and simulation results.

(6th survey day)

(Cont'd)

Unit	Parameters	Value
Throttle valve (I)	Outlet pressure	215. psia
Dist. column (II)	Number of stages	40
	Feed stage	20
	Pressure	215. psia
	Tops flowrate	230. psia
	Tops vapour flow	230. moles/hr
	Tops composition (approximate)	.95 C <sub>2</sub> H <sub>4</sub> .05 C <sub>2</sub> H <sub>6</sub>
	Light component	1
	Heavy component	2
Reflux ratio	4.6	
Heat Exchanger (III)	Temperature diff.	6.5 °F
Compressor (IV)	Outlet pressure	585. psia
	Isentropic eff.	.75
	Mechanical eff.	.94
	Parameter k	1.23
Throttle valve (V)	Outlet pressure	457. psia
Splitter (VI)	Split 1	.91
	Split 2	.09
Cooler (VII)	Outlet temperature	414. °R
Throttle valve (VIII)	Outlet pressure	414. psia
Throttle valve (IX)	Outlet pressure	414. psia
Throttle valve (XI)	Outlet pressure	402. psia
Splitter (XII)	Split 1	.82
	Split 2	.18
Throttle valve (XIII)	Outlet pressure	215. psia

Table 3 - 2 : Units Parameters for the C<sub>2</sub>H<sub>4</sub> / C<sub>2</sub>H<sub>6</sub> Plant.

The three programs gave very similar results because of their fairly similar models and thermo-physical evaluation routines.

### 3.1.9 General comments.

CONCEPT was much easier to use because of the available thermo-physical evaluation routines whereas all the data for PACER and GEMCS had to be obtained through a literature search and accurately correlated

The data input into CONCEPT demanded less preparation but required long sessions of teletype interaction with the computer because of the over-booking of the Atlas Computer at the C.A.D. centre. Its updating facilities were very useful whereas data correction in PACER required a tedious search for the wrong cards.

The intermediate printout generated by PACER and GEMCS was very helpful in debugging the models. CONCEPT's models required no debugging though CONCEPT also provided an intermediate printout, but the postal delay in receiving the line-printer output reduced its importance in data and model checking but the partial printout on teletype and deal in overcoming the slowness of the teletype printing and helped in updating the input data.

The speed of the three programs could not be compared because of the differences in the computers used and because of the absence of the calculation order finder in GEMCS. To keep the running times as low as possible, initial iterate streams

guesses had to be close to the final answer (these are shown in the input data in table 2-2). The simulation converged to a tolerance of 1% in 4 loop iterations.

No extra executive facilities were needed to complete the simulation successfully.

### 3.2 Example Two:

In Example 1, the fundamental requirements of simulation programs were illustrated; however, using the basic tools demonstrated there, the results that can be obtained are rather limited. It wasn't shown how to update unit parameters internally from improved designs nor was it shown how to optimise some objective function. In this example these tools and other uses are described through the presentation of an optimiser routine and the optimisation of a cascade extraction plant.

Little has been reported about introducing optimisers in plant simulators mainly because of the unfeasible proposition to optimise automatically an already expensive simulation and also because of the difficulty in incorporating such a program without changing the form of the executive. In an attempt to introduce an optimiser, at McMaster University, the GEMCS executive was transformed into a subroutine of the optimiser. This required rearranging the data input to avoid multiple data readings, and restructuring the executive; however recalculating the units outside the optimised sequence could not be easily avoided.

The alternative was to write an optimiser that would have all the characteristics of a unit model such as parameters and associated streams and which as a readily available package could be included in any study as a fictitious unit at the end of the sequence to optimise.

The requirements for such a module are:

1. In case of constraint violation, when varying the direction of search, the unit sequence should not be calculated, but the direction changed.
2. The optimiser would have access to the parameter list of other units where it would update the design variables.
3. Only units in the specified sequence would be recalculated.
4. The module would not change for different problems.

This routine has been written based on an optimisation program developed at McMaster University (66). It uses the Hooke and Jeeves method for simplicity, although any other method can be incorporated in a similar way, and it incorporates the above-mentioned features in the following way:

1. The constraints are checked in a separate routine before calculating the units. The search pattern or step length is changed in cases of constraint violation and the constraints are rechecked.
2. All the information the optimiser requires, except the constraint equations, are introduced through its parameter list.
3. The constraints are set in an easily updatable routine.

4. Access to other units' parameters is from within the optimiser.
5. Repetition of the calculation sequence is executive-dependent.

To access other unit-parameters, in GEMCS, the parameter handling routine of the executive is called and supplied with the number of the unit where parameters should be changed. It extracts them and returns them to the optimiser. After modification, the same routine is recalled to replace them in the cumulative array. In PACER, the parameter vector is directly accessible in the equipment matrix when the unit number is known, (the row subscript in the matrix). In CONCEPT such a possibility is not yet available since the parameters are relocated in a new matrix and the units are renumbered internally. This deficiency is overcome by introducing the design variable in a fictitious recycle stream from the optimiser to the various units, however the models require some modification.

To control the sequence repetition, in GEMCS, a variable (NC) pointing at the position of the unit calculated in the calculation order can be returned in the optimiser to point to the first unit in the sequence by subtracting from its value the number of units in the sequence. In PACER such a pointer does not exist. The problem is resolved by introducing a fictitious recycle stream containing the objective function value, for convergence check purposes, from the optimiser to the first unit in the sequence. The same method is used in CONCEPT.

The listing of the GEMCS version of the optimiser is illustrated in Appendix 2.

### 3.2.1 The application of the optimiser

To illustrate the use of this optimiser, the financial benefit from a three stages cross-current extractor plant (67), figure 3-3, was optimised using PACER and GEMCS. Because of financial restrictions this problem could not be solved on CONCEPT. The plant of example 1 was too long and too expensive to optimise.

In these extractors, a valuable solute X costing one unit per pound is extracted from a carrier Q by an immiscible solvent W costing 0.05 units per pound. Pure solvent is fed to each extractor. The maximisation of the profit is achieved by selecting the optimum solvent flow rates to each extraction.

$$\text{Profit} = \sum_{i=1}^3 (Q * (X \text{ in}_i - X \text{ out}_i) * 1 \text{ cost unit/pound} - W_i * 0.05 \text{ cost unit/pound}) \quad (3.2-1)$$

Feed Concentration = 0.135 pounds X per pound of Q

The Mass balance equations are:

$$\text{Carrier flow rate } Q \text{ in} = Q \text{ out} = 1000 \text{ pounds/hour} \quad (3.2-2)$$

$$\text{Solvent flow rate } W \text{ in}_i = W \text{ out}_i \quad (3.2-3)$$

The Equilibrium relationship between the solute in the solvent Y and in the Carrier (X) :

$$Y \text{ out}_i = F (X \text{ out}_i) \quad (\text{see figure 3-4}) \quad (3.2-4)$$

(the relationship was approximated by four straight lines).

The solute mass balance:

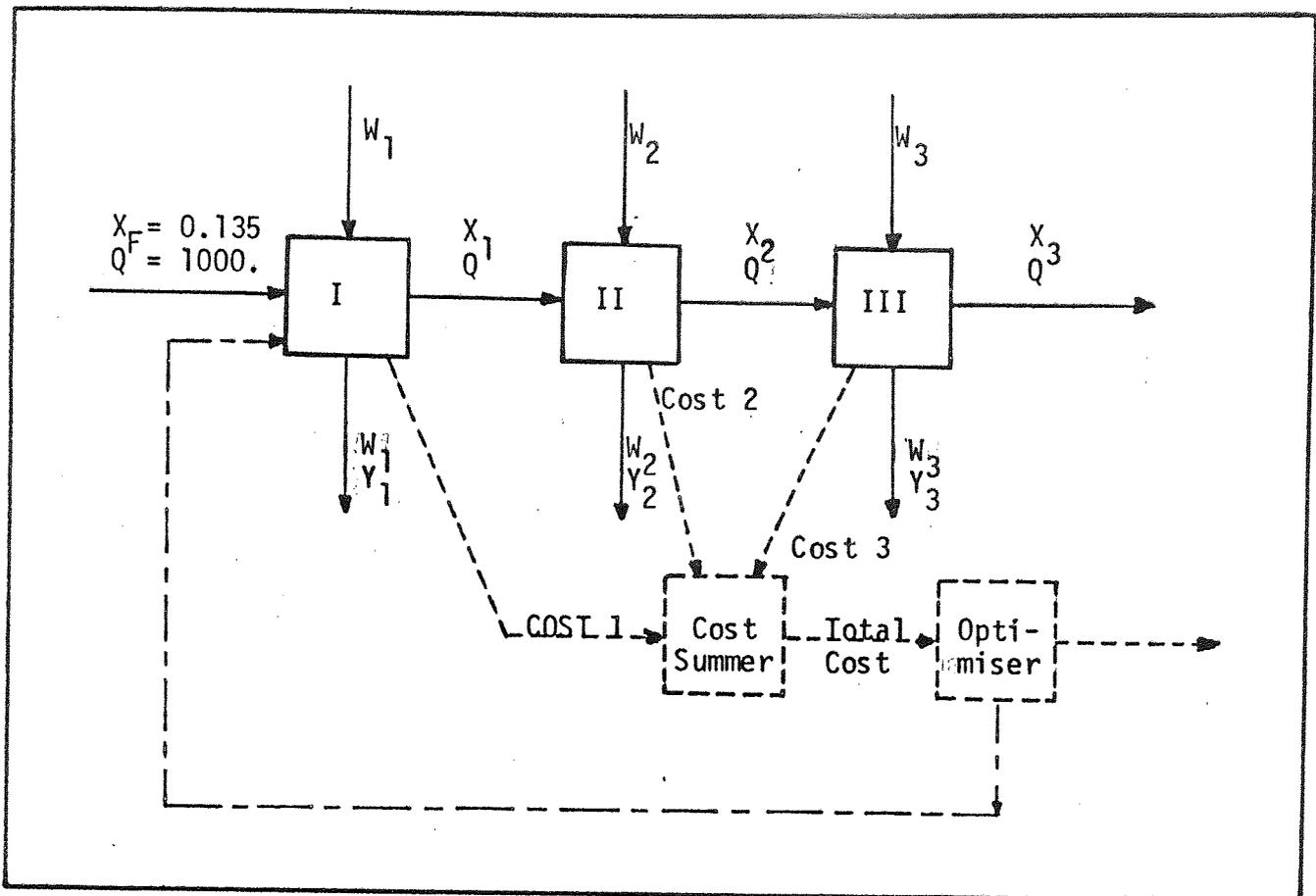


Figure 3 - 3 : Three-Stage Cross-current Extraction Plant

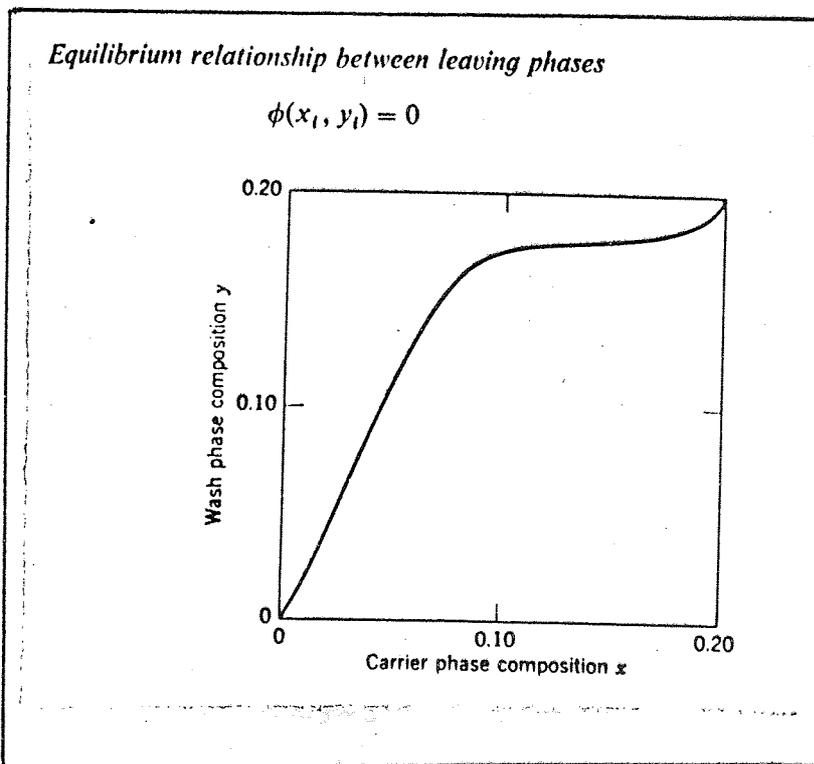


Figure 3 - 4 : The Equilibrium Relationship

$$Q * X \text{ in}_i = Q * X \text{ out}_i + W_i * Y \text{ out}_i \quad (3.2-5)$$

The problem was solved by varying the outlet solute concentrations in the carrier (the design variables) for each extractor between 0.00 and 0.135 (the constraints), and by condensing equations (3.2-4) and (3.2-5) into:

$$W_i = Q * (X \text{ in}_i - X \text{ out}_i) / F (X \text{ out}_i) \quad (3.2-6)$$

The initial guesses for the outlet concentrations  $X \text{ out}_i$  were 0.08, 0.05 and 0.02. The search step length was initially set at 0.02 and allowed to decrease to 0.001.

The problem should have been solved by using the solvent flowrates as design variables, but running time would have risen because of the trial and error searches to compute output concentrations at every extractor calculation. The faster, but less obvious, approach was preferred for demonstration purposes only.

In GEMCS the only convergence controller was the minimum step length allowed, while in PACER because of the extra fictitious recycle stream, all streams were checked to a preset tolerance of 0.1%. This hindered the convergence as even small step changes in different directions kept on changing the streams compositions by more than 0.1%. Increasing the tolerance on the convergence check superseded the check on the design variables step lengths. The solution finally adopted was to set the step lengths to zero when the minimum step lengths were reached, and repeat the

calculations with the same design variables, ensuring a 0.00% deviation and forcing the termination of the iterations. The number of calculations was hence increased by one only.

### 3.2.2 The results

The final results showed an increase in benefits from 53.4 units per hour at the initial variables values, to 62.1 units per hour at the optimum, an increase of 14%. The flow rates of solvent changed from 314, 169 and 184 pounds/hour to 517, 170 and 198 pounds/hour. The optimum outlet concentrations were 0.06, 0.0425 and 0.0294 pounds of X per pound of Q. The number of sequence calculations was however very large (84 sets) for this 3-dimensional problem due to the poor optimisation technique used.

The use of automatic optimisers in multi-dimensional problems is debatable, but such programs should be available should a justifiable need arise.

The object of this exercise was to demonstrate methods of adapting unconventional routines to flowsheeting programs and overcome the barriers presented by rigid executives. It also helped to point out some differences between GEMCS, PACER and CONCEPT. It may be correctly assumed for the latter part of this exercise that had CONCEPT been used, it would have performed similarly to PACER.

### 3.3 Conclusion

A number of other types of problems were solved with the flowsheeting programs. The two problems described here do not cover all the possibilities of these programs, but they point out most clearly to their main differences.

## CHAPTER 4

### AN APPRAISAL OF THE PROGRAMS AND THE NEED FOR A

#### NEW ONE

The three programs have their own advantages and disadvantages which make them more suitable to certain types of users. The following appraisal emphasises the desirable aspects of each program in the light of users requirements, leading to the selection of the more suitable one for each of the following three classes of users:

1. The undergraduate student.
2. The postgraduate research student.
3. The practicing engineer.

The limitations of the programs will then be studied to support and justify the need for a new flowsheeting program.

The aims of each group of program users are different and each user expects to find in his program the necessary features that will help him achieve his goals most efficiently. For the undergraduate the goal is an increased awareness of the possibilities of computer simulation and a better understanding of unit operation modelling through the preparation of models. The post graduate researcher in computer-aided simulation and design will soon become familiar with flowsheeting details and will probably want to improve the executive systems to handle more complex plants faster, and to provide new features and facilities. The practicing engineer generally wants good,

consistent and quick results he can easily interpret and use. Table 4-1 summarises the main features that should exist in the program appropriate to each case. They emphasise the learning process for the undergraduate, the flexibility of the executive for the post-graduate and the ease and speed of case studies for the engineer. The speed of calculation is an obvious desirable feature in all cases and is not mentioned in table 4-1. Because of the different computers used to run the three programs described earlier, and because of their different features, running times could not be compared meaningfully. All the arguments presented the author's opinion based on his personal experience.

#### 4.1 Case 1: The undergraduate student

The most important aim for the student is to understand how unit operations work and learn to write their mathematical models. Model writing teaches him to apply the principles of physics, chemistry and engineering to the solution of heat and mass balances in various unit operations at various levels of sophistication. It also gives him a feeling for the effect of the essential parameters on the units performance. The thermodynamic and physical data are an integral part of modelling. To use them, the student must first obtain them from the literature then correlate them as accurately as required and set them in routines compatible with his own models. Existing models and data may either be used along with his own models or serve as guide-lines for better models. Model writing for all

The Undergraduate	The Postgraduate	The Engineer
<p>Usually involved with :</p> <ul style="list-style-type: none"> <li>The writing of models</li> <li>The study of parameters</li> <li>The study of initial guesses of operating conditions.</li> </ul> <p>Requires :</p> <ul style="list-style-type: none"> <li>An optional calculation order finder</li> <li>Large intermediate printout</li> <li>Ease of data input</li> <li>Batch processing</li> <li>A Library of models.</li> </ul>	<p>Usually involved with :</p> <ul style="list-style-type: none"> <li>The writing of design orientated models</li> <li>The improvement of the executive.</li> </ul> <p>Requires :</p> <ul style="list-style-type: none"> <li>Calculation order finder usually unnecessary</li> <li>Large intermediate printout</li> <li>Short data input</li> <li>Batch or interactive processing</li> <li>Internal re-ordering of calculation order should be possible</li> <li>Referencing to other unit arrays</li> <li>Small executive size.</li> </ul>	<p>Usually involved with :</p> <ul style="list-style-type: none"> <li>Plant design</li> <li>Plant simulation and optimization.</li> </ul> <p>Requires :</p> <ul style="list-style-type: none"> <li>Most basic models to be available</li> <li>A calculation order finder</li> <li>Minimum printout</li> <li>Interactiveness</li> <li>Ease of data input and update</li> <li>Fast access to the results</li> <li>Immediate restart of run</li> <li>Limited interest in the executive.</li> </ul>

Table 4 - 1 : Requirements in simulation programs for different users.

three programs have similar requirements except that the available property estimation facilities in CONCEPT make their writing easier.

The speed of recycle loop convergence depends strongly on the initial iterate stream estimates. The closer the guesses are to the final answer, the faster and more certain is the convergence. Initial guesses reflect the insight of students for the overall plant behaviour and reduce the cost of runs, however they depend very much on the chosen order of calculation. For simple plants the student should be able to find the optimum order himself, but for more complex plants an order finder routine may be necessary and should be available. GEMCS allows the most direct stream guessing as the user sets himself the calculation order. He may introduce as many guessed streams as thought desirable. CONCEPT allows the user to pick his iterate streams and supply guesses for them but the user may not introduce his own unit order. PACER permits the guessing of any stream description, but these may not necessarily be used as starting streams in recycle loops calculations unless the user has studied the plant diagrams in detail and suggested the correct iterate streams or his own calculation sequence.

Intermediate printout is very essential to the student. By generating large amounts of printout from his models, he can check their correctness and accuracy. He can also check the consistency of the convergence by analysing the changes in the

stream descriptions. Both PACER and GEMCS can provide intermediate printout by the setting of particular switches. The new version of CONCEPT generates a trace of the streams into and out of each unit calculated, but this printout, unless specifically listed on the line-printer, is automatically erased when solving a new simulation case.

The ease of data input is another important requirement. A standard but free-format data input, preferably from coded-sheets, prevents omissions and facilitates the check of input data, however conciseness is also an asset. PACER offers a coded-sheet-type of input but requires too much unnecessary data, usually input as zeroes, to fit the coded input. GEMCS has a very concise and simple data input mainly because of the little work the executive does. CONCEPT's interactive data input is very easy to feed, but its interactiveness is a drawback at the undergraduate level and batch-mode input is much preferable. The student can hand in a number of problems in one batch and collect the results together at a later stage rather than sit at the teletype inputting his data at a slow pace because of the computer's divided attention. On-site batch processing has the advantage of providing large amounts of immediately-available printout. CONCEPT's line-printer output must still be posted to the user thus delaying its reception by 24 to 48 hours.

This last point, probably more than any other, makes CONCEPT a less interesting alternative until it can be made freely

available in source form (at least partly) to academic institutions and on their own computers. The main advantage of PACER is its optional calculation order finder routine, and its coded data input may be of some help to beginners with no experience of flowsheeting programs. GEMCS offers a more concise data input and can be run on small or medium computers (8 K to 32 K core) as opposed to PACER's need for larger computers. GEMCS is also very much simpler to understand or teach.

#### 4.2 Case 2: The postgraduate researcher

The research of postgraduates specialising in the development of new features in flowsheeting would normally centre around three main topics:

1. The preparation of general and accurate design-orientated models.
2. The optimisation of plant performance from either the cost of the design or the parameter variation aspects.
3. The improvement of the features offered in executive programs to handle larger and more complex simulations faster.

With respect to the first point, the emphasis is on high level model writing. The presence of prepared models may serve as an initial base for more advanced models, but for most unit operations more complete design-orientated models will have to be written. The use of a flowsheeting at this stage, if at all necessary, will be to assist in the data communication and the tying-up of

the auxiliary property and data communication routines. The second point is illustrated in example 2 in chapter 3. There, the importance of referring back to the parameter arrays of other units is clearly demonstrated. This requirement might prove of even greater importance in design-type flowsheeting programs where the unit parameters may have to be set internally. The internal reordering of the calculation order, i.e. the momentary by-passing of the preset order, either for optimisation or design checking is another important requirement in design programs. PACER offers the easiest access to other unit parameter arrays but resequencing the calculation is not possible. GEMCS is much more flexible on reordering the calculation sequence but access to other unit parameters is less simple. CONCEPT does not yet provide an easy solution to either problem.

In design-orientated problems, the models, the physical property and cost evaluation routines will be larger than their equivalents in simulation problems. The executive which may evolve from a simulation-type executive will also be larger in size with more checking and optimisation routines, but to have more control on the design procedure, the calculation order finder may probably be by-passed. To accommodate these new requirements within reasonably sized computers, a short and highly modular executive should be used. PACER is highly interconnected internally and cannot have its order finder easily removed. CONCEPT is more modular but its internal set-up has not been revealed and may well be as complex as PACER. GEMCS

is the most compact and modular executive and has no order finder.

Intermediate printout is as important to the postgraduate as to the undergraduate and for the same reasons. The program's most suitable processing mode would be a semi-batch mode where the data input and updated through teletype or by cards and the bulk of the execution is performed as a background job. However fully-batch operations are preferable to fully-interactive ones, at present, because of the type of slow computers most widespread in Britain.

In summary, the most suitable program is the shortest and most modular one and GEMCS is the best choice.

#### 4.3 Case 3: The practicing engineer

The engineer's main use of flowsheeting programs is to get information about a plant layout of interest to him, either to run the plant at more profitable conditions or to study the effect of higher feed rates or different feed stocks. He may also be interested in obtaining consistent initial estimates of flows and energy demands and approximate unit dimensions for preliminary design (a complete plant design in one run is not yet possible on most present-day flowsheeting programs).

Since computer-aided simulation and design is a fairly new field, the engineer will usually have little or no experience at all in computer-orientated modelling, and thus will want to

have most of the basic models and thermo-physical data available within the package used, though he will still have to provide the models for the special units characterising his plant. The models and data should be fairly general and accurate and only CONCEPT answers this need.

Flowsheeting programs that update unit parameters internally for overall consistency in simulation results, or that optimises a plant performance automatically are not wide-spread. The engineer must try different combinations of parameters to achieve his aims. To produce his work rapidly he needs a fast turnround time per run, immediate access to the results of each run, the option of changing the minimum amount of data and the facility to re-run immediately. The engineer cares only for the final results in a run and may even be interested in only a few key streams, the description of which he would ask for specifically as offered in CONCEPT. The fastest access to a computer is via a teletype terminal and only interactive programs with access to computer data-storage files can provide all these features. An interactive version of GEMCS exists (54) but it does not have access to data storage files. PACER offers none of these requirements.

The method used to obtain the calculation order is of little interest to the engineer as long as it is quick, general and efficient. He will certainly prefer to have a calculation order finder available in the package he uses to free him from

a tedious and time-consuming task that can be done more efficiently by computer. Only PACER and CONCEPT offer calculation order finders and CONCEPT's method is far superior and more general. CONCEPT also provides a convergence promoter to accelerate the simulation calculations.

CONCEPT is clearly the only choice to the practicing engineer, on all accounts.

#### 4.4 The importance of selecting the appropriate program.

The importance of assigning the appropriate type of program to each user cannot be overstressed. All programs could be used by all users, however the additional work needed to achieve one's aims when using the wrong type of program, the inconvenience of over- or under-specific programs and data inputs, the lack of suitable models, data, facilities, etc., would soon lead the user to frustration, waste of time and eventually underestimating or abandoning computer-aided simulation and design. PACER and GEMCS are satisfactory for academic exercises, but only CONCEPT is truly Industry-orientated.

#### 4.5 The need for a new flowsheeting program.

There are over 30 flowsheeting programs presently in use or at various stages of development in North America and in Europe (1) and to propose a new one for public use would seem almost pointless if the author and his research supervisor did

not have strong reasons to justify it.

The prime justification for a new flowsheeting program is drawn from the above statement: It should be for PUBLIC, i.e. FREE, use.

The appraisal made above concentrated on three of the best known or more easily available programs in Britain, but of these only one is freely available (GEMCS). Save two or three other programs either less well known or as incomplete as GEMCS such as ESSPROSS (47) and UNICORN (48), or less easily, but not freely, available such as I.C.I's FLOWPACK, the wider consumer group of chemical engineers in British industries and Universities has little choice among other available programs and no choice at all in non-proprietary ones that can claim an attempt at comprehensiveness, high flexibility and ease of use.

CONCEPT, Britain's best developed program, has recently only (late 1973) been offered to Universities but at considerable cost and only in object form (compiled binary version) to be installed by the providers. FLOWPACK is also being offered at the same conditions to Universities but with only part of its features made available (56). On the other hand, small chemical companies that are not capable of creating and maintaining their own complete chemical process simulator must rely entirely on a limited number of computer bureaus offering this black-box type facility. Their approach to computer-aided design might be totally different

if a new executive with an initial bank of data models and physical property estimation routines was made available at a reduced cost.

It may be argued that the smaller, GEMCS-type, executives are a better answer to the present needs of many users because of their simplicity and adaptability to different situations and different computers. This is correct for occasional jobs, but once they are used for general problems and thermo-physical data routines are gradually added to them, do they not grow into large-scale but poor flowsheeting programs because they were not initially designed as large-scale executive programs? Some modifications are initiated and an almost complete restructuring of the executive is necessary if not unavoidable.

The other justification, of equal importance, for a new program is its technical contents. Though his use of a number of simulators, the author has realised that some important and common deficiencies existed in most of them. These were:

1. The poor input/output data communication between the programs and the user.
2. The difficulty in tracing back errors because of the lack of internal diagnosis.
3. The various technical limitations present in the programs such as the poor calculation order finders, the lack of convergence promoters, the absence of the most basic thermo-dynamic functions etc.

4. The difficulty in teaching the principles of flowsheeting and the time consuming learning process to apply the programs to plant simulation caused by the limited documentation on the programs.

In the light of the above discussion, the following criteria (41) should be met to justify the preparation of a new flowsheeting program.

1. The program must be made freely available to all interested users.
2. It must be constructed in a highly modular form and in a high level language, for ease of installation, understanding and maintenance.
3. It must be easily adaptable to specific problems by ad hoc modular extension.
4. The Input data structure must be flexible to allow a wide variety of small and large applications and to facilitate the information flow within the network of programs.
5. Algorithms and computational sequencing methods must be planned to minimise numerical sensitivity and convergence instability in iterative processes.
6. The models and thermo-physical property evaluation routines should be based on well-documented theoretical and numerical methods (especially for academic use).
7. Error trapping and tracing facilities should make the program run smoothly and help the user diagnose most error sources.

8. The program should incorporate as many of the requirements set in table 4-1 to be useful to all classes of users.

Fulfilling the above criteria adequately should justify the development of a new flowsheeting program. Proposing it as a Ph.D. topic is probably the safest way of ensuring a more homogeneous base to the whole project, the most justifiable way financially and time-wise and possibly the only way of producing a non-proprietary package.

#### 4.5.1 PEETPACK

PEETPACK (Process Engineering Evaluation Techniques Package) is Aston University contribution towards fulfilling this need for a new flexible, complete, and above all non-proprietary flowsheeting program. All the prerequisites cited above have been taken into account directly or indirectly and the program was developed entirely on local resources to ensure its non-proprietary nature. It is hoped that the final product, although still at the development stage, will be considered by users as a good justification for the effort put in. The critical appraisal of the three programs given earlier has had a direct influence on shaping the program in the hope of it being received favourably by undergraduates, postgraduates and process engineers alike.

## CHAPTER 5

### THE GENERAL OUTLINE OF PEETPACK

PEETPACK has been written with the aim of fulfilling a number of important criteria listed previously. From a technical point of view its major objectives are:

1. Ease of use, learning and teaching.
2. Extensive computerisation to minimise the input data.
3. High modularity for ease of installation and updating.
4. A wide range of thermodynamic and physical property routines.
5. A basic set of general unit models to draw from or to complement the users model sets.

These aims were achieved through a thorough study and redevelopment of the various constituents of conventional flowsheeting programs, retaining nevertheless the overall structure of these programs for three main reasons.

1. It provides the most logical approach in dealing with modular simulation orientated programs.
2. It is very easy to teach to students for whom it is primarily intended.
3. Flowsheeting programs remain simulation-orientated because of the lack of standardised plant design approaches and the need for consistent heat and mass balances before implementing design procedures.

## 5.1 Meeting the objectives.

The first two objectives were achieved by writing completely new data input/output/update packages for batch and interactive mode inputs into which the user inputs the minimum amount of data, most of which is in comprehensible alphanumeric form. The task of counting the number of streams or components or any unnecessary tasks is delegated to the program. The ease of data input was enhanced by creating a completely free "FORMAT" to read alphanumeric strings of characters of unspecified length and position.

A longer description of the data input facilities and of the novel alphanumeric free-format is presented in chapter 6.

The high modularity of the whole program is achieved by splitting the various tasks in the package into a number of semi-independent or free-standing programs and dividing the tasks within each program into a number of fairly short routines all connected together by labelled COMMON statements. Thus PEETPACK is made up of two data communicators, an executive program which contains only the routine to control and accelerate the simulation, one library of models and two libraries of property estimation routines. In every program (except for the properties) each subroutine performs one duty only and relies directly on as few other routines as possible so that its replacement is easy and does not disturb the rest of the program. The break-down of the executive into its eleven different routines

is detailed in chapter 10 and the rest of the package breakdown, illustrated in figure 5-1, is explained below.

A set of fifteen unit models are presented in chapter 8. They deal mainly with the basic physical processes such as heat transfer and separation processes but a reactor model is also offered although chemical processes units are usually very process-specific and users are advised to add on their own models. The models are set in a "LIBRARY" package divorced from the executive program to push the modularity concept even further.

The thermodynamic and the physical property routines are also set in their LIBRARY's. These property routines cover non-polar hydrocarbons and pure gases, but the data in the data banks (two magnetic tape files accessed as data files) are given for only 45 components. These files and the routines are described in chapter 9.

The objective of ease of implementation and usage are also met in the general layout of the program and the addition of models and property data input facilities offered by the MACRO's (Job Control Language commands) which control the structure of the package and allow a free mixing of interactive and batch mode data input and update.

## 5.2 The structure of the package.

The structure of conventional flowsheeting programs has been briefly summarised in chapter 1. The same structure applies

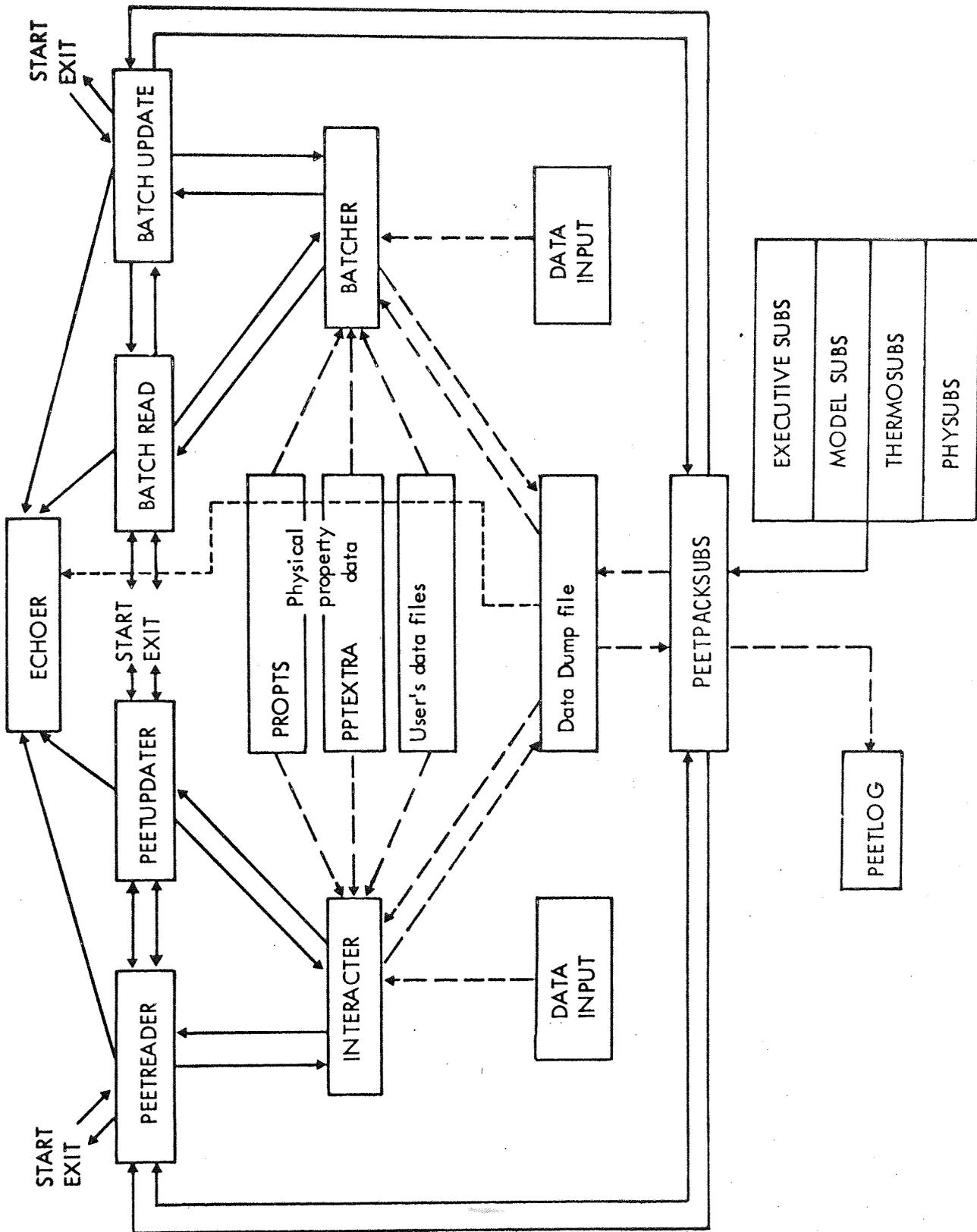


Figure 5 - 1 : The structure of PEETPACK.

also to PEETPACK although its implementation is very much more elaborate.

PEETPACK is subdivided into six major sections (and 10 programs).

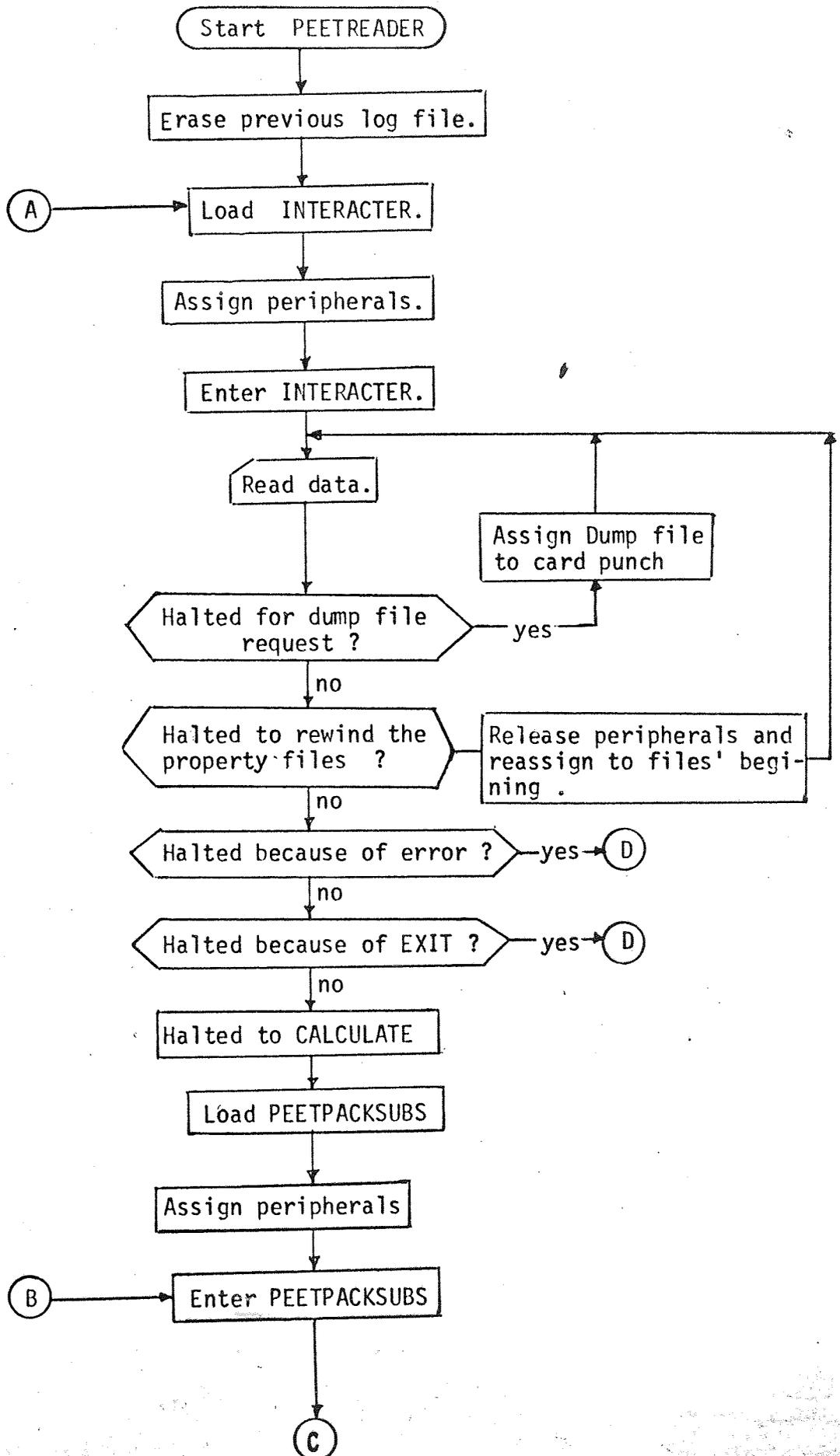
1. An interactive-mode and a batch-mode data communicators (INTERACTER, BATCHER) containing the calculation order finder.
2. The executive control program (PEETPACKSUB, EXECUTIVESUB).
3. A library of unit models (MODELSUBS).
4. A library of thermodynamic property routines (THERMOSUBS) and one of physical property routines (PHYSUBS).
5. A data-echoer program to print-out the stored data (ECHOER).
6. A bank-file of thermodynamic data constants (PROPTS) and one of physical property data constants (PPTEXTRA).

Two sets of two MACRO's each (one set for interactive-mode control, PEETREADER and PEETUPDATER and the other for batch-mode, BATCHREAD and BATCHUPDATE) control the sequencing of these six sections and the assignment of the various peripherals (card readers, line printers, card punches) which are used to communicate data and information between the different package sections and with the user. The MACRO's listings is given in Appendix 3, and the flowchart of PEETREADER is shown in figure 5-2, the other three being very similar.

### 5.3 The Package sections sequencing.

A user may input his data either interactively or batch-

Figure 5 - 2 : Flowchart of the PEETREADER Macro.



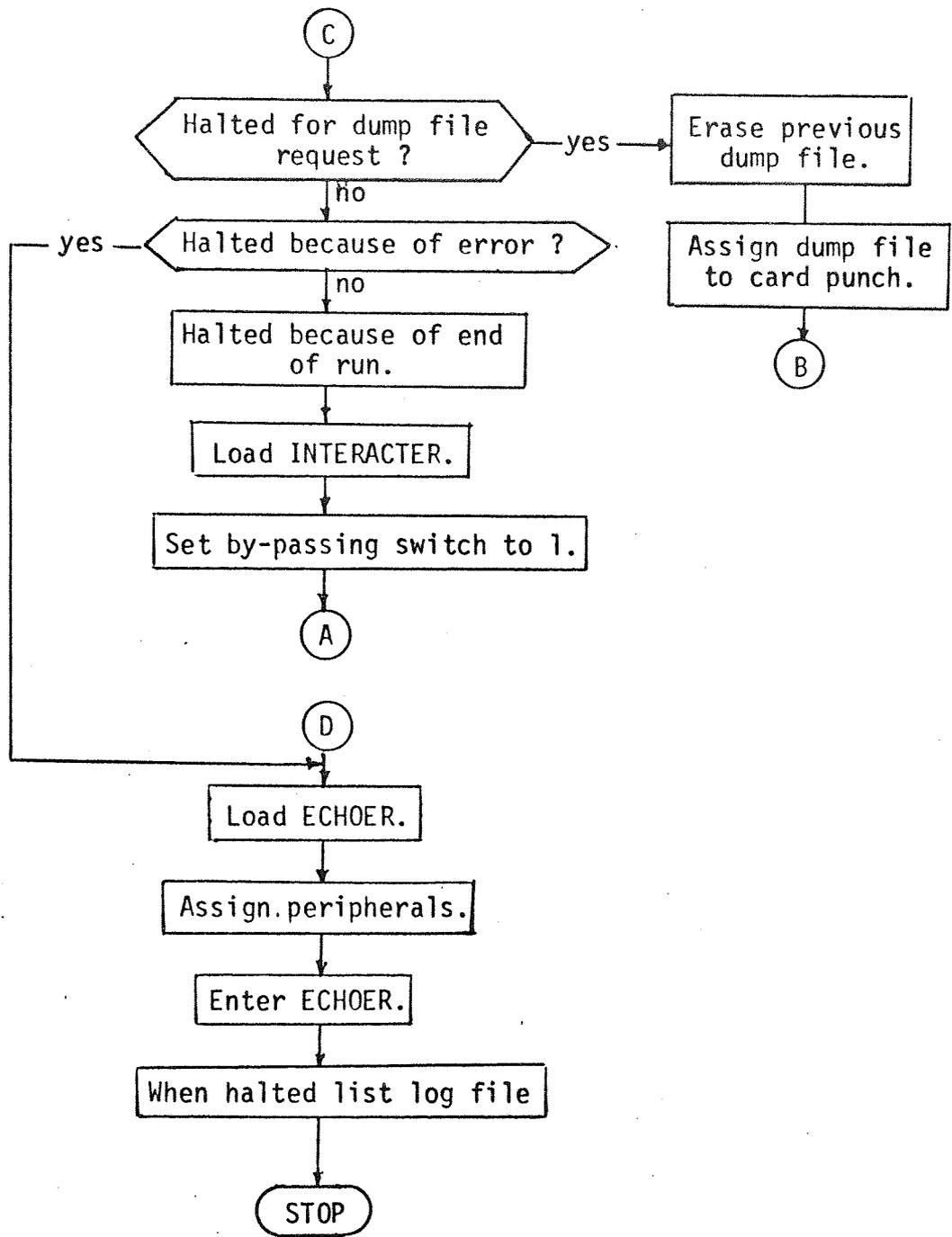


Figure 5 - 2 : Flowchart of the PEETREADER Macro.

(Cont'd)

wise or partly in each mode. To that effect he calls the appropriate MACRO by commanding on a card or by teletype the MACRO name followed by the names of certain files to be assigned to the package (this point is fully described in the next section). The MACRO loads in core the requested data-input program and assigns to its various peripherals the files which contain the data it needs. Since the executive merely controls the simulation run, all the thermo-physical data must be available as pre-processed data. Thus the peripherals used by the data reader are two card-readers assigned to the two data files (PROPTS and PPTEXTRA) from which the reader will extract the data for only the compounds in the plant simulated, a card-punch attached to the new dump-file which will contain the interpreted data for access by the executive, the data echoer or the updater, a card-reader and a line-printer (both of which are the teletype terminal in interactive mode) for data communication with the user, and three additional card-readers to read the (optional) additional property data input by the user (see section 5.4 and table 5-1). Following these assignments the data-reader program is entered and reads the data from the card-reader or teletype terminal. A number of PAUSE commands within the reader return the command to the MACRO to re-assign to some of the peripherals the same or other files after either rewinding these files or erasing their contents. The first case arises when the data files have been scanned entirely looking for component data and need to be re-read for

INTERACTIVE MODE :

Data Reader :

PEETREADER File a, File c, LIB :user number.File e

Data Updater :

PEETUPDATER File a, File c, LIB :user number.File e

BATCH MODE :

Data Reader :

BATCHREAD File a, File b, File c, File d, LIB :user number.File e

Data Updater :

BATCHUPDATE File a, File b, File c, File d, LIB :user number.File e

File a	: Data Dump-File Name.	(Compulsory)
File b	: New Properties Data File Name.	} (Optional Files User-Supplied.)
File c	: New Thermodynamic Data File Name.	
File d	: New Physical Property Data File Name.	
File e	: User's Library of models and prproperties.	

Table 5 - 1 : Commands to initiate PEETPACK.

additional data collection. The second case is necessitated when new data are written in the dump-file, the previous contents of which must first be erased. The data reading programs are free-standing programs of about 24,000 words each. They are separated from the main executive program to reduce the size of each section of program loaded into core thus increasing their efficiency and quick loading. The size of the readers and their slowness is due to the continuous interpretation and conversion of the alphanumeric data input into comprehensible numeric format for the program, (more details are given in chapter 6).

Two data-type commands are available to exit the data reader: EXIT and CALCULATE. EXIT causes the data stored in the reader to be dumped into the dump-file and deletes the reader from core. The MACRO then loads the data echoer which reads these data from the dump-file and prints out a well documented summary of the data input; the run is then stopped. The same procedure is followed if an irremediable error in the data input occurs. The data already input is not lost thanks to two error trapping routines FTRAP and CHECKERR and the user is informed of what data has been stored through the data echo. The echo helps him check the correctness of his inputs should a simulation run fail at a later stage or should he suspect wrong data input.

The CALCULATE command may cause the reader to create its own model calling routine (EQCALL) if the "industrial" version (section 10.6) is used, otherwise the data are immediately

dumped and the reader is deleted from core. The executive is then loaded into core with all the necessary models and property routines and its peripheral assigned to their appropriate files. The executive's only card-reader is connected to the dump-file to read the "digested" simulation and control data. Its first line-printer peripheral is connected to a log file to receive all the printout information from all parts of the package thus freeing the teletype terminal from long and slow printouts (in interactive mode) and to create a more permanent record of the run. A second line-printer peripheral is connected, in interactive mode, to the teletype terminal to inform the user of the end of a simulation run.

The executive reads the data from the dump-file and calls each unit model according to the calculation order. It provides it with its input streams descriptions and its parameters and the model computes the heat and mass balances and returns the output streams information and the parameters to the executive program for re-storing pending the unit's next calculation, if it is part of a recycle loop solved by iterative calculation, or pending the streams printout at the end of a run. For units partaking in a loop, the relative change in the description of all their output streams over two consecutive calculations is tested against a present tolerance to check for convergence, and the loop's iterate streams (those initially guessed to start the calculation) may be manipulated by one of two convergence promoters to accelerate the run. The loop calculation is repeated until either convergence

is reached or the limit on the number of loop calculations is exceeded, then the next set of units is calculated. At the end of the run, the final stream description is printed in the log-file along with any comments, messages or warnings from the models and the complete set of data are sent to the dump-file following its clearing. Should an error abort the run, the error trapping routines save the stored data and send them to the cleared dump-file and the data echoer is loaded instead of the executive to issue a data echo of the state of the final data.

Because of the slowness of ICL computers, users are not advised to run the executive program interactively. At the end of an interactive data input session, they should EXIT and command the CALCULATE phase from cards.

Following a simulation run, the data updaters are loaded in core automatically by the MACROs to read updating or correction data commands, which in batch-mode may be added in the deck of cards following the CALCULATE command. The data updaters are the same programs as the readers except that a switch set in the MACRO's causes their initialisation section to be by-passed and the dumped data to be read-in. Then peripheral assignments are identical to those of the readers, except for an added card-reader to which the dump-file is assigned. In the interactive mode the question-and-answer input dialogue is no longer available and the user must state explicitly the data he wishes to change,

as in batch-mode, through very explicit commands (see chapter 6). Otherwise the updaters' responses are similar to the readers. A call to the data reader instead of the updaters would cause the eradication of all dumped data and an unnecessary restart at zero.

A successful simulation run can only be terminated by an EXIT command since otherwise the updaters are always reloaded to accept new data changes.

Different simulation cases can be run successively or mixed in sequence by assigning to each case a different dump-file title.

A simplified flow diagram of the package structure is shown in figure 5-1 and table 5.2 summarises the sizes of the various segments. A novel feature introduced in PEETPACK, a data reconciliation routine (MINIMAX), will be discussed in chapter 12 exclusively.

#### 5.4 The two types of executive programs

Two different approaches have been followed in preparing the executive programs and the data readers have been consequently slightly expanded.

The first approach is geared towards academic institutions where users have access to the executive and to the data readers. In this program a restricted number of model routine names

Segment Name.	Approximate Size in ICL-1900 Words	Approximate Number of Statements
INTERACTER	24000	1840
BATCHER	24000	1510
PEETPACKSUBS	7000	85
EXECUTIVESUBS	15000	875
MODELSUBS	19000	1475
THERMOSUBS	10000	850
PHYSUBS	15000	760
ECHOER	8000	210
PEETREADER	-	85
PEETUPDATER	-	85
BATCHREAD	-	115
BATCHUPDATE	-	115
INDPEETREAD	-	110
INDPEETUP	-	110
PROPTS	-	900
PPTEXTRA	-	1400

Table 5 - 2 : Segments sizes.

(twenty) are stored and have appropriate CALL statements in the executive (see chapter 8). Models exist for 15 of them in the MODELSUBS library of models and the remaining 5 names UNIT 1, UNIT 2, . . . . , UNIT 5 are for user-provided models which can bear only these names. To allow for other names or more routines, the user should alter the VNAME array in the readers and correct the EQCALL routine (see section 10.6) accordingly, then recompile the readers and the executive, in the presence of the libraries and new routines, before calling the MACRO's which operate on the compiled programs. The recompilation may be initiated by the MACRO's if these models are set in a LIBRARY (section 5.6.5).

In Industry, however, or wherever the executive is not easily accessible, the second approach is recommended. Here the data-reader creates a routine containing the sequence of CALL statements which it appends to the executive and the MACROs recompile the executive in the presence of all the LIBRARYs before the calculation phase, thus overwriting the available EQCALL routine. The routine is created only when a new flow diagram is input (in the reader) or when the existing one is modified (in the updater). This approach is longer than the first one because of the on-line compilation, but is much more flexible as it allows the user to incorporate his own models to which he can give any name (including those of PEETPACK models, hence overwriting them) and include as many models as he wishes.

The effect of this second approach on the data-communicators is that the recognition of the model names input in the data is not performed but that a new file creation routine must be accessed. To avoid writing two different sets of data-readers, the new features are incorporated in the programs for both versions but a switch in the MACROs is set or left unset to indicate whether these new features should be operated or not. The MACRO's are different to account for the creation of the file and its compilation but otherwise no change is needed. To incorporate his own LIBRARY of models the user need only specify its name on the implementation command as described next.

### 5.5 Initiating a run on PEETPACK

PEETPACK is very easy to use despite its apparent complex structure. The MACROs do all the control and sequencing of the different parts and the user needs only to call the appropriate MACRO by stating its name on a card or on a teletype line and declare the external files the programs need to access. Table 5.1 lists the four commands available and the five parameters that may follow the MACRO name are:

File a: This first parameter separated from the MACRO name by one or more blanks in the name of the dump-file into which the interpreted data will be stored for access by other parts of the package. This parameter must always be quoted.

Different simulation cases may be assigned different dump-file

names of up to 9 alphanumeric characters each.

File b: This parameter is the name of an optional data file containing physical constants for property functions provided by the user for his models. The order in which the data are introduced in it is left to the user, however the format of the file must coincide with those of PROPTS and PPTEXTRA (section 9.14). Additional components above those provided in PEETPACK may be used if the following files are supplied.

File c: is the name of a file containing physical data identical in nature to those in PROPTS but for additional compounds.

File d: The name of a file containing additional data similar to those in PPTEXTRA. In batch-mode this file must be created and declared even if only the thermodynamic data are needed.

File e: The name of a LIBRARY containing the user's semi-compiled models and property routines.

Files b to e are optional. Files not required are not specified and their parameter is left blank, but should a subsequent parameter need be specified, an unset parameter location must be indicated by including a comma as in:

```
BATCHREAD DUMPNP, MYDATA,,, LIB:ECPO750.MYMODELS
```

Once the command is issued the data communicator is loaded in core and the user may input or update his data ending with either EXIT or CALCULATE which is the only way of loading the execution, models and properties into core. A run may only be stopped by an EXIT command, and may be resumed at any time

by calling one of the two updaters.

## 5.6 Major implementation drawbacks in PEETPACK

The creation of an elaborate package such as PEETPACK, with its wide assortment of peripherals, its large dependence on stored data files, the flexible connections between its many sections, the run-time generation and compilation of new routines, its very simple alphanumeric data input, etc., requires unfortunately a large use of compiler facilities and Extended Fortran features, both of which are usually computer-dependent rather than High-Level Language dependent. In PEETPACK some ICL 1900-series computer facilities have been extensively used. These will create some problems when PEETPACK is implemented on computers of different makes, but this is unavoidable. These problems can be classified in five categories.

1. The need for large-core computers.
2. The use of Extended Fortran facilities.
3. The use of Compiler Library routines.
4. The dependence on Job Control Language commands (MACRO).
5. The creation of Libraries for model and property routines.

The following analysis of these problems should help interested users implementing PEETPACK on different computers.

5.6.1 Size Problems: The large size of the executive program and its associated models and property routines (table 5.2) compels the use of computers with more than 32,000 words of free

core store for running overlaid programs or 60,000 words for unoverlaid programs. 32 K computers are not suitable because their free storage capacity is only about 20 K words. For small or medium size computers a number of short simulation programs (GEMCS, ESSPROSS, UNICORN....) exist, and it was not thought necessary nor useful to create a very simplified version of PEETPACK which would only resemble the above-mentioned programs and lose all its advantages and new features.

5.6.2 Extended Fortran facilities: The main extended Fortran facility used is the Free F-Format (FO.0) which allows the user to read any number, with or without a decimal point, anywhere on a data card. This simplifies the data handling for both the user and the computer. Another ICL facility is the use of variable names of up to 8 characters long. These were used throughout the package to make certain variable and routine names more explicit and to facilitate the copying and comparison of command and routine names.

5.6.3 Compiler library routines: A number of compiler software routines have been used extensively in PEETPACK. Their role is mentioned below but the reader is referred to the ICL Manuals (68) for detailed descriptions.

1. Routine FTRAP and CHECKERR: They prevent execution errors from stopping a run completely and allow the user to initiate alternative procedures or issue appropriate error messages. Here when an error occurs the stored data are dumped on

file and a data echo issued before the run is stopped, unfortunately the program cannot correct the error.

2. Routines COPY, COPY8, COMP, COMP8: These routines are used to copy or compare strings of up to 8 alphanumeric characters long (one-word). They are used in interpreting the alphanumeric data input.
3. Routine DEFBUF: It acts as a card-image store into which a card's contents are introduced by a READ statement and from which it can be re-read as often as necessary in PEETPACK to locate and read alphanumeric and numeric fields.
4. Routine SSWTCH: Switches set in the MACROS can be tested in the programs and action taken according to their setting. They are used in PEETPACK to operate the data communicators either as readers thus going through the initialisation section or as updaters hence by-passing this section and reading the dumped data. They are also used to activate the EQCALL routine creation in the Industry-orientated version.

These routines are used exclusively in the data communicators but the error trapping routines are also included in the executive, though.

5.6.4 The MACRO programs: The correct sequencing and file assigning of the various segments making-up PEETPACK is controlled by four programs (MACRO's) written in ICL GEORGE 3 Job Control Language. They will have to be rewritten for any installation

not using GEORGE 3 or 4 compilers however the present MACRO's are very simple and very clear and their conversion should pose no problems. The MACROs are indispensable to the easy and efficient running of the package otherwise all the model and property routines would have to be carried in the program at all times and the physical property data stored entirely in huge and useless arrays.

5.6.5 LIBRARY's of routines: To save compiling all the models and property routines every time a new routine is added, the standard routines are set in semi-compiled form in LIBRARY files. When a new routine is compiled with the very short MASTER program, the consolidator selects from these LIBRARY's the routines it needs for the simulation case and the new program is created. Valuable compilation time is saved by having the routines already compiled. The format of the programs needed to create a LIBRARY is illustrated in figures 5-3 to 5-6.

These drawbacks are serious enough to weaken a user's interest in implementing PEETPACK on different computers, but they are all unavoidable and the facilities used are indispensable to the smooth and efficient running of the package as a whole.

## 5.7 Conclusion

The full range of facilities offered by the package and its flexibility will become more apparent as its various sections are described in the following five chapters and their application illustrated in chapter 11.

```

UAFORTRAN PROG MODELS,OWNPD,NORUN,EXIT
CE MODELSUBS(*DA,BUCK 1,KWOR 10)
UADISCSUB *CR,EDLIB MODELSUBS,*ED !
  CDF      DISC,0,0,0
  IDF      DISC,0
  IS       0,COMPRESR,1,ED
  IS       0,DEG,1,ED
  IS       0,EVALUE,1,ED
  IS       0,DISCOL,1,ED
  IS       0,FLASHER,1,ED
  IS       0,HEATER,1,ED
  IS       0,HEATEXCH,1,ED
  IS       0,MIXER,1,ED
  IS       0,PUMP,1,ED
  IS       0,PURIFYER,1,ED
  IS       0,REACTOR,1,ED
  IS       0,SETBP,1,ED
  IS       0,SETDP,1,ED
  IS       0,SPLITTER,1,ED
  IS       0,SPLITMIX,1,ED
  IS       0,VALVE,1,ED
  IS       0,MASSMIX,1,ED
PRT
F

```

Figure 5 - 3 : The "MODELSUBS" LIBRARY creation program.

```

UAFORTRAN PROG SUBS,OWNPD,NORUN,EXIT
CE THERMOSUBS(*DA,BUCK 1,KWOR 10)
UADISCSUB *CR,EDLIB THERMOSUBS,*ED !
  CDF      DISC,0,0,0
  IDF      DISC,0
  IS       0,RKDATA,1,ED
  IS       0,VAPRES,1,ED
  IS       0,BOILPT,1,ED
  IS       0,HDATA,1,ED
  IS       0,TOSI,1,ED
  IS       0,TOBRIT,1,ED
  IS       0,TOMASS,1,ED
  IS       0,TOMOLE,1,ED
  IS       0,KVALUE,1,ED
  IS       0,DEWBUB,1,ED
  IS       0,FLASH,1,ED
  IS       0,ENTHALPY,1,ED
  IS       0,HSAT,1,ED
  IS       0,TEMP,1,ED
PRT
F

```

Figure 5 - 4 : The "THERMOSUBS" LIBRARY creation program.

```

UAFORTRAN PROG PHYSSUB,OWNPD,NORUN,EXIT
CE PHYSUBS(*DA,BUCK 1,KWOR 10)
UADISCSUB *CR,EDLIB PHYSUBS,*ED !
  CDF      DISC,0,0,0
  IDF      DISC,0
  IS       0,DENSTY,1,ED
  IS       0,SPHEAT,1,ED
  IS       0,VISC,1,ED
  IS       0,CNDVTY,1,ED
  IS       0,SURFTENS,1,ED
  PRT
  F

```

Figure 5 - 5 : The "PHYSUBS" LIBRARY creation program.

```

UAFORTRAN PROG EXECUTIVE,OWNPD,NORUN,EXIT
CE EXECUTIVESUB(*DA,BUCK 1,KWOR 10)
UADISCSUB *CR,EDLIB EXECUTIVESUB,*ED !
  CDF      DISC,0,0,0
  IDF      DISC,0
  IS       0,DREAD,1,ED
  IS       0,EQCALL,1,ED
  IS       0,TEST,1,ED
  IS       0,PUTSTM,1,ED
  IS       0,PROMOT1,1,ED
  IS       0,PROMOT2,1,ED
  IS       0,RESULT,1,ED
  IS       0,DPRINT,1,ED
  IS       0,LOADST,1,ED
  IS       0,COMPUT,1,ED
  IS       0,MINIMAX,1,ED
  PRT
  F

```

Figure 5 - 6 : The "EXECUTIVESUB" LIBRARY creation program.

## CHAPTER 6

### THE DATA INPUT AND OUTPUT FACILITIES

The input and output of data in a flowsheeting program are its most frequently used facilities and thus all users must learn and understand these features properly before using a simulator. The first impression they make on a user will normally affect his subsequent attitude towards the simulator. From the author's experience these facilities are the most vital link in the program and the one on which most opinions on flowsheeting are based. This stage has also proved to be, in the author's and his supervisor's experiences, a major obstacle in learning and teaching the use of simulators and a major source of errors in case runs. The obstacle in using some flowsheeting programs efficiently stems from either the need to provide large quantities of often unnecessary data or from the rigid format and sequencing of these data.

The next important feature in data communicators is their mode of operation. In Chapter 4 an analysis was made of the requirements of different users belonging to different categories. The necessity to provide the correct mode of data communication for each class of user cannot be overstressed, however the advantage of providing them with the alternative mode of data communication is also valuable as many computer installations are either over-worked, thus offering slow or restricted interactive-mode communication, and others are efficient

or under used, thus allowing students the possibility of updating their case-data interactively in short sessions. In both cases the availability of the alternate mode is most profitable and the simulator should be written so as to accept both modes of data input without imposing on either one the requirements or restrictions of the other mode.

The data communications in PEETPACK have been given particular attention to overcome these practical problems. PEETPACK offers two separate data communicators each designed to operate efficiently in its own mode. In either of the two programs the user inputs the essential minimum data in very free numeric and alphanumeric formats and in a fairly flexible sequence. The extensive use of backstore files allows him to input all his data once only, in either mode, and then access them again, using either communication mode, to update or change any part of them for subsequent runs without having to re-input them all again. The two communicators have been designed as two separate free-standing programs each responding to its own set of input commands. At the end of a data input session, they "dump" the "interpreted" data onto a dump-file which is then accessible to the main simulator executive or to the two data communicators or to the data-echoer, all programs having identical data retrieval sections. The need to interpret the input data arises from the form in which the user is allowed to input them. A number of restrictions inherent in the older or less developed programs have been waived. In PEETPACK units and streams may be

given names of up to 8 alphanumeric characters long. Data recognition is effected through the use of title cards or identification tags which are recognised by the programs and the correct data-interpretation section entered. In Interactive-mode the user is led through the data-input by the program though he may choose to input his data in his own sequence by exiting fairly soon in the input session from the data reader and entering the updater which does not contain the question-and-answer data input facility; but a direct entry to the updater for data initialisation is not permitted.

PEETPACK requests from the user the same data as do most flowsheeting programs, these are of two types:

Type 1: Process layout and operating conditions. These data include the process flow diagram (PROCESS MATRIX), the operating conditions for each unit (PARAMETER MATRIX), the physical description of the feed streams and possible iterate streams (STREAMS MATRIX), and the chemical COMPONENTS NAMES used or generated in the process. These data are classified as Set-A commands in figures 6-1 and 6-2.

Type 2: Simulation control data (Set B and C commands). The course of the simulation is controlled by the specification of the sequence in which the units are to be calculated (ORDER OF CALCULATION). This may be done either automatically by the data communicators or may be input manually as explained later. Model routine

A- Section - Tilde Commands :

PROCESS MATRIX  
PARAMETER MATRIX  
ROUTINES NAMES  
COMPONENTS NAMES  
LIBRARY COMPONENTS  
ADDITIONAL COMPONENTS  
NEW COMPONENTS  
STREAMS MATRIX  
PREFERRED STREAMS  
ORDER OF CALCULATION  
VARIABLES NAMES

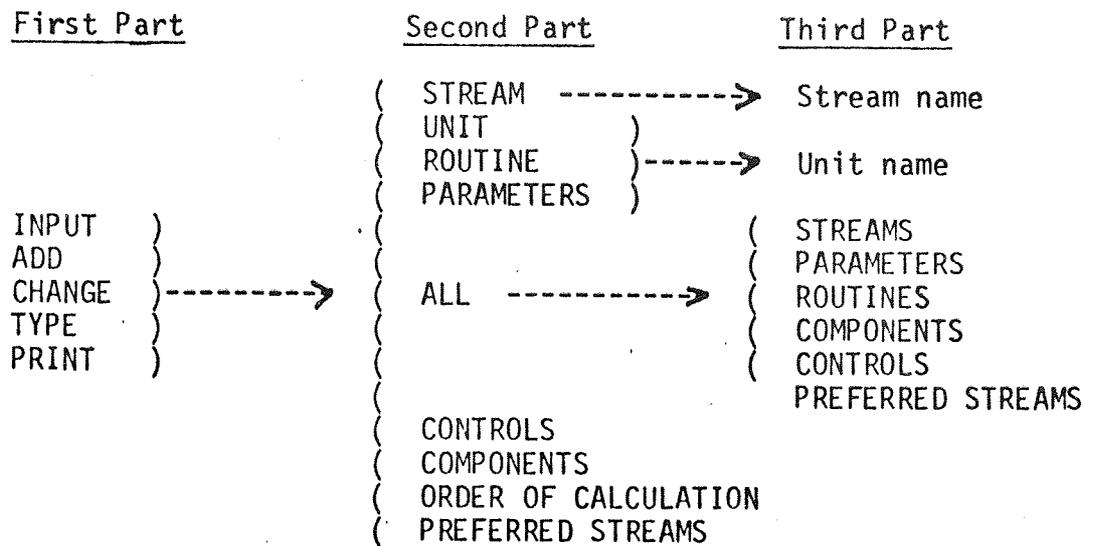
B- Data Specification Followed by Data Value :

	<u>Default Value :</u>
ACCELERATOR i	0
LOOPS l	0
TOLERANCE t %	0
TRACE n	0

C- Commands requiring no additional data :

BRITISH UNITS  
S.I. UNITS  
NOACCELERATOR  
NOTRACE  
NOORDER OF CALCULATION  
DATA ECHO  
KEEPON  
NOGO  
CALCULATE  
EXIT  
END

A- Commands for Data Access and Alteration :



B- Data Specification Followed by Data Value :

	<u>Default Value;</u>
ACCELERATOR i	0
LOOPS 1	0
TOLERANCE t %	0
TRACE n	0

C- Commands requiring no additional data :

BRITISH UNITS  
 S.I. UNITS  
 DATA ECHO  
 KEEPON  
 NOGO  
 CALCULATE  
 EXIT  
 END  
 YES / NO

Figure 6 - 2 : Interactive-Mode Data Input Commands.

names must also be assigned to each unit (ROUTINES NAMES). To check the convergence of the simulation calculations a TOLERANCE must be set, and the maximum number of LOOPS allowed for each iterative sequence calculation controls the length of a run. Two ACCELERATORS are available to promote convergence (section 10.9) and a flag (KEEPON/NOGO) decides upon the course of action to take should a loop fail to converge in "LOOPS" number of loop iterations. PEETPACK offers also a dual system of units for data communication between the user and the program, BRITISH and S.I. UNITS.

The names in capital letters above are the commands understandable to the communicators as presented below. The sequencing of the data is discussed later. Unless otherwise specified all alphanumeric data may be punched anywhere on a card with at least one blank space between successive strings of characters (blanks not being accepted as characters). Blank cards are ignored except in the process matrix specification. In most cases names may be given on one or more cards. This free-alphanumeric READ format is described in section 6.6. The numeric data are input in free F.0.0 format anywhere on the cards.

#### 6.1 The Batch-mode data input:

The initial data input is initiated by the command:

BATCHREAD followed on the same card by the various file-name declarations as described in chapter 5, and this command must always be followed by a case-study title occupying not more than one card. Data updating is initiated by the command BATCHUPDATE but no title card follows it. Then comes the data cards each section of which is either preceded by a section-title card (Set-A commands in figure 6-1) or by an introductory identifier (Set-B commands) or the identifier itself is the data information (Set-C commands). The commands tabulated in figure 6-1 are the only words the program will recognise and take action upon.

#### 6.1.1 Set-A Commands:

a. The PROCESS MATRIX command card should be followed by a number of sets of 2 cards each, equal to the number of process units. The first card in each set should show a unit name, followed on the same card by all its input streams (all on one card only). The second card should give the names of the output streams. The streams order should be as expected by the unit model (see chapter 8). A maximum of 6 inputs and 6 outputs are allowed per unit and 30 streams and 30 units per process. The program creates two library arrays the first, ENAME, contains the unit names and the second, SNAME, the stream names. Each name is represented or referred to internally by its position in these arrays and used thereafter only when communicating with the user. From this input data the program creates the process matrix which shows in each row the unit number followed by its input stream numbers and its output stream numbers bearing a

negative sign for differentiation. This matrix is used in finding the calculation order and assigning the input and output streams to each unit calculation.

b. The PARAMETER MATRIX command is followed by sets of cards each bearing a unit name, punched from column 1, followed on as many cards as necessary by its parameters values in the order requested by the units. There is no fixed format or number of data to be entered on each card but each unit is allowed a maximum of 30 parameters. Units with no parameters need not appear in this list. The parameters are stacked head-to-tail in a cumulative array, EEN, and a matrix, NEMAT, indicates the starting point and length of each unit parameter array in EEN.

c. The ROUTINES NAMES command is followed by one card for each unit showing the unit name followed by its model name. The routines names are stored in array RNAME and array MODULE contains a reference number to each model according to the order of CALL statements in routine EQCALL and obtained by comparing each routine name with the acceptable names given in array SMODN in the program. However when the industrial version of PEETPACK is used MODULE is not created since EQCALL is created according to the process flow diagram.

d. The COMPONENTS NAMES command is followed by these names which must conform to those in table 9.1.

To differentiate between the PEETPACK library components and others specified by the user, four different commands exist.

COMPONENTS NAMES and LIBRARY COMPONENTS refer to PEETPACK's

components, ADDITIONAL COMPONENTS refers to new components for which the user is introducing data identical to those in data banks PROPTS and PPTXTRA. NEW COMPONENTS precedes new component names for which the user specifies his own data.

e. The STREAMS VALUES command is followed by a set of cards each showing a stream name, starting in column 1, and followed in free-format by its description in the order. Molar flow rate, temperature ( $^{\circ}\text{C}$  or  $^{\circ}\text{F}$ ), absolute pressure (bars or p.s.i.a), enthalpy (usually set to 0 and calculated by the program in K Joule/Kg mole or b.t.u./lb.mole). The vapour fraction and the components' mole fractions in the order the component names were entered. The number of streams specified is left to the user, unspecified streams are zeroed in all their descriptors.

f. The PREFERRED STREAMS command is followed by their names and used to help create the calculation order (see chapter 7).

g. The ORDER OF CALCULATION when not obtained automatically is introduced following this command as follows: Each set of sequential and recycle units is introduced in turn starting on a new card with either the word SEQ or the word LOOP thus designating a complete sequence or set of nested loops to be calculated before another set is met. The sequence may extend on an unrestricted number of cards. It is stopped either by the appearance of a new command or of one of the two words SEQ or LOOP.

h. The VARIABLES NAMES command is used when the user wishes to change the stream descriptors names. These names may be up to 16 alphanumeric characters long.

### 6.1.2 Set-B Commands:

- a. ACCELERATOR *i* indicates which of the two available convergence promoters to use (*i* = 1 for Wegstein-type, *i* = 2 for Orbach-type promoters; see chapter 10). The default value of *i* (flag KONVRG) is 0, no promoter set.
- b. TOLERANCE *t* gives a value to the tolerance variable TOL, "*t*" should be read as a percentage figure.
- c. LOOPS *l* sets the maximum number of loop calculations, NLOOP allowed for each recycle loop unit sequence.
- d. TRACE *n* sets the trace flag ITRACE. For *n* = 0 all input and output streams for each unit are printed at each unit calculation. For *n* greater than zero the complete stream and parameter matrices are printed every *n* loop calculations for each set of recycle loops, but no trace is given for sequential sets. *n* = -1 suppresses all tracing.

### 6.1.3 Set-C Commands:

- a. BRITISH UNITS and S.I. UNITS convert the thermo-physical property data to the requested system of units, and the flag IUNITS is set to 0 or 1 respectively to control the properties calculations in the executive. Once the executive is entered the unit system may not be changed except by re-specifying the components names in the updater then resetting the unit-system type.
- b. KEEPON and NOGO control the simulation should a recycle loop fail to converge in NLOOPS. KEEPON (flag NOGO = 0) sees that

the last loop calculation results are accepted as "converged" answers and the simulation continued. NOGO (NOGO = 1) aborts the run and causes the loading of the data echoer.

c. NOACCELERATOR, NOTRACE, NOORDER OF CALCULATION suppress these facilities and set: KONVRG = 0, ITRACE = -1, NLIST = 0 respectively.

d. The DATA ECHO produces an echo of the stored data at the time of command and may be used at any time. It is most useful just before a CALCULATE command since the ECHOER is not loaded and the user has no hard copy of the accepted input data except his cards. This command activates a series of WRITE statements to produce a well-documented output.

e. CALCULATE and EXIT activate the next stage in the simulation. CALCULATE initiates a data dump and causes the MACRO to load the executive. EXIT initiates the dump then causes the MARCO to load the ECHOER to produce a data echo and the run is stopped.

f. END has no function at all. It is allowed only for the user's convenience in ending data-sections input.

## 6.2 The Interactive-mode data input.

Both mode data communicators accept similar data and have fairly similar commands for data input. The major difference between them is that for the most part of the initial data input the user is led through the input steps by the interactive program in a series of questions and answers to ensure that no information is missing or misplaced.

The questions the program asks take the place of the section-title cards in the batch-mode program and the user answers them in free alphanumeric or numeric format. Once the explicit question-and-answer section is over, indicated by the appearance of the first: COMMAND?, all the simulation data should be available to the program. The user may then EXIT, CALCULATE or examine (TYPE-PRINT) and update (CHANGE, ADD, INPUT) his input data. From this point onwards the data reader and the updater are identical. They accept data commands in response to their question COMMAND? and to which there are three sets of data answers illustrated in figure 6-2. Sets B and C are introduced as answers to COMMAND? and have the same effect as in batch mode. Set-A commands are more elaborate in that not only are input or updating commands available, but data retrieval and typing commands are also given. These allow the user to check the input data on the simulation results visually before or after updating them or following a successful run for which he has not yet received the log-file printout. The TYPE command types the information he asks for on the teletype terminal. PRINT sends the same data to the log file to be subsequently printed on line-printer. These two commands are made-up of two or three parts depending on the data requested. Each of the first four "second-parts" must be followed by a stream or unit name to fully define the command, the fifth one presents all the data available under the third-part heading. The five following second-parts are self sufficient requests to follow the TYPE or PRINT command.

INPUT and ADD commands act identically on their second and third parts. They both add information to the data contents of the program. To INPUT or ADD a STREAM implies giving its description in response to explicit requests from the program for each descriptor's value. To INPUT or ADD a UNIT means extending the process flowdiagram by an extra unit of the given name and for which the program asks for the input and output streams names which must be input on one and only one line for each set of streams as in batch mode. To INPUT/ADD a ROUTINE for a unit is to declare the routine to use for modelling it. All the other INPUT/ADD commands are similar to the batch-mode Set-A commands. The CHANGE command is similar to INPUT or ADD except that it implies changing a value already available in the program rather than inputting a new piece of information. For most of these data inputs, the user is guided through the input of the relevant information by simple questions identical to those asked for in the initial data input section.

Three remarks should be made about the interactive data input:

1. Since the program is in constant communication with the user, it can detect and reject most common typing errors and request the user to retype wrong inputs thus avoiding erroneous data input such as unrecognised unit and stream names or mis-spelt commands.
2. Should the user realise at any time during the process matrix input that he has set wrong stream connections, he

may retype the unit name and update its connections thus overwriting his previous error. Similar mistakes may also be corrected by the updating commands.

3. During a parameter changing session, the program enquires about the parameter to change. The user may either alter existing ones or add new parameters by typing higher parameter numbers. These will be automatically registered and the parameter array size increased accordingly. To terminate this input session the user should type a zero for the next parameter to update.
4. The interactive communicator accepts less external file assignments than the batch-mode communicator (Table 5.1) and the user should be aware of this limitation imposed by the slowness of present ICL computers. This limitation has the advantage that for runs not involving PPTEXTRA-type data (see chapter 9), the user does not have to create such a file for his own components, as required in the batch-mode data input.

### 6.3 The order of data input.

The sequence in which data are input to the batch-mode data reader/updater or to the interactive data-updater is to a certain extent immaterial. The data and information communicated via the Sets-B and C commands, in figure 6-1 and 6-2, except for CALCULATE and EXIT, set certain flag or variable values and are independent of the rest of the data. As long as commands are

not imbedded within Set-A data actions, they will be recognised upon input and action taken immediately. The program is written in such a way that almost every new card read-in is checked for a new command by the first level data recognition check point (described in section 6.4). Thus sets of data input following Set-A commands need not terminate by any particular identifier, although an END card is allowed as terminator for the user's satisfaction. Hence each of the data sections input by a Set-A command is automatically terminated by the recognition of a new command. This procedure is marginally longer than checking for the presence of a single section terminator and users are not allowed to use command names as stream, unit or component names, but it is less subject to errors from terminator omission, though a misplaced command within a Set-A data section will create confusion and abort the run. Set-B and C commands may be inserted before, among or following Set-A commands in any chosen order.

Set-A commands will be recognised anywhere in the sequence but unless certain other Set-A type data are already input, some of the new data may not be recognised. The data set of prime importance is the PROCESS MATRIX which defines the units and streams present in the plant. Following its input, the PARAMETER MATRIX, the ROUTINES NAMES and the optional ORDER OF CALCULATION and PREFERRED STREAMS may come in any order. The STREAMS VALUES however can only be set following the declaration of the COMPONENTS NAMES. The recommended input sequence is the one shown in figure 6-1 in the order Set-A, Set-B then Set-C

commands. The same ordering applies to the interactive updater although once the data are input through the reader, their updating is usually minimal, but any flow diagram and component name updating must nevertheless precede any stream, parameter or routine name updating.

In the interactive-mode data input, the program leads the user through the input to reduce confusion, but data checking and correction using the updating commands as in the updater is also possible from the reader after a complete data input.

#### 6.4 The CALCULATE Stage:

All data input commands control the setting of a limited number of variables. The CALCULATE command, which is the last command, does more than command the MACRO to load the executive and initiate the simulation. Three major steps must first be fulfilled before any calculation can start.

1. The gathering of the property data from the data banks.
2. The creation of the routine calling sequence (EQCALL).
3. The dumping of the data for access by the other parts of the package.

These steps are initiated by CALCULATE as follows.

##### 6.4.1 The gathering of property data.

In batch mode, a PAUSE in the communicator causes the MACRO to assign certain card-readers to the beginning of the

data-bank files. The first component name in the files is read and compared to the input components names. If it is one of them, the data from all the data banks following this name are read into their appropriate arrays and the next component name is read-in. For components not used in the simulation, their properties are read only as a way of passing over them.

Upon completion of the data gathering, the type-of-unit system flag is checked and for S.I. units all the data (in British units) are converted to S.I. by suitable conversion factors.

In interactive mode this data gathering is effected in a similar way but at the time the components names are introduced to detect unrecognisable names.

Data gathering is effected every time the components are changed, and they must all be re-input even when only one component name is changed.

#### 6.4.2 The Routine calling sequence.

In the industrial-type communicators, the EQCALL routine, which contains the sequence of CALL statements to access the models, is created just before dumping the data. The steps in its creation consist merely in a series of WRITE statements writing a series of CALL statements one for each unit followed by its model name and a RETURN statement as described in chapter 10. This routine is created every time a process matrix is input or updated.

### 6.4.3 The data dump.

At dumping, the data are summarised and condensed into a minimum number of arrays and matrices containing the necessary and sufficient data to perform the simulation and into a number of single variables that define the size of these arrays and variables and that control the run.

The matrices in their dumping order are as follows.

- a. PROPRT, PPTXTRA and USERPPT referring to the thermo-dynamic, the physical and the user-provided data stores respectively. Each row corresponds to one component in the plant and bears the data read from files and introduced manually.
- b. KPM, the process matrix: Each row represents a unit in the plant and bears its internally generated sequence number, input streams (positive numbers) and output streams (negative streams).
- c. KSEM, the stream connection matrix: Each row shows a stream number, its source unit number and its destination unit.
- d. NEMAT, the parameter control matrix: Each row shows the starting point of each unit's parameter array in the cumulative parameter array EEN and the number of parameters assigned to the unit.
- e. NELIST, the calculation order matrix gives the order of the units by their number sequence, in the first column, and the type of sequence to which they belong in the second column.

The arrays in their dumping order are:

- a. SNAME, ENAME, RNAME give the streams, units and models names used in the plant. The position of each name in the arrays is its internal numerical representation.
- b. VNAME contains the alphanumeric description of each stream value.
- c. MODULE contains the number of the CALL statement associated with the model representing each unit.
- d. SMATRX, the stream matrix, contains the streams descriptions in their order in SNAME stacked head to tail. Unset and uncalculated streams are zeroed. All streams are of equal description length.
- e. EEN, the cumulative parameter array, contains each unit's parameter array, stacked head to tail according to the units ENAME order.
- f. KSFLAG, contains the stream flags which help in discerning iterate streams (flag = 4), on which convergence promoters operate, from other streams (flag = 0 or 1).
- g. KPS contains the preferred streams numbers.

The single variables are:

- NEMAX : Total number of units in the plant.
- NSMAX : Total number of streams in the plant.
- NOCOMP : Number of PEETPACK library components used in the plant.
- NSLMAX : Length of the stream array, usually NCTOT + 5.
- NCTOT : Total number of components in the plant.

NPAR : Total number of parameters for all units.

NCOLPM : Number of columns in the process matrix KPM, equal to 1 + the number of input + output streams to the unit with most streams attached to it.

KONVRG : Type of convergence accelerator to use.

NPS : Total number of preferred streams.

ITRACE : Type of tracing requested.

NLOOPS : Maximum number of loops calculations allowed per recycle loop.

NLIST : Length of the NELIST matrix.

NVNAME : Number of stream descriptors (2 words per name).

NPROP : Total number of thermodynamic property data for each component in file PROPTS.

NOGO : Flag controlling unconverged loop calculations.

IUNITS : System of units used.

NUSRPT : Total number of properties per component entered by the user.

NPROPX : Total number of physical property data per component in file PPTEXTRA.

IRT : Flag checking whether model names have been set (= 1) or not (= 0).

A number of other variables are used internally to help the book-keeping of new information, but are of lesser importance and are omitted from this description.

## 6.5 Data recognition:

The description of each section of data recognition in the data communicators would be prohibitively long and is omitted, however, most sections follow one of two data recognition patterns referred to as Levels one and two.

### 6.5.1 Level one of data recognition

This recognition is effected at the initial card-reading or teletype line-reading level and applies to the data input through Sets B and C commands. A card is read in 10A8 format and its contents are stored in a buffer area (BUFFER) created by a CALL to routine DEFBUF. The card-image is then analysed character by character in routine COPYER to separate and count the various sequences of alphanumeric characters (word) it contains and to set it in successive elements of array A, left-justified. Following this classification the first data recognition process occurs. The first word in array A is compared to a number NCOMND of command words stored in the V NAMES array. If this word is recognised as a new command, an assigned GOTO statement directs the action to the appropriate section of the program. For Set-B commands the appropriate sections will read from the buffer area the numeric information it contains in free FO.0 format. For Set-C commands, either a flag is set (1) or unset (0) or the appropriate command (CALCULATE, EXIT, DATA ECHO) obeyed. For Set-A commands however, this first level of recognition directs control to the relevant program section where

additional cards are read. If the first word on the card is not recognised, the program checks whether the previous command was one in Set-A and redirects the data to be analysed in that section or if it is not the case it prints out an error message and either the run is aborted in batch-mode, but without the loss of the previously input data, or in interactive-mode the user is asked to retype his command. A restart in batch-mode would have to be through the update MACRO.

#### 6.5.2 Level two:

This second recognition level occurs in Set-A type data-input sections following the input of the section title card. The card following the title-card or any unrecognised card in level one is sent to the relevant section where it is analysed through routine COPYER for its contents. A card in the PROCESS MATRIX section should contain a unit name followed on the same card by its input streams names, and followed by a second card showing its output stream names thus any word (except a command word) is automatically accepted as a unit or stream name and compared to previously input names to avoid duplication. COMPONENT names are either immediately checked (interactive-mode) for acceptability or stored for later check (batch-mode) as described previously. PARAMETER values are read-in in free-format following the location of the ending of the unit name. The same applies to the STREAMS VALUES. Should a unit or stream name not be recognised the program assumes that the first word

which should have been such a name is actually a numeric field and represents parameters belonging to the previous unit and reads and appends them as such. Thus a wrong unit or stream name would cause the program to fail. PREFERRED STREAMS and the ORDER OF CALCULATION are registered by recognition of the unit/stream/control names. Here again wrong names cause a run failure. In most cases the number of data to read or compare is available following a card analysis in routine COPYER as explained below.

#### 6.6 The Free-Alphanumeric-Numeric READ Format.

Three types of data input have been used for which Fortran compilers offer no READ formats.

1. Reading an unspecified number of alphanumeric strings of characters (Set-A Commands) and counting them.
2. Reading a specified number of alphanumeric strings of undefined length and position (Set-C Commands).
3. Reading as above followed by a numeric field (Set-B Commands).

The major obstacle in Fortran is that blank spaces are acceptable alphanumeric characters, thus making it impossible to define the end of character strings of various lengths. The fixed number of bits per computer-word adds a further complication in that strings over 8 characters long (on ICL 1900-series computers) need more than one word of storage.

A number of rules were set to overcome these obstacles

and simplify the reading procedure with little inconvenience to the user, and the creation of a new compiler FORMAT was avoided to reduce the difficulty of program adaptation to other computers. A special routine COPYER was created to perform all the reading requirements mentioned above in the light of the following rules.

1. All units and streams names must not exceed 8 characters in length.
2. Components and stream variable names must not exceed 16 characters but may be less than 8.
3. Both types cannot be included on the same card.
4. Blank characters are not considered as alphanumeric characters, they act only as field separators.
5. Any non-blank character is accepted in a name.
6. The routine may be used for any other alphanumeric reading requirements as long as short and long words are not mixed or special precautions taken to differentiate between them.

#### 6.6.1 The reading procedure:

Data cards can only be used once thus an analysis of their contents is impossible unless they are first stored in a buffer area. The buffer area, CALL DEFBUF (7, 80, BUFFER) provided in ICL-software (68) was used to store a card-image, one at a time, to allow it to be re-read as a normal card a number of times according to the following procedure.

The card to be analysed is read into array BUFFER in 10A8 Format. The program decides which of the following cases

applies to the specific card and sets N accordingly.

1. An unknown number of strings of characters to be read,  
N = -1.
2. A known number of sets to be read-in, N = number of sets.
3. A known number of sets, followed by one numerical field,  
N = the word-order of the numeric field preceded by a negative sign.

The program then decides whether to store the strings of characters in 8 or 16 bit words (1 or 2 words) and sets NBITS to 8 or 16 accordingly. The command:

```
CALL COPYER (NBITS, A, BUFFER, N)
```

calls the analysing routine which sets into array A (of maximum length 20) the strings of characters in BUFFER, left-justified, one string per one or two positions, blanks the unused bits and returns in N the number of 8 -character strings read, i.e. the number of locations filled in A.

Example: The card.

```
TOLERANCE 1.0
```

may contain any number of blanks before and after the strings TOLERANCE and 1.0. The statements N = -3 and CALL COPYER (8, A, BUFFER, N) will cause the first two words in A to contain the characters: TOLERANC and E (followed by 7 blanks in A(2)). The numeric field is read into position A(3).

The routine cannot however read either mixed sets of alphanumeric and numeric fields or more than one numeric field.

The COPYER procedure is illustrated in figure 6-3.

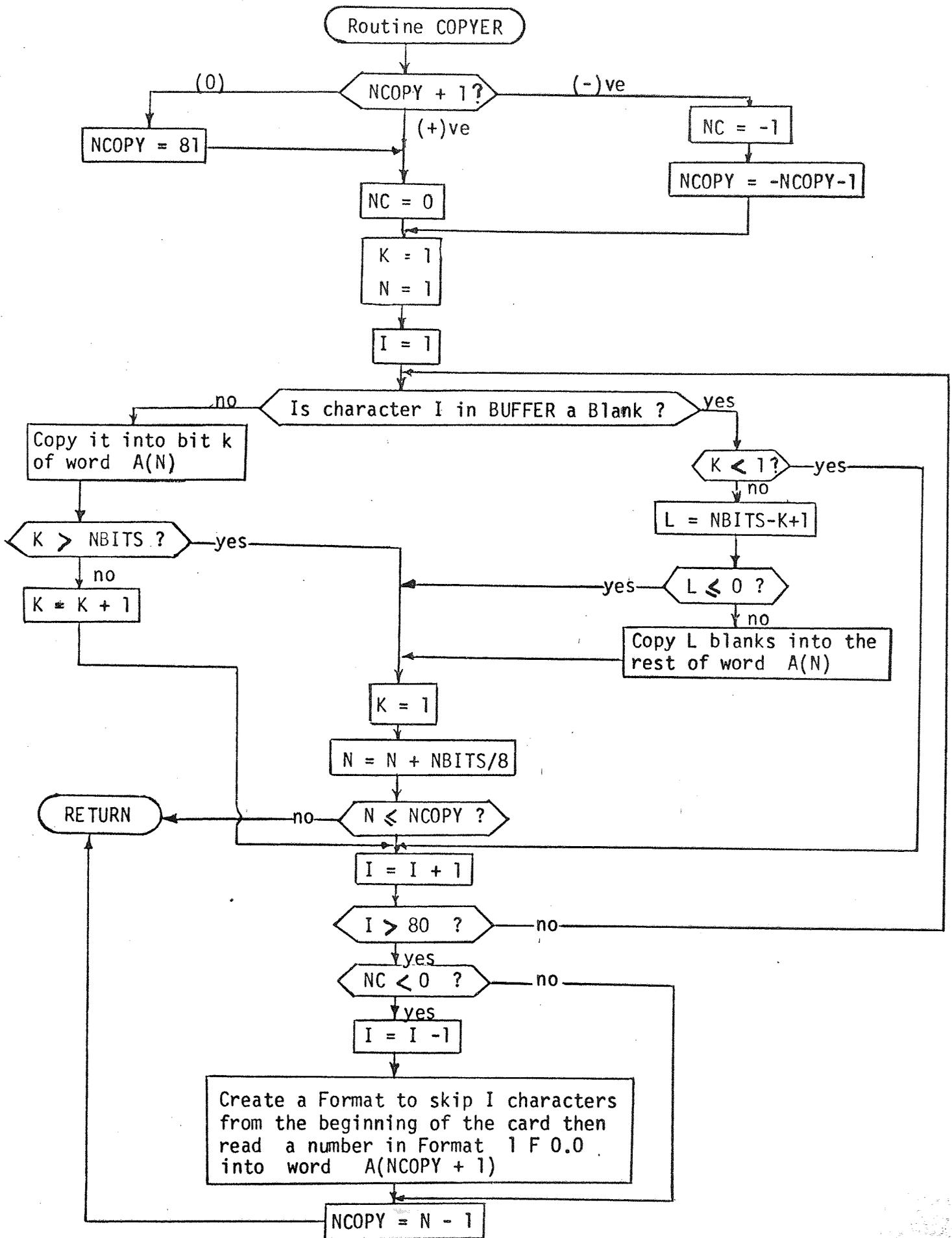


Figure 6 - 3 : Flowchart of routine COPYER.

## 6.7 Conclusion.

The data communicators are very easy to understand and use mainly because of their great flexibility and the need to input a minimum amount of data per run.

Their main drawback is in the use of FORTRAN for the recognition of alphanumeric characters for which the language is not very suitable. The only way to avoid this would have been to use PLAN or other machine-code languages but at the expense of making the communicators more machine dependent and less readily understandable and implementable by most interested users.

## CHAPTER 7

### THE CALCULATION ORDER FINDER

A calculation order finder routine to order the units in a feasible and calculable sequence is almost as important in a chemical plant simulator as the unit model routines it offers. The need for an automatic sequencing program arises in the following cases:

1. Certain users have little time to spend locating a feasible unit sequence, let alone an optimum one, or are not familiar with flowsheeting concepts to understand the requirements in unit calculation sequencing.
2. The calculation order is sometimes difficult to determine in flow diagrams incorporating many nested loops. Overlooking hidden loops in the maze of interconnected units may cause the determination of a poor sequence requiring more stream tearing and guessing to cut the system open.

Moreover the permanent nature of a sequencing routine justifies its development and incorporation in the simulator, as long as it is added as an easily removable block when implementing the program on smaller computers. Unit models on the other hand usually have to be rewritten for different simulation cases.

An optimal calculation order has been defined as the order which minimises the total number of individual guesses needed to start the calculation of a cyclic system (33) since these values must converge on their final answer to obtain the "true" plant

simulation. It has since been redefined in user-orientated flowsheeting program as the order which minimises the number of variables the user has good estimates for at the expense of estimating other necessary but unknown variables (33). By variables it is generally but not always meant the streams in the flow-diagram since all streams usually bear the same amount of information. This definition is not always accepted in flow-sheeting programs where streams contain varying amounts and types of information (69).

To determine the optimum calculation order, the flow diagram of a process is best regarded as a directed graph composed of nodes (the units) and edges (the streams). This facilitates the sequencing since a well-developed branch of mathematics, Graph Theory, exists to analyse such networks and locate loops.

Ordering the unit sequence has been approached in a variety of ways (26) since the early 1960's, but the more useful methods are equivalent to locating the initial streams to guess and following them through the plant to obtain the feasible sequence. Most modern methods include two or all three of the following stages:

1. Locating and extracting the closed recycle loops (partitioning).
2. Searching for the iterate streams to tear the system open (tearing).
3. Ordering the units in a calculable sequence (sequencing).

The evolution and development of unit ordering procedures is presented under these three headings, except for the methods not following this approach, usually older methods, which are described separately. Each section ends with the description of the procedures used in this flowsheeting program for finding the sequence.

### 7.1 The Initial ordering attempts.

The initial work on recycle systems analysis and solution is attributed to Nagiev (16) who describes ways of solving simultaneous sets of equations describing units in loops. His methods are still being used in simultaneous equation approach flowsheeting programs (70) but are of little importance in modular-type simulators where units are calculated in sequence and no restriction is imposed on their mathematical description.

One of the first algorithms for determining the optimum sequence of units calculation by computer was described by Rubin (22) in 1962 . It consisted in preparing a square matrix of connections between the nodes, showing the number of parameters needed to calculate each unit in each position (each row and each column representing a unit and the parameters bearing a positive sign if they represented inputs to the units and a negative sign for outputs). For all possible sequences of units represented by reordering the rows, the sum of the matrix upper triangle was computed and the matrix with the minimum sum provided the

optimum sequence. This long and tedious method was used in one of the early flowsheeting programs (MAEBE) (17) but gained little support later.

The next published algorithm was by Sargent and Westerberg (14) who divided the sequencing problem into a series of operations. They used a tracing method to locate the initial sequential sets of units by working backwards from unit to unit until a plant feed is met without meeting an uncalculated stream. Chains of units reaching a previously added unit formed a recycle loop and were then ordered by dynamic programming. They chose subgroups of these recycle units and permuted the combination of units until they reached a combination requiring the minimum number of parameter estimations (input streams descriptions are considered as unknown parameters), and then added to it the other units in the order that kept the number of parameter guessing to a minimum. To reduce their efforts they devised a way of merging together nodes joined by certain combinations of streams which minimised the number of parameter estimations for these units and the number of unit permutations. Here also the method gained little support because of its apparent complexity and its need for a specialist processing computer language. It did however influence the methods to be published later mainly by its concept of searching for sequential and recycle loops as separate sequences and by its manipulation and merger of units in loops to minimise searches.

Following Sargent and Westerberg's efforts and with the

appearance of new applications of matrix manipulation for loop extraction, the three-step pattern mentioned above emerged as the accepted pattern for unit ordering.

## 7.2 The location of recycle loops (Partitioning)

The location of recycle loops is one of the preferred methods used to help locate iterate streams. Two distinct partitioning methods have evolved over the past ten years:

1. The powers of the adjacency matrix method.
2. The exhaustion path searching method.

### 7.2.1 The adjacency matrix method

The adjacency matrix  $A$  is a square boolean matrix where each column and each row represents a node. A non-zero element  $A_{i,j}$  represents a path from nodes  $i$  to  $j$ . Zero, columns and rows represent initial and terminal nodes. By deleting these rows and columns and any other appearing following these deletions, initial and terminal unit calculation sequences are obtained (71). The reduced matrix represents the remaining recycle loops. This matrix does not differentiate though between the various recycle loops and intermediate sequential sets. By raising it to successive powers  $p$ , its non-zero elements represent units connected by  $p$  number of edges, and non zero diagonal elements represent units involved in loops of  $p$  units. The boolean summation of all matrices of successively higher order (up to order  $p$ ), intersected by its transpose shows all units in loops

of all sizes up to  $p$  (72), but only as "maximal cyclic nets" (all units involved in all interconnected loops).

Maximal cyclic nets provide a good visual aid to locate sets of interconnected loops but have been of limited use in helping locate iterate streams until Kehat and Sacham (26) proposed their iterate stream finding of method (section 7.3). They also use a condensed two column version of the cumulative adjacency matrix (73), the reachability matrix. For the location of elementary loops (all possible closed recycle loops) Norman (71) proposed to observe the diagonal elements in the adjacency matrix raised to the power  $p$  to obtain the units in loops of size  $p$ . However his method has the disadvantage of confusing loops of the same order, and the sequence of these units is not easy to extract as they are also intermingled with sequential sets also of length  $p$ .

The author devised a method to locate all elementary loops of size  $p$  by intersecting  $(A)$  with  $(A^{p-1})$  transposed. The resulting matrix shows only the units involved in all loops of size  $p$ . This is based on the fact that if a path of size 1 is traced from nodes  $i$  to  $j$  meets the inverse (transpose) of another path traced from  $i$  to  $j$  then a loop is met. All units not in a loop of size  $p$  do not meet this condition and do not appear in the intersection matrix. Only recycle units obey this rule since each one obeys it independently. To locate all elementary loops, each connection must be traced through

completely by starting at element  $A_{i,j}$  ( $i$  and  $j$  are both 1 initially,  $j$  is incremented first and then  $i$  until the first non-zero element is met) then going to  $A_{j,k}$  and so on until the  $p$ th connection forces the sequence to return to  $A_{i,j}$ . A path that cannot be completed is rejected. Every time a connection is followed, its element ( $A_{m,n}$ ) is raised in value by 1 and extensions are followed through the row element of lowest value to ensure that no unit is left out of the loops. However to locate all loops it seems necessary to follow-through each non-zero element in the matrix to avoid overlooking paths, but at the expense of duplicating loop traces.

So far there seems to be no satisfactory application of the adjacency matrix method in locating all elementary loops. Even the above mentioned method was rejected in favour of an exhaustive search technique for faster computation time. Kehat and Sacham's application (26) appears as the only successful one of the adjacency matrix idea, but they operate on "maximal cyclic nets" rather than elementary loops.

#### 7.2.2 Exhaustive path searching.

Path searching consists in starting at each node in a directed graph and following its feeds or outputs through all the nodes they meet until the sequence of nodes contains a stream returning to the starting node (recycle loop) or reaches an extremity node (sequential set).

Path searching was used by Sargent and Westerberg to trace the initial sequential sets of units in process diagrams (14). Their search for loops was complicated as they grouped all recycle units together as pseudo-units and did not bother searching for individual elementary loops.

Christensen and Rudd (74) applied a similar search to locate both initial and terminal sequential sets. Intermediate recycle loops were also treated as maximal cyclic sets for which they proposed a different ordering theory (described in section 7.3).

Forder and Hutchison (33) programmed a modified version of Sargent and Westerberg's method to locate the loops for CONCEPT. They do not however group the recycle units into pseudo units, but by leaving the units in the graph, they can trace all paths and hence locate and separate the various loops having one or more common units (elementary loops). Their method seems unduly complicated and difficult to program particularly in the light of the very simple method described in section 7.2.3.

The systematic analysis of graphs by the Mason rule (75) is a different search technique based on a list processing algorithm of enumerating all paths and loops in a network by analysing combinations of an increasing number of nodes and deleting invalid combinations where a path between any two consecutive nodes in the combination does not exist. This method is feasible only when both straight paths and recycle loops are

required and usually those only of a certain length.

A number of other search techniques have been proposed (76, 77, 78) but these are all based on circuit-finding in undirected graphs, which is not the case in chemical processes.

### 7.2.3 The Method used.

The method adopted in PEETPACK is based on a recent very simple but thorough graph search technique developed by Tiernan (79). This theoretically very efficient algorithm also uses the exhaustive search but locates all elementary loops (each individual loop in a set of nested loops) rather than the maximal cyclic nets (blocks of nested loops) and the algorithm considers each loop once only during the search.

The algorithm utilises two arrays in addition to the graph description which in this case is the process matrix. The first array IP (one dimensional) contains the nodes of the loop being extended. The second array IH (two-dimensional) is initially zeroid, and will contain the list of nodes through which the path cannot extend. The first path starts at node 1 and is extended one node at a time but only if these conditions are fulfilled before each tentative extension.

1. The extension node is not already in IP.
2. The node number is larger than the first node in IP.
3. The extension node is not a forbidden node.

Condition 1 assures that an elementary path is formed. Condition

2 assures that each circuit is only considered once. Condition 3 assures that no path is considered more than once. When no more nodes are available for extension, a circuit confirmation is performed. If a stream connects the last node in IP to the first one, a loop is reported. In any case a node closure occurs in a three-step process.

1. The last node in IP is entered in the IH matrix on the row representing the last but one node in IP.
2. The row in IH representing the last node is cleared.
3. IP is shortened by eliminating the last node.

Step 1 assures that the recent path extension is not repeated.

Step 2 allows the correct forward continuation from the last node if it is reached by a different path in the future.

Step 3 leads to the location of new loops.

The extension process is continued until the path is eventually backed down to node 1. Then the initial node is incremented by 1, IH is completely cleared and the loop finding process continues from a new starting point. The search ends when the first node in IP is the last one in the diagram.

In the calculation order finder, the sequence of nodes (units) forming each new loop is copied into another matrix (LOOP) and at the end of the search the node sequences are converted into stream sequences through a study of the stream connection matrix (KSEM giving the stream number, its source and destination unit members), and LOOP is used later to obtain the iterate streams.

Figure 7 - 1 : Flowchart of the loop finder routine.

Initialisation.

LOOP = 0  
IP = 0  
IH = 0

k = 1  
IP(k) = 1

(A)

Path Extension.

Search Process Matrix KPM for  
from row j = 2 to No. of  
columns, so that :  
 $KPM(IP(k), j) > IP(1)$   
~~IP~~  
~~IH(IP(k), m)~~  
m=1, n

j found ?

yes

no

k = k + 1

IP(k) = KPM(IP(k-1), j)

Circuit Confirmation.

For j = 2, No. of columns

IP(1) in KPM(IP(k), j) ?

no

(B)

yes

No. of loops = No. of loops + 1

(B)

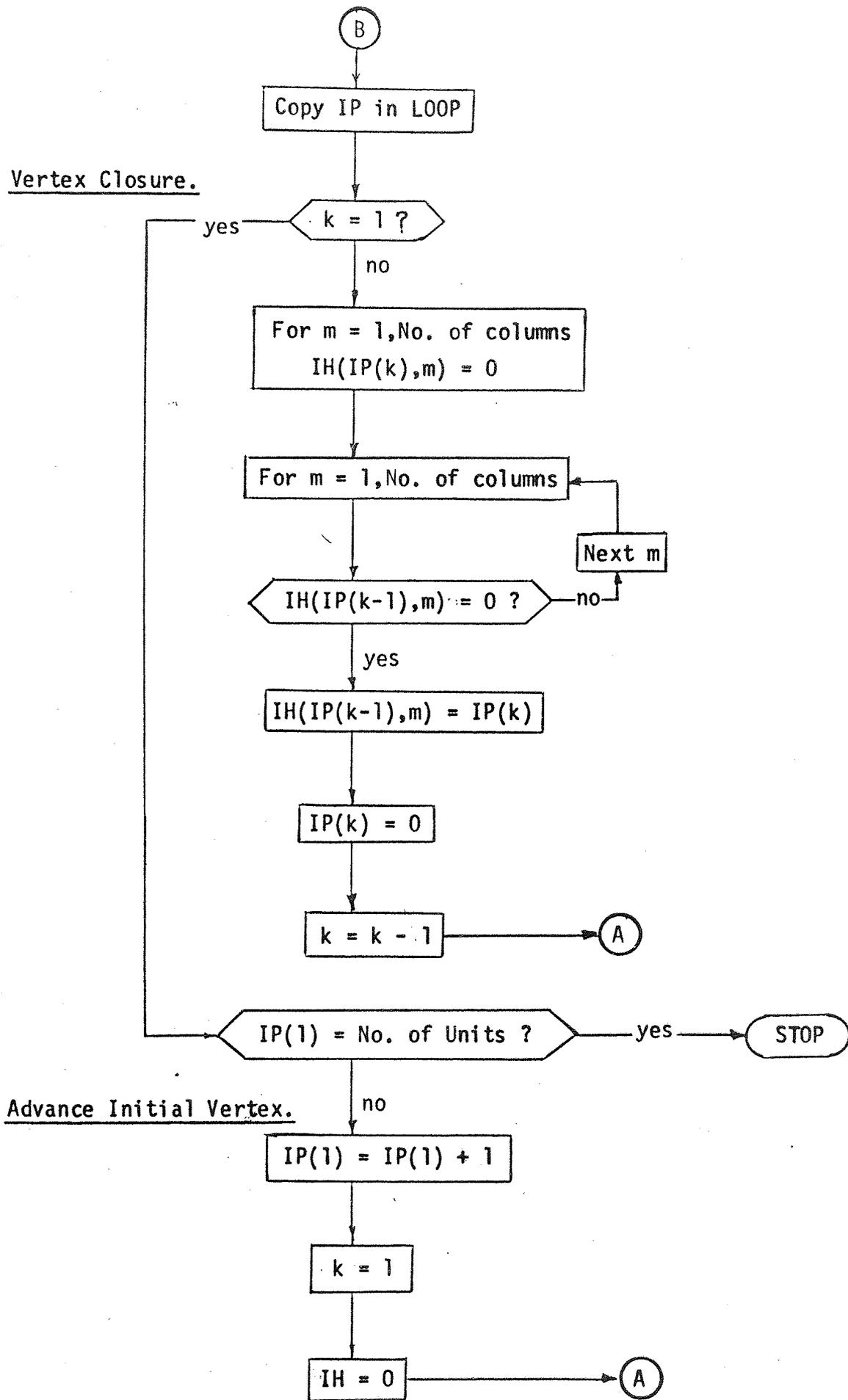


Figure 7 - 1 : Flowchart of the loops finder routine.

(Cont'd)

The flowchart of the algorithm is shown in figure 7-1.

### 7.3 The search for iterate streams

The search for iterate streams is usually, but not necessarily, based on the study of the loops (elementary or maximal) forming the directed graph.

Locating the optimum iterate streams has been largely based on Steward's approach to solving large systems of simultaneous equation (80). He treats each set of equations connected together by certain variables as rows in a matrix where the columns represent the equations and the rows, the sets numbers. Each element of the matrix which represents an equation in the set of connected equations is represented by the number of the next equation that could be solved when this one is solved. The appearance of the same equation number more than once in the same column identifies the equations to be solved initially to cut most cyclic sets of equations. Here each elementary cyclic set of equations must be treated independently. Unfortunately the method does not provide the optimum number of variables to guess initially to tear the system open (26).

Lee, Christensen and Rudd (81) improved Steward's method of tearing sets of equations. They replaced the sets of equations by the equations themselves and used the columns to represent the variables in the equations. Their method consisted in

eliminating the variables appearing in one equation only and the equations with one variable only. The procedure is repeated until a minimum irreducible number of equations remains. These equations and variables form recycle loops. Various combinations of variables, starting with those appearing in the least number of equations are selected, assumed known and the above procedure for deleting known equations and variables is applied. The combination of variables giving the least number of tears and cutting the system completely open is chosen.

Lee and Rudd (69) used the same approach for the solution of recycle loops of units connected by streams. In their final boolean matrix, the rows represent the elementary loops and the columns represent the streams. Non-zero elements in the matrix show the streams forming the loops. Each elementary loop requires only one torn stream to be solved, but since many loops may have one or more streams in common, these streams are the most suitable ones to tear since a lesser number will have to be cut. Their method analyses streams appearing in successively larger number of loops. If all the streams in the loops these streams appear in have not been deleted, these streams under consideration are deleted. By successively eliminating the streams appearing in the least number of loops, the optimum few streams to tear remain as iterate streams. Lee and Rudd provide also an algorithm to deal with streams described by different numbers of parameters. In this case the streams containing the maximum number of parameters are

eliminated first since they require more guessing. However they are eliminated only if another set of streams, needed to tear the loops those streams should have cut, do not contain together more parameters to guess. Thus the criterion of minimum parameters to guess replaces that of minimum number of streams.

Forder and Hutchison improved Lee and Rudd's method and incorporated an option to allow the user to set his own preferred streams. By so doing, the user overwrites the procedure's option to delete these streams during the stream elimination procedure, but these user-suggested iterate streams must appear in recycle loops otherwise they are refused. The procedure then searches for any additional streams required to tear the whole system open, with the limitation that the total number of parameters to guess does not exceed a user-specified multiple of the minimum of iterate parameters, otherwise the preferred streams are rejected and the user asked for other ones.

Kehat and Schem's new approach (26) is very different. They do not consider elementary loops at all but operate on the maximal cyclic nets obtained from an adjacency matrix-type method. To obtain the calculation order they choose their iterate streams one by one as required. An iterate stream is the one or more unknown input streams to a recycle unit characterised by the least number of unknown inputs and the maximum number of outputs. When two or more units obey these requirements, list of units are made up by following all cases through as many possible units until

the first unknown streams is met and the longer list is kept, thus determining the chosen set of iterate streams.

A number of other methods have also been applied to the selection of iterate streams. PACER's method is based on a trial and error selection of successively greater number of streams to be followed through to obtain the calculation order. Christensen and Rudd select trial combinations of recycle streams having a successively greater number of describing parameters. Their method resembles the PACER method when all streams have an equal number of parameters. Both methods guarantee the minimum number of iterate streams but at the expense of longer search times.

#### 7.3.1 The method used.

A procedure similar to Forder and Hutchison's was programmed but without the option of minimising the number of stream descriptors since all streams in PEETPACK are assumed to bear the same parameters (this option has also recently been dropped in CONCEPT). The procedure is very flexible and allows the user to propose his set of preferred streams which are checked for sufficiency and other ones added if they do not cut all loops. The procedure is as follows.

The loop matrix (LOOP) prepared during the search for loops is converted so that each row represents one of the elementary loops and each column represents a plant stream. Non-zero elements in LOOP represent the streams forming the loops. In an additional

row, the flag row, the number of loops in which each stream is involved is summed up. Preferred iterate streams have their sum (or flag) set to the number of loops + 1 so that they are checked last in the search (for over-specification). The number of such streams is recorded in a counter (number of flagged streams). An index is started at zero. The elements of the flag row are checked against this index and the counter raised by 1 every time an element of equal value is met. When the row is completely checked, the stream counter is compared to the total number of streams. If it is equal to it the search ends otherwise the index is raised by 1 and the flag row elements are compared to it. When an element of the same value is met, each element of value 1 in this column of the matrix is searched for. When such an element is located, all the other elements in this row are checked to locate an element of value 1. This implies that there are still at least one other undeleted stream in the loop. If such an element is met, the element in the column under consideration is set to 2 (equivalent to erasing it) and the next non-zero element in the column is checked likewise. Otherwise all elements in the column previously set to 2 are reflagged to 1 and the flag-row element is set to 100. This means that no other stream is left to tear open one of the loops and this stream is thus necessarily an iterate stream. However if all non-zero elements in the column are set to 2, the stream-flag is set to 0 (not an iterate stream). The counter of the flagged stream is raised by 1. The remaining streams are checked

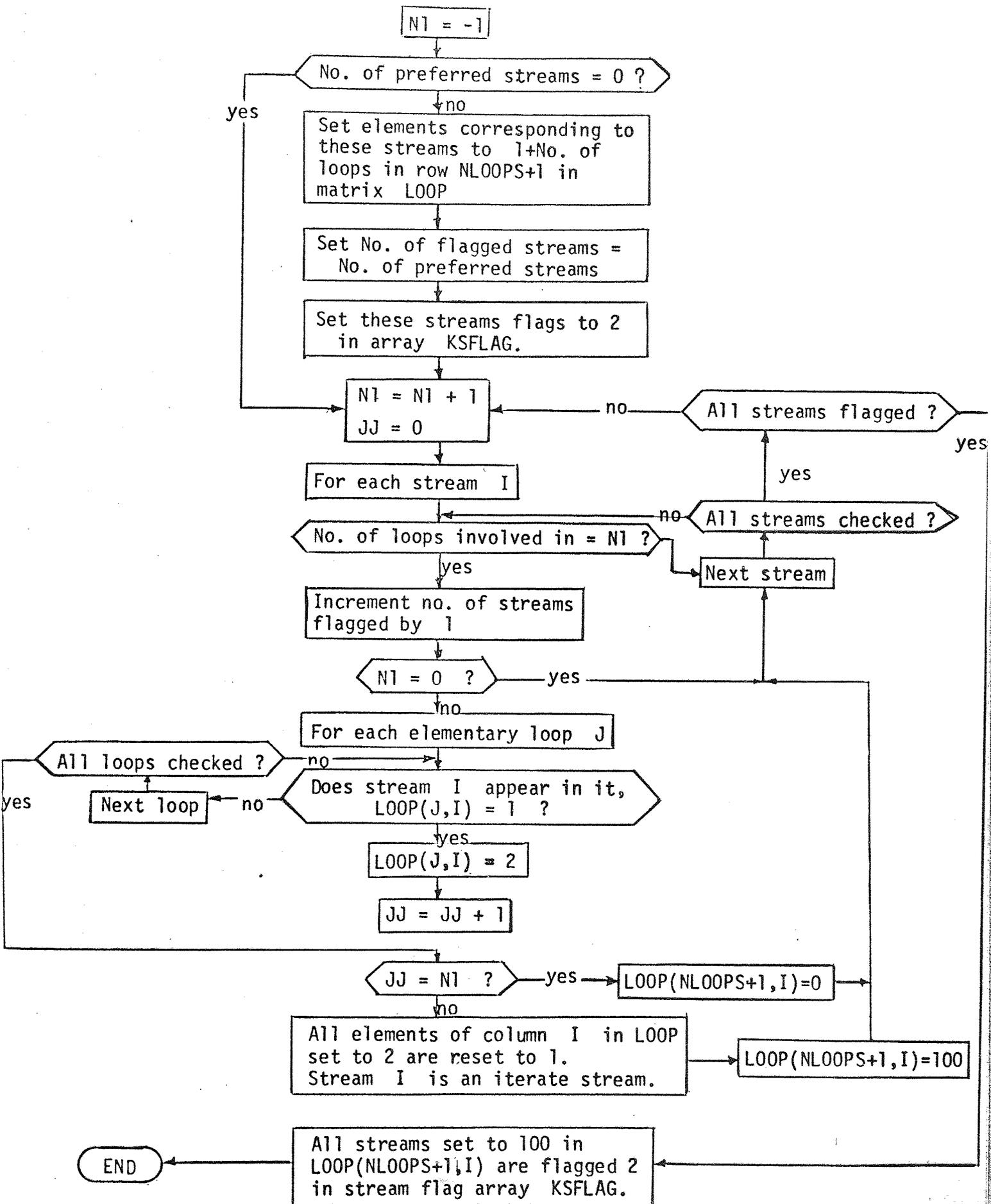


Figure 7 - 2 : Flowchart of Iterate Stream Finder routine.

in the same way as long as they are not flagged 0 or 100. After each matrix scan the counter is checked against the number of streams in the plant and the index raised by 1. The search is ended when all streams are flagged. In the interactive data-input mode, the user is informed of the set of iterate streams found by the procedure, including his own preferred streams. He may either accept them and the program then finds the calculation order, or he may reject them and propose another set of iterate streams. The matrix analysis is repeated to accept or complement his new choice. The procedure can be repeated as often as necessary until the user is satisfied with the whole set of iterate streams. Then the calculation order is obtained. The method's flowsheet is shown in figure 7-2.

#### 7.4 The calculation order

Few methods for forming the calculation sequence have been described in the literature. Most of them are based on following the iterate streams through the plant although the methods of following them varies.

Sargent and Westerberg obtain the ordering of recycle loop units in their dynamic programming of search. The combination of units minimising the number of iterate streams gives the final order.

Christensen and Rudd follow Sargent and Westerberg's

idea of merging together units connected by certain combinations of streams to accelerate their search for iterate streams; these mergers create part of the recycle loop order. The other units are added to it by examining their inputs and only those units with known (or guessed) inputs are added to the list and their outputs set as known. The method follows roughly the PACER method where units are scanned and added to the list only if their inputs are known. Their outputs are flagged as known and the list of units rescanned repetitively until all possible additions are made. If units remain unlisted, a change in iterate streams is necessary.

Kehat and Sacham, following the identification of the iterate streams, delete them then search for units with no inputs and add them to the sequence and delete their outputs. They repeat the procedure until no more units with no input streams are met, then either all units have been added to the sequence or new iterate streams are needed to continue the search. They claim that their method is very simple to program and as useful as more complex ones, but it does not always produce the optimum sequence.

Of the few known methods, PACER's seemed the most useful and appropriate for further development. The method presented below follows PACER's overall concepts, but treats the recycle loop ordering very differently taking into account the different sets of nested loops and orders them separately.

#### 7.4.1 The method used.

The process Matrix, the LOOP matrix and three new arrays are required. In the first two new one-dimensional arrays (KEFLAG, KSFLAG), the unit and stream flags are set and the third two-dimensional arrays (NELIST) will contain the calculation order and the flags indicating the type of sequence the units belong to: sequential (flag = 1), recycle loop (flag = 2), or end of a set of nested loops (flag = 3). Initially the arrays are zeroed. The plant input streams are flagged 1 and the iterate streams flagged 2. A knowledge of the elementary loops was needed to obtain the iterate streams, but converging the calculation for each elementary loop is wasteful and inefficient because of their interdependence. Loops having one or more streams in common are merged together and the LOOP matrix condensed to show only the maximal cyclic nets. This is done by boolean addition to the top rows of rows having non-zero elements in common columns, and pushing towards the top the lower unmerged rows so as to overwrite the redundant ones. The modified matrix is then converted back to show the units forming the maximal cyclic nets, and a list is made up of the number of units involved in each maximal cyclic block.

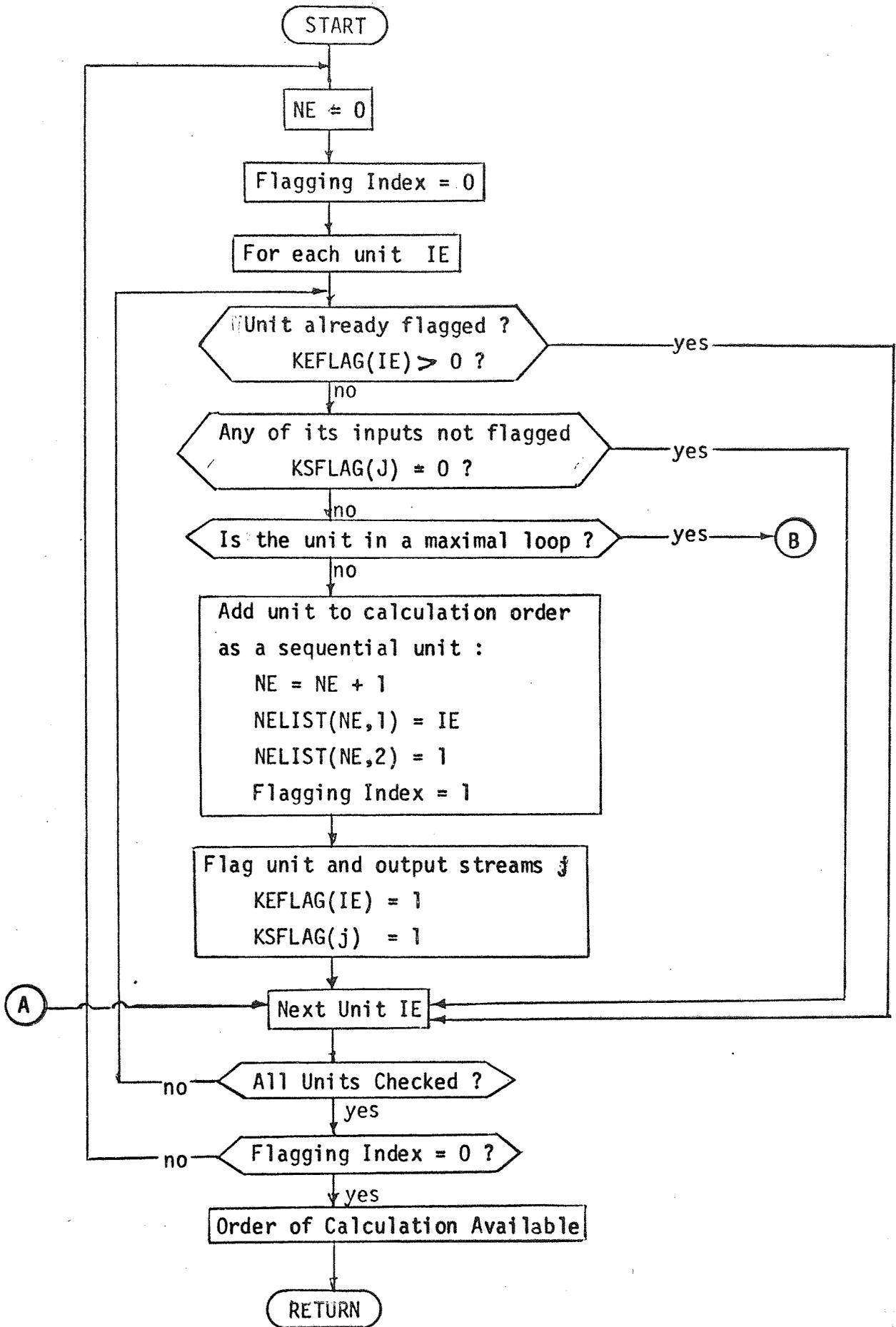
The list of plant units is scanned unit by unit. A flagged unit (non-zero flag in KEFLAG) is already in the order and is by-passed. The flags (in KSFLAG) of the input streams to an unflagged unit are checked. If at least one of them is

unflagged, the unit is rejected and the next one is checked. If all input streams are flagged the list of units in the loops is checked. A unit not partaking in a loop is flagged 1 (in both KEFLAG and NELIST) and so are its outputs (in KSFLAG) and its number is added to the calculation order (NELIST). For a unit in a maximal cyclic net, this block of units is treated itself as a sub-system with each unit, starting with this initial one, checked. In this sub-system a unit with flagged inputs is flagged 3 and so are its unflagged output streams (flag 3 is a temporary flag). Its number is added to the calculation order and its order flag set to 2. The number of recycle units added to the order is recorded. The maximal cyclic net is scanned repetitively until no more flagging is possible. Then if a unit remains unflagged, the loop cannot yet be calculated. All units and streams flagged 3 are deflagged and the units added to the calculation order deleted. However, if all the units in the loop are added to the order of calculation list, the unit and stream flags of value 3 are set to 1 and the last unit in the sequence is given an order flag 3 to indicate the end of a recycle loop.

The scanning of the main list of units is repeated as above until all the units are flagged thus producing the complete calculation order.

The flow chart of the calculation order finder method is illustrated in figure 7-3 and that of the method used in dealing with recycle loops is shown in figure 7-4.

Figure 7 - 3 : Flowchart of the Calculation Order Finder.



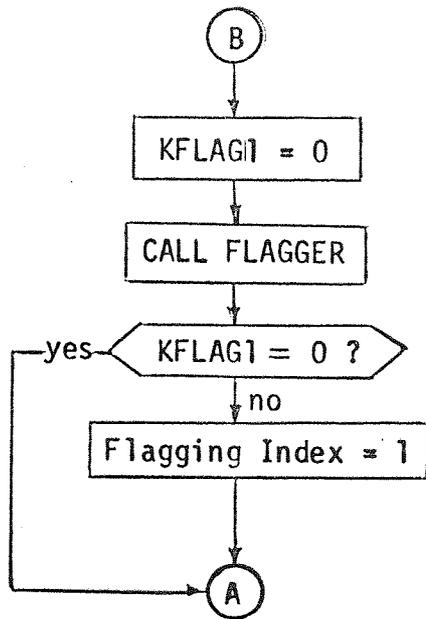
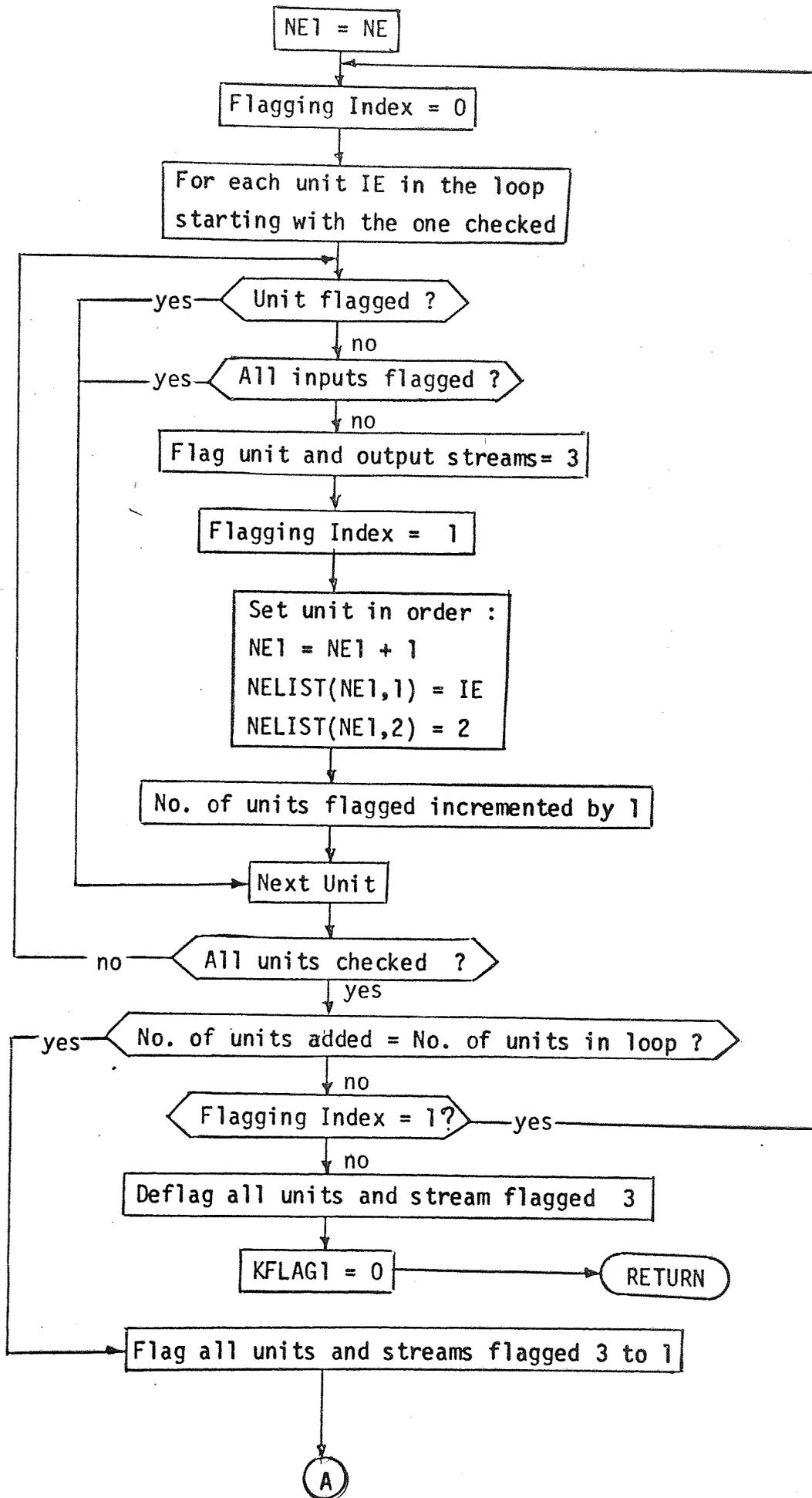


Figure 7 - 3 : Flowchart of The Calculation Order Finder.  
(Cont'd)

Figure 7 - 4 : Flowchart of routine FLAGGER.



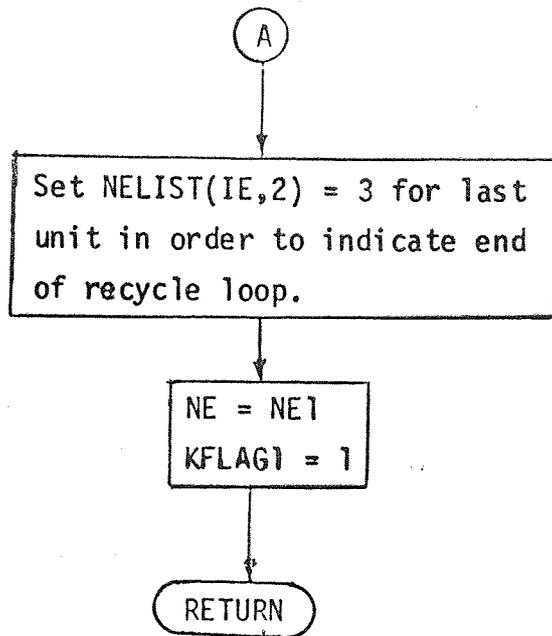


Figure 7 - 4 : Flowchart of routine FLAGGER.  
(Cont'd)

## 7.5 Conclusion.

The calculation order finder proposed here is very well suited to interactive dialogue and preferred stream setting, and will always give the minimum number of iterate streams subject to the user's choice of preferred streams.

Little evidence exists to compare the efficiency of the various methods described in this chapter, and because of the difficulty of programming some of the earlier methods no comparative analysis was attempted.

## CHAPTER 8

### THE UNIT OPERATION MODELS

The unit operation models and the thermodynamic and physical property evaluation routines are the major differences between the reasonably comprehensive package CONCEPT and the executives PACER and GEMCS. These features more than any other are the deciding factor in the wide acceptance or rejection of simulators at the industrial level, but are of lesser importance as teaching tools. The absence of general data routines complicates tremendously the preparation of unit models, and with no models available as examples, writing even the simplest model can be a relatively difficult task.

To overcome these problems, a number of models are proposed in PEETPACK. They are mainly heat, mass and energy balancing routines operating on their input streams information to produce the output streams descriptions subject to operating conditions or constraints imposed in the models parameters and based on the thermo-physical property data. The models emphasise the heat and mass balances rather than the mechanical design of the units for the following reasons.

1. Consistent heat and mass balances for the plant are necessary before mechanical design calculations can be made.
2. Standard methods for mechanical design are not yet widely accepted by all manufacturers who prefer to apply their own trusted methods.

3. Complete and optimum designs are too expensive to run on simulators.
4. Optimality criteria are not unique for all design cases. They are time and job dependent and accounting for most of them in a single routine would make its running prohibitively long and uneconomical.
5. The design of many units often requires an educated choice between several alternatives, made following the calculation of heat and mass balances.

Some of the more comprehensive or longer standing programs do offer mechanical design features, but details of these routines or of their effect on the simulation time are not usually published. Since PEETPACK is proposed as a teaching tool as well as an industrial package, and as elaborate design models are of limited use at academic level, such routines have not been included but some do exist in other theses (2, 58).

The unit operation modelled in PEETPACK are the following:

1. A mass-only mixer (routine MASSMIX).
2. A mass and heat mixer (MIXER).
3. A stream divider (SPLITTER).
4. A combined mixer then splitter model (SPLITMIX).
5. A throttle valve (VALVE).
6. An adiabatic flash column (FLASHER).
7. A stream purifier (PURIFYER).
8. A pump or pressure setter (PUMP).

9. An adiabatic gas compressor (COMPRESR).
10. A stoichiometric reactor (REACTOR).
11. A multiphase counter-current heat exchanger (HEATEXCH).
12. A heater/cooler or temperature setter (HEATER).
13. A multicomponent, tray-to-tray, distillation column with optional reboiler model (DISCOL).
- 14,15 - A dew point and bubble point setters (SETDP, SETBP).

All the models rely entirely on the thermodynamic and physical property packages described in the next chapter for their data and all components used in the simulation must have their data set in the data files. The accuracy of the models depends entirely on that of the thermo-physical data evaluation routines unless otherwise stated.

The models parameter requirements are listed in table 8-1 and flow charts are given only for the more complex routines as most models are very simple in programming.

#### 8.1 The MASSMIX model.

This model mixes the flows of an unrestricted number of input streams and computes the new compositions of the single output streams, as described in section 8.2. However, it does not compute the stream's outlet temperature or vapour fraction.

Table 8 - 1 : The parameters of the PEETPACK models.

Model	Parameter position.	Description.
MASSMIX	-	None.
MIXER	-	None.
SPLITTER	1 to No. of outputs	Fractional splits.
SPLITMIX	1 to No. of outputs	Fractional splits.
VALVE	1	Output pressure.
FLASHER	1	Output pressure.
PURIFYER	1 2 to N+1	Number of components to remove (N). Components positions in the components order read in.
PUMP	1 2	Output pressure. Mechanical efficiency (optional).
COMPRESR	1 2 3 4	Output pressure. Isentropic efficiency (optional). Mechanical efficiency (optional). Ratio of $C_p/C_v$ . (optional).
REACTOR	1 to No. of components N + 1 N + 2	The reaction stoichiometric coefficients.  Extent of reaction factor (optional). Heat of reaction (optional).

Table 8 - 1 (continued)

HEATEXCH

- 1 . Temperature of first outlet stream.
- 2 Exchanger's surface area.
- 3 Heat transfer coefficient.
- 4 Pressure drop in the first stream.
- 5 Pressure drop in the second stream.
- 6 Set to 1 if parameter 1 represents a difference between  $T_{out\ 1}$  and  $T_{in\ 2}$  otherwise set to 0.  
(use parameter 1 or 2 & 3, set unused ones to 0).

HEATER

- 1 Output temperature.
- 2 Pressure drop in the stream.

DISCOL

- 1 Number of stages in the column.
  - 2 Feed stage.
  - 3 Tops product flow rate.
  - 4 Vapour flow rate in the Tops.
  - 5 Reflux ratio.
  - 6 Column pressure.
  - 7 Light Key component number.
  - 8 Heavy Key component number.  
(as in the order read in).
  - 9 Pressure drop in the input stream to the reboiler.
  - 10 Number of terms in the K-value vs. Temp. correlation (optional).
  - 11 Temperature code used =
    - 1 for degrees F
    - 2 for degrees C
    - 3 for degrees R
    - 4 for degrees K
  - 12 The coefficients for the K-value correlation starting with the first component in the correlation :
- onwards

		$K = A + B \times T + C \times T^2 + D \times T^3$ $+ E \times T^4 + F \times T^5 + G \times T^6$ <p>The coefficients are read as :  A, Bx1.0E2, Cx1.0E4, Dx1.0E7,  Ex1.0E10, Fx1.0E12, Gx1.0E14  (parameters 9 onwards are optional).</p>
SETDP	-	None.
SETBP	-	None.

Table 8 - 1 : The parameters for the PEETPACK models.

(Cont'd)

It sets them both to the mass average of all the inputs and sets the pressure to the lowest of all input pressures.

## 8.2 The MIXER model.

The mixer sums up the flows of an unrestricted number of streams and computes the composition of the unique output stream based on the following equations.

$$\text{New Flow} = \sum_i \text{Input flows} = \sum_i F_i \quad (8.2-1)$$

$$\text{New Compositions} \quad X_j^n = \left( \sum_i X_{i,j} * F_i \right) / \sum_i F_i \quad (8.2-2)$$

where subscript  $i$  refer to the input stream and  $j$  to the component.

The output's enthalpy is calculated as

$$\text{New Enthalpy} = \left( \sum_i F_i * H_i \right) / \sum_i F_i \quad (8.2-3)$$

where  $H_i$  refers to the enthalpy of the  $i$ th input stream.

The stream output temperature and vapour fraction are calculated in routine TEMP of the thermodynamic property package.

The restrictions imposed on the model are as follows:

1. No heat of mixing is considered (i.e. ideal solutions).
2. No chemical reaction takes place in the mixer.
3. No change in composition due to accumulation is considered.

## 8.3 The SPLITTER model.

The model divides a single input stream into an unrestricted number of streams. It may be used to model a flow-dividing

function or a single-input, multiple-output steady-state tank. All output streams variables are set identical to the input stream except for the flowrate divided among the output streams according to the fractional splits set in the units parameter array. The model normalises the splits to allow the user to set them in any form.

#### 8.4 The SPLITMIX model.

This model may represent multiple input, multiple output steady-state tanks. It calls first the MIXER model to mix the input streams into one stream which is then fed to the SPLITTER model to split it into the number of outputs specified by the user. It requires the same parameters as the SPLITTER model.

#### 8.5 The throttle VALVE model.

The throttle valve routine can be used either to model throttle or control valves or to simulate pressure drops in pipes or other units. In the routine, the stream's pressure is decreased to the specified outlet pressure set by the user. The valve operates adiabatically thus affecting its output's temperature and vapour fraction.

The program sets the output stream description equal to that of the input and changes its pressure to the set value. Routine TEMP is called to calculate its new temperature and vapour fraction, but the stream is not divided into its pure-phase fractions.

## 8.6 The FLASHER model

This adiabatic flash column model is similar to the VALVE model except that the two-phase stream is split into its vapour and liquid phases by a call to routine FLASH of the thermodynamic package. The vapour output is its first output stream and the liquid the second output. Their enthalpy is calculated in routine ENTHALPY.

The model's only parameter is its operating pressure. It is accurate within the limitations of the ENTHALPY and the vapour-liquid equilibrium ratio routine (KVALUE).

## 8.7 The PURIFYER model.

This model is not based on any particular unit operation or on any theoretical bases. It may be used either as a decanter model where the user knows that certain components can be removed completely by physical separation, or as an over-simplified distillation column model for quick calculations.

The model separates into the second output stream the components the user specifies for removal from the first output stream, and it calculates these streams' total flows based on simple mass balances on the individual components. These output streams compositions are updated and their enthalpy calculated in routine ENTHALPY based on an isothermal separation. The list of parameters should contain the number of components to be

removed from the input stream and their incidence numbers as entered in the data.

#### 8.8 The PUMP model.

The pump model may be used either to model isothermal fluid pumping and set its pressure at a higher specified value, or as a pressure changer for a fully defined stream. In both cases the stream's pressure is changed and its new enthalpy and vapour fraction calculated in routine ENTHALPY. An optional mechanical efficiency may be set (the default value of which is 100%) and the pump's idealised energy requirement is calculated as the difference between its input and output stream's enthalpy divided by its mechanical efficiency.

This model should usually be used for liquids and routine COMPRESR for gases.

#### 8.9 The COMPRESR model.

Gas compressors, whether centrifugal or reciprocating, are generally represented by a polytropic gas compression law of the forms

$$P V^n = \text{Constant} \quad (8.9-1)$$

for each stage of a single or multi-stage compressor. When the compressors are cooled, the power  $n$  may take different values between:

a.  $n = 1$  when the compressor is cooled to the point that it works isothermally, therefore:

$$\frac{PV}{T} = \text{constant} \quad \text{or} \quad PV = \text{constant} \quad (8.9-2)$$

since  $T$  is constant, and

b.  $n = \gamma$  ( $C_p/C_v$ ) the ratio of the specific heats when the compressor operates adiabatically (82).

In general compressors never operate at these extreme unfeasible conditions (too much cooling is required in case a and too much energy input is required in case b). The power  $n$  is usually calculated from practical data taken at the inlet and outlet of the compressor, and  $n$  is forced to fit them, however it tends to be closer to than to 1. By rearranging equation (8.9-1) assuming the ideal law to hold even at high pressures and temperatures, the following relationships are obtained (82):

$$\frac{T_{out}}{T_{in}} = \frac{P_{out}}{P_{in}}^{(n-1/n)} = \frac{V_{in}}{V_{out}}^{(n-1)} \quad (8.9-3)$$

and the ideal work required to achieve the compression at each stage is:

$$\begin{aligned} W &= \int_{in}^{out} PdV = \frac{(P.V)_{out} - (P.V)_{in}}{n-1} \quad (8.9-4) \\ &= \frac{\text{Flow} * \text{Gas Constant } R * (T_{out} - T_{in})}{n-1} \quad (8.9-5) \end{aligned}$$

To account for the deviation of the actual mechanical energy required from the ideal energy calculated, a mechanical

efficiency factor  $\eta_m$  is introduced, where:

$$\eta_m = \frac{\text{Air horse-power}}{\text{Shaft horse-power}} * 100 = \frac{\text{Ideal energy required}}{\text{Actual energy required}} \quad (8.9-6)$$

In situations where compressors are not cooled and are considered to operate adiabatically (or isentropically;  $n = \gamma$ ), an isentropic efficiency  $\eta_s$  is introduced to account for the heat losses to the surroundings, the heat generated by friction, etc. It is introduced as:

$$\eta_s = \frac{\text{Ideal Work}}{\text{Actual Work}} = \frac{T_{out \text{ ideal}} - T_{in}}{T_{out \text{ actual}} - T_{in}} \quad (8.9-7)$$

In polytropic compressors ( $1 < n < \gamma$ ) this efficiency is introduced in the power term as follows (82):

$$\frac{T_{out \text{ actual}}}{T_{in}} = \frac{P_{out}}{P_{in}}^{(n-1)/(n * \eta_p)} \quad (8.9-8)$$

where  $\eta_p$  is the polytropic efficiency.

In the COMPRESR model, the polytropic and adiabatic cases are modelled insofar as their governing equation (8.9-1) is the same, but the isentropic efficiency rather than the polytropic efficiency is used in the calculations because its adjustment to match simulated data is easier.

The routine calls DEWBUB initially, to obtain the mixture's bubble point at its output pressure, and compares it against its temperature. If the stream is not all vapour, its new enthalpy at the outlet pressure and the same temperature is calculated and the stream's new vapour fraction is computed. Otherwise it applies

equations (8.9-3), (8.9-8), (8.9-5) and (8.9-6) to compute the mixture's pressure at the output pressure set in the parameter array. The power  $n$  is either introduced as a parameter or computed using routine SPHEAT ( $C_v = C_p - R$ ). In the latter case  $C_p$  and  $C_v$  are calculated at the stream's input temperature and the output temperature calculated based on these constants.  $C_p$  and  $C_v$  are computed again at the average of the input and output temperatures and the calculations are repeated until convergence on the output temperature is achieved. The enthalpy of the output stream is computed in routine ENTHALPY and the work required by the pump is computed using the mechanical efficiency set in the parameter array. Both efficiencies have a default value of 100% when unset.

If at any time during the iterative procedure to calculate the correct output temperature, or when  $n$  is known the temperature falls below the dew point of the mixture at its output pressure, a warning message is issued to the effect that the mixture cannot be gaseous at this pressure and the stream is treated as a liquid and compressed isothermally.

The accuracy of the model depends primarily on that of the power  $n$  and the efficiencies provided by the user, or on the accuracy of the SPHEAT calculations, provided the stream is always in the vapour phase.

The routine's flowchart is given in figure 8.1.

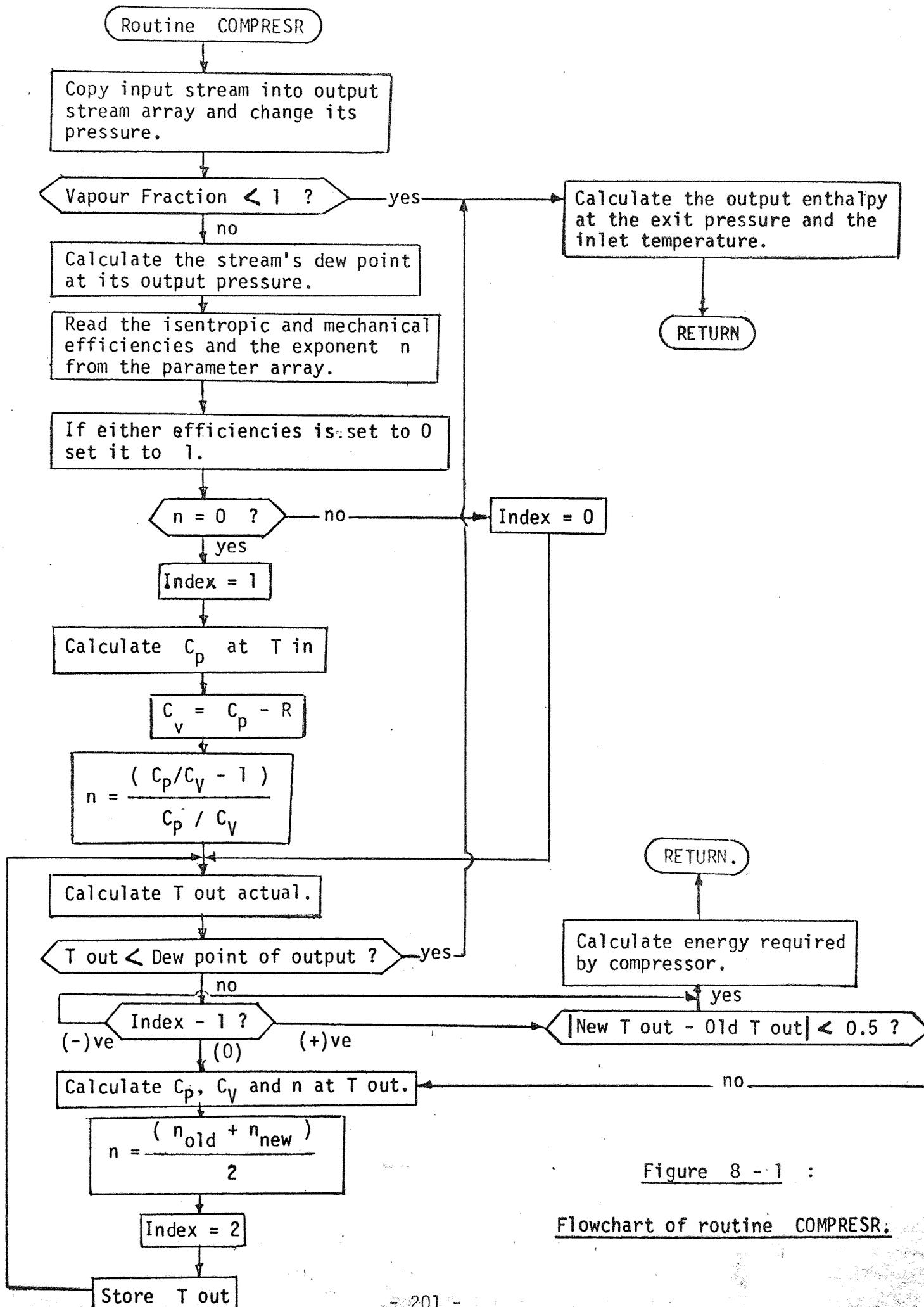


Figure 8 - 1 :

Flowchart of routine COMPRESR.

## 8.10 The HEATEXCH model.

This cross-current heat exchanger model operates on two streams exchanging heat. It allows the user either to set the outlet temperature of the first output stream or to set the difference between this stream's temperature and that of the second input, thus specifying the heat load, or he may set instead the internal heat exchange area (A) and the heat transfer coefficient (U) and let the model calculate the heat load and output temperatures. In both cases the pressure drop across both sides may also be set. The model does not define either stream as being on either side (shell or tube) since this adds no more information to the model.

### 8.10.1 Case 1: One outlet temperature specified.

The output stream thus specified must be the first one and consequently the extension of the first input. The temperature is set either to the value in parameter 1, if parameter 6 is set to 0 or to the temperature of the second input stream plus the value of parameter 1, if parameter 6 is set to 1. Its pressure is reduced by the pressure drop set in parameter 4. Its new enthalpy and vapour fraction are calculated in routine ENTHALPY. The stream's total heat change is then subtracted from the second input stream's enthalpy and this stream's new temperature and vapour fraction are calculated at its input pressure minus the pressure drop set in parameter 5.

A number of consistency checks are included to avoid simulating unrealistic conditions such as requesting to heat up the warmer of the two streams or cool down the cooler one, or attempting to heat or cool a stream beyond the input temperature of the counter-current stream. If the second stream temperature rises or drops beyond the first stream's input temperature because of the heat load required from it, its temperature is set equal to the first stream's input temperature and the first stream's output temperature is then computed according to the new heat load demanded by the second stream, thus overwriting the parameters values. Warning messages are issued if any of these problems arises.

8.10.2 Case 2: The heat transfer area and coefficient specified

An initial guess of the total heat exchange is set equal to half the difference between the two input temperatures multiplied by the area and the heat transfer coefficient. This load is added to the cooler stream and subtracted from the hotter one and the two new outlet temperatures computed at their respective pressures less the pressure drops on their respective sides (parameters 4 and 5). A new log-mean temperature difference is calculated as

$$\Delta T \text{ log-mean} = \frac{(T_2 \text{ out} - T_1 \text{ in}) - (T_2 \text{ in} - T_1 \text{ out})}{\ln \frac{T_2 \text{ out} - T_1 \text{ in}}{T_2 \text{ in} - T_1 \text{ out}}} \quad (8.10-1)$$

and a new heat transfer load is calculated:

$$H = U * A * \Delta T \text{ log-mean} \quad (8.10-2)$$

This new heat load is compared to the previous one and a converged solution is achieved if the change is less than 0.5%. Otherwise the new heat load is substituted for the previous assured one and the procedure repeated until convergence. Routine TEMP is called at every iteration to set the outlet temperature and the vapour fraction of each stream at their new enthalpy, hence fully specifying both streams at convergence. The method is stable in that the more heat is exchanged, the closer the temperatures at the same extremes get and the smaller is the log-mean temperature difference and heat load computed from equation (8.10-2), and vice-versa for small heat-load guesses. It is however extremely time consuming because of its numerous calls to routine TEMP and thus indirectly to routines ENTHALPY, FLASH, DEWBUD and KVALUE. A convergence promoter has not yet been introduced to accelerate the convergence.

Of the above two cases, case one should preferably be used when computing time economy is required or good estimates of U and A unavailable. The model's accuracy depends on the property package accuracy and on the accuracy of the parameters set by the user. The model's flowchart is illustrated in figure 8-2.

#### 8.11 The HEATER model.

This model may serve two purposes, either as a stream heater or cooler when the heating or cooling medium is of no interest to the simulation, or as a temperature setter for a

Figure 8 - 2 : Flowchart of routine HEATEXCH.

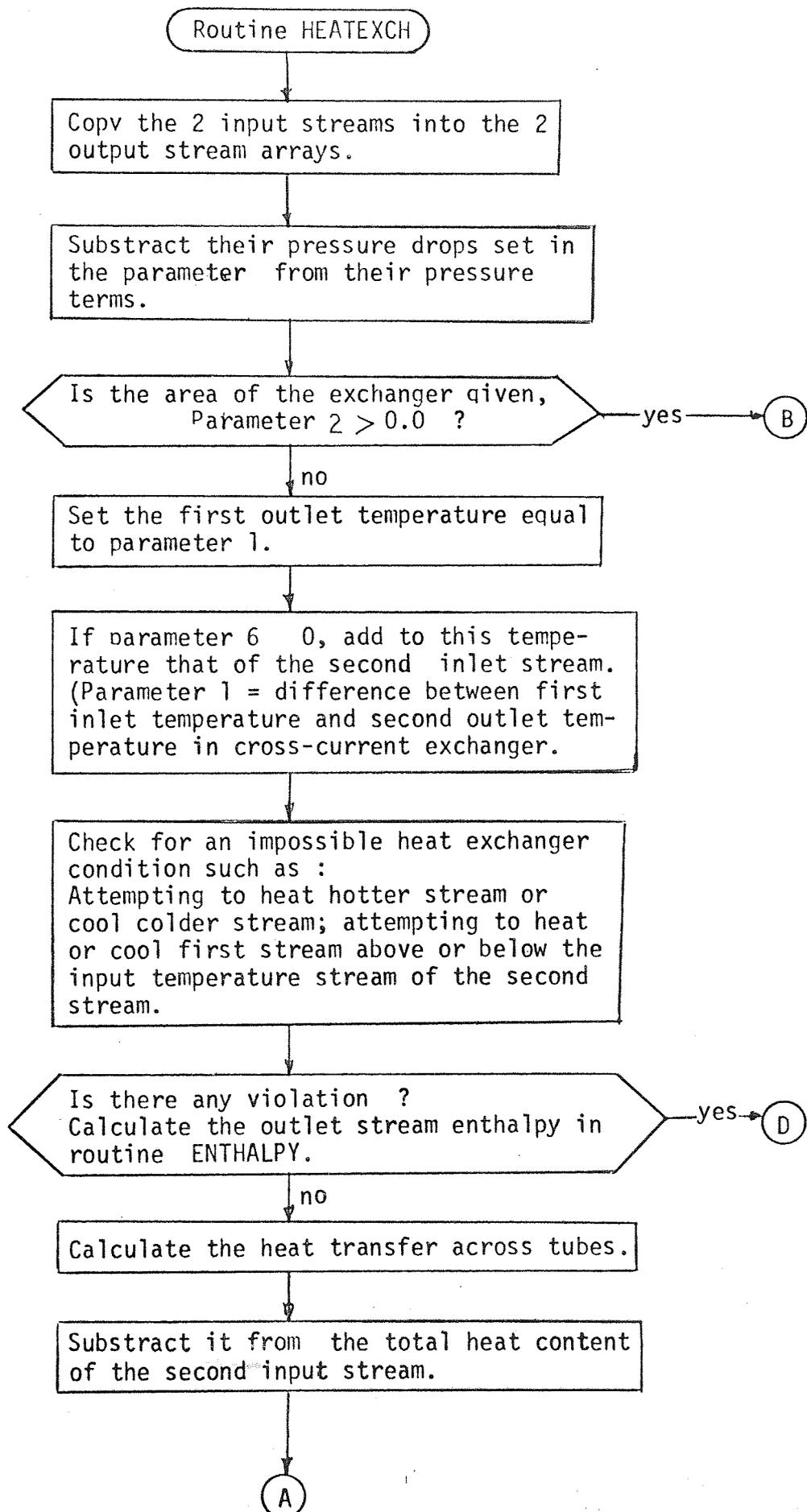
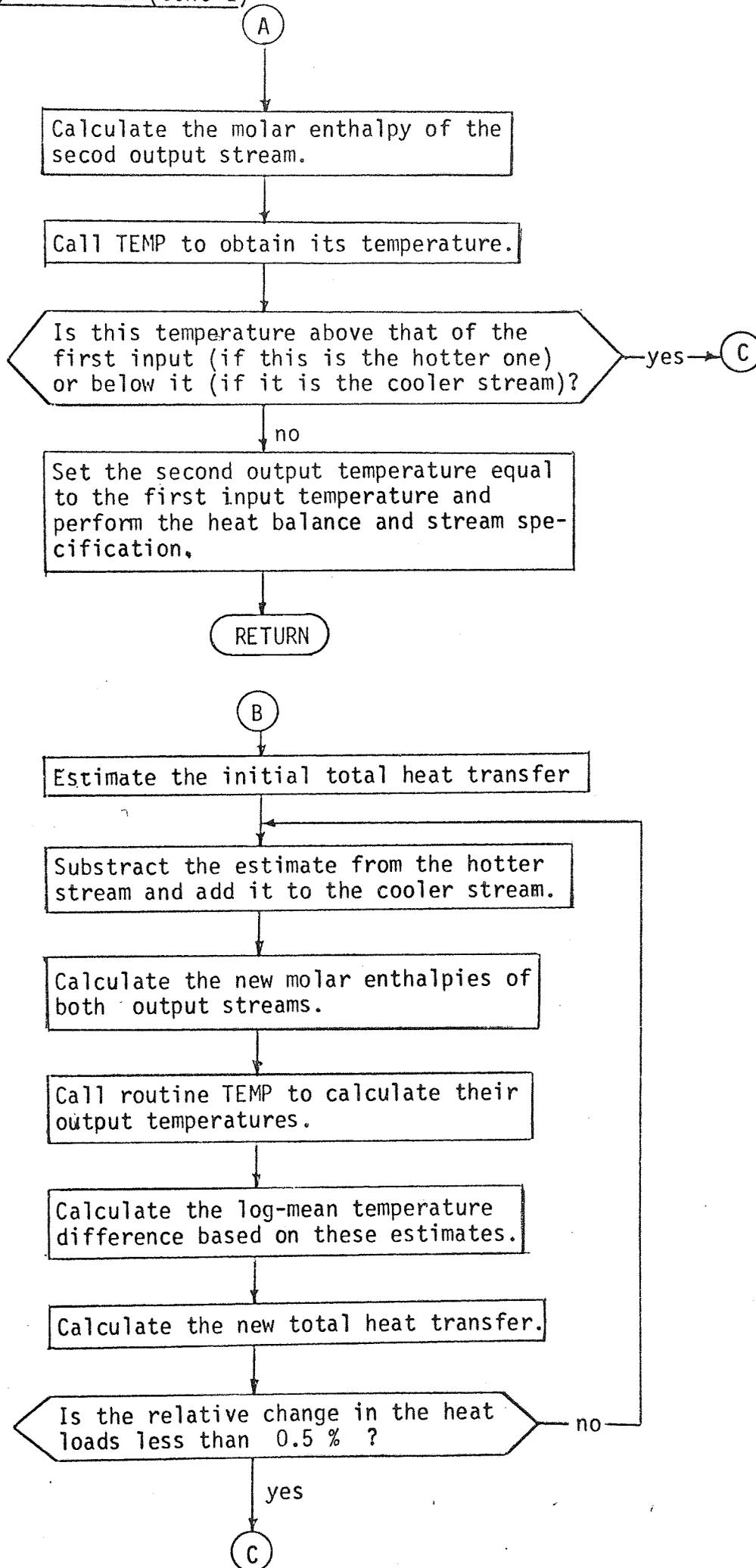


Figure 8 - 2 (Cont'd)



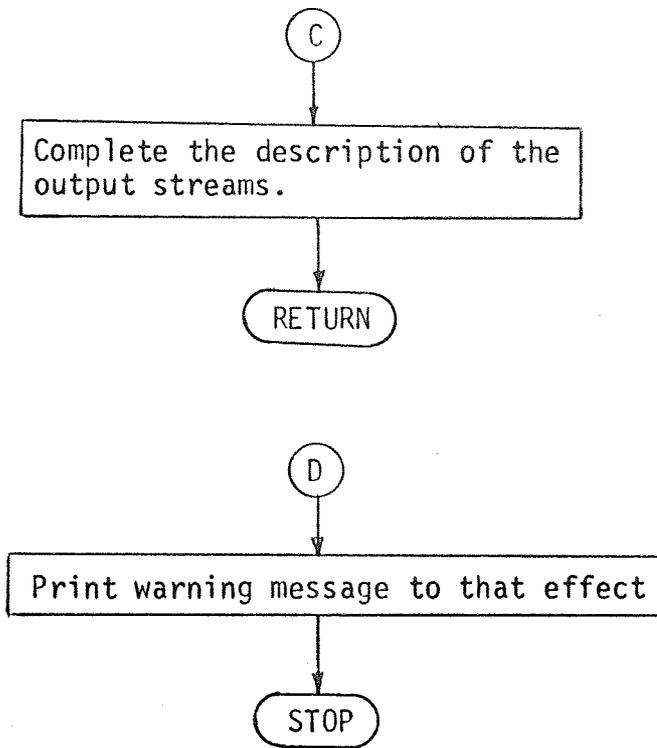


Figure 8 - 2 : Flowchart of routine HEATEXCH.

(Cont'd).

stream otherwise specified. The pressure drop in the unit may also be specified by the user although the model has no means of checking its consistency.

The routine sets the outlet stream description equal to that of the input stream but sets its temperature and decreases its pressure in accordance to the user-set parameters. It then calls the ENTHALPY routines to calculate its new enthalpy and find its new vapour fraction. The absolute difference in enthalpy between the output and input stream enthalpy is printed to inform the user of the heat load on the heater/cooler unit.

#### 8.12 The REACTOR model.

The wide variety and range of chemical reactions and reactors and the difficulty in choosing the ones to model at the expense of the others make it impossible to propose reasonably accurate models. This routine solves partially the dilemma by offering a simplified and idealised stoichiometric reaction model which may be applied to many types of reactions. The model does not compute the chemical equilibrium ratios, but requests the coefficients of the chemically balanced equation. For example, for the equation:



where A, B, C and D are the chemical compounds and a, b, c and d their stoichiometric molar reaction coefficients, the model requires that a and b be entered as negative values (disappearing reactants) and c and d as positive numbers (created products).

Factors must be given for all components in the system, even if they do not partake in the reaction (the reaction coefficient are zeroed), and in the order the components were input in the data input section. An optional "extent of reaction" factor (default value = 1 (=100%)) may be set to control the extent of conversion of the reactants. The molar heat of reaction based on the input stoichiometry may also be provided as a parameter. It will affect the output stream's enthalpy and temperature. The model operates as follows.

The MIXER model is called to mix the input streams to the reactor when there is more than one. REACTOR then divides the molar flowrate of each reactant (negative stoichiometric coefficient) by the absolute value of this factor and retains the smallest quotient. This is based on the consideration that the reaction will go on as long as the least present reactant still exists. Thus these ratios give the equivalent number of product moles each reactant can help generate, and the reaction is controlled by the lowest equivalent:

$$\text{Equivalent Factor} = \text{Total Flow} * X(i) / (-\text{Parameter}(i)) \quad (8.12-2)$$

where  $X(i)$  is each reactant's input mole fraction. The smallest equivalent factor is multiplied by the extent of reaction to account for the reaction conditions. The new flow rate of each component is obtained by adding to its input molar flow rate its new contribution from the reaction:

$$\text{Change in flow of component } i = \text{Equivalent factor} * \text{stoichiometric coefficient} \quad (8.12-3)$$

This represents the number of times the reaction represented by the input stoichiometry can be repeated in the input flow (the equivalent factor) and hence generating or consuming the number of molar specified by the stoichiometric coefficients.

The total heat of reaction is the product of the heat of reaction specified in the parameters and of the equivalent factor. It is added to the input stream's total enthalpy and the output stream's new temperature and vapour fraction are calculated through a call to routine TEMP.

The model is useful in that it can be easily adapted to all reactions for which the chemical kinetics are known, however none of the stoichiometric factors nor the "extent-of-reaction" factor are temperature or pressure dependent.

### 8.13 The DISCOL routine.

This multi-component distillation column model gives accurate heat and mass flow estimates on all plates and condenser and reboiler duties. It operates on one input stream (first input) of any quality and on a liquid reflux (second input) in equilibrium with the tops composition, but the number of flow and heat feeds and withdrawals can be increased by simple routine modifications (83). The model however is only a heat and mass balancing model. It does not calculate the number of trays needed to effect the required separation nor does it compute the minimum or optimum reflux ratio, both must be input as parameters. The top and bottom products are

removed as output streams 1 and 2 respectively. An optional reboiler model operates on a third (optimal) input stream. It strips it of the heat load required by the reboiler as calculated in the column model and outputs a third, cooler, stream.

The theoretical column model is by Wang and Henke (83). It is based on the simultaneous solution of the material balance equation (M), the equilibrium equation (E), the summation equation (S) of mole fractions and the heat balance equations (H) on each stage (they are known as the MESH equations). The M equation is:

$$M_{i,j} = (L_{j-1})(X_{i,j-1}) - (V_j + W_j)(Y_{i,j}) - (L_j + U_j)(X_{i,j}) + (V_{j+1})(Y_{i,j+1}) + (F_j)(Z_{i,j}) = 0 \quad (8.13-1)$$

The E equation is:

$$E_j = (Y_{i,j}) - (K_{i,j})(X_{i,j}) = 0 \quad (8.13-2)$$

The S equation is:

$$S_j = \sum_i Y_{i,j} - 1.0 = 0 \quad (8.13-3)$$

The H equation is

$$H_j = (L_{j-1})(h_{j-1}) - (V_j + W_j)(H_j) - (L_j + U_j)(h_j) + (V_{j+1})(H_{j+1}) + (F_j)(H_{Fj}) - Q_j = 0 \quad (8.13-4)$$

Subscript (i) refers to the component and (j) to the plate.

V, L, F are the vapour, liquid and feed flowrates on each plate.

Y, X, Z are their respective compositions.

H, h, H<sub>F</sub> are their respective enthalpies.

W, U, Q are the vapour, the liquid and the heat taken off each plate.

They are usually set to zero except for U<sub>1</sub> which is the tops liquid

flow rate.

Equations (8.13-1) and (8.13-2) are combined and the L's expressed as functions of the V's by an overall mass balance involving all trays from the one studied up to the condenser. The M-equation is reduced to a tridiagonal matrix form where each row represents a stage and the matrix is solved to obtain all the  $X_{i,j}$  following the initialisation of the vapour flow rate on each plate which is assumed to vary linearly from  $((R + 1) * \text{Top flow rate (DD)})$  on the top plate to  $(R * \text{DD})$  on the bottom plate, where R is the reflux ratio. The initial temperature profile is also assumed to vary linearly from the dew point at an assumed tops composition to the bubble point at a bottoms composition. A trial and error procedure is applied to the temperature of each plate to satisfy the S-equation, then the H-equation is used to update the vapour flowrates. The procedure is repeated until the temperature profile reaches a steady answer.

To perform the heat and mass balances, the vapour-liquid equilibrium ratios are constantly needed. To avoid repeating the long procedure proposed to the KVALUE routine (section 9.1), the user may insert his own K-value vs. temperature correlations via the unit's parameter array (table 8-1). A special routine (EVALUE) is available to use his coefficients and return the K-values. The correlations may use any temperature units and a special parameter specifies the units used. The use of this routine could reduce the routine calculation time by three-quarters.

The accuracy of the model is related to that of the K-values used. They would affect mainly the temperature profile of the column, and variations in the tray compositions may also reflect these inaccuracies. Poor enthalpy relationships will affect the vapour and liquid flow rates and reboiler and condenser duties. However, the main source of errors is due to the limitations imposed by the model.

1. The trays are assumed to behave ideally.
2. Heat losses from the column are neglected.
3. The model is insensitive to the reflux which is assumed to be condensed liquid from the tops output.
4. The number of plates and the reflux ratio are set by the user.
5. The separation of azeotropic mixtures cannot be modelled.

The simplified reboiler model subtracts the heat required by the reboiler (stage N) from the total heat content of the optional third input stream. Routine TEMP is called to calculate the stream's new temperature and vapour fraction and this temperature is checked for consistency against the reboiler's temperature. A warning is issued when the heat requirements cannot be met by the input stream and the stream's output temperature is set equal to that of the reboiler, but the results are incorrect. The reboiler can be used with inputs of any composition as long as all components are defined in the input data. A pressure drop across the reboiler may also be set.

The routine's flowsheet is shown in figure 8-3.

Figure 8 - 3 : Flowchart of routine DISCOL.

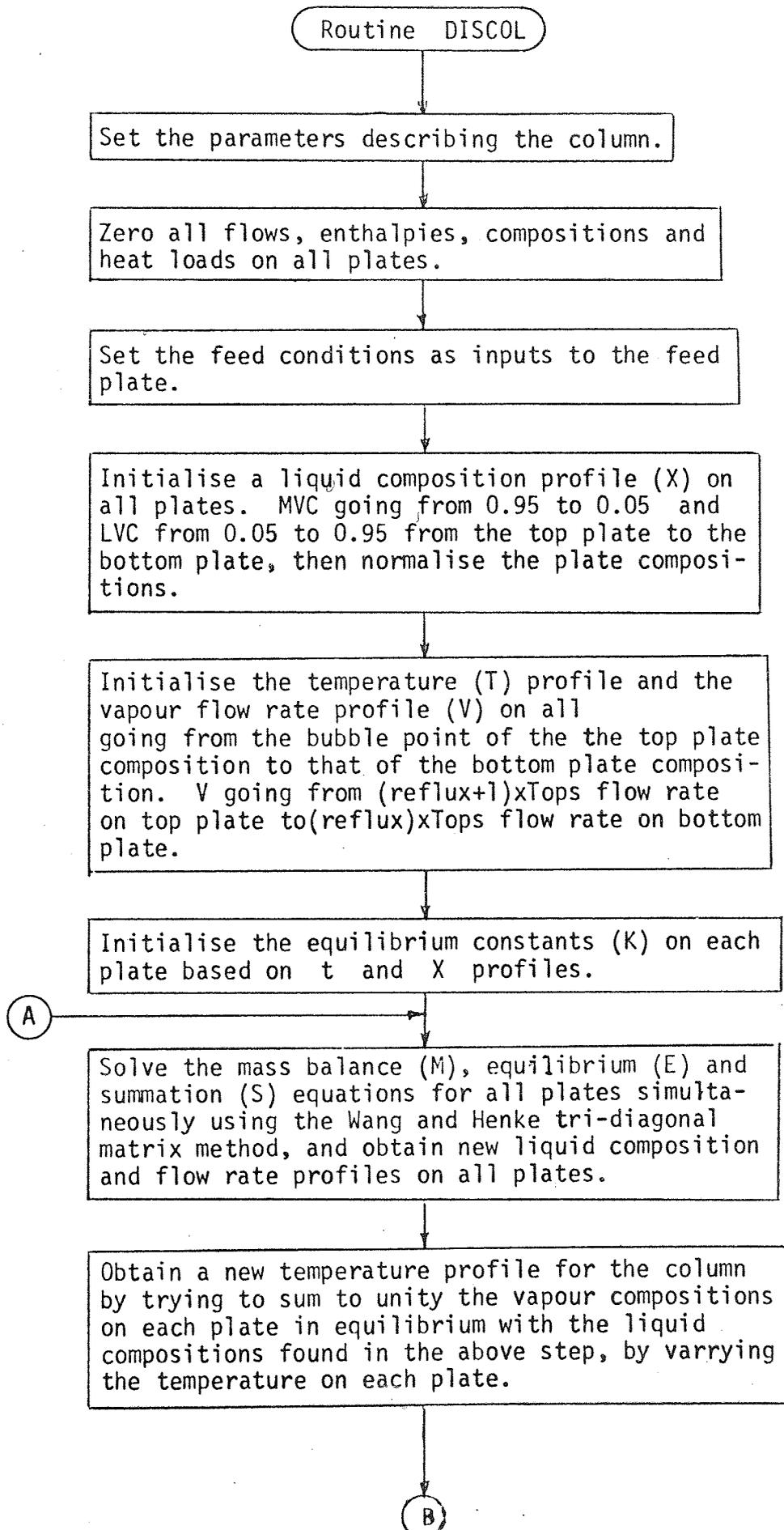
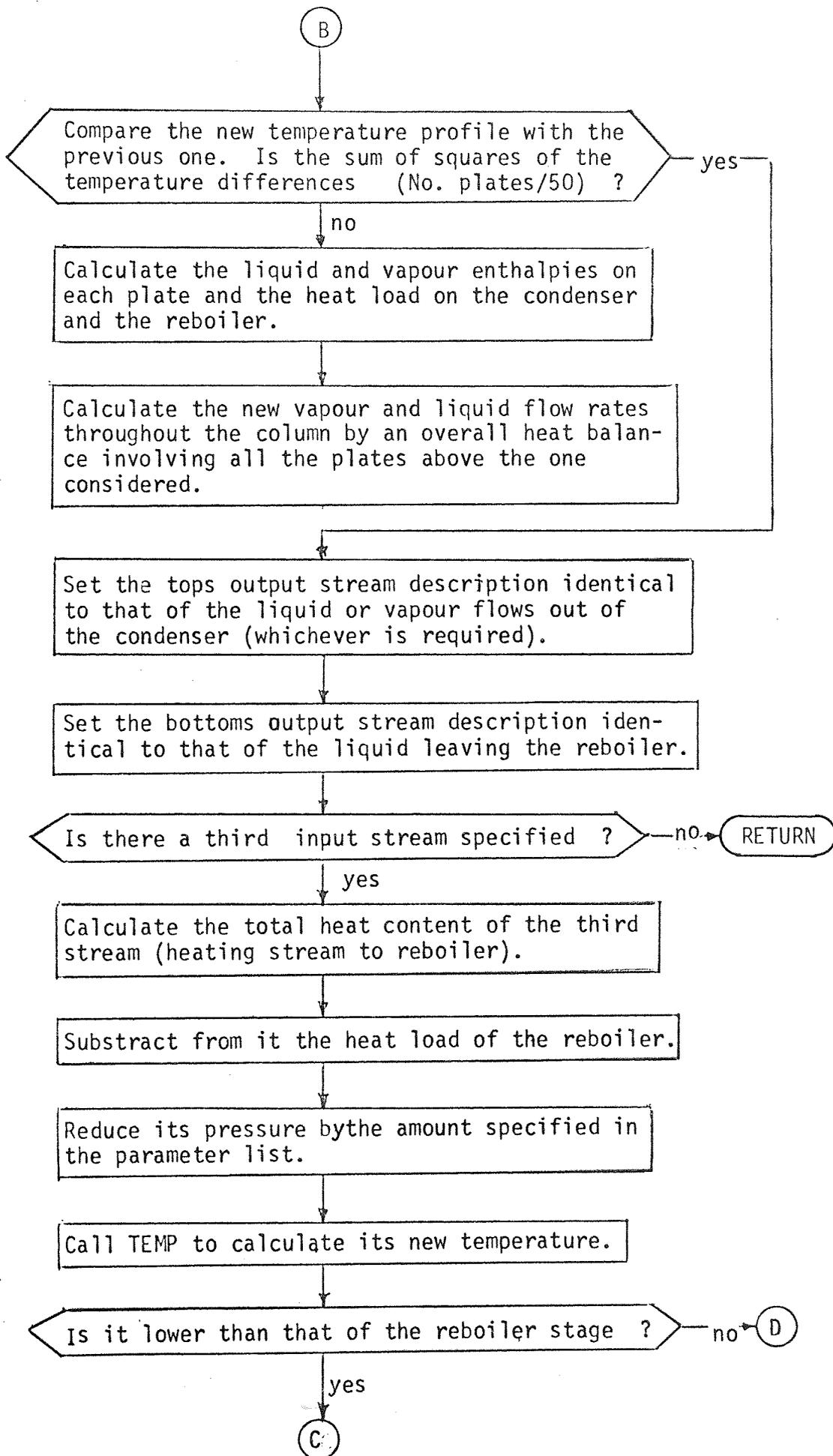


Figure 8 - 3 (Cont'd)



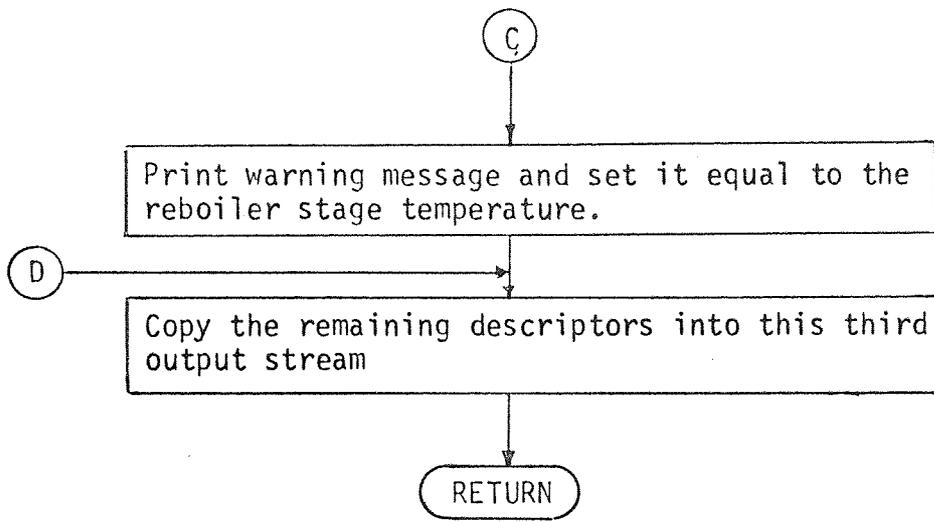


Figure 8 - 3 : Flowchart of routine DISCOL.

(Cont'd).

#### 8.14 The SETDP and SETBP routines.

They may be used either at the beginning of a simulation to set the dewpoint or bubble point of a stream assumed to be at these temperatures, thus relieving the user from the task of calculating them by hand, or they may be used internally as short-cut methods to force these conditions on the streams or to replace more complicated models leading to similar results, such as heaters, boilers, condensers, etc.

They call routine DEWBUB to calculate the appropriate temperature at the input stream's conditions and routine ENTHALPY sets its enthalpy and a fully described stream is output.

#### 8.15 Conclusion.

All the unit operation models presented here are basic but fairly general and accurate within the limitations imposed by their objectives, their parameters and the thermo-physical data. Their apparent simplicity stems from their complete dependence on the property packages.

Their extension to model more advanced or complex operations or the addition of design sections should create no extension or linking problems because of the high modularity of the models and the availability of many property generation routines.

## CHAPTER 9

### THE THERMODYNAMIC AND PHYSICAL PROPERTIES PACKAGE

The thermodynamic and physical property evaluation routines are an essential complement to the unit operation models just presented. The models perform the heat and mass balances, i.e. they add or divide heat and mass flow estimates, but they rely entirely on a different set of routines to compute these estimates or supply the physical constants needed to perform the balances.

These properties and constants may be divided in two categories:

- A. The thermodynamic and thermal properties such as the vapour-liquid equilibrium ratios (K-values), enthalpies, dew points and bubble points, phase separation, etc....
- B. The physical properties of pure components and mixtures such as density, specific heat, viscosity, thermal conductivity and surface tension amongst others.

#### 9-A The thermodynamic and thermal routines

This first type of data are the most important ones in heat and mass balances and are sufficient in helping to specify completely the state of pure components and mixture systems. They have been set in a number of routines each one fulfilling one purpose only. The main routines are:

KVALUE : To supply vapour-liquid equilibrium ratios.

ENTHALPY : To compute a system's enthalpy.  
TEMP : To set its temperature knowing its enthalpy.  
FLASH : To separate a two-phase flow into its pure phases.  
DEWBUB : To calculate the bubble point or dew point of a mixture.  
VAPRES and BOILPT : To calculate the vapour pressure or boiling  
point of pure components.  
TOMASS, TOMOLE, TOSI, TOBRIT : unit-conversion routines.

The properties most easily gathered in general packages are those of hydrocarbons since their behaviour is fairly similar and easily predictable over wide temperature and pressure ranges. They are also the compounds most commonly processed in the petrochemical Industry for which the models in chapter 8 are most suitable. These property routines may however be replaced by others covering a wider classification of chemicals as long as their names remain identical.

Pure compounds exhibit occasionally properties or behaviours different to those of mixtures, especially in the two-phase region. Their deviations are mentioned below where relevant and particular care has been taken to avoid erroneous or unpredictable situations.

A list of the chemical compounds covered in this package is given in table 9-1 and the property data constants needed in the above-mentioned routines are listed in table 9-2.

Table 9 - 1 : Chemical compounds available in  
PEETPACK's property file.

Classification	Compound name	Synonyms
Non-hydrocarbon	Hydrogen	
Paraffins (alkanes)	Methane; Ethane Propane N-Butane Iso-Butane N-Pentane Iso-Pentane Neo-Pentane N-Hexane N-Heptane N-Octane N-Nonane N-Decane N-Undecane N-Dodecane N-Tridecane N-Tetradecane N-Pentadecane N-Hexadecane N-Heptadecane	2-Methylpropane  2-Methylbutane 2,2-Dimethylpropane
Olefins (alkenes)	Ethylene Propylene 1-Butene Cis-2-Butene Trans-2-Butene Iso-Butene 1-Pentene Cis-2-Pentene Trans-2-Pentene	Ethene Propene  2-Methylpropene

	2-Methyl-1-Butene 3-Methyl-1-Butene 2-Methyl-2-Butene 1-Hexene	
Diolefins	1,3-Butadiene	
Naphthenes	Cyclopentane Methylcyclopentane Cyclohexane Methylcyclohexane	
Aromatics	Benzene Toluene Ethylbenzene O-Xylene M-Xylene P-Xylene	Methylbenzene  1,2-Dimethylbenzene 1,3-Dimethylbenzene 1,4-Dimethylbenzene

Table 9 - 1 : Chemical compounds available in  
PEETPACK's property files.  
(Cont'd)

Property No. in PROPRT	Symbol	Meaning	Units in Library
1	-	Component Name	-
2	-		-
3	KODE	Component No. in Library	-
4	M.W.	Molecular Weight	-
5	$P_c$	Critical Pressure	P.s.i.a.
6	$T_c$	Critical Temperature	Deg. R
7	$V_c$	Critical Volume	ft <sup>3</sup> /lbmole
8	$T_{bp}$	Normal Boiling Point	Deg. R
9	$\omega$	Acentric Factor (60)	-
10	$\delta$	Solubility Parameter (60)	(cal/ml) <sup>1/2</sup>
11	V	Liquid Molar Volume (60)	ml/gmole
12	$n_a$	Redlich-Kwong Equation	-
13	$n_b$	of State Parameters	-
14	a	Ideal Enthalpy Parameters	btu/lbmass
15	b	in the Equation :	"
16	c	$H^\circ = a + b \times T + c \times T^2$	"
17	d	$+ d \times T^3 + e \times T^4$	"
18	e	$+ f \times T^5$ (T in °F)	"
19	f	where H is per unit mass	"
20	A	Vapour Pressure Constants	-
21	B	in the Equation : $\text{Log } P(\text{mm Hg}) = \frac{B + 0.2185 \times A}{T \text{ } ^\circ\text{K}}$	-

Table 9 - 2 : Data set in PROPRT Data file.

## 9.1 Routines KVALUE and RKDATA

Vapour-liquid equilibrium is the most important and the most common phenomenon that occurs in almost all unit operations, and modelling any unit operation involves these calculations in some way or other. Whether it be distillation, mixing, dew point calculation, etc., they all require K-values over a wide range of pressure, temperature and compositions. There are probably over 15 different ways of calculating these equilibrium ratios (52, 84, 85) for ideal systems, petroleum fractions, cryogenic conditions, low pressure systems, etc., but no single method can cover adequately more than one or two types of systems, and few are suitable for repetitive calculations in flowsheeting simulators because either they are too long to compute or their storage requirements are excessively large. The well known and well tested Chao-Seader method (60) was chosen because it combines fair accuracy with moderate storage requirements and acceptable speed. Its inclusion in all the best simulators is an adequate guarantee of its suitability for hydrocarbon systems.

A very simple polynomial-fit K-value routine (EVALUE) has also been provided for users wishing to input their own K-value correlations to overcome the longer KVALUE calculations. This routine has been described in chapter 8.

In KVALUE, the vaporisation equilibrium ratio (K) is calculated by the equation:

$$K = \frac{v^{\circ} \delta}{\phi} \quad (9.1-1)$$

The liquid fugacity coefficient,  $v^{\circ}$  is a pure compound liquid property correlated by Pitzer (86) and the liquid activity coefficient  $\delta$  is calculated from Hildebrand's equation for regular solutions (87). The vapour phase fugacity coefficient  $\phi$  is calculated from the Redlich-Kwong equation of state (61):

$$P = \frac{R T}{v-b} - \frac{a}{T^{0.5} v(v+b)} \quad (9.1-2)$$

The calculation of the K-values relies on a knowledge of both the vapour and the liquid phases compositions. Since these are not usually known at the beginning of a K-value calculation, an iterative method is followed to compute them simultaneously with the K-value. In this routine the K-values may be calculated from an initial knowledge of either phase composition. The other phase composition is initially equated to the known composition and the K-values calculated.

The liquid mixture solubility parameters and the vapour phase constants for the equation of state are calculated initially but at successive iterations the parameters of the phase of changing composition only are calculated. The liquid fugacity ( $v^{\circ}$ ) which is a function only of the reduced properties is calculated once only at the beginning. The other two variables ( $\delta, \phi$ ) vary with composition and are continuously updated but the number of logarithm and exponential calculations has been minimised

by rearranging the given equations (88) to save computing time.

Following the calculation of the K-values, the guessed phase composition is updated by the appropriate set of equations:

$$Y_i = K_i X_i \quad \text{or} \quad X_i = Y_i / K_i \quad (9.1-3)$$

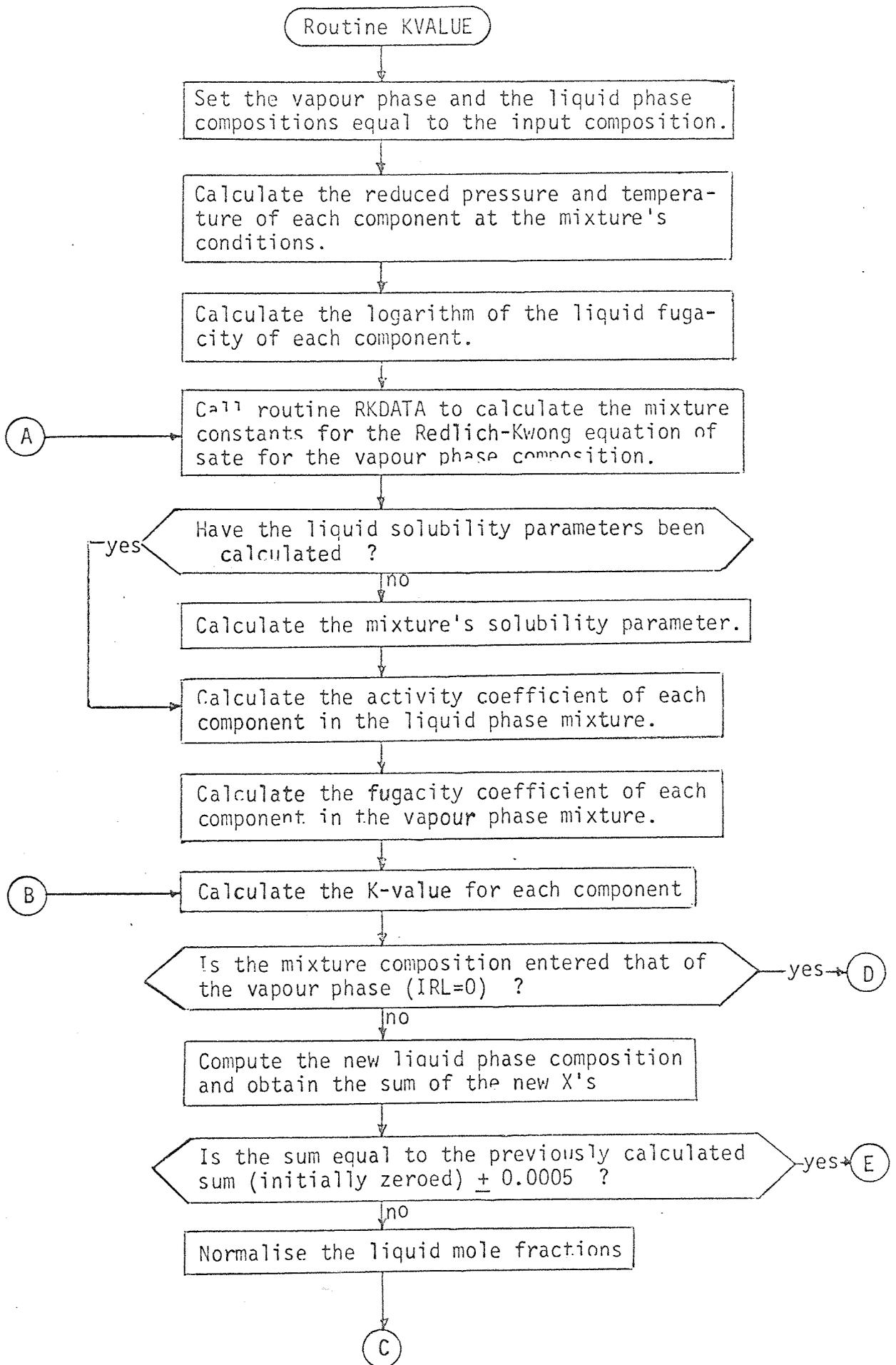
where  $X_i$  and  $Y_i$  are the individual liquid and vapour molar fractions. The updated compositions are used again to compute improved K-values. The procedure is repeated until the summation of the mole fractions of the updated phase does not change by more than  $5 * 10^{-4}$  over two consecutive iterations. At this point consistent K-values and phase compositions exist based on the input temperature, pressure and the set phase compositions. These however are not necessarily correct values since the input phase composition may not physically exist at the given temperature and pressure. In this case the mole fractions of the newly calculated phase do not add up to unity. An external trial and error solution for the temperature or composition giving overall consistency in both phases is required, as in the distillation column and the flasher routines respectively.

Pure component systems are given a K-value of unity and identical phase composition. Temperature and pressure checks are also included to warn the user of out-of-range conditions (60).

Routine RKDATA computes the constants for the Redlich-Kwong equation of state according to its mixing rules (61).

The accuracy of the method has only been tested for light

Figure 9 - 1 : Flowchart of routine KVALUE.



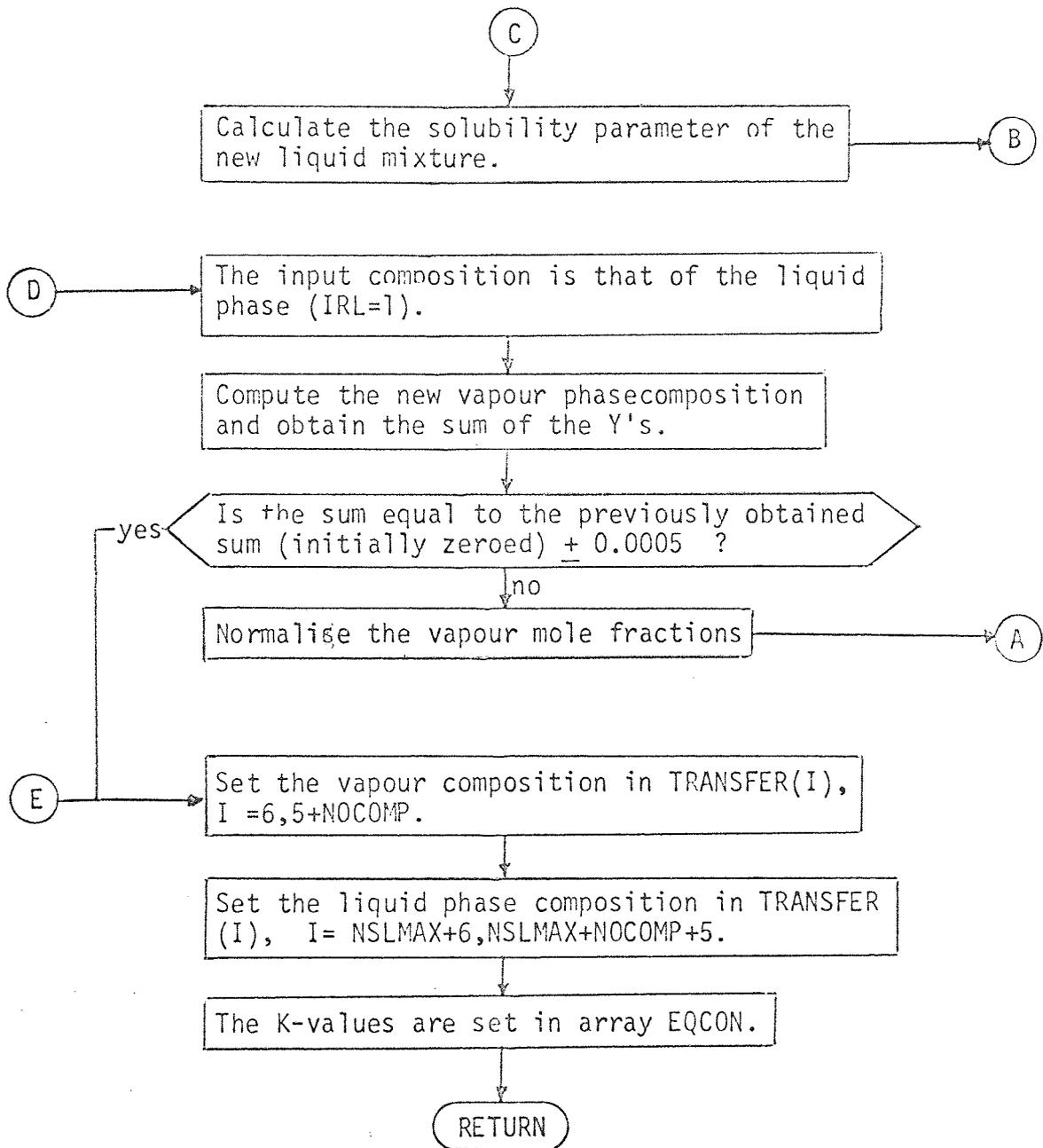


Figure 9 - 1 : Flowchart of routine KVALUE.

(Cont'd)

6

hydrocarbons and gives dew point calculation deviations of 3°C at 15 bars.

The routine is called by the statement:

```
CALL KVALUE (I, TRANSFER, EQCON)
```

where I = 0, when the vapour phase description is given in (TRANSFER (J), J = 1, NSIMAX) as base composition.

I = 1, when the liquid phase description is given in (TRANSFER (NSIMAX + J), J = 1, NSIMAX).

I = -1, when both phase descriptions are given initial estimates but the system is treated as for I = 1.

TRANSFER contains initially the description of the known phase but on return from KVALUE, it will contain the description of both phases in equilibrium and EQCON (DIMENSIONED at 10) returns the K-value for each component in the order they were read into the system.

The data required here for each component are: its molecular weight and critical properties, its accentric factor, its Hildebrand solubility parameters and the Redlich-Kwong equation of state constants  $\omega_a$  and  $\omega_b$  from which a and b are calculated (89) and for which the default values of 0.4278 and 0.0867 are set (61) if not available.

A flowchart of the routine is shown in figure 9.1.

## 9.2 Routines ENTHALPY, HDATA and HSAT

Routine ENTHALPY is another important thermo-physical

property evaluation routine. It computes the enthalpy of streams or systems otherwise specified and is essential in all heat balances. Such a routine must be able to compute the enthalpy of vapour, liquid and two-phase mixtures over wide temperature and pressure ranges with good accuracy and for a large number of components. These requirements can be met in one of two ways:

1. By accurate equations of state transformed to calculate enthalpies.
2. By generalised equations also based on the corresponding-state principle but avoiding the need for P-V-T calculations.

Both methods have their advantages and disadvantages. The equation-of-state approach can be very accurate, but the more accurate the equation used, the more complex and time consuming it becomes. Iterative Specific-Volume calculations are needed for each enthalpy calculation and these equations usually apply only to the vapour phase. For the liquid phase, saturation isotherms are normally used since very few equations of state can map accurately the sub-cooled region. This region is thus poorly modelled. The generalised equations using the corresponding-state approach model fairly well both the superheated vapour phase and the subcooled liquid region over wide pressure and temperature ranges, but the critical region stands less well the test. The enthalpies are generally more consistent in both phases but less accurate in the vapour phase than those calculated by the equations of state.

This second approach was chosen because of its better

treatment of the liquid phase and its overall speed. In this approach only two methods have been widely accepted:

1. The Yen and Alexander method (64).
2. The Hirschfelder method (90, 91, 92).

The Hirschfelder did not suit the needs of the simulation package because of its excessive complexity and length. It also suffers from the following problems:

1. It requires an accurate knowledge of the density of both phases, for which long trial and error calculation methods are proposed.
2. It is not suitable for non-hydrocarbons.

The Yen and Alexander method does not claim high accuracy but is much simpler and faster despite the fact that the calculations must be repeated at two discrete values of the compressibility factor and linear interpolation applied between them. It has instead the advantage of applying to non-hydrocarbon gases and water although mixtures involving non-hydrocarbons give poorer results.

Their method computes the enthalpy-departure from the ideal zero-pressure enthalpy based on the improved Lydersen-Greenkorn-Hougen enthalpy departure charts for pure compounds (93). The correlations developed for pure components can be used to estimate enthalpies of mixtures. Comparisons between the calculated data and the experimental results show good agreement. The average deviation of the improved charts from experimental

results is about 5 btu/lb increasing to about 15 btu/lb at the critical region. The routine operates as follows. It checks the input mixture (or pure component) vapour fraction. For systems of undefined quality (vapour fraction set to -1) their temperature is compared to their dew point, then if necessary to their bubble point to set their phase. Two-phase mixtures are separated into their pure phase constituents in routine FLASH and each phase is treated separately as a pure phase system.

The pseudo-critical and reduced properties for each pure phase system are calculated by Kay's rule and a flag set to define the type of enthalpy departure equations to use. Routine HDATA is called to compute the enthalpy departure and on return this value is added to the ideal gas zero-pressure enthalpy ( $H^0$ ) computed from polynomials fitted to the A.P.I. Project 44 tabulated data. The pure components enthalpies are weighted according to the mixture's mole fractions and summed up. In the case of two-phase mixtures, each phase is dealt with separately in turn and their enthalpy departure is summed up in proportion to their molar flow rates.

The equations representing the enthalpy departure have been classified in four sections: Superheated vapour, saturated vapour, saturated liquid and subcooled liquid regions. The equations used depend on the flag set above and on the vapour fraction in the mixture. Normally the superheated vapour and subcooled liquid equations are used as this saves checking the

mixture's conditions for saturation. The saturation isotherms are thus used as special cases of the superheated or subcooled states, but two-phase mixtures have their enthalpy departures calculated by the saturated-states equations. The use of the saturated isotherm equations may be specifically requested by adding the constant 2.0 to the vapour fraction in the TRANSFER array or calling routine HSAT instead of ENTHALPY. HSAT merely adds 2.0 to the vapour fraction and calls ENTHALPY.

Routine HDATA is a compilation of the enthalpy departure equations grouped according to region and critical compressibility factor ( $Z_c$ ). It selects the equations for the  $Z_c$  closest to that of the mixture and calculates the departure then repeats the calculations for the closest  $Z_c$  on the opposite side of the mixture's  $Z_c$ , and linear interpolation is used to obtain the final enthalpy departure.

Pure components are treated identically to mixtures except that for systems of undefined quality (vapour fraction) at their boiling point, they are treated as pure vapour since they are insufficiently defined.

ENTHALPY requires the pure components' molecular weights, critical constants and ideal enthalpy ( $H^0$ ) coefficients. The routine is called by the statement

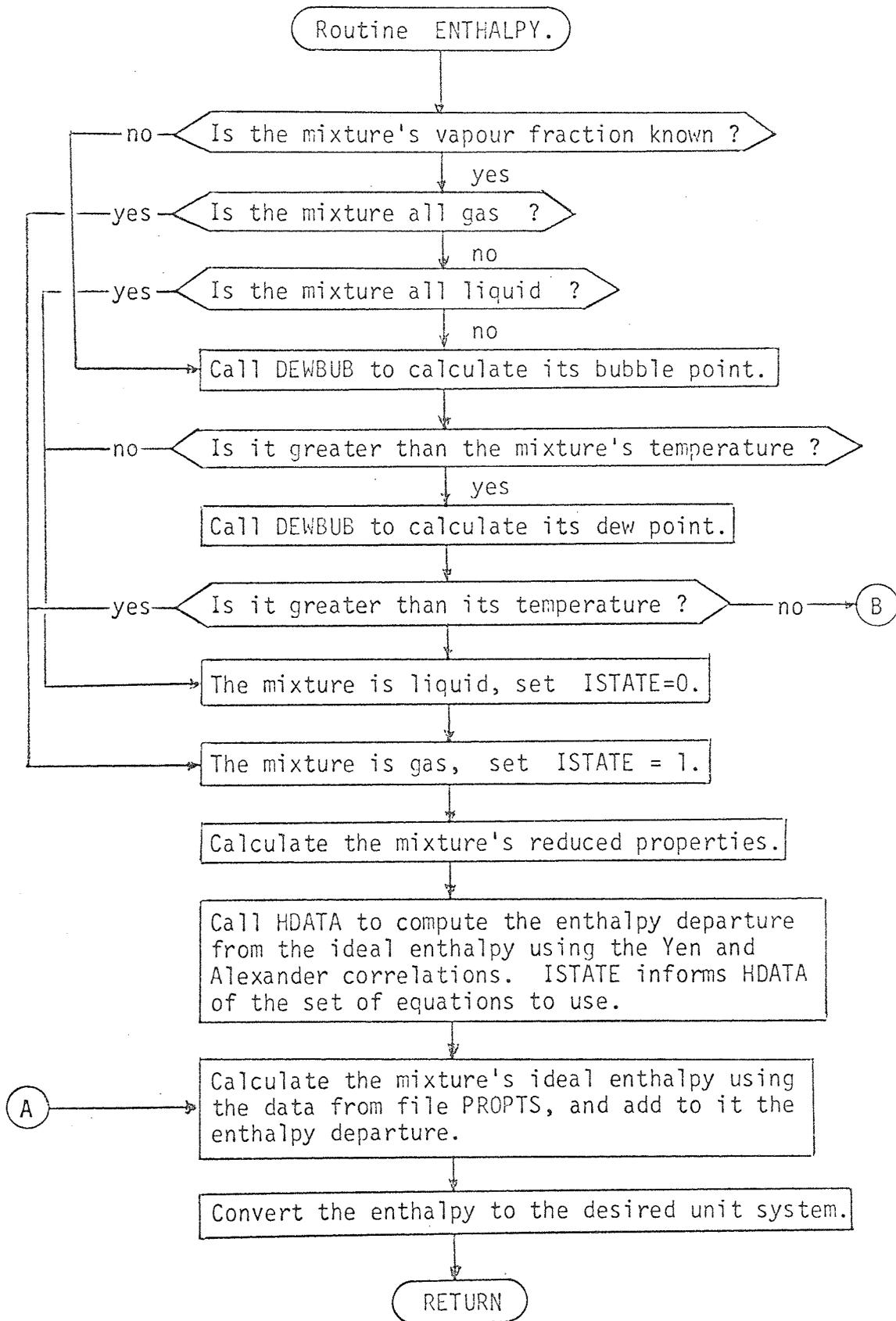
```
CALL ENTHALPY (TRANSFER)
```

or

```
CALL HSAT (TRANSFER)
```

where TRANSFER contains the stream or system description (in the

Figure 9 - 2 : Flowchart of routine ENTHALPY.



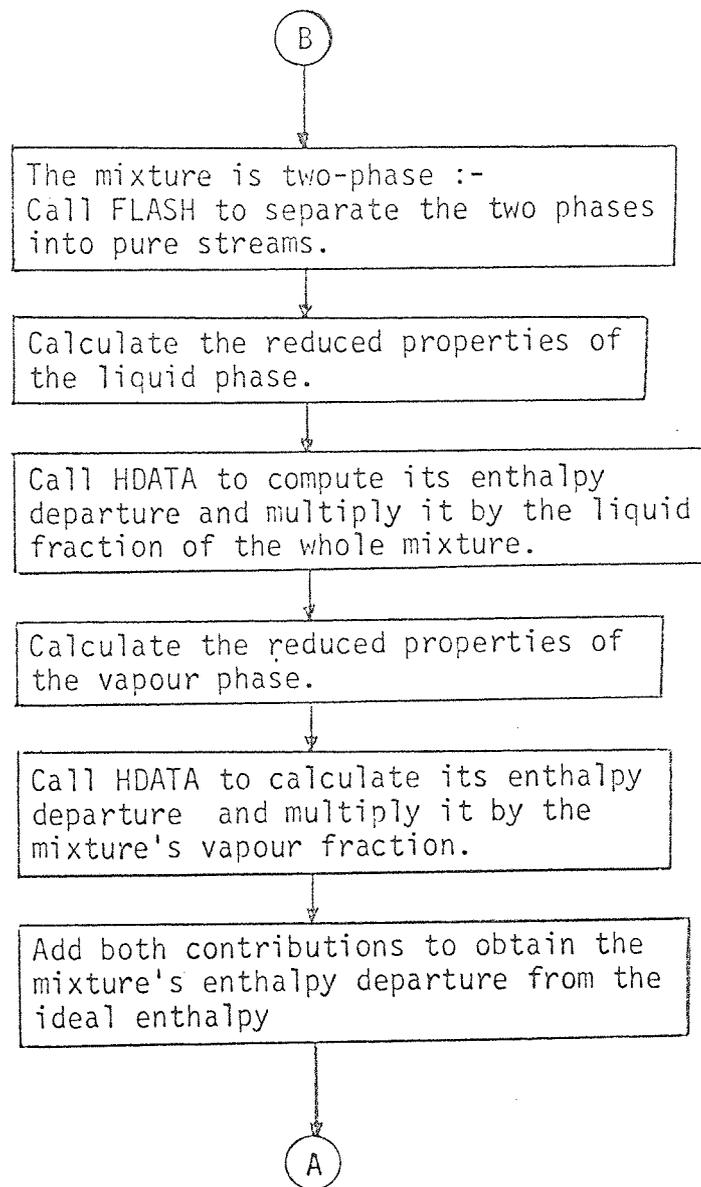


Figure 9 - 2 : Flowchart of routine ENTHALPY.

(Cont'd)

order used to describe streams) and its enthalpy is returned in element TRANSFER (4). A flowchart of the method is shown in figure 9-2.

### 9.3 Routines FLASH and PUREVF

FLASH separates an input stream of undefined or unset phase into two streams bearing the descriptions of the mixture's pure phase constituents at a given pressure. If at this pressure and at the stream's temperature and composition it is single phased, the output stream referring to the non-existing phase is zeroed. The routine is useful to separate a two-phase stream into its pure phases or to find the unknown phase or vapour fraction of a stream.

The flash method consists in an isothermal mass balance involving on the one hand the individual components in the output streams related by the equilibrium ratios (K-values) and on the other the input flow rate and composition. The mass balances are governed by the following sets of equations:

$$F * Z_i = F * R_L * X_i + F * (1 - R_L) * Y_i \quad (9.3-1)$$

$$\text{and } Y_i = K_i * Z_i \quad (9.3-2)$$

where  $F$  = Input molar flowrate,

$R_L$  = Liquid fraction in the input stream,

$Z_i$  = Individual components mole fractions in the input,

$X_i, Y_i$  = Liquid and vapour phases mole fractions,

$K_i$  = Vapour-Liquid equilibrium ratios.

The objective is to satisfy the mass balance obtained by rearranging these two equations and summing them over all components:

$$(1-RL) * \sum_i X_i K_i + RL * \sum_i X_i = \sum_i Z_i = 1 \quad (9.3-3)$$

Equation (9.3-1) is rewritten as:

$$X_i = Z_i / (RL + (1-RL) * K_i) \quad (9.3-4)$$

and used in the following to compute RL.

- Q - Initially all X's and Y's are set equal to Z's and estimates of  $K_i$ , based on the X values are obtained from routine KVALUE.
- b - Equation (9.3-4) is solved for all  $X_i$ 's based on these K values and on an initial estimate of  $RL = 0.2$ .
- c - To calculate a better estimate of RL, the Newton-Raphson search technique is used on equation (9.3-4) as follows:

$$G(RL) = \sum_i X_i - \sum_i (Z_i / (RL + (1-RL) * K_i)) = 1 \quad (9.3-5)$$

$$\partial G / \partial RL = \sum_i ((X_i * (1-K_i)) / (RL + (1-RL) * K_i)) \quad (9.3-6)$$

$$RL^* = RL - G(RL) / (\partial G / \partial RL) \quad (9.3-7)$$

where  $RL^*$  = new liquid fraction estimate,

and  $G(RL)$  = a function of the variable RL.

- d - New estimates for  $X_i$  are again computed by equation (9.3-4) and summed up and compared to unity.
- e - The procedure is repeated until either the sum of  $X_i$  reaches unity ( $\pm 10^{-5}$ ), in which case an answer has been obtained, or until the number of iteratives exceeds 30, then the output is assumed all liquid as a default answer.

The following checks also may terminate the search:

1. RL exceeds  $10^7$  : the stream is assumed all liquid,
2. RL reached  $0 \pm 10^{-4}$  : the stream is all vapour,
3. RL reaches  $1 \pm 10^{-4}$  : the stream is all liquid,
4. The function G and its derivative with respect to RL are both positive : experience has shown that no solution exists but the case occurrence is highly improbable. The run is abandoned.

Once the solution is obtained, the output streams are set in TRANSFER (I),  $I = 1$ , NSLMAX, for the vapour phase and TRANSFER (NSIMAX + 1) for the liquid phase. The vapour fraction is set to 1 for the vapour phase and to 0 for the liquid phase and the input flow rate is divided according to the vapour fraction  $(1-RL)$  and the liquid fraction.

In a pure component system, its dew point (boiling point) is calculated and compared to its input temperature. The stream is set to liquid if its temperature is below the boiling point, or to vapour if it is above it. If however it is equal, function PUREVF is called to compute the vapour fraction based on its enthalpy. In PUREVF, the difference between the stream enthalpy and that of the pure liquid at its boiling point is divided by the heat of vapourisation to provide the vapour fraction.

In either system case, the following statement calls the routine:

```
CALL FLASH (TRANSFER, PRES)
```

Figure 9 - 3 : Flowchart of routine FLASH.

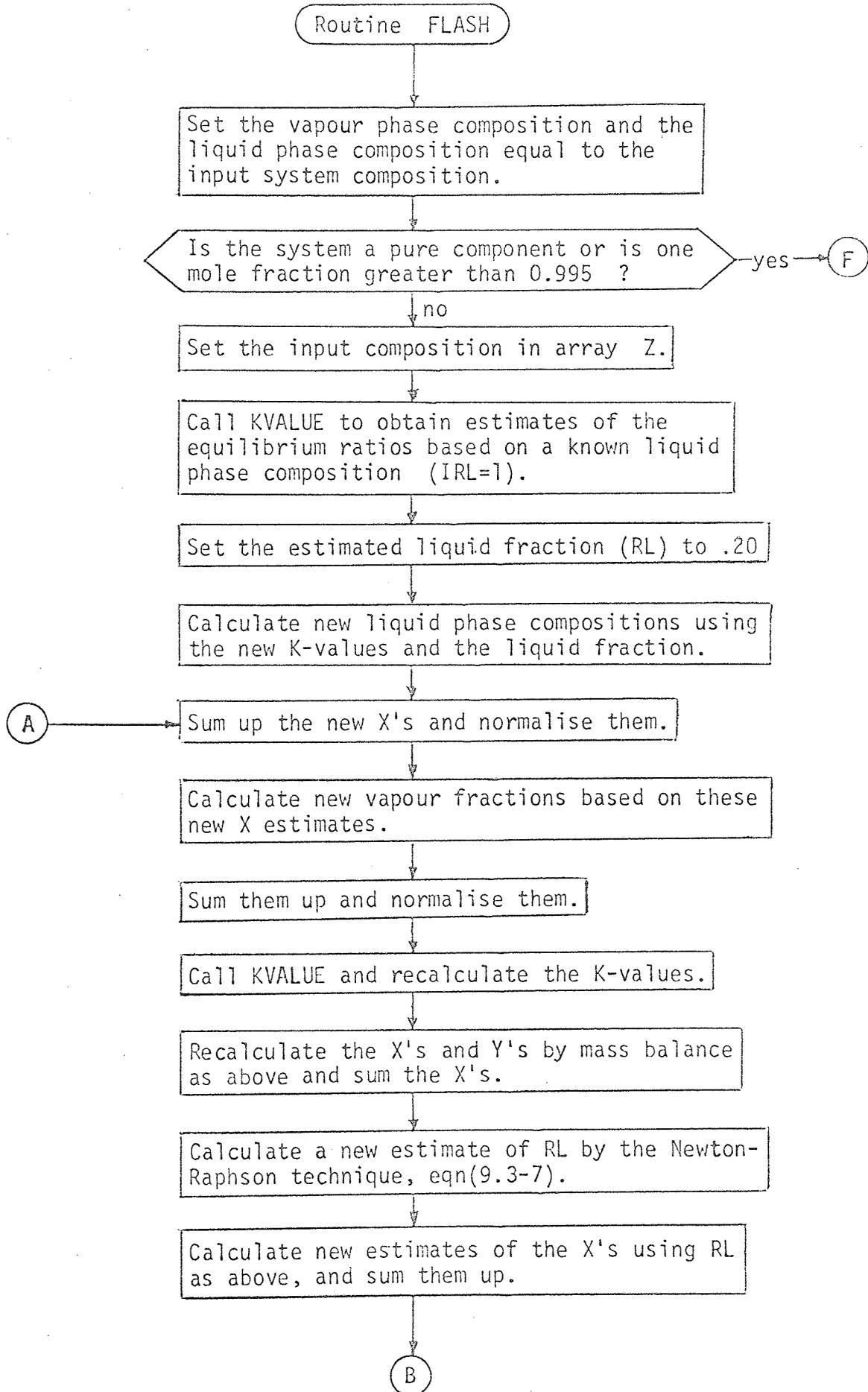
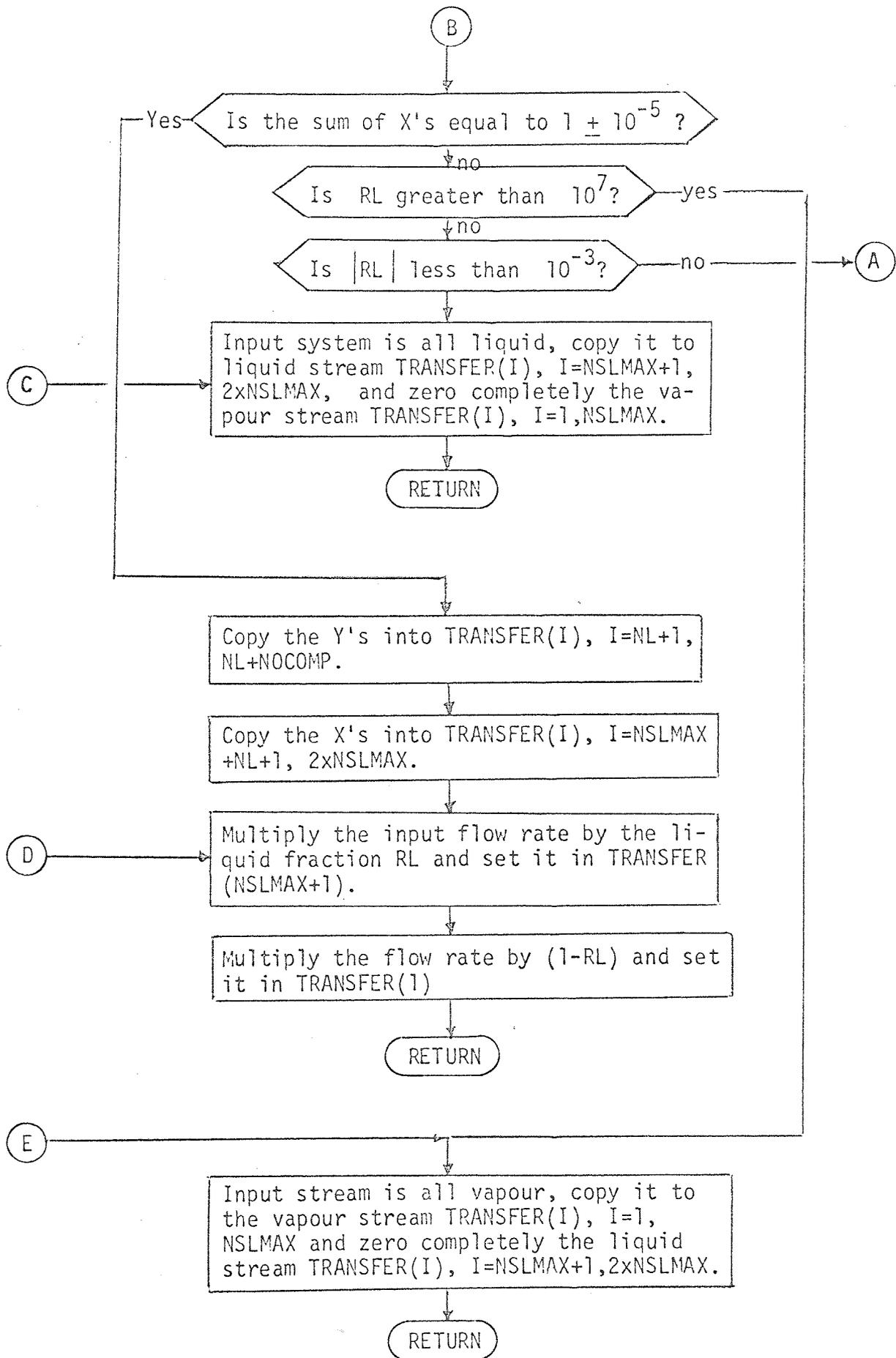


Figure 9 - 3 (Cont'd)



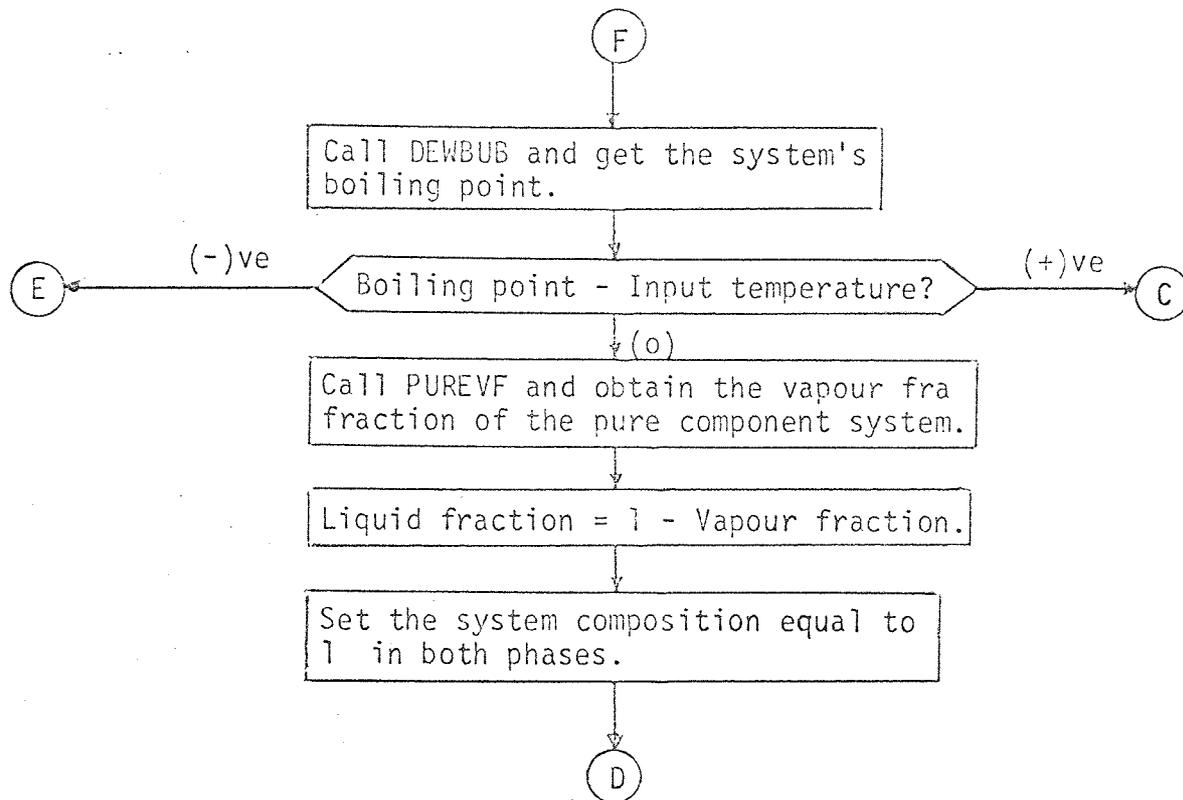


Figure 9 - 3 : Flowchart of routine FLASH.

(Cont'd)

where PRES is the pressure at which the system is to be checked. On return TRANSFER will have doubled its length and should be handled accordingly as mentioned above. The input stream is not returned.

The routine requires no data for itself but calls KVALUE to supply the equilibrium constants.

The flowchart for routine FLASH is shown in figure 9-3.

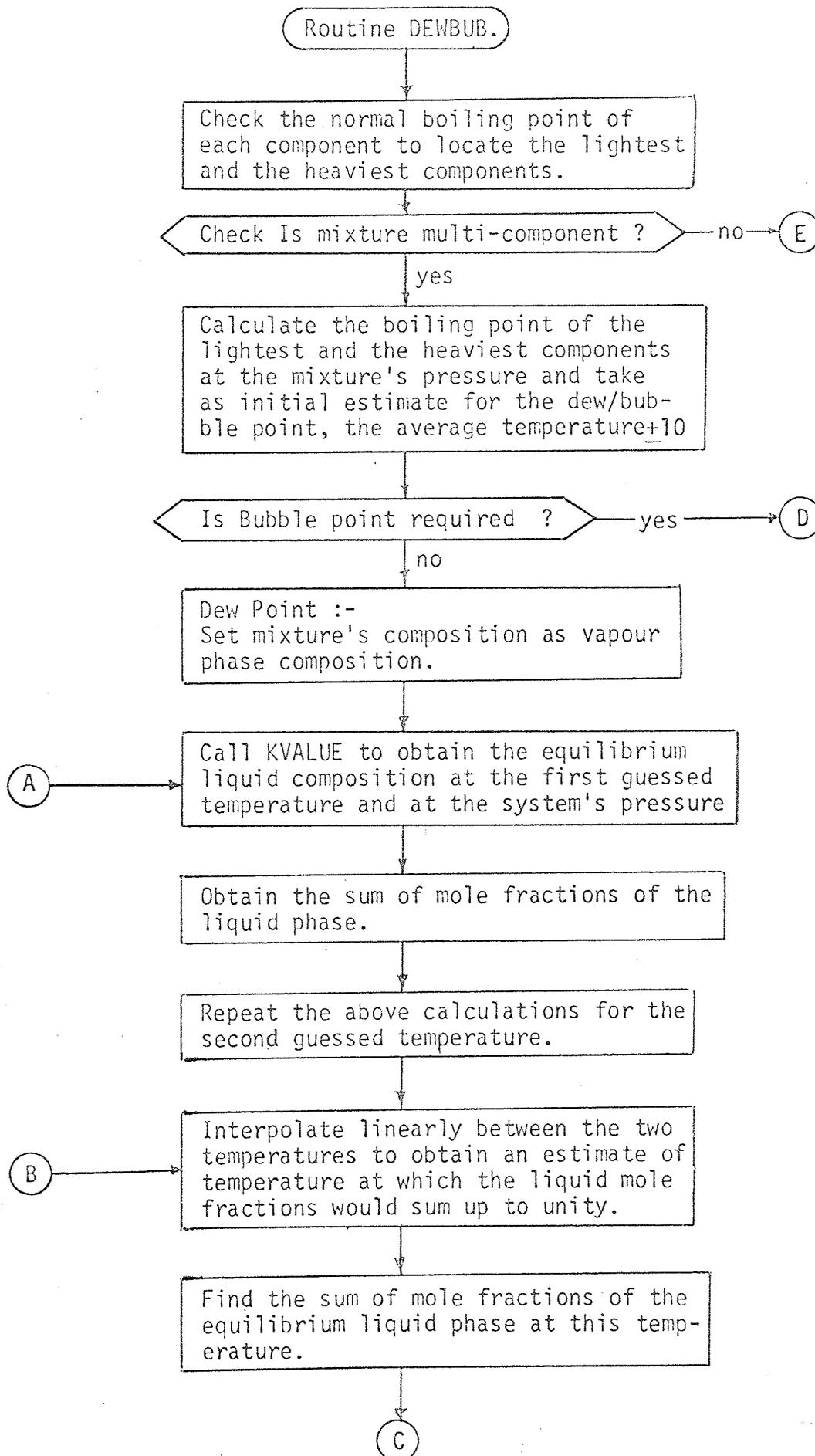
#### 9.4 Routine DEWBUB

Routine DEWBUB calculates the dew point or bubble point of a stream at its given pressure.

It is based on the theory that for a stream to be at its dew point or bubble point, it must be in equilibrium with a mixture in the other phase (liquid or vapour respectively) and therefore the mole fractions in that equilibrium mixture must add up to unity. For pure substances or mixtures where one component exceeds 99% mole fraction, the boiling point of this component is calculated (by calling function BOILPT), and returned since its bubble and dew point are identical (or very close). The following method is used.

The boiling points of all components are calculated at the stream's pressure and the average of the highest and lowest temperatures is computed. Two temperatures on either side of it are used as initial limits on the Reguli-Falsi iterative technique. At each of these two temperatures, the vapour-liquid equilibrium

Figure 9 - 4 : Flowchart of routine DEWBUB.



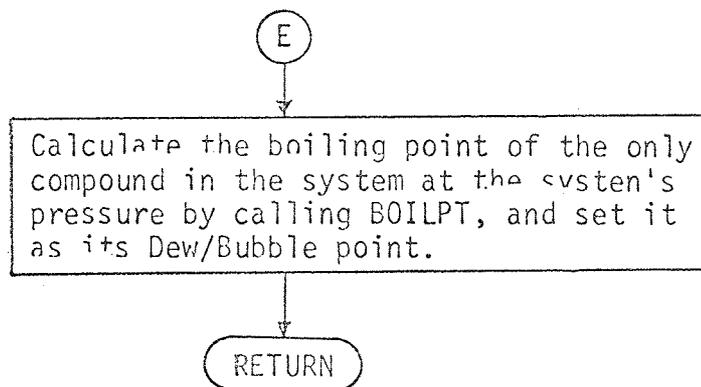
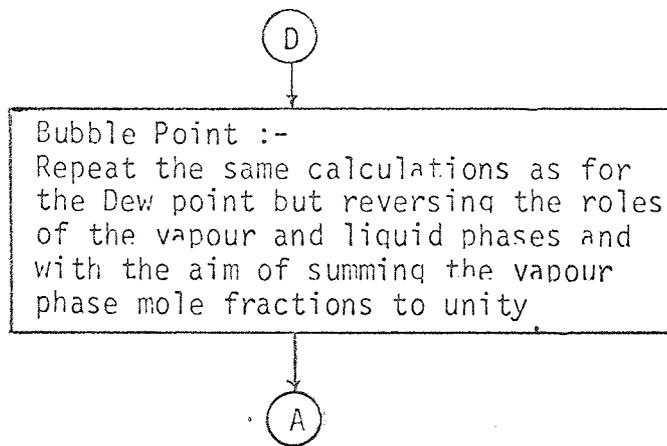
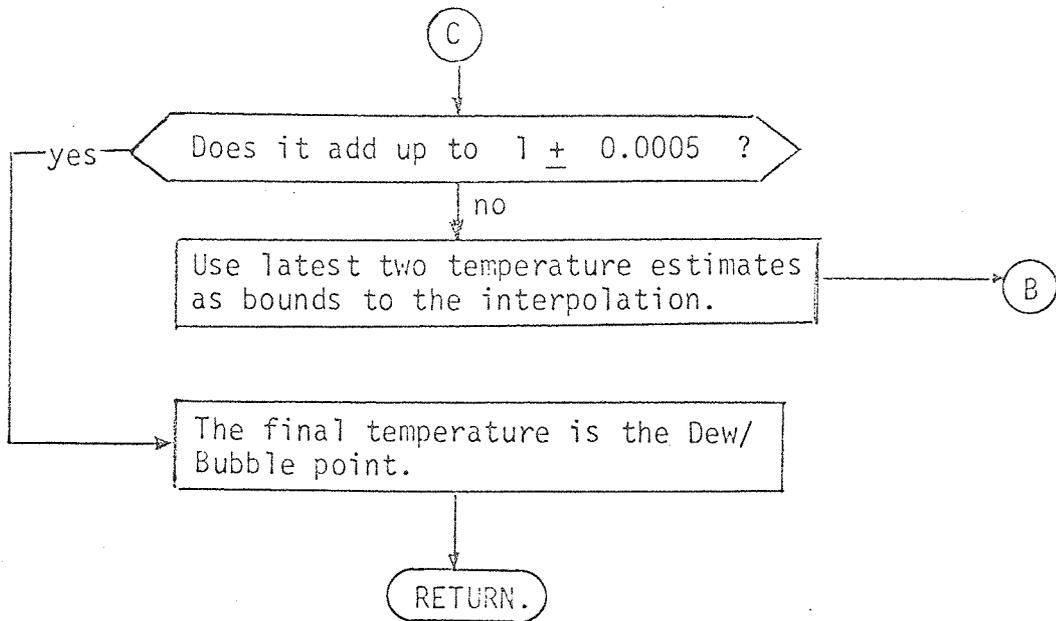


Figure 9 - 4 : Flowchart of routine DEWBUB.

(Cont'd)

ratios are calculated using the stream's composition either as correct vapour compositions (for the dew point) and obtaining the K-values based on them at the assumed temperatures, or as correct liquid compositions (for the bubble point). The liquid or vapour mole fractions are computed and summed up. The Reguli-Falsi method uses these sums and the initial temperatures to estimate a new temperature which would give a liquid or vapour phase composition sum of unity. The older temperature is rejected and replaced by the new one and new K-values calculated at this temperature. The iterative procedure is repeated until the mole fractions in the equilibrium phase sum up to unity ( $\pm 0.5 \times 10^{-3}$ ). This temperature represents then the dew point or bubble point of the mixture. It is returned as the variable TF in the statement

```
CALL DEWBUB (IRL, TRANSFER, TF)
```

where IRL = 0 if the dew point of the stream described in TRANSFER is required,

or IRL = 1 for its bubble point.

The routine estimates fairly accurately the dew and bubble points, but the accuracy, which diminishes as the pressure rises, is only a function of the KVALUE routine which is the only source of thermo-physical data.

The flowchart for DEWBUB is shown in figure 9-4.

#### 9.5 Routine TEMP:

This routine computes the temperature of a stream otherwise

described by its enthalpy, pressure and composition. It is based on a trial and error computation of the enthalpy at assumed temperatures. The temperature giving an enthalpy within 5 btu/lbmole of the input enthalpy is set as the stream's temperature.

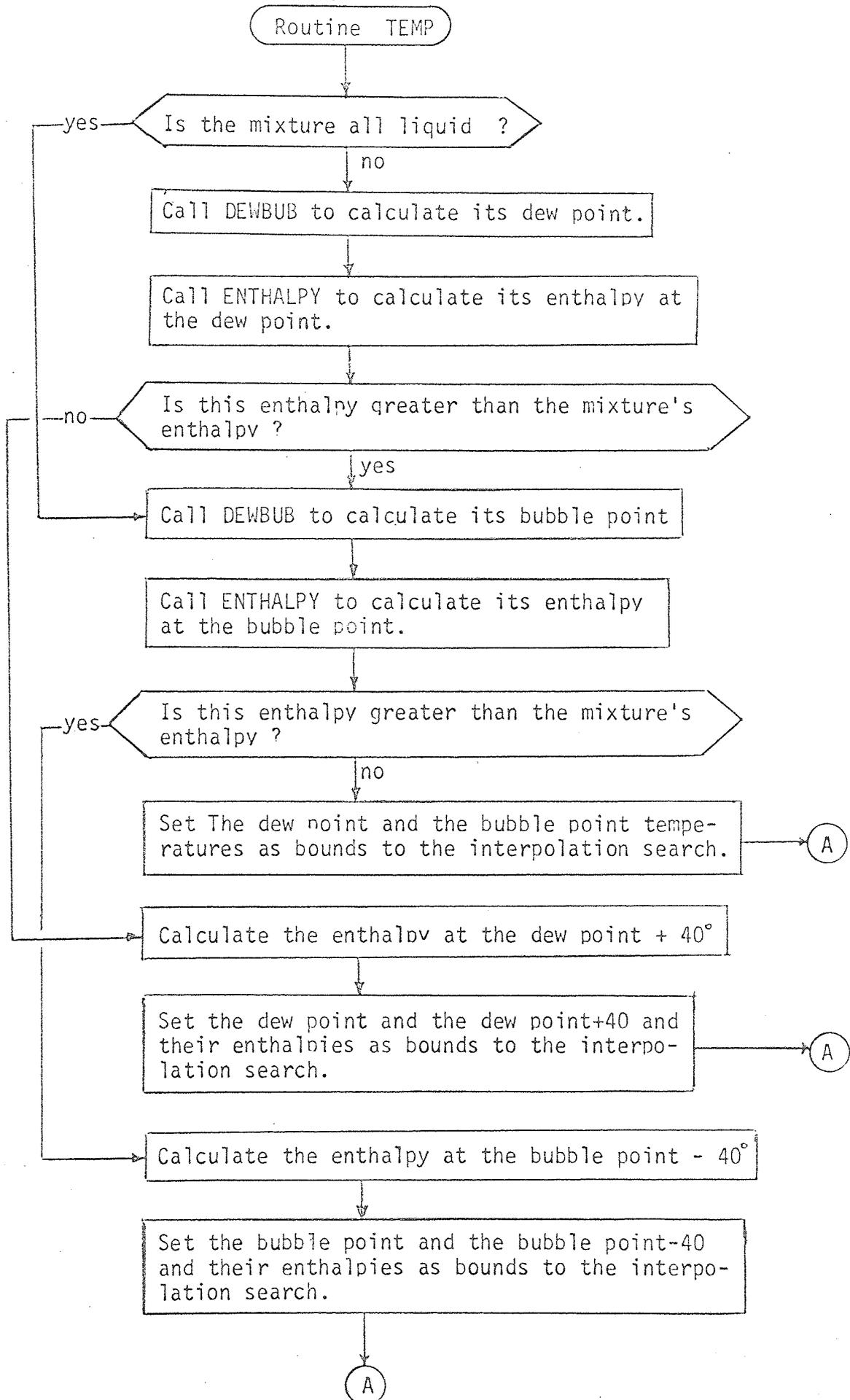
The routine uses an unaccelerated Reguli-Falsi iteration technique on the temperature variable. A Lagrange interpolation was initially used by the author on CONCEPT, in its equivalent TEMP routine, but it proved less stable than a straight-forward Reguli-Falsi method, and was subsequently abandoned.

TEMP first checks the stream's vapour fraction if it is not set (V.F. set to -1.0) by comparing its enthalpy first to that at its dew point then to the enthalpy at its bubble point, if necessary. Once its phase is known, the routine applies the Reguli-Falsi iteration techniques between one of the following sets of temperatures:

- a. Dew point and dew point + 40<sup>o</sup>, for the vapour phase.
- b. Dew point and bubble point, for the two-phase region.
- c. Bubble point and bubble point - 40<sup>o</sup>, for the liquid phase.

The limits in cases a and c do not bind the search which can extrapolate beyond them in the superheated and subcooled regions respectively. A presetting of the stream phase (TRANSFER(5)) will avoid the phase search and the program branches out immediately to the relevant section to set the initial search limits. The routine relies on DEWBUB to provide the dew and bubble points and on ENTHALPY to compute its enthalpy based on the iteration

Figure 9 - 5 : Flowchart of routine TEMP.



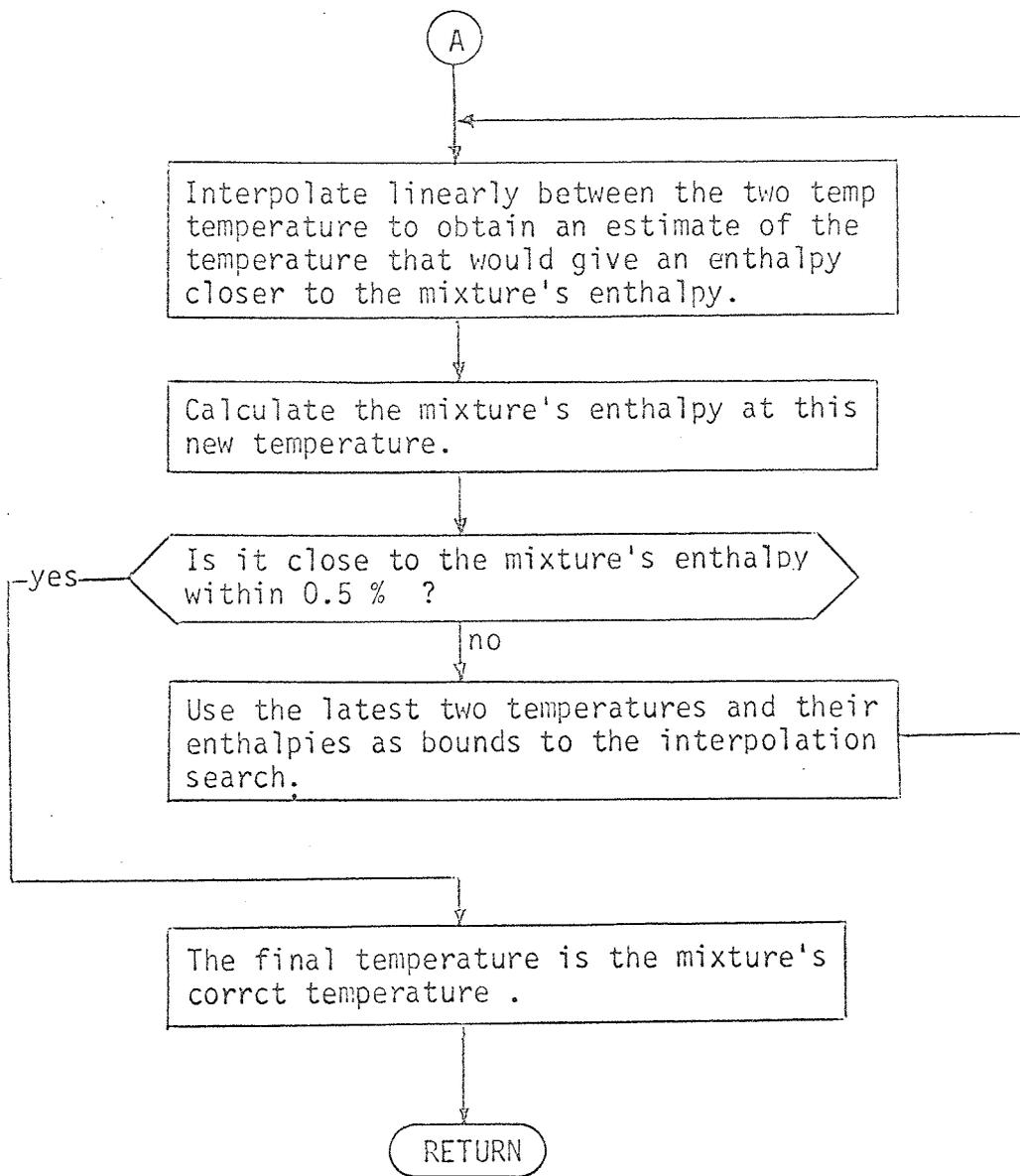


Figure 9 - 5 : Flowchart of routine TEMP.

(Cont'd)

temperature. In calling ENTHALPY the whole stream description is sent over and its vapour fraction is set to 0, 0.5, or 1 depending on the region of interest. The value 0.5 causes ENTHALPY to call FIASH to separate the two-phase mixture into its pure constituents then compute the mixture's enthalpy.

The convergence tolerance or the maximum number of iterations (15 iterations) control the length of this very slow search, and the final answer is the stream's temperature.

This routine may be used by calling it with the statement:

```
CALL TEMP (TRANSFER)
```

where TRANSFER contains the stream description except for the temperature, TRANSFER (2), left initially unset and set on return from TEMP. The temperature is returned either in degrees C or F depending on the unit system used in the simulation.

Pure components do not require any special treatment in this routine.

The flowchart of this routine is shown in figure 9-5.

## 9.6 Function VAPRES

The vapour pressure of a pure component at a given temperature, T, is obtained by an Antoine-type equation of the form:

$$\text{Log}_{10} P = (-0.2185 \times A/T) + B \quad (9.6-1)$$

where  $P$  = Vapour Pressure in mm Hg.

$T$  = Temperature in  $^{\circ}\text{K}$ .

$A, B$  = constants for pure components.

The equation, its constants  $A$  and  $B$  and its range of applicability for each component are provided in the Handbook of Chemistry and Physics (54th edition, P.D-162), but no references as to their accuracy or source are given there, and they have not been checked for the 45 components included in PEETPACK, because of time and computer budget restrictions. The temperature range of application for most components is fairly wide, however range extensions should be checked in the above reference for reduced temperatures above 0.6.

This function is used by calling VAPRES ( $I, T$ ) where  $I$  is the component sequence number as read into the program and  $T$  the temperature given in either  $^{\circ}\text{F}$  or  $^{\circ}\text{C}$  depending on the unit system used.

### 9.7 Function BOILPT

The boiling point of a pure compound at a given pressure PRES is calculated from the reciprocal of the Antoine equation used in function VAPRES. The same constants and limitations apply also here.

The function is used by calling BOILPT ( $I, \text{PRES}$ ) where  $I$  is the sequence number of the component. PRES should be given either in p.s.i.a. or bars depending on the unit system used.

## 9.8 Routines TOMOLE, TOMASS, TOSI and TOBRIT

These unit conversion routines operate on a stream or system described in array TRANSFER.

TOMASS converts the molar flow rate and composition to a mass basis using the components molecular weights.

TOMOLE converts the mass-basis flow rate and composition to a molar basis.

TOSI operates on a molar-flow system and converts its temperature, pressure and enthalpy from degrees Fahrenheit, p.s.i.a and B.t.u per pound mole to degrees centigrade, bars and K.Joules per Kgram mole, respectively.

TOBRIT converts a molar-flow system from S.I. units back to British units.

## 9.B The Physical property routines.

Many user prepared models and generally all mechanical design and specification routines will require additional physical properties of the chemical compounds used. Five property routines have been included in PRETPACK after having been used successfully in design models for heat exchangers and distillation columns in a joint project between the author and an M.Sc. student (58).

These routines are:

- DENSTY : to calculate the density of mixtures in any phase.
- SPHEAT : to calculate the specific heat of liquid or vapour phases.
- VISC : to calculate the viscosity of liquid or vapour phases.

Table 9 - 3 : Data Set in the PPTXTRA data file.

Property No. in PRTEXTRA	Symbol	Meaning	Units in PRTEXTRA
1	$T_{ch}$	Characteristic Temperature	$^{\circ}R$
or	-	in DENSTY ; or Blank	-
2	A	Constants for the liquid	-
3	B	density equations giving the	-
4	C	answer in gm/cc for an input	-
5	D	temperature in $^{\circ}F$	-
6	G		-
7	H		-
or 2	d	Liquid density at $T_{sp}$	gm/cc
3	$T_{sp}$	Temperature for given density	$^{\circ}F$
4 to 7	-	Blanks	-
8	a	Constants for the ideal specific	-
9	b	heat of gases, equation of the	-
10	c	form : $C_p = a(T/100) + b(T/100)^2 +$	-
11	d	$c(T/100)^3 + d(T/100)^4$ ; where $C_p$	-
		is in (btu/lb/ $^{\circ}F$ ) and T in $^{\circ}R$	
or 8	-	Blanks	-
to 11	-		-
12	a	Constants for the liquid specific	-
13	b	heat equation of the form :	-
14	c	$C_p = a + bxT + cxT^2 + dxT^3 + exT^4 + fxT^5$	-
15	d	where T is in $^{\circ}C$ and $C_p$ in	-
16	e	(cal/gm/ $^{\circ}C$ ).	-
17	f		-
18	$\epsilon / k$	Potential parameter and	$^{\circ}R$
19	$\sigma$	Collision diameter for the gas	$\text{A}^{\circ}$
		viscosity equation (9.11-1).	
20	$a_1$	Constants for the liquid viscosity	-
to 26	to $a_7$	equation (9.11.5).	-
or 20	-	Blank	-
21	I	Type of liquid viscosity estimation	-
		equation to use (9.11-6 or -8)	
22	I or $\Theta$	Viscosity constant used in the above	-
		equations.	

23	-	Blank	-
to 26	-		-
27	K	Type of vapour thermal conductivity equation to use.	-
28	H	Parameter in the liquid thermal conductivity equation (9.12-1)	-
29	$H_{vb}$	Molal heat of vapourisation at the normal boiling point for equation (9.12-2)	btu/lbmole
30	P	Parachor for the surface tension equation (9.13-1)	-

Table 9 - 3 : Data set in the PPTXTRA data file.  
(Cont'd)

CNDVTY : to calculate the conductivity of liquid or vapour phases.

SURFTENS : to calculate the surface tension of liquids.

The most accurate methods were chosen (95) and in certain cases more than one method is included to offer the user more flexibility and accuracy when including constants for new components. In many cases least-square fit polynomials are used for high accuracy especially for liquid properties not much affected by pressure.

The constants used for each component are set in the property data bank PPTXTRA and are copied for use into array PRTEXTRA in the way shown in table 9-3.

### 9.9 Function DENSTY

This function computes the density of a mixture of any phase. The density of a vapour mixture is obtained by applying the corrected ideal gas law:

$$\rho_v = \frac{1}{V} = \frac{P}{Z * R * T} \quad (9.9-1)$$

The compressibility factor  $Z$  is calculated from the Redlich-Kwong equation of state (61) applied at the mixture's conditions rewritten in terms of the compressibility factor (60):

$$Z = \frac{1}{1-h} - \frac{A^2}{B} * \frac{h}{1+h} \quad (9.9-2)$$

$$h = \frac{EP}{Z} \quad (9.9-3)$$

where A and B are the equation's constants. By rearranging these equations, equation (9.9-4) is obtained

$$Z^3 - Z^2 + B \cdot P \cdot \left( \frac{A^2}{B} - B \cdot P - 1 \right) \cdot Z = A^2 \cdot B \cdot P^2 \quad (9.9-4)$$

The equation is then solved by the Newton-Raphson iteration technique starting at  $Z = 0.9$ .

The vapour molar density is obtained by equation (9.9-1) and is multiplied by the mixture's average molecular weight to get the mass density.

The density of liquid mixtures is also based on the theory of corresponding states. The density of each compound is calculated at a base point on its saturation isotherm and the density at the required temperature and pressure is obtained by multiplying this density by a ratio of correction factors computed at the initial and desired conditions. The mixture density is computed on the basis of the theory of additive mole volumes.

$$\rho_m = \frac{\sum_i X_i \cdot MW_i}{\sum_i \frac{X_i \cdot MW_i}{\rho_i}} \quad (9.9-5)$$

The initial or base point density is calculated by one of three methods:

1. By equations fitted to the pure compounds' saturated densities over wide temperature ranges (90). The base point density is thus calculated at the mixture's temperature and multiplied by the ratio of correction factors to account for the pressure effect.

2. If these equations do not cover the temperature range required, their highest limit is used to calculate the base point density and the ratio of factors corrects the temperature and pressure effects.
3. Components for which no correlations exist, a single liquid density at 1 atm must be supplied. It is then corrected for temperature and pressure deviations. The temperature at which the density is given must be specified. It is usually 20°C.

The density factors exist in graphical form (90) as functions of reduced pressure and temperature and have been accurately curve-fitted for this routine.

The density equations for procedure 1 are of the form:

For  $T \leq T_{\text{characteristic}} (T_{\text{ch}})$

$$\rho_L = A - B * T - (C/(E-T)) \quad (9.9-6)$$

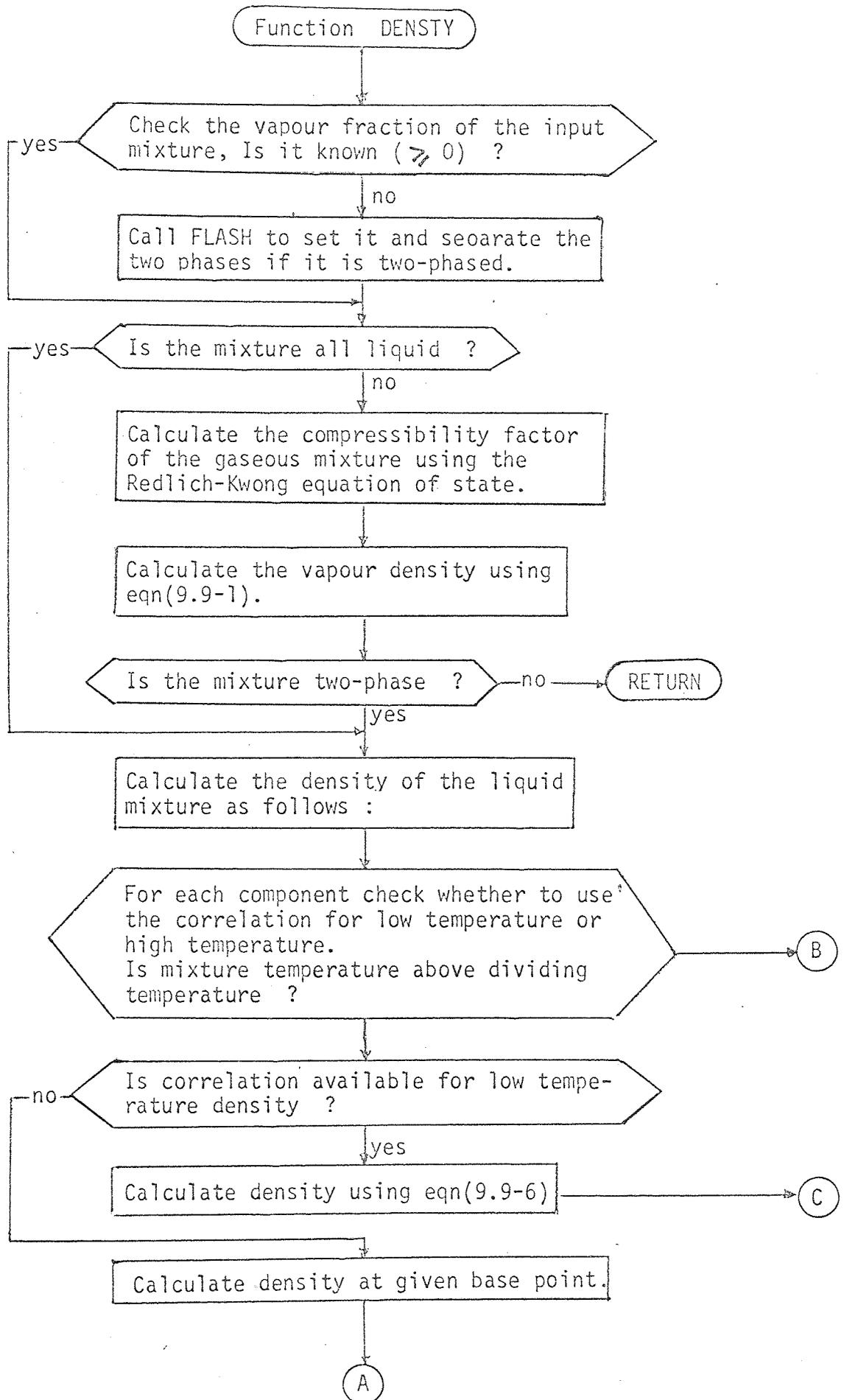
For  $T > T_{\text{ch}}$

$$\rho_L = (G * (T_c - T))^{1/H} + \rho_c \quad (9.9-7)$$

where  $\rho_L$  and  $T$  are in gm/ml and °F respectively and so are the critical density  $\rho_c$  and Temperature  $T_c$ . The way to set the constants various constants, for these equations, in the property array PRTEXTRA is shown in table 9-3.

When the high temperature range constants are not available,  $T_{\text{ch}}$  the temperature range upper limit should be negated in warning. When neither equation exist,  $T_{\text{ch}}$  is set to zero, and

Figure 9 - 6 : Flowchart of routine DENSTY.



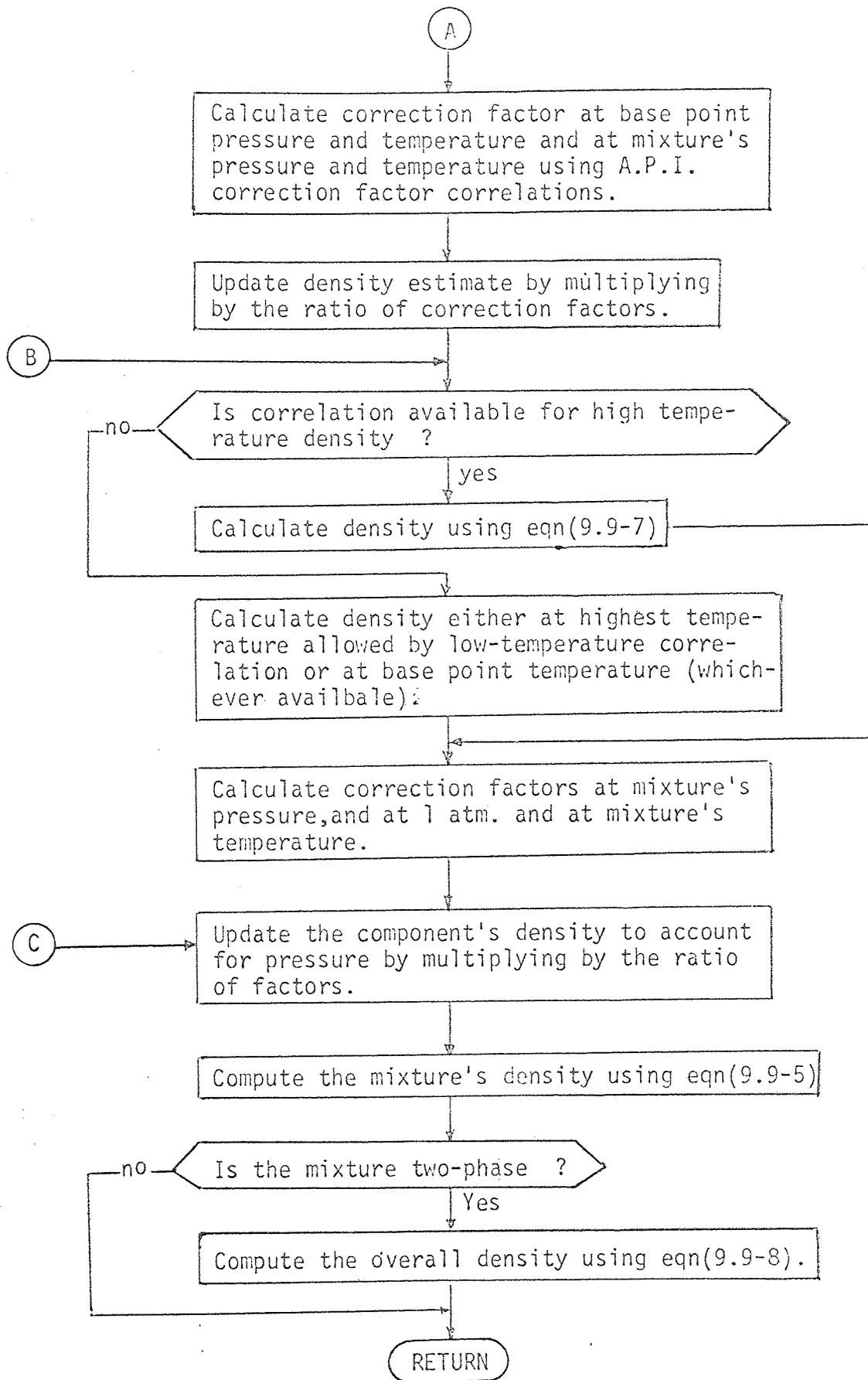


Figure 9 - 6 : Flowchart of routine DENSITY.

(Cont'd)

the base point density and its temperature set in the following positions in the array.

The average error in equation (9.9-6) is 0.5% and that in equation (9.9-7) 1% (90). The average error in estimating the liquid density of pure hydrocarbons with the correction factors method is 2%, however errors up to 10% can be expected at reduced temperatures above 0.9. The error added in estimating mixtures densities is about 4% (at  $Tr < 0.9$ ).

Two-phase mixtures are separated into their pure phases which are then treated separately as above and the two-phase density is computed based on the ideal law of volumetric additions:

$$\rho_m = \frac{VF * \sum_i (Y_i * MW_i) + (1-VF) * \sum_i (X_i * MW_i)}{\frac{VF * \sum_i (Y_i * MW_i)}{\rho_v} + \frac{(1-VF) * \sum_i (X_i * MW_i)}{\rho_l}} \quad (9.9-8)$$

where VF is the molar vapour fraction of the mixture.

In all cases the mixtures are assumed ideal and no heat or mixing effects taken into account for simplicity.

The flowchart of this routine is shown in figure 9-6.

#### 9.10 Function SPHEAT

The specific heat of gaseous hydrocarbons is calculated by the Hirschfelder et al. method (90, 91). This method is based on

the corresponding state approach to correct the ideal 'low pressure' vapour phase specific heats. These low pressure specific heats are given as polynomials of the absolute temperature in the A.P.I. Technical data book (90). The effect of pressure on the specific heats is sufficiently appreciable to be accounted for. The equations modelling this effect are particularly complicated and long and are here omitted. They do not require any additional data above the critical constants and the molecular weight of each component. This method has been chosen despite its complex programming because of its excellent reliability over a wide temperature and pressure ranges and because the lack of other reliable methods. Errors between the calculated and the experimental pressure corrections rarely exceed  $0.05 \text{ cal/gm } ^\circ\text{C}$  except in the critical region where errors ten times as large should be expected. The accuracy of the method decreases only above 2000 atm. However in certain cases where the ideal 'low-pressure' heat capacity equations are not available, the ideal low-pressure enthalpy ( $H^\circ$ ) equations are differentiated with respect to the temperature to obtain the ideal heat capacities ( $C_p^\circ$ ). This technique is not recommended (90) but is unavoidable because of data limitations. The average error in the A.P.T equations is less than 0.5%.

For liquid hydrocarbons, the Reid and Sobel method (90) proposed in the A.P.I. Technical data book was discarded on the grounds of its large storage requirements for graphs representations, and on the fact that pressure effects are generally small except

in the critical region (95). Instead graphical and point-data specific heats obtained from a selection of sources (98, 105) were fitted by polynomials to give an excellent agreement (about 99%). Special care was taken to select sources presenting data for large numbers of hydrocarbons to ensure an adequate consistency between the various compound data curve-fitted. The way to introduce new data and the units to use are shown in table 9-4.

The mixing rules applied to both the liquid and the vapour phases is

$$C_p^m = \frac{\sum_i (X_i * MW_i * C_{p_i})}{\sum_i (X_i * MW_i)} \quad (9.10-1)$$

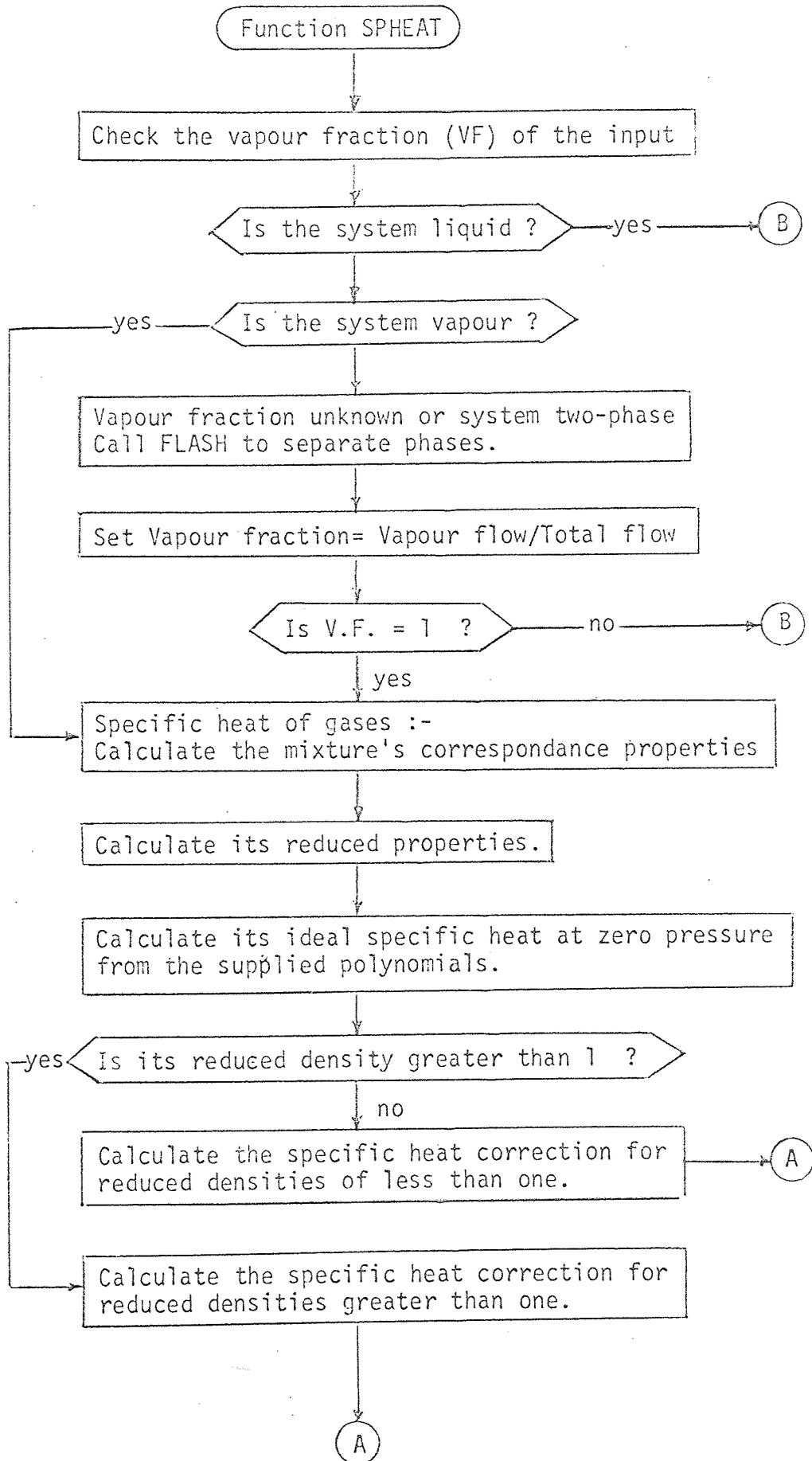
where the  $C_p$ 's are based on mass rather than moles. The error introduced by this mixing rule is about 5% (90).

The specific heat of a two-phase system is a less obvious and tangible property than that of pure phases. A rise in temperature of one degree in a multi-component two-phase system involves a certain amount of vapourisation and phase composition change and the heat required there includes a certain amount of heat vapourisation in addition the heat needed to raise the temperature of both pure phases. The author believes that this value is anyhow rather meaningless and no special action is taken in the program to calculate it. The liquid specific heat is returned instead.

The heat capacity is calculated in either B.t.u/(lb mass °F) or KJoule/(Kg mass °C) depending on the system of units used.

The flowchart of this routine is shown in figure 9-7.

Figure 9 - 7 : Flowchart of routine SPHEAT.



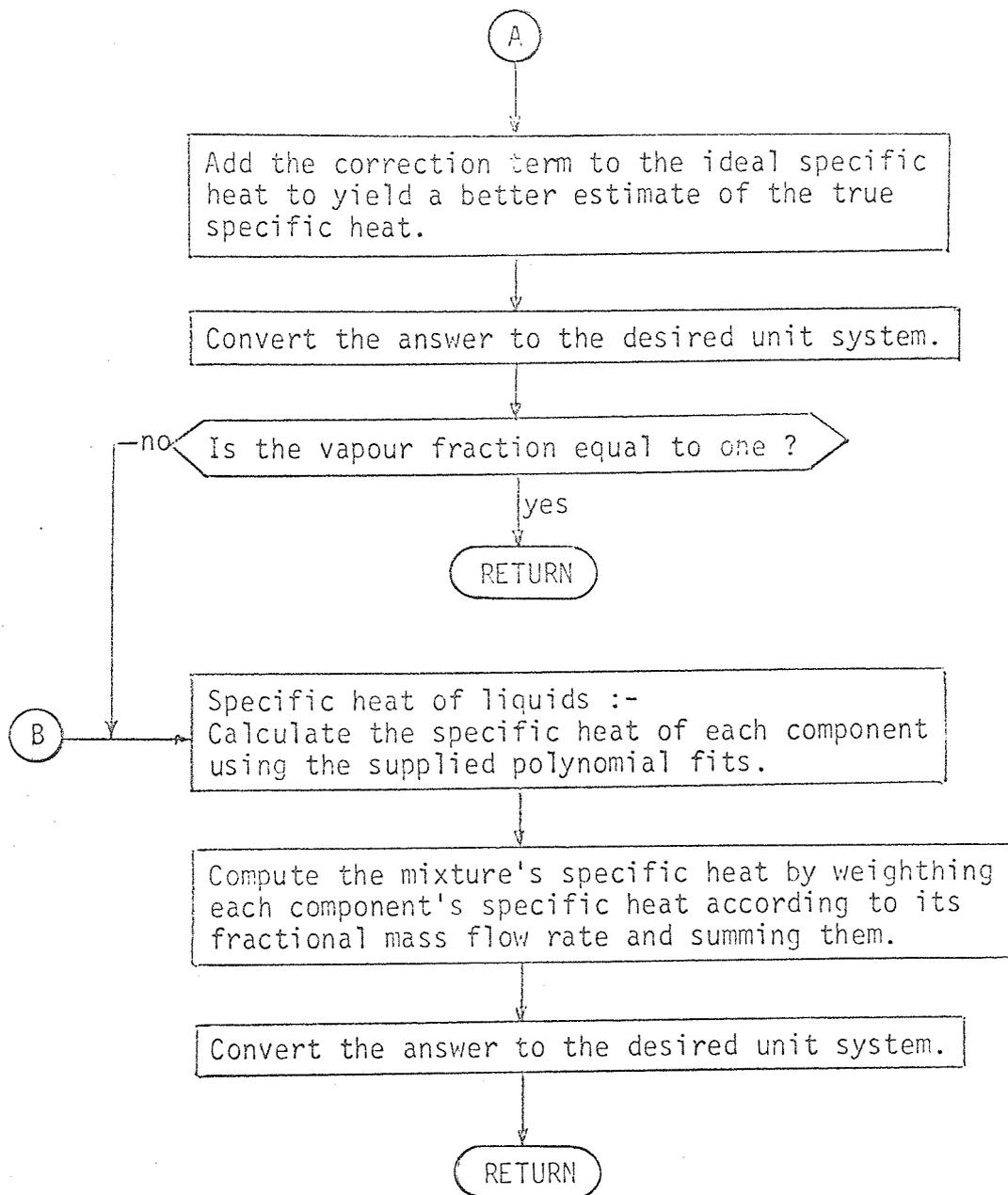


Figure 9 - 7 : Flowchart of routine SPHEAT.

(Cont'd)

## 9.11 Function VISC

The Chapman-Enskog-Cowling for vapour viscosities (90, 94) is the only equation among many which may be used over a wide temperature range, but at low pressures (Pr 0.6). It takes the form:

$$\mu^v = 0.001989 * \sqrt{MW * T} / (\sigma^2 * \Omega_v) \quad (9.11-1)$$

where T = Temperature in °R.

$\sigma$  = Lennard-Jones collision diameter in Å°.

$\Omega_v$  = Collision integral for viscosity.

$\mu^v$  = Vapour viscosity in centipoise.

The collision integral has been accurately curve-fitted from tabular data as a function of temperature and of the potential parameter  $\epsilon/k$  (58). The equation is very simple and default values are calculated for compounds of the which collision diameter and potential parameter are not provided.

$$\sigma = 0.7402 * (MW * V_c)^{V3} / Z_c^{1.2} \quad (9.11-2)$$

$$\epsilon/k = 65.3 * T_c * Z_c^{3.6} \quad (9.11-3)$$

where  $T_c$  and  $V_c$  are the critical temperature in °R and the critical volume in ft<sup>3</sup>/lb mass,  $Z_c$  the critical compressibility factor and MW the molecular weight of the pure compound.

For gaseous mixtures, the viscosity of each component is computed separately and the mixture viscosity obtained using the mixing rule:

$$\mu_m^v = \frac{\sum_i (Y_i * \sqrt{MW} * \mu_i^v)}{\sum_i (Y_i * \sqrt{MW_i})} \quad (9.11-4)$$

where  $Y_i$  is the components mole functions.

This mixing rule does not hold at reduced pressures above 0.6 (90) nor does the pure component viscosity equation (9.11-1).

High pressure effects on viscosity are very complicated to compute and have not been incorporated in this function because of time constraints.

The pure component liquid viscosities are calculated by one of four different methods because of the difficulty of representing accurately the liquid viscosities of all compounds by any one method (95). These methods are:

a. A polynomial of the form:

$$\mu^l = \sum_{i=1}^7 a_i * T^{i-1} \quad (9.11-5)$$

where  $T$  is in degrees C and  $\mu^l$  in centipoise. Although an inverse temperature term in the equation would give better fits, this equation was preferred to make it easier for users to provide the constants  $a_i$ .

b. Souder's method (95, 96) for estimating liquid viscosities at temperatures below the normal boiling point;

$$\text{Log} (\text{Log } 10 * \mu^l) = m * \rho_l - 2.9 \quad (9.11-6)$$

$$\text{where } m = I/MW \quad (9.11-7)$$

$I$  = a viscosity constant calculated from the component's molecular structure,

$\rho_l$  = the pure compound liquid density in gm/ml.

c. Thomas's method (95, 97) also used at temperatures below the normal boiling point:

$$\text{Log} (8.569 * (\mu^L / \sqrt{\rho})) = \Theta * (1./\text{Tr} - 1) \quad (9.11-8)$$

where  $\Theta$  also is a viscosity constant calculated from the compound's structure. The units are the same as those in Souders's method.

d. Stiel and Thodos's method (95) used above the normal boiling point. Their method is presented graphically, correlating the viscosity as a function of the critical and reduced properties for three discrete values of the critical compressibility factor  $\%c$ .

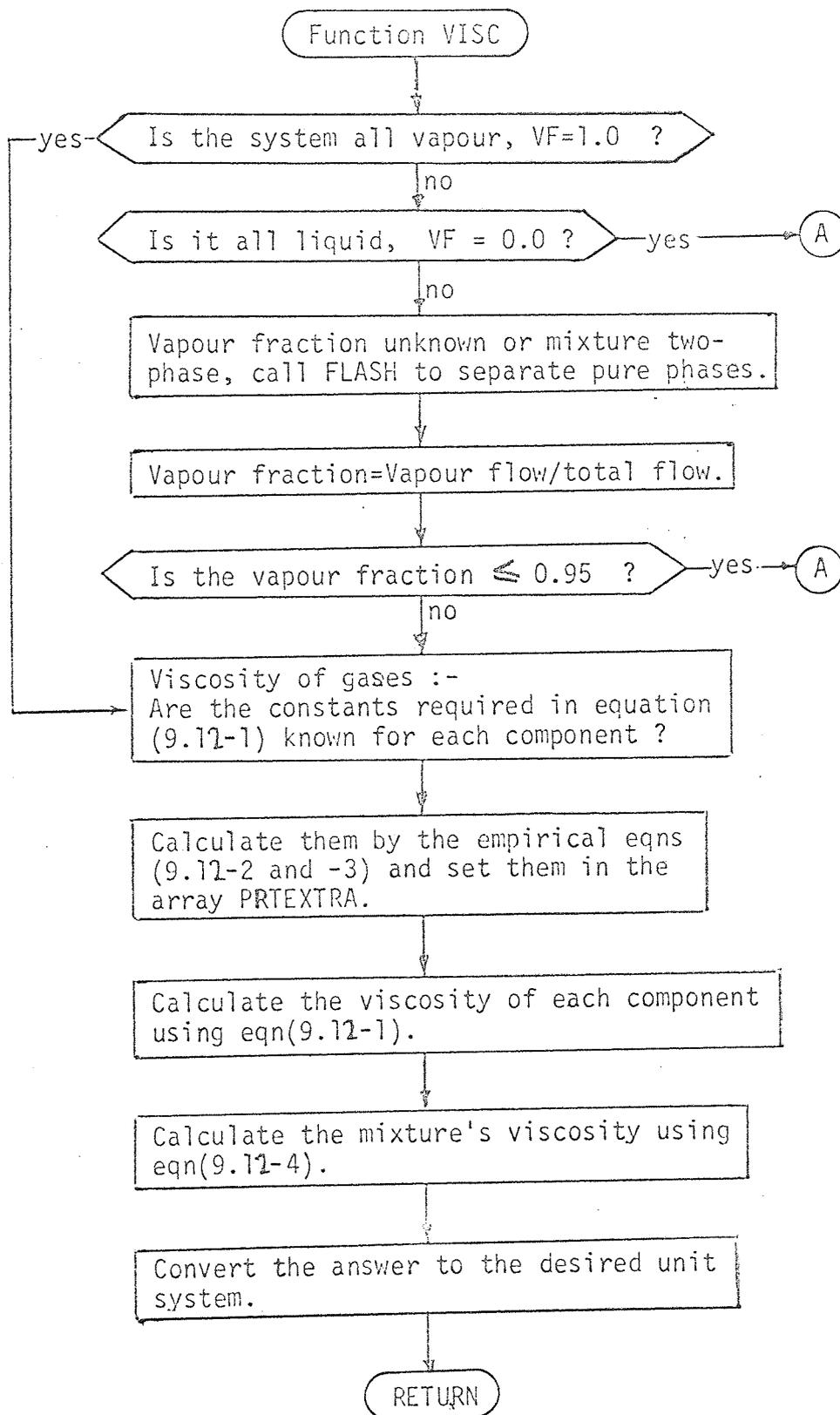
$$\frac{\mu^L}{\sqrt{\text{MW}}} * \left[ \frac{\%c}{\%c} \right]^{1/6} = f(\text{Tr}, \%c) \quad (9.11-9)$$

The curves have been fitted by polynomials to an accuracy of better than 99% and a linear interpolation technique is provided to compute the viscosities at any critical compressibility factor, however the method itself is only accurate to within 10%.

The method to use should depend on the availability of data and on their comparison to calculated data obtained from the empirical equations. For the components provided in the data bank the choice has been made on published comparisons (95) and priority given to polynomial fits where data were available (98). If polynomial fits are not available, position 20 in PRTEXTRA should be zeroed and the choice between equations b and c set in position 21 (1 for b and 2 for c) followed by the viscosity constant in position 22. Equation d needs no data.

For liquid mixtures, the following rule is used:

Figure 9 - 8 : Flowchart of routine VISC.



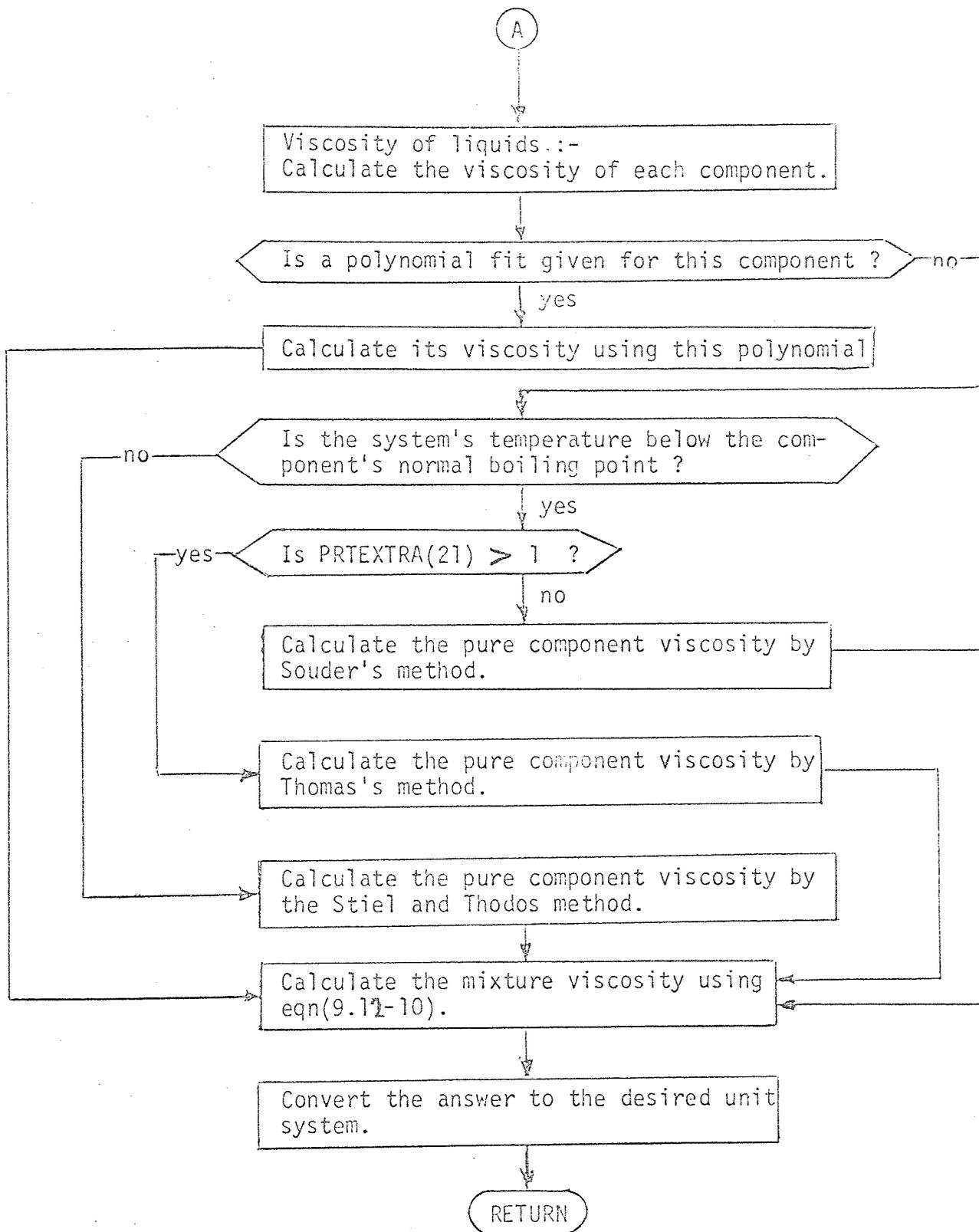


Figure 9 - 8 : Flowchart of routine VISC.

(Cont'd)

$$\mu_m^L = \left( \sum_i (X_i * \mu_i^{1/3})^3 \right) \quad (9.11-10)$$

The effect of pressure on liquid viscosities is negligible below 30 or 40 atms (95) and has been ignored.

The error in both the pure vapour and liquid viscosity methods is variable and ranges from 0.5% to over 30% (95) but for the components used in the data bank, the average error should be less than 10%.

The viscosity is calculated only for pure phases. The vapour phase contribution to the viscosity of two-phase systems is assumed minimal and the liquid phase viscosity is returned as the system's viscosity except when the vapour fraction exceeds 95%, then the system is assumed gaseous. The viscosity is returned in Kg/(meter sec.) or lb/(foot sec.) depending on the system of units used.

The flowchart of the routine is shown in figure 9-8.

#### 9.12 Function CNDVTY

Liquid phase conductivities are calculated for each component by the Robbins and Kingrea method (95, 99), the only accurate method available. Their equation bears the form:

$$k_\ell = \left\{ \left[ (88.0 - 4.94 * H)(10^{-3}) \right] / \Delta S \right\} * (0.55/Tr)^N * (Cp * \rho_e^{1/3}) \quad (9.12-1)$$

where  $k_\ell$  = Liquid phase pure component conductivity.

$$\begin{aligned}\Delta s &= \text{Modified Everett Entropy of vapourisation.} \\ &= \Delta H_{vb} / T_{bp} + R * \ln (T_z / T_{bp}) \quad (9.12-4)\end{aligned}$$

$\Delta H_{vb}$  = Molar heat of vapourisation at the normal boiling point  $T_{bp}$ .

$T_z$  = Absolute temperature of melting ice at 1 atm.

$R$  = Gas constant.

$H$  = A component-dependent parameter.

$N$  = 1 if specific gravity < 1, 0 otherwise.

In CONDVTY each component is treated separately. DENSITY is called to provide the liquid density for each component and function SPHEAT provides their specific heat ( $C_p$ ). Each of the three sets of terms in equation (9.12-1) is calculated separately using the constants provided in the arrays PROPTS and PRTEXTRA. The pressure effect on conductivity is felt only at pressures above 35 atms (95). At these pressures the conductivity is increased by an amount  $\Delta k$  computed by one of the following equations:

$$\text{For } \rho_R \leq 0.5, \Delta k = 14 * (\text{EXP}(-0.535 * \rho_R) - 1) * 10^{-8} / C \quad (9.12-3)$$

$$\text{For } 0.5 < \rho_R \leq 2, \Delta k = 13.1 * (\text{EXP}(0.67 * \rho_R) - 1.069) * 10^{-8} / C \quad (9.12-4)$$

$$\text{For } \rho_R > 2, \Delta k = 2.976 * (\text{EXP}(1.155 * \rho_R) + 2.016) * 10^{-8} / C \quad (9.12-5)$$

where  $\rho_R$  = Reduced density

$$\text{and } C = \sqrt{MW} * (T_c / P_c^4)^{1/6} * z_c^5 \quad (9.12-6)$$

The units to be used in these equations are given in table 9.4.

Two mixture conductivities are calculated, the first on a molar basis

$$k_m = \sum_i (X_i * k_i) \quad (9.12-7)$$

the second on a mass basis

$$k_m = \sum_i (X_i * M_{Wi} * k_i) / \sum_i (X_i * M_{Wi}) \quad (9.12-8)$$

where  $M_{Wi}$  is the pure components molecular weights and  $X_i$  their mole fractions. The thermal conductivity of mixtures is generally below that predicted by the mixing rules and therefore the lower of the two averages is chosen as the more correct answer (95).

The range of applicability of this method is for reduced temperatures between 0.4 and 0.9 where its average deviation is about 4% but rarely does it exceed 10%.

The vapour phase conductivity is calculated by the Misic and Thodos method (95, 100, 101) which treats only hydrocarbons and simple non-polar gases. Their method is based on an empirical dimensional analysis of the variables affecting the conductivity. There was found to be several equations satisfying their analysis, but each one describing certain types of compounds and at different temperature ranges. These equations are summarised in table 9-4 and classified according to the compounds they model. They are all used in the function `CNDVPEY` but the equation to be used for each compound must be specified in array `PRTEXTRA` in position 27. The temperature range is checked within the program and that also helps select the correct equation to use. The pressure effect on vapour conductivity is identical to that on liquid conductivity and is computed similarly.

A- For Methane, Naphthenes and aromatic hydrocarbons,  
at  $T_r < 1$  :

$$\frac{k \delta}{C_p} = 0.445 \times 10^{-5} \times T_r \quad (9.12-10)$$

B- For all hydrocarbon gases at all temperatures except for cases mentioned above :

$$\frac{k \delta}{C_p} = (10^{-6}) \times (14.52 \times T_r - 5.14)^{2/3} \quad (9.12-11)$$

C- For non-hydrocarbons, at  $T_r < 1$  :

$$\frac{k \delta}{C_p^{.75}} = (10^{-6}) \times (2 \times Z_c + 1.08) \times T_r^p \quad (9.12-12)$$

$$p = 1.81 - 2.604 \times Z_c \quad (9.12-13)$$

D- For non-hydrocarbons, at  $1 \leq T_r \leq 3$  :

$$\frac{k \delta}{C_p^{.75}} = (10^{-6}) \times ((195 \times Z_c - 31.94) \times T_r + 16.83 - 82.5 \times Z_c)^p \quad (9.12-14)$$

$$p = 1.524 - 2.8 \times Z_c \quad (9.12-15)$$

E- For non-hydrocarbons, at  $3 < T_r < 15$  :

$$\frac{k \delta}{C_p^{.75}} = (10^{-5}) \times ((7.18 - 18.25 \times Z_c) \times T_r + 10.21 \times Z_c + 4.91)^p \quad (9.12-16)$$

$$p = 1.079 - 1.97 \times Z_c \quad (9.12-17)$$

$$\text{In all cases : } \delta = \sqrt{MW} \times \left[ \frac{T_c}{P_c^4} \right]^{1/6} \quad (9.12-18)$$

Table 9 - 4 : Equations for gaseous thermal conductivity.

The units to be used are :

$$k = \text{cal./}(\text{gm. sec. Deg. K})$$

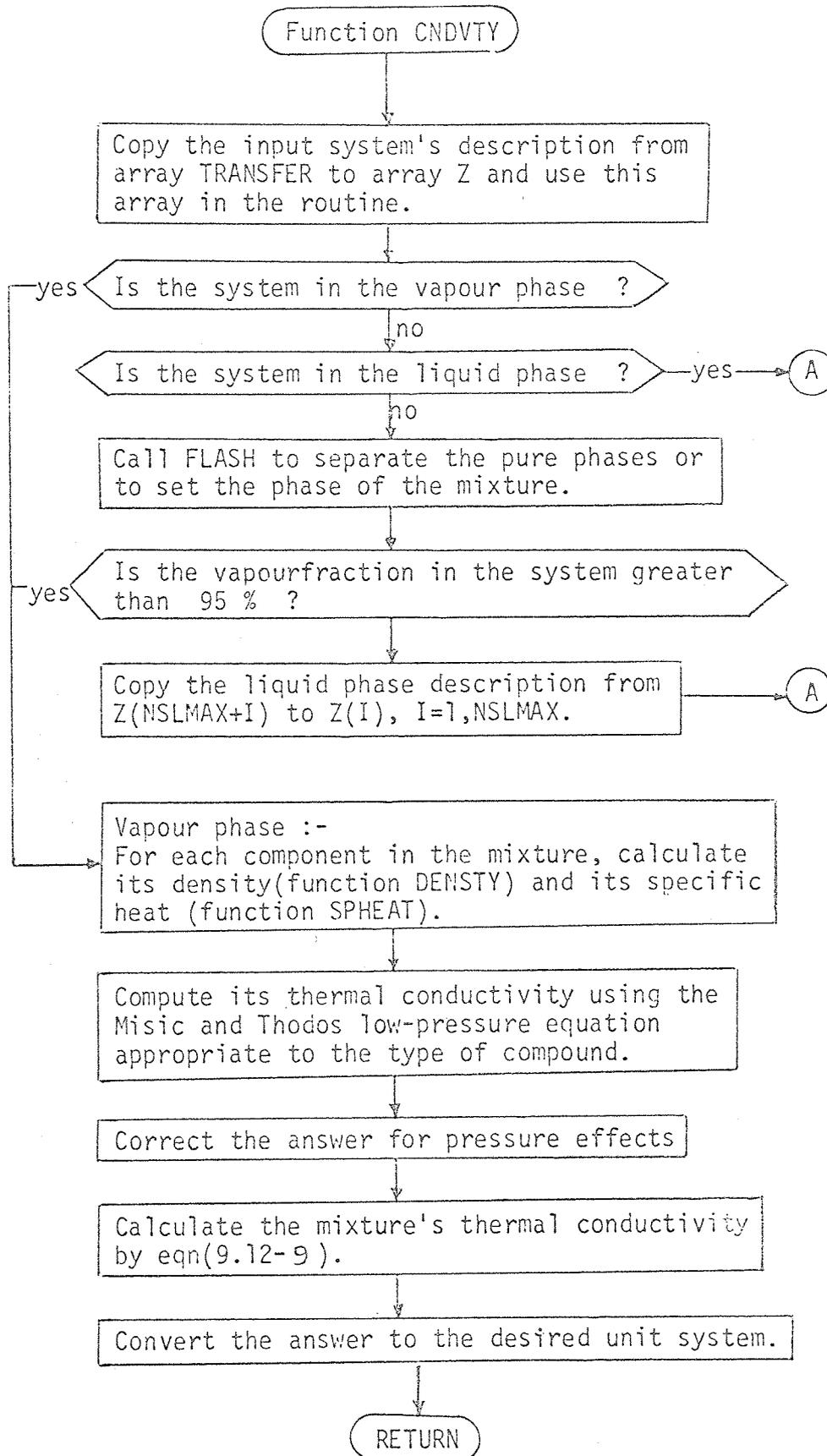
$$T_c = \text{Degrees Kelvin}$$

$$P_c = \text{Atmospheres}$$

$$C_p = \text{cal./}(\text{gm.mole Deg. K})$$

Table 9 - 4 : (Continued).

Figure 9 - 9 : Flowchart of routine CNDVTY.



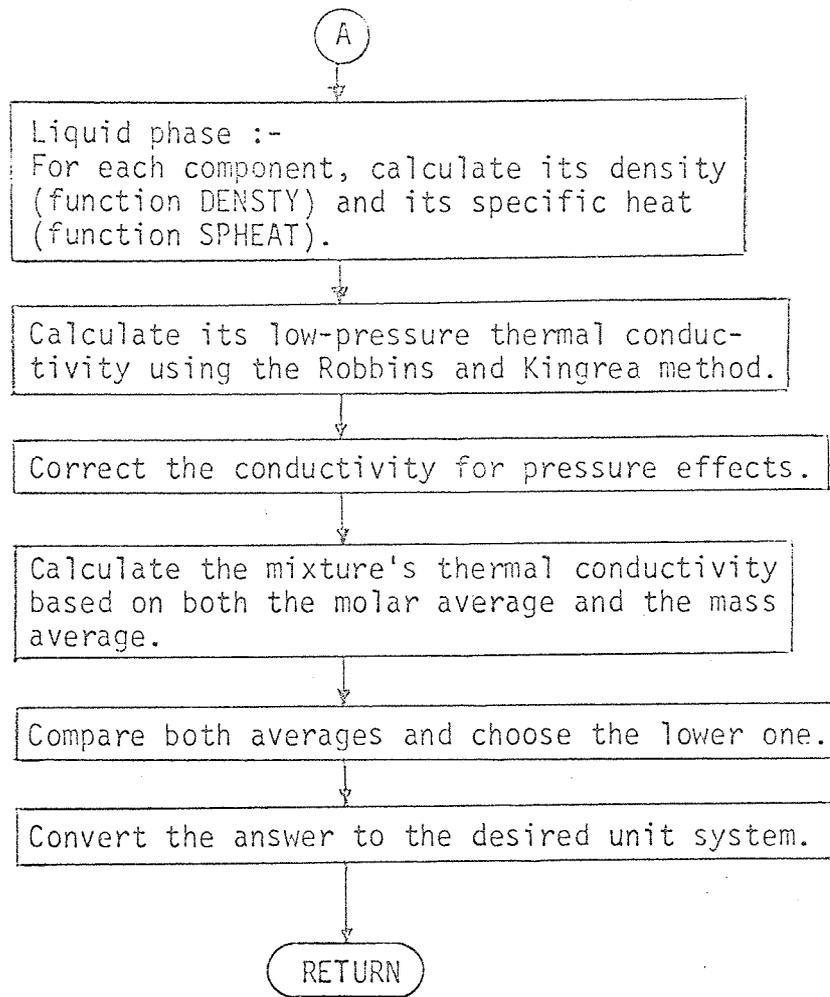


Figure 9 - 9 : Flowchart of routine CNDVTY.

(Cont'd)

Mixture conductivities are calculated by the equation:

$$k_m = \frac{\sum_i (Y_i * MW_i^{1/3} * k_i)}{\sum_i (Y_i * MW_i^{1/3})} \quad (9.12-9)$$

This mixing rule adds an extra 2% average error to the average 2.5% error given by the above methods for pure components.

The conductivity of two-phase mixtures is not calculated and the liquid phase conductivity is returned instead because of its greater contribution, but only for vapour fractions below 95%. Otherwise the mixture is assumed all vapour. In these cases FLASH is called to separate the mixture into its pure phase constituents.

The thermal conductivity is returned in either B.t.u./ (ft.sec.<sup>o</sup>F) at KJoules/(M.sec.<sup>o</sup>C) depending on the system of units used in the simulation.

The flowchart for this routine is shown in figure 9-9.

### 9.13 Function SURFTENS

The surface tension of non-polar liquids is computed by the Macleod Sugden correlation (90, 95, 102, 103):

$$\sigma = \left[ \frac{P}{MW} * (\rho_L - \rho_V) \right]^4 \quad (9.13-1)$$

where P is a parachor or component-dependent constant estimated from the contributions of the chemical groups in the compound, as developed by Quayle (90, 104). The surface tension of mixtures

( $\sigma_m$ ) is obtained by either equation (9.13-2) at pressures below 35 atm (90) or equation (9.13-3) at higher pressures.

$$\sigma_m = \sum_i X_i \sigma_i \quad (9.13-2)$$

$$\sigma_m = \left[ \sum_i P_i \left( \frac{\rho_L}{MW_L} X_i - \frac{\rho_V}{MW_V} Y_i \right) \right]^4 \quad (9.13-3)$$

where  $X_i$  and  $Y_i$  are the liquid and vapour mole fractions and  $\rho_L$ ,  $\rho_V$ ,  $MW_L$ ,  $MW_V$  refer to liquid and vapour mixture densities and molecular weights.

For polar compounds no reliable method is available. The above method is as good as any but errors are possible for some polar compounds (95).

In SURFTENS, the mixture's phase is checked and a warning message is issued if it is not pure liquid, in which case the surface tension is computed for the saturated liquid at the same pressure. SURFTENS calls KVALUE to obtain the vapour phase composition in equilibrium with the liquid and DENSITY computes the vapour and liquid densities. The correct mixture equation is used to compute the surface tension depending on the system's pressure. The surface tension is returned in either Newtons per meter or Pounds force per foot depending on the system of units used.

The only physical constant required is the parachor, set in position 30 in array PRTEXTRA. This evaluation method is accurate within 2.5% (90) for non-polars but should not be used when the reduced temperature of the pure substances exceeds 0.85.

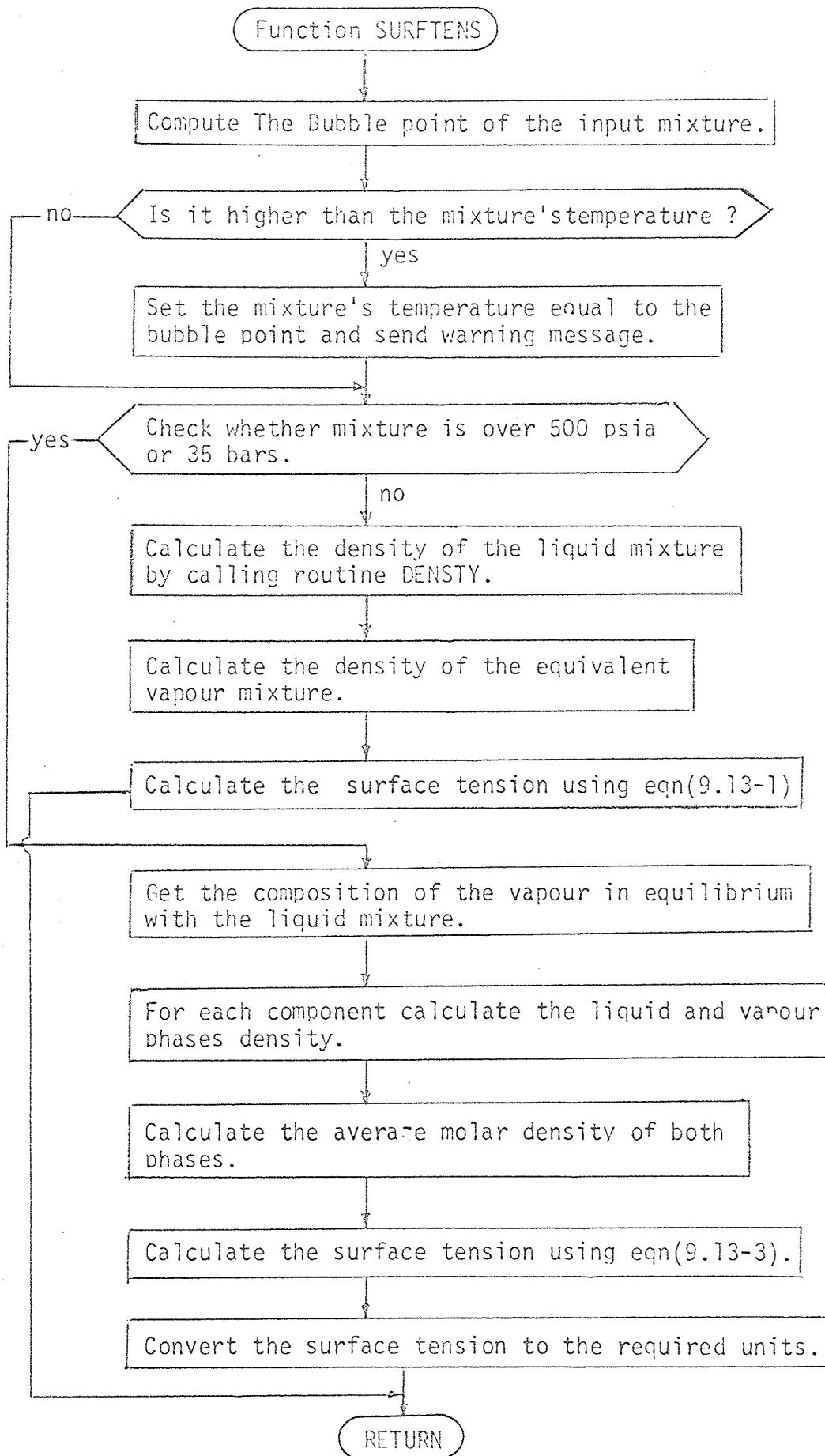


Figure 9 - 10 : Flowchart of routine SURFTENS.

For polar compounds the error may exceed 10% (95).

The flowchart of this method is shown in figure 9-10.

#### 9.14 Conclusion

The set of thermodynamic, thermal and physical property estimation routines included in PBEETPACK have been selected from amongst the best known or the more accurate methods presently known in fulfillment of the criterion that all methods used should be based on well-documented theoretical and numerical methods (section 4.5). They have been used very satisfactorily in a number of design cases (58) and have proved very reliable.

The list of data offered in PBEETPACK in its data banks PROPTS and PRTEXTRA are given in Appendix 3. They show first the number of components included in each bank, 45, and the number of data points for each component. PROPRT counts the component names as 2-words of data followed by 19 numbers, totalling 21 data points, whereas PRTEXTRA shows the names but does not include them in its 30 data positions. To extend these arrays tables 9-2 and 9-3 should be consulted and the data introduced in 2A8 Format for the names followed by the data in 1E15.7 format. Both data banks should be extended identically as the programs reads them both simultaneously to pick up the data, but unknown data may be replaced by zeroes if they are not going to be used during the simulation.

## CHAPTER 10

### THE EXECUTIVE PROGRAM

The Executive program in PEETPACK controls only the simulation calculations and it accelerates their convergence in recycle loop calculations. The executive's functions have purposely been reduced to a minimum to contain its overall size and its running time as the data can be read and interpreted in a separate program. The size of the executive program with the model and property routines attached to it is about 64000 words in un-overlaid mode. In its functions, the executive resembles the GEMCS executive except that it relies entirely on data pre-processed by the communicators. Contrary to PACER it has been written in a highly modular form which permits most of its sections to be modified or replaced by a simple exchange of segments, and the sections inter-dependence is very limited.

The executive is called by a MACRO following the input or update of the simulation data which it reads from the dump file (in routine DREAD). The executive then calls routine COMPUT which checks the calculation order and calls routine LOADST to load the input streams for each unit to be calculated. LOADST then calls EQCALL to call the requested unit model routine for the unit in question. On return from this model COMPUT checks whether the unit is in a loop and if so, calls routine TEST to check the

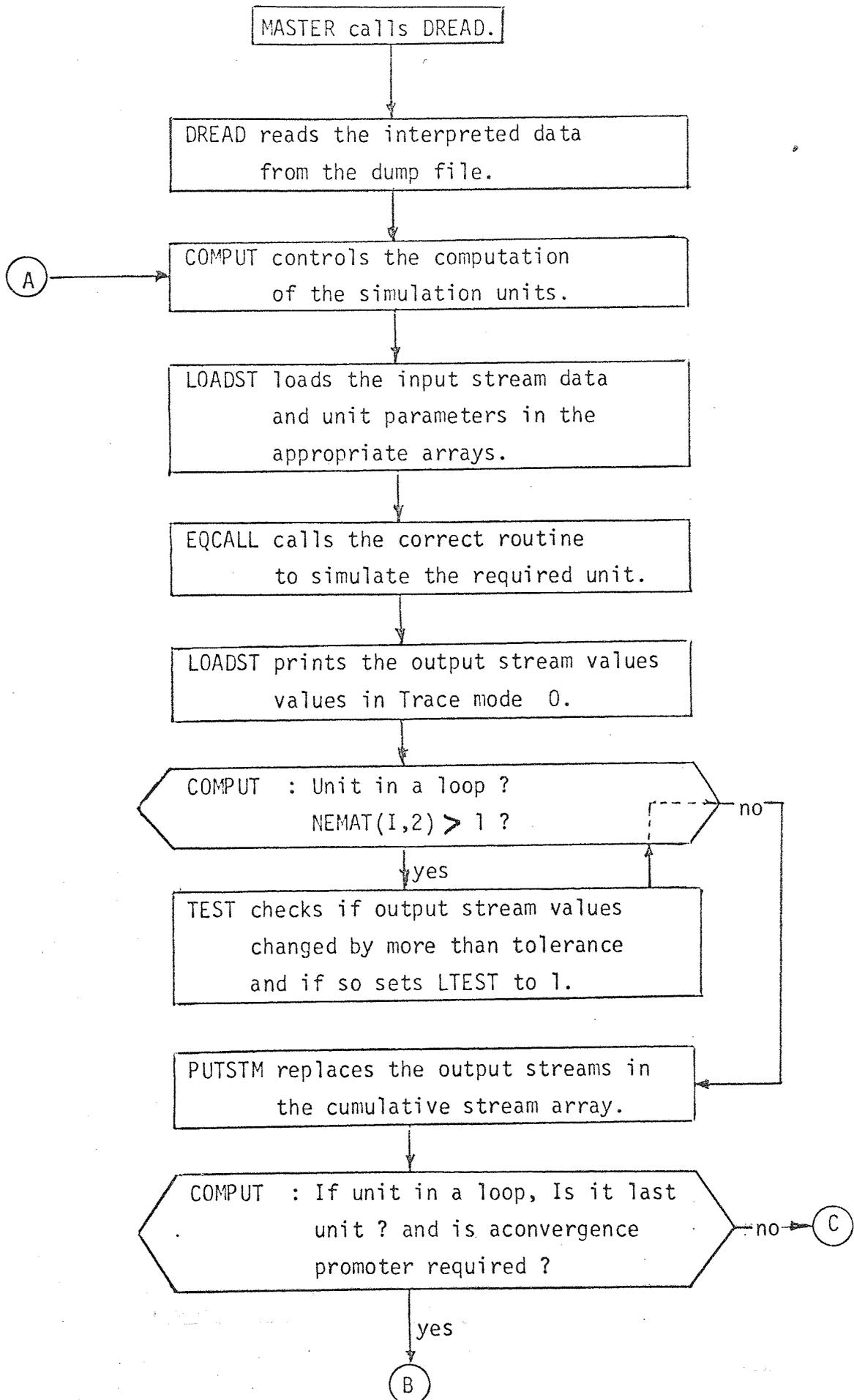
convergence of all the unit's output streams and sets a flag accordingly. COMPUT then calls routine PUTSTM to copy the output streams back into the stream matrix. At the end of each loop calculation, when requested, COMPUT calls one of the two convergence accelerators (PROMOT1 or PROMOT2) which operates on the iterate streams only. A loop calculation is repeated until convergence is achieved, or the maximum number of loop calculations exceeded, then the next recycle or sequential unit set is computed. At the end of the simulation the final results are printed out by routine RESULT; and DPRINT restores the final data back into the dump file which is then accessed either by the data echoer or the updater. The flow diagram of the procedure is illustrated in figure 10-1.

The eleven segments making up the executive are described below, and a short discussion on convergence promoters is given before the description of routines PROMOT1 and PROMOT2.

#### 10.1 The MASTER segment

This main routine is merely a calling program. It initially calls DREAD to input the interpreted data from the dump file which are passed to other routines via labelled COMMON statements. It then calls COMPUT to perform and control the simulation. At convergence it calls RESULT to print the final

Figure 10 - 1 : Organisation of the Executive.



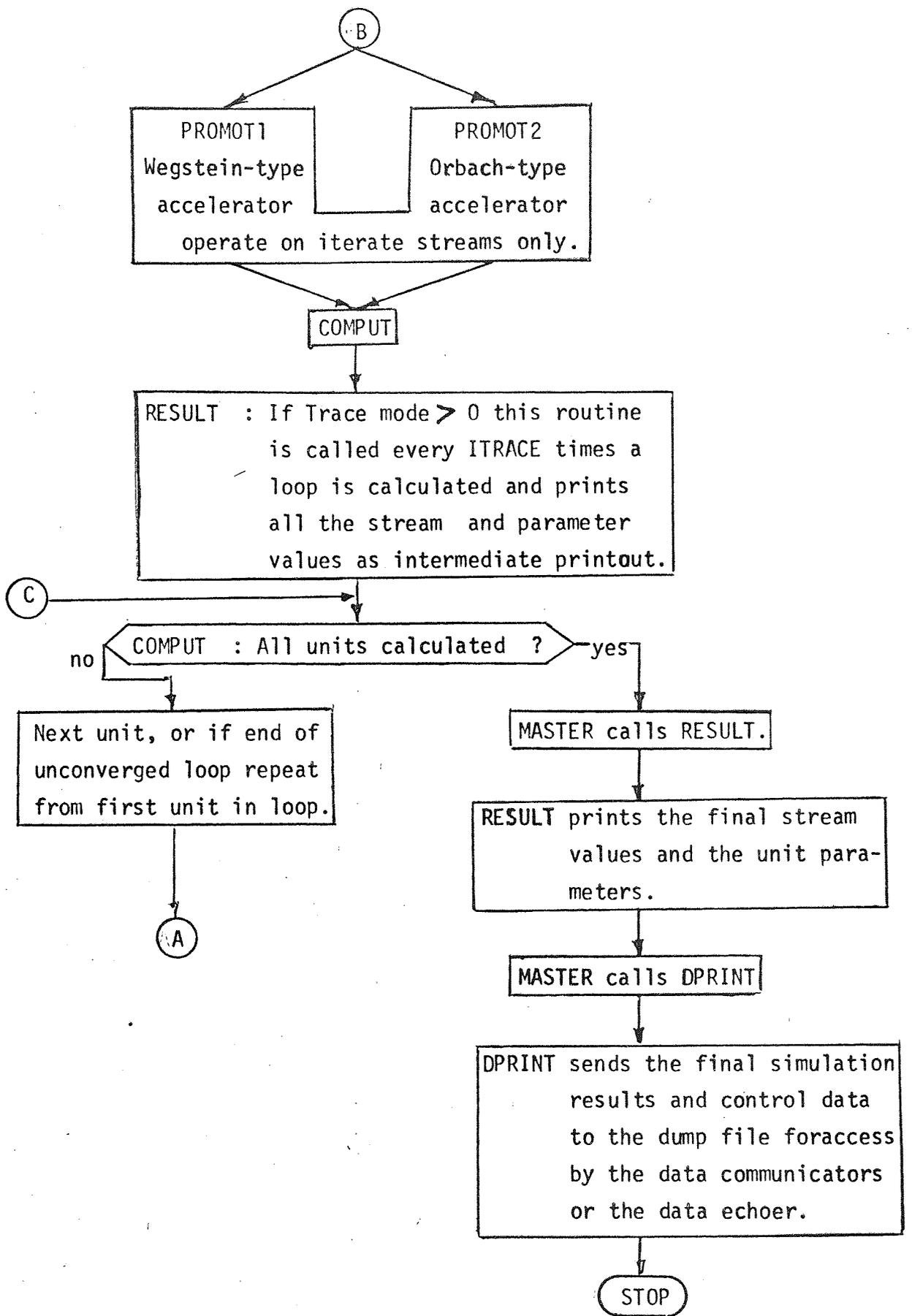


Figure 10 - 1 : Organisation of the Executive.  
(Cont'd)

simulation results onto the PEETLOG file which contains all the messages and intermediate and trace printout generated during the run. These data and the control data are then sent back to the dump file by a call to routine DPRINT and the program stops.

This segment is short to reduce compilation time when the EQCALL must be created and compiled with MASTER during the run in the industrial version of PEETPACK. The other routines are available as semi-compiled segments in three or more LIBRARY'S.

The arguments in all routine names are the input/output channels unless otherwise specified.

### 10.2 The DREAD routine

This routine reads the data from the dump file using the same compressed formats as the data communicators, and the data are stored in various arrays and matrices in six labelled COMMON statements. The variable names are described in section 6.4.3.

### 10.3 The DPRINT routine

DPRINT performs DREAD'S task in the reverse direction. A PAUSE statement in the MASTER program prior to calling DPRINT causes the MACRO to erase the contents of the dump file and assign it to the program card punch peripheral to receive the new data.

#### 10.4 The COMPUT routine

COMPUT controls the computational sequence. It decides which unit to calculate at every step, checks whether it is part of a recycle loop and, if so, tests the convergence of its output streams and calls the requested convergence promoter. It repeats this sequence of events until the unit computations converge or exceed the maximum allowed number of loop iterations. It operates as follows:

The operating MODE is set to zero (calculations not converged). The unit counter (NC) is set to 1 and the unit number of similar position in the calculation order is set for calculation. If its calculation order flag is less than 2 it is in a sequential set and routine LOADST is called to load its input streams into arrays STRM1 and STRM0 for easy access by the models. The unit model routine is called and, on return, PUTSTM copies its output streams back into the cumulative stream array. The unit counter is increased by one and the next unit in the calculation order is called.

For a unit starting a new loop, a "number of units in the loop" counter (NELOOP) is still zero. The KONVRG flag checks whether a convergence promoter is required. If KONVRG is greater than 0, all iterate streams numbers are sent to the convergence promoter which uses them to initialise its arrays. Following this

initialisation procedure and for subsequent recycle units, LOADST is called to load the input streams and call the models then TEST is called to check the convergence of the unit's output streams. On return the flag LTEST is set either to 1 (unconverged) or to 0 (all streams converged). KTEST is then set to 1 if LTEST = 1 or unchanged otherwise, and routine PUTSTM copies back the output streams to the cumulative array. Every time a loop unit is calculated, NELOOP is raised by 1. Each loop unit is treated similarly until the last loop unit is calculated. At the end of the loop, the number of loop calculations (LOOP) is incremented by 1 and if it is an integer multiple of the ITRACE flag, routine RESULT is called to printout the stream and parameter arrays. If the loop has not converged, the requested convergence promoter is called to improve the iterate streams descriptions. The unit counter is reduced by the value of NELOOP and the calculation is repeated for the whole loop unless the number of loop iterations (LOOP) exceeds the allowed maximum (NLOOPS). In this case, or if the loop calculations have converged, the next unit in the plant sequence is calculated unless the NOGO option is set, in which case the plant simulation is abandoned after NLOOPS loop calculations.

At the end of a successful simulation, the run MODE is set to 1 and all the units are called again, once only each, to

print any final message or to print the converged answers and proceed with the mechanical design of routines fitted with design sections. All models should do a test on MODE at the beginning to avoid recalculating the routines in this last run.

Following this check, command is returned to the MASTER to print the results and stop.

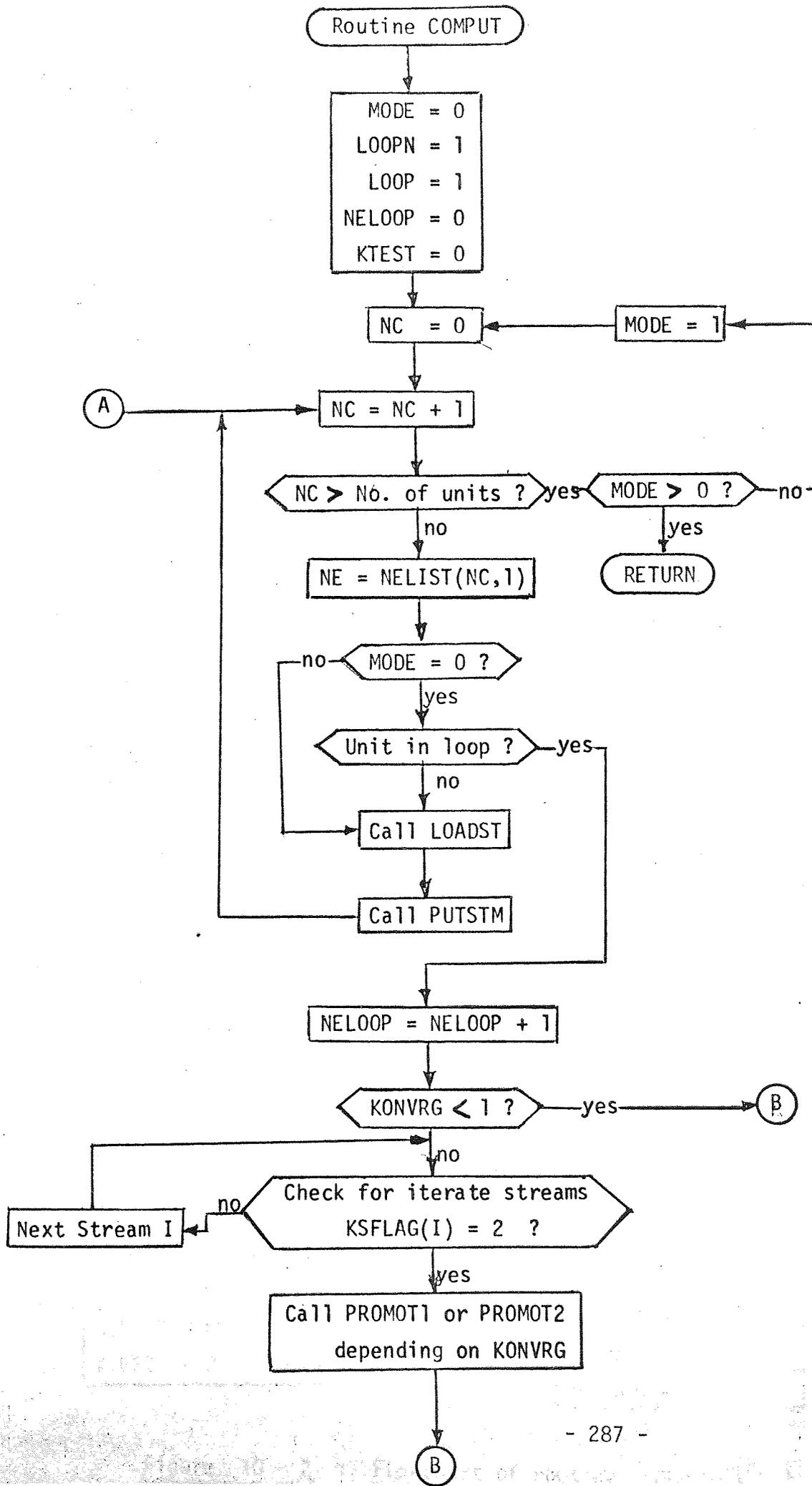
The routine's flowchart is shown in figure 10-2.

#### 10.5 The LOADST routine

This routine is called to find the number of input (NIN) and output (NOUT) streams attached to the unit being calculated and to copy their latest values into matrices STRMI and STRMO accessible to all models.

LOADST sets NIN and NOUT initially to zero and zeroes the STRMI and STRMO matrices. It then checks the process matrix row corresponding to the unit studied and counts the number of positive stream number (inputs) and negative numbers (outputs) to get NIN and NOUT. Every time a positive (or negative) stream number is met, NIN (or NOUT) is incremented by 1 and the NSLMAX descriptors of the stream are copied from position  $((J - 1) * NSLMAX + I)$  onwards in array SMATRX into array NIN (or NOUT) of STRMI (or STRMO) (J refers to the stream number and

Figure 10 - 2 : Flowchart of routine COMPUT.



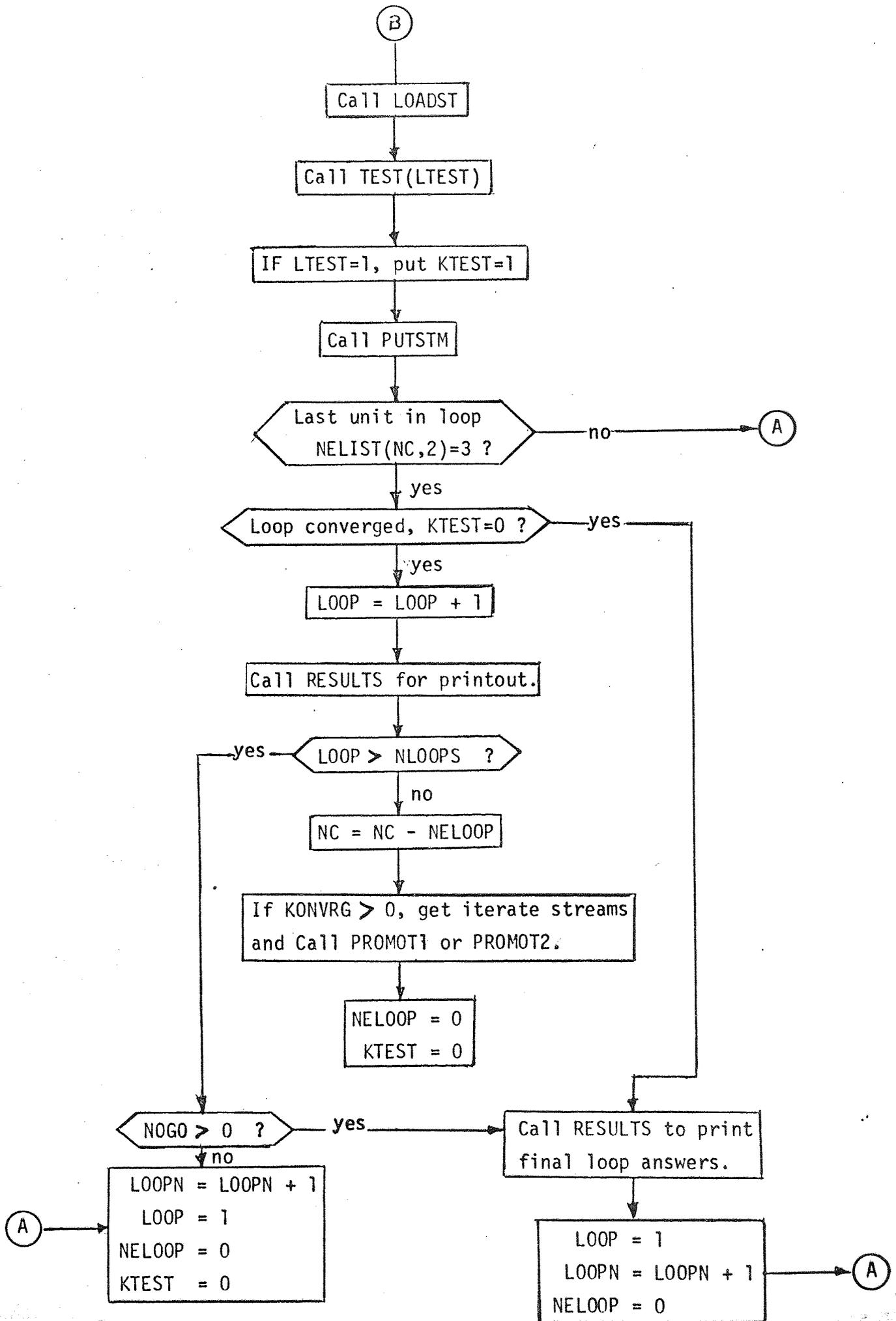


Figure 10 - 2 : Flowchart of routine COMPIIT (Cont'd) - 288 -

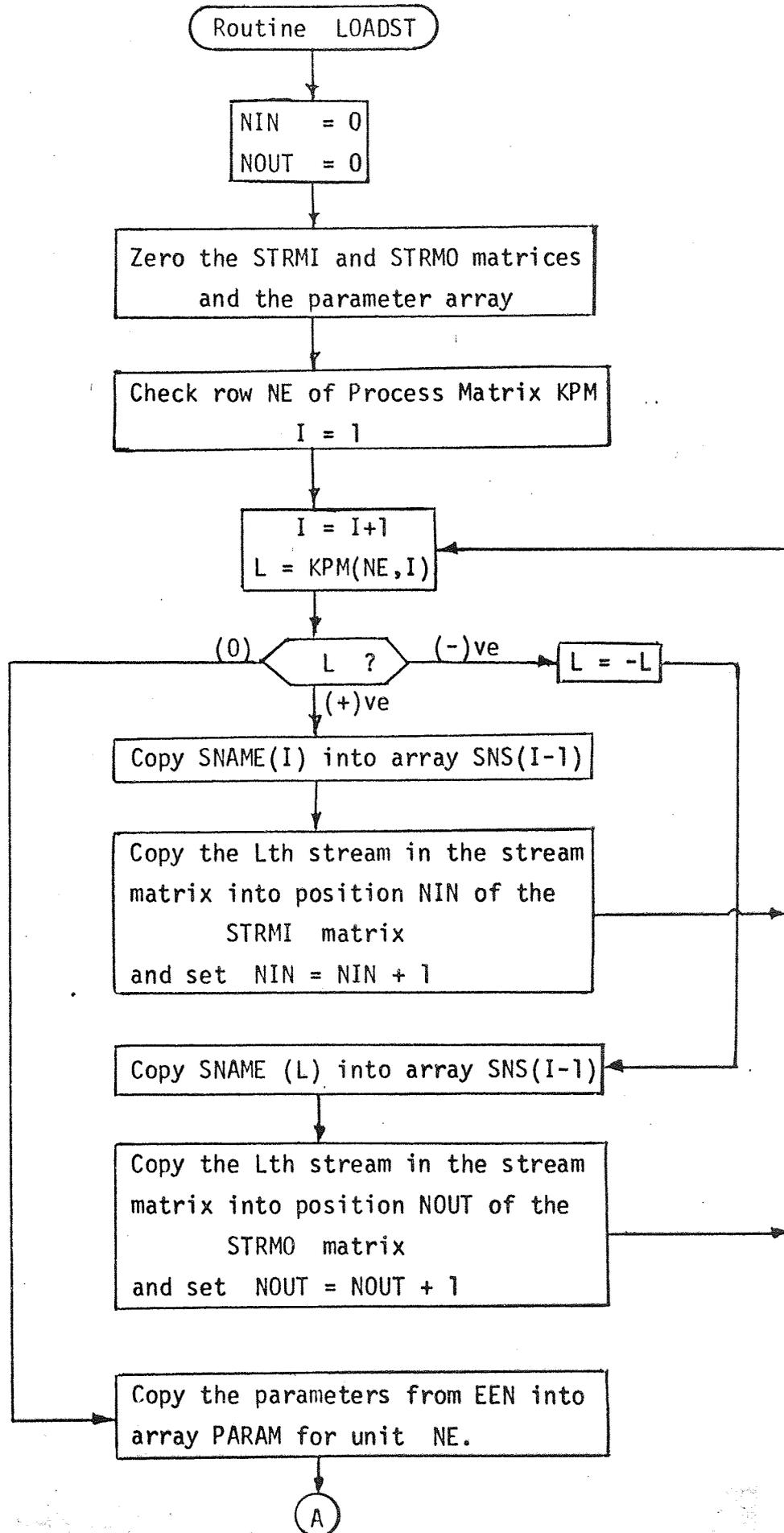
NSIMAX to the length of the stream description, SMATRIX is the cumulative stream array).

The parameters are then copied from the cumulative parameter array EEN into the zeroed PARAM array. As the number of parameters per unit is variable, matrix NEMAT records for each unit (J) the starting position of its parameters and their number. A number NEMAT (J,2) parameters will be copied from position NEMAT (J,1) into PARAM unless NEMAT (J,2) is zero.

Following these two data-copying sequences, each input stream has its flow rate and enthalpy checked. Streams bearing no flow cause a warning message to be printed to this instance since this can give rise to a number of errors in the models. For a zero enthalpy stream (enthalpy unset) and when the number of components is greater than zero, LOADST calls routine ENTHALPY to remedy to this omission. This option of not setting stream enthalpies can be used with input streams to relieve the user from calculating mixture enthalpies by compatible methods.

Before calling the unit models for computation, and in trace mode 0 only, the name of the routine called to simulate the unit is printed followed by its input streams descriptions. Routine EQCALL is then called and calls in turn the model routine. Upon return from EQCALL, and in trace mode zero only, the output

Figure 10 - 3 : Flowchart of routine LOADST.



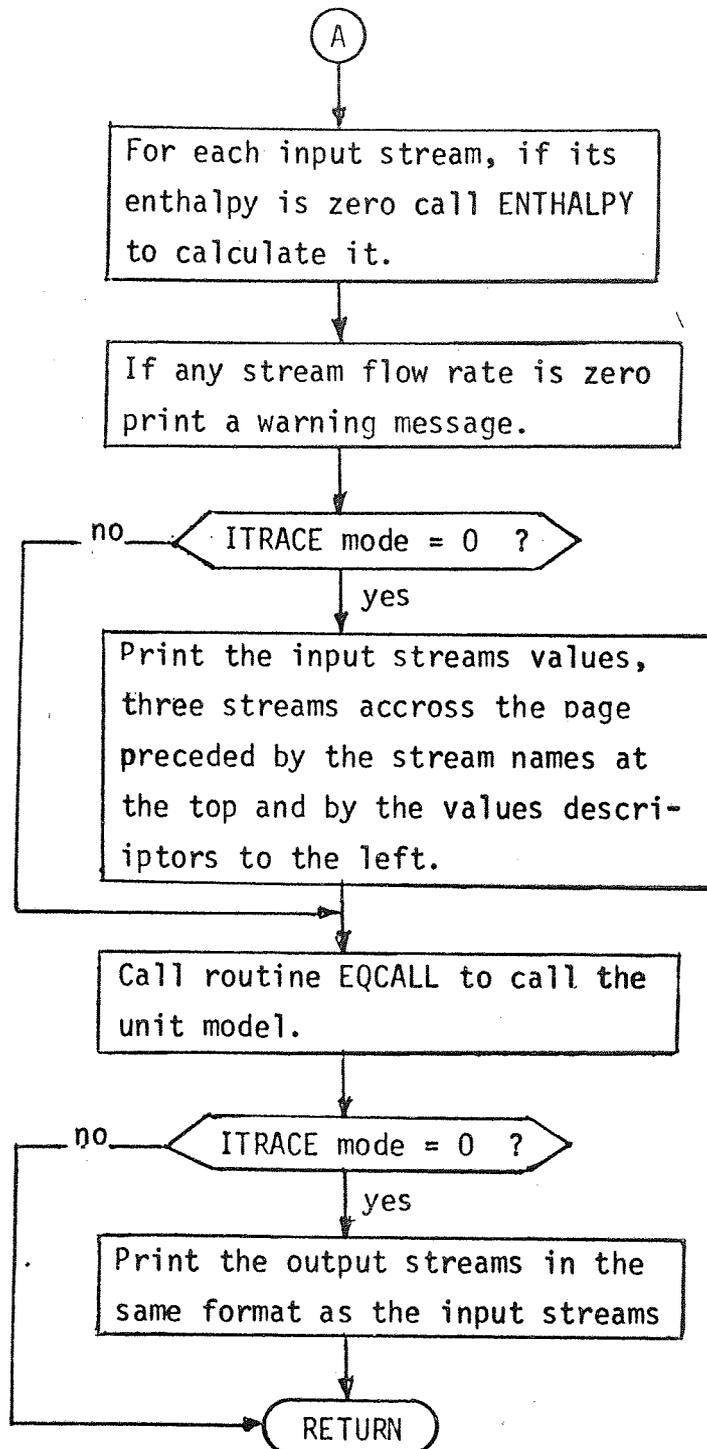


Figure 10 - 3 : Flowchart of routine LOADST.  
(Cont'd)

streams are printed and LOADST exited. The routine's arguments are the unit's internal number and the trace printout channel. The routine's flowchart is illustrated in figure 10-3.

## 10.6 The EQCALL routine.

This model calling routine is a sequence of CALL statements each one followed by a RETURN command. Two versions of this routine have been written, each giving a completely different stature to the whole package.

### 10.6.1 Version 1

For teaching purposes, when the models used are the standard ones provided in the package, the EQCALL routine is a fixed part of the executive. The order in which the CALL statements are set is identical to the order in which the model names are set in their DATA statement in the communicator (SMODN) and used for model name recognition. Each position in array MODULE refers to one of the units and gives the sequence number of the routine modelling it. In EQCALL this number is used in an assigned GO TO statement preceding the set of CALL statements to branch out to the appropriate CALL statement and the unit is called.

Users are allowed to add up to five new model routines

if they give them one of the following names recognisable in the communicator: UNIT1, UNIT2, UNIT3, UNIT4 and UNIT5. These routines are compiled and consolidated with the MASTER segment in the presence of all the other LIBRARY'S either before loading the executive or externally.

#### 10.6.2 Version 2

The EQCALL routine is created by the data communicators writing into a file which is then appended to the MASTER segment and compiled in the presence of the executive routines LIBRARY and all the model and property routine LIBRARY'S. This routine is made up of a set of CALL statement identical in order to that of the plant units each of which is given its own CALL statement bearing the model name the user gives it, thus any name may be used. No name recognition takes place, but any new model named should be available in a user-provided LIBRARY of new models the name of which is inserted as a sixth parameter on the program calling card as

LIB : User Number . Library file name.

This version is useful in industrial context where users have no direct access to the executive. It adds flexibility to the program but at the expense of a longer running time.

The argument in this subroutine is the number of the unit to be calculated.

### 10.7 The TEST routine

The routine tests the relative change in each output stream from each unit in a recycle loop against a preset tolerance, following the unit calculation. If any descriptor value in any of the output streams has changed by more than the preset fraction, the variable LIMIT, previously zeroed, is set to 1 and the routine exited. The executive requires that all streams converge before stopping a loop calculation.

One tolerance is set for all checks rather than a different one for each descriptor, as in PACER, to reduce the amount of data input and the trouble of selecting correct tolerances, however for temperatures near the 0 level ( $\pm 10^0$ ) a tolerance of  $0.5^0$  is used instead to accelerate convergence.

The routine's arguments are the unit's number and the test flag LIMIT.

### 10.8 The PUTSIM routine

This routine checks the process matrix KPM to obtain the output streams numbers to the unit just calculated and copies their values back into SMTRX from the STRMO matrix. It also returns the unit's parameters to their cumulative array as the user is allowed to change the parameters in the models.

The routine's argument is the current unit number.

## 10.9 The convergence of recycle loops

The presence of recycle material in a plant prevents the material and energy balances from being calculated in a once-through serial manner. To correct the error generated by these balances when using guessed estimates for the recycle streams, a number of methods have been proposed.

### 10.9.1 Unaided convergence Succession substitution.

In the equation describing recycle processes:

$$\underline{X} = F(\underline{X}) \quad (10.9-1)$$

the newly calculated value of the function  $F(\underline{X})$  is substituted for the original guesses of  $\underline{X}$  and a new function value is calculated. The process is repeated until convergence is attained within a preset tolerance. The method will produce a diverging series of estimates if the interaction between the various elements making up  $\underline{X}$  is large (assuming  $\underline{X}$  is a vector), or if the initial guesses are far from final answers. It is also a very slow converging method when recycle ratios are very high (over 10 to 1) (23).

### 10.9.2 Accelerated convergence Non-Analytical.

If a history is kept of the progress of the  $\underline{X}$  estimates and of the function values they generate, new estimates of  $\underline{X}$  can

be predicted closer to the true answer instead of relying on direct substitution. The best known prediction methods are:

A. Wegstein's Method (57) Wegstein solves simultaneously the two sets of equations:

$$\underline{Y} = \underline{X} \quad (10.9-2)$$

and 
$$\underline{Y} = F(\underline{X}) \quad (10.9-3)$$

by an iterative Newton-Raphson procedure where  $\underline{Y}$  is the final answers for the recycle stream values,  $\underline{X}$  are estimates and  $F(\underline{X})$  is the plant response. By applying the two-point Newton-Raphson

technique on points obtained either initially by successive substitution or later by applying the method, an improved value  $\bar{X}_{n+1}$  of the estimates  $X_{n+1}$  may be obtained from the equation

$$\bar{X}_{n+1} = \frac{X_{n+1} * \bar{X}_{n-1} - X_n * \bar{X}_n}{X_{n+1} + \bar{X}_{n-1} - X_n - \bar{X}_n} \quad (10.9-4)$$

The subscripts refer to the iteration number and the superscript bar to previous improved values, variables without superscript refer to function evaluations. For the first two iterations the improved values are set identical to the function evaluations. The method will almost always converge.

B. The Hyperbolic method (106) This method is based on passing two hyperbolae through the three most recent function evaluations. The first rectangular hyperbola is passed through the first two points and the other one passes through the second two points

using the same Y asymptote but with its X asymptote displaced by a length D which depends on certain parameters. By solving simultaneously the two equations thus obtained the following sets of equations are obtained and give an improved estimate of  $\underline{X}$ :

$$\underline{X}_{2n} = F(\bar{X}_{2n} - 1) \quad (10.9-5)$$

$$\underline{X}_{2n+1} = F(\underline{X}_{2n}) \quad (10.9-6)$$

$$\bar{X}_{2n+1} = \underline{X}_{2n+1} + \left[1.0 - \frac{T}{S}\right] \left[\underline{X}_{2n} - \underline{X}_{2n+1}\right] \quad (10.9-7)$$

where 
$$T = \frac{\bar{X}_{2n} - 1 - \underline{X}_{2n}}{\underline{X}_{2n} - \underline{X}_{2n+1}} \quad (10.9-8)$$

and 
$$S = 1.0 \text{ for } 0.5 < T < 1.5$$
  

$$= T - 1.0 \text{ otherwise.} \quad (10.9-9)$$

This method differs from Wegstein's in that it is applied only every other step, hence the reason for the 2n subscript. Its error half-life, in a published example (106) is of about 1.7 iterations compared to about 1.5 iterations for the Wegstein method on the same example, however for both methods there are cases where the interdependence of the  $\underline{X}$  elements opposes the benefits of these accelerators, but such cases have not been met by the author.

C. Orbach's method (23) Orbach assumes the plant to behave linearly with respect to the input streams thus obeying the set of equations:

$$\underline{X}_n = \underline{A} \underline{X}_{n-1} + \underline{B} \quad (10.9-10)$$

By solving these equations for successive pairs of points but continuously substituting a function of the initial variable value  $\underline{X}_0$  for the old variable  $\underline{X}_{n-1}$ , a converging geometric series is obtained as follows:

$$\underline{X}_1 = \underline{A} \underline{X}_0 + \underline{B} \quad (10.9-11)$$

$$\begin{aligned} \underline{X}_2 &= \underline{A} \underline{X}_1 + \underline{B} \\ &= \underline{A} (\underline{A} \underline{X}_0 + \underline{B}) + \underline{B} \end{aligned} \quad (10.9-12)$$

$$\underline{X}_n = \underline{A}^n \underline{X}_0 + \frac{\underline{A}^n - 1}{\underline{A} - 1} \underline{B} \quad (10.9-13)$$

The recursive formula for updating the new estimates becomes:

$$\underline{X}_n = \frac{\underline{X}_{n-1}^2 - \underline{X}_{n-1} * \underline{X}_{n-2}}{2\underline{X}_{n-1} - \underline{X}_{n-2}} \quad (10.9-14)$$

Although it is similar in concept to Wegstein's method, it could converge linear problems, such as sets of mixers and splitters faster than Wegstein's method, but it operates slower on non-linear problems. Its successive applications should be delayed by two or three iterations to allow the perturbations introduced by the application of the method to die out (23).

In these three methods each element in the  $\underline{X}$  vector of variable is treated independently and the vector multiplication refers to the multiplication of corresponding terms only.

D. The Dominant eigen value method (107) This method is an

improvement of Orbach's method in that the interaction between the various elements in  $\underline{X}$  is taken into account to a certain degree thus reducing further the number of convergence iterations. The dominant eigen value method consists in carrying out the initial iterations until the iterates approach a geometric progression of the form:

$$\left\| \Delta \underline{X}_n \right\| / \left\| \Delta \underline{X}_{n-1} \right\| = \lambda \text{ a constant (10.9-15)}$$

The highest proportionality constant for the various elements of  $\underline{X}$  is chosen and used to improve all the elements in  $\underline{X}$  by the following equation:

$$\bar{X}_n = \underline{X}_{n-1} + A * (\underline{X}_n - \underline{X}_{n-1}) / (1 - \lambda_1) \text{ (10.9-16)}$$

where  $\lambda_1$  is the dominant (highest) eigen value and  $A$  a damping factor between 0 and 1 to suppress oscillations, guessed or set from previous observations.

The calculations are continued, possibly using another convergence promoter, until the geometric progression is again obeyed and the method is applied again.

The major drawback of this method is the unguided setting of the damping factor  $A$  which depends on the experience gained with the system under study.

### 10.9.3 Analytical methods for convergence promotion

A few analytical methods based on the Newton-Raphson iteration technique have been applied to the acceleration of processes. They are generally used in conjunction with processes that can be represented by a set of simultaneous equations because the methods require the computation of derivatives. Their application in flowsheeting programs is of lesser value as these processes are not analytical in form, i.e. they are not represented by explicit sets of equations, and thus cannot be differentiated. Obtaining their partial derivatives by difference methods is both time consuming (two or more  $n - 1$  dimensional iterations being required) and the results are not very accurate since the models are not always very sensitive to small changes in their inputs. The best known of these methods are:

1. The Newton method: The derivatives of the function  $F(\underline{X})$  are calculated for all the variables and a new set of variables  $\underline{X}_{n+1}$  is obtained from the equation:

$$\underline{X}_{n+1} = \underline{X}_n - \underline{A}_n^{-1} * F(\underline{X}_n) \quad (10.9-17)$$

where  $\underline{A}^{-1}$  is the matrix of partial derivatives (the Jacobian).

2. The modified Newton method: Here the Jacobian is only evaluated once every few iterations to reduce the computation time. The method is efficient only if it can be ascertained that the Jacobian does not change much at each iteration? (2).
3. Quasi-Newton method (Secant method or Broyden method): An approximate estimate of the Jacobian matrix is calculated and corrected at each iteration step based on a vector function of errors between the predicted and the calculated values and on an arbitrary damping factor vector.

Some of these methods have been successfully applied to the convergence of certain process calculations (2) however they were usually developed in conjunction with these processes and fitted to their requirements, moreover these processes were very slow to converge and required more powerful convergence promoters.

In general flowsheeting programs are not usually used to solve long and slow converging problems completely because of the expense involved, therefore analytical methods, where the calculation of the Jacobian may take more time than can be allocated to the whole simulation, are not useful.

Two methods were incorporated in PEETPACK: Wegstein's

because of its almost certainty in converging stiff problems and Orbach's method as an alternative because of its simplicity and its faster solution of linear recycle problems.

#### 10.10 The PROMOT1 routine

This convergence promotion routine uses the Wegstein acceleration method in the form revised and presented by the publication editor (57).

The new value of a variable  $\underline{X}$  ( $\bar{X}_{n+1}$ ) is based on the knowledge of the variable's latest value ( $\underline{X}_{n+1}$ , i.e. the function value  $F(\underline{X}_n)$ ), the improved and unchanged values at the previous stage ( $\bar{X}_n$  and  $\underline{X}_n$ ) and the improved value of the previous iteration ( $\bar{X}_{n-1}$ ), so that:

$$\bar{X}_{n+1} = \frac{\underline{X}_{n+1} * \bar{X}_{n-1} - \underline{X}_n * \bar{X}_n}{\underline{X}_{n+1} + \bar{X}_{n-1} - \underline{X}_n - \bar{X}_n} \quad (10.10-1)$$

The method thus requires the storage of 3 sets of past values. The routine has been written to accelerate up to five iterate streams, each stream referred to by a serial number IS taking values between 1 and 5, and the promotion is applied to each stream descriptor.

The improved values  $\bar{X}_{n-1}$  and  $\bar{X}_n$  are stored in CONVY (IS, 1 to NSIMAX) and CONVY (IS, 16 to 15 + NSIMAX) respectively.

The unchanged values, or function values,  $X_n$  and  $X_{n+1}$  are stored in CONVX (IS, 1 to NSLMAX) and CONVX (IS, 16 to 15 + NSLMAX).

Since past history values are needed, successive substitution is used for the first two iterations and the acceleration begins from the third iteration onwards and is applied at every iteration, unless the denominator in equation (10.10-1) is zero ( $\approx 10^{-3}$ ) in which case  $\bar{X}_{n+1}$  is set equal to  $X_{n+1}$ ; then the variables are pushed down the stack:

$$\bar{X}_{n-1} = \bar{X}_n \quad \text{and} \quad \bar{X}_n = \bar{X}_{n+1} \quad (10.10-2)$$

The values of  $X_{n+1}$  are extracted directly from the cumulative stream matrix SMATRIX.

Before the iterative calculations begin, the routine is called to initialise the variables  $\bar{X}_{n-1}$

The routine is called only for iterate streams to the first loop unit and is supplied in its argument list with the stream number (NS), the iterate stream serial number (IS) and the loop number (LOOP) which decides whether the variables are to be used for initialisation (LOOP < 3) or for promotion (LOOP ≥ 3).

### 10.11 The PROMOT2 routine

PROMOT2 uses Orbach's acceleration method (23). The new value of  $X_{n+1}$  ( $\bar{X}_{n+1}$ ) is calculated by the equation:

$$\bar{X}_{n+1} = \frac{X_{n+1} + K(X_{n+1} - X_n)}{1 + K} \quad (10.11-1)$$

where

$$K = \frac{X_n - X_{n-1}}{2X_{n-1} - X_n - X_{n-2}} \quad (10.11-2)$$

Following Orbach's recommendation (23), but without any proof, the convergence accelerator is applied every third loop iteration only to allow the results to "recover" following the upset in their trend and to obtain a new series of consistently generated data. For the other two loop calculations the iterate stream variables are copied from SMATRX into CONVST (IS,J) (for  $X_{n-2}$ ) and CONVST (IS,J + 15) (for  $X_{n-1}$ ), and successive substitution is used to obtain  $X_{n-1}$  and  $X_n$ . At each third iteration the SMATRX ((NS - 1) \* NSLMAX + J) values are updated according to the above two equations, but so long as the denominator in equation (10.11-2) is not close to zero ( $\pm 10^{-4}$ ) to avoid numerical overflow. The improved values in SMATRX are not stored for the next iteration which depends only on the subsequent three iterations.

The argument and data required in PROMOT2 are identical to those in PROMOT1.

### 10.12 The RESULT routine

RESULT prints out the value of all the plant streams and all the unit parameters in two tables. It may be called in one of two cases:

- a. Every ITRACE times a recycle loop is completely calculated. In this case the flag option IOPTN, in its argument list, is equal to the serial number of the cumulative loop being calculated, and this number is printed in the output's heading.
- b. At the end of a successful simulation IOPTN is set to zero in MASTER and the same data are printed but headed with:  
FINAL STREAM (or EQUIPMENT PARAMETER) MATRIX.

Streams are typed in three columns across the page, preceded to the left by the descriptors names and each stream column is headed with the stream name. The unit parameters are typed in rows, four parameters per line following the unit name.

### 10.13 Conclusion

It is difficult to judge whether this executive program is more or less efficient than others because it performs less duties than many others and because little is known about most other programs internal structure. Its highly modular form increases its flexibility and its ease of understanding, and the methods adopted in most routines have been kept as simple and basic as possible to increase its receptivity at academic levels.

## CHAPTER 11

### PRACTICAL APPLICATIONS OF PEETPACK

Three examples solved on PEETPACK are presented in this chapter. They have been selected to demonstrate some of PEETPACK'S features and advantages and to point out to some of its deficiencies. The proposed examples are:

1. The simulation of the Ethane-Ethylene separation plant previously discussed in chapter 3 to demonstrate the use of PEETPACK and to discuss some instability problems in flowsheeting.
2. The mass balancing of four splitters set-up in nested recycle loops to demonstrate the application of convergence accelerators.
3. The heat and mass balancing of four flash columns in nested recycle loops which gives rise to equilibrium constant problems.

#### 11.1 Example One

The first example demonstrates the application of PEETPACK to the simulation of an industrial-type example and illustrates its data input and output. The example chosen for this demonstration is the Ethane-Ethylene plant described in chapter 3. In this simulation based on the data of the third survey day, the third component Propane is accounted for, and the feed composition is taken to be: Ethane 54 mole %, Ethylene 44% and Propane 2%.

Figure 11-1 shows the batch-mode data input to a typical case run for this plant. The data themselves are very similar to those used in the PACER/GEMCS/CONCEPT exercise, but their presentation is very different and much more understandable than for PACER or GEMCS (Appendix 1). With a minimum knowledge of PEETPACK, any user may read and recognise them. The interactive data-input, shown in figure 11-2 is similar to CONCEPT'S, but PEETPACK'S advantage over CONCEPT is that the sequence of data input is taken care of by the program whereas in CONCEPT the user must request each input in turn and hence may omit some information although CONCEPT does check the data for completeness before initiating the calculation. PEETPACK still suffers from the lack of a complete data check before attempting a simulation and is thus more subject to simulation failures due to incomplete batch-mode data input.

Once the data is input in either mode, the data communicators interpret this free-format alphanumeric data and store them in a dump-file illustrated in figure 11-3. The executive program reads the interpreted results which contain the property data, as extracted from files PROPTS and PPTEXTTRA, and initiates the calculation phase. In trace mode 0 a continuous printout of the input and output streams to each unit calculated is produced (figure 11-4) whereas in higher trace modes, a printout of the stream and parameter matrices only is produced at the specified loop intervals (figure 11-5). At the

end of a run, whether successful or not, the same type of matrix printout is obtained, and unless other data updating commands are issued, a data echo is produced (figure 11-6) summarising all the results and the unchanged input data. Otherwise the new updating commands are obeyed and a new run initiated.

The results of the plant simulation using the third survey day data are tabulated in table 11-1. All the assumptions stated in chapter 3 regarding the feed conditions and the representation of pressure drops by throttle valves were also included here, and the same discrepancy in the results regarding the flow to the reboiler and the mis-match of temperatures and pressures occurred here also. However using some of PEETPACK'S facilities, these discrepancies that could not be explained in the PACER/GEMCS simulation were overcome. The feed stream which was previously assumed to be at  $-20^{\circ}\text{F}$  was sent first through routine SETBP to have its temperature correctly calculated at its pressure of 345 p.s.i.a. This major correction had the immediate effect of restoring the flow requirement to the reboiler to near its survey data value and hence explained the previous discrepancy. The pressure drops in the reboiler and the cooler that were previously represented by throttle valves could now be incorporated in these units' parameter arrays and allowed a small process diagram change and a reduction in computing time; but it is still not possible to let the models compute these pressure drops from

geometric calculations. The corrections to the data of the first case simulation on PEETPACK, to change the process diagram and the unit parameters, are illustrated in figure 11-7. The new simulation results are tabulated in table 11-2.

11.1.1 The plant instability and the limitations of conventional flowsheeting

All the simulation results reported in chapter 3 and above refer only to the successful runs and reveal nothing of the plant instability from a flowsheeting point of view. The effect of the amount of flow sent to the column's reboiler on the behaviour of the whole plant is very critical. A larger than necessary flow rate to the reboiler causes its output temperature and enthalpy to remain high, so that as its pressure drops in the process pipes, a certain amount of vapourisation occurs and its temperature drops appreciably, thus affecting directly and negatively the temperature of the first output stream from the heat exchanger to the compressor (stream 4). The resulting effect is an insufficiency in the heat load to the reboiler, and an impossible situation arises and causes the run to fail. Similarly a low initial flow rate to the reboiler, or a low temperature guess for the compressor inlet develop the same unacceptable situation and cause a run failure.

The solution to this problem at present, is in the repeated simulation attempt with various guesses for the split

Figure 11 - 1 : Batch-mode Data Input to PEETPACK.

```

BATCHREAD PLANTDATA
ETHYLENE / ETHANE PURIFICATION PLANT ON PEETPACK MK. 1
PROCESS MAIRIX
VALV1 FEED1
      S2
DC1  S2  S17  S7
      S3  PRODUCT1  S8
HE1  S3  S14
      S4  S15
COMP1  S4
      S5
VALV2  S5
      S6
SPLIT1  S6
      S7  S10
COOLR  S10
      S11
VALV3  S8
      S9
VALV4  S11
      S12
MIXER1  S9  S12
      S13
VALV5  S13
      S14
SPLIT2  S15
      S16  PRODUCT2
VALV6  S16
      S17
PARAMETERS
VALV1  215
DC1  40  20  256  256  4.25  215  1  2  0
      4  1  1.6063  1.6915  .4841  -0.0741
      1.0613  1.209  0.354  -0.2148  0.2769
      0.4408  0.2098  0.1259
HE1  -7  0  0  0  100  1
COMP1  585  .75  .94  1.23
VALV2  464
SPLIT1  .905  .095
COOLR  -23  0
VALV3  414
VALV4  414
MIXER1
VALV5  399
SPLIT2  .19  .81
VALV6  215

```

```

ROUTINES
VALV1 VALVE
DC1 DISCOL
HE1 HEATEXCH
COMP1 COMPRESR
VALV2 VALVE
SPLIT1 SPLITTER
COOLR HEATER
VALV3 VALVE
VALV4 VALVE
MIXER1 MIXER
SPLIT2 SPLITTER
VALV5 VALVE
VALV6 VALVE
COMPONENTS
ETHYLENE ETHANE PROPANE
PREFERRED STREAMS
S4 S17
ORDER OF CALCULATION
SEQ. VALV1
LOOP COMP1 VALV2 SPLIT1 COOLR VALV4 DC1 VALV3 MIXER1
      VALV5 HE1 SPLIT2 VALV6
STREAMS
FEED1 595 -20 345 0 0 .44 .54 .02
S4 1340 -8 215 0 1 .95 .05 0
S17 1180 -40 215 10 1 .95 .05 0
TOLERANCE 0.1
LOOPS 8
TRACE 0
NOGO
DATA ECHO
CALCULATE
EXIT

```

Figure 11 - 1 : Batch-mode Data Input to PEETPACK.

(Cont'd)

Figure 11 - 2 : Interactive Data Input to PEETPACK.

\*\*\*\* PEETPACK MK. 1 \*\*\*\*

CASE STUDY TITLE : ETHANE - ETHYLENE SPLITTER PLANT

PROCESS FLOWSHEET DATA

NUMBER OF UNITS IN THE PLANT : 13

UNIT NAME : VALV1  
INPUT STREAMS : FEED1  
OUTPUT STREAMS : S2

UNIT NAME : DC1  
INPUT STREAMS : S2 S17 S7  
OUTPUT STREAMS : S3 PRODUCT1 S8

UNIT NAME : HE1  
INPUT STREAMS : S3 S14  
OUTPUT STREAMS : S4 S15

UNIT NAME : COMP1  
INPUT STREAMS : S4  
OUTPUT STREAMS : S5

UNIT NAME : VALV2  
INPUT STREAMS : S5  
OUTPUT STREAMS : S6

UNIT NAME : SPLIT1  
INPUT STREAMS : S6  
OUTPUT STREAMS : S7 S10

UNIT NAME : COOLR  
INPUT STREAMS : S10  
OUTPUT STREAMS : S11

UNIT NAME : VALV3  
INPUT STREAMS : S8  
OUTPUT STREAMS : S9

UNIT NAME : VALV4  
INPUT STREAMS : S11  
OUTPUT STREAMS : S12

UNIT NAME : MIXER  
INPUT STREAMS : S9 S12  
OUTPUT STREAMS : S13

UNIT NAME : VALV5  
INPUT STREAMS : S13  
OUTPUT STREAMS : S14

UNIT NAME : SPLIT2  
INPUT STREAMS : S15  
OUTPUT STREAMS : S16 PRODUCT2

UNIT NAME : VALV6  
INPUT STREAMS : S16  
OUTPUT STREAMS : S17

EXIT ? NO

DO YOU WISH TO INPUT YOUR OWN CALCULATION ORDER ? NO

LIST OF PREFERRED STREAMS

S6  
S16

DO YOU ACCEPT THEM ? NO

HOW MANY PREFERRED STREAMS DO YOU WISH TO INPUT ? 2

NAMES OF YOUR PREFERRED STREAMS : S17 S4

LIST OF PREFERRED STREAMS :

S17  
S4

DO YOU ACCEPT THEM ? YES

EXIT ? NO

UNITS PARAMETERS

UNIT : VALV1

HOW MANY PARAMETERS ? 1  
PARAMETERS : 215

UNIT : DC1

HOW MANY PARAMETERS ? 9  
PARAMETERS : 40 20 235 235 5.2 215 1 2 0

UNIT : HE1

HOW MANY PARAMETERS ? 6  
PARAMETERS : -7 0 0 0 0 1

UNIT : COMP1

HOW MANY PARAMETERS ? 4  
PARAMETERS : 585 .75 .94 1.23

UNIT : VALV2

HOW MANY PARAMETERS ? 1  
PARAMETERS : 450

UNIT : SPLIT1

HOW MANY PARAMETERS ? 2  
PARAMETERS : .91 .09

UNIT : COOLR

HOW MANY PARAMETERS ? 2  
PARAMETERS : -17 0

UNIT : VALV3

HOW MANY PARAMETERS ? 1  
PARAMETERS : 414

UNIT : VALV4

HOW MANY PARAMETERS ? 1  
PARAMETERS : 414

UNIT : MIXER

HOW MANY PARAMETERS ? 0

UNIT VALV5  
 HOW MANY PARAMETERS ? 1  
 PARAMETERS : 295

UNIT : SPLIT2  
 HOW MANY PARAMETERS ? 2  
 PARAMETERS : .85 .15

UNIT : VALV6  
 HOW MANY PARAMETERS ? 1  
 PARAMETERS : 215

EXIT ? NO

UNITS MODEL NAMES :

TYPE THE MODEL NAME AFTER THE UNIT NAME :

VALV1 VALVE  
 DC1 DISCOL  
 HE1 HEATEACH  
 COMP1 COMPRESR  
 VALV2 VALVE  
 COOLR HEATER  
 VALV3 VALVE  
 VALV4 VALVE  
 MIXER MIXER  
 VALV5 VALVE  
 SPLIT2 SPLITTER  
 VALV6 VALVE

EXIT ? NO

COMPONENTS :

TOTAL NUMBER OF LIBRARY COMPONENTS : 3

LIBRARY COMPONENTS NAMES : ETHYLENE ETHANE PROPANE

DO YOU WISH TO ADD OTHER COMPONENTS NOT IN THE PEETPACK LIBRARY ?  
 NO

DO YOU WISH TO INPUT ANY ADDITIONAL PROPERTIES ? NO

WHICH TYPE OF UNITS ? (BRITISH OR S.I.) BRITISH

EXIT ? NO

STREAMS SPECIFICATIONS :

INPUT STREAMS :

STREAM FEED1

FLOW (MOLAR) : 511  
 TEMPERATURE DEGF : -20  
 PRESSURE PSIA : 345  
 ENTHALPYBTU/LBML : 0  
 VAPOUR FRACTION : 0  
 ETHYLENE : .44  
 ETHANE : .54  
 PROPANE : .02

ITERATE STREAMS :

STREAM S17

FLOW (MOLAR) : 1200  
 TEMPERATURE DEGF : -40  
 PRESSURE PSIA : 215  
 ENTHALPYBTU/LBML : 0  
 VAPOUR FRACTION : 0  
 ETHYLENE : .95  
 ETHANE : .05  
 PROPANE : 0.0

STREAM S4

FLOW (MOLAR) : 1450  
 TEMPERATURE DEGF : -6  
 PRESSURE PSIA : 215  
 ENTHALPYBTU/LBML : 0  
 VAPOUR FRACTION : 0  
 ETHYLENE : .95  
 ETHANE : .05  
 PROPANE : 0.0

EXIT ? NO

SIMULATION CONTROLS :

MAXIMUM PERCENTAGE TOLERANCE : 1  
 MAXIMUM NUMBER OF LOOPS : 10  
 IS A CONVERGENCE PROMOTER REQUIRED ? YES  
 TYPE OF PROMOTER : 1  
 IS TRACING REQUIRED ? YES  
 FREQUENCY OF TRACING : 0  
 IS NOGO OPTION REQUIRED ? YES

EXIT ? NO

COMMAND ? TYPE STREAM FEED1

STREAM FEED1

FLOW (MOLAR) 511.00000  
 TEMPERATURE DEGF -20.00000  
 PRESSURE 345.00000  
 ENTHALPYBTU/LBML 0.00000  
 VAPOUR FRACTION 0.00000  
 ETHYLENE 0.44000  
 ETHANE 0.54000  
 PROPANE 0.02000

COMMAND ? TYPE PARAMETERS DC1

UNIT DC1 HAS 9 PARAMETERS

40.00000  
 20.00000  
 235.00000  
 235.00000  
 5.20000  
 215.00000  
 1.00000  
 2.00000  
 0.00000

COMMAND ? TYPE ROUTINE HE1

UNIT HE1 IS REPRESENTED BY MODEL HEATEXCH

COMMAND ? TYPE UNIT DC1

UNIT DC1

INPUT STREAM NO. 1 S2  
 INPUT STREAM NO. 2 S17  
 INPUT STREAM NO. 3 S7  
 OUTPUT STREAM NO. 1 S3  
 OUTPUT STREAM NO. 2 PRODUCT1  
 OUTPUT STREAM NO. 3 S8

COMMAND ? TYPE ALL COMPONENTS

COMPONENTS

ETHYLENE  
 ETHANE  
 PROPANE

COMMAND ? TYPE ORDER OF CALCULATION

CALCULATION ORDER  
 \*\*\*\*\*

SEQUENTIAL SET

VALV1

RECYCLE LOOP

COMP1  
 VALV2  
 SPLIT1  
 COOLR  
 VALV4  
 DC1  
 VALV3  
 MIXER  
 VALV5  
 HE1  
 SPLIT2  
 VALV6

COMMAND ? TYPE ALL ROUTINES

UNITS AND MODELS NAMES  
 \*\*\*\*\*

UNIT VALV1	REPRESENTED BY ROUTINE	VALVE
UNIT DC1	REPRESENTED BY ROUTINE	DISCOL
UNIT HE1	REPRESENTED BY ROUTINE	HEATEXCH
UNIT COMP1	REPRESENTED BY ROUTINE	COMPRESR
UNIT VALV2	REPRESENTED BY ROUTINE	VALVE
UNIT SPLIT1	REPRESENTED BY ROUTINE	SPLITTER
UNIT VALV3	REPRESENTED BY ROUTINE	VALVE
UNIT VALV4	REPRESENTED BY ROUTINE	VALVE
UNIT MIXER	REPRESENTED BY ROUTINE	MIXER
UNIT VALV5	REPRESENTED BY ROUTINE	VALVE
UNIT SPLIT2	REPRESENTED BY ROUTINE	SPLITTER
UNIT VALV6	REPRESENTED BY ROUTINE	VALVE

COMMAND ? EXIT

```

ETHANE / ETHYLENE PLANT ON DEETPACK MK.1
13 19 3 8 45 7 1 2 0 8 13 16 21 1 0 0
3 30 1 0
0.1000000E-02
FEED1 S2 S17 S7 S3 S3 PRODUCT1S8 S14 S4 S15
S5 S6 S10 S9 S9 S12 S16 PRODUCT2
VALV1 DC1 HE1 VALV2 VALV3 VALV4 MIXER1
VALV5 SPLIT2 VALV6
VALV6 DISCO1 HEATXCHCOMPRESRVALVE SPLITTERHEATER VALVE MIXER
VALVE SPLITTERVALVE
13 14 10 12 13 5 9 13 13 2 13 3 13
FLOW (MOIAR) TEMPERATURE PRESSURE ARS. ENTHALPY VAPOUR FRACTION
ETHYLENE ETHANE
0.2200000E 02 0.2810000E 04 0.7421000E 03 0.5092100E 03 0.1980000E 01
0.3053000E 03 0.9490000E 01 0.6080000E 01 0.6100000E 02 0.4323000E 00
0.8760000E 01 0.1333033E 05 0.3579026E 00 0.183274E 03 0.4920709E 07
-0.4794824E 01 0.9551773E 04 0.3455700E 04 0.7298506E 01
0.4400000E 03 0.5222000E 00 0.6100000E 03 0.1260000E 02 0.1112000E 03
0.2130000E 03 0.2730000E 01 0.7738070E 01 -0.2337480E 02 0.2804980E 04
0.1013940E 00 0.1000417E 01 0.9078175E 02 0.9192708E 04 0.4869002E 08
0.1006155E 08 0.0000000E 00 0.4140000E 03 0.4066000E 01 0.7611536E 01
0.2058882E 02 0.8486484E 04 0.9453242E 06 0.3804823E 08 0.0000000E 00
0.0000000E 00 0.2000000E 01 0.0000000E 00 0.5826600E 04 0.1000000E 03
ETHANE
0.3000000E 01 0.3010000E 02 0.7098000E 03 0.5500100E 03 0.2370000E 01
0.3325000E 03 0.1604000E 00 0.6050000E 01 0.6800000E 02 0.4340000E 00
0.8800000E 01 0.1590360E 05 0.3950500E 00 0.2354850E 03 0.5419241E 07
-0.5226095E 01 0.1011986E 05 0.3757500E 04 0.7277561E 01
0.4750000E 03 0.5166000E 00 0.5500000E 03 0.1080000E 02 0.1508000E 03
0.8675000E 04 0.3190000E 01 0.8300870E 01 -0.1891880E 02 0.1474410E 04
0.1437280E 00 0.8045569E 00 0.9207847E 02 0.1653079E 03 0.1271182E 05
0.3302101E 08 0.0000000E 00 0.4248000E 03 0.4384000E 01 0.6217944E 01
-0.1509117E 02 -0.1432870E 04 0.6807195E 06 0.2158669E 07 0.1893508E 09
0.5433102E 02 0.2000000E 01 0.0000000E 00 0.6330600E 04 0.1105000E 03
PROPANE
0.4000000E 01 0.4410000E 02 0.6174000E 03 0.6657500E 03 0.3210000E 01
0.4163000E 03 0.1538000E 00 0.6400000E 01 0.8400000E 02 0.4380000E 00
0.8890000E 01 0.1133416E 05 0.367178E 00 0.2768229E 03 0.9134956E 08
-0.3367751E 01 0.7178678E 04 0.4811800E 04 0.7392262E 01
0.5950000E 03 0.5922000E 00 0.5400000E 03 0.1080000E 02 0.2642000E 03
0.1637000E 03 0.2910000E 01 0.8801200E 01 -0.2412530E 02 0.2551270E 04

```

Figure 11-3 : Dump file for the ethane-ethylene separation plant problem.





ROUTINE MIXER CALLED TO SIMULATE UNIT MIXER1

INPUT STREAMS :

	S8	S11
FLOW (MOLAR)	1126.27200	217.72800
TEMPERATURE	6.05718	-23.00000
PRESSURE ABS.	414.00000	414.00000
ENTHALPY	-946.77146	-1746.94970
VAPOUR FRACTION	0.00000	0.00000
ETHYLENE	0.96577	0.96577
ETHANE	0.03423	0.03423
PROPANE	0.00000	0.00000

OUTPUT STREAMS :

	S12
FLOW (MOLAR)	1344.00000
TEMPERATURE	1.50421
PRESSURE ABS.	414.00000
ENTHALPY	-1076.40033
VAPOUR FRACTION	0.00000
ETHYLENE	0.96577
ETHANE	0.03423
PROPANE	0.00000

ROUTINE VALVE CALLED TO SIMULATE UNIT VALV5

INPUT STREAMS :

	S13
FLOW (MOLAR)	1344.00000
TEMPERATURE	1.50421
PRESSURE ABS.	414.00000
ENTHALPY	-1076.40033
VAPOUR FRACTION	0.00000
ETHYLENE	0.96577
ETHANE	0.03423
PROPANE	0.00000

OUTPUT STREAMS :

	S14
FLOW (MOLAR)	1344.00000
TEMPERATURE	0.99807
PRESSURE ABS.	399.00000
ENTHALPY	-1076.40033
VAPOUR FRACTION	0.00000
ETHYLENE	0.96577
ETHANE	0.03423
PROPANE	0.00000

ROUTINE HEATEXCH CALLED TO SIMULATE UNIT HE1

INPUT STREAMS :

	S3	S14
FLOW (MOLAR)	1344.00000	1344.00000
TEMPERATURE	-39.05606	0.99807
PRESSURE ABS.	215.00000	399.00000
ENTHALPY	2805.47644	-1076.40033
VAPOUR FRACTION	1.00000	0.00000
ETHYLENE	0.96577	0.96577
ETHANE	0.03423	0.03423
PROPANE	0.00000	0.00000

HEAT LOAD IN HEAT EXCHANGER = 0.3128875E 06 ENTHALPY UNITS

Figure 11 - 4 : Example of TRACE 0 mode printout.

Figure 11 - 5 : Example of TRACE 1 printout.

STREAM MATRIX FOR Loop 1

\*\*\*\*\*

	FEED1	S2	S17
FLOW (MOLAR)	595.00000	595.00000	1088.64000
TEMPERATURE	17.32600	-12.37423	-54.22677
PRESSURE ABS.	345.00000	215.00000	215.00000
ENTHALPY	-620.79000	-620.78892	-1311.72638
VAPOUR FRACTION	0.00000	0.17404	0.18218
ETHYLENE	0.44000	0.44000	0.96577
ETHANE	0.54000	0.34000	0.03423
PROPANE	0.02000	0.02000	0.00000

	S7	S3	PRODUCT1
FLOW (MOLAR)	1126.27200	1344.00000	339.00000
TEMPERATURE	101.24585	-39.05606	-4.69819
PRESSURE ABS.	464.00000	215.00000	215.00000
ENTHALPY	4214.50751	2805.47644	-1115.31665
VAPOUR FRACTION	1.00000	1.00000	0.00000
ETHYLENE	0.96577	0.96577	0.04032
ETHANE	0.03423	0.03423	0.92448
PROPANE	0.00000	0.00000	0.03520

	S8	S14	S4
FLOW (MOLAR)	1126.27200	1344.00000	1344.00000
TEMPERATURE	5.52929	0.35582	-7.94418
PRESSURE ABS.	414.00000	399.00000	215.00000
ENTHALPY	-961.17876	-1088.47302	3028.72980
VAPOUR FRACTION	0.00000	0.00000	1.00000
ETHYLENE	0.96577	0.96577	0.96577
ETHANE	0.03423	0.03423	0.03423
PROPANE	0.00000	0.00000	0.00000

	S15	S5	S6
FLOW (MOLAR)	1344.00000	1344.00000	1344.00000
TEMPERATURE	-13.71176	115.66879	101.24585
PRESSURE ABS.	299.00000	585.00000	464.00000
ENTHALPY	-1311.72638	4214.50751	4214.50751
VAPOUR FRACTION	0.03041	1.00000	1.00000
ETHYLENE	0.96577	0.96577	0.96577
ETHANE	0.03423	0.03423	0.03423
PROPANE	0.00000	0.00000	0.00000

	S10	S11	S9
FLOW (MOLAR)	217.72800	217.72800	1223.04000
TEMPERATURE	101.24585	-23.00000	4.09624
PRESSURE ABS.	464.00000	414.00000	414.00000
ENTHALPY	4214.50751	-1746.24578	-974.33470
VAPOUR FRACTION	1.00000	0.00000	0.00000
ETHYLENE	0.96577	0.96577	0.97608

PROPANE	0.00000	0.00000	0.00000
	S12	S13	S16
FLOW (MOLAR)	120.06000	1344.00000	1088.64000
TEMPERATURE	-23.80340	1.07208	-13.71176
PRESSURE ABS.	414.00000	414.00000	279.00000
ENTHALPY	-1761.92700	-1088.47302	-1311.72638
VAPOUR FRACTION	0.00000	0.00000	0.03041
ETHYLENE	0.97608	0.96577	0.96577
ETHANE	0.02392	0.03423	0.03423
PROPANE	0.00000	0.00000	0.00000
	PRODUCT2	S1	
FLOW (MOLAR)	255.36000	595.00000	
TEMPERATURE	-13.71176	17.52626	
PRESSURE ABS.	299.00000	345.00000	
ENTHALPY	-1311.72638	-620.78892	
VAPOUR FRACTION	0.03041	0.00000	
ETHYLENE	0.96577	0.44000	
ETHANE	0.03423	0.56000	
PROPANE	0.00000	0.02000	

Figure 11 - 5 : Example of TRACE 1 printout

(Cont'd)

Figure 11 - 6 : Data echo.

ETHANE / ETHYLENE PLANT ON DEFTPACK MK. 1

\*\*\*\*\*

PROCESS MATRIX

\*\*\*\*\*

UNIT NAME : VALV1

INPUT STREAM NO. 1 FEED1  
 OUTPUT STREAM NO. 1 S2

UNIT NAME : DC1

INPUT STREAM NO. 1 S2  
 INPUT STREAM NO. 2 S17  
 INPUT STREAM NO. 3 S7  
 OUTPUT STREAM NO. 1 S3  
 OUTPUT STREAM NO. 2 PRODUCT1  
 OUTPUT STREAM NO. 3 S8

UNIT NAME : HF1

INPUT STREAM NO. 1 S3  
 INPUT STREAM NO. 2 S14  
 OUTPUT STREAM NO. 1 S4  
 OUTPUT STREAM NO. 2 S15

UNIT NAME : COMP1

INPUT STREAM NO. 1 S4  
 OUTPUT STREAM NO. 1 S5

UNIT NAME : VALV2

INPUT STREAM NO. 1 S5  
 OUTPUT STREAM NO. 1 S6

UNIT NAME : SPLIT1

INPUT STREAM NO. 1 S6  
 OUTPUT STREAM NO. 1 S7  
 OUTPUT STREAM NO. 2 S10

UNIT NAME : COOLR

INPUT STREAM NO. 1 S10  
 OUTPUT STREAM NO. 1 S11

UNIT NAME : VALV3

INPUT STREAM NO. 1 S8  
 OUTPUT STREAM NO. 1 S9

UNIT NAME : VALV4

INPUT STREAM NO. 1 S11  
 OUTPUT STREAM NO. 1 S12

UNIT NAME : MIXER1

INPUT STREAM NO. 1 S9  
 INPUT STREAM NO. 2 S12  
 OUTPUT STREAM NO. 1 S13

UNIT NAME : VALV5

INPUT STREAM NO. 1 S13  
 OUTPUT STREAM NO. 1 S14

UNIT NAME : SPLIT2

INPUT STREAM NO. 1 S15  
 OUTPUT STREAM NO. 1 S16  
 OUTPUT STREAM NO. 2 PRODUCT2

UNIT NAME : VALV6  
 INPUT STREAM NO. 1 S16  
 OUTPUT STREAM NO. 1 S17

PLANT FEEDS :  
 \*\*\*\*\*

FEED1

PLANT OUTPUTS :  
 \*\*\*\*\*

PRODUCT1  
 PRODUCT2

STREAM MATRIX  
 \*\*\*\*\*

	FEED1	S2	S17
FLOW (MOLAR)	595.00000	595.00000	1142.40000
TEMPERATURE	17.32600	-23.81096	-34.60901
PRESSURE ABS.	345.00000	215.00000	215.00000
ENTHALPY	-620.79000	-1640.88600	-1541.35400
VAPOUR FRACTION	0.00000	0.00000	0.18633
ETHYLENE	0.44000	0.44000	0.97608
ETHANE	0.54000	0.54000	0.02392
PROPANE	0.02000	0.02000	0.00000

	S7	S5	PRODUCT1
FLOW (MOLAR)	1223.04000	1344.00000	539.00000
TEMPERATURE	104.76320	-39.51389	-4.27023
PRESSURE ABS.	464.00000	215.00000	215.00000
ENTHALPY	4267.24800	2797.51500	-1105.61400
VAPOUR FRACTION	1.00000	1.00000	0.00000
ETHYLENE	0.97608	0.97608	0.03087
ETHANE	0.02392	0.02392	0.93387
PROPANE	0.00000	0.00000	0.03526

	S8	S14	S4
FLOW (MOLAR)	1223.04000	1344.00000	1344.00000
TEMPERATURE	5.38842	1.25285	-5.75000
PRESSURE ABS.	464.00000	399.00000	215.00000
ENTHALPY	-994.33470	-1063.41800	3075.50000
VAPOUR FRACTION	0.00000	0.00000	1.00000
ETHYLENE	0.97608	0.97608	0.97608
ETHANE	0.02392	0.02392	0.02390
PROPANE	0.00000	0.00000	0.00000

	S15	S3	S6
FLOW (MOLAR)	1344.00000	1344.00000	1344.00000
TEMPERATURE	-8.60600	118.93020	104.76320

PRESSURE ABS.	399.00000	585.00000	464.00000
ENTHALPY	-1341.35400	4262.24800	4262.24800
VAPOUR FRACTION	0.00000	1.00000	1.00000
ETHYLENE	0.97608	0.97608	0.97608
ETHANE	0.02392	0.02392	0.02392
PROPANE	0.00000	0.00000	0.00000

	e10	s11	s9
FLOW (MOLAR)	120.96000	120.96000	1223.04000
TEMPERATURE	104.76320	-23.00000	4.09624
PRESSURE ABS.	464.00000	464.00000	414.00000
ENTHALPY	4262.24800	-1761.92700	-994.33470
VAPOUR FRACTION	1.00000	0.00000	0.00000
ETHYLENE	0.97608	0.97608	0.97608
ETHANE	0.02392	0.02392	0.02392
PROPANE	0.00000	0.00000	0.00000

	e12	s13	s16
FLOW (MOLAR)	120.96000	1344.00000	1142.40000
TEMPERATURE	-23.80340	1.74442	-8.60600
PRESSURE ABS.	414.00000	414.00000	599.00000
ENTHALPY	-1761.92700	-1063.41800	-1541.35400
VAPOUR FRACTION	0.00000	0.00000	0.00000
ETHYLENE	0.97608	0.97608	0.97608
ETHANE	0.02392	0.02392	0.02392
PROPANE	0.00000	0.00000	0.00000

PRODUCT2

FLOW (MOLAR)	201.60000
TEMPERATURE	-8.60600
PRESSURE ABS.	399.00000
ENTHALPY	-1341.35400
VAPOUR FRACTION	0.00000
ETHYLENE	0.97608
ETHANE	0.02392
PROPANE	0.00000

UNITS PARAMETERS

\*\*\*\*\*

VALV1 215.00000

DC1	40.00000	20.00000	256.00000	256.0
	4.25000	215.00000	1.00000	2.000
	0.00000	4.00000	1.00000	1.606
	1.69150	0.48410	-0.07410	1.061
	1.20900	0.35400	-0.21480	0.276
	0.44080	0.20980	0.12590	

HE1	-7.00000	0.00000	0.00000	0.0
	100.00000	1.00000		

COMP1	585.00000	0.75000	0.94000	1.2
-------	-----------	---------	---------	-----

VALV2 464.00000

SPLIT1 0.86000 0.14000

```

COLLR      -23.00000      0.00000
VALV3      414.00000
VALV4      414.00000
MIXER1     HAS NO PARAMETERS
VALV5      399.00000
SPLIT2     0.85000      0.15000
VALV6      215.00000

```

UNITS AND MODELS NAMES  
\*\*\*\*\*

UNIT	VALV1	REPRESENTED BY ROUTINE	VALVE
UNIT	DC1	REPRESENTED BY ROUTINE	DISCOL
UNIT	HE1	REPRESENTED BY ROUTINE	HEATEXCH
UNIT	COMP1	REPRESENTED BY ROUTINE	COMPRESR
UNIT	VALV2	REPRESENTED BY ROUTINE	VALVE
UNIT	SPLIT1	REPRESENTED BY ROUTINE	SPLITTER
UNIT	COOLR	REPRESENTED BY ROUTINE	HEATER
UNIT	VALV3	REPRESENTED BY ROUTINE	VALVE
UNIT	VALV4	REPRESENTED BY ROUTINE	VALVE
UNIT	MIXER1	REPRESENTED BY ROUTINE	MIXER
UNIT	VALV5	REPRESENTED BY ROUTINE	VALVE
UNIT	SPLIT2	REPRESENTED BY ROUTINE	SPLITTER
UNIT	VALV6	REPRESENTED BY ROUTINE	VALVE

PROBLEM DIMENSIONS  
\*\*\*\*\*

```

NUMBER OF UNITS =      13
NUMBER OF STREAMS =     19
NUMBER OF COMPONENTS =   3
STREAM LENGTH =        8

```

PROBLEM CONTROLS  
\*\*\*\*\*

```

MAXIMUM NUMBER OF LOOPS ALLOWED =     6
TOLERANCE ALLOWED =    0.00010
CONVERGENCE PROMOTER NO. 1 SET
TRACE SET ON ALL UNITS

```

CALCULATION ORDER  
\*\*\*\*\*

SEQUENTIAL SET

VALV1

RECYCLE LOOP

COMP1  
VALV2  
SPLIT1  
COOLR

VALV4  
DC1  
VALV3  
MIXER1  
VALV5  
HE1  
SPLIT2  
VALV6

END OF DATA ECHO

\*\*\*\*\*

DATA ECHO

Figure 11 - 6 : Data echo.

(Cont'd)

```

BATCHUPDATE PLANTDATA
PROCESS MATRIX
SETTER FEED1
      S1
VALV1  S1
      S2
VALV3
VALV4
MIXER1 S8 S11
      S13
ROUTINES
SETTER SETBP
PARAMETERS
DC1  40  20  256  256  4.25  215  1  2  50
     4  1  1.6063  1.6915  .4841  -.0741
     1.0613  1.209  .354  -.2148  .2769
     .4408  .2098  .1259
COOLR -23  50
NOORDER OF CALCULATION
STREAMS MATRIX
FEED1  595  -20  345  0  0  .44  .54  .02
S4     1340  -8  215  0  0  .95  .05  0
S17    1180  -40  215  0  0  .95  .05  0
DATA ECHO
CALCULATE
EXIT

```

Figure 11 - 7 : Updating commands in Batch-mode for the Ethane -Ethylene separation plant simulation.

factor in unit 6 (SPLIT1) and for the temperature of the compressor input stream (stream 4) until an acceptable situation is simulated. In relatively small plants, like this one, this solution may be acceptable, but for more complex and bigger plants, the cost could be prohibitively high and a different approach to correct iterate stream guessing should be developed, but at this stage the author cannot propose a better method.

### 11.2 Example Two

The Wegstein (57) convergence promotion algorithm has been used in many flowsheeting programs, including CONCEPT and PEETPACK, for three reasons:

1. It is a well known convergence accelerator which has been in existence for over fifteen years.
2. It operates very reliably and quickly on non-linear convergence problems.
3. It is known to cause diverging problems to converge with fair certainty (57).

Other more recent convergence promoters such as Orbach's (23) or Crowe's (107), have also received a certain share of attention, but to a lesser extent than Wegstein's method, because of their more recent publication. PEETPACK incorporates Orbach's method, not only to give the users a choice between different

accelerators, for choice-sake, but because it has proven to operate as well or even better than Wegstein's method on "linear" problems (unpublished results show however that Wegstein's method is though more powerful for non-linear or unstable systems).

The advantage of Orbach's method over that of Wegstein is illustrated in this second example.

Figure 11-8 shows a set of four splitters (triangular boxes) and two mixers (square boxes) forming three nested recycle loops (or one maximal loop). Both mixers and splitters are examples of "linear" systems or units since their outputs are described by a linear mathematical relationship in terms of their inputs. The feed stream to the system is set at 100 units of flow per unit of time and only mass balances on the total flow are considered (no components).

Initially the split factors in the SPLITTER models, representing the splitters, were set fairly evenly as follows:

<u>Unit</u>	<u>Stream order</u>	<u>Split ratios</u>
S1	3 to 4	1 to 4
S2	5 to 6	2 to 1
S3	8 to 9	1 to 2
S4	10 to 11	1 to 3

The iterate streams 2 and 10 were assumed free of flow

(zero flow rate) at the beginning of the iterations and the convergence tolerance was set at 0.1%. With no convergence promoter specified, the mass balances converged in 10 loop calculations following the unit order: S1 - S2 - M2 - S3 - S4 - M1. This fairly rapid convergence was due to the relatively even separations of flow between the two output streams of each splitter. The application of the convergence promoters had little effect because of the few number of iterations they could have saved on this fast converging system and because Orbach's accelerator is only applied once every 3 loop calculations. Nevertheless the Wegstein accelerator, which had had time to be applied 6 times to the first iterate stream (stream 2), helped the calculations to converge in 8 iterations rather than 10 and the Orbach accelerator achieved the same results in 9 iterations but was used only 3 times. Figure 11-9 shows the convergence of the flow rate in stream 2 for all three cases.

The same system was solved again but this time initial estimates were provided for the iterate streams. Stream 2 was assumed to bear 200 units of flow and stream 10, 50 units of flow. The unaccelerated calculations converged as expected in less iterations (9 iterations), but marginally so since more iterations are usually needed to reach the fine convergence tolerance than to jump ahead initially by a large change. Here again the effect

of the accelerators was not appreciable. Convergence was achieved in 8 iterations using Orbach's method and in 9 iterations using Wegstein's method (see figure 11-10). All the streams' converged flows are tabulated in column 2 in table 11-3.

The split factors in the splitters were then changed to the values shown below to slow down the convergence and hence put in evidence the value of the convergence promoters.

<u>Unit</u>	<u>Streams</u>	<u>Split fractions</u>
S1	3/4	1 to 10
S2	5/6	1 to 10
S3	8/9	10 to 1
S4	10/11	10 to 1

The convergence tolerance was decreased to 0.01% to ensure a more accurate convergence.

In unaccelerated mode and with no initial guesses, the flow in the first iterate stream, at the 100-iteration limit, was still very far from its converged value, as shown in figure 11-11. The application of the Wegstein promoter achieved the calculation convergence in 85 iterations, whereas Orbach's promoter yielded the same results in only 25 iterations. (Heavy dots on figure 11-11 show where the calculations would have stopped had a convergence tolerance of 0.1% been used).

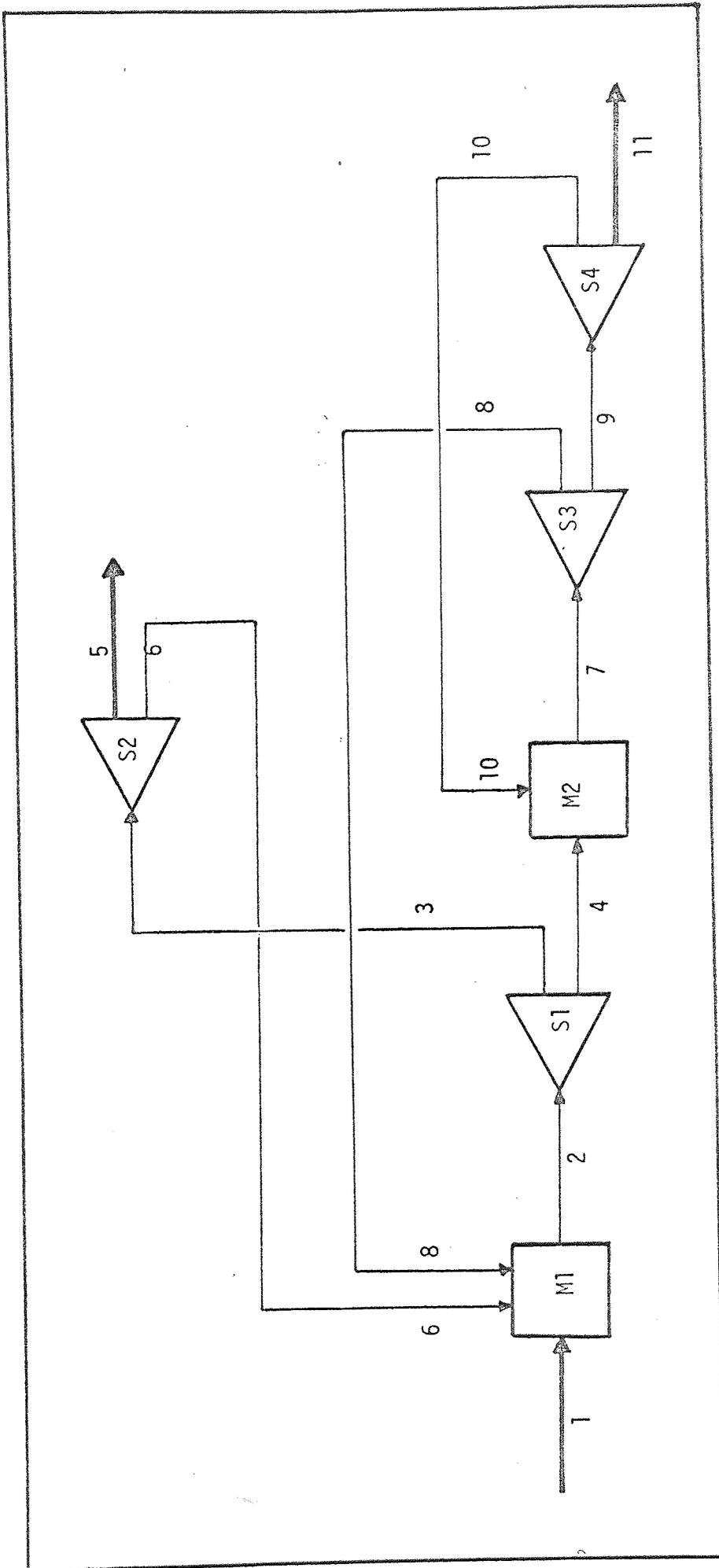


Figure 11 - 8 : The Mixer plant diagram

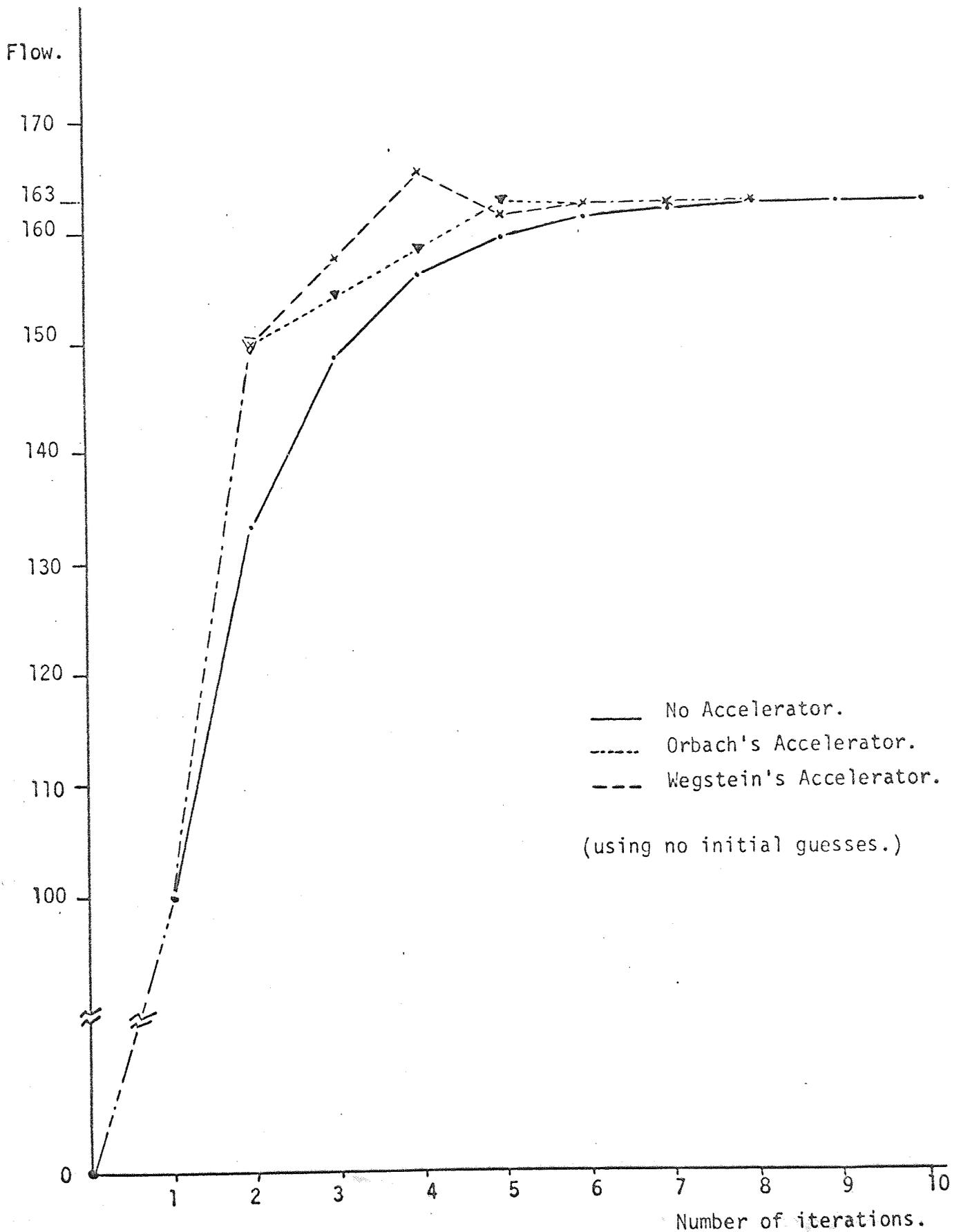


Figure 11 - 9 : First iterate stream convergence.

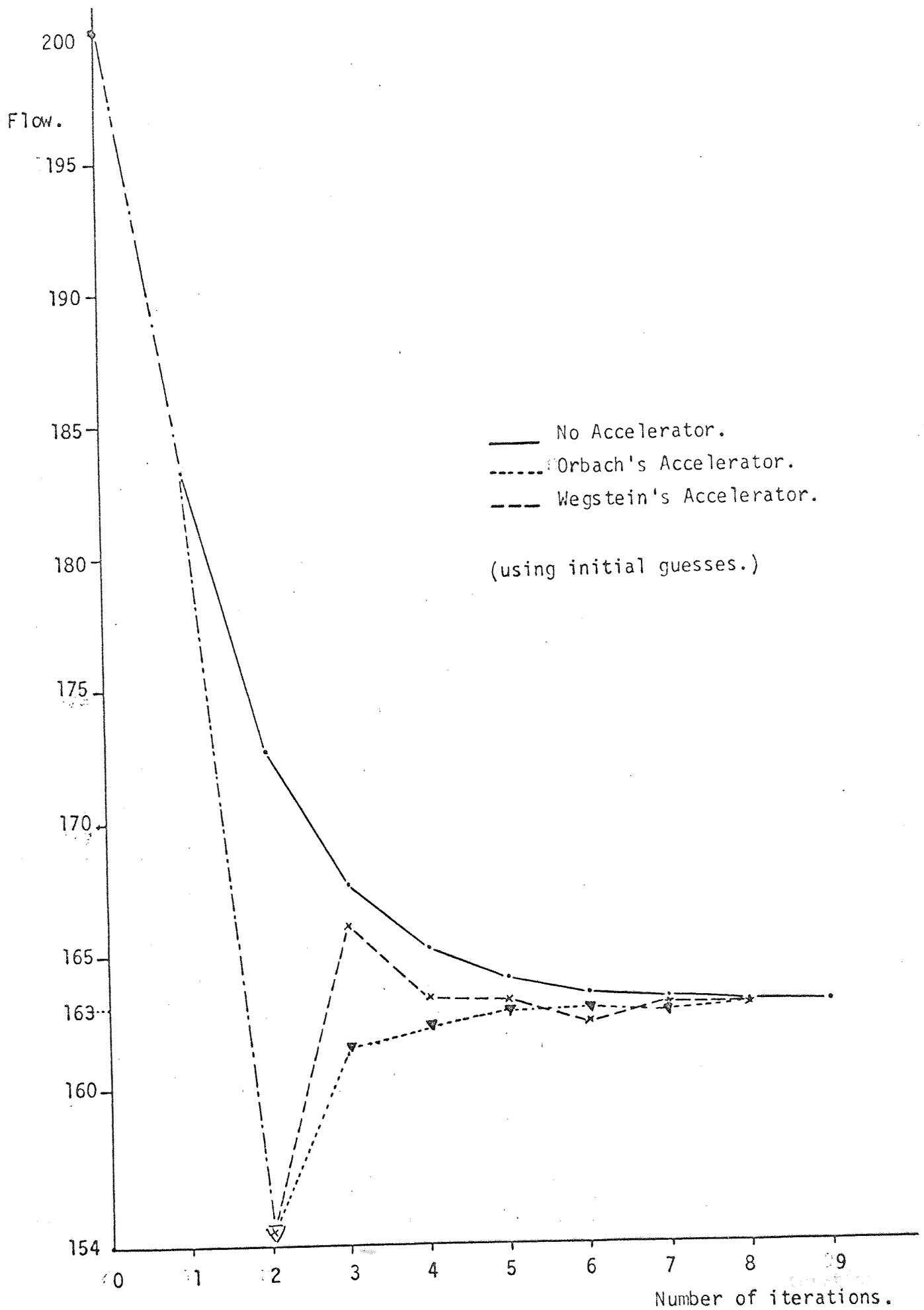


Figure 11 - 10: First iterate stream convergence.

(with initial estimates)

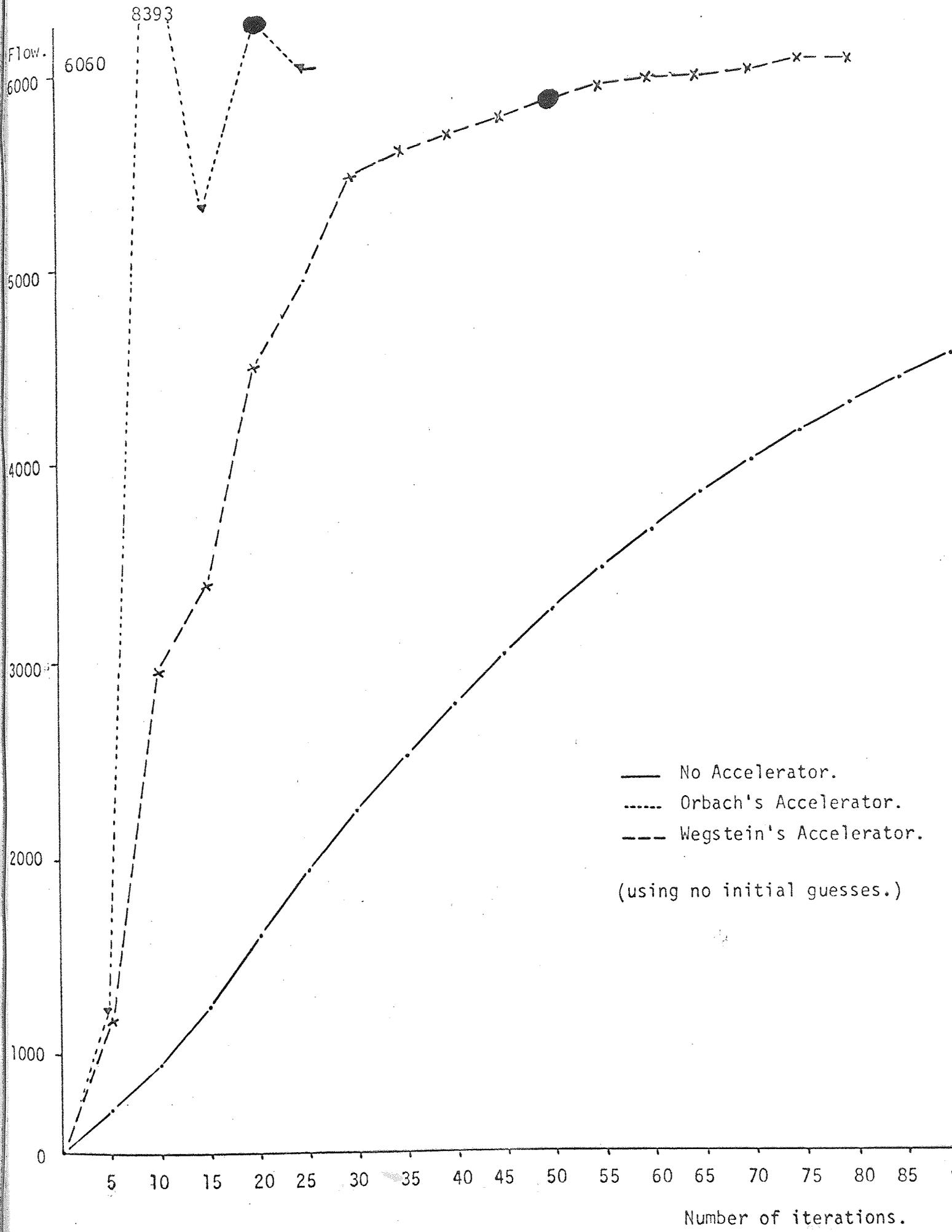


Figure 11 - 11 : First iterate stream convergence.

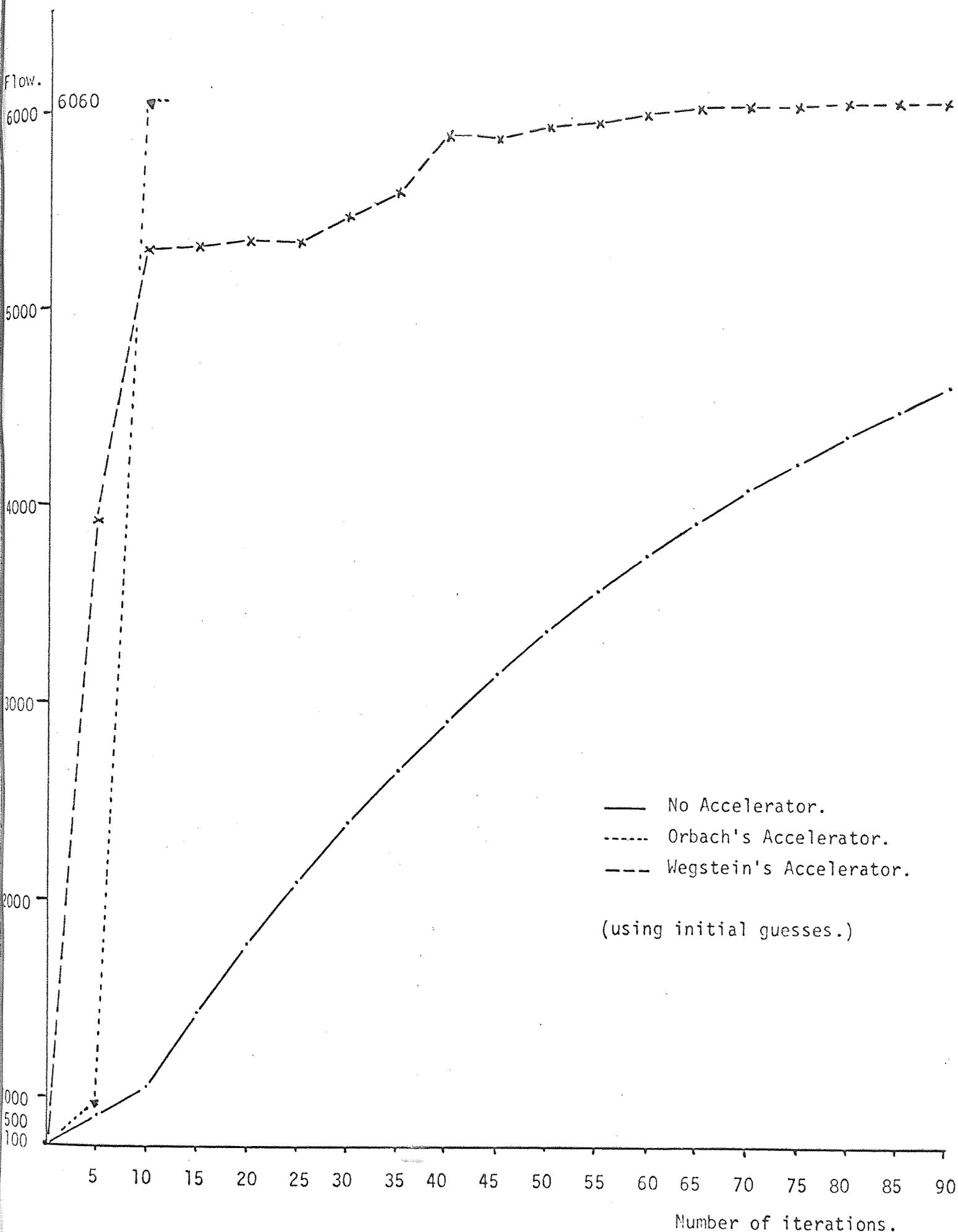


Figure 11 - 12 : First iterate stream convergence.

The iterate streams were given initial estimates of 100 and 50 units of flow for streams 2 and 10 respectively. These estimates, though nowhere near their converged values, had a disturbing effect on the accelerated convergence calculations. In the Orbach convergence calculations the number of iterations was reduced to an astonishing 9 iterations, whereas using Wegstein's accelerator, the calculations bore a sustained disturbance of low magnitude, yet above 0.01% of the stream value, which prevented them converging in 100 iterations though the stream value had reached the convergence region in about 80 iterations, as shown in figure 11-12. The investigation of this unexpected response was not investigated further due to time limitations. The converged values for all the streams in this example with large split factors are tabulated in column 3 in table 11-3.

This example shows clearly that Orbach's accelerator deserves more attention and is in certain cases superior to the well-tested Wegstein method.

### 11.3 Example Three

This example presents some of the limitations of the Chao-Seader Vapour-Liquid equilibrium ratios method (routine KVALUE).

Figure 11-13 shows the diagram of four flash columns in recycle (units F1 to F4). One of the two outputs from each column is compressed (in units P1 to P3) and returned as a recycle stream to be mixed (units M1 and M2) with the inputs to earlier columns in the system. The top product of the first flash column is cooled (in unit C1) and flashed at a lower pressure.

In this example two conditions were imposed on the operating conditions to test the response of the KVALUE routine.

1. The mixture would contain a large number of components (ten) some of which would be very light.
2. The pressures in the flash columns would be such that phase separation occurred in each column.

It was later found out that these two conditions could not be implemented separately and had to be incorporated by trial and error to meet the KVALUE routine operating conditions.

The major limitation of the Chao-Seader method resides in its range of allowable operating conditions. It may be applied only to mixtures with reduced pressures below 0.8 and where the reduced temperature of each component does not fall outside the range 0.5 to 1.3. This last condition allows the mixture to contain components with widely differing critical temperatures, since the ratio of the critical temperature of the heaviest component to that of the lightest can be as high as  $1.3/0.5 = 2.6$ ,

a very large range indeed. The constraint arises however out of the limitation on the range of the reduced temperature itself rather than the range of its critical properties. The added constraint imposed by the reduced pressure not exceeding 0.8 curtails even further the range of components allowed in a mixture undergoing large pressure changes.

In this example, the inclusion of components lighter than Ethylene (such as Hydrogen and Methane) was found almost impossible without violating the method's lower constraint. Ethylene was the lightest component that could be included. To avoid violating the upper constraint, the least volatile component had to be of the order of  $C_6$  or  $C_7$  (here n- Pentane).

To ensure a certain amount of vapourisation in each column, their pressures were set at the following values: 200 p.s.i.a., 175 p.s.i.a., 100 and 40 p.s.i.a. for columns F1, F2, F3 and F4 respectively.

The listing of the mixture components and of the feed stream are given in the data echo for the complete plant data and results, shown in figure 11-14. The iterate streams (S3 and S6) were also given initial estimates to avoid computation errors in the models (these are now being improved to take special precautions should they receive streams bearing no information).



Figure 11 - 14 : Data echo for the flash towers problem.

FLASH COLUMNS IN RECYCLE

PROCESS MATRIX

\*\*\*\*\*

```

UNIT NAME : M1
  INPUT STREAM NO 1      IN1
  INPUT STREAM NO 2      S1
  INPUT STREAM NO 3      S2
  OUTPUT STREAM NO 1     S3

UNIT NAME : F1
  INPUT STREAM NO 1      S3
  OUTPUT STREAM NO 1     S41
  OUTPUT STREAM NO 2     S5

UNIT NAME : C1
  INPUT STREAM NO 1      S41
  OUTPUT STREAM NO 1     S4

UNIT NAME : F2
  INPUT STREAM NO 1      S4
  OUTPUT STREAM NO 1     OUT1
  OUTPUT STREAM NO 2     S11

UNIT NAME : M2
  INPUT STREAM NO 1      S5
  INPUT STREAM NO 2      S6
  OUTPUT STREAM NO 1     S7

UNIT NAME : F3
  INPUT STREAM NO 1      S7
  OUTPUT STREAM NO 1     S21
  OUTPUT STREAM NO 2     S8

UNIT NAME : F4
  INPUT STREAM NO 1      S8
  OUTPUT STREAM NO 1     S61
  OUTPUT STREAM NO 2     OUT2

UNIT NAME : P1
  INPUT STREAM NO 1      S11
  OUTPUT STREAM NO 1     S1

UNIT NAME : P2
  INPUT STREAM NO 1      S21
  OUTPUT STREAM NO 1     S2

UNIT NAME : P3
  INPUT STREAM NO 1      S61
  OUTPUT STREAM NO 1     S6
  
```

PLANT FEEDS :

\*\*\*\*\*

PLANT OUTPUTS :  
\*\*\*\*\*

Figure 11 - 14 (Cont'd)

OUT1  
OUT2

STREAM MATRIX  
\*\*\*\*\*

	IN1	S1	S2
FLOW (MOLAR)	100.00000	56.82490	31.05230
TEMPERATURE DEGF	150.00000	22.62490	116.09500
PRESSURE PSIA	250.00000	250.00000	250.00000
ENTHALPYBTU/LBML	0.00000	-1781.60000	5299.15000
VAPOUR FRACTION	-1.00000	0.00000	0.93402
ETHYLENE	0.15000	0.19770	0.21072
ETHANE	0.15000	0.26736	0.28714
PROPYLENE	0.10000	0.16689	0.17135
PROPANE	0.10000	0.15016	0.15110
N-BUTANE	0.10000	0.05331	0.04457
I-BUTENE	0.10000	0.06245	0.05359
1,3-BUTADIENE	0.10000	0.05827	0.04933
1-PENTENE	0.10000	0.02156	0.01594
I-PENTANE	0.05000	0.01246	0.00925
N-PENTANE	0.05000	0.00984	0.00703

	S3	S41	S5
FLOW (MOLAR)	187.87700	88.58300	99.29420
TEMPERATURE DEGF	109.83200	97.85500	97.85540
PRESSURE PSIA	250.00000	200.00000	200.00000
ENTHALPYBTU/LBML	2573.05000	5380.22000	68.69790
VAPOUR FRACTION	0.42119	1.00000	0.00000
ETHYLENE	0.17447	0.28580	0.07514
ETHANE	0.20816	0.31281	0.11481
PROPYLENE	0.13202	0.13536	0.12904
PROPANE	0.12362	0.11784	0.12877
N-BUTANE	0.07672	0.03635	0.11272
I-BUTENE	0.08097	0.04305	0.11480
1,3-BUTADIENE	0.07901	0.04000	0.11385
1-PENTENE	0.06238	0.01416	0.10540
I-PENTANE	0.03191	0.00822	0.05304
N-PENTANE	0.03075	0.00645	0.05243

	S4	OUT1	S11
FLOW (MOLAR)	88.58300	31.69887	56.82490
TEMPERATURE DEGF	30.00000	22.62490	22.62490
PRESSURE PSIA	200.00000	175.00000	175.00000
ENTHALPYBTU/LBML	268.67300	3844.92000	-1725.88000
VAPOUR FRACTION	0.32582	1.00000	0.00000
ETHYLENE	0.28580	0.44534	0.19770
ETHANE	0.31281	0.39363	0.26736
PROPYLENE	0.13536	0.07832	0.16689
PROPANE	0.11784	0.05969	0.15016
N-BUTANE	0.03635	0.00588	0.05331
I-BUTENE	0.04305	0.00821	0.06245
1,3-BUTADIENE	0.04000	0.00701	0.05827

I-PENTANE	0.00822	0.00058	0.01246
N-PENTANE	0.00645	0.00034	0.00984

	S6	S7	S21
FLOW (MOLAR)	14.64890	114.11100	31.05250
TEMPERATURE DEGF	111.51100	101.78400	68.45880
PRESSURE PSIA	200.00000	200.00000	100.00000
ENTHALPYBTU/LBML	5097.10000	709.84300	5299.15000
VAPOUR FRACTION	0.87068	0.09994	1.00000
ETHYLENE	0.13979	0.08355	0.21072
ETHANE	0.23679	0.13024	0.28714
PROPYLENE	0.20810	0.13903	0.17135
PROPANE	0.18848	0.13642	0.15110
N-BUTANE	0.05707	0.10567	0.04457
I-BUTENE	0.06903	0.10905	0.05359
1,3-BUTADIENE	0.06295	0.10743	0.04933
1-PENTENE	0.01872	0.09425	0.01594
I-PENTANE	0.01101	0.04764	0.00925
N-PENTANE	0.00806	0.04672	0.00703

	S8	S61	OUT2
FLOW (MOLAR)	83.05900	14.64890	68.41050
TEMPERATURE DEGF	68.45900	31.04150	31.04150
PRESSURE PSIA	100.00000	40.00000	40.00000
ENTHALPYBTU/LBML	-1001.10000	5097.10000	-2507.90000
VAPOUR FRACTION	0.00000	1.00000	0.00000
ETHYLENE	0.03601	0.13979	0.01379
ETHANE	0.07159	0.23679	0.03621
PROPYLENE	0.12694	0.20810	0.10956
PROPANE	0.13093	0.18848	0.11861
N-BUTANE	0.12852	0.05707	0.14381
I-BUTENE	0.12978	0.06903	0.14279
1,3-BUTADIENE	0.12915	0.06295	0.14333
1-PENTENE	0.12353	0.01872	0.14598
I-PENTANE	0.06200	0.01101	0.07292
N-PENTANE	0.06156	0.00806	0.07301

UNITS PARAMETERS

\*\*\*\*\*

M1 HAS NO PARAMETERS

F1 200.00000

C1 30.00000

F2 175.00000

M2 HAS NO PARAMETERS

F3 100.00000

F4 40.00000

P1 250.00000

P2 250.00000

P3 200.00000

UNITS AND MODFLS NAMES

\*\*\*\*\*

UNIT	M1	REPRESENTED BY ROUTINE	MIXER
UNIT	F1	REPRESENTED BY ROUTINE	FLASHER
UNIT	C1	REPRESENTED BY ROUTINE	HEATER
UNIT	F2	REPRESENTED BY ROUTINE	FLASHER
UNIT	M2	REPRESENTED BY ROUTINE	MIXER
UNIT	F3	REPRESENTED BY ROUTINE	FLASHER
UNIT	F4	REPRESENTED BY ROUTINE	FLASHER
UNIT	P1	REPRESENTED BY ROUTINE	PUMP
UNIT	P2	REPRESENTED BY ROUTINE	PUMP
UNIT	P3	REPRESENTED BY ROUTINE	PUMP

PROBLEM DIMENSIONS

\*\*\*\*\*

NUMBER OF UNITS = 10  
NUMBER OF STREAMS = 15  
NUMBER OF COMPONENTS = 10  
STREAM LENGTH = 15

PROBLEM CONTROLS

\*\*\*\*\*

MAXIMUM NUMBER OF LOOPS ALLOWED = 10  
TOLERANCE ALLOWED = 0.00100  
CONVERGENCE PROMOTER NOT SET  
TRACE SET ON ALL UNITS

CALCULATION ORDER

\*\*\*\*\*

RECYCLE LOOP

F1  
C1  
F2  
M2  
F3  
F4  
P1  
P2  
P3  
M1

END OF DATA ECHO

\*\*\*\*\*

ATA ECHO

Figure 11 - 14: Data echo for the flash towers problem.

Table 11 - 1 : Results from the initial simulation.

Property	Location	Data	
		From survey	From simulation
Flows (lb moles/hr)	Feed	595.	595.
	Tops	256.	256.
	Bottoms	339.	339.
	Reflux	1088.	1088.
	Vapour Ov'head	1344.	1344.
	C <sub>2</sub> H <sub>4</sub> to reboiler	1142.	1216.
	C <sub>2</sub> H <sub>4</sub> to condenser	202.	127.
Compositions (C <sub>2</sub> H <sub>4</sub> mole %)	Feed	44.	44.
	Tops	95.	97.6
	Bottoms	3.	3.1
	Throughout plant	95.	97.6
Temperatures (Degrees F)	To plant	?	-20.
	Column top	-39.5	-39.5
	C <sub>2</sub> H <sub>6</sub> at reboiler	-1.	-4.3
	C <sub>2</sub> H <sub>4</sub> to reboiler	101.	101.5
	C <sub>2</sub> H <sub>4</sub> out of reboiler	5.1	2.5
	Inlet 2 to heat exchanger	-1.	-1.3
	Outlet 2 from heat exchanger	-15.	-9.4
	Compressor inlet	-8.1	-8.3
	Compressor outlet	113.	115.8
Pressures (p.s.i.a.)	To plant	345.	345.
	Column	215.	215.
	Compressor inlet	215.	215.
	Compressor outlet	585.	585.
	Inlet 2 to heat exchanger	405.	399.
	Outlet 2 from heat exchanger	295.	399.
Reflux ratio	To column	4.25	4.25
Vapour fraction	In reflux	.10	.17

Table 11 - 2 : Results from the improved simulation.

Property	Location	Data	
		From survey	From simulation
Flows (lb mole/hr)	Feed	595.	595.
	Tops	256.	256.
	Bottoms	339.	339.
	Reflux	1088.	1088.
	Vapour ov'head	1344.	1344.
	C <sub>2</sub> H <sub>4</sub> to reboiler	1142.	1142.
	C <sub>2</sub> H <sub>4</sub> to condenser	202.	202.
Compositions (C <sub>2</sub> H <sub>4</sub> mole %)	Feed	44.	44.
	Tops	95.	96.6
	Bottoms	3.	3.1
	Throughout plant	95.	96.6
Temperatures	To plant	?	17.3
	Column top	-39.5	-39.0
	C <sub>2</sub> H <sub>6</sub> at reboiler	-1.	-4.7
	C <sub>2</sub> H <sub>4</sub> to reboiler	101.	102.
	C <sub>2</sub> H <sub>4</sub> out of reboiler	5.1	6.
	Inlet 2 to heat exchanger	-1.	+1.
	Outlet 2 from heat exchanger	-15.	-13.7
	Compressor inlet	-8.1	-7.5
	Compressor outlet	113.	116.6
Pressures (p.s.i.a.)	To plant	345.	345.
	Column	215.	215.
	Compressor inlet	215.	215.
	Compressor outlet	585.	585.
	Inlet 2 to heat exchanger	405.	399.
	Outlet 2 from heat exchanger	295.	299.
Reflux ratio	To column	4.25	4.25
Vapour fraction	In reflux	.10	.18

Stream	Flow rates for	
	low splits case	high splits case
1	100.0	100.0
2	163.0	6058.6
3	32.6	550.8
4	130.4	5507.5
5	21.7	50.1
6	10.9	500.7
7	156.5	6003.7
8	52.2	5457.9
9	104.3	545.8
10	26.1	496.2
11	78.2	49.6

Table 11 - 3 : Flows for the Mixer plant example.

Stream S3 was set identical to the feed stream and stream S6 bore the same composition but its feed, temperature and pressure were set to 50 lb moles/hr, 0°F and 200 p.s.i.a. respectively.

The simulation converged to a 0.5% tolerance in 8 iterations and in about 2500 seconds of ICL-1905E computing time. The results are shown in figure 11-14.

This long computing time is another disadvantage of the Chao-Seader K-value method. Plants of more than a dozen units and operating on a large number of components would be prohibitively long to simulate. Because of the high cost of this run (£25), the investigation of the effect of the convergence promoters was not attempted although the original objective was in fact to study the effect of the Orbach and the Wegstein accelerators on this "non-linear" plant (hence the resemblance between figures 11-13 and 11-8).

#### 11.4 Conclusion

These examples point out to some of the more important features and drawbacks in PEETPACK. Other examples have been solved on PEETPACK to test its various facilities but are omitted for brevity. Out of these examples, a number of improvements and modifications to PEETPACK have become apparent; some of them were immediately implemented, the others are discussed in greater length in the following two chapters.

## CHAPTER 12

### "BEYOND CONVENTIONAL FLOWSHEETING" - DISCUSSION AND IDEAS

The modular approach to computer simulation of the steady-state heat, mass and energy flows in chemical plants has reached a high degree of development and sophistication. The various modern flowsheeting programs offer the user the utmost ease and flexibility in inputting and updating his data either from his own office by interactive, discussion type data communicators, or more quickly, but less comfortably and with less control on the correctness of the data, through cards and a card-reader in a computer terminal room. The user has also a large number of unit models to choose from to simulate his plant units, and the addition of his own exotic models is facilitated with the advent of flexible compiler languages (MACRO'S) which allow the incorporation of new routines into available programs during the course of a simulation. The range of physical and thermodynamic property data is constantly increasing in size and accuracy and is covering wider physical condition ranges. It is not surprising to find two, three or even up to twelve different vapour-liquid equilibrium ratio calculation routines in a single industrial flowsheeting program (52). These programs might also include a number of enthalpy calculation routines or other physical property estimation methods for various levels of accuracy or different physical conditions or mixtures of components.

From the point of view of conventional modular flowsheeting

executives, no major publicised break through has so far upset the pattern followed over the past ten years. Calculation order finders are being continuously refined though little evidence exists to decide whether any one of the order finders described in the literature or the one proposed in PEETPACK is really better than the others. Convergence promoters are also being varied and improved but here too the published evidence as to the decisive advantages of any one method is scarce (26, 33).

It may be argued that there is still plenty to do in flowsheeting since no program is capable yet of simulating completely any chemical plant proposed to it without the user adding certain models or data evaluation routines. This is not a matter of research as much as one of compiling and adding more and more data and models to existing programs, a feat not very practical nor even desirable in most application centres. Nevertheless, the researcher is still faced with the question.

#### 12.1 What can still be done?

There is still a lot of research to be done, not as much at the flowsheeting level as at the physical and thermodynamic property levels and in efficient and general model writing.

Most flowsheeting programs offer very sophisticated property packages that calculate thermodynamic, thermal and physical

properties of hydrocarbons, pure gases and related common compounds. Unfortunately little research has been done on modelling the behaviour of non-hydrocarbons, electrolyte solutions, inorganic liquids or solid mixtures etc. from a generalised equation-of-state or reduced-property type approach. Consequently flowsheeting programs are either confined broadly speaking to the petroleum and petro-chemical field (a large field, admittedly, but not all encompassing) or users in other chemical industries must provide their data in specific equations inserted in very specialised models (54). And consequently generalised unit model libraries contain mainly extractive-type routines based on these hydrocarbon and pure gases physical property evaluation packages.

It should be noted that even in the well studied field of hydrocarbon property prediction, research is still needed in dealing with polar compounds and in the development of generalised methods of predicting properties such as viscosity, specific heat and thermal conductivity of liquid, gas and two-phase mixtures over wide temperature and pressure ranges (95).

Another aspect of flowsheeting which does not seem to have been examined is the problem of requesting from the user some initial guesses for the iterate streams. All known flowsheeting programs, including PEETPACK, expect the user to supply fairly good guesses for these stream descriptions except for their

enthalpy which is calculated by the more elaborate programs. These then promote the convergence of the plant calculations based on these initial guesses. But should the guesses be in an utterly unfeasible region, the run will probably fail and the user will have to propose new (better?) guesses until a reasonable answer is obtained. This expensive trial and error game ought to be studied more carefully to reduce the cost and frustrations generated.

An important aspect in executive writing presently under development in Czechoslovakia (70) is the integration of the modular and the simultaneous equation approaches in the solution of a single problem. The simultaneous equation approach solves the mass balances in all units based on the representation of each unit by a fictitious splitter with fictitious split fractions. The modular program is then used to solve the particular unit models based on these flows and updates the split fractions for the fictitious models. The two methods are used successively until the heat and mass balances converge. The advantage of this method is that the mass balance equations are linear and can be solved very quickly by linear programming techniques. The efficiency of the method has not been meaningfully compared to either of the other two approaches. It does however deserve more consideration and does not seem to have been investigated in Western countries.

These and further developments along the same lines

retain flowsheeting concepts in the initial and restricted field they were developed for, namely simulating existing plant configurations based on assumed well-known or idealised operating conditions and feed streams, or the improvement of plant production by a trial and error application of these concepts. Such applications are obviously desirable and useful, otherwise flowsheeting would not have reached its present degree of sophistication. However, it is becoming increasingly imperative to develop flowsheeting beyond its conventional applications towards new applications where its available techniques coupled with new ideas could widen its areas of application and make it even more useful to a wider range of process engineers most of whom are not always interested in plant design or development.

## 12.2 "Beyond Conventional Flowsheeting"

A particular idea developed in conjunction with PEETPACK, and for which flowsheeting concepts are very appropriate, is the checking and reconciliation of plant survey reports and the estimation of consistent heat and mass flows from inaccurate or inconsistent spot stream values.

On-line computers are already commonly used in modern plants for the control of flows, pressures, temperatures etc., unfortunately they are usually restricted either to the most modern and expensive plants or to very delicate plants

necessitating accurate and tight control of their operating conditions, chiefly because of the complex mathematics involved and the expensive equipment required.

A more wide-spread, but cruder control technique is by individual meters and controllers on important variables. At regular time intervals all meters are recalibrated and possibly extra flows measured and a more accurate model of the plant is obtained. But even then all flow-meters, thermometers etc. are subject to certain errors in their readings, and the reconciliation of these readings becomes a trial and error problem. It is at this stage that flowsheeting programs may find a new field of application. This application may be worded as follows:

#### 12.2.1 Problem statement

"Given the description of a number of stream variables in a plant, each, including the feed streams, subject to a known deviation from its true value, and assuming that these errors are independent of each other and can occur on either side of the true value, - recreate a consistent heat and mass model of the plant so that the sum of the deviations between all meter readings and their calculated values is minimised."

The program should work given any number of specified readings, but not less than the minimum number of streams (basically the feed and the iterate streams).

### 12.3 The methods of data reconciliation

Computerised data reconciliation is as recent as flowsheeting but has received comparatively little attention in the West where only two approaches have evolved, and only one publication in English has appeared in the Eastern block.

Kuehn and Davidson (108) published the first known method of automatic off-plant data reconciliation. It uses the Lagrange method of undetermined multipliers to adjust the flow (and optionally the enthalpy) measurements to satisfy the total heat and mass balance equations around each plant unit. The algorithm aims at minimising the sum of the squares of the deviations between the meter readings and the reconciled flows. The approach is advantageous in that it takes into account the accuracy of the measurements of each flow separately and weights accordingly their contribution towards the minimisation of the sum of the squares of errors. It also operates only on the heat and mass balance equations the user wishes to set to account only for the known flows. The method is thus very general and easy to adjust to any plant or any set of metered flows merely by prior manual lumping together of units connected by unmeasured streams. The mathematics are fairly simple and presented in a way to reduce the computation time.

Ripps (109) has extended Kuehn and Davidson's method to account in the lack of data consistency for two different types of errors:

- a - Gross errors arising from strong instrument bias, major blunders in data reading, wrong instruments, etc.
- b - Small random errors due to normal meter inaccuracies.

His method of data adjustment permits this distinction. The analyst must first classify his data into either group having gross errors or else small errors, the adjustment method then balances the flows around all units; here also the least-squares criterion is satisfied. The mathematics involved are more elaborate than those in Kuehn and Davidson's method since two different types of errors must be distinguished.

A somewhat different application of the Lagrange undetermined multipliers was published in Czechoslovakia by Vaclavek (110). The material balance equations are set similar to those in the above two methods and the readings accuracies are also used to weight the deviations in the minimisation of the sum of squares. The method differs from the Kuehn and Davidson method in that it accepts and takes into account plant flows that are not measured and attempts to calculate an estimate either for their values or for the flows of certain combinations of streams in certain recycle cases. The presence of these additional unknown variables complicates the mathematical derivations; clear-cut

methods such as those developed in the above methods cannot be obtained because of the possibility of the final equations being underspecified, as illustrated in the same paper. In these cases stream combinations must be set manually to reduce the system of equations to a singular system. The final set of equations is then solved by a Gaussian elimination technique.

The second approach to data reconciliation, proposed by Clementson (111) does not involve the Lagrange undetermined multipliers. The method consists in setting the total mass balance equations around each plant unit in terms of its input and output streams, measured and unmeasured, thus obtaining  $M$  equations ( $M$  units) in  $N$  variables ( $N$  streams).  $N$  minus  $M$  variables are eliminated and represented in terms of the remaining  $M$  variables. The measured flows are then expressed in terms of the remaining  $M$  variables and a regression analysis method (112) is used to compute the best estimates for these  $M$  variables minimising the error sum of squares. The  $N-M$  remaining variables are then calculated by back-substitution into the original mass balance equations.

#### 12.3.1 Their advantages and drawbacks

The Vaclavek and Clementson methods are very similar in their critical analysis. They both estimate the missing flows in a plant and provide error bounds for the final results.

Unfortunately they both require manual intervention in setting the correct number of correct equations that will give a single solution to the given system. In the Vaclavek method this implies merging streams together or obtaining an estimate of the flow difference between two or more counter-current streams. In the Clementson method, the number of mass balances around the units must be reduced by combining the mass balances around the units with the equations defining the measured streams in terms of the unmeasured flows. Also this method does not account for the variation in meter accuracies. These are strong hindrances to the easy and general programming of the methods and user intervention which is unwanted is unavoidable.

Ripps' extension to Kuehn and Davidson's method is useful when both types of readings can be distinguished, those suffering of large errors and those subject to random instrument inaccuracies. However when this distinction is difficult to make, the user will have to investigate many possible trial assignments of the gross errors until a minimum error sum of squares is obtained. In the author's (N.P.) opinion, measurements known to be subject to gross errors should rather be discarded; and if the method cannot do without them, they should be calculated from unit models as described below.

Both Ripps' and Kuehn and Davidson's methods benefit of straight-forward programming because they expect either all streams

to be measured or unmeasured streams not to appear in the balance equations, thus no manual manipulation of these equations is necessary. They do not however estimate unmetered flows.

The advantages of Ripps' method having been discarded because of the trial and error procedure involved, the Kuehn and Davidson method remains as the more acceptable one. It is still unsatisfactory in that no unmeasured stream should appear in a reconciliation problem.

### 12.3.2 The Kuehn and Davidson reconciliation method

The problem is to minimise the error sum of square  $\phi(X)$  between the readings  $X_i^*$  and the corrected flows  $X_i$ , each error weighted according to the meter's error variance  $\sigma_i^2$

$$\phi(X) = \sum_i \frac{(X_i - X_i^*)^2}{\sigma_i^2} \quad (12.3-1)$$

subject to the constraint  $\psi(X)$  imposed by each unit (j) balance equation

$$\psi(X) = \sum_i a_{ij} X_i = 0 \quad (12.3-2)$$

where  $a_{ij}$  are the elements of the matrix A representing the input streams i to unit j ( $a_{ij} = +1$ ) and the outputs ( $a_{ij} = -1$ ) in an otherwise zeroed matrix.

The method of Lagrange multipliers solves the equations:

$$\frac{\partial}{\partial X_i} \left[ \phi(X_i) + \sum_j \lambda_j \psi_j(X_i) \right] = 0 \quad (12.3-3)$$

and

$$\frac{\partial}{\partial \lambda_j} \left[ \phi(X_i) + \sum_j \lambda_j \psi_j(X_i) \right] = 0 \quad (12.3-4)$$

In matrix notation the problem is

$$\left[ \begin{array}{c|c} C^{-1} & A^T \\ \hline A & 0 \end{array} \right] \begin{bmatrix} X \\ \lambda \end{bmatrix} = \begin{bmatrix} a \\ 0 \end{bmatrix} \quad (12.3-5)$$

where

- $C$  = the diagonal matrix with elements  $\sigma_i^2/2$
- $A$  = the stream connection matrix  $a_{ij}$
- $a$  = the vector  $2 X_i^*/\sigma_i^2$
- $\lambda$  = the Lagrange multipliers

A more computationally efficient form of (12.3-5) is:

$$\left[ \begin{array}{c|c} C & 0 \\ \hline -AC & I \end{array} \right] \left[ \begin{array}{c|c} C^{-1} & A^T \\ \hline A & 0 \end{array} \right] \begin{bmatrix} X \\ \lambda \end{bmatrix} = \left[ \begin{array}{c|c} C & 0 \\ \hline -AC & I \end{array} \right] \begin{bmatrix} a \\ 0 \end{bmatrix} \quad (12.3-6)$$

yielding: 
$$\left[ \begin{array}{c|c} I & CA^T \\ \hline 0 & -ACA^T \end{array} \right] \begin{bmatrix} X \\ \lambda \end{bmatrix} = \begin{bmatrix} Ca \\ -ACa \end{bmatrix} \quad (12.3-7)$$

$$\lambda = (ACA^T)^{-1} AX^* \quad (12.3-8)$$

or

$$X = X^* - CA^T \lambda \quad (12.3-9)$$

This system of equations is solved by applying a recursive equation of the form:

$$X_{\text{better}} = X_{\text{poor}} + \Delta X \quad (12.3-10)$$

where  $\Delta X$  would give an improvement and gradually decrease to zero so that the following equations would be satisfied:

$$\text{From (12.3-5): } C^{-1} X + A^T \lambda = a \quad (12.3-11)$$

$$\text{or at convergence: } C^{-1} (X + \Delta X) + A^T (\lambda + \Delta \lambda) = a \quad (12.3-12)$$

$$\text{so that by subtraction: } C^{-1} \Delta X + A^T \Delta \lambda = 0 \quad (12.3-13)$$

$$\text{the recursive equation is obtained: } \Delta X = CA^T \Delta \lambda \quad (12.3-14)$$

Similarly from equation (12.3-5) at convergence:

$$AX = 0 \quad (12.3-15)$$

But initially using the meter readings:

$$AX^* = \delta \quad (12.3-16)$$

where  $\delta$  are the material discrepancies (unbalances) in the balance equations.

Assuming the  $\Delta X$  give the correct answer then:

$$A(X^* + \Delta X) = 0 \quad (12.3-17)$$

and by subtraction:

$$A \Delta X = -\delta \quad (12.3-18)$$

In the same way equation (12.3-8) may be manipulated to give:

$$\Delta \lambda = (ACA^T)^{-1} A \Delta X \quad (12.3-19)$$

but through (12.3-18) and (12.3-16) a recursive equation may be

derived of the form:

$$\Delta \lambda = - (ACA^T)^{-1} AX \quad (12.3-20)$$

which by back-substitution into (12.3-14) and (12.3-16)

yields: 
$$X_n = X_{n-1} - CA^T (ACA^T)^{-1} AX_{n-1} \quad (12.3-21)$$

Initially  $X_{n-1}$  is set equal to  $X^*$  and the value of  $X_n$  ( $X_1$ ) represents an improvement which is substituted for  $X_{n-1}$  and the procedure repeated until

$$\left| \frac{(X_n - X_{n-1})}{X_n} \right| < \epsilon \quad (12.3-22)$$

where  $\epsilon$  is a preset convergence Tolerance.

### 12.3.3 The advantage of a flowsheeting context

A flowsheeting program with adequate unit models will obviate the need to arrange the mass balance equations manually to suit the available readings. There the user need only input the known readings, the process matrix describing the unit interconnections and other simulation data. The program should "sense" by itself the insufficiency of flow data and, rather than rearrange the mass balance equations, and generate the missing data by simulation. This has the added advantage of providing a complete and consistent plant model. The major disadvantage lies in the need for good models for the simulation of particularly complex or non-linear units.

#### 12.4 The implementation of Kuehn and Davidson's method in PEETPACK

An improved version of Kuehn and Davidson's method has been incorporated into PEETPACK and may be used via the data communicators. It has access to all the models and data routines in the package to calculate any unmeasured stream flow rates and other descriptions. For this reason the complete description of all measured streams must be input unless all streams are measured and their flows are known in mass terms since the conservation law does not apply to moles. The internal reconciliation is done on a mass basis and is applied only to the total flow rates. The program distinguishes between the mass and mole flow rates by the command used to run the program. The user should not attempt to guess unmeasured stream flows unless he has no good models to calculate them by.

##### 12.4.1 The use of the program

The user may use this facility by calling PEETPACK as for a normal simulation. He inputs his data as explained in chapter 6 except that since the calculation order is not needed and its calculation cannot be by-passed, he should input it manually to save time. Any order can be input as long as all units appear in it. The parameters of the units fully defined by measured values may be omitted since their model will not be called.

At the end of an interactive input session the user commands ERRORS and inputs the meters error percentage ranges following the name of each plant stream. Unmeasured streams or unknown errors should be given zero error ranges. In batch-mode a title-card saying ERRORS is followed by a number of cards each bearing a stream name followed by its error percentage range. Only known errors need be input. The program stores these data in array ERR.

To perform the calculation the user commands:

CALCULATE ERRORS.

The data communicator sets the flag IERR to 1 and the executive is loaded in core. It tests IERR and calls the reconciliation routine MINIMAX. The interactive features shown in figure 6-2 for data input checking and updating do not apply to ERRORS.

If the input flow rates and compositions (if entered) are on a mass basis rather than a mole basis, the calculation command should be:

CALCULATE ERRORS (MASS).

#### 12.4.2 The program description

The program checks all the meter error ranges and sets zero meter errors to the largest input error. It then checks for all unset flows by examining each unit's output flow rates.

For a unit with at least one unknown output, its inputs are checked and if they are all known, the appropriate model is called to calculate the outputs, but only the unknown streams descriptions are copied back to the stream matrix to avoid overwriting known plant readings. If the streams descriptions are input on a mass basis routine TOMOLE is called to convert them to a mole basis before each unit simulation and the outputs are converted back to a mass basis after the simulation. Their meter error ranges are set arbitrarily to twice the maximum error to give them less weight in the minimisation of the error sum of squares since the models may not be very accurate. The calculation of units with unknown input and output streams is postponed until their inputs are calculated. The scanning of all the units is repeated until all streams are known, then equation (12.3-21) is calculated in 11 steps.

1. If the input data are given on a molar basis (TERR = 1) routine TOMASS is called to convert all descriptions to a mass basis.
2. The connection matrix A is created by analysing the process matrix KPM. Each row in A represents a unit and each column a stream. Input streams to a unit are represented by + 1 in the appropriate column and outputs by - 1. The heat exchanger model is treated as two units each balancing one stream to ensure that the mass balancing

around the whole unit does not attribute to the corresponding input and outputs different flows for the sake of total mass balancing around the unit. Likewise the distillation column and its reboiler are treated as two separate units.

3. The error variance matrix C is created by setting each diagonal term to  $0.5 * \sigma_i^2$  where

$$\sigma_i^2 = \frac{\text{Stream error range } E(i) * \text{Flow } (i)}{100}^2 \quad (12.4-1)$$

since each row and each column represents a stream.

4. The transpose of matrix A is multiplied by matrix C and the results set in matrix B:

$$B = C * A^T = CA^T. \quad (12.4-2)$$

5. Matrix B is multiplied by matrix A to give matrix D:

$$D = A * B = ACA^T. \quad (12.4-3)$$

6. Matrix D is inverted:

$$D = D^{-1} = (ACA)^{-1}. \quad (12.4-4)$$

7. It is multiplied by matrix B:

$$D = B * D = (CA^T)(ACA^T)^{-1}. \quad (12.4-5)$$

8. Matrix A is multiplied by matrix D:

$$D = D * A = (CA^T(ACA^T)^{-1})A. \quad (12.4-6)$$

9. The improvement terms  $\Delta X$  to the meter readings are obtained and added to the known stream flow rates:

$$X_n = X_{n-1} + (CA^T(ACA^T)^{-1}A) * X_{n-1}. \quad (12.4-7)$$

10. The absolute value of the relative change in all

streams is compared to a preset tolerance and steps 9 and 10

are repeated until convergence is reached.

11. If the input stream descriptions were given on a molar basis, the results are converted back to a molar basis and the final reconciled data are printed by routine RESULTS and stored in a dump file by routine DPRINT.

The user has now got a reconciled set of flow rates and a consistent plant model.

## 12.5 Examples

### 12.5.1 Example One

This first example is a straight-forward application of the program to a set of four mixers/splitters in recycle. All 8 plant streams are measured to an accuracy of 10%. Figure 12-1 shows the plant diagram.

A number of inherent assumptions exist in the method and ought to be pointed out in the context of this example.

1. Since all flows are measured, the reconciliation method will calculate their reconciled values based solely on the criterion of minimising the error sum of squares. No precaution is taken to satisfy any of the unit parameters, such as the split fractions in this example, even if they are actually obeyed by the unit.

2. The stream errors  $E\%$  are used to find the absolute value of the stream deviation ( $E * \text{Flow}$ ) therefore, even though meters may be read to the same accuracy, the weighting factor  $\sigma$  depends on the total flow rate in the stream.
3. The absolute value of the error  $E$  is not important. What matters is the relative value of the errors  $E$  with respect to each other and more important the relative values of the variances  $\sigma^2$ . Therefore the same problem solved with all streams known within an error  $E$  or  $10 * E$  will yield the same answer, but the problem solved with equal variance  $\sigma^2$  on all measurements will yield very different results as illustrated below.

Table 12-1 shows the measured flow rates in row 2 and in row 3 the reconciled flows assuming an error meter of 10% (or even 5%) on all measurements. Row 5 shows the reconciled data assuming a flow meter error of 5 units of flow in all ranges. In this case the errors  $E$  were first calculated based on the assumption that  $\sigma = 5 = \text{Flow} (i) * E (i)$  and each flow's error  $E (i)$  is shown in row 4. It is significant to point out that the error sum of squares in the first case was 86.0 compared with only 60.1 in the second case. This is due to the larger weight given to the higher flow rates in the second case (a lower variance), thus forcing their reconciled values closer to their readings to minimise the error sum of squares.

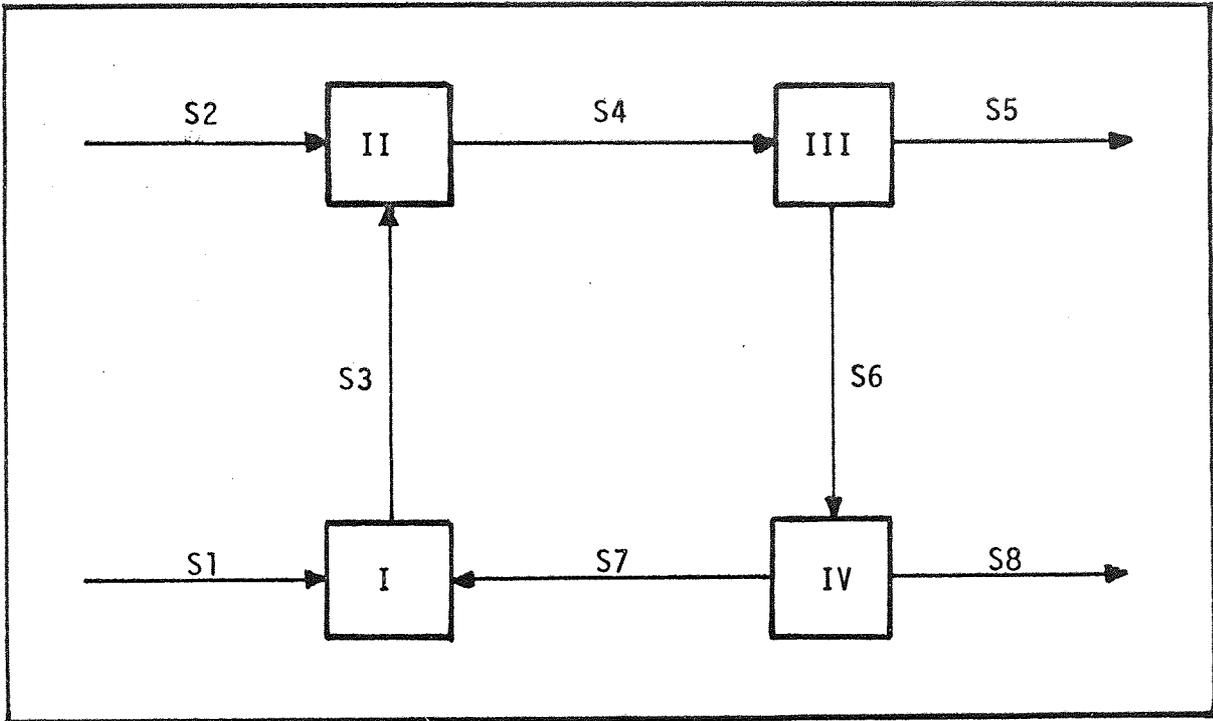


Figure 12 - 1 : Diagram for Example One.

Stream Number	1	2	3	4	5	6	7	8
Flow Measurement	65.	38.	95.	145.	85.	55.	38.	22.
Reconciled Flows	65.	39.2	102.1	141.3	82.5	58.8	37.1	21.7
Calculated Errors %	3.	5.	2.	1.5	2.5	4.	5.	10.
Reconciled Flows	63.4	42.7	99.7	142.1	85.	57.1	36.1	21.

Table 12 - 1 : Reconciled Flow rates for Example One.

### 12.5.2 Example Two

The second example is more realistic. The ethane-ethylene plant is used again and certain of its streams are given fictitious flow measurements that do not balance. The streams compositions, pressures and temperatures are assumed correctly known and set equal to the results of example 1 in chapter 11. All flows are assumed known within 5% of their true value except for the feed known to within only 10%.

Table 12-2 shows the measurements of the metered flows. Unmeasured flows are calculated using the same models as in the complete plant simulation and their parameters are set identically. In this case, the unit parameters have an important effect on the data reconciliation as they help set the unknown streams flows, but these are given a lesser weight in the reconciliation since the program assigns to them an error term  $E$  equal to twice the maximum error (10%) to reduce the effect of poor models or wrong parameters on the overall data reconciliation. The input data for this reconciliation are listed in Table 12-2.

The final reconciled data are also tabulated in Table 12-2 but since the program does not print the descriptions of the unmeasured streams as calculated by the models, only the final reconciled plant flows are shown.

Stream Number	Measurement (Actual)	Reconciled Flow
1	505.	533.4
2	-	533.4
3	1480.	1455.5
4	1485.	1455.5
5	-	1455.5
6	1460.	1455.5
7	1310.	1305.7
8	-	1305.7
9	-	1305.7
10	-	149.8
11	150.	149.8
12	-	149.8
13	1450.	1455.5
14	-	1455.5
15	-	1455.5
16	1200.	1215.5
17	-	1215.5
18	240.	239.9
19	-	293.1

Table 12 - 2 : Measurements and reconciled flows in example 2.

## 12.6 Conclusion

This new application of flowsheeting programs seems very useful to the author based on his short engineering experience in a Canadian Oil Refinery where manual data reconciliation occupied a process engineer periodically for a few days. It has the advantage that for plants where all streams are not measured it is still possible to obtain a complete and consistent plant model via unit modelling, or where good models are not available, guessed stream descriptions may be entered and given a very low weight in the reconciliation (a large error  $E$ ).

The method as proposed here has all the advantages and disadvantages of conventional flowsheeting since it uses most of the package features and routines. All the limitations described in the early part of the chapter may limit the scope of the plant reconciliation, but with time (or with a few good guesses) these can be overcome.

## CHAPTER 13

### CONCLUSION AND RECOMMENDATIONS

"It takes two years to write a flowsheeting program, but it takes a life-time to develop it" (Professor L.B. Evans, Massachusetts Institute of Technology).

The present state of PEETPACK is the fruit of two years of research work following a year's analysis of three of the best known or more easily available programs, PACER, GEMCS and CONCEPT.

During the first year of research, the requirements and objectives of flowsheeting programs were studied with the aim of improving them and making them more readily accessible to students and new users. When it was found that the better programs were all proprietary and not available for free use by interested users and that the nonproprietary programs were devoid of the most basic models and property data, a new program, PEETPACK, was created to make publicly available (113, 114) many of the features thought to be offered in the better industrial packages. To achieve this end a number of objectives were laid down, these were as follows:

1. The program should be written in a highly modular form.
2. Its data interface with the user should be very flexible and easy to learn.
3. It should contain a number of models and property generation

routines based on well documented methods.

4. It should offer an efficient calculation ordering algorithm.
5. Error tracing should help the user diagnose most error sources.

Most of these and other objectives have been achieved to a certain degree in PEETPACK as summarised below.

A. The objective of high modularity and high level language

PEETPACK is written entirely in FORTRAN IV, except for a very limited number of ICL compiler routines used to manipulate alphanumeric characters since these cannot be dealt with in standard FORTRAN. The Job Control Language commands (MACRO's) that control the operation and sequencing of the whole package had to be made computer dependent, here also because of the lack of standardised command languages. Otherwise the well documented package is easy to follow primarily because of its high modularity. The package itself is divided into a large number of subroutines (about 50 routines) each performing as few tasks as possible and relying on the minimum of other routines to reduce their interdependence and ease their improvement or change. The program's modularity ought to ease its overlaying to reduce its core requirements but this aspect was not studied.

B. The objective of ease of learning and data interface

The difficulty in teaching flowsheeting or understanding

flowsheeting programs stems from two points, the rigid and often confusing data input format and the complexity and poor documentation of the segments in the package. PEETPACK solves both problems. Its data input structure is very flexible. It allows the user to input his data in free format in a fairly variable sequence and in either interactive or batch mode or partly in both modes. The use of alphanumeric to represent his stream/units/compounds and the use of command words rather than depending on fixed data formats also help the user in learning and appreciating the features of the package. For the more ambitious users, the program's high modularity and the simplicity of its various routines should ease his familiarity with the whole package. In addition, no major deviations away from the standard format of known simulation programs were introduced to help other researchers follow its logic and improve it. This did however constrain the program's adaptability to large size problems as the dimensioning of array blocks was fixed in accordance with previous practice, but ways of overcoming this limitation are described later.

C. The program's models and property routines

PEETPACK offers a number of basic but fairly general unit models and its thermodynamic and physical property evaluation routines are based on recommended and well documented methods, though for the present time these are suitable mainly for non-polar hydrocarbons. Consequently the range of unit models is

aimed primarily at the petro-chemical Industry, and because of time limitations, only heat exchange, mass balancing and a few extractive processes have been modelled. The physical property package offers a number of properties necessary in the mechanical design of unit operations, though no mechanical design model is offered but these property routines may be applied to a larger range and different types of chemical compounds.

D. The calculation order finder

None of the known non-proprietary packages offers an efficient calculation order finder. This situation has been remedied in PEETPACK where an efficient and fast calculation ordering algorithm is proposed. It takes into account user-specified iterate streams and always specifies the minimum number of iterate streams, in the light of the user's choice. The maximum number of iterate streams allowed is 10 but this can be changed by correcting the array sizes in the common blocks. Users may nevertheless input their own calculation order manually hence by-passing this, occasionally time-consuming, algorithm.

E. Error Tracing

Most models and physical property routines in PEETPACK test the input stream conditions against the routine's ranges of applicability and check the sufficiency of their input information such as the unit parameters and other data. Insufficient data,

incorrect or out of range conditions cause the routine to issue warning or error messages. The warning messages help the user trace back incoherent or wrong final answers to possible data extrapolation in the physical property routines. The error messages are caused by insufficient data in the models. When possible the data are estimated or the calculation done without their use, otherwise appropriate messages are issued to help the user correct his data and the run is aborted. It was suggested at the Annual Research Symposium at Imperial College, London, (April 1974) that the program should be able to diagnose the error causes through a study of the progression of the run calculations and propose an improved strategy of calculation, but such a philosophy, though worth investigating, is not yet possible at this early stage in flowsheeting concepts.

The objective of error tracing and trapping does fall short of expectation on one crucial point, that of the thorough checking of the input data. In interactive mode, all data are input via the question-and-answer dialogue, but should this sequence be interrupted and the rest of the data input through the updater, or should it all be input batchwise, there is no program yet available to check them for sufficiency or consistency. This addition should be the first improvement brought to PEETPACK and followed subsequently by the recommendations for further additions described below.

The objectives set when writing PEETPACK were later

extended to include a new flowsheeting application. PEETPACK now offers a data reconciliation program which completes and "makes consistent" unbalanced plant survey data. This new feature does not seem to have been incorporated in other packages. This novel application of flowsheeting programs could be of interest to maintenance engineers and thus extend the field of application of these programs.

PEETPACK has been used to simulate a commercial chemical plant and has given results in good agreement with the actual data. Its convergence promotion facilities have been tested on recycle problems and have accelerated certain calculations appreciably.

PEETPACK's various features have been tested within the constraints imposed by the research budget and the available computer facilities, and have proved to operate smoothly and satisfactorily. However the package could be improved if the additions and alterations recommended below were accomplished.

### 13.1 Recommendations for future work

The improvements to PEETPACK may be classified as:

1. Further improvements to the executive and routine libraries.
2. Long-term additions and improvements to the models and thermo-physical properties.

### 13.1.1 Further improvements

A - The major area of improvement is the COMMON block.

The maximum number of units or streams allowed in any simulation is 30 in either case, and the maximum number of inputs or outputs to any one unit is 6. These maximum limits are imposed by the array size in the COMMON statements. The program should be changed to allow it to set these maximum sizes itself depending on the case simulated. This may be achieved either by introducing dynamic dimensioning or by restructuring the arrays so that they share common boundaries moveable at will from within.

B - The second problem is the interactive running of the executive. Because of the limitations imposed by the computing facilities at Aston University, a simulation cannot be performed interactively following the data input, consequently no rearrangement of the simulation was attempted to reduce its core requirements and improve its chances of quick loading. This may require certain changes to the CALL pattern between the models and the property routines to facilitate the overlaying of the various subroutines.

C - The length of a simulation run seems controlled by two major factors:

1. The computing time of the KVALUE routine which may take up to 50 or 80% of the whole simulation time.
2. The intermediate convergence checks in routines such as FLASH, DEWBUB, KVALUE, TEMP, etc.

In PEETPACK these checks drive the convergence in each routine and at every calculation to a very fine tolerance; this is a long and unnecessary requirement when the streams are not close to their converged values. These fixed controls should be relaxed at the beginning of a simulation and tightened progressively as convergence is achieved.

D - The property data files assignment in the data communicators is such that additions to file PROPTS or PPTEXTRA must be met by appropriate additions in the other file. And when the user assigns his own file equivalent to PROPTS or PPTEXTRA, he must also assign an equivalent one to the other file, containing all the compound names and at least one data point for each compound even if it is a zero. The files assignments and data reading should be re-programmed to impose less constraints on the user.

### 13.1.2 Long-term additions and improvements

In chapter 12 some topics for future research were discussed. They may all be applied to or incorporated into PEETPACK, however the main areas where the efforts should be concentrated are the following:

A - The addition of thermo-physical properties for water, carbon dioxide and oxygen. These components along with hydrogen are the principle inorganic compounds present most commonly in hydrocarbon mixtures and are very often present in reactor effluents. Property estimation for water, and particularly the vapour liquid equilibrium constants, is different from that of hydro-carbons

because of its strong polarity and its immiscibility in most non-polar hydrocarbon mixtures. Major routine modifications or new routine additions might be required.

B - The improvement of the K-value routine. Besides being the routine most often used in the property package, the vapour-liquid equilibrium ratio routine is also the most sensitive one to the range of critical (or reduced) properties it applies to. The Chao-Seader method operates on limited ranges of reduced properties and types of hydrocarbons. Mixtures of light and heavy hydrocarbons such as effluents from crackers or mixtures containing light gases ( $H_2$ ,  $CH_4$ ,  $CO_2$ ,...) cannot be modelled correctly. A variety of K-value routines ought to be incorporated into PENTPACK to cater for polar and non-polar light and heavy, organic and inorganic compounds. If the K-value routine cannot be extended to cover wide ranges of reduced properties, appropriate measures should be taken in routines FLASH to recognise two-phase systems where each phase contains compounds of extreme boiling point not present in the other phase.

C - Extending the ranges of applications of the physical property routines. The routines in chapter 9, section B, have defined ranges of applications which should be extended by additional relationships to suit new compounds or different operating conditions. In both cases the necessary research is more in the field of physical chemistry than programming.

D - More generalised simulation-orientated models. Simulation in its broadest meaning implies the reproduction of temperature and pressure drops as they seem within the units themselves rather than by imposing them externally via parameter arrays as is the case in the PNETPACK models. Better models should be written that would compute heat transfer rates, pressure drops, etc., based on actual unit geometries.

The type and amount of additions and improvements that can be brought to a flowsheeting program are infinite, as was so rightly implied by Professor Evans's comment earlier. Those proposed here are the more obvious and the more useful ones in simulation orientated programs.

### 13.2 The future of flowsheeting programs

The author is not in a position to predict in what direction flowsheeting programs will develop, as his industrial experience with these programs is nil, and different Universities tend either to repeat the same type of research or diversify in very new and different applications. However the kind of research both types of institutions ought to concentrate upon should be as follows:

13.2.1 Industry. Industry has a real need for large all-encompassing programs. Its main task should be in preparing generalised but accurate methods for estimating properties and

costs over wide ranges of conditions, since they are best equipped with the information and facilities to do so. Whether they retain the form of programs developed by Diamond Shamrock Corp. (35, 36, 37) or by Monsanto (38), or opt for smaller types of more specialised programs is unimportant as long as the information is made publicly available.

13.2.2 University. Academic institutions are not equipped with the same facilities or needs as Industry and their use of large packages is very restricted. University research should concentrate more on the theoretical bases of physical chemistry to produce better property correlations, on the mathematical techniques to provide better unit models, and above all on new applications of flowsheeting programs such as the simulation of processes operating in batch-mode, the dynamic (unsteady state) response of large plants to input disturbances, and the design of optimal control systems for whole sections of plants.

The author's strongest recommendation is in favour of freer exchange of information and programs between Universities to avoid unproductive repetition of efforts and to promote more advanced and coordinated research.

## BIBLIOGRAPHY.

- 1 - Flower J.R. and Whitehead B.D. ; The Chemical Engineer, No. 273 (1973), P 271.
- 2 - Leather A.J. ; Ph.D. Dissertation (1971), The University of Birmingham, U.K.
- 3 - Bubb F.W., Nisle R.G. and Carpenter P.G. ; J.Pet.Tech., 2 (1950), P 143.
- 4 - Peiser A.M. ; Proc. Am. Petrol. Inst. , 36 (1956), III, P 232.
- 5 - Peiser A.M. ; Ref. Eng., 29 (1957), P 11.
- 6 - Greenstadt J., Bard Y. and Morse B. ; Ind. Eng. Chem., 50 (1958), No. 11, P 1644.
- 7 - Taborek J.J. ; Chem.Eng. Prog., 55 (1959), No. 10, P 45.
- 8 - Tayyabkhan M.T. ; Ind. Eng. Chem., 54 (1962), No. 10, P 25.
- 9 - Loux P.C. ; Chem. Engng., 78 (1971), No. 16, P 66.
- 10- Kesler M.G. and Kessler M.M. ; World Petr. Annual Refn. Rev. 60.
- 11- Kesler M.G. and Griffiths P.R. ; Proc. Am. Petr. Inst., 43 (III) (1963), P 49.
- 12- James J.L. et al. ; Inst. Chem. Eng. Symp. Ser., 18 (1966), p 11.
- 13- Rosen E.M. ; Chem. Eng. Prog., 58 (1962), No. 10, P 69.
- 14- Sargent R.W.H. and Westerberg A.W. ; Trans. Inst. Chem. Engrs., 42 (1964), P T190.
- 15- Ravicz A.E. and Norman R.L. ; Chem. Eng. Prog., 60 (1964), No. 5, P 71.
- 16- Nagiev M.F. ; "The theory of recycle processes in chemical

- Engineering", McMillan, New York (1964).
- 17- Evans L.B., Steward D.G. and Sprague C.R. ; Chem. Eng. Prog., 64 (1968), No. 4, P 39.
  - 18- Johnson A.I. et al. ; Brit. Chem. Eng., 10 (1965), No. 11, P 770.
  - 19- Shannon P. et al. ; Chem. Eng. Prog., 62 (1966), No. 6, P 49.
  - 20- - ; Design and Construction, Chem. Engng., 73 (1966), No. 1, P 59.
  - 21- - ; CHIPS simulates chemical plants, Chem. and Eng. News, 44 (1966), No. 51, P 30.
  - 22- Rubin D.I. ; Chem. Eng. Prog. Symp. Ser., 58 (1962), No. 37, P 54.
  - 23- Crowe C.M. et al. ; "Chemical Plant Simulation", McMaster University Press, Canada (1969) and Prentice-Hall, New York (1971).
  - 24- Lederman P.B. ; Chem. Engng., 75 (1968), No. 25, P 127.
  - 25- Kehat E. and Sacham M. ; Process Technol. Internat., 18 (1973), No. 1, P 35.
  - 26- Kehat E. and Sacham M. ; Process Technol. Internat., 18 (1973), No. 3, P 115.
  - 27- Kehat E. and Sacham M. ; Process Technol. Internat., 18 (1973), No. 4, P 181.
  - 28- - ; Simulation programs speed optimisation, Chem. and Eng. Week, 46 (April 1, 1968), No. 15, P 46.
  - 29- Johnson A.I. et al. ; Brit. Chem. Eng., 13 (1968), No. 10, P 1432.
  - 30- Andrew S.M. ; Trans. Inst. Chem. Engrs., 46 (1968), No. 4,

P T123.

- 31- Andrew S.M. ; Brit. Chem. Eng., 14 (1969), No. 8, P 1057.
- 32- Andrew S.M. ; Trans Inst. Chem. Engrs., 47 (1969), No. 4,  
P T79.
- 33- Forder G.J. and Hutchison H.P. ; Chem. Eng. Sci., 24 (1969),  
P 771.
- 34- Forder G.J. and Hutchison H.P. ; Can. J. Chem. Eng., 48 (1970),  
No. 1, P 79.
- 35- Klumpar I. ; Chem.Engng., 76 (1969), No. 20, P 114.
- 36- Klumpar I. ; Chem. Engng., 77 (1969), No. 1, P 107.
- 37- Klumpar I. ; Chem. Engng., 77 (1969), No. 14, P 62.
- 38- Rorschach R.L. and Harris R.E. ; Oil and Gas J., 68 (1970).  
No. 33, P 62.
- 39- Monsanto Co. ; News letter, Dec. 10, 1973.
- 40- Menzies M.A. and Johnson A.I. ; Can. J. Chem. Eng., 49 (1971),  
No. 3, P 407.
- 41- Worley F.L. and Motard R.L. ; "Chemical Engineering Computing",  
Vol. 2, P 4, A.I.Ch.E., New York (1972).
- 42- Davies C. ; The Chemical Engineer, No. 248 (1971), P 49.
- 43- Johnson A.I. et al. ; Brit. Chem. Eng., 16 (1971), No. 10,  
P 923.
- 44- Johnson A.I. ; Brit. Chem. Eng., 17 (1972),No. 1, P 28.
- 45- Johnson A.I. ; Brit. Chem. Eng., 17 (1972),No. 2, P 119.
- 46- Johnson A.I. ; Brit. Chem. Eng., 17 (1972),No. 3, P 217.
- 47- ESSPROSS ; Dept. of Chemical Engineering, The University of  
Edinburgh, Kings Buildings, Mayfield Road, Edinburgh 9.
- 48- UNICORN ; Systems Engineering Group, Techn. Chem. Labor, E.T.H.

Zurich, Switzerland.

- 49- Flower J.R. and Whitehead B.D. ; The Chemical Engineer, No. 272, (1973), P 208.
- 50- Andrews J.M. and Flower J.R. ; First Annual Research Symposium, Inst. Chem. Engrs, London (April 1974).
- 51- Rippin D.W. ; Tech. Chem. Labor, E.T.H., Zurich, Switzerland, Private Communication (January 1973).
- 52- Seider W.D. ; "Chemical Engineering Computing", Vol. 2, P 32, A.I.Ch.E., New York (1972).
- 53- Johnson A.I. ; Dean of Engineering Sciences, The University of Western Ontario , London, Ontario, Canada.
- 54- GEMCS ; User Manual, The press of the University of Western Ontario, London, Ontario, Canada.
- 55- Hutchison H.P. and Forder G.J. ; I. Chem. E. Symp. Ser., No. 23, (1967), P 169.
- 56- Bending M. ; Dept of Chemical Engineering, The University of Cambridge, Cambridge U.K., Private Communication (April 1973).
- 57- Wegstein J.H. ; Comm.A.C.M., 1 (1958), No. 6, P 9.
- 58- Kilikas A.C. ; M.Sc. Thesis, Dept. of Chemical Engineering, The University of Aston, Birmingham, (1973).
- 59- Cooper-Bessemer of Canada Ltd. ; Rexdale, Ontario, Private Communication with Professor P.E. Barker.
- 60- Chao K.C. and Seader J.D. ; A.I.Ch.E.J., 7 (1961), No. 4, P 598.
- 61- Redlich O. and Kwong J.N.S. ; Chem. Review, 44 (1949), P 233.
- 62- York R. and White E.F. ; Trans. Am. Inst. Chem. Engrs, 40 (1940), P 227.
- 63- Barkeley C.H. et al. ; Chem Eng. Prog., 43 (1947), P 25.

- 64- Yen L.C. and Alexander R.E. ; A.I.Ch.E.J., 11 (1965),  
No. 2, P 334.
- 65- Petryschuk W.F. and Johnson A.I. ; Can. J. Ch. E., 44  
(1966), P 241.
- 66- Siddall J.N. ; OPTISEP Manual (1970), Dept. of Chem. Eng.,  
MacMaster University, Hamilton (Ontario), Canada.
- 67- Rudd D.F. and Watson C.C. ; "Strategy of Process Engineering",  
P 221, J. Wiley & Sons, New York (1968).
- 68- International Computers Limited ; "George 3 and 4 Operating  
System Manual" and "FORTRAN Compiler Libraries", 1900-series,  
ICL Technical Publications, London (1969).
- 69- Lee W. and Rudd D.F. ; A.I.Ch.E.J., 12 (1966), No. 6, P 1184.
- 70- Klemes J. et al. ; Coll.Czeck. Chem. Comm., 38 (1973), P 3544.
- 71- Norman R.L. ; A.I.Ch.E.J., 11 (1965), No. 3, P 450.
- 72- Himmelblau D. ; Chem. Eng. Sci., 21 (1966), P 425.
- 73- Jain Y.V.S. and Eakman J.M. ; "Chemical Engineering Computing",  
Vol. 2, P 40, A.I.Ch.E. , New York (1972).
- 74- Christensen J.H. and Rudd D.F. ; A.I.Ch.E.J., 15 (1969), No. 1,  
P 94.
- 75- Henley E.J. and Williams R.A. ; "Graph Theory in Modern Engi-  
neering", Academic Press, New York (1973).
- 76- Gottlieb C.C. and Corneil D.G. ; Comm.A.C.M., 10 (1967), No. 12,  
P 780.
- 77- Paton K. ; Comm.A.C.M., 12 (1969), No. 9, P 514.
- 78- Welch J.T. ; J.A.C.M., 13 (1966), No. 2, P 205.
- 79- Tiernan J. ; Comm.A.C.M., 13 (1970)? No. 12, P 722.
- 80- Stewart D.V. ; J.S.I.A.M.Numer.Anal., 2 (1965), No. 2, P 345.

- 81- Lee W. et al. ; A.I.Ch.E.J., 12 (1966), No. 6, P 1004.
- 82- Marter D.H. ; "Thermodynamics and the heat engine", Thames and Hudson, London (1960).
- 83- Wang J.C. and Henke G.E. ; Hydr. Process., 45 (1966), No. 8, P 155.
- 84- Guerreri G. ; Brit. Chem. Eng., 15 (1970); No. 7, P 927, and No. 8, P 1049.
- 85- Ellis S.R.M. ; The Chemical Engineer, No. 230 (1969), P 289.
- 86- Curl R.F. and Pitzer K.S. ; Ind Eng. Chem., 50 (1958), P 265.
- 87- Hildebrand J.H. and Scott R.L. ; "The solubility of non-electrolytes", 3rd edn., Reinhold, New York (1950).
- 88- Cavett R.H. ; Proc. Am. Petr. Inst. Div. Ref., 42 (III) (1962), P 351.
- 89- Prausnitz J.M. and Chueh P.L. ; "Computer calculations of high-pressure vapour-liquid equilibria", Prentice-Hall, New York (1968).
- 90- American Petroleum Institute ; Technical Data Book, A.P.I. Division of Refining, New York (1966).
- 91- Hirschfelder J.O. et al. ; Ind. Eng. Chem., 50 (1958), P 386.
- 92- Hirschfelder J.O. et al. ; Ind. Eng. Chem. Fund., 1 (1962), P 224.
- 93- Lydersen A.L. et al. ; Wisc. Univ. Eng. Exp. Sta. Rept. No. 4 (1965).
- 94- Chapman S. and Cowling T.G. ; "The mathematical theory of non-uniform gases", Cambridge University Press, Cambridge (1952).
- 95- Reid R.C. and Sherwood T.K. ; "The properties of gases and liquids", 2nd edn., McGraw-Hill, New York (1966).

- 96- Souders M. ; J. Am. Chem. Soc., 60 (1938), P 154.
- 97- Thomas L.H. ; J. Chem. Soc., (1946), P 573.
- 98- Gallant R.W. ; Hydr. Process., 47 (1968), No. 4, P 128.
- 99- Robbins L.A. and Kingrea C.L. ; Hydr. Process., 41 (1962),  
No. 5, P 133.
- 100- Misic D. and Thodos G. ; A.I.Ch.E.J., 7 (1961), No. 2, P 264.
- 101- Misic D. and Thodos G. ; J. Chem. Eng. Data, 8 (1963), P 540.
- 102- Sugden S. ; J. Chem. Soc., (1924), P 32.
- 103- Sugden S. ; J. Chem. Soc., (1924), P 1177.
- 104- Quayle O.R. ; Chem. Review, 53 (1953), P 439.
- 105- Timmermans J. ; "Physico-chemical constants of pure organic  
compounds", Elsevier, Amsterdam (1950).
- 106- Cavett R.H. ; Proc. Am. Petr. Inst. Div. Ref., 43 (III) (1963),  
P 57.
- 107- Orbach O. and Crowe C.M. ; Can. J. Chem. Eng., 49, (1971),  
P 509.
- 108- Kuehn D.R. and Davison H. ; Chem. Eng. Prog., 57 (1961), No. 6,  
P 44.
- 109- Ripps D.L. ; Chem. Eng. Prog. Symp. Ser., 61 (1965), No. 55, P 8.
- 110- Vaclavak V. ; Coll. Czech. Chem. Comm., 34 (1969), P 364.
- 111- Clementson A.T. ; Brit. Chem. Eng., 8 (1963), No. 8, P 564.
- 112- Davies O.L. ; "Statistical methods in research and production",  
3rd edn., Oliver and Boyd, London (1967).
- 113- Peters N. and Barker P.E. ; "PEETPACK, a new non proprietary  
Flowsheeting program", 1st Annual Research Symposium, Inst.  
Chem Engrs, London (April 1974).

114- Peters N. and Barker P.E. ; "PEETPACK, a new non-proprietary Flowsheeting program, (to be published in The Chemical Engineer, London).