DYNAMIC SIMULATION OF CHEMICAL PLANT

by

Mehmet Cetin Kocak, M.Sc.

A Thesis submitted for the
Degree of Doctor of Philosophy
to the
University of Aston in Birmingham

Department of Chemical Engineering

June 1980

DYNAMIC SIMULATION OF CHEMICAL PLANT

Mehmet Cetin Kocak Ph.D. June 1980

SUMMARY

A CSSL-type modular FORTRAN package, called ACES, has been developed to assist in the simulation of the dynamic behaviour of chemical plant. ACES can be harnessed, for instance, to simulate the transients in startups or after a throughput change.

ACES has benefited from two existing simulators. The structure was adapted from ICL SLAM and most plant models originate in DYFLO. The latter employs sequential modularisation which is not always applicable to chemical engineering problems. A novel device of twice-round execution enables ACES to achieve general simultaneous modularisation. During the FIRST ROUND, STATE-VARIABLES are retrieved from the integrator and local calculations performed. During the SECOND ROUND, fresh derivatives are estimated and stored for simultaneous integration. ACES further includes a version of DIFSUB, a variable-step integrator capable of handling stiff differential systems.

ACES is highly formalised. It does not use pseudo steady-state approximations and excludes inconsistent and arbitrary features of DYFLO. Built-in debug traps make ACES robust.

ACES shows generality, flexibility, versatility and portability, and is very convenient to use. It undertakes substantial housekeeping behind the scenes and thus minimises the detailed involvement of the user. ACES provides a working set of defaults for simulation to proceed as far as possible. Built-in interfaces allow for reactions and user-supplied algorithms to be incorporated. New plant models can be easily appended. Boundary-value problems and optimisation may be tackled using the RERUN feature. ACES is file-oriented; a STATE can be saved in a readable form and reactivated later. Thus piecewise simulation is possible.

ACES has been illustrated and verified to a large extent using some literature-based examples. Actual plant tests are desirable however to complete the verification of the library.

Interaction and graphics are recommended for future work.

Key Words

Dynamic chemical plant simulation CSSL.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to several people who contributed to this work in different ways.

As supervisor Dr. B. Gay continually guided and encouraged the author. Dr. J.P. Fletcher offered many useful criticisms. He also performed the tests on the ICF machine.

The Research School members, especially the Computation Group, proved to be great company. Technical issues, such as distillation and control, were discussed with them and useful tips resulted.

Both the technical and the clerical staff in the Department were very friendly. Mrs. M. Glendenning is worth particular mention. Her moral support eased the pressure of work.

My family were sympathetic and understanding as always. They kindly put up with my absence from home all these years.

Finally, the Turkish Government provided the financial aid without which this course could not have taken place.

This thesis is dedicated to my Family

CONTENTS

		Page
CHAPTER 1	INTRODUCTION	1
1.0	Introducing Simulation	1
1.1	Introducing Computers	2
1.2	Program Packages and Libraries	3
1.3	Modularity in Digital Computation	4
1.4	This Work in Outline	5
CHAPTER 2	LITERATURE SURVEY	6
2.0	Introduction	6
2.1	Integrated Automation	6
2.1.1	Types of Process Control	7
2.1.2	Control System Design	8
2.2	Integrated Chemical Plant Simulation	10
2.3	Dynamic Simulation	12
2.3.1	Some Chemical Engineering Applications	15
2.4	Dynamic Modelling in Chemical Engineering	16
2.4.1	Conservation Laws	16
2.4.2	Approximate Dynamic Modelling	18
2.5	Solution Tools of Dynamic Simulation	20
2.5.1	Numerical Solution of Algebraic Equations	22
2.5.2	Numerical Integration Requirements	23
2.5.3	Integration Stability and Stiff Systems	24
2.6	Digital Dynamic Simulators	26
2.6.1	Existing Simulators for Chemical Engineering	28

2.6.2 Continuous System Simulation Language (CSSL) 29
2.6.3 CSSL-Type Simulators 30
2.6.4 ICL SLAM 31
2.6.5 A Discrete Continuous Combined Simulator : GASP 33
2.7 Summary and Conclusions 34
CHAPTER 3 TOWARDS A NEW DYNAMIC SIMULATOR 36
3.0 Introduction 36
3.1 A Review of DYFLO 36
3.1.0 Introduction 36
3.1.1 Cause-and-effect 38
3.1.2 Information-Flow Diagrams 38
3.1.3 DYFLO : Integrators 39
3.1.4 DYFLO : Equilibrium Subprograms 40
3.1.5 Enthalpy Calculations in DYFLO 41
3.1.6 Dynamic Modules of DYFLO 41
3.1.7 DYFLO : Modules for Distillation 43
3.1.8 DYFLO : Miscellaneous 47
3.2 A Description of DIFSUB 48
3.3 The Conversion of DIFSUB 51
3.4 Summary and Conclusions 52
CHAPTER 4 ACES : THE NEW DYNAMIC SIMULATOR 54
4.0 Introduction 54
4.1 A Dynamic Simulation Methodology 54
4.2 The SIDES of an ACES Program 56

4.2.1 The ACES Executive 57

4.2.2 A Brief Study of the ACES Library 60

4.2.3 The USER SIDE 61

4.3 Housekeeping in ACES 65

4.3.1 SUBROUTINE HSKPSI 66

4.3.2 SUBROUTINE HSKPIN 67

4.3.3 SUBROUTINE HSKPDE 67

4.3.4 SUBROUTINES HSKPPA and HSKPTE 71

4.3.5 SUBROUTINE FRWRD and FUNCTION RTRV 71

4.3.6 SUBROUTINE GINT 72

4.3.7 SUBROUTINE KATC1 73

4.3.8 SUBROUTINE DINLE 74

4.4 The Integrators 76

4.5 The Control Subprograms 79

4.5.1 SUBROUTINE SNSR 79

4.5.2 SUBROUTINE CNTR 80

4.6 MODULES 81

4.6.1 SUBROUTINE MHLDP 81

4.6.2 SUBROUTINE MDHX 84

4.6.3 SUBROUTINE MGPLT 85

4.6.4 SUBROUTINE MHTR 87

4.6.5 SUBROUTINE MTRBC 88

4.6.6 SUBROUTINE MLXTR 88

4.7 SEMI-MODULES 90

4.7.1 SUBROUTINE SFLSH 90

4.7.2 SUBROUTINE SPCND 91

4.7.3 SUBROUTINE SSHX 92

4.7.4 SUBROUTINE SSLSH 92

4.8 PROCESS AUXILIARIES 93

4.8.1 SUBROUTINES XDTF and XDTR 94

4.8.2 SUBROUTINE XICHX 95

4.9 ALGEBRAIC AUXILIARIES 96

4.9.1 FUNCTION XDRVT 96

4.9.2 SUBROUTINES XTFN1 and XTFN2 97

4.10 Clearance Subprograms 97

4.11 Input/Output and ROLL-OUT 97

4.11.1 The Display Subprograms 98

4.11.2 The Reading Subprograms 99

4.11.3 The ROLL-OUT Feature 99

4.12 The Current Interfaces in ACES 100

4.13 The Reservations 101

4.14 The Exclusions 101

4.15 Summary 101

CHAPTER 5 ACES ILLUSTRATED AND VERIFIED 103

5.0 Introduction 103

5.1 The Computing Environment 103

5.2 Some Ordinary Differential Equations 104

5.2.1 TEST CK20 104

5.2.2 TEST CK21 109

5.3 TEST CK24 : PHSYBE 110

5.4 TESTS CK8 and CK7 113

5.5 TEST CK22 : Control on a CSTR Series 115

5.6 TEST CK5 : Distillation Control 117

5.7 TEST CK4 : Turbocompressor Control 126

5.8 TESTS CK3 and CK2 128

5.9 TEST CK11 : Dynamic Heat-exchangers 130

5.10 TEST CK10 : Dead-Time 134

5.11 What Remains Untested 136

5.12 Discussion and Conclusions 138

CHAPTER 6 DISCUSSION, CONCLUSIONS AND RECOMMENDATIONS 141

APPENDIX 1 A NOVEL CONVERGENCE PROMOTER 147

APPENDIX 2 DEFINITIONS OF ACES VARIABLES 150

APPENDIX 3 A CLASSIFICATION OF THE ACES LIBRARY 154

APPENDIX 4 THE STORAGE ARRANGEMENTS 155

APPENDIX 5 A PROGRAM DESCRIPTION SEGMENT FOR USERS 160

APPENDIX 6 ON THE PORTABILITY OF ACES 161

REFERENCES 163

ACES Library Variables 170

List of Figures

		Page
Figure 3.1	An Ordinary Stage in Distillation	44
Figure 4.1	The ACES Executive : SUBROUTINE HSPEX	58
Figure 4.2	An Example of SUBROUTINE USMLND	62
Figure 4.3	The UDE PART for a Chemical Plant Model	64
Figure 4.4	A Documentation Produced by SUBROUTINE HSKPIN	68
Figure 4.5	A Documentation Produced by SUBROUTINE HSKPDE	70
Figure 4.6	A High-level Report by SUBROUTINE DINLE	77
Figure 4.7	The Conventional Controllers in ACES	82
Figure 5.1	A Portion of SUBROUTINE USMLND for TEST CK20	105
Figure 5.2	SUBROUTINE UAL	107
Figure 5.3	SUBROUTINES UPED and UGINT for TEST CK20	108
Figure 5.4	The UDE PART for TEST CK24	112
Figure 5.5	The Plant Model for TEST CK22	116
Figure 5.6	The UDE PART for TEST CK22	118
Figure 5.7	The Plant Model for TEST CK5	120
Figure 5.8.1	The UIN PART for TEST CK5	122
Figure 5.8.2	The UTE PART for TEST CK5	122
Figure 5.8.3	The UDE PART for TEST CK5	123
Figure 5.9	The Plant Model for TEST CK4	127
Figure 5.10	The UDE PART for TEST CK3	129
Figure 5.11	The Plant Model for TEST CK2	132

List of Tables

		Page
Table 5.1	A Comparison of Integrators Using TEST CK21.	111
Table 5.2	A Summary of TEST CK22 Results.	119
Table 5.3	A Summary of TEST CK5 Results.	125
Table 5.4	A Summary of TEST CK3 Results.	131
Table 5.5	A Summary of TEST CK2 Results.	133
Table 5.6	A Summary of TEST CK11 Results.	135
Table 5.7	A Summary of TEST CK10 Results.	137

CHAPTER 1 INTRODUCTION

1.0 Introducing Simulation

Simulation is in a broad sense experimentation with models. The relationship between the model and the system it represents depends on the objectives of the investigation. For example, at various stages of process development a set of mathematical equations, an analogue, or a pilot plant may be harnessed in place of, or in conjunction with, an actual plant. This research is concerned with chemical process simulation by means of a mathematical model.

Simulation may be classified according to the nature of the mathematical models it employs such as stochastic against deterministic, transient (or dynamic) against steady-state, and discrete against continuous (Jacoby and Kowalik 1980, Perry and Chilton 1973). Stochastic simulation relies on the laws of probability and uses mean-values. On the other hand, deterministic simulation disregards probabilistic distributions and considers all variables to be single-valued. Steady-state models are time-invariant but dynamic models define how a system's behaviour alters with time. Irrespective of the actual system the model variables may change continuously or in discrete jumps and this is the basis of the final classification above.

Each class of simulation is equipped with analytical solution techniques albeit applicable only in some cases.

Matrices, Laplace- and z-transforms are examples of such methods. More frequently numerical algorithms are used because an analytical answer is either unavailable or intractable. This happens very often in continuous simulation, for instance, where the model generally comprises a set of simultaneous partial differential, ordinary differential/difference, and algebraic equations. The solution, numerical or otherwise, can be a formidable task and that is where computers can aid simulation.

1.1 Introducing Computers

The advent of computers has helped the cause of simulation considerably. Their versatility and convenience in solving a problem and providing a graphical display have invited users to move towards more elaborate and perhaps more representative models.

Analogue computers utilize electrical analogues of thermal, chemical, mechanical, or other physical systems in order to solve problems related to these systems. Analogue machines achieve parallel integration but require scaling of the problem and laborious "patching" of circuits. In addition, provision of non-linear functions, dead-time, and differentiation is difficult. Furthermore, analogue computers may necessitate a massive board for a large problem and since "storing" a solution for future use is almost impossible they are wasteful. In fact they have become obsolete as digital computers can outperform them and offer a greater potential.

Unlike their analogue counterparts, digital machines operate sequentially. They have capacity to handle a batch of programs at the same time but may be dedicated to one task. They may be employed off-line or interactively so that they communicate with a user, an actual plant, or another computer. Languages used for digital computation are low or high level. The most basic language is machine code, which is usually binary. This is the language in which the computer works and programs in any higher language have to be converted into this by compilation. FORTRAN and ALGOL are well established high-level languages for scientific programming. There exist also higher variants, specially designed for simulation, which may translate into FORTRAN or another high-level language prior to compilation.

1.2 Program Packages and Libraries

For the specialist computing is an enjoyable and challenging activity albeit time-consuming. Engineers and scientists on the other hand simulate a system in order to gain insight without detailed involvement. Specialists can develop, certify, and document program packages and libraries for general use. A package can concentrate in a certain area of computing such as optimization, statistics, numerical integration, sparse matrix operations, and graphics. Further, various simulators have been produced and distributed. This

approach makes economic sense since it facilitates the saving of experience and diminishes individual effort as well as the number of failed cases. For widespread usage however a package must be powerful, versatile, robust and also easy and convenient to harness.

1.3 Modularity in Digital Computation

A monolithic program may be satisfactory for a simple problem. In complex cases it leads to code repetition and makes debugging difficult. It is preferable to resort to some kind of devolution where a master program connects several subprograms with pre-assigned functions. In this approach, called modular, only an outline of logic remains set and subprograms are mutable, that is they may be replaced by a more sophisticated or more simple one as necessary. Thus, modularity leads to convenience and flexibility and provides room for expansion. Subprograms can be transferred from one case to the next which yields economy.

The notion of modularity can be extended to hardware thereby reducing dependence on a particular computer or peripheral. Pertinently Brignell and Rhodes (1975) suggest that hardware communication in computer systems be programmed at software level and not at hardware level; an important consideration for on-line applications. All programming, including packages, must be standardised in order to minimise machine-dependence.

1.4 This Work in Outline

This thesis is about a new digital dynamic simulator for chemical engineering. Chapter 2 reviews the relevant literature and prepares the background for the rest of the thesis. It identifies the requirements and desirable features of a dynamic simulation package. Chapter 3 concentrates on a well-known simulator that provides some plant models. A code for numerical integration is also considered. ACES, the product of this research, is detailed in Chapter 4 together with a methodology for dynamic simulation. The next chapter is devoted to illustration and verification. Chapter 6 discusses this contribution and makes recommendations for future work.

CHAPTER 2 LITERATURE SURVEY

2.0 Introduction

Computers are established entities in the chemical industry. On the one hand they facilitate via simulation elucidation of various aspects of the industry. On the other hand they may be employed on-line, that is as part of the plant, for automation and supervisory purposes and may in turn be simulated by other computers. This chapter prepares the background for the rest of the thesis by reviewing automation, steady-state and dynamic simulation, modelling, and the relevant numerical techniques.

2.1 Integrated Automation

The basic operating aim of any process plant is profitable production. Industrial experience indicates (Lee and Weekman 1976) that a plant should operate at a constraint unless it is over-designed. The constraints may be altered by disturbances, including policy changes, with the result that the profit margin may depend strongly, and sometimes crucially, on the capability of predicting and achieving in a reasonable manner and time the optimal point and maintaining the plant at that point. Computers have proved very useful in this respect.

Most chemical processes contain some element of sequential operation. Defining this broadly, Edwards

and Lees (1973) cite as capable of automation (i) plant startup (hot or cold start), (ii) plant shutdown (scheduled or emergency), (iii) batch or semi-batch operations, (iv) equipment changeover, (v) product quality changes, (vi) product throughput changes, (vii) equipment availability changes, and (viii) mechanical handling operations. Some unforeseen situations and some operations require manual control however. These must be allowed for at the design stage. Edwards and Lees (1973) make a distinction between automation and computerisation, the latter implying lack of integration in man-machine systems. The authors stress the importance of a balanced division of labour between man and automatic equipment.

2.1.1 Types of Process Control

Edwards and Lees (1973) are quoted on this topic:

"Normally on a process plant the control system is based on single loops on each of which is an analogue controller, either pneumatic or electronic. This controller receives from the measuring instrument a measured value of the controlled variable, compares this with a desired value or setpoint to obtain an error, performs a mathematical operation on the error and sends an output signal to the control valve, which then adjusts the regulated variable. The variables controlled and regulated may be the same, as in the case of simple control of flow, or different, as in the control of temperature by the regulation of steam flow...

Setpoint control means that the computer alters the setpoints of conventional, analogue controllers... This type of control is

normally carried out in order to implement methods such as feedforward control or optimisation. The plant measurements which these require are taken into the computer. Frequently some form of mathematical model is used in the calculations...

In direct digital control (DDC) the computer takes in measurements from the plant and sends out signals to the plants regulating elements... There need be no conventional controllers at all...

DDC is also much more flexible. It is possible to carry out many kinds of operation on the measurements, including the calculation of indirect measurements and the filtering of the measurement signals. Different control algorithms can be used, including non-linear or asymmetrical gains or algorithms with some logic in them. The configuration of the control system can be readily altered and different systems can be investigated. A DDC computer is normally supplied with all the significant plant measurements. It is therefore well-equipped to carry out operations such as data logging and alarm scanning on these. A setpoint computer often takes in only measurements required for its supervisory programs."

2.1.2 Control System Design

It is seen that digital computers increase the variety of control types and configurations. They can perform a multiplicity of tasks and are very flexible. Objectivity may be introduced by pre-programming

(Brignell and Rhodes 1975). The logs produced on-line may provide a post-mortem facility to find out possible causes of failure (Edwards and Lees 1973). Computer reports may improve man-man communication. Reliability and reproducibility are increased and the time scales of different phases reduced.

An integrated approach to automation is recommended from the design stage. To quote Lee and Weekman (1976) "if the process design engineer had had the foresight, the whole advanced control project would not have been necessary." Stainthorp and Valdman (1979) present a structure for a control system design package. They envisage reduction of possibilities by worst-case, steady-state feasibility studies prior to dynamic analyses to finalise the design. The authors describe a steady-state package which is an extension of work by Stainthorp and Benson (1974). According to Crowe et al. (1969), Stainthorp and Valdman (1979), and Lee and Weekman (1976) formal dynamic analysis is not carried out for some processes but the designer relies on past experience to devise a control scheme. Lee and Weekman (1976) are critical of this. Their criticism is justified considering the benefits to be derived from dynamic analysis on the one hand, and the evolution of computers and simulation packages on the other.

Worst-case, steady-state studies can lead to over-design and computer-aided analysis can be used to

balance this against risk. For example, error distributions could be assigned to many of the parameters and Monte Carlo experiments could be performed with parameter values within these error limits to ascertain their effect on plant operation and final product quality (Crowe et al. 1969).

2.2 Integrated Chemical Plant Simulation

The operation of a process depends on material, energy, and signal flows, and on local holdups. Computer simulation can conveniently determine these quantities. Nonetheless it is subject, like any other engineering activity, to economic constraints and its use has to be justified in comparison with available alternatives. Its successful utilisation may permit inferences to be drawn about systems (Pritsker 1974) (i) without building them if they are only proposed systems, (ii) without disturbing them if they are operating systems that are costly or unsafe to experiment with, (iii) without destroying them if the aim of an experiment is to determine their limits of stress.

Steady-state simulation is employed at the design stage. In reality however steady-state hardly exists - it is only an idealisation. It seems transient behaviour is prevalent in all chemical processes. Sequential operations mentioned in Section 2.1 generally cause substantial deviations from "steady-state". Dynamic

simulation can predict the transient path and this information can prove or rule out a control scheme, for instance.

Chemical plant calculations are notoriously characterised by large dimensionality, non-linearity, and interaction among process variables. This may explain why steady-state simulators like FLOWPACK, FLOWTRAN, and PACER are peculiar to chemical engineering (Motard et al. 1975, Myers and Seider 1976, Crowe et al. 1969). In other branches of science it may be preferable to solve the small set of equations directly.

Steady-state and dynamic simulation merge at several levels. Both may yield a set of partial differential or ordinary differential/difference equations coupled to algebraic equations and so may require a similar repertoire of numerical techniques in search of a solution. In fact, it is possible to solve a steady state problem using a crude dynamic model (Smith et al. 1970). Steady-state models frequently partake in dynamic simulation for approximation purposes and also to provide the initial conditions. Further the two types of simulation may share a common process data base and a physical properties package. In addition, both may be subject to optimisation to satisfy technical or managerial criteria.

A hierarchy of packages is evident then descending from optimisation through steady-state/dynamic simulation

to physical properties, and to numerical techniques. Evans and Seider (1976) depict process development steps and suggest that an advanced computing system should be file-oriented in order to retain the current description of an active problem and perhaps use it as input to the next analysis. The authors add that the description of a flowsheet should be independent of the programs for analysis. The same description should serve all analyses for which that flowsheet is applicable, for example, steady- and unsteady-state simulation, equipment sizing, thermodynamic availability balance, and economic evaluation.

For further unification it can be stipulated here that a systematic and consistent methodology is adopted throughout and that lower ranking packages accommodate the requirements of their superiors. Thus in the design of packages particular attention should be paid to information handling.

2.3 Dynamic Simulation

Simulation can be regarded as an activity harnessing a similarity between two systems in order to extend the knowledge of the less familiar system. Either the better known system or its equivalent could be employed in the process to derive the information sought after. The systems concerned could be physical or socioeconomic and the range of dynamic simulation covers

such diverse fields as the aerospace industry, chemical or nuclear reactors, and world dynamics (Pritsker 1974, Korn and Wait 1978).

In digital dynamic simulation the model is always a set of equations and these have the same general form whatever is the system under scrutiny. What differs from one system to another is the physical connotations of the variables. Distance and time are two possible independent variables. In systems theory (Smith et al. 1970, Koenig et al. 1967, Jacoby and Kowalik 1980) a dependent variable is either an across variable (propensity) or a through variable (flux). An across variable relates the conditions at one point in the region of interest to that at some other point in the region. One of these points could be a reference point or "ground". A through variable represents the flux of some quantity traversing an elemental cross-section in the region. According to Smith et al. (1970) concentration, temperature, and pressure are examples of across variables in chemical engineering and mass flux, heat flux, and flowrate are the corresponding through variables. For electrical systems voltage is an across variable and current a through variable.

Similarities between systems can be exploited then in both formulating and solving dynamic simulation problems. A unified approach can benefit all branches of science if due allowance is made for peculiarities of individual cases. Chemical engineering peculiarities are exemplified by reactions and equilibrium calculations.

It is not easy to choose between various kinds of dynamic simulation to apply to chemical engineering. Most processes are subject to some discrete event or sequential operation. Perry and Chilton (1973) point out that a series of discrete events can be treated stochastically. Brignell and Rhodes (1975) illustrate how a continuous signal could become distorted through sampling with zero hold and subsequent quantization and that the noise so introduced could be treated as a random addition to the original signal. (Measurement is a central issue in control whose importance has already been stressed.) These authors add that discrete techniques may be easier to interpret. A discrete-continuous combined simulator (Pritsker 1974, Oren 1977) can handle continuous, discrete, or random changes but is probably inefficient and may be unable to deal with large cases.

Dynamic simulation has suffered from independent development. Much repetition and waste have accrued together with confused terminologies. Some guidelines have emerged for continuous-system simulation languages (The SCI Simulation Software Committee 1967). The development of discrete-continuous combined simulators is another step in the right direction. However a package built up by additions to another may eventually become rather patchy and inefficient and a fresh start may be wise. Innovations culminating in new and powerful solution techniques may also force a restart (Elzas 1979). Foresight and anticipation are essential to integrated simulation.

However, the field is very dynamic at the present which is an impediment.

2.3.1 Some Chemical Engineering Applications

Dynamic simulation can contribute to understanding, designing and controlling chemical processes. In general simulation is performed off-line but dynamic models are also harnessed on-line to implement feedforward schemes (Perry and Chilton 1973, Daie 1980) or partial simulation (Farabi 1978, Mukesh 1980).

The potential for dynamic simulation in chemical engineering is amply illustrated by Franks (1972). His modelling principles have been adopted by other workers. For instance Harvey and Fowler (1976) have simulated a quadruple effect evaporator and so located a number of bottlenecks prior to startup, examined overall stability, and tuned the controllers. The same authors have also studied a PVC process (Fowler and Harvey 1978) and improved the basic design through simulations which have suggested novel operating procedures and enabled an unexpected increase in productivity.

Other applications include vacuum evaporation of tomato paste (Hon et al. 1979), a partial simulation of a pilot scale crystalliser (Akhtar et al. 1979), a CSTR system (Shah 1977), a butadiene plant and a chlorine plant (Bobrow et al. 1971), a digester train for the Bayer Process (Ponton et al. 1972), a fat hydrolysis unit (Johnson et al. 1975), a multi-stage liquid-liquid

extraction column and an activated sludge waste treatment column (Ingham and Dunn 1974), a phosphoric acid process and polymerisation (Overturf et al. 1978a, 1978b), thin film evaporators (Schaal and Aschner 1978), and several others in books (Luyben 1973, Shah 1976, Speckhardt and Green 1976, Pritsker 1974). However a large, complex case with the relevant data is conspicuously absent (Motard et al. 1975, Fowler and Harvey 1978). These simple cases are not treated further here.

2.4 Dynamic Modelling in Chemical Engineering

Aside from empirical and statistical methods there are deterministic methods applied to chemical plants. Two of the most widely used (Maudsley 1978) deterministic techniques are the variational approach (Lagrange's equation) and the utilisation of conservation laws. This thesis concentrates on these laws.

2.4.1 Conservation Laws

Any system has certain entities which are conserved (Maudsley 1978), for example mass, energy and momentum. Applied to a region R in the system the conservation law (or the continuity equation) for an entity E is usually expressed as

$$\begin{array}{l}\text{Accumulation of E} \\ \text{in R}\end{array} = \begin{array}{l}\text{flows of E} \\ \text{into R}\end{array} - \begin{array}{l}\text{flows of E} \\ \text{out of R}\end{array} + \begin{array}{l}\text{generation of E} \\ \text{in R}\end{array}$$

(If E is momentum, then

Time rate of change of E in i direction = sum of forces in i direction)

Partial differential equations accrue in general but some may be reduced by neglecting the distribution inside the regions which make up the system. For example a well-mixed region yields uniform composition and temperature at any time. Alternatively by means of finite-differencing a set of difference-differential equations may be obtained. The fewer partial differential equations required the better since in general these are computationally more severe.

Algebraic equations also contribute to the dynamic model as intermediate steps in the calculation of derivatives. They are exemplified by equations of state relating physical properties to pressure, temperature, and composition, and by various types of equilibrium operations such as bubble point and flash calculations.

A complete model can define exactly as many variables as the number of equations it comprises. In other words the degrees of freedom of the system must be zero in order to obtain a solution (Luyben 1973). The equations employed must be derived independently without including redundancies (Franks 1972).

Strong interaction is typical among chemical plant variables. The defining equations for such variables

constitute a loop and must be solved simultaneously as a group. All algebraic loops such as bubble point calculations must be handled before their outputs are harnessed elsewhere. On rare occasions the differential equations may be completely uncoupled as formulated originally thereby facilitating sequential integration if desired. More frequently a similarity transform may decouple them (Brewer 1974) prior to solution. However most generally decoupling is not practical and simultaneous integration is then essential.

Many books elaborate on modelling chemical plant (Franks 1972, Luyben 1973, Ramirez 1976, Smith et al. 1970, Robinson 1975). A series of papers has been initiated under the general title of 'Industrial Simulation and Modelling' in the journal Measurement and Control. So far the series has covered the general principles (Maudsley 1978), counter-current mass transfer processes (Wilson and Smith 1979), and injectors, pumps, and compressors (Martin 1979).

The author of this thesis has endeavoured to generalise as much as possible. Only models devised by Franks (1972) have been considered in depth. Modified versions of some of these have been incorporated into the current work.

2.4.2 Approximate Dynamic Modelling

A simulating engineer's task is to decide through experience what assumptions can be made in modelling a

plant and how good the results are. It is not imperative to include every phenomenon down to microscopic detail as simple models may suffice. A balance must be struck between a rigorous description and an answer that is good enough. This has been called optimum sloppiness (Luyben 1973).

Often the model developed initially may have to be simplified for on-line application. According to Edwards and Lees (1973):

> *"This may be done by rewriting the model retaining only the most important assumptions and equations. Or numerical experiments may be carried out on the original model and another model formulated which is capable of reproducing these experimental results. Alternatively for linear systems a generalised technique of model reduction may be used."*

A simplifying assumption which has been employed (Luyben 1973) in chemical engineering is ideal behaviour expressed by Dalton's and Raoult's laws for vapour and liquid phases respectively. Another approximation involves assuming equilibrium between phases in contact with each other, which is not always attained (Wilson and Smith 1979). Distribution within a region of interest may be neglected thereby producing a lumped model. Some units may be regarded as operating isothermally or adiabatically. Non-linear systems are sometimes linearised by truncating the Taylor series representation of functions after the first two terms (Maudsley 1978).

The introduction of simplifications may bring about complications in implementation and solution. For instance constant holdup for a unit cannot be guaranteed a priori. Zero holdup steady-state units may require a different solution technique as they lead to linking of local loops usually observed in dynamic simulation. A similar problem arises in applying pseudo steady-state models whereby a differential equation is reduced to an algebraic one.

Idealised staged systems can be adequately represented by an approximate transfer function of the form

$$G(s) = \frac{\exp(-T_d s)}{(1+T_1 s)(1+T_2 s)}$$

Kim and Friedly (1974) put forward methods by which the phenomenological parameters T_1, T_2 and T_d can be estimated. This kind of approximation may be very useful in on-line applications but the dead-time function in the numerator is rather awkward to implement.

2.5 Solution Tools of Dynamic Simulation

A large chemical plant can give rise to a conservation model comprising partial differential, difference-differential, and algebraic equations. Differential equations can be of initial- or boundary-value type. Algebraic equations can be linear or non-linear; they can be uncoupled, single equations or sets to be handled simultaneously. An analytical solution may be

unavailable or intractable so that a numerical scheme becomes essential.

The overall efficiency of a digital simulation program could hinge on how it implements numerical methods. Specialists working in these areas have made substantial progress and some promising algorithms have appeared "ready" to be tailored into a simulation package - a task not to be underestimated. These can be replaced at a later date as superior ones are developed.

Numerical procedures differ from analytical techniques. The former do not produce exact answers but are iterated until an error criterion is satisfied. Depending on the particular case they may take a large number of iterations to converge or may even diverge in which case it may be necessary to switch to another algorithm.

Matrix algebra plays an important part in numerical analysis. This neat and powerful approach has been harnessed in the elucidation of multi-variable problems including partial and ordinary differential equations, difference equations, algebraic equations, and optimisation. Various sources in the literature (Rosenbrock and Storey 1970, Wilkinson 1965, Fröberg 1972, Koenig et al. 1967, Jenson and Jeffreys 1977) deal with this topic. Some (Jennings 1977, Forsythe and Moler 1967) even provide programs. Applications to chemical engineering are illustrated by Amundson (1966) and Smith et al. (1970).

2.5.1 Numerical Solution of Algebraic Equations

This is another topic which has received wide coverage in previous works (Fröberg 1972, Franks 1972, Luyben 1973). Convergence of various algorithms have been investigated and some pathological cases shown. Franks (1972) has applied both Newton-Raphson and partial substitution to chemical engineering cases.

In the solution of an algebraic equation a numerical scheme is initiated by providing a guessed value for the root sought. The recursive steps below are then executed until successive values of the iterates become equal subject to a pre-specified tolerance:

$$r_n = F(x_n)$$

$$x_{n+1} = r_n$$

For instance in Newton-Raphson formulation

$$F(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$$

In direct substitution $F(x_n)$ is obtained by a rearrangement of the function $f(x) = 0$ such that only x remains on the left-hand side. A more general form is to use a gain factor G so that $x_{n+1} = r_n + G\ (r_n - x_n)$ which is recognised as partial substitution.

Such algorithms must be both powerful and robust for inclusion in a dynamic simulation package as they are accessed at many points in the program every time derivatives are required. Their behaviour can be briefly studied in individual cases and their performance can be

improved by taking special measures. For instance a better guessed value may be furnished intuitively or a memory feature may be incorporated. In cases displaying oscillatory convergence partial substitution may be imposed using a small positive gain factor (G less than 1). Oscillatory divergence may require a negative gain factor or limiting the change in the iterates. Slow, steady convergence may be forced by applying a gain factor larger than 1. There exist also promotion techniques such as Wegstein's (Franks 1972). These are based on some form of extrapolation using the information from past iterations.

2.5.2 Numerical Integration Requirements

Conservation models are almost invariably dependent on numerical integration which proceeds in a piecewise fashion unless an approximate solution is wanted at a single point. Several reasons may be advanced for piecewise integration. Firstly the plant may be subject to scheduled changes which the model must reflect. Secondly the model may contain either singularities or discontinuities which disrupt smooth integration. In addition all integration formulae are approximate and there is round-off on digital computers. The local error introduced is always propagated to the next step with some amplification. If the step size is small, such errors are manageable and may die out completely. On the other hand large steps produce unacceptable errors which may grow exponentially swamping the true solution and eventually leading to

infinity. This is known as numerical instability and is dealt with in the next section.

Each integration step involves at least once derivative estimation with associated intermediate calculations. Obviously it would be desirable to employ step-sizes commensurate with the above impositions but large enough to give reasonable computation times. Various strategies have been devised for integration but a general-purpose implementation does not exist.

2.5.3 Integration Stability and Stiff Systems

Dynamic models can be represented in matrix notation by

$$\underline{x}' = \underline{f}\ (\underline{x}) \quad \text{or} \quad \underline{x}' = A\ \underline{x}$$

where $\underline{x}$ = the state-variable vector,

$\underline{x}'$ = the derivative vector,

$\underline{f}$ = a function vector,

and A = a matrix of coefficients.

If the system is linear, then $\underline{f}$ is linear, A is constant, and the solution is characterised by the eigenvalues of A. (The eigenvalues of A are defined as the roots of the algebraic equation Det $(\lambda I - A) = 0$ where I is the identity matrix.) For non-linear systems the eigenvalues of the local Jacobian matrix J are considered (Korn and Wait 1978). J contains the partial derivatives of $\underline{f}$ with respect to $\underline{x}$.

Positive eigenvalues yield growing components in the solution and the system modelled is unstable. On the

other hand negative eigenvalues correspond to decaying components and the system is stable. The growth of error in unstable physical systems only has a relative significance and may be tolerated. Of interest however are physically stable systems and these do not allow any growth of error in the numerical scheme. This constitutes a major problem in dynamic simulation.

A system of differential equations with widely separated eigenvalues is called stiff and the ratio $\lambda_{max}/\lambda_{min}$ is the stiffness ratio (Tyreus et al. 1975). Many physical systems are potentially stiff. Seinfeld et al. (1970) quote examples from different sources including the flow of a chemically reacting gas, exothermic reaction in a tubular reactor, and process dynamics and control. Tyreus et al. (1975) cite distillation and attribute the stiffness to low relative volatility and high product purities.

Shampine and Gear (1979) distinguish two phases in the integration of potentially stiff systems: initial transient and slow decay. During the former phase the step-size is limited by accuracy rather than stability and so this part is not stiff. Stiffness arises later when the fastest decaying component has died out.

Shampine and Gear (1979) report that problem solvers have sometimes removed stiffness by altering the model:

"Quasi-static or pseudo steady-state approximations hold one set of components fixed in value over suitable time periods either

because they change so slowly that changes can be neglected or because they change so rapidly that steady-state values are achieved almost instantaneously. Such approximations lead to sets of algebraic equations coupled to (hopefully) non-stiff sets of ordinary differential equations. In some cases approaches of this kind have worked very well, but it is hard to relate the solution of the modified model to that of the original model."

The authors continue to say that current codes for stiff systems are sufficiently efficient and do much the same thing automatically and that model manipulation is extra work and is rather risky as it could lead to the wrong steady-state. Franks (1972) advocates pseudo steady-state approximations and applies them to distillation in particular.

Shampine and Gear (1979) comment that non-stiff integration methods do not "blow up" but become rather inefficient if the problem is stiff. Gear has published a code known as DIFSUB (1971) which includes two stiff options. DIFSUB is further treated in Chapter 3. New versions and extensions of DIFSUB have been reported by Shampine and Gear (1979) but are beyond this research. Some applications and comparisons are given by Enright and Hull (1976), Johnson and Barney (1976), Stutzman et al. (1976), and Hindmarsh and Byrne (1976).

2.6 Digital Dynamic Simulators

Packages are designed to diminish the detailed involvement of users with computers. For widespread

acceptance a package must be convenient and easy to use as well as powerful, versatile, and robust otherwise it may not be employed at all except by specialists. Modularity can promote flexibility and convenience (Chapter 1). A dynamic simulator may segregate integration as a first step in this direction. It may further supply basic calculation units (modules) related to the field in which it specialises thereby reducing the modelling chores that have to be performed.

Analogue simulation circuits include one integrator per (first order) differential equation. In digital simulation the same integrator may handle all the differential equations. Further, analogue simulation synchronises "derivative calculation" and "integration" but digital simulation executes these two operations sequentially which underlines the duality attached to the state-variables. The latter have to re-emerge as plant variables after each "integration" step before they are used in derivative estimations. The simulator should incorporate a mechanism to facilitate this. It is evident for instance that each chemical engineering module which includes derivatives has to handle the relevant duality of the state-variables in harmony with the other modules. This is most conveniently achieved through twice-round execution of the dynamic model each time derivatives are required. During the first round updated state-variables are converted into plant variables and local calculations are carried out within each module. The second round is reserved for estimating the derivatives and forwarding

them to the integrator. Once-through execution can be correct if the differential equations are uncoupled per module or modules are further fragmented into smaller units.

2.6.1 Existing Simulators for Chemical Engineering

Motard et al. (1975) have reported dynamic simulators addressing chemical engineering : REMUS, DYNSYS, PRODYC, DYFLO, ACME and DYSCO. Further details are available on DYFLO (Franks 1972, Goh 1978) and DYNSYS (Perry and Chilton 1973, Bobrow et al. 1971, Ponton et al. 1972, Johnson et al. 1975, Johnson and Barney 1976). Both of these simulators provide chemical engineering modules. DYFLO integrates the differential equations individually and advocates arranging of module calls "opposite to the information flow" in order to achieve simultaneity. It is known however that this flow is not unidirectional generally which precludes such an arrangement, Franks, the author of DYFLO, seems to have changed his view recently (Franks and Miller 1979) and now suggests that all derivatives should be stored and integrated simultaneously but gives no indication as to how this may be implemented. DYNSYS reportedly has two kinds, namely sequential and simultaneous. The suggestion is put forward (Perry and Chilton 1973, Bobrow et al. 1971) that the use of linear extrapolation should compensate for errors due to misplaced calculations in the sequential approach. This is not a desirable feature since it means extra work and a modified model.

DYFLO philosophy constitutes a reasonable base for a new chemical engineering simulator. Chapter 3 reviews DYFLO more thoroughly and considers the conversion of its modules to twice-round execution.

2.6.2 Continuous System Simulation Language (CSSL)

The history of digital dynamic simulators is detailed in the literature (Stephenson 1971, Chu 1969). Analogue simulation has influenced their development from the beginning. Analogue-like expressions appealed to users and became entrenched in digital simulation. The latter has now proved its worth so that it can move away from analogue simulation. Much wastage took place initially due to independent research but some guidelines have now emerged. Desirable features in digital simulation were produced and an example package was created by The SCI Simulation Software Committee (1967). The package was given the name Continuous System Simulation Language and the acronym CSSL. Stephenson (1971) supplies substantial information on CSSL.

The primary purpose of CSSL is to provide an easy, convenient, and powerful means for programming dynamic simulation cases. Two options are offered. IMPLICIT is very simple but somewhat limited. EXPLICIT is slightly more involved but much more flexible and powerful. The most important feature of CSSL is probably automatic sorting of operations into a proper computational sequence which

is attractive to non-specialists particularly. Another CSSL capability is generating expressions through macros. Further facilities include integration methods, good diagnostics, easy input/output, overrideable defaults, and various forcing functions.

A CSSL source program goes through a pre-processing phase and emerges as a FORTRAN code unless it contains a fatal error. During this translation phase a structure reflecting CSSL's philosophy is imposed on the final program. Compilation and execution may then be initiated.

Stephenson (1971) does not give any CSSL translations and so how CSSL handles state-variable duality is unknown to the author of this thesis. Dynamic modules can conceivably be inserted into CSSL through the macro facility which can generate the relevant equations prior to sorting. Any algebraic loop however causes CSSL to abort the translation and the user becomes involved. Further the translator is limited in the number of macros it can invoke and the number of variables it can handle. A large case can easily cause overloading. The translator forbids the use of array elements in sorted areas and that certainly inconveniences the user. The macro route is also inefficient and wasteful for dynamic module generation.

2.6.3 CSSL-type Simulators

Many general-purpose simulators exist today roughly conforming to CSSL guidelines : DSL (Chu 1969, Shah 1976),

CSMP (Speckhardt and Green 1976), DARE-P (Korn and Wait 1978), ACSL (Mitchell and Gauthier 1976), ICL SLAM (ICL SLAM Manual 1974), and BEDSOCS (Ord-Smith and Stephenson 1975) to name but a few. New simulators are being developed and old ones are updated including CSSL.

The aforesaid simulators translate into FORTRAN except BEDSOCS which uses BASIC. DARE-P translation is not printed but judging by the accompanying FORTRAN package in the book (Korn and Wait 1978) it is similar to DSL and CSMP. Only some features of these simulators are detailed here. Firstly, state-variables and their derivatives are all placed by the translator in one or more COMMON blocks which precludes dynamic module insertion except by invoking macros. The limitations of this route have already been covered. Secondly, integrators seem to supervise the execution which wastes substantial coding and makes it inconvenient to add a new integrator. Thirdly, the model is expected in a subprogram with a standard name which is an impediment to inter-connected simulations. This subprogram is called as frequently as the integrator dictates. The fourth point applies to DARE-P only. This simulator introduces unconventional symbols which is a needless move away from FORTRAN.

2.6.4 ICL SLAM

This is a powerful, CSSL-type simulator supplied by the International Computers Limited, ICL. The acronym SLAM

stands for Simulation Language for Analogue Modelling. The package satisfies most CSSL requirements some of which have already been mentioned. There are also some advanced features such as ARRAY INTEGRATION, RERUN, graphics, MULTIDERIVATIVE SECTIONS, chain-linking of simulations, and limited run-time interaction. The potential of ICL SLAM is obvious.

ICL SLAM has a structure radically different from the ones introduced above. The translator does not place state-variables and their derivatives in a COMMON block but in the argument list of the integrator selected by the user. An integrator has no executive power and is called like any other subprogram. It does not call a derivative subroutine but returns to the calling program where the derivatives are calculated. State-variables are transferred to their plant variable names before they are used. Derivatives are stored and communicated simultaneously to the integrator. The translator counts the state-variables and creates the necessary work arrays.

ICL SLAM structure can easily accommodate dynamic modules tailored to twice-round execution. The translator can be useful but is not essential. Modification at FORTRAN level is possible after the pre-processing phase. State-variables, derivatives, and the corresponding plant variables can all be put in different COMMON blocks without a clash.

2.6.5 A Discrete-Continuous Combined Simulator : GASP

Scheduling is important in chemical engineering. GASP can be beneficial in this respect since it is designed to handle events. Pritsker (1974) gives a broad definition of an event : An EVENT occurs at any point in time beyond which the status of system cannot be projected with certainty. Decisions to be taken at EVENT times must be pre-programmed. Some EVENTS can only be defined implicitly by conditions involving one or more plant variables. These are called STATE-EVENTS as opposed to TIME-EVENTS which occur at definite points in time. All EVENTS must be detected by the simulator adjusting the step-size if necessary. The testing of the said conditions and the disruption of smooth integration can decrease the program efficiency substantially. There is also a possibility of missing a state-condition.

The GASP executive is closely linked to an integrator which makes it difficult to insert user-supplied integrators. If combined simulation is selected, then GASP expects separate subprograms for the initial conditions (INTLC), derivatives (STATE), state-conditions (SCOND), and decisions (EVNTS). This segregation is needless and can prevent the use of BLOCK DATA facility of FORTRAN. SUBROUTINE STATE must be coded such that the state-variables and derivatives are returned in a COMMON block. GASP offers no help in handling state-variable duality and sorting.

A translator has recently been added to GASP (Pritsker and Pegden 1979) and the new version is called

SLAM (!). This however should not cause any confusion in this thesis. Past this point any reference to SLAM implies ICL SLAM.

2.7 Summary and Conclusions

A picture of chemical engineering has been built identifying the objectives and components of plant operation and pointing out the need for an integrated approach from the beginning. The useful role of computers has been detailed both as part of a plant and as simulation aids.

Most chemical processes are transient in operation and scheduling is a frequent necessity. A dynamic simulator with event handling can conveniently illuminate many aspects of transient behaviour. Current dynamic simulators have been critically reviewed paying particular attention to modularity, flexibility, versatility and ease of use. It seems that ICL SLAM structure is very attractive and can easily accommodate modules tailored to twice-round execution which is a convenient route to achieve general simultaneity. DYFLO modules are based on the conservation laws and deserve further investigation for inclusion in the new digital dynamic simulator. Although potentially useful GASP will not be amalgamated during this work. The author appreciates however the importance of a unified approach to simulation. The system developed will be file-oriented and will allow for future expansion.

Numerical difficulties in dynamic simulation have been covered. The most crucial element appears to be

integration stability. Stiffness is probably a common feature in chemical engineering problems. A widely publicised integrator DIFSUB will be studied with the aim of adoption so that the simulator can handle stiff cases.

CHAPTER 3 TOWARDS A NEW DYNAMIC SIMULATOR

3.0 Introduction

The literature survey has led to the notion that efforts should be concentrated on combining the structure of ICL SLAM (ICL SLAM Manual 1974) with DYFLO's modules (Franks 1972) tailored to twice-round execution. This chapter is devoted to a critical study of DYFLO and of DIFSUB (Gear 1971). Suitable modifications are suggested for incorporating these into the new simulator.

3.1 A Review of DYFLO

3.1.0 Introduction

As a simulator oriented towards chemical engineering DYFLO was a pioneering creation. It applies cause-and-effect reasoning to the derivation of conservation models and uses equation blocks as a device to present the solution strategy. It takes computation one stage further by providing a suite of illustrative modules to operate within its framework. It is adequately equipped to be harnessed as a general-purpose simulator.

Special storage in a labelled COMMON block is allocated to stream data and physical properties. This block also contains reaction information in the form of flow terms. Stream indices, holdups, and heat loads are communicated through argument lists.

A major flaw in DYFLO is insufficient supervision. Co-ordination among modules is almost non-existent. Much housework is left to the user which means needless

inconvenience and extra risk of logic corruption. The absence of graphics is another drawback.

DYFLO integrates state-variables sequentially. This should not cause any problems if 'derivative estimation' and 'integration' phases are performed in separate sections. DYFLO modules contravene this requirement. They are accessed in the DERIVATIVE SECTION and yet they integrate the local state-variables thereby giving rise to a simultaneity problem. Franks advice is to arrange the orientation of the modules opposite to the information flow in the plant. But chemical plants generally present more than one direction in which information flows and so any order may be strictly unsatisfactory.

Ideal behaviour is usual for DYFLO models. Ideal mixing rule, Raoult's Law and Dalton's Law are assumed. The saturation vapour pressure PN for a component is given by the Antoine equation:

$$PN = \exp\ (A1 + A2/(A3 + T))$$

where T represents the temperature and A1, A2 and A3 are constants. Further the heat capacity C_p and enthalpy H for a component are defined by

$$C_p = a + bT$$

and

$$H = c + aT + \frac{b}{2} T^2$$

where a, b, and c are constants. c depends on the reference temperature and, for vapours, on the latent heat. These assumptions and relations are incorporated into the new simulator.

3.1.1 Cause-and-effect

A unique solution is possible only if the model comprises as many non-redundant equations as there are unknowns so that each equation can yield one variable. It is however difficult to find the best pairing arrangement when this condition is satisfied. Franks favours the application of cause-and-effect. Physical systems are usually stable and so models closely imitating them should be numerically stable. This arrangement also provides insight into the true mechanism of the system.

Franks observes that natural processes can only accumulate, that is integrate, but not differentiate and that fluxes are caused by differentials. Thus if applied properly, cause-and-effect should result in an arrangement wherein differences in holdup quantities would yield the fluxes which define the differential equations and subsequent integration would update the holdup quantities.

Natural processes integrate simultaneously. Franks does not explain why he chose sequential integration for DYFLO but one conceivable explanation is to save storage space and computer time. Experience has shown during this research that extra storage is needed in simultaneous integration not only for state-variables but also some intermediate quantities.

3.1.2 Information-flow Diagrams

DYFLO is accompanied by a scheme to depict diagrammatically the solution strategy applied to the system.

Every equation is placed in a block whose output is the variable it solves for. This is conveyed to wherever it is needed as input. Integration blocks are indicated by a solid triangle on the output line(s).

This scheme is rather confusing since it leads to a criss-cross of arrows creating apparent information loops that do not exist in the plant. But the general idea is appealing. Few new modules have been created in the course of this work and the need has not arisen to use such a presentation. However, the author believes that it is potentially very useful and suggests some modifications. In the new arrangement algebraic equations will be sequenced as they are solved; the quantity on the left-hand side will be understood to be the output of that equation. Algebraic loops will be placed in blocks or suitably linked together marking the guessed quantity optionally. All differential equations will follow algebraic equations; their outputs will be made available by the simulator before they are employed anywhere. The modified arrangement is totally compatible with ICL SLAM where each algebraic loop is handled in a NOSORT block whose argument list indicates the relevant inputs and outputs; the whole block is then treated like a single equation.

3.1.3 DYFLO : Integrators

DYFLO facilitates integration by means of two

subprograms. SUBROUTINE INTI advances the independent variable and SUBROUTINE INT updates the state-variable in its argument list. Three fixed-step algorithms are implemented, namely Euler, Trapezoidal or Modified Euler, and Fourth-Order Runge-Kutta. These complete a step in 1, 2, and 4 stages respectively, each stage requiring fresh derivatives. Franks has manipulated the trapezoidal formula in order to eliminate the need to store the state-variable at the beginning of a step. This manipulation precludes arbitrary changes to be applied to the corresponding plant variable within the model during the second stage - a point that Franks has overlooked.

3.1.4 DYFLO : Equilibrium Subprograms

These include (i) EQUIL for bubble point estimation at a given pressure, (ii) DEWPT for dew-point calculation at a given pressure, (iii) FLASH for various flash operations at a given pressure, and (iv) PCON for a partial condenser at a given temperature. All of these routines rely on Newton-Raphson technique.

Two mistakes are observed in the formulation of the said algorithm FLASH but these do not appear in the code. The latter however is found to suffer from a weakness. The convergence becomes increasingly suspect as the vapour fraction R diminishes. R is initialised by the user to start the iterations.

PCON is troubled by the existence of a trivial root R = 1. A positive derivative forces the iterations to this value instead of the desired root. DYFLO attempts to cure this failure by changing the sign of the derivative, or pushing the iteration uphill so to speak.

3.1.5 Enthalpy Calculations in DYFLO

Using $H = C + AT + B\,T^2$ (c.f. Section 3.1.0) SUBROUTINES ENTHL and ENTHV calculate the enthalpy of a liquid and vapour stream respectively given the composition and temperature. On the other hand SUBROUTINE TEMP finds the temperature of a stream when its composition and enthalpy are known.

3.1.6 Dynamic Modules of DYFLO

A dynamic module represents an elemental volume. The latter need not be a complete item on the plant. It may be a region in one item or a few items put together. It will help however if a one-to-one correspondence is observed which not only makes the program read like the flowsheet but also facilitates easy transfer from one kind of simulation to the next.

The complexity of a module varies considerably. In the simplest case only one or two differential equations are involved. Some modules further include intermediate algebraic equations, for example bubble point calculations. Moreover reactions may occur with external heating or cooling. A general package is limited in the number of modules it can provide but built-in interfaces make expansion possible.

For an elemental volume where complete mixing is applicable the natural arrangement would be to calculate all the relevant flows, take a total mass balance, a mass balance for each component, an energy balance, and then integrate the differential system to update respectively elemental holdup, composition, and energy. The algebraic equation which states that mole fractions must add up to unity can be utilised for one component thereby reducing the total number of differential equations. The literature shows unnecessary controversy over this issue. For instance, both Luyben (1973) and Ramirez (1976) insist that the algebraic equation must be employed. DYFLO depicts this scheme in the information flow diagram for distillation but the codes incorporate the differential alternative. This author believes

that if mole fractions add up to unity initially, then either approach is correct. The differential scheme seems more convenient to code however.

The most basic module of DYFLO is SUBROUTINE HLDP which models a single-phase holdup with a single-input and single-output. The use of this module is well illustrated in a few cases. A counter-current liquid-liquid extractor and a counter-current shell-and-tube heat-exchanger have also been coded, called STGR and DHE respectively, but these have not been verified. SUBROUTINE DHE is sectionalised and calls HLDP for each section. This author does not expect DHE to work since it violates simultaneity within the item. These modules will not be considered any further at this stage. Distillation modules are thought however to be worth elaboration.

3.1.7 DYFLO Modules for Distillation

SUBROUTINES VVBOIL, STAGE, STGF, STGS, BOT, and REB of DYFLO model respectively, a variable-volume boiler, an ordinary distillation stage, a feed stage, a sidestream stage, a bottom stage, and a reboiler. STGH is an alternative to STAGE. These subprograms are dwelt upon here in order to illustrate DYFLO's philosophy.

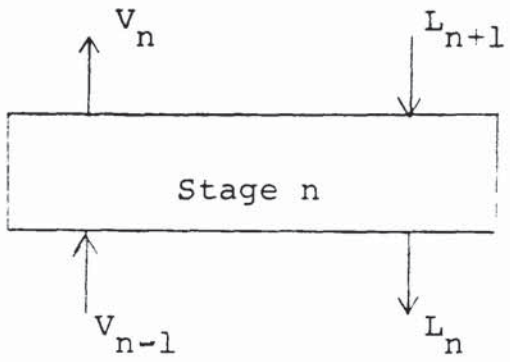


Figure 3.1 An Ordinary Stage in Distillation.

Figure 3.1 depicts an ordinary distillation stage. The more important assumptions are (i) that equilibrium exists between the vapour and liquid streams, V_n and L_n respectively, leaving the nth stage, (ii) that each phase is completely mixed so that bulk properties are applicable, (iii) that the vapour phase holdup is negligible compared with that of the liquid phase, and (iv) that all energy forms other than heat are negligible.

Based on the third assumption DYFLO applies the balance equations to the liquid phase only. Thus the total mass balance gives

$$\frac{d}{dt}(HL) = V_{n-1} + L_{n+1} - V_n - L_n \qquad (3.1)$$

where HL is the stage (liquid) holdup and t is time. The reaction term is omitted for brevity. Similarly the mass balance for the Nth component is

$$\frac{d}{dt}(HL.X_{n,N}) = V_{n-1}\,Y_{n-1,N} + L_{n+1}\,X_{n+1,N} - V_n Y_{n,N} - L_n\,X_{n,N} \qquad (3.2)$$

where X and Y are liquid and vapour mole fractions. Further the energy balance is

$$\frac{d}{dt}\ (HL.H_{n,L}) = V_{n-1}\ H_{n-1,V} + L_{n+1}\ H_{n+1,L}$$

$$- V_n\ H_{n,V} - L_n\ H_{n,L} + Q \qquad (3.3)$$

where H is the molar enthalpy, Q the external heat load, and subscripts L and V indicate the phase.

Now DYFLO regards HL as constant and yet calculates L_n using

$$\frac{d}{dt}\ (L_n) = \frac{1}{HTC}\ \frac{d}{dt}\ (HL) \qquad (3.4)$$

where HTC is the stage hydraulic time constant, which amounts to assuming a linear relationship between L_n and HL. This is a gross inconsistency.

Briefly, the solution strategy built into STAGE is (i) to integrate Equation (3.4) and a form of Equation (3.2) to update L_n and the liquid phase composition respectively, (ii) to perform a bubble point calculation (at fixed pressure) and determine the stage temperature as well as the vapour composition, and (iii) to work out the backward derivative for $H_{n,L}$ and insert it into a form of Equation (3.3) to obtain V_n. The manner in which L_n is estimated has already been criticised in the preceding paragraph. There is also a point to be made regarding the use of the backward derivative. This causes the time lag involved in the integration process to vanish and V_n becomes instantaneous, like other steady-state approximations, making it necessary to determine simultaneously all the vapour flows up the column which constitutes an impediment

to stagewise modularisation. Franks ignores this point completely. By comparison Ramirez (1976) and Luyben (1973) both apply steady-state energy balances and accordingly handle the simultaneity. Their codes however do not include stage modules. Franks has also included a pseudo-steady-state approximation in the code as a measure to cope with stiffness due to a component balance equation but this will not be considered here.

The alternative subprogram STGH discards backward differentiation. Instead it integrates a modified form of equation (3.3) to update $H_{n,L}$. Then it harnesses empirically the difference between $H_{n,L}$ and its counterpart obtained from the liquid composition at the bubble point temperature in order to bring V_n up to date. This author believes that having so determined V_n these two enthalpies should be matched by relaxing the stage pressure, but Franks does not attempt this.

A linear relationship between L_n and HL is supported in the literature (Luyben 1973, Robinson 1975) but a similar device for V_n is absent. It is questionable whether equation (3.3) alone should be assigned to the determination of V_n. The advantages of ignoring the vapour phase holdup are also vague. If and when an equation for V_n becomes available, then a new strategy may be proposed. This author suggests that the phases should be considered as a whole. Equations (3.1, 3.2, and 3.3) can be integrated to find the overall holdup, composition, and enthalpy. The distribution of all quantities between the

phases can be obtained from a flash calculation for example. The estimation of L_n and V_n completes the step.

SUBROUTINES STGF and STGS include a feed stream and a sidestream respectively. BOT represents a bottom stage module with externally manipulated L_n. There is no vapour stream entering and a suitable form of equation (3.1) is integrated for variable holdup. VVBOIL has one input stream and one output stream the latter being in equilibrium with the holdup content.

Certain observations are worthy of mention here. A clash occurs in DYFLO when the external heat load Q depends on the stage temperature. Q is required by the module before its temperature is updated within. Franks overlooks this. Secondly DYFLO automatically includes reaction terms in the balance equations which forces the user to resort to clearance before calling such a module if it represents a user item without reactions. A steering flag can avoid this inconvenience. Finally DYFLO may reset variables without any warning. For instance a flow can be corrected to zero if it becomes negative during the simulation. Such arbitrary actions may hide a fundamental modelling error and are not recommendable.

3.1.8 DYFLO : Miscellaneous

Included in the library is a steady-state routine CSHE for a counter-current shell-and-tube heat-exchanger. It incorporates an iterative scheme to find the outlet temperatures. This model is tested by Franks but the

reported results fail the heat transfer equations. This failure has also been observed by Goh (1978) who has obtained the correct solution without any alterations. The discrepancy may be a printing error. A non-iterative scheme exists using "effectiveness" for heat-exchanger calculations (Crowe et al. 1969).

SUBROUTINES CONTR1, CONTR2, and CONTR3 implement conventional P, PI, and PID controllers respectively. CONTR2 and CONTR3 require the user to make calls to the integrator INT. Franks realises that these calls cannot be made within for simultaneity reasons. This arrangement inconveniences the user.

DYFLO offers a first-order (TFN1) and a second-order (TFN2) transfer function. Other subprograms worth mentioning are CPS (partial substitution), CONV (Wegstein's algorithm), FUN1 (linear table-lookup), FUN2 (second-order table-lookup), DER (backward differentiation), TDLY (dead-time), LDL (line-delay), CVBOIL (constant-volume boiler), HTX (heat-exchanger), SUM (summer), SPLIT (splitter), and VALVE.

The lessons learnt from DYFLO are summarised at the end of this chapter.

3.2 A Description of DIFSUB

Gear's book gives three FORTRAN codes, namely a variable-step Runge-Kutta method, an extrapolation method, and DIFSUB. The latter is a combination of three methods

accessible individually. These are classified as $P(EC)^M$ type where P is the prediction step, E an evaluation of the derivatives based on the latest values of the state-variables $\underline{x}$, C a correction step, and M the number of iterations. The prediction is a simple polynomial extrapolation on previous values. The purpose of the corrector is to provide stability; it does not generally improve accuracy (Shampine and Gear 1979). The corrector is implicit since it needs the value that it is supposed to estimate and that is why it is coupled to a predictor. Each iteration amounts to solving a set of algebraic equations.

One option in DIFSUB is nonstiff employing variable-step Adams-Moulton formulae of order up to 7. The other two options are designed to handle stiff cases. They use backward derivative formulae of orders up to 6 and rely on the local Jacobian as they implement a Newton-Raphson variant to solve the aforesaid algebraic set. In one choice DIFSUB obtains an analytical Jacobian from a user-supplied routine and in the other estimates numerically an approximate Jacobian. In practice the Jacobian is assumed constant and updated infrequently (Enright 1978). Shampine and Gear (1979) illustrate the use of the Jacobian and claim that about 50% accuracy in its calculation will cause the iteration to converge.

DIFSUB automatically advances the independent variable through a given range subject to (i) a minimum and a maximum step-size, (ii) a maximum order of the method selected, and (iii) an error criterion. It has a built-in

strategy to choose a suitable order and step-size and to act upon failures. The derivatives are expected in a SUBROUTINE DIFFUN and the analytical Jacobian in SUBROUTINE PEDERV. Both are called by DIFSUB as often as it deems necessary. If the case proves too formidable, then integration is abandoned and a flag is set appropriately before returning to the calling program.

The basic strategy in the non-stiff option allows a maximum of three corrector iterations. If convergence is not achieved, the step-size is reduced and a return is made to the prediction step. If it is already the minimum specified however, then a no convergence exit is taken. On convergence the error is checked. If it is unacceptable, a decision part is entered where the possibility of changing the order and/or the step-size is considered. However, if the number of multiple failures exceeds 3, then DIFSUB aborts without referring to the decision part. The execution is directed to the prediction step after the decision part. The latter is also entered if the error is found satisfactory and a preset number of steps have been completed. On exit this time the variables are scaled and a new step initiated unless the finish point has been reached in which case a return is made to the calling program.

The stiff options employ the same strategy. They estimate the Jacobian initially and update it either when the order is changed or the corrector fails to converge. The Jacobian estimation always succeeds the prediction step.

3.3 The Conversion of DIFSUB

In ICL SLAM structure an integrator has no executive power but is called like any other routine to perform a certain function. In this new environment DIFSUB has to mark the stage it reaches, return to the executive, obtain fresh derivatives, and carry on where it has left off as opposed to calling DIFFUN. New derivatives obtained at the beginning of a step are used which amounts to a $P(EC)^{M}E$ implementation except for the final step ending the simulation. This may improve the stability (Gear 1971).

This author believes that DIFSUB is needlessly complex. Indeed it has proved difficult to scrutinise it and make firm conclusions. Some alterations have been introduced into the code over a long period in the light of experimental studies. First, the two-dimensional SAVE array has had to be fragmented in order to eliminate machine-specific access. Secondly, the multiplication by the Pascal triangle, which takes place in the prediction step, has been recoded. Thirdly, the Jacobian is now calculated before prediction and not after. This should circumvent uncertainties and corruption in the Jacobian. Fourthly, the numerical estimation of this matrix has been made independent of the error parameter. The stiff options of DIFSUB employ an inversion routine (MATINV) to solve the implicit equations. For efficiency reasons this was replaced by LU factorisation followed by forward and backward substitutions.

DIFSUB has been found rather insensitive to stiffness. It often becomes overconfident which causes aborted runs. Step-size increases have had to be restrained severely. Several new parameters have been incorporated in order to be able to influence the decision part externally. Thus the bias on different orders can be altered through run-time data. Similarly the maximum number of corrector iterations can be increased beyond 3 or the step-size can be further limited. In the new version the decision part is avoided for 10 steps if any trouble is experienced. A change in order does not automatically require a Jacobian update. In fact this update is delayed until all else has failed and the minimum step-size has been tried.

3.4 Summary and Conclusions

A chemical plant is governed by material, energy, and signal flows, and by local holdups. Physical flow loops do not necessarily give rise to computational loops. The latter seem to infest steady-state simulation. In contrast dynamic conservation models generally display local loops. Steady-state approximations and the use of backward derivatives may lead to the linking of these loops however. Further simultaneity problems stem from what can be termed "sequential modularisation" in DYFLO. These are only artificial and twice-round execution will probably cause them to disappear.

A substantial portion of DYFLO philosophy has been adopted including cause-and-effect and information-flow diagrams. Some reservations have been expressed in this chapter and alterations suggested. Steady-state models are allowed but pseudo steady-state approximations of DYFLO have been excluded. Arbitrary actions are ruled out.

DYFLO offers little supervision leaving much housework to the user. The logic is open to corruption. Behind-the-scenes performance can minimise this. DYFLO is not file-oriented and does not echo any inputs. Generalisation is limited and no attempt is made to provide defaults. Graphics are also lacking.

Convergence has received some attention in this work. Models have been improved by a judicial choice of a gain factor and by including a memory. A novel promotion technique has been devised (Appendix 1).

DIFSUB has been studied and amalgamated into the new simulator. It is a variable-step supplement to DYFLO's integrators and can handle stiff cases.

CHAPTER 4 ACES : THE NEW DYNAMIC SIMULATOR

4.0 Introduction

This research has culminated in a modular, CSSL-type digital simulator supplemented by chemical plant subprograms. It has been named Aston Chemical Engineering Simulator or ACES in short. After the introduction of some basic definitions this chapter moves on to describe ACES and how it may be used. A listing of the ACES library is given by Kocak (1980a).

4.1 A Dynamic Simulation Methodology

This section is based on ICL SLAM (ICL SLAM Manual 1974) and GASP (Pritsker 1974) both of which provide elaborate and consistent terminologies. Extensions are included here and in Section 4.2.

The system under investigation is called the SIMULAND - normally a chemical plant in this research. The independent variable is usually time but can be distance in some cases. Each continuous sweep from some initial time to a finish time constitutes a RUN. The objects within the boundaries of the SIMULAND are called ENTITIES and are characterised by their ATTRIBUTES. The status of the SIMULAND at a given time is referred to as its STATE and must be uniquely determined by the model. Simulation is then the dynamic portrayal of the STATES of the SIMULAND.

A complete investigation may comprise several RUNS. Some quantities display the same values throughout the

investigation and are identified as CONSTANTS. Some quantities change from RUN to RUN only and are called PARAMETERS. VARIABLES are those quantities which may take on different values during the same RUN. STATE-VARIABLES are those which are integrated. Together with the CONSTANTS and PARAMETERS they sweep, that is completely determine, the STATE at any instant. STATE-VARIABLES change continuously.

An EVENT occurs at any point in time beyond which the status of the SIMULAND cannot be projected with certainty. EVENTS can cause changes in the status of the SIMULAND or in the model definition. They mark decision points where the decision may be not to impose any alterations. Those that occur at a specified projected point in time are termed TIME-EVENTS. Those that take place when the SIMULAND reaches a particular STATE are called STATE-EVENTS. Unlike TIME-EVENTS they are not scheduled in the future but are "detected" by the fulfilment of prescribed conditions.

Individual simulations may be programmed as separate SEGMENTS and chained together through argument lists or COMMON blocks. A SEGMENT normally comprises an INITIAL, a DYNAMIC, and a TERMINAL REGION which are executed in that order. The DYNAMIC REGION may be subdivided into pairs of DERIVATIVE and PARALLEL SECTIONS. The TERMINAL REGION may optionally link back to the INITIAL REGION for a RERUN. This feature may be harnessed to try various PARAMETERS or to solve boundary-value problems.

A complete simulation program may be said to have two SIDES; one written by the user and one contributed by the simulator. Most CSSL-type simulators impose their SIDE on the source program in a translation phase. Some changes may take place during the process. Macros may be invoked, defaults and housekeeping statements inserted, and automatic sorting provided. Some pseudo-operations disappear. For instance ICL SLAM may add (i) a group of statements under SEGMENT INITIALISATION, (ii) a PRE-INITIAL REGION, and (iii) an overrideable set of integration parameters in an INTINF BLOCK.

A CSSL-type simulator which provides a translation phase should accept a source program with an IMPLICIT or EXPLICIT structure. The IMPLICIT option has limited capabilities but is much simpler and does not require demarcation at all. The EXPLICIT option, on the other hand, is much more powerful but requires the user to mark with keywords the REGIONS and the SECTIONS.

4.2 The SIDES of an ACES Program

ACES amalgamates the two SIDES neatly and permanently as a FORTRAN program without involving a translation phase. Control is shuttled between the SIDES systematically until the job is finished. The USER SIDE starts the simulation through a call to the ACES executive, SUBROUTINE HSKPEX. The latter initialises the SEGMENT by setting defaults and limits. Subsequently the SIDES co-initialise the problem in the INITIAL REGION. A twice-round scheme is

implemented in the DYNAMIC REGION to portray the STATES of the SIMULAND with general simultaneity. Fresh derivatives are always obtained in two ROUNDS as follows. The updated STATE-VARIABLES are retrieved onto the USER SIDE during the FIRST ROUND; they become PLANT VARIABLES once again. In the SECOND ROUND the derivatives are estimated and forwarded to the SIMULATOR SIDE for integration. The arrival from the INITIAL REGION needs special treatment; integration is skipped over and no retrievals take place. Output is produced in the DYNAMIC REGION at regular COMMUNICATION INTERVALS. The TERMINAL REGION handles, quite appropriately, terminal matters and initiates a RERUN if desired. When there are no additional RUNS to supervise HSKPEX returns control to the calling program on the USER SIDE which may choose to stop or call HSKPEX again to start another simulation.

4.2.1 The ACES Executive

SUBROUTINE HSKPEX (Figure 4.1) occupies the top hierarchical position on the SIMULATOR SIDE. Its function is to supervise and co-ordinate the simulation from the beginning to the end. It calls on various housekeeping subprograms to perform the detailed chores. Normally HSKPSI initialises the SEGMENT by setting defaults and limits. HSKPIN is the INITIAL REGION housekeeper. UGINT is the general integrator interface, that is a route to access a chosen integrator. ACES design currently excludes

```
      SUBROUTINE HSKPEX(USMLND)
C-----              THE EXECUTIVE INTERFACE                          --------
C     EXTERNAL SUBPROGRAM USMLND HOUSES THE MODEL
      COMMON/YBNT1/KUL,NF,NFC,NXD,ITAM,IPRT,JPS,IDOF,JEEP,ISTP,TL,DT,
     1             NIN,JDM,KAT,ILK,MSY(100),SY(100),DJ(200)
      COMMON/YBNIZ/XX(50),XD(50),XL(50)
      COMMON/YBNIb/IDUG,I,MN,MX,IM,PME,T,E,ME,
     1             TFIN,USET,PROJ,DATE,BAGSIZ,MDR
      COMMON/SLAM01/KEEP,I001,I002,IC0(10)
      LOGICAL IDOF
      EXTERNAL USMLND
C*********  S E G M E N T              I N I T I A L I Z A T I O N  ****
      CALL HSKPSI
 5000 CONTINUE
C                     ENTIRE                          BLOCK
C*********  P R E - I N I T I A L                      R E G I O N  ****
C*********  I N I T I A L                              R E G I O N  ****
      CALL HSKPIN(USMLND)
C*********  D Y N A M I C                              R E G I O N  ****
 6000 CONTINUE
C                     DERIVATIVE                    SECTION
      IF(IDOF)GOTO 6200
 6100 CONTINUE
      JPS=-1
      ILK=JPS
      CALL UGINT
 6200 CONTINUE
      CALL HSKPDE(USMLND)
      IF(ITFM.NE.0)GOTO 7000
      IF(.DOG.NE.1)GOTO 6100
C                     PARALLEL                      SECTION
      CALL HSKPPA(USMLND)
      IF(T.GT.TFIN)GOTO 7000
      IF(ITFM.NE.0)GOTO 7000
      GOTO 6000
C*********  T E R M I N A L                            R E G I O N  ****
 7000 CONTINUE
      CALL HSKPTE(USMLND)
      IF(IAPT.EQ.0)RETURN
      IF(IAPT.GT.0)GOTO 5000
C     REPEAT WITH RESET REQUIRED
      KTEP=0
      DO 7100 K=1,JDV
 7100   XX(K)=XI(K)
      CALL HSKPDE(USMLND)
      GOTO 5000
      END
```

Figure 4.1 The ACES Executive SUBROUTINE HSKPEX.

MULTIDERIVATIVE SECTIONS and includes only one DERIVATIVE and one PARALLEL SECTION. These employ HSKPDE and HSKPPA as housekeepers. HSKPTE attends to the chores in the TERMINAL REGION.

The main interface with the USER SIDE is an EXTERNAL SUBROUTINE and has a default name of USMLND. Supplied by the user, this routine embodies the complete model, calling subprograms on either SIDE if necessary. USMLND is called by HSKPIN, HSKPDE, HSKPPA, and HSKPTE with an appropriate flag.

As can be seen, after a call to HSKPSI to initialise the SEGMENT, HSKPEX executes the INITIAL, DYNAMIC, and TERMINAL REGIONS in that order. This constitutes a RUN. Simulation can be diverted to the TERMINAL REGION in two ways. The independent variable T may exceed the finish point TFIN; this condition is tested in the PARALLEL SECTION after the call to HSKPPA. Alternatively a condition formulated in USMLND may be satisfied in which case ITRM is set to a nonzero value. A test is made on this after the calls to HSKPDE and HSKPPA.

Different actions are taken after the HSKPTE call in the TERMINAL REGION according to the value of IRPT. If it is 0, the simulation is finished and the control is returned to the calling program. A negative value signals a user request to return to the initial conditions, that is a RESET. The STATE-VARIABLES are first set to their initial values and then converted to PLANT VARIABLES as usual through HSKPDE before re-entering the INITIAL REGION. A positive IRPT leads to a RERUN without RESET.

Thus three modes have been identified namely SIMULATION, INTEGRATION and RESET.

Each time that the DYNAMIC REGION is traversed constitutes a PASS. IOO5=.TRUE. indicates the FIRST PASS when ACES skips integration. No retrievals are needed during this PASS since all variables are already on the USER SIDE. LATER PASSES are flagged by IOO5=.FALSE. More details on IOO5 are offered in Appendix 2 which includes a major portion of the ACES variables. Some of these variables are observed in HSKPEX. T, TFIN, ITRM, and IRPT have been covered already. JPS=-1 marks the INTEGRATION MODE, ILK=JPS=-1 the UGINT interface (thus keeping track of where the USER SIDE was last contacted), and KEEP=O an intermediate step.

4.2.2 A Brief Study of the ACES Library

ACES provides several dozens of subprograms as listed in Appendix 3. The housekeepers implement and maintain the simulation and together with the integrator suite they form the core of the package. Included also are clearance routines, readers, and display routines which boost convenience. Then there are subprograms to assist in the modelling of chemical plants. A cascade controller, a general-purpose controller, and a general-purpose sensor have been incorporated. A subprogram which at least partially represents a plant item, other than those just mentioned, is termed a MODULE or a SEMI-MODULE depending on whether it requires integration or not. A subprogram is a PROCESS AUXILIARY if it cannot

represent an item either wholly or partially and yet performs a related function such as enthalpy estimation. Purely algebraic subprograms are ALGEBRAIC AUXILIARIES.

Constant dead-time rather resembles integration in that it occurs in two stages, namely storage and retrieval after a number of steps. Unlike MODULES dead-time is implemented as a pair of subprograms. XDTR handles retrieval and XDTF forwarding.

Various interfaces are needed between the SIDES for flexibility and generality. ACES provides defaults for such interfaces aiming to proceed with simulation as far as possible.

4.2.3 The USER SIDE

The call to HSKPEX names the EXTERNAL SUBROUTINE which houses the model definition. This is declared to be USMLND in a default MASTER (Kocak 1980a) included in the ACES library. USMLND (see, for example, Figure 4.2) is divided into four PARTS, namely UIN, UDE, UPA, and UTE as indicated by comment records in the listing. These are for initialisation, derivative estimation, output, and terminal matters respectively. Integration takes place on the SIMULATOR SIDE. The relative orientation of the PARTS is irrelevant provided they are executed according to the value of JPS as adjusted by ACES (Appendix 2). By convention JPS is set to 0 for UIN, 1 or 2 for UDE, 3 for UPA and 4 for UTE.

```
      SUBROUTINE USMLND
      COMMON/YEN01/KUL,MF,NFC,MXD,ITER,IPT,JPS,JOB,JBEP,ASTP,TL,DT,
     1             MIN,JDY,KAT,ICK,HSY(100),SY(100),DJ(200)
      COMMON/YEN02/XX(50),XD(50),XL(50)
      COMMON/YEN03/SN(50,10),YDS(30,2),CN(20,20),PCC(20,4)
      COMMON/YEN04/YY(50,15),PP(50,15),TT(50),ZATP,ZATG
      COMMON/YEN05/D(10,15),C(50,10),S(50,15),YS(50,15)
      COMMON/YEN06/TOD,E,AN,MX,IA,PI,T,E,ME,
     1            TFIN,USER,PROJ,DATE,BACSIZ,NUP
      COMMON/YEN07/H,HN,HK,JGN,JSTART,FU,FS,FD,FX,XMAX,
     1             YBS,FT,FZ,NAT,NP,NC,NJ,NS
      COMMON/YEN08/IPEN(50),NGN(7),YMAX(50),C1(50),SAVE(3,50),SD(7,50)
      COMMON/SLAVE1/KEEP,I001,I002,I00(10)
      EQUIVALENCE (BOS,SY(77)),(CIZ,SY(78))
      LOGICAL I00*4
      IF(JPS.NE.0)GOTO 999
C-------------    U               I               N
      READ(1001,100)A1,A2,Y1,Y2
      C-LL TVSP(0,EAGSIZ,PHY1       ,PHY2       ,BOS,BOS,3)
  100 FORMAT(8BL.0)
      RETURN
  999 GOTO(1000,1100,1600,2100),JPS
C-----------    U               D               E
 1000 CONTINUE
  900 IF(JPS.EQ.2)GOTO 950
C                       FIRST ROUND
      IF(I005)GOTO 990
      Y1=ATRN(DY1)
      Y2=ATRN(DY2)
      GOTO 990
C                       SECOND  ROUND
  950 CALL F W D(Y1,A1*Y2)
      CALL F W D(Y2,A2*Y1)
  990 IF(IOTP.GT.KUL)RETURN
C                   DEBUG SECTION
      RETURN
C-------------    U               P               A
 1500 CONTINUE
      CALL TVS.(3,T,Y1,Y2,000,BBE,1)
      RETURN
C-------------    U               T               T
 2100 CONTINUE
      RETURN
      END
```

Figure 4.2 An Example of SUBROUTINE USMLND.

UDE is called twice per PASS in accordance with the twice-round principle. The FIRST ROUND is flagged by JPS=1 and UDE retrieves the STATE-VARIABLES onto the USER SIDE by repeated calls to FUNCTION RTRV. The FIRST PASS however is an exception and RTRV calls are bypassed. The logical variable IOO5 is .TRUE. during this PASS only to distinguish it from the rest. The SECOND ROUND is indicated by JPS=2 and UDE calculates the derivatives and forwards them to the ACES SIDE for integration by means of FRWRD calls. The latter must match RTRV calls in order. All PLANT VARIABLES must have a unique and current value before they are referenced. A PLANT VARIABLE which has never been assigned a value is UNDEFINED in ACES terminology. A variable is UNDERDEFINED when its value is out of date.

UDE needs further structuring (Figure 4.3) if chemical plant routines are to be employed. A CONTROL BLOCK is placed at the top of UDE and is followed by a PROCESS BLOCK and a SENSOR BLOCK. The middle BLOCK comprises the SUBBLOCKS of DEAD-TIME RETRIEVALS, MODULES/ SEMI-MODULES, and DEAD-TIME FORWARDINGS. Any BLOCK or SUBBLOCK may be omitted if not needed. PROCESS AUXILIARIES can be accessed anywhere.

An optional SECTION can be appended to the end of UDE in order to detect and handle STATE-EVENTS. The author implemented this option simply to trace some PLANT VARIABLES and produce extra output during the first few

```
C------------      U                   U                   E
 100  CONTINUE
      IF(NUP.LT.101)GOTO 591
C                  CONTROL          BLOCK
      CALL CNTR(8HPC-1         ,1)
      CALL CNTR(8HPC-2         ,2)
      IF(JPS.EQ.1)GOTO 400
      X2=CN(1,2)
      E1=CN(2,2)
  590 IF(JPS.EQ.2)W2=CV2*P2*XFNCN(X2,11,AVX,AVY)
C                  PROCESS          BLOCK
  400 CONTINUE
      CALL MT  EC(8HTCMPS-1,1)
      CALL UANFLD(W1,W2,W3,ITEV,F2)
      IF(NUP.LT.101)GOTO 500
C                  SENSOR           BLOCK
      CALL SNSR(8HFT-1         ,1,W1)
      CALL SNSR(8HPT-2         ,2,P2)
      IF(KEEP.EQ.1.AND.JPS.EQ.2.AND.T.GT.TLATE.
     1                    AND.MIN.EQ.1)ITM=MIN
  500 IF(ISTP.GT.KUL)RETURN
C                  DEBUG SECTION
      IF(KEEP.EQ.1)CALL TVSTT
C------------      U                   P                   A
```

Figure 4.3 The UDE PART for a Chemical Plant Model.

steps in trial RUNS. Hence the name the DEBUG SECTION. The SECTION is not executed if ISTP exceeds KUL where ISTP is maintained by ACES and KUL is user-supplied.

ITRM may be set to a nonzero value in UDE or in UPA to request a premature termination of the RUN. ACES then directs execution to the TERMINAL REGION where UTE is called. ITRM is set to 0 in HSKPIN so that normal termination occurs when T exceeds TFIN.

IRPT may be set to a nonzero value in UTE to demand a RERUN. A negative IRPT causes a return to the initial STATE; UDE will be executed twice as usual before linking back to the INITIAL REGION. A positive IRPT avoids RESET. HSKPIN sets IRPT to zero so that the default is a single RUN.

4.3 Housekeeping in ACES

Under the supervision of HSKPEX various routines tend to the chores of simulation performing behind the scenes as much as possible. SUBROUTINE HSKPSI is the SEGMENT INITIALISER. SUBROUTINES HSKPIN, HSKPDE, HSKPPA, and HSKPTE are assigned to operate respectively in the INITIAL REGION, DERIVATIVE SECTION, PARALLEL SECTION, and TERMINAL REGION. SUBROUTINE FRWRD forwards a STATE-VARIABLE and its derivative to the ACES SIDE and FUNCTION RTRV retrieves a STATE-VARIABLE to the USER SIDE. Called by the UGINT interface SUBROUTINE GINT handles the housekeeping related to integration. SUBROUTINE KATC1 registers calls to the library and examines compatibility

between them. SUBROUTINE DINLE is allocated the task of producing a suitable report when ACES detects a fault.

4.3.1 SUBROUTINE HSKPSI

ACES relies for flexibility on two FORTRAN features, namely DATA and EQUIVALENCE statements. Some array elements are given unique names for easy recognition and equivalenced to their proper locations. This also means less work and less risk of corruption in editing the library.

HSKPSI initialises many variables using DATA statements. With the exception of NN these variables are transferred to labelled COMMONS through assignments. Other subprograms harness, usually under equivalenced names, the elements of the MSY and SY arrays. Among those initialised here are (i) storage limits, (ii) defaults, and (iii) a group consisting of KUL, NUR, IOO1, IOO2, IOO8, IOO9, NIN, and KAT. The defaults include certain stream, controller, sensor, and module flags. Integration defaults are in a block like the INTINF BLOCK of ICL SLAM supplemented by a few parameters for GEAR, that is adapted DIFSUB (Gear 1971).

A DATA statement initialises NN to 0 which is later incremented by 1 each time HSKPSI is called. This device distinguishes the first simulation from the subsequent ones. Only KUL, NUR, IOO1, IOO2, NIN and KAT will be reset here thereby nullifying any changes

introduced by the user during the simulation. Other defaults provided by HSKPSI the first time and overridden by the user will not be recovered.

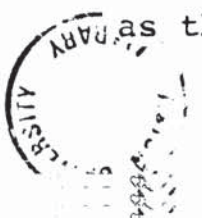
4.3.2 SUBROUTINE HSKPIN

The first statement here increments the INITIALISATION NUMBER, NIN. The variables set or reset are JEEP, KEEP, IOO6, IOO5, JGR, JSTART, IRPT, ITRM, ISTP, and TL. VRSN and NIN are printed out to help documentation. Subsequently HSKPIN assigns JPS to O and calls UIN marking the interface by ILK. UIN is the local name for USMLND. On the USER SIDE the UIN PART is executed to complete initialisation. On return from UIN some checks are carried out on integration specifications. If a fault is detected, DINLE is called upon to report it to the user. Otherwise HSKPIN proceeds with documentation. Figure 4.4 exemplifies a HSKPIN report.

ACES uses KIM to identify individual subprograms in the library. KIM is always set in a DATA statement so that future updates can be implemented conveniently. DINLE converts KIM to a name using an internal stack AD.

4.3.3 SUBROUTINE HSKPDE

HSKPEX calls HSKPDE every PASS to obtain fresh derivatives from the USER SIDE. For this purpose HSKPDE refers to USMLND, known locally as SUBROUTINE UDE. HSKPDE's main task is enveloped in a DO loop, alluded to as the JPS-LOOP. JPS is either 1 or 2 here.



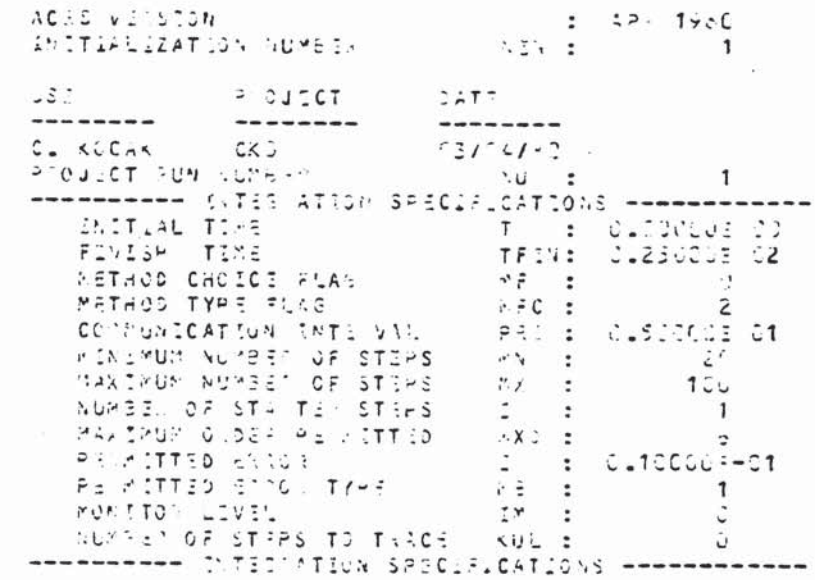


Figure 4.4 A Documentation Produced by SUBROUTINE HSKPIN.

The last labelled subprogram accessed in the UDE PART is always recorded in SAN. HSKPDE blanks SAN on entry. The STEP COUNTER ISTP is incremented by 1 if KEEP equals 1. ISTP is reset to 1 if it reaches a large number denoted by ISTPL.

The JPS-LOOP handles various counters. JDV, JLF, JLR, and JD are cleared every PASS but JCN, JMD, and JSN are zeroed only if ISTP is less than 3. (These variables are described in Appendix 2). ILK marks the interface before JEEP, TL, and DT are updated and UDE called. On return, if it is the FIRST ROUND (JPS=1) of the FIRST PASS (IOO5=.TRUE.), then JCNS, JMDS, JLS, JSNS, and JDS store respectively JCN, JMD, JLF, JSN, and JD. When IOO5 is .FALSE. the last task of the JPS-LOOP is to verify that JDV equals JDVS.

If it is not the FIRST PASS, that is if IOO5 is .FALSE., the control is returned to HSKPEX on exit from the JPS-LOOP. Otherwise there are more chores to attend to. HSKPDE stores in JDVS the value of JDV and checks that JCN, JMD, JLF, JLR, JSN, and JDS all reach consistent values during the first two ROUNDS. JDV is not allowed to go negative. The compatibility of DJ allocation with JDV is tested if GEAR's stiff options have been selected in the INITIAL REGION. Finally, as exemplified in Figure 4.5, HSKPDE outputs the name and value of the independent variable followed by a list of MAPPABLE CALLS registered in the UDE PART. A model without derivatives will prompt HSKPDE to issue a warning message.

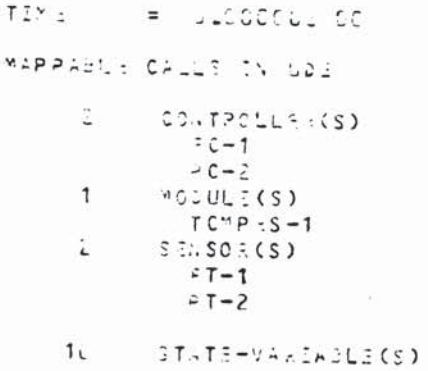


Figure 4.5 A Documentation Produced by SUBROUTINE HSKPDE.

4.3.4 SUBROUTINES HSKPPA and HSKPTE

These are among the shortest subprograms in the library. HSKPEX calls HSKPPA at communication times only. This routine sets JPS to 3, marks the interface, and calls USMLND (or UPA as it is locally known). The FIRST PASS is a communication time. On return IOO5 is switched to .FALSE. which signals the end of the FIRST PASS. If and when MULTIDERIVATIVE SEGMENTS are implemented, this switch would be made in the last PARALLEL SECTION. The USER SIDE executes the UPA PART when JPS is 3.

HSKPTE sets JPS and ILK to 4 calls USMLND (or UTE as HSKPTE knows it) and this executes the UTE PART.

4.3.5 SUBROUTINE FRWRD and FUNCTION RTRV

These subprograms operate inside the JPS-LOOP and handle the duality of STATE-VARIABLES. FRWRD provides the dispatch from the USER SIDE to the ACES SIDE. It does not allow calls unless JPS is 2. If the allocation is not already full, it increments JDV and proceeds. If KEEP is 5, then a numerical Jacobian is being calculated for GEAR and so the derivative is stored in DJ instead of the normal array XD. The STATE-VARIABLE is registered in XX and XI arrays during the FIRST PASS. The integrator always updates XX but leaves XI intact in case a RESET is required. Re-registering the STATE-VARIABLE is needless during LATER PASSES provided there are no discontinuities in the model. FRWRD assumes this and so requires a RESET to be initiated

when this condition is not fulfilled. Another condition imposed by FRWRD is that JDV should not exceed JDVS during LATER PASSES.

FUNCTION RTRV retrieves the updated STATE-VARIABLE in XX to the USER SIDE. It does not allow calls unless JPS is 1. FIRST PASS calls are illegal even if JPS is 1 since no integration has yet taken place in the current RUN. JDV is forced to remain less than or equal to the storage allocation JDVL and JDVS. If everything is in order, then the STATE-VARIABLE is returned in the FUNCTION name. It is redundant to retrieve the derivative.

4.3.6 SUBROUTINE GINT

HSKPEX calls GINT through the UGINT interface. The value of MF determines which integrator is called in GINT. This is detailed in Appendix 2. The argument lists of the ACES integrators deliberately match those of ICL SLAM. M denotes the size of the STATE-VARIABLE vector XX. ICL SLAM automatically adjusts M during the translation.

ACES provides a suitable default set for integration specifications which can be altered in the UIN PART. The default integrator is EULR. MF values larger than 10 signal user implementations in the UINTG interface. This can be used to integrate difference equations for instance.

HN and HX, calculated in GINT, represent the minimum and the maximum step-size permitted in a particular COMMUNICATION INTERVAL. H is the current step-size and

is always equal to HX for EULR, TRPD, and RKD. For these integrators I, MX, IM, E, and ME are all redundant.

4.3.7 SUBROUTINE KATCI

CCNT, CNTR, SNSR, all the library MODULES/SEMI-MODULES and some others call KATCI to be registered so that a more comprehensive debug report can be output if needed. Some routines which normally constitute the last link in a chain do not call KATCI in order to lessen housekeeping. KATCI also enforces various structural rules of ACES. Obviously it would be prohibitive to guard these rules at every PASS and so ACES compromises to police them during the first two steps, that is when ISTP is less than 3.

The caller's identity and type are supplied in KIMCIK and KONAK respectively. KONAK ranges from 1 to 6 to indicate a MODULE, a CCNT call, a CNTR call, a SNSR call, a SEMI-MODULE, or a miscellaneous type subprogram. The first four types in thisseries carry a label AD and an index I in their argument list and pass them on to KATCI.

The call register is KAYIT, equivalenced to MSY, and KAT is its counter. KAT is incremented here and decremented later by the caller prior to a RETURN statement. KATCI next performs a series of tests on KONAK. If KONAK exceeds 5, no further action is needed. SEMI-MODULES outside the UDE PART are also given the same treatment.

If KONAK is less than 5, then AD is copied into SAN as the last label encountered. Outside the UDE PART this is a mishap and DINLE is called to report it to the user. Control is returned to the caller if ISTP exceeds 2. The rest of the code concerns the first two steps only when KATCI undertakes to supervise the consistency of calls from PASS to PASS and the possible compartmentalisation of the UDE PART into BLOCKS or SUBBLOCKS. The procedure is illustrated by considering the cases where KONAK is 1 which is relevant to MODULE calls.

When called by a MODULE KATCI ensures that both I and JMD are between 1 and JMDL before copying AD into XM(I,1) and incrementing JMD. During LATER PASSES it checks that JMD does not exceed JMDS and that JCN equals JCNS and JLR equals JLS. JLF and JSN should always be 0. Again during LATER PASSES KATCI verifies that I matches the record and also that the same number of derivatives have been handled so far. I is stored in MM (JMD,7) and JDV in MM(I,8). KATCI ensures that the Ith MODULE is encountered once a PASS only. Some of these checks are also performed when a SEMI-MODULE calls KATCI.

KATCI is syncronised with HSKPDE which also stores some counters and performs a few tests.

4.3.8 SUBROUTINE DINLE

ACES has delegated to DINLE the task of producing a debug report. Here ILK, JPS, SAN, and KAYIT are analysed and translated into a reasonable output to aid the user in

identifying the cause of a failure.

A prominent feature is the stacking of the names of possible interfaces in GIRIS and of the library programs in AD. Both GIRIS and AD can be altered and expanded in the future and little editing will be needed elsewhere. This promotes flexibility and convenience.

The caller sets NENE to 2 if a high-level report is needed. This prompts DINLE to output VRSN, USER, PROJ, DATE, NUR, and NIN followed by the last interface with the USER SIDE. The variables USER, PROJ, DATE, and NUR are user-supplied, the first three being alphanumeric. They are all defaulted by HSKPSI. VRSN is an alphanumeric name to identify the version of ACES in use. NIN is the INITIALISATION NUMBER.

Next a branching takes place in DINLE. If JPS is -1, the RUN is in the INTEGRATION MODE and the logic joins the low-level report section. Otherwise the relevant PART of USMLND is printed. If the PART is UDE, then the ROUND NUMBER is output in addition. If KAT is less than 1, then KAYIT is empty, that is no library subprograms have been registered, and the logic is again directed to join the low-level report section. On the other hand, a positive KAT results in a map of KAYIT, in order of calls, after printing SAN if not blank. Then the low-level report section is executed.

The low-level report includes the caller's name and the failure location KOZA side by side, BAGSIZ, T,

KEEP, and ISTP. The RESET MODE is indicated separately if IRPT is negative. BAGSIZ actually names the independent variable for which the default is TIME. A subprogram to be replaced is identified by KOZA=0 in which case DINLE issues an appropriate message. Figure 4.6 depicts a high-level report produced by DINLE.

Now the convention is such that the error is not fatal if NENE is 0. Having produced a low-level report in this case DINLE returns control to the caller which then proceeds to the next statement in the code. If NENE is not 0 however, DINLE calls on SUBROUTINE UERR to take corrective action. The default for this interface simply ceases execution. UERR can be used in the future as an opening for interaction.

4.4 The Integrators

EULR, TRPD, and RKD are based on DYFLO. They implement respectively the simple Euler, the modified Euler, and the fourth-order fixed-step Runge-Kutta using the same symbols as in DYFLO.

EULR is the default integrator in ACES. GINT sets the step-size H to the maximum allowed. Theoretically H can be changed at communication times. JDV is the STATE-VARIABLE COUNTER and should not exceed M, the dimension of the vectors XX and XD. EULR requires one PASS per step. For this reason it allows KEEP to remain at 1 as set by HSKPIN initially and switches IOO6 to 1 every MN PASSES to flag a communication time. GINT alters IOO6 to 0 before calling EULR again.

```
ACES VERSION          : APT 196
USER                  : C. KOCAK
PROJECT               : CKJ              03/04/80
PROJECT SUB NUMBER    :      1
INITIALIZATION NUMBER :      1
USERLND PART          : UDE
FOUND NUMBER JPS      :      1
LAST LABELLED ITEM    : HOLDUP-1
   CALLED     SSLSH
   CALLED     SFLSH
   CALLED     XDAPT
   FAILED IN XDAPT  , LOCATION    5
            TIME    =  0.196000E 02    KEEP=    0
INTEGRATION STEP      :     94
```

Figure 4.6 A High-level Report by SUBROUTINE DINLE.

TRPD employs a work vector DXA to accommodate the intermediate PASS. Since two PASSES are needed per step TRPD sets KEEP to 1 every other PASS and waits for the number of PASSES to be twice MN before flagging a communication time by setting IOO6 to 1. Again GINT switches IOO6 to O.

RKD involves two work vectors, XA and DXA. It requires four PASSES per step and accordingly takes care of KEEP and of IOO6.

GEAR depends on two routines XDFSB1 and XDFSB2. When IOO6 is 1 GEAR determines the next communication time, switches IOO6 to O, and bounds IM by O and 2. IM is the MONITOR FLAG applicable to variable step algorithms only. It indicates what action the integrator should take if the error is unacceptable. If IM is O and H is already the minimum allowed, then GEAR aborts the RUN. If IM is 1, GEAR proceeds despite large error printing a warning. This is also the case if IM is 2 where extra detail is produced every time H is reduced. Now GEAR can repeat a step which the user may request by setting KEEP=-1. If ISTP is 1, GEAR will reject such a request however.

XDFSB2 AUXILIARY is entered if MFC exceeds 1. It uses output channel IOO8 to produce a trace of its procedures. This has been very useful in investigating GEAR and can be further harnessed in the future. XDFSB1 is without a trace option and is entered if MFC is 1. MFC can be assigned a value in the UIN PART.

PJ is the Jacobian employed by the stiff options of GEAR. PJ has to be dimensioned correctly in the UGINT interface. UPED supplies an analytical PJ if MF is 1.

4.5 The Control Subprograms

Three distinct phases are observed in process control, namely measurement, decision, and actuation. The first phase can be taken broadly to mean "getting to know the STATE", the second "working out what to do", and the third "implementing the decision made". Usually there are substantial delays involved in measurement and in actuation but (automatic) decision making is almost instantaneous.

In CCNT, CNTR, and SNSR ACES provides a cascade controller, a general-purpose controller, and a general-purpose sensor respectively. The relevant storage is given in Appendix 4.

4.5.1 SUBROUTINE SNSR

This routine incorporates a first-order and a second-order transfer function to represent measurement delays. In addition it allows for zero delay and provides an exit for higher-order transfer functions. This interface is UTFN the default for which halts execution. The KF argument is set to 2 for a SNSR call.

For the Ith measuring instrument SN(I,1), SN(I,2), SN(I,3), and SN(I,5) store respectively the label AD, the output SNSD, the input XMSR, and the steady-state gain SNG. MDS(I,1) contains the delay order ISD and is

set to 0, 1, 2, or higher.

SNSR illustrates (i) identification of a call through a label AD and an index I, (ii) a call to KATCI with a suitable KONAK value, (iii) how to mark an interface with ILK, and (iv) decrementing of KAT before returning to the caller.

4.5.2 SUBROUTINE CNTR

CNTR undertakes the decision and actuation phases. P, PI, and PID conventional controllers are delineated in the code. Two interfaces are incorporated, namely UCNT for user-supplied decisions and UXCNT for biasing and limiting the decision before the actuation phase. The control action flag IT takes on values 1, 2, 3, or higher to signal P, PI, PID or user-supplied algorithms. IS identifies the sensor whose output SNSD is the input to the controller. CNTR retrieves SNSD from SN(IS,2). ID is the order of delay involved in the actuation phase. ID takes on values 0, 1, 2, or higher to indicate zeroth, first, second or higher order delays. Higher order delays are supplied by the user in the UTFN interface again. Here the KF argument is 1 for a CNTR call.

CNTR first obtains SNSD and estimates "the decision" CSG given SP, CG, X, RPT, RT and RA as needed. It modifies CSG in UXCNT if necessary and then actuates it to yield the final output DCS.

The conventional algorithms are depicted in Figure 4.7. DYFLO uses a fraction of the controller gain to normalise the error ER and also requires the user to make separate integrator calls for a PI or PID controller. ACES eliminates both these features.

CNTR requires SNSD to be made available by SNSR at the end of the FIRST ROUND and this can be problematic if there is no measurement delay and the input XMSR to the sensor is updated during the SECOND ROUND. Similarly no actuation delay causes CNTR to deliver DCS during the SECOND ROUND and yet the PROCESS BLOCK may need DCS in the FIRST ROUND. These however can be regarded as "approximation" problems as they will vanish with the inclusion of a suitable delay.

4.6 MODULES

These routines incorporate a dynamic conservation model of a piece of plant item. They are used in the MODULE/SEMI-MODULE SUBBLOCK within the UDE PART. They are amenable to labelling and indexing which makes them easy to recognise (Sections 4.3.7 and 4.6.1).

Unlike DYFLO, ACES allots special storage to MODULES. The general storage arrangement is shown in Appendix 4.

4.6.1 SUBROUTINE MHLDP

The purpose of this MODULE is to model single-phase holdups with one input (II) and one output stream (IO).

Algorithm	Equations
P	CSG=CG.X.ER
PI	CSG=CG.X.(ER + PI.RPT)
	$PI = \int ER.dt$
PID	DA = RA.(ER + PID/RT)
	CSG=CG.X.(DA + PI.RPT)
	$PID = \int (ER-DA).dt$
	$PI = \int DA.dt$

Figure 4.7 The conventional controllers in ACES. ER, CG, X, RPT, RT, and CSG denote respectively the instantaneous error, controller gain, action flag, repeat factor, rate time, and control signal. RA is introduced to limit the noise amplification by the derivative mode. ER = SNSD-SP where SNSD is the measurement and SP the setpoint.

It embeds a dynamic continuity equation for (i) the total mass, (ii) each component, and (iii) the total energy (not if isothermal). MRX can be 0, 1, or 2 to flag respectively that there are no reactions involved, that the relevant reaction terms are determined outside the MODULE prior to entry, or that these terms are obtainable in the interface URX.

MRX is specified in MM(I,1) where I is the MODULE index. MHL or MM(I,2) can be 0 to flag constant holdup in which case the MODULE simply sets S(IO,1) to S(II,1) instead of using the dynamic total mass balance. The latter is employed if MHL is not 0. Reactions are allowed for in either case.

KATCI oversees the usual housekeeping. MHLDP undertakes some tests for the first two steps calling KOLCU also. JWH or MM(I,6) indicates on which ROUND S(IO,1) can be estimated. This choice is applicable to constant holdup cases only and is further restricted to the SECOND ROUND if reactions are present. If the holdup is externally heated, then MHLDP expects the load Q to be placed in XM(I,5).

The estimation of the derivatives and the forwarding are delegated to the PROCESS AUXILIARY XHLD. Other MODULES may also harness XHLD. A call to XSTMP yields the holdup temperature.

4.6.2 SUBROUTINE MDHX

The author believes that this MODULE probably exhibits the most radical departure from DYFLO. It models a counter-current, shell-and-tube heat-exchanger divided into N sections. The initial temperature profiles are established in the AUXILIARY XICHX in the INITIAL REGION. Section by section MDHX handles the wall, the shell-side and the tube-side in that order. Each side is treated as a series of completely mixed holdups; hence the use of XHLD.

The wall dissipates heat to each side where the sectional temperature used in the heat-transfer equation is the average of those of the streams entering and leaving. HABN is the sectional product of the film coefficient and the heat-transfer area. The wall temperature is in S(JS,9) where JS is equal to the sum of IS, the first shell-side stream, plus the section number J. IT is the first tube-side stream, XM(I,5) is set either to QS or to QT depending on whether XHLD is called for a shell-side or a tube-side section.

Like DYFLO's DHE, MDHX assumes constant holdup on each side and allows no reactions. It is believed that sequential modularisation spoils DHE. MDHX however relies on XHLD, not MHLDP, and should not present such problems.

4.6.3 SUBROUTINE MGPLT

The DYFLO approach to distillation has been detailed (Section 3.1.7). ACES has introduced relatively minor modifications due to lack of alternatives. The basic assumptions about a stage are retained including (i) negligible vapour phase holdup, (ii) complete mixing in each phase, and (iii) equilibrium between the two phases. The pseudo steady-state methods have been rejected however. The flow rate for the liquid stream leaving the stage is allowed to vary linearly with the liquid holdup. Both the empirical and the backward derivative choices have been included for the estimation of the local boil-up. Reactions are permitted as well as external heating.

MGPLT replaces DYFLO SUBROUTINES STAGE, STGS, STGF, STGH, BOT, and VVBOIL. It includes differential equations constituting the total and componentwise mass continuities. The energy balance is harnessed to estimate the local boil-up, S(IV,1).

First consider the backward derivative approach. An enthalpy balance for a stage is

$$\frac{d}{dt}\ (HL.S(IL,3)) = \text{input} - \text{output} + RXTQ + Q$$

where HL = the liquid holdup,

t = time,

RXTQ = the heat generation due to reactions,

and Q = the external heat load.

Here, input (output) denotes the sum of the enthalpies of the streams entering (leaving) the stage. Output includes the product S(IV,1). S(IV,3). Let DHLEN represent the derivative, and define ENER to equal the right-hand side of the equation above less this product. The boil-up S(IV,1) is thus given by

$$S(IV,1) = (ENER-DHLEN)/S(IV,3).$$

The code calculates DHLEN as a backward derivative at the end of a step, that is when JEEP is 1. The need has been pointed out (Section 3.1.7) to solve such column equations simultaneously. ACES has made a compromise in order not to inconvenience the user. The value so estimated is stored in XM(I,9) for one step before being assigned to S(IV,1).

The backward derivative option is a discretisation since it updates S(IV,1) once every step assuming the boil-up to remain constant meanwhile. This may introduce artificial difficulties into the integration process.

ACES employs the difference between two temperatures as opposed to enthalpies so that the empirical approach is less dependent on engineering units.

Although fairly lengthy the code for MGPLT delineates various parts carefully and clearly and needs little commentary. Molar flowrates rather than mole fractions are made STATE-VARIABLES. The total continuity yields the updated (liquid) holdup and the change in this over the PASS is used algebraically to alter S(IL,1) where

applicable. The component continuities return the new composition of the holdup (and of the liquid stream leaving when this exists). A bubble point calculation in XBBBL gives the stage temperature and the vapour composition.

4.6.4 SUBROUTINE MHTR

The heat-transfer equation using UA is a familiar sight in chemical engineering:

$$Q = UA\ (TX-TEM)\ ,\ \text{say.}$$

If the wall heat capacity is large, then a two-flux mechanism may be formulated:

$$QX = HAX\ (TX-TW),$$

$$Q = HA\ (TW-TEM),$$

$$\text{and}\ \frac{d}{dt}(TW) = (QX-Q)/WC.$$

MHTR encapsulates these approaches. The MODULE serves as a simple heater or cooler. TEM is the main stream temperature. TX and TW mean the utility and wall temperatures. WC denotes the wall heat capacity. Q represents the heat flowrate into the main stream while QX is the heat flowrate from the utility to the wall. t is time.

The choice is flagged to MHTR by MQT. MQT=3 is reserved for a reboiler where condensation is assumed as in DYFLO's REB. TX is then determined iteratively. When TX is arrived at, the part for MQT=2 is joined which codes the two-flux mechanism above. The simple UA case is executed only if MQT=1.

MHTR is always to precede the MODULE it serves. The master MODULE is designated the index MMM and stores TEM in XM(MMM,10) during the FIRST ROUND. Q is then supplied by MHTR in XM(MMM,5) during the SECOND ROUND. If there are additional heat loads on MMM, then XM(MMM,5) should be suitably incremented after the call to MHTR in the SECOND ROUND as MHTR cannot serve the same MODULE more than once per PASS.

MHTR ensures that MMM does not precede itself in the UDE PART. This demonstrates how seriously ACES takes housekeeping duties. MHTR also shows how iterative algorithms are supervised. An iteration counter ITF is introduced. An upper limit ITLH is imposed on ITF. Negative square roots are guarded against. On convergence ITF is output if KUL exceeds ISTP. ITLH is defaulted by HSKPSI. (Refer to Appendix 2.)

4.6.5 SUBROUTINE MTRBC

This MODULE is based on a DYFLO case investigating the control of a turbocompressor. An iterative algorithm coupled to JWH uses the interface FUNCTION UPRBN2 supplying the compressor characteristics in order to determine the gas flowrate. The latter is then employed in the turbine momentum balance which is integrated for the speed.

4.6.6 SUBROUTINE MLXTR

Liquid-liquid extraction is not uncommon. The aim

of this MODULE is to model a simple mixer-settler stage. JI and JO denote the streams entering and leaving the raffinate phase and II and IO are the corresponding extract streams. Reactions are allowed in the raffinate only. Each holdup is regarded as completely mixed. The other major assumptions are (i) isothermal operation, (ii) constant holdups, (iii) constant distribution coefficients, and (iv) negligibly small mass-transfer between the phases.

As usual XM(I,1) is reserved for the label AD where I is the MODULE index. XM(I,2) and XM(I,3) store respectively the holdups for the raffinate (HJ) and the extract (HI). XM(I,N+5) contains the constant distribution coefficient for the Nth component. The total continuity equations are

$$S(IO,1) = S(II,1) \text{ and } S(JO,1) = S(JI,1) + RXTM.$$

Reactions are allowed for when they occur. The component balance over the whole stage is

$$\frac{d}{dt} (HJ.\ C(JO,N)+HI.C(IO,N)) = \text{input-output + reacted}$$

where input = S(II,1).C(II,N) + S(JI,1).C(JI,N),

output = S(IO,1).C(IO,N) + S(JO,1).C(JO,N),

and reacted = RX(N).

Inserting S(IO,1) = S(II,1) and C(IO,N) = XM(I,N+5).C(JO,N) yields after manipulation the differential equation

$$S(II,1).(C(II,N)-C(IO,N)) + S(JI,1).C(JI,N)$$

$$\frac{d}{dt}(C(JO,N)) = \frac{-S(JO,1).C(JO,N) + RX(N)}{HJ+HI.XM(I,N+5)}$$

The translation into a FORTRAN code is straightforward. Each raffinate mole fraction becomes a STATE-VARIABLE and the corresponding extract mole fraction is determined in the FIRST ROUND using the distribution coefficient. JWH is operative. A maximum of 10 components is allowed.

4.7 SEMI-MODULES

These subprograms are for steady-state operations. They can be employed in transient analysis as approximations but this requires some care since steady-state simulation is infested with algebraic loops. Within the UDE PART SEMI-MODULES are compelled to operate during the FIRST ROUND only. They can be used in the INITIAL and TERMINAL REGIONS also.

SEMI-MODULES SCVBL, SFLSH, SHTX, SPCND, SSHX, SSMMR, and SSPLT are modified versions of DYFLO subprograms CVBOIL, FLASH, HTEXCH, PCON, CSHE, SUM, and SPLIT respectively. SSLSH is based on a DYFLO case.

4.7.1 SUBROUTINE SFLSH

SFLSH is designed to handle flash operations at a given pressure and with a variable heat load. Stream II

is split into a liquid (IL) and a vapour (IV) stream in equilibrium with each other.

Based on XBBBL and XDWPT calls SFLSH first determines whether a two-phase split is possible. All liquid or all vapour exits can also come about. In the two-phase part R is guessed using a memory feature if needed. There is an inner loop to converge on the equilibrium where the Newton-Raphson scheme is used and Wegstein's formula (XLWGS) promotes the solution. At the end of the inner loop a new estimate of R is produced in R1. Subsequently R is updated by partial substitution (XPSBS) and the outer loop continues until it converges.

As an approximation the activity coefficients are assumed constant for differentiation purposes but the interface UACTY updates them within the inner loop. This can conceivably cause convergence problems in non-ideal cases.

4.7.2 SUBROUTINE SPCND

This can be regarded as a special case of SFLSH where the inner loop is eliminated by fixing the (condensation) temperature TC. R is found by a Newton-Raphson implementation and the heat load HT is calculated internally.

SPCND first ensures what type of condensation would prevail, that is total, partial, or none. A guessed value for R is provided in the two-phase part. The trivial root at R = 1.0 (Section 3.1.4) is avoided by multiplying R by 0.3 when the derivative is positive.

4.7.3 SUBROUTINE SSHX

This subprogram is an adaptation of DYFLO's CSHE. It is a steady-state representation of a counter-current, shell-and-tube heat-exchanger. IS(ISO) is the stream entering (leaving) the shell-side and IT(ITO) is the counterpart on the tube-side. UA has the usual meaning. The iterative scheme in SSHX is oscillatory. The steps are (i) guess the amount of heat transferred Q, (ii) impart Q to IS by calling SHTX, (iii) add -Q to IT by calling SHTX, (iv) determine the log-mean temperature difference LMTD, (v) estimate QC = UA. LMTD, (vi) update Q by partial substitution (XPSBS) and go back to (ii) unless Q has converged.

SSHX solutions are found to satisfy the heat transfer equations.

4.7.4 SUBROUTINE SSLSH

SFLSH operates at a given pressure. SSLSH incorporates SFLSH within a pressure loop assuming the ideal gas law. If RBV denotes the gas constant divided by the volume and TABS converts the temperature to the absolute, then the ideal pressure PIC is given by PIC = S(IV,1). RBV.(S(IV,2) + TABS). The full scheme is (i) guess the pressure S(IL,4), (ii) flash, (iii) find PIC, and (iv) update S(IL,4) by partial substitution and go back to (ii) unless S(IL,4) has converged.

SSLSH expects the user to supply the first guess for the pressure but uses a memory feature thereafter.

4.8 PROCESS AUXILIARIES

Of these XACTl sets the activity coefficients to 1 for a given stream. XMRG3 formulates Margules' equation for a ternary mixture. XBBBL and XDWPT perform a bubble-point and a dew-point calculation at a given pressure. Dead-time is treated like a pseudo-integration process using a forwarder (XDTF) and a retriever (XDTR) for single variables. Pipe-line delay routines XSDTF and XSDTR are designed to operate on several stream variables concurrently, calling XDTF and XDTR as appropriate. XHLD estimates holdup derivatives and forwards them to the ACES SIDE. Both MHLDP and MDHX rely on XHLD. XICHX establishes the temperature profiles for MDHX. XICHX calls SSHX. By comparison, ICHE, its counterpart in DYFLO, succeeds a call to CSHE within DHE which may lead to the inclusion of upsets in the initial conditions. XNTHL and XNTHV calculate liquid and vapour stream enthalpies. XPABCT codes the Antoine equation. XSTMP's assignment is to find a stream temperature given the enthalpy and the composition. Finally XVLV contains a few valve equations.

XDTF, XDTR, XHLD, XSDTF, XSDTR, and XVLV are called in the UDE PART only. The rest of the subprograms mentioned here can be called anywhere in USMLND. XHLD is further restricted to the SECOND ROUND.

A notable change in the equilibrium routines is the addition of a memory feature. As mentioned before activity coefficients are assumed constant in derivative calculations within these subprograms.

4.8.1 SUBROUTINES XDTF and XDTR

Coupled together these subprograms implement fixed-step dead-time and should not be employed with variable-step integrators. In the UDE PART XDTR calls succeed the CONTROL BLOCK but precede the MODULE/SEMI-MODULE SUBBLOCK which is followed by XDTF calls.

The input YIN is stored in a work array YA by XDTF during the SECOND ROUND and retrieved after NTL steps by XDTR during the FIRST ROUND. The first NTL steps need to be initialised. The user may choose to fill either YIC (NAR=0) or YA (NAR≠0). In the former case XDTR handles the necessary transfer during the FIRST PASS.

Unlike STATE-VARIABLES dead-time variables have to be ordered by the user matching the storage of NTL values in NTLL vector with YA or YIC. The size of these arrays are determined by MVD and NDS, that is the maximum number of variables delayed and the maximum number of delay steps needed. NCD is a work vector reserved for step counters. Every NTL steps XDTF sets the right counter to 0.

4.8.2 SUBROUTINE XICHX

This subprogram prepares for a MDHX call. A call to SSHX establishes the exit temperatures which are then used to determine the sectional temperatures within the exchanger. I denotes the MODULE index and IS (IT) is the stream entering the shell-side (tube-side). The streams on either side are numbered consecutively starting from the inlet stream. The number of sections is N and is stored in MM(I,4). Thus the outlet streams are ISN=IS+N and ITN=IT+N. XM(I,4) contains HABN, the sectional film coefficient times the area. UA is estimated from UA=HABN.N/2 prior to calling SSHX. Then the molar flowrates and compositions are transferred to intermediate and exit streams. DTA denotes the log-mean temperature difference between the IS end of the exchanger and the end of the Jth section. It is estimated for each section as follows:

$$DT1 = S(IS,2) - S(ITN,2) ,$$

$$DT2 = S(ISN,2) - S(IT,2) ,$$

$$R = \ln (DT1/DT2)/N ,$$

$$DTJ = DT1.\exp (-R.J) ,$$

$$DTA = (DT1 - DTJ)/(R.J) .$$

A call to SHTX transfers -Q to IS where Q = 0.5 HABN.DTA.J and establishes S(IS+J,2). S(IT+N-J,2) is found by subtracting DTJ from S(IS+J,2). The sectional wall temperature, stored in S(IS+J,9), is assumed to be average of those of the streams related to the section. The initialisation is completed by setting the MDH flag or MM(I,3) to -2.

4.9 ALGEBRAIC AUXILIARIES

These include XDFCO, XDFSB1, XDFSB2, XDRVT, XFNCN, XLUSLV, XLWGS, XMWGS, XPSBS, XRNWT, XSOL, XTFN1, and XTFN2. XDFCO simply sets the algorithmic coefficients for GEAR. XDFSB1 and XDFSB2 are identical GEAR auxiliaries except that XDFSB2 uses a special communication channel to inform the user of what it is doing. GEAR calls XDFSB1 if MFC is 1, XDFSB2 if MFC is greater than 1. XDRVT, XFNCN, XLWGS, XPSBS, XTFN1 and XTFN2 correspond to DYFLO's DER, FUN1, CONV, CPS, TFN1, and TFN2. XLUSLV (Jennings 1978) applies LU decomposition to a matrix and XSOL (Forsythe and Moler 1967) then uses the decomposed product to solve a set of linear algebraic equations by forward and backward substitutions. GEAR's stiff options rely on XLUSLV and XSOL. XMWGS is an extension of XLWGS to nested iterations. XRNWT implements Newton-Raphson for a single equation.

4.9.1 FUNCTION XDRVT

This subprogram is accessible in the UDE PART only. It returns in XDRVT the backward derivative of Y with respect to X. Both X and Y need to be on the USER SIDE during the FIRST ROUND and the responsibility of correctly positioning the call is left to the user. XDRVT is set in the FIRST PASS to DRIC supplied by the user and updated internally thereafter when JEEP becomes 1.

4.9.2 SUBROUTINES XTFN1 and XTFN2

These subprograms transform first and second-order transfer functions into a state-variable representation for integration. Let the bar indicate the Laplace domain. The first-order transfer function $\overline{OUT} = \frac{GAIN.\overline{XIN}}{\tau.s+1}$ transforms into D = (GAIN.XIN-OUT)/τ. Similarly the second-order transfer function $\overline{OUT} = \frac{GAIN.\overline{XIN}}{\tau^2 s^2+\varepsilon\tau s+1}$ transforms into D = (GAIN.XIN-OUT)/τ and DD = $\frac{(GAIN.XIN-OUT)}{\tau^2} - \frac{\varepsilon D}{\tau}$.

In the codes, TT, TNV2, and EBT denote respectively τ, the inverse of τ^2, and ε/τ. D and DD are the first and second derivatives. XTFN1 requires OUT to be initialised while XTFN2 wants both OUT and D initialised.

These subprograms can be utilised as general-purpose MODULES. Higher-order transfer functions can be coded in the UTFN interface if needed. The last argument (KF) of UTFN is set to 1 or 2 to identify a call from CNTR or SNSR.

4.10 Clearance Subprograms

ZDT, ZRCT, ZSTC, and ZXIXD clear respectively the whole of D array, the reaction terms, the composition of a given stream, and the XI and XD arrays.

4.11 Input/Output and ROLL-OUT

ACES provides a convenient suite of subprograms for communicating groups of variables. Input and output

are made compatible so as to enable ACES to save a certain STATE in a file and reactivate it later. This capability is called ROLL-OUT in ACES terminology and is in line with the suggestion (Evans and Seider 1976) that a computer system should be file-oriented.

4.11.1 The Display Subprograms

TVSTT displays the elements of the STATE arrays XI, XX, and XD as registered. No output is produced by TVSTT in the FIRST ROUND since forwarding takes place in the SECOND ROUND. TV5R outputs up to 5 variables placed in its argument list.MBS=0 flags that these are alphanumeric. A non-zero MBS indicates REAL variables in the FORTRAN sense. TVDATA prints physical properties NPF to NPL for components NCF to NCL, up to 4 components per line. There are two kinds of stream display. TVSTR4 outputs up to 4 streams as indexed in its argument list. Printed for each stream are S(I,1), S(I,2), S(I,3) and S(I,4). NPC=0 suppresses the compositions. Otherwise the mole fractions for components NCF to NCL are also displayed. The second stream display routine is TVSSTR which produces similar output for a series of streams from NSF to NSL, up to 4 streams per line.

ACES registers MODULES, CNTR calls, and SNSR calls within the UDE PART. The relevant counters are JMD, JCN and JSN. TVCNT displays the controllers up to 4 per line and in order of registration where the level of display is flagged in KCN. If JCN is 0, then no output is

displayed. TVMOD and TVSNS use KMD and KSN as level flags and produce information on MODULES and SENSORS.

The display routines will proceed as far as possible correcting some mis-specifications.

4.11.2 The Reading Subprograms

These include ALCNT, ALDATA, ALMOD, ALSNS, ALSSTR, ALSTT, and AL5R which correspond to the display routines TVCNT, TVDATA, TVMOD, TVSNS, TVSSTR, TVSTT, and TV5R. ALSSTR can also read in data produced by TVSTR4.

The reading subprograms are confined to the INITIAL REGION. Unlike their display counterparts they are very strict and resort to abortion rather than "correction".

4.11.3 The ROLL-OUT Feature

TVNTG has been created to dump all the essential flags and counters. Called after the UDE PART it would have the values of JCN, JMD, and JSN. The user supplies NSF and NSL. The output is in the order CONTROLLERS, MODULES, SENSORS, and streams. The plant data can then be dumped by calling the relevant display routines.

Reactivation begins by a call to ALNTG. This is followed by calls to the reading subprograms corresponding in type and order to the display routines involved in the dumping.

The ROLL-OUT feature can also be used to input a

file suitably created by the user in order to start a simulation case. Free-format reading has been generally incorporated into ACES.

4.12 The Current Interfaces in ACES

The SIDES communicate with each other through interfaces to be supplied by the user. The simulator however has to include defaults or dummies for these in order to satisfy the loader and to keep the program running as far as possible.

ACES includes a default MASTER which declares USMLND in an EXTERNAL statement, calls HSKPEX, and on return stops. This facilitates an independent simulation where USMLND houses the model. USMLND is expected from the user.

The current list of interfaces includes UACTY (Section 4.7.1), UCNT (Section 4.5.2), UERR (Section 4.3.8), UGINT (Section 4.4), UINTG (Section 4.3.6), UPED (Section 4.4), UPRBN2 (Section 4.6.5), URX (Section 4.6.1), UTFN (Sections 4.5.1 and 4.5.2), and UXCNT (Section 4.5.2). ACES never enters UACTY, UCNT, UINTG, UPED, URX, or UTFN unless specifically steered by a user flag.

All these defaults, with the exception of UGINT and UXCNT, will halt execution on entry. A message will be printed.

4.13 The Reservations

The variables that appear in the labelled COMMON blocks should be accessed with caution. The following are reserved for all IJ : S(IJ,5) to S(IJ,9) inclusive, MS(IJ,5), MM(IJ,7), MM(IJ,8), MDC(IJ,4), and MDS(IJ,2).

4.14 Exclusions

Several features have been excluded from the current version of ACES, notably interaction, graphics, combined simulation capabilities, forcing functions, MULTIDERIVATIVE SECTIONS, physical properties supplement, and optimisation. ACES has assumed however a flexible, modular structure which conveniently allows for future expansion.

The author has the opinion that interaction and graphics should be attempted in the next phase of development. The other features seem less important at this stage.

4.15 Summary

A new digital dynamic simulator has been presented in this chapter after the introduction of the relevant methodology. This simulator, called ACES, has a modular, CSSL-type structure and is oriented towards chemical engineering. Emphasis has been laid on convenience and flexibility rather than sophistication. Substantial housekeeping is performed behind the scenes. The implementation includes three major novelties in dynamic

simulation, namely (i) general simultaneity through twice-round execution, (ii) ROLL-OUT, and (iii) special debugging. ACES can be maintained and expanded very easily.

The utility of ACES is illustrated in Chapter 5.

CHAPTER 5 ACES ILLUSTRATED AND VERIFIED

5.0 Introduction

ACES is a digital dynamic simulator oriented towards chemical engineering. Chapter 4 has described the philosophy and methodology of ACES and detailed most of the library subprograms. This chapter illustrates and verifies ACES by means of a selection of examples. The literature-based cases preserve the original symbology as far as possible. A general discussion of the results concludes the chapter.

5.1 The Computing Environment

A batch environment is provided by an ICL 1904S computer in the University Computer Centre. This machine has been used extensively both in the development and in the verification phases. A late arrival on the scene was a GEC 4070 minicomputer made available through the SRC Interactive Computing Facility (ICF). The portability of ACES has been demonstrated by also running two of the example cases successfully on the ICF machine.

The ACES SIDE resides on the ICL computer as a semi-compiled library. The USER SIDE is supplied as a source program. The two SIDES are brought together in a compilation phase which furnishes a complete ACES program in binary form whereby subprograms in the source input override their defaults in the library. A multiplicity

of input/output channels can be used provided they are specified in a program description segment as exemplified in Appendix 5.

Semi-compilation has not been employed on the ICF machine. Each simulation exercise has been completed without a break in the process. The interactive capability of the ICF computer at run time has not been utilised in this work.

5.2 Some Ordinary Differential Equations

Continuous simulation hinges on integration. The testing of ACES integrators thus seems to be the logical start to verification. A number of differential systems have been selected for this purpose.

5.2.1 TEST CK20

A kinetic system has been reported by Hindmarsh and Byrne (1976) as a benchmark problem for stiff integrators. Here it is used to investigate the GEAR options. The equations are

$$\dot{y}_1 = -0.04\, y_1 + 10^4\, y_2 y_3 \quad , \quad y_1(0) = 1$$

$$\dot{y}_2 = 0.04\, y_1 - 10^4\, y_2 y_3 - 3.10^7 y_2^2 \quad , \quad y_2(0) = 0$$

and

$$\dot{y}_3 = 3.10^7 y_2^2 \quad , \quad y_3(0) = 0.$$

Figure 5.1 depicts a portion of the source input for TEST CK20. (This and subsequent listings in this

```
      IF(JPS.NE.0)GOTO 999
C-----------  U                 I                 Y
      READ(5,100)EPS,NOM
      WRITE(IOO2,31)EPS
      WRITE(IOO2,32)
      DO 20 M=1,NOM
         READ(5,100)TH,NSTEP,NFE,NJE,HE,Y1,Y2,Y3
   20    WRITE(IOO2,33)TH,NSTEP,NFE,NJE,HE,Y1,Y2,Y3
      CALL UAL(KOM,KMD,KSM)
      DO 10 M=1,14
   10   Y(M)=0.0
      READ(IOO1,100)(Y(M),M=1,4)
      RETURN
  999 GOTO(1000,1100,1200,2000),JPS
C-----------  U                 O                 E
 1000 CONTINUE
  900 IF(JPS.EQ.2)GOTO 950
C                          FIRST ROUND
      IF(IOOS)GOTO 990
      Y1=RTNV(DY1)
      Y2=RTNV(DY2)
      Y3=RTNV(DY3)
      GOTO 990
C                          SECOND ROUND
  950 CALL FNWRD(Y1,-0.04*Y1+1.0E04*Y2*Y3)
      CALL FNWRD(Y2,0.04*Y1-1.0404*Y2*Y3-3.0E07*Y2*Y2)
      CALL FNWRD(Y3,3.0E07*Y2*Y2)
  990 IF(ISTP.GT.KUL)RETURN
C                    DEBUG SECTION
      IF(JPS.EQ.1)RETURN
      WRITE(IOO2,250)NUT
      CALL TVSTT
C-----------  U                 P                 S
 1500 CONTINUE
      IF(.NOT.ITOS)GOTO 1502
      WRITE(IOO2,35)E
      WRITE(IOO2,34)
      RETURN
 1502 P 1=9.0*T
      WRITE(IOO2,33)T,HS,NNTI,NJ,H,Y1,Y2,Y3
      IF(NNTI.GT.1500)IOO(1)=4
      RETURN
C-----------  U                 T                 -
 2000 CONTINUE
      RETURN
      END
```

Figure 5.1 A Portion of SUBROUTINE USMLND for TEST CK20.

chapter show only relevant extracts; the complete listing of an example is given in Figure 4.2, see also Kocak (1980a) for other examples.) y_i are equivalenced individually to elements of Array Y which is in turn equivalenced to Array SY. The UIN PART first reads and echoes reference results. A call to UAL inputs variables such as USER, PROJ, and DATE, and also integration specifications. UAL (Figure 5.2) is a user-supplied subprogram used in almost every test in this thesis and so it is included in the library in order to save on compilation. On return from UAL, USMLND inputs the initial conditions and this concludes the UIN PART. The UDE PART clearly displays the twice-round principle. FRWRD calls in the SECOND ROUND are matched by RTRV calls in the FIRST ROUND. Retrievals are avoided during the FIRST PASS. The UPA PART illustrates communication with the user. This case employs a variable COMMUNICATION INTERVAL. The DEBUG SECTION is entered if ISTP $\leq$ KUL and leads to the UPA PART. The UTE PART simply returns control to the ACES SIDE which stops execution; no RERUNS have been programmed.

Two library defaults have been overridden, namely UPED and UGINT. UPED provides the analytical Jacobian for this stiff case and UGINT dimensions this array (Figure 5.3.).

The stiff options of GEAR successfully reach T=0.4, the first print time in the reference. Both solutions at T=0.4 are y_1=0.98517, y_2=0.33864.10^{-4}, and y_3=0.14793.10^{-1}

```
      SUBROUTINE UAL(KCN,KMD,KSW)
C------------- USER-SUPPLIED
      COMMON/YENI1/KUL,MF,MFC,MXD,TTIM,EPT,JPS,IOUS,JEEP,ISTP,TL,DT,
     1              NIN,JDV,KAT,ILK,MSY(100),SY(100),DJ(200)
      COMMON/YENI8/IOUS,E,MN,MX,IM,PMI,T,E,MI,
     1             TFIN,USER,PROJ,DATE,BAGSIZ,NUR
      COMMON/YENI7/H,HN,HX,JGR,JSTART,FU,FS,FD,FX,IMAX,
     1               YES,FF,FZ,NNT,NF,NC,ND,NS
      COMMON/SLAM1/KEEP,IOU1,IOU2,IOU(10)
      EQUIVALENCE (BOS,SY(77)),(CIZ,SY(78))
      READ(IOU1,99)USER,PROJ,DATE,NUR,KUL,KF1,KF2,KF3,KF4
      IF(KF1.EQ.0)GOTO 1
      READ(IOU1,100)BAGSIZ,T,TFIN,PSI,MN,MX
    1 IF(KF2.EQ.0)GOTO 2
      READ(IOU1,101)MF,MFC,MXD,I,E,MS,IM
    2 IF(KF3.EQ.0)GOTO 3
      READ(IOU1,101)SY(32),SY(33),SY(51),SY(52),SY(55),MSY(69)
      CALL TVSP(0,8HCNVE     ,8HCNYES    ,8HDFHL     ,8HDFP      ,
     1          8HDFGN     ,0)
      CALL TVSP(0,CIZ,CIZ,CIZ,CIZ,CIZ,0)
      CALL TVSP(5,SY(32),SY(33),SY(51),SY(52),SY(55),1)
      WRITE(IOU2,105)MSY(69)
    3 IF(KF4.EQ.0)GOTO 4
      READ(IOU1,101)YES,FU,FS,FD,FX,FF,FZ,IMAX
      CALL TVSP(0,8HFS       ,8HFD       ,8HFX       ,8HFF       ,
     1          8HFZ       ,0)
      CALL TVSP(0,CIZ,CIZ,CIZ,CIZ,CIZ,0)
      CALL TVSP(5,FS,FD,FX,FF,FZ,1)
      CALL TVSP(0,8HYES      ,8HFU       ,BOS,BOS,BOS,0)
      CALL TVSP(0,CIZ,CIZ,BOS,BOS,BOS,0)
      CALL TVSP(2,YES,FU,CC,DD,EE,1)
      WRITE(IOU2,106)IMAX
    4 READ(IOU1,101)KCN,KMD,KSW
      RETURN
   99 FORMAT(3(A8,2X),6G0.0)
  100 FORMAT(A6,5G0.0)
  101 FORMAT(12G0.0)
  105 FORMAT(1H ,13X,'ITLH=',I4)
  106 FORMAT(1H ,13X,'IMAX=',I4)
      END
```

Figure 5.2 SUBROUTINE UAL.

```
      SUBROUTINE UPED(T,XY,YD,NI,PJ,MI)
C**** USER-SUPPLIED
      REAL PJ(MI,MI),Y(14),XX(NI),XD(NI)
      COMMON/YENI1/KUL,MF,MFC,MXD,ITRM,IRPT,JPS,IOUS,JEEP,ISTP,TL,DT,
     1             MIN,JDV,KAT,ILK,MSY(100),SY(100),DJ(200)
      COMMON/YENI3/IOUS,I,MN,MX,IN,PRI,TIME,E,ME,
     1             TFIN,USER,PROJ,DATE,BASSIZ,MUS
      EQUIVALENCE  (Y1,SY(1),Y(1)),(Y2,Y(2)),(Y3,Y(3))
      PJ(1,1)=-0.04
      PJ(1,2)=1.0E04*Y3
      PJ(1,3)=1.0E04*Y2
      PJ(2,1)=0.04
      PJ(2,3)=-PJ(1,3)
      PJ(3,1)=0.0
      PJ(3,2)=6.0E07*Y2
      PJ(3,3)=0.0
      PJ(2,2)=-PJ(1,2)-PJ(3,2)
      RETURN
      END

      SUBROUTINE UGINT
C-----             THE INTEGRATOR INTERFACE INSERT              --------
C**** SINCE MI IS USED TO DIMENSION PJ ARRAY DYNAMICALLY
C     DATA MI MUST MATCH THE PJ SIZE SPECIFIED ...
C     DXA AND XA WORK VECTORS MUST MATCH JDVL
      REAL DXA(50),XA(50),PJ(3,3)
      COMMON/YENI1/KUL,MF,MFC,MXD,ITRM,IRPT,JPS,IOUS,JEEP,ISTP,TL,DT,
     1             MIN,JDV,KAT,ILK,MSY(100),SY(100),DJ(200)
      COMMON/YENI2/XX(50),XD(50),XI(50)
      EQUIVALENCE (JDVL,MSY(61))
      DATA MI/3/
      CALL GINT(JDVL,XD,XX,DXA,XA,PJ,MI)
      RETURN
      END
```

Figure 5.3 SUBROUTINES UPED and UGINT for TEST CK20.

(The full outputs for TEST CK20 and others are given by Kocak (1980b)). The reference results are the same except for y_3 which is $0.14794.10^{-1}$. This small discrepancy may be due to round-off errors and may lead to instability. The non-stiff option (MF=0) accumulates error very rapidly and abandons integration.

From the limited number of TEST CK20 RUNS the author has deduced that GEAR still tends to lose the feel of the model and uses unacceptably large steps when caution should be exercised. (See Section 3.3). Firm conclusions require further experimentation however.

5.2.2 TEST CK21

This program comprises 5 differential systems. The first is constructed by the author and has four uncoupled differential equations of the form $\dot{y} = ay$. Cases 2, 3 and 4 are taken from Johnson and Barney (1976) and Case 5 is the one studied in TEST CK20. All these latter cases are coupled systems with varying degrees of stiffness.

Which case is simulated depends on the value of NUR supplied by the user. The initial conditions, constants, and parameters are all read in at run-time. Case 1 (NUR=1) alone is used to test all ACES integrators as well as those of ICL SLAM (ICL SLAM Manual 1974). The decay constants are stacked in Array Y through equivalencing. The UDE PART is similar to that in TEST CK20. The UPA PART outputs both the numerical and the analytical solutions.

In the UTE PART KCN is tested. If KCN is positive, then IRPT is set to NIN to signal a RERUN request without RESET since NIN is positive. The program overrides the UPED, UGINT, and UINTG defaults.

MF is varied from 0 to 5 for ACES integrators and from 11 to 15 for ICL SLAM integrators. The latter are accessed in the UINTG interface which illustrates how a user-supplied algorithm may be inserted.

Table 5.1 displays the solutions at T=1.0. EULR returns the least accurate results but may perform better at smaller step-sizes.

5.3 TEST CK24 : PHYSBE

PHYSBE models a blood circulation system which shows an oscillatory behaviour with a period of 1 second. It has been employed by Korn and Wait (1978) to compare the performance of various simulators but will be used here for illustration only since efficiency is not the primary concern in this chapter.

ACL and ACR are functions of T as listed in the reference. These are set in DATA statements and later used by means of the table look-up function XFNCN. The initial conditions are also provided in DATA statements. The UIN PART calls UAL and returns. The UDE PART is given in Figure 5.4 and is a straightforward translation of the model equations. The program simulates one period only.

MF	Integrator	The STATE at t = 1.0			
0	GEAR	0.90484	0.81873	0.74082	0.67032
1	GEAR	0.90484	0.81873	0.74081	0.67032
2	GEAR	0.90484	0.81873	0.74081	0.67029
3	EULR	0.90461	0.81791	0.73914	0.66761
4	TRPD	0.90484	0.81873	0.74083	0.67034
5	RKD	0.90484	0.81873	0.74082	0.67032
11	TRPZ	0.90484	0.81873	0.74083	0.67034
12	SIMP	0.90484	0.81873	0.74082	0.67033
13	RKFS	0.90484	0.81873	0.74082	0.67032
14	RKVS	0.90484	0.81873	0.74082	0.67032
15	ADMN	0.90484	0.81873	0.74082	0.67032

Table 5.1 - A comparison of integrators using TEST CK21. TRPZ, SIMP, RKFS, RKVS, and ADMN are all ICL SLAM algorithms. The analytical solution is (0.90484, 0.81873, 0.74082, 0.67032).

```
C-----------    U               D               E
 1000 CONTINUE
      IF(JPS.EQ.2)GOTO 1100
C                          FIRST ROUND
      IF(IODE)GOTO 1050
      VAP=RTRV(DVAP)
      VVP=RTRV(DVVP)
      VLV=RTRV(DVLV)
      VAO=RTRV(DVAO)
      VSC=RTRV(DVSC)
      VVC=RTRV(DVVC)
      V-V=RTRV(DV-V)
 1050 PAP=0.138*VAP
      PVP=0.033*VVP
      P_V=XFNCN(T,12,AT,ACL)*VLV
      PAO=0.5*VAO
      PSC=0.0153*VSC
      PVC=0.006*VVC
      P-V=XFNCN(T,12,AT,ACL)*V-V
      FPV=90.0*AMAX1(P-V-PAP,0.0)
      FPS=7.0*(PAP-PVP)
      FVV=17.0*AMAX1(PVP-PLV,0.0)
      FAV=80.0*AMAX1(PLV-PAO,0.0)
      FAS=1.53*(PAO-PSC)
      FVS=1.65*(PSC-PVC)
      FTV=78.0*AMAX1(PVC-P-V,0.0)
      RETURN
C                          SECOND ROUND
 1100 CONTINUE
      CALL FRWRD(VAP,FPV-FPS)
      CALL FRWRD(VVP,FPS-FMV)
      CALL FRWRD(VLV,FMV-FAV)
      CALL FRWRD(VAO,FAV-FAS)
      CALL FRWRD(VSC,FAS-FVS)
      CALL FRWRD(VVC,FVS-FTV)
      CALL FRWRD(V-V,FTV-FPV)
      IF(ISTP.GT.KUL)RETURN
C                    DEBUG SECTION
      IF(KEEP.EQ.1)CALL TVSTT
C-------------    U               P               A
```

Figure 5.4 The UDE PART for TEST CK24.

Korn and Wait (1978) provide a rough plot of two variables, namely PAO and PLV. ACES results agree with this plot. The same results have been obtained on the ICF after some modifications.

5.4 TESTS CK8 AND CK7

The purpose of these programs is to illustrate further the utility of RERUN. Both are taken from Franks (1972). Throughout this chapter examples given by Franks will be referred to as DYFLO examples.

TEST CK8 simulates a steady-state, counter-current tubular heat exchanger. The inlet temperatures are known : TSI on the shell-side and TBI on the tube-side. The outlet temperatures TSO and TBO have to be found. Two alternatives have already been covered (Section 3.1.8) one of which is incorporated into the SEMI-MODULE SSHX (Section 4.7.3). There is also a differential approach which constitutes a boundary-value problem and is adopted in TEST CK8.

The independent variable is X and is equivalenced to T in order to preserve DYFLO's symbology. The differential system is

$$Q = (TS-TB)\ UA$$

$$\frac{d}{dX}\ TB = \frac{Q}{FBCP}$$

$$\frac{d}{dX}\ TS = \frac{Q}{FWCP}$$

The solution procedure involves a marching technique where TS is guessed at the outlet to initiate integration towards the tube-side outlet. When this end is reached (X=200ft) the value of TS is compared with TSI. For a satisfactory answer the difference should be less than CONV, a specified tolerance. Otherwise TSO is updated using TSO=TSO+GAIN(TS-TSI), X is reset to 0 and the scheme is repeated.

The UIN PART calls UAL and also reads in GAIN and CONV if it is the first initialisation (NIN=1). TS and TB are set respectively to TSO and TBI before returning to HSKPIN. The UDE PART conforms to twice-round execution. The UPA PART is traversed only if KUL exceeds 0 or NUR exceeds NIN so that communication can be suppressed optionally. The UTE PART handles convergence and initiates a RERUN if needed.

TEST CK8 uses RKD, GAIN=-0.3, and CONV=0.1 and achieves convergence in three iterations. It has also run on the ICF after some modifications. The results are TBO=95.3^{o}C and TSO=39.8^{o}C compared with DYFLO's 95.3^{o}C and 39.7^{o}C.

This case illustrates the need for an option to suppress the display of integration specifications when these do not change from RUN to RUN.

TEST CK7 involves a similar iteration. It applies the age distribution theory (Franks 1972) to a heterogeneous reaction in a CSTR. The problem is to determine the constant concentration of particles (CA) in the reactor. The total reaction RT and the moles of solid particles

PP are the STATE-VARIABLES. CA is guessed and integration is carried out until T=5 which is deemed sufficient to achieve steady-state. Then CA is updated using a steady-state balance over the reactor and iteration repeated until convergence accrues.

TEST CK7 employs the non-stiff option of GEAR and achieves convergence in 3 iterations. The results are RT=1.1947 mole min^{-1}, CA=0.26105 mole ft^{-3}, and CNV=59.736% corresponding to DYFLO figures of 1.1981, 0.26037, and 59.907. However this case (for which E=0.005) displays some instability as PP becomes slightly negative in some instances. It is thought that smaller values of E may eliminate this. DYFLO simply sets PP=0 when it becomes negative; this is not a satisfactory attitude.

5.5 TEST CK22 : Control on a CSTR Series

This is a case based on an example from Luyben (1973). Figure 5.5 depicts the construction of the plant model using MHLDP, SNSR, and CNTR. The concentration CA3 at the end of a CSTR series is monitored and the inlet concentration CAO is manipulated. The tanks have constant holdups and operate isothermally. The sensor has no delay. The controller is of the PI type and the actuation is instantaneous. The plant is at steady-state initially. CAO=0.4 mole ft^{-3} and CA3 is at the setpoint 0.1 mole ft^{-3}. Then CAO is disturbed to 0.6 mole ft^{-3} by changing CAD and the resultant transient simulated.

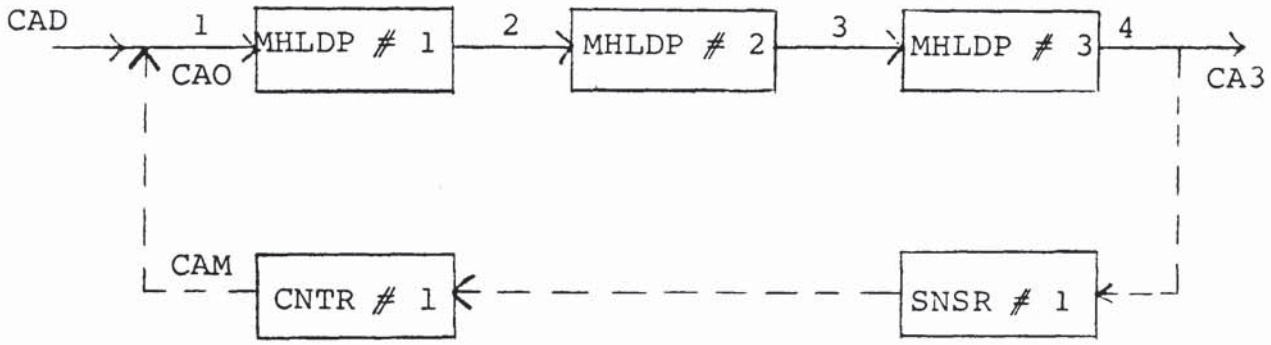


Figure 5.5 The Plant Model for TEST CK22.

The UIN PART attends to initialisation without calling any subprograms. The UDE PART is shown in Figure 5.6. It is different from the previous examples in that it is further compartmentalised into BLOCKS. It should be noted that the MHLDP calls must be in the order displayed if there is a flow change or if RXTM is not zero. This is a consequence of the constant holdup assumption. CAO is updated in the SECOND ROUND.

The UXCNT interface adds the specified bias of 0.4 mole ft^{-3} to the control decision. The reaction term is calculated in the URX interface which is called in the SECOND ROUND by each MODULE.

This case is simulated using the non-stiff GEAR option. Luyben (1973) employs Euler. The results are in good agreement (Table 5.2).

5.6 TEST CK5 : Distillation Control

Based on a DYFLO example this column (Figure 5.7) processes a ternary mixture. All MGPLT holdups are variable but the MHLDP holdup is constant. No reaction is present. SNSR # 1 is a first-order lag and CNTR # 1 is of PI type with instantaneous actuation. This controller manipulates the input of the heating oil to MHTR # 1 which constitutes the reboiler. The plates are assumed adiabatic. Some heat HT is extracted from the partial condenser SPCND. The bottoms flow rate S(21,1) is calculated from

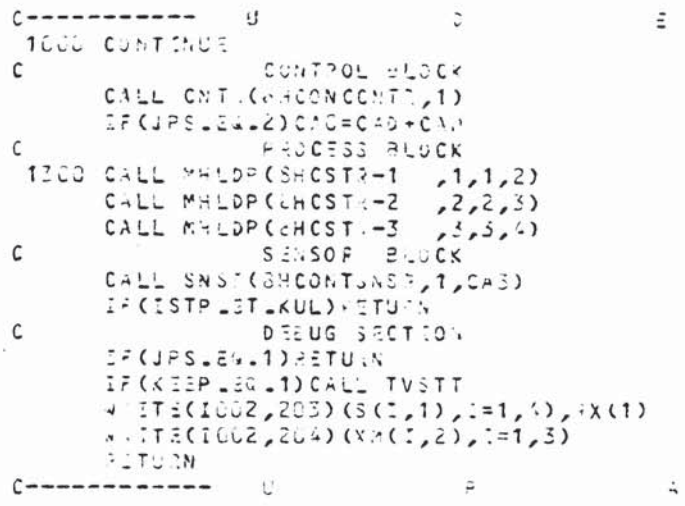


Figure 5.6 The UDE PART for TEST CK22.

Time (min)	CA3(mole ft^{-3})	CAM(mole ft^{-3})
0.0	0.10000 (0.10000)	0.40000 (0.40000)
1.0	0.10195 (0.10194)	0.33804 (0.33836)
2.0	0.10651 (0.10653)	0.17544 (0.17482)
3.0	0.10705 (0.10709)	0.11562 (0.11437)
4.0	0.10315 (0.10314)	0.20087 (0.20104)
5.0	0.10024 (0.10018)	0.27983 (0.28142)
6.0	0.10113 (0.10111)	0.25068 (0.25137)
7.0	0.10313 (0.10317)	0.17740 (0.17612)
8.0	0.10295 (0.10298)	0.16329 (0.16195)
9.0	0.10103 (0.10100)	0.20865 (0.20918)

Table 5.2 - A summary of TEST CK22 results. The figures in brackets are given by Luyben (1973).

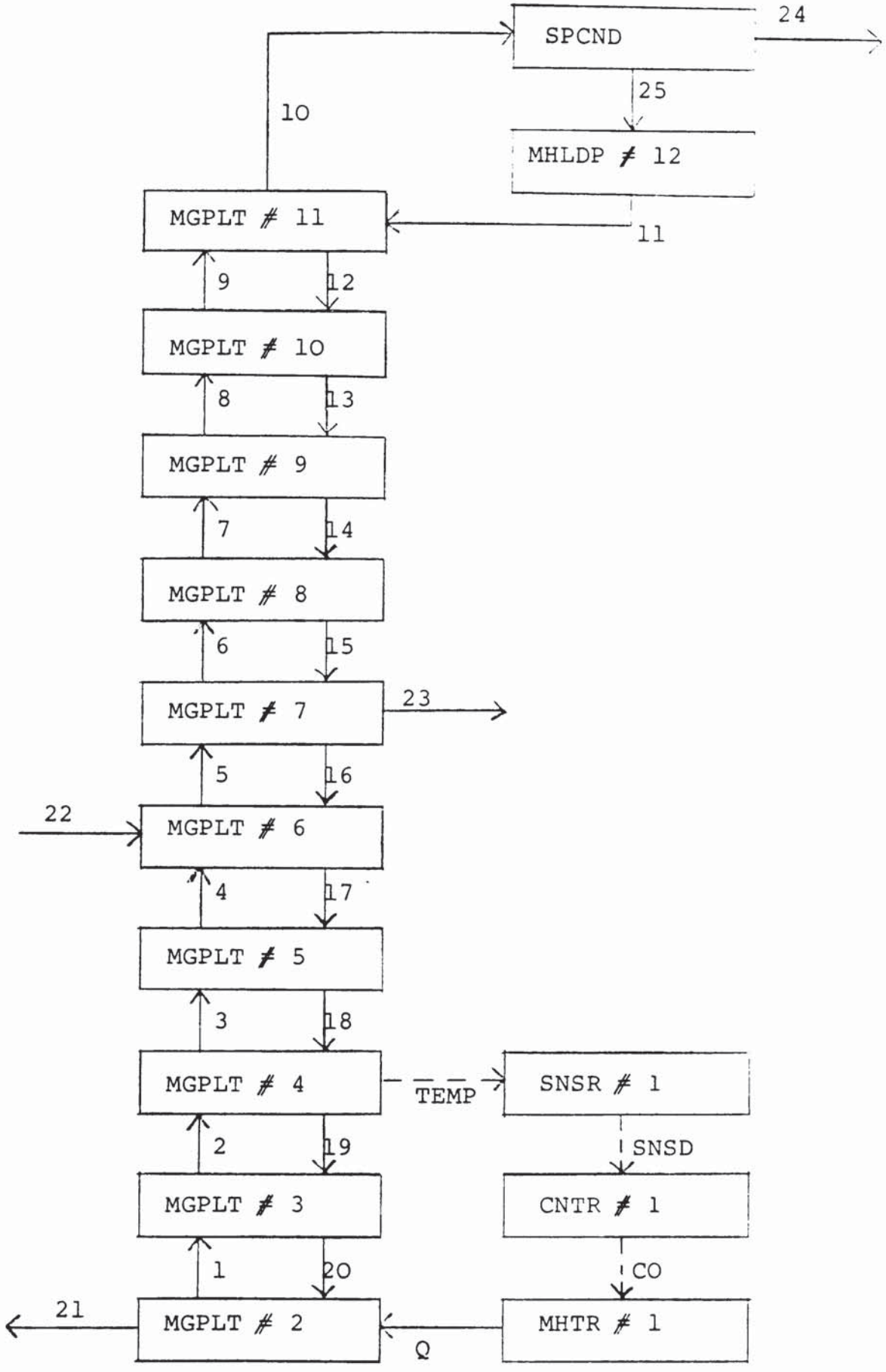


Figure 5.7 The Plant Model for TEST CK5.

S(21,1) = MAX (0,FLSET + (RHL-RHLSET)/RHTC).

This is rather like the estimation of the liquid flowrate leaving any other plate in the column and can be regarded as a simple level controller.

TEST CK5 employs the ROLL-OUT feature where the UIN PART (Figure 5.8.1) matches the UTE PART (Figure 5.8.2). The program is coded to investigate open-loop response if NUR is less than 101. The UDE PART (Figure 5.8.3) will not then register the CNTR and SNSR calls. Thus the UTE PART has to store JCN and JSN in order not to lose the relevant data.

It should be noted that MHTR # 1 precedes its master MGPLT # 2. The relative orientations of MGPLT # 11, SPCND, and MHLDP # 12 are also important because the holdup of the latter MODULE is constant. Other MODULE calls in the UDE PART can be reshuffled without upsetting the information flow.

The UPA PART displays streams 1 to 25 inclusive and also outputs SNSD, CO, A, Q, RHL, DELM, and DELQ. The last two variables denote the mass and enthalpy accumulations in the whole column.

After a number of open-loop simulations a steady-state was obtained to be harnessed as a base for further investigations. CNTR # 1 was brought in with a setpoint equal to the steady-state value of TEMP and two further runs were carried out to verify the steady-state. These

```
      IF(JPS.NE.0)GOTO 999
C------------   U                   I                   \
      CALL UAL(KCN,KMD,KSN)
      READ(IUC1,1UC)TC
      IUC1=5
      CALL ALDATA(2,1,7)
      CALL ALNTS(JCN,JMD,JSN,NSF,NSL)
      IF(JCN.GT.0)CALL ALCNT(KCN,JCN)
      CALL ALMOD(KMD,JMD)
      IF(JSN.GT.0)CALL ALSNS(KSN,JSN)
      CALL ALSSTA(25,1,1,3)
      IUC1=1
      RETURN
  999 GOTO(1000,1000,1500,2000),JPS
C------------   U                   D                   S
```

Figure 5.8.1 The UIN PART for TEST CK5.

```
C------------   U                   T                   S
 2000 CONTINUE
      IF(NUR.LT.101)MSY(42)=JCN
      IF(NUR.LT.101)MSY(47)=JSN
      IUC2=6
      CALL TVDATA(1,4,1,3)
      CALL TVNTS(NSF,NSL)
      IF(JCN.GT.0)CALL TVCNT(KCN)
      CALL TVMOD(KMD)
      IF(JSN.GT.0)CALL TVSNS(KSN)
      CALL TVSSTA(1,25,1,1,3)
      CALL TVSTT
      IUC2=2
      RETURN
  100 FORMAT(4G8.0)
      END
```

Figure 5.8.2 The UTE PART for TEST CK5.

```
C------------   U              D              E
 1100 CONTINUE
      IF(NUM.LT.101)GOTO 1050
C                 CONTROL BLOCK
      CALL CNTL(&mFC-1     ,1)
      IF(JPS.EQ.2)A=0.02*EXP(CO/25.0)
C                 PROCESS BLOCK
 1050 CALL MHTR(&HBOILER-1,1)
      IF(JPS.EQ.2)S(21,1)=AMAX1(0.0,FLSET+(PHL-PHLSET)/HTC)
      CALL MGPLT(&HPLATE- 0,2 ,20   ,IIV,13,21,1)
      CALL MGPLT(&HPLATE- 1,3 ,19   ,1   ,13,20,2)
      CALL MGPLT(&HPLATE- 2,4 ,18   ,2   ,13,19,3)
      CALL MGPLT(&HPLATE- 3,5 ,17   ,3   ,13,18,4)
      CALL MGPLT(&HPLATE- 4,6 ,16   ,4   ,22,17,5)
      CALL MGPLT(&HPLATE- 5,7 ,15   ,5   ,23,16,6)
      CALL MGPLT(&HPLATE- 6,8 ,14   ,6   ,13,15,7)
      CALL MGPLT(&HPLATE- 7,9 ,15   ,7   ,13,14,8)
      CALL MGPLT(&HPLATE- 8,10,12   ,8   ,13,13,9)
      CALL MGPLT(&HPLATE- 9,11,11   ,9   ,13,12,10)
      CALL SPCND(10,25,24,TC,HT)
      CALL MHLDP(&HOLDUP-1,12,25,11)
      IF(NUM.LT.101)GOTO 1100
C                 SENSOR BLOCK
      CALL GNS#(&HTR-    1,1,TEMP)
 1100 IF(ISTP.GT.KUL)RETURN
C                 DEBUG SECTION
      IF(KEEP.EQ.1)CALL TVSTT
      IF(JPS.EQ.2)GOTO 1550
      RETURN
C-----------   U              P              A
```

Figure 5.8.3 The UDE PART for TEST CK5.

employed different step-sizes and simulated 5 minutes of real time. The solutions agreed with each other showing little change in the STATE over this period. Both DELM and DELQ remained negligible compared with the pertinent flows through the column.

The transient response of the column to a 10% increase in the feed flowrate S(22,1) is summarised in Table 5.3. It is observed that the surplus causes both the product flowrates, S(24,1) and S(21,1), to rise. The temperature measurement also increases a little. CNTR # 1 decides to diminish slightly the input of heating oil. After 5 minutes a major part of the transient seems to have decayed already. Both DELM and DELQ are negligible and SNSD is now below the setpoint. CNTR # 1 boosts the input of heating oil and SNSD begins to rise slowly. The STATE at T = 10 minutes is almost steady.

The simulation includes a ROLL-OUT and reactivation at T = 5 minutes. The UXCNT interface checks the bounds of the control decision after the addition of the bias; the lack of warning messages indicates "normal" operation for CNTR # 1. The backward derivative choice in the MGPLT MODULE is utilised throughout. The integrator employed is EULR. The case shows that EULR can give stable and accurate results provided a small enough step-size is selected. The usual criterion of acceptability is little discrepancy between the results of two different step-sizes.

Time min	SNSD °C	CO %	Q PCU min^{-1}	S(24,1) mole min^{-1}	S(21,1) mole min^{-1}	DELM mole min^{-1}	DELQ PCU min^{-1}
0.0	96.55	70.00	21371	16.70	6.30	2.50	22500
	96.55	70.00	21371	16.70	6.30	0.00	16
2.5	96.57	69.63	21353	18.04	7.67	-0.20	1226
	96.55	69.99	21361	16.71	6.29	0.00	16
5.0	96.33	70.98	21965	18.23	7.15	0.08	24
	96.55	69.98	21352	16.71	6.29	0.00	7
7.5	96.36	71.86	22266	18.29	7.22	0.00	132
	96.55	69.97	21345	16.71	6.30	0.00	8
10.0	96.38	72.74	22513	18.32	7.17	0.01	-23
	96.55	69.96	21341	16.70	6.30	0.00	7

Table 5.3 - A summary of TEST CK5 results. The PI controller forces the column towards a new steady-state after a step change in the feed flowrate from 25.0 to 27.5 mole min^{-1}. The setpoint for the measurement (SNSD) is 96.55°C. The secondary lines show how the column behaves in the absence of an upset. Both simulations include a ROLL-OUT and reactivation at TIME=5 min.

Modelling differences are substantial enough to rule out comparison with DYFLO. An actual plant is needed to check on the simulation.

5.7 TEST CK4 : Turbocompressor Control

This is another example taken from DYFLO. A diagram of the model is shown in Figure 5.9. The initial STATE is arbitrarily set. At T=TLATE a load change is imposed (W3=WLATE) and the subsequent transient simulated.

The UDE PART is displayed in Figure 4.3. As in DYFLO ACES utilises TRPD for this case. There are alterations however. The load change is a discontinuity and requires an early termination (when KEEP is 1) followed by a RERUN. This is lacking in DYFLO. The MTRBC MODULE prints a warning message when W1 becomes zero, that is when the compressor is in surge. This is also absent in DYFLO.

The controller GAIN in ACES is related to the DYFLO parameters by GAIN = $\frac{100}{PB}.\frac{100}{SPAN}$ where PB and SPAN are the proportional band of the controller and the span of the measuring instrument.

The interfaces included are UXCNT and UPRBN2. SUBROUTINE UMNFLD is a user-supplied MODULE representing a manifold.

The initial transient up to T=TLATE=5 minutes very closely matches that given by DYFLO. Later however ACES detects a surge at a few instances. This shows the

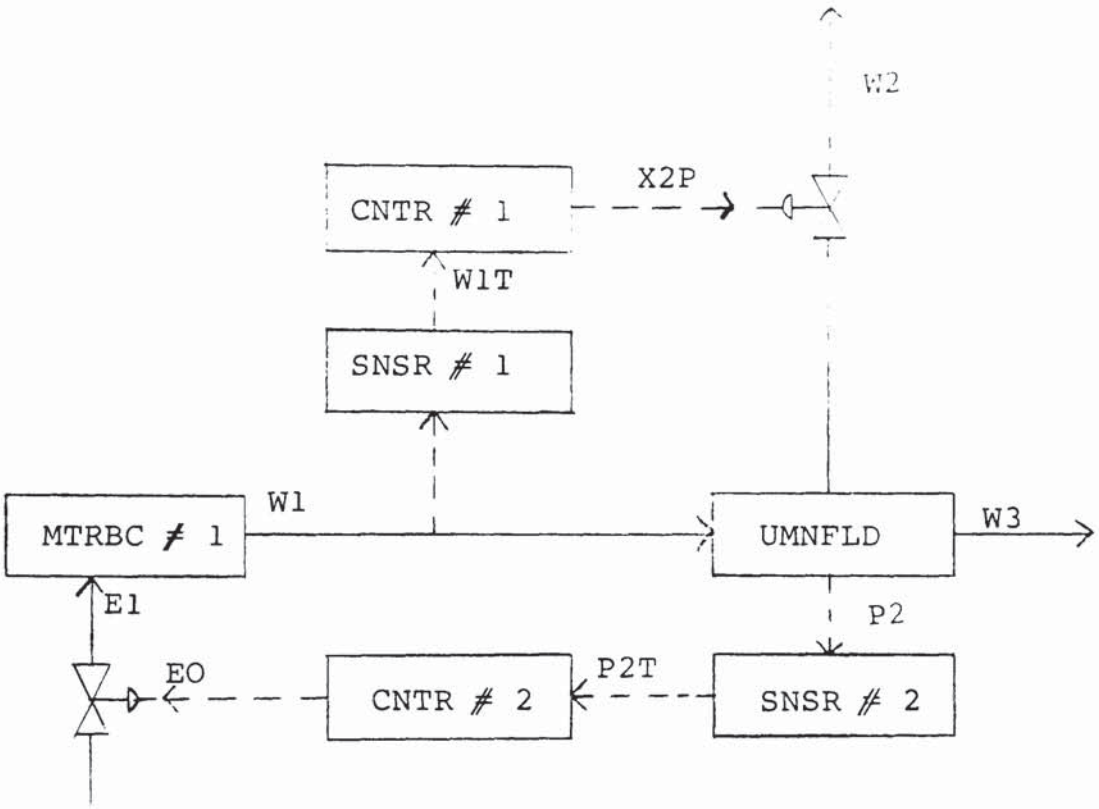


Figure 5.9 The Plant Model for TEST CK4.

importance of built-in checks. The results do not differ substantially. The case is not treated any further.

5.8 TESTS CK3 and CK2

These cases originate in DYFLO; the first simulates a batch reactor and the second a two-stage batch system.

A description of the case in TEST CK3 is as follows. A hydrogenation reaction occurs in the liquid phase of a high pressure autoclave which is operated batch-wise:

$$A + H_2 \longrightarrow B$$
$$B + H_2 \longrightarrow C.$$

In the program Components 1 to 5 are A, B, C, H_2, and the solvent. MHLDP # 1 represents the total reactor holdup. SSLSH finds the split between the liquid and vapour phases which are assumed to be in equilibrium. Cooling coils immersed in the liquid phase are modelled by MHTR # 2. Some heat is lost to the walls of the vessel.

The MHLDP MODULE is converted into batch operation by specifying Stream 2 both as input and as output. In fact streams 2, 3 (vapour) and 4 (liquid) are all holdups. In the UDE PART (Figure 5.10) MHTR precedes its master. The heat lost to the walls leads to an extra differential equation and also necessitates the incrementation of the heat supplied to MHLDP # 1.

Two transfers are essential: S(2,1) is set to the

```
C------------     U               D               E
 1000 CONTINUE
C                 PROCESS         BLOCK
      CALL PHTKC(5HCOOLER   ,3)
        IF(JPS.EQ.2)GOTO 1100
        IF(.NOT.ECON)TW=STEM(DTW)
        GOTO 1200
 1100   HTW=UAW*(TW-S(?,2))
        DTW=-HTW/600.0
        CALL FRWRD(TW,DTW)
        IF(MPX.EQ.1)CALL REACT
        HT=HT+HTW
 1200 CALL PHLDF(5HHOLDUP-1,1,2,2)
      IF(JPS.EQ.1)S(2,1)=H...
      CALL SSUSH(2,4,3,HTS,TRES,SV)
      IF(JPS.EQ.1)XM(1,10)=S(4,2)
      IF(ISTP.GT.KUL)RETURN
C                 DEBUG SECTION
      IF(KDEP.EQ.1)CALL TVSTT
      RETURN
C------------     U               P               A
```

Figure 5.10 The UDE PART for TEST CK3.

total holdup and XM(1,10) to S(4,2). Reactions are coded in SUBROUTINE REACT. This is called by MHLDP through the URX interface if MRX is 2 or by USMLND before MHLDP is entered but always during the SECOND ROUND.

The ROLL-OUT feature is included in the code but not resorted to. The integrator is GEAR (MF=0) as opposed to Euler in DYFLO. The results are in good agreement up to T = 15 minutes but ACES aborts the simulation soon after that since XDWPT needs more iterations than allowed (ITLH=20). This may be a manifestation of numerical instability. A high-level debug report produced in this case is given in Figure 4.6. A summary of results is displayed in Table 5.4.

TEST CK2 simulates the case shown in Figure 5.11. The variable-volume boiler with backward derivatives is selected in MGPLT ≠ 1. The mixture processed is ternary. In DYFLO the still temperature S(2,2) is UNDERDEFINED. The agreement between the results (Table 5.5) is good however.

5.9 TEST CK11 : Dynamic Heat-exchangers

This is to verify SSHX, XICHX, and MDHX. Streams 1 and 4 are the inlet and outlet streams on the shell-side and streams 5 and 8 are their counterparts on the tube-side. There are 3 sections in the exchanger. The data are taken from DYFLO.

Time min	Pressure atm	Temperature °C	C(4,1) mole fraction	C(4,2) mole fraction	C(4,3) mole fraction	C(4,4) mole fraction	S(4,1) mole	S(3,1) mole
0.0	7.56	77.4	0.1840	0.0	0.0	0.4048	23.92	3.08
5.0	13.09	110.6	0.1150	0.0591	0.0252	0.3135	19.49	4.88
10.0	18.56	153.3	0.0068	0.0486	0.1807	0.1099	12.22	6.22
15.0	14.53	145.4	0.0001	0.0156	0.2355	0.0797	12.39	4.96
15.0	(14.65)	(145.8)	(0.0001)	(0.0155)	(0.2352)	(0.0794)	(12.32)	(5.0)

Table 5.4 - A summary of TEST CK3 results. The batch reactor supports a hydrogenation reaction in the liquid phase (≠4). Both Reactant A and hydrogen (components 4 and 1 respectively) are rapidly consumed. Numbers in brackets are DYFLO results.

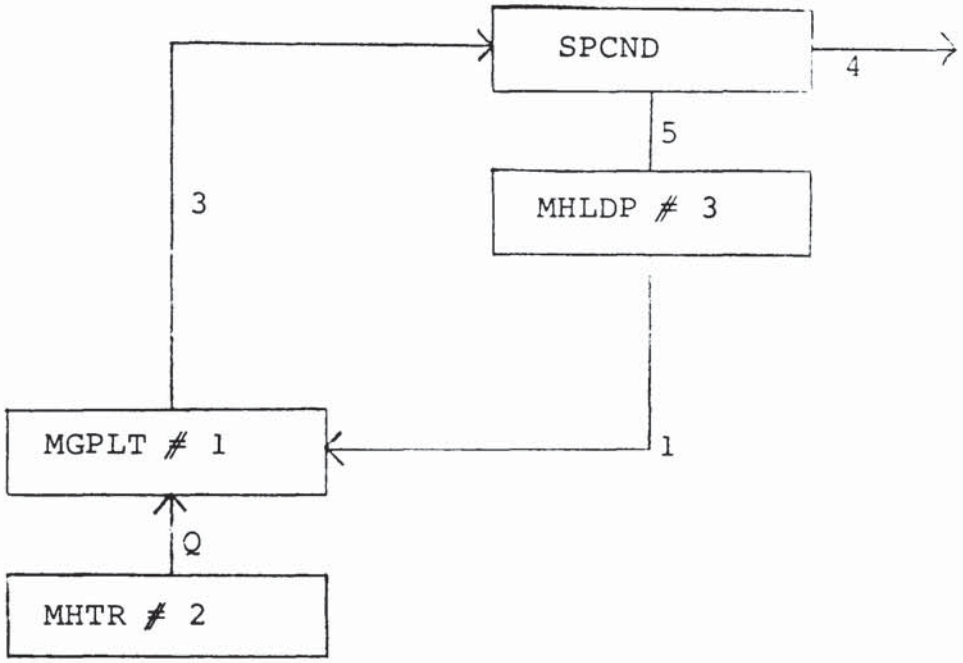


Figure 5.11 The Plant Model for TEST CK2.

Time min	HL mole	S(2,2) °C	C(2,1) mole fraction	S(3,1) mole min^{-1}
0.0	350.0	95.3	0.4300	7.29
	(350.0)	(95.4)	(0.4300)	(7.29)
10.0	318.0	99.6	0.4626	5.21
	(317.4)	(99.7)	(0.4634)	(5.43)
20.0	301.3	102.8	0.4831	4.82
	(300.6)	(102.9)	(0.4837)	(4.80)
30.0	293.2	104.7	0.4938	4.52
	(292.8)	(104.8)	(0.4940)	(4.46)
40.0	289.7	105.7	0.4986	4.35
	(289.4)	(105.7)	(0.4987)	(4.31)
50.0	288.3	106.1	0.5007	4.26
	(288.0)	(106.1)	(0.5007)	(4.25)

Table 5.5 - A summary of TEST CK2 results. HL and S(2,2) are respectively the holdup and temperature of the batch. S(3,1) denotes the boilup rate and C(2,1) is the mole fraction of Component 1 in the still. Figures in brackets refer to DYFLO.

The UIN PART reads in the relevant information and calls XICHX to establish the initial, steady-state temperature profiles. On return a step change is introduced either in S(1,1) (NUR less than 101) or in S(1,2) (NUR greater than 100). MDHX is called in the UDE PART. The UTE PART includes the ROLL-OUT feature but the wall temperatures are not dumped which is essential for reactivation.

The initial flowrates are 50 mole min^{-1} and 30 mole min^{-1} on the shell-side and on the tube-side respectively. The inlet temperatures are S(1,2)=20^{o}C and S(5,2)=120^{o}C. Called by XICHX, SSHX returns the outlet temperatures as S(4,2)=54.28^{o}C and S(8,2)=59.97^{o}C. The call to XICHX seems to corrupt S(5,2) slightly to 119.97^{o}C. This will probably vanish as the number of sections increases. The transient depicted in Table 5.6 is the result of a step change in S(1,1) to 100 mole min^{-1}. At TIME=40 minutes a steady-state is apparent. No oscillation is observed. As expected the intermediate and outlet temperatures all diminish. The accuracy of the dynamic solution cannot be ascertained however.

5.10 TEST CK10 : Dead-time

This program is constructed to investigate the dead-time routines XDTF and XDTR as well as the backward differentiator XDRVT.

Time (min)	S(2,2) (°C)	S(3,2) (°C)	S(4,2) (°C)	S(6,2) (°C)	S(7,2) (°C)	S(8,2) (°C)
0.0	27.97	41.12	54.28	96.79	76.95	59.97
10.0	25.14	31.75	40.07	96.46	74.60	58.63
20.0	24.89	31.18	39.22	93.14	72.66	56.91
30.0	24.75	30.91	38.87	92.75	71.93	56.04
40.0	24.69	30.80	38.74	92.62	71.66	55.68

Table 5.6 - A summary of TEST CK11 results: The transient behaviour of the heat-exchanger following a step change in the shell-side flowrate from 50.0 to 100 mole min^{-1}.

$X=0.5\ T^2$ and DX=T are respectively delayed for 10 and 8 steps and then returned in XL and DXL. Thus $XL=0.5\ (T-10HLL)^2$ and DXL= T-8HLL where HLL is the integration step-size.

The program regards X and DX as pseudo STATE-VARIABLES and "integrates" them in the UINTG interface. Difference equations can be similarly projected. Dead-time is implemented by XDTF and XDTR calls coupled together. XDRVT finds the backward derivative of XL with respect to T. DRVT is set to XDRVT and should equal 0.5(2T-21HLL). Initial values are user-supplied. The UPA PART outputs T, ZL, XL, DZL, DXL, and DRVT where ZL and DZL denote the analytical values for XL and DXL.

Two RUNS were made each lasting 20 time units. The first RUN assumed the initial dead-time profiles in Array YIC (NUR=1). Special attention must be paid to the ROLL-OUT of Array YA as the latest retrievals will have been overwritten. In this RUN they happened to occupy the first locations in store and so were easily recovered before the dump. The second RUN reactivated this STATE and continued the simulation.

The solutions are presented in Table 5.7. They are confirmed exactly by the analytical expressions.

5.11 What remains untested

Certain features have not been verified in this work. The empiric approach in MGPLT and the PID controller

T	XL	DXL	XDRVT
0.0	0.00	0.00	0.00
5.0	8.00	4.20	3.95
10.0	40.50	9.20	8.95
15.0	98.00	14.20	13.95
20.0	180.50	19.20	18.95
25.0	288.00	24.20	23.95
30.0	420.50	29.20	28.95
35.0	578.00	34.20	33.95
40.0	760.50	39.20	38.95

Table 5.7 - A summary of TEST CK10 results. The analytical expressions for XL and DXL are XL=0.5(T-10HLL) and DXL=T-8HLL where HLL is 0.1. XDRVT is the backward derivative of XL with respect to T and should equal 0.5 (2T-21HLL). All initial values are user-supplied. A ROLL-OUT occurs at T=20.0 followed by reactivation. The numerical values match their respective equations exactly.

in CNTR may be cited as examples. XDFSB1 has the same logic as XDFSB2 but incorporates far fewer WRITE statements; it does not require special testing. SCVBL, SSMMR, SSPLT, CCNT, XMRG3, XMWGS, XSDTF, XSDTR, and XVLV are all straightforward and can be verified by inspection. A suitable benchmark problem was not available to test the MLXTR MODULE. In the absence of analytical solutions it would be desirable to compare numerical solutions with published cases and the author attempted to do just that in the verification phase.

5.12 Discussion and Conclusions

This chapter has demonstrated the principles of ACES as well as its utility. It is evident that ACES has the capability to simulate a wide range of problems without involving the user in housekeeping chores.

The coding of differential systems is straightforward enough with as little demarcation in USMLND as in an EXPLICIT program for ICL SLAM. A variety of integrators are provided. TEST CK21 verifies these and shows how easily a user algorithm can be inserted. TEST CK10 indicates a way of formulating difference schemes. As exemplified by TEST CK8 the independent variable can be distance or another variable instead of time.

Twice-round execution facilitates a high degree of modularity. Compartmentalisation of USMLND into BLOCKS/SUBBLOCKS helps to achieve simultaneity in a

modular chemical engineering model. This is confirmed by TESTS CK22 and CK5 for instance.

The RERUN feature is demonstrated by TESTS CK8 and CK7. The feature can be harnessed for various ends including the solution of boundary-value problems, parametric studies and optimisation.

Communication is easy and convenient in ACES. The novel facility of ROLL-OUT is an asset. It enables simulations to be linked and provides standard, readable documentation. TEST CK5 relies on ROLL-OUT for piecewise simulation.

A tenet of ACES is robustness. DYFLO incorporates much arbitrariness and numerous inconsistencies ruled out by ACES. The latter is further provided with built-in checks and a debug reporter. This feature proved indispensable in initial trials especially. Its importance is evident in TEST CK4 where a surge condition is so detected and the user warned.

ACES is flexible and versatile. A model can be expanded by changing the number of chemical components or delay orders for instance without needing to recode the program. TESTS CK2 and CK3 show that batch processes can be simulated as easily as continuous ones. Integrators and integration specifications may be altered without recompilation. Furthermore ROLL-OUT makes it possible to inspect and manually tune the total data to physical realities.

The successful RUNS on the ICF machine indicate a large degree of portability (Appendix 6). One impediment in this area seems to be lack of generality in the number of alphanumeric characters that a machine stores per word. ACES assumed 8-characters per real variable which is acceptable on the ICL 1904S.

CHAPTER 6 DISCUSSION, CONCLUSIONS, AND RECOMMENDATIONS

Computers play a useful role in chemical engineering. On the one hand they are employed to predict and achieve the optimal point of operation for a chemical plant. On the other hand they help elucidate via simulation various aspects of the industry. Chapter 2 shows that transients are prevalent in most processes and that they must be considered early in the design stage. There exist numerous dynamic simulation packages but the literature survey revealed the need for a new dynamic simulator with a chemical engineering bias. The indications were that such a package should be (i) general, (ii) flexible, (iii) versatile, (iv) robust, (v) portable, and (vi) convenient to use. It was found that modularity and file-orientation were essential features to achieve these qualifications. The housekeeping chores should be carried out behind the scenes as far as possible thus minimising the user's detailed involvement.

The starting point for ACES was DYFLO (Franks 1972), ICL SLAM (ICL SLAM Manual 1974), and DIFSUB (Gear 1971). ICL SLAM has a very flexible structure for a general purpose simulator but lacks chemical engineering modules and is also restricted to ICL machines. DYFLO provides a suite of these on a sequential framework which often violates simultaneity. DIFSUB is a well-known integration code capable of handling stiff differential equations and as such constitutes a useful simulation tool. The structure

of ICL SLAM was adopted for ACES and the device of twice-round execution was conceived in order to achieve simultaneous modularisation. DYFLO and DIFSUB were tailored to suit the new structure. Inconsistent and arbitrary features of DYFLO were excluded. ACES was further strengthened by a series of built-in checks coupled to a debug report interface. Generalisation was preferred to sophistication.

ACES performs substantial housekeeping out of sight of the user and documents the user model for future reference. Communication subprograms are provided for various categories of plant data. The compatibility among these subprograms enables the user to save a STATE in a readable form and harness it at a later date. This capability constitutes the ROLL-OUT feature in ACES.

The user may furnish the plant model as a set of equations. Alternatively the library subprograms may be employed to construct the model with interspersed equations or calls to user-supplied subprograms. STATE-VARIABLES must be retrieved in the same order as they are forwarded to the ACES SIDE. Retrievals in the FIRST PASS are spurious. The library subprograms require further structuring in USMLND for the sake of simultaneity. Nevertheless the demarcation left to the user remains minimal. Few restrictions are imposed on the relative orientation of calls within a BLOCK or SUBBLOCK.

Modularity renders ACES flexible and versatile. It also brings the model closer to the flowsheet. The CCNT, CNTR, SNR, and MODULE calls carry a label which leads to easy recognition. The CONTROLLER and SENSOR BLOCKS are rather reminiscent of the control room in a plant.

ACES can handle boundary-value type problems as well as initial-value type since it allows RERUN with or without RESET. This feature also facilitates optimisation.

On the 1904S, the ACES library is stored as a semi-compiled file which includes default interfaces that can be overridden. Other machines probably allow a similar arrangement.

Chapter 5 verifies the fundamentals of ACES. Simultaneous modularisation through twice-round execution is thoroughly demonstrated. RERUN and ROLL-OUT both function as described and the integrators return correct results. The engineering subprograms produce outputs comparable to reference outputs. Experimentation with actual plant data remains desirable however in order to verify some of the dynamic solutions in this work.

Built-in checks proved their worth in the verification phase. Mis-specifications were intercepted on several occasions which would otherwise have resulted in wastage to say the least. Numerical instabilities were trapped before they caused an execution error in unexpected places. In all cases the debug device identified the failure location exactly.

Differences exist in FORTRAN implementations on different machines. Portability must be achieved despite these differences. For example the 1904S provides zero defaults for uninitialised variables but the ICF aborts compilation. This sometimes helps to find spelling mistakes. The ICF approach is preferable since it is more robust. Another peculiarity of the ICF is in its carriage control : it does not allow field specifications in free FORMATS.

GEAR needs further study. Its main problem seems to be loss of feel of the model. It may be suffering from round-off errors. Two things are apparent in the current code. First, it changes the order even if an increase in step-size is limited by the maximum. This is undesirable. Secondly, decisions are made and implemented at the end of step. The implementation should be postponed until the next entry in case the user decides to use a step size other than selected by GEAR. This requires a slight modification of the code.

Discontinuities disrupt smooth integration. The author regards RERUN as a good way of introducing discontinuities into the plant model. This is possible when the user is aware of them. Certain scheduling operations may constitute a discontinuity. It is evident then that numerous RERUNS may be needed in a combined simulation study.

ICL SLAM is a general purpose simulator. DYFLO and DYNSYS may be considered to belong to a first generation

modular simulators addressing chemical engineering in particular. ACES probably marks the beginning of a second generation in dynamic simulation where simultaneous modularisation is achieved.

The contribution of this work is manifold. The first and foremost is the development of a flexible, versatile, and easy to use modular dynamic simulator. Secondly, the need for an integrated approach has been appreciated and steps have been taken in that direction. ACES is accompanied by a consistent methodology which aids unification. ACES includes interfaces and ROLL-OUT to link it to other packages. This work has also established new guidelines for deriving and presenting conservation models. Certain avenues such as pseudo steady-state approximations have been ruled out. Furthermore, convergence of iterative schemes has been investigated and a novel, general promotion technique devised.

This research has laid the groundwork for further expansion. Doubtless there are many openings for future investigations. For instance the addition of interaction and graphics would considerably enhance the simulator. Scheduling may be incorporated. The library may be enriched by new models. On-line applications are desirable.

The conclusions are as follows:

(1) As a digital dynamic simulator ACES is flexible, versatile and easy to use.

(2) Simultaneous modularisation has been achieved.

(3) ACES is robust and portable.

(4) ACES can handle both initial-value and boundary-value cases. RERUN is included.

(5) ROLL-OUT facilitates piecewise simulation.

(6) ACES can be linked to other packages.

(7) New models can be appended to the library.

(8) Among the contributions of this research are a novel convergence promoter and guidelines for deriving and presenting conservation models.

(9) RERUN can be employed to introduce discontinuities into the plant model.

The author's recommendations for future work can be itemised as follows:

(1) The addition of interaction and graphics should be investigated in the next phase of development.

(2) On-line applications may be attempted.

(3) The chemical plant models may be tested further by using actual plant data.

(4) Scheduling may be provided as a feature.

APPENDIX 1 A NOVEL CONVERGENCE PROMOTER

An iterative algorithm executes the steps below until two successive iterates become equal subject to a tolerance:

$$r_n = F(x_n) \ ,$$

$$x_{n+1} = r_n$$

Such schemes are widely used in solving algebraic equations and optimisation for instance. A new method has been devised by the author to promote the convergence of these numerical procedures.

Let $z_n = \frac{r_n - x_n}{x_n + d}$ where d is a number to be found by fitting a straight line to three points (x_n, z_n) from the original algorithm. The x-intercept of this line should identify with the solution since z=0 gives $r_n = x_n$. Now the slope is given by $\frac{z_1 - z_2}{x_1 - x_2} = \frac{z_2 - z_3}{x_2 - x_3}$. This equality yields $z_1 - (1+b)\, z_2 + b z_3 = 0$ where $b = \frac{x_1 - x_2}{x_2 - x_3}$. After some manipulations a quadratic is finally obtained of the form $a_1 d^2 + a_2 d + a_3 = 0$ where

$$a_1 = h_1 - h_2\,(1+b) + h_3 b,$$

$$a_2 = h_1\,(x_2+x_3) - h_2(1+b)(x_1+x_3) + h_3 b(x_1+x_2),$$

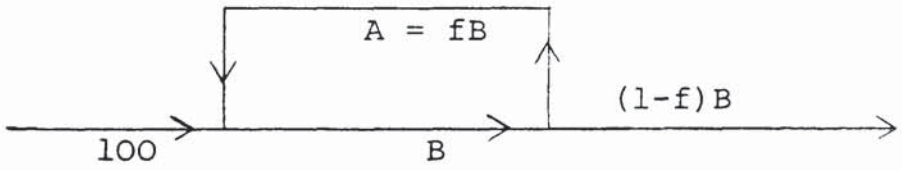
and $$a_3 = h_1 x_2 x_3 - h_2(1+b)\, x_1 x_3 + h_3 b x_1 x_2.$$

Here $h_n = r_n - x_n$. Hence d may be calculated.

In practice, the line so produced is parallel to the x-axis. However the eventual solution becomes equal to $-d_1$. The method is modified to use $-d_1$ as the next approximation to the solution and the iterations are continued.

A subprogram has been coded in FORTRAN to apply the promoter to find a real root of an algebraic equation. If the roots of the quadratic are complex, then the real part is taken. If two real roots are produced, then the one closer to the last x value is selected. The program employs partial substitution in the first two iterations. After the third iteration the worst pair (x_n, r_n) is overwritten. A test program is given by Kocak (1980a).

The first test tried is based on a recycle problem from Crowe et al. (1969). A diagrammatic presentation is as follows.



A fraction f of stream B is recycled back. The balance equations are B = 100 + A and A = fB. The solution is via direct substitution:

$$r_n = f(100+A_n)$$

$$A_{n+1} = r_n \ .$$

Without any promotion the convergence is steady but very

slow. The promoter proceeds through (x_n, h_n) pairs of (0,90), (90,81), (171,72.9), and (900,0.14901E-07), and decides that the solution is 900 which indeed it is, for f = 0.9.

The second test is based on DYFLO (Franks 1972). The problem is to determine the volume of a gas using Beattie-Bridgeman equation. Again direct substitution is resorted to:

$$r_n = (RT + \frac{BT}{V_n} + \frac{GM}{V_n^2} + \frac{DEL}{V_n^3})/P$$

$$V_{n+1} = r_n \ .$$

The steps are (0.93727, -0.27172), (0.66555, -0.93445E-01), (0.57210, -0.47962E-01), (0.44776, -0.82275E-02), (0.40469, -0.14954E-02), (0.38969, -0.79013E-04) and (0.38867, 0.26540E-06). The solution is then indicated as 0.38867. The improvement is remarkable, as is shown from the fourth iterate onwards. The promoter has also been used in conjunction with Newton-Raphson in this case.

APPENDIX 2 DEFINITIONS OF ACSL VARIABLES

Variable	Detail
A	GEAR array for corrector coefficients.
AC	GEAR array for cumulative corrector corrections.
AB	Alphanumeric label for a CONT,CNTR,SAVE,or MODULE call.
BAGSIZ	Alphanumeric independent variable.Defaulted to TIME in HSKPSI.
BOS	Alphanumeric blank.Set in HSKPSI.
C	Array for stream mole fractions.See APPENDIX 4.
CHZ	Alphanumeric underliner.Set to -------- in HSKPSI.
CN	CNTR storage array.See APPENDIX 4.
CNVR	Universal base for relative tolerance.Defaulted to 0.001 in HSKPSI.
CNVE	Universal convergence tolerance.Defaulted to 0.0.1 in HSKPSI.
CNVER	Convergence tolerance for equilibrium.Defaulted to 0.001 in HSKPSI.
CP	GEAR array for corrector correction.
D	Physical properties array.See APPENDIX 4.
DATE	Alphanumeric date.Defaulted to blank in HSKPSI.
DFGN	NSPLT gain for the empiric option.Defaulted to 0.01 in HSKPSI.
DFHL	Minimum holdup.Defaulted to 0.0001 in HSKPSI.
DFP	Minimum pressure.Defaulted to 0.01 in HSKPSI.
DFTH	Maximum temperature.Defaulted to 2500.0 in HSKPSI.
DFTL	Minimum temperature.Defaulted to -300.0 in HSKPSI.
DJ	GEAR array for numeric Jacobian estimation.
DT	T differential over the last step.Use when JEEP=1 only.
E	Integration error specification for variable-step methods.Defaulted to 0.05 in HSKPSI.
FD	Inverse weight for 'lower' order in GEAR.Defaulted to 1.0 in HSKPSI.
FJ	Used in numerical Jacobian estimation in GEAR.
FS	Inverse weight for 'same ' order in GEAR.Defaulted to 1.0 in HSKPSI.
FU	Inverse weight for 'higher' order in GEAR.Defaulted to 1.0 in HSKPSI.
FX	Inverse limiter for step-size increases in GEAR. Defaulted to 0.3 in HSKPSI.
FZ	Used in Jacobian estimation in GEAR.
H	The current step-size.SINT sets H to HX for RUNGE, TRPD,and RKD.SINT sets H to HN for GEAR at COMMUNICATION TIMES.GEAR may then vary H.
HD	GEAR array for higher-order derivatives.
HN	The minimum step-size.Calculated in SINT at COMMUNICATION TIMES as HN=PRI/MX.
HX	The maximum step-size.Calculated in SINT at COMMUNICATION TIMES as HX=PRI/MN.
I	Auxiliary steps for non-selfstarting integrators.
INK	The INTERFACE MARKER.
INKT	JDV on entry to a MODULE.

IM	The MONITOR FLAG.See Section 4.4.
IMAX	Maximum number of corrector iterations in GEAR. IMAX must exceed 2.
IPERM	Permutation array set by XLUSLV and used in XSOL.
IRPT	The RERUN FLAG.See Section 4.2.
ISTP	The STEP COUNTER.HSKPIN sets ISTP to 0 before calling USRLND.HSKPDE increments it by 1 before calling USRLND if KEEP is 1.ISTP is reset to 1 if it exceeds ISTPL.
ISTPL	Upper limit on ISTP.Set to 5 300 000 in HSKPSI.
ITLM	Limit on the number of iterations per algebraic loop.Defaulted to 20 in HSKPSI.
ITRM	The TERMINATION FLAG.See Section 4.2.
IOO	Array for communication channels.See IO08 and IO09.
IOO1	Input channel.Defaulted to 1 in HSKPSI.
IOO2	Output channel.Defaulted to 2 in HSKPSI.
IOO5	The PASS FLAG.IOO5=.TRUE. marks the FIRST PASS. HSKPIN sets IOO5 to .TRUE. before calling USRLND. HSKPPA switches it to .FALSE. after calling USRLND.
IOO6	The COMMUNICATION FLAG.IOO6=1 indicates a COMMUNICATION TIME.HSKPIN sets IOO6 to 1 before calling USRLND and the integrator maintains it thereafter.
IOO8	Output channel equivalent to IOO(1).Defaulted to 8 in HSKPSI.
IOO9	Output channel equivalent to IOO(2).Defaulted to 9 in HSKPSI.
JCML	Limit on the number of chemical components.Set to 10 in HSKPSI.
JCN	The CNTR COUNTER.See Sections 4.3.3 and 4.3.7.
JCNL	Limit on JCN.Set to 20 in HSKPSI.
JCNS	The JCN TOTAL.See Section 4.3.3.
JD	The XDOVT COUNTER.See Section 4.3.3.
JDL	Limit on JD.Set to 10 in HSKPSI.
JDS	The JD TOTAL.See Section 4.3.3.
JDV	The STATE-VARIABLE COUNTER.Maintained by HSKPDE.
JDVL	Limit on JDV.Set to 100 in HSKPSI.
JDVS	The JDV TOTAL.See Sections 4.3.3 and 4.3.5.
JEEP	The DIFFERENTIATION FLAG.Backward differentiation is allowed at the end of a step (JEEP=1).HSKPIN sets JEEP to 0 before calling USRLND.HSKPDE couples it to KEEP during LATER PASSES.
JGT	The stage marker in GEAR.
JLF	The DEAD-TIME FORWARDING COUNTER.See Section 4.3.3.
JLT	The DEAD-TIME RETRIEVAL COUNTER.See Section 4.3.3.
JLS	The DEAD-TIME TOTAL.See Section 4.3.3.
JMD	The MODULE COUNTER.See Sections 4.3.3 and 4.3.7.
JMDL	Limit on JMD.Set to 50 in HSKPSI.
JMDS	The JMD TOTAL.See Section 4.3.3.
JPS	The ROUND NUMBER.ACES adjusts it to -1,0,1,2,3,or 4 to mark integration,the INITIAL REGION,the FIRST ROUND,the SECOND ROUND,the PARALLEL SECTION,or the TERMINAL REGION.
JSN	The SNSV COUNTER.See Sections 4.3.3 and 4.3.7.

JSNL	Limit on JSN.Set to 30 in HSKPSI.
JSNS	The JSN TOTAL.See Section 4.3.2.
JSTART	GEAR variable.
JSTL	Limit on stream index.Set to 30 in HSKPSI.
JWM	See Section 4.2.1 for example.
KNT	The STACK COUNTER.See Sections 4.3.1,4.3.7,and 4.3.8.
KNTST	The CALL RECORD.See Sections 4.3.1,4.3.7,and 4.3.8.
KEEP	The FULL-STEP FLAG.KEEP=1 marks the end of an integration step.Set to 1 in HSKPIN and maintained by the integrator thereafter.GEAR sets KEEP to 5 for numerical Jacobian estimation.GEAR will repeat a step if the user sets KEEP to -1.
KTP	The SUBPROGRAM TAG.See Section 4.3.2.
KUL	The TRACE STEP FLAG.Defaulted to 0 in HSKPSI. See Section 4.2.3.
LDJ	Limit on DJ storage.Set to 200 in HSKPSI.
MDC	CNTR flag array.See APPENDIX 4.
MDS	SAVE flag array.See APPENDIX 4.
MF	The METHOD CHOICE FLAG.Defaulted to 3 in HSKPSI. Set to 0,1,2,3,4,or 5 for GEAR (non-stiff),GEAR (analytical Jacobian),GEAR (numerical Jacobian), EULER,TRPD,or RKDAS values between 4 and 10 are reserved for future ODES integrators.MF=11+ flags a user-supplied algorithm.
MFC	The METHOD TYPE FLAG.0 for fixed-step methods.See Section 4.2.
MM	MODULE flag array.See APPENDIX 4.
MN	The MINIMUM NUMBER OF STEPS per PRJ.Defaulted to 10 in HSKPSI.
MS	Stream flag array.See APPENDIX 4.
MSY	System integer array.
MX	The MAXIMUM NUMBER OF STEPS per PRJ.Defaulted to 100 000 in HSKPSI.
NC	GEAR corrector steps.
NIN	The INITIALIZATION NUMBER.Set to 0 in HSKPSI and incremented by 1 in HSKPIN.
NJ	GEAR Jacobian counter.
NNTR	GEAR entry counter.
NP	GEAR predictor steps.
NQN	GEAR order usage array.
NS	GEAR successful step counter.
NUM	The PROJECT RUN NUMBER.Defaulted to 1 in HSKPSI.
PERTST	GEAR order selection array.
PJ	The Jacobian array for the model.See Section 4.4.
PRT	The COMMUNICATION INTERVAL.Defaulted to 1.0 in HSKPSI.
PROJ	Alphanumeric project title.Defaulted to blank in HSKPSI.
RX	Array for molar reaction rate terms for chemical components.
RXTM	The total molar reaction rate term.
RXTQ	The total heat generation rate term due to reactions.
S	Stream array.See APPENDIX 4.
SAVE	GEAR safety storage array.

SAV	AD for the last labelled call.Defaulted to blank in MSKPDE before calling USRLND.See Section 4.3.7.
SD	GEAR scaled derivative array.
SN	SAVE storage array.See APPENDIX A.
SY	System storage array.
T	The independent variable.Defaulted to 0.0 in MSKPSI.
TFIN	The FINISH POINT.Defaulted to 1.0 in MSKPSI.
TL	T at the beginning of a step.Use when JTEP=1 only.
USER	Alphanumeric to identify the user.Defaulted to blank in MSKPSI.
VRSN	Alphanumeric to identify the version of ACES.Set in MSKPSI.
XD	The STATE derivative vector.
XI	The initial STATE vector.
XM	MODULE storage array.See APPENDIX A.
XX	The STATE vector.
YES	GEAR minimum base for relative tests.
YMAX	GEAR array used in relative tests.

APPENDIX B A CLASSIFICATION OF THE ACED LIBRARY

1. Housekeeping :

   DINLP, FRWID, GINT, HSKPDE, HSKPEX, HSKPIN, HSKPPA, HSKPSI, HSKPTE, KATCH, KOLCH, and STPV.

2. Integrators :

   CULR, GEAP, FKD, and TRPD.

3. Clearance :

   ZDT, ZPCT, ZSTC, and ZXIXD.

4. Readers :

   ALCNT, ALDATA, ALMOD, ALNTG, ALSNS, ALSSTS, ALSTT, and ALSVA.

5. Display :

   TVCNT, TVDATA, TVMOD, TVNTG, TVSNS, TVSSTS, TVSTT4, TVSTT, and TVSVA.

6. MODULES :

   MDHX, MGPLT, MHLDP, MHTR, MLXTS, and MTIBC.

7. SEMI-MODULES :

   SCVBL, SFLSH, SHTX, SPCND, SSHX, SSLSH, SSMP, and SSPLT.

8. SENSORS and CONTROLLERS :

   CCNT, CNTP, and SNSP.

9. Default interfaces :

   UACTY, UCNT, UCRS, UGINT, UINTG, UPED, UPI3N2, UPIX, UTFN, and UXCNT.

10. Miscellaneous :

   XACT1, XBBAL, XDFCO, XDFSE1, XDFSE2, XDIVT, XDTF, XDTP, XDWPT, XFNCN, XHLD, XICPX, XLUSLV, XLWGS, XMPGD, XMWGS, XNTBL, XNTHV, XPABCT, XPSES, XPNWT, XSDTF, XSDTP, XSOL, XSTMP, XTFN1, XTFN2, and XVCV.

APPENDIX 4 THE STORAGE ARRANGEMENTS

MODULE, CNTR, and SNSR variables marked with an asterisk must be initialized if applicable to the option.

1. The storage for the Ith CNTR :

Location	Name	Detail
CN(I,1)	AD	Alphanumeric label.
* CN(I,2)	DCS	The actuated response.
CN(I,3)	CSG	The decision(control signal).
* CN(I,4)	SNSD	The input(measurement).
* CN(I,5)	SP	The setpoint.
* CN(I,6)	CG	The controller gain
* CN(I,7)	X	The action flag.-1.0,0.0,or 1.0 for reverse,nil,or direct action
* CN(I,8)	PE	See Section 4.5.2.
* CN(I,9)	PED	See Section 4.5.2.
* CN(I,10)	TPT	See Section 4.5.2 and Franks(1972).
* CN(I,11)	TT	See Section 4.5.2 and Franks(1972).
* CN(I,12)	GA	See Section 4.5.2 and Franks(1972).
* CN(I,13)	VG	The valve gain.
* CN(I,14)	D	See Section 4.9.2.
* CN(I,15)	TTNV2	TT or TNV2.See Section 4.9.2.
* CN(I,16)	EBT	See Section 4.9.2.
* MDC(I,1)	IT	The controller type flag.(Section 4.5.2)
* MDC(I,2)	IS	The sensor index.See Section 4.5.2.
* MDC(I,3)	ID	The actuation delay order.(Section 4.5.2)
MDC(I,4)	MCO	The CNTR call order.Maintained by ACES.

ACES allows 20 CNTR calls only.CN(I,17) to CN(I,20) are spare.

2. The physical properties of the Nth component

Location	Name	Detail
D(N,1)	A1	Antoine coefficient.(Section 3.1.6.)
D(N,2)	A2	Antoine coefficient.(Section 3.1.6.)
D(N,3)	A3	Antoine coefficient.(Section 3.1.6.)
D(N,4)	AV	Vapour enthalpy coefficient A.(Section 3.1.5.)
D(N,5)	BV	Vapour enthalpy coefficient B.(Section 3.1.5.)
D(N,6)	VLT	vapour enthalpy coefficient c See Section 3.1.6.
D(N,7)	AL	Liquid enthalpy coefficient A.(Section 3.1.5)
D(N,8)	BL	Liquid enthalpy coefficient B.(Section 3.1.5.)
D(N,9)	GAMMA	Activity coefficient.

ACES allows 10 chemical components.D(N,10) to D(N,15) ARE SPARE.

3. The storage for the Ith Stream :

Location	Name	Detail
S(I,1)	TMF	The total molar flow rate.
S(I,2)	TST	The temperature.
S(I,3)	EST	The molar enthalpy.
S(I,4)	PST	The pressure.
S(I,5)	SBB	The bubble-point memory.
S(I,6)	SDW	The dew-point memory.
S(I,7)	SP	The split memory for SFLSH or SPCND.
S(I,8)	SPIC	The split memory for SSLSH.
S(I,9)	TW	The sectional wall temperature in MDHX.
MS(I,1)	MF	The first component index.
MS(I,2)	ML	The last component index.
MS(I,3)	MP	The phase index.1,2,or 3 for solid,liquid ,or vapour.
MS(I,4)	MA	The activity flag.MA=1 for ideal.
MS(I,5)	ME	The equilibrium integer.Aces maintained.
C(I,N)		The mole fraction of the Nth component.

ACES allows 50 streams and 10 components.S(I,10) to S(I,15) are spare.

4. The storage for the Ith MDHX MODULE :

Location	Name	Detail
XM(I,1)	AD	Alphanumeric label.
* XM(I,2)	HS	The sectional holdup on the shell-side.
* XM(I,3)	HT	The sectional holdup on the tube-side.
* XM(I,4)	HABN	See Section 4.6.2.
XM(I,5)	QS,QT	The heat flow rate.See Section 4.6.2.
* XM(I,6)	WBN	The sectional wall heat capacity.
* MM(I,1)	MFX	See the MHLDP storage.Set to 0 in XICHX.
* MM(I,2)	MHL	See the MHLDP storage.Set to 0 in XICHX.
* MM(I,3)	MDH	The XICHX flag.Set to -2 in XICHX.
* MM(I,4)	N	The number of sections in MDHX.
* MM(I,5)	ISO	See the MHLDP storage.Set to 1 in XICHX.
* MM(I,6)	JWH	See the MHLDP storage.
MM(I,7)	MMO	The MODULE call order.Maintained by ACES.
MM(I,8)	ILKT	JDV on entry to MDHX.

5. The storage for the Ith MGPLT MODULE :

Location	Name	Detail
XM(I,1)	AD	Alphanumeric label.
* XM(I,2)	HL	The holdup.
* XM(I,5)	Q	The external heat input rate.
* XM(I,6)	CPHL	The plate equivalent heat capacity.(ITY.GT.5)
* XM(I,7)	HTC	The hydraulic time constant(Section 3.1.7).
XM(I,8)	BDV	The backward derivative storage.
* XM(I,9)	VAP	The vapour flow rate storage(ITY.LT.6).
XM(I,10)	TEM	The temperature.See Section 4.6.4.
* MM(I,1)	MRX	See the MHLDP storage.
* MM(I,2)	ITY	The plate type selector.For the backward derivative choice ITY can be 1,2,3,4,or 5 for 'ordinary,feed,sidestream,bottom,or boiler' plate.For the empiric choice add 5 to these numbers.
MM(I,7)	MMO	The MODULE call order.Maintained by ACES.
MM(I,8)	ILKT	JDV on entry to MGPLT.

6. The storage for the Ith MHLDP MODULE :

Location	Name	Detail
XM(I,1)	AD	Alphanumeric label.
* XM(I,2)	HL	The holdup.
* XM(I,5)	Q	The external heat input rate.
XM(I,10)	TEM	The temperature.See Section 4.6.4.
* MM(I,1)	MRX	The reaction flag.See Section 4.6.1.
* MM(I,2)	MHL	The holdup flag.See Section 4.6.1.
* MM(I,5)	ISO	The isothermal operation flag.0 for isothermal.
* MM(I,6)	JWH	The FOUND INDICATOR.See Section 4.6.1.
MM(I,7)	MMO	The MODULE call order.Maintained by ACES.
MM(I,8)	ILKT	JDV on entry to MHLDP.

b. The storage for the Ith MHTR MODULE :

	Location	Name	Detail
	XM(I,1)	AD	Alphanumeric label.
*	XM(I,2)	HA	See Section 4.6.4.
*	XM(I,3)	HAX	See Section 4.6.4.
*	XM(I,4)	UA	See Section 4.6.4.
*	XM(I,5)	QX	See Section 4.6.4.(See below.)
*	XM(I,6)	TX	See Section 4.6.4.
*	XM(I,7)	TW	See Section 4.6.4.
*	XM(I,8)	WC	See Section 4.6.4.
*	XM(I,9)	PS	The heating medium supply pressure. (MGT=3 only,see Section 4.6.4.)
*	XM(I,10)	AV	The fractional valve opening(MGT=3).
*	XM(I,11)	CV	The valve constant(MGT=3).
*	XM(I,12)	FLM	The heating medium flowrate(MGT=3).
	XM(MMM,5)	Q	The heat flowrate to MODULE MMM.
*	XM(MMM,10)	TEM	The temperature on the MMM side.
*	MM(I,3)	MAM	The master MODULE served by MHTR.
*	MM(I,4)	MGT	The service flag.See Section 4.6.4.
*	MM(I,5)	MCM	The index for the heating medium. For MGT=3 only.Initialize A1,A2,A3, and VLT for MCM.
	MM(I,7)	MMO	The MODULE call order.ACES maintained.
	MM(I,8)	ILKT	JDV on entry to MHTR.

The reboiler option (MGT=3) involves the loop below.

```
┌→ QX=FLM*VLT
│  TX=TW+QX/HAX
│  PX=XPAECT(A1,A2,A3,TX)
└  FLM=AV*CV*SQRT(PS*(PS-PX))
```

where FLM is the guessed variable.

c. The storage for the Ith MLXTR MODULE :

	Location	Name	Detail
	XM(I,1)	AD	Alphanumeric label.
*	XM(I,2)	HJ	The holdup on raphinate side.
*	XM(I,3)	HI	The holdup on the extract side.
*	XM(I,N+3)	D.C.	The distribution coefficient for the Nth chemical component.
*	MM(I,1)	MPX	See Section 4.5.5 and the MHLDP storage.
*	MM(I,4)	JWH	See the MHLDP storage.
	MM(I,7)	MMO	The MODULE call order.ACES maintained.
	MM(I,8)	ILKT	JDV on entry to MLXTR.

9. The storage for the Ith MTSEC MODULE :

Location	Name	Detail
XM(I,1)	AD	Alphanumeric label.
* XM(I,2)	E1	The energy input rate into the turbine. See Franks(1972).
* XM(I,3)	PS	The gas supply pressure.
* XM(I,4)	PE	The exit pressure.
* XM(I,5)	AV	The fractional valve opening.
* XM(I,6)	AN	The turbine speed.
* XM(I,7)	W	The output flowrate.
* XM(I,8)	FR	The friction constant.
* XM(I,9)	AC	A constant defined below.
* XM(I,10)	AMOM	The turbine moment of inertia.
* XM(I,11)	CV	The valve constant.
* XM(I,12)	A	Characteristic point.See Franks(1972).
* XM(I,13)	B	Characteristic point.
* XM(I,14)	C	Characteristic point.
* MM(I,6)	JWH	The SOUND INDICATOR for W iteration.
MM(I,7)	MMO	The MODULE call order.ACES maintained.
MM(I,8)	ILKT	JOV on entry to MTSEC.

The work done by the turbine=AN**3*(FR+AC*W/AN).

10. The storage for the Ith SNSR :

Location	Name	Detail
SN(I,1)	AD	Alphanumeric label.
* SN(I,2)	SNSD	The measurement(output).
* SN(I,3)	XMSR	The measured variable(input).
* SN(I,4)	D	See Section 4.9.2.
* SN(I,5)	SNG	The sensor gain.See Section 4.5.1.
* SN(I,6)	TTNV2	TT or TNV2.See Section 4.9.2.
* SN(I,7)	EBT	See Section 4.9.2.
* MDS(I,1)	ISD	The sensor delay order(Section 4.5.1).
MDS(I,2)	MSO	The sensor call order.ACES maintained.

ACES allows 30 sensors.SN(I,8) to SN(I,10) are spare.

APPENDIX 3 A PROGRAM DESCRIPTION SEGMENT FOR USERS

```
LIST (LP)
PROGRAM (USER)
INPUT  1=CR0
INPUT  3=TT0
INPUT  5=CR1
OUTPUT 2=LP0/132
OUTPUT 6=LP1/132
OUTPUT 8=LP2/132
OUTPUT 9=LP3/132
COMPRESS INTEGER AND LOGICAL
COMPACT PROGRAM
EXTENDED DATA
TRACE 2
END
TRACE 1
```

APPENDIX 6 ON THE PORTABILITY OF ACES

Machine independence is an important requirement for a program package like ACES. The latter is coded in ANSI FORTRAN. The author developed ACES on the 1904S paying particular attention to the question of portability; machine-specific features were avoided as far as possible. Tests on the ICF revealed however that a transfer to another computer may still necessitate some changes in the code due to differences in character handling, FORTRAN implementation, and carriage control. In the verification of ACES on the ICF, efforts were focussed on obtaining the same numerical answers as those on the 1904S; the layout carried less weight. The transfer to the ICF took about four man-hours after which the program ran to produce identical results. The modifications needed are detailed below.

On the 1904S up to eight characters can be packed into a REAL variable. ICL routines COPY8 and COMP8 are used respectively to copy and compare REAL variables so packed. On the other hand the ICF stores a maximum of four characters per REAL variable and eight per DOUBLE PRECISION variable. This machine does not require special routines for character handling. ACES variables initialised in DATA statements with eight characters were made DOUBLE PRECISION to gain error-free compilation on the ICF. Further rearrangement of storage is still necessary to accommodate the difference. Dummy routines were inserted to replace COPY8 and COMP8.

The ICF runs required further EXTERNAL statements in SUBROUTINES where the structure shown occurred:

```
      SUBROUTINE X (Y)
      EXTERNAL Y
      CALL Y
      RETURN
      END
```

The ICF free-format input is of the form

```
      READ (1,*)X
```

compared with the ICL implementation

```
      READ (1,100)X
  100 FORMAT (G0.0)
```

Where free-format numeric input followed alphanumeric (A-format) input on the same line, the lines were divided with parallel changes in the data files.

The ICF compiler seems to be more strict than that of the 1904S. The former compiler detected two types of error, namely "no path to this statement", and "uninitialised variable". These errors can come about in long codes due to misspelling for instance. Therefore the more strict compilers are preferable.

The work on the ICF has shown that ACES is portable in principle and requires little editing to produce identical results in a new environment. This enhances the potential of ACES which deserves further development.

REFERENCES

Akhtar N, McGrath L, Roberts P D 1979 Dynamic modelling and partial simulation of a pilot scale column crystalliser. Desalination 28:1-11.

Amundson N R 1966 Mathematical Methods in Chemical Engineering, Volume 1. Prentice-Hall, Englewood Cliffs.

Bobrow S, Ponton J W, Johnson A I 1971 Simulation of the transient behaviour of complex chemical plants using a modular approach. Can J Chem Engg 49:391-397.

Brewer J W 1974 Control systems:analysis, design and simulation. Prentice-Hall, Englewood Cliffs.

Brignell J E, Rhodes G M 1975 Laboratory on-line computing. International Textbook Company Limited, London.

Chu Y 1969 Digital simulation of continuous systems. McGraw-Hill Book Company, New York.

Crowe C M, Hamielec A E, Hoffman T W, Johnson A I, Shannon P T, Woods D R 1969 Chemical plant simulation. McMaster University Printing Department.

Daie S 1980 Computer control of chemical plants with special reference to distillation column control. Ph.D. thesis in preparation, The University of Aston, Birmingham.

Edwards E, Lees F P 1973 Man and computer in process control. Charlesworth and Co. Ltd, Huddersfield.

Elzas M S 1979 What is needed for robust simulation? In: Zeigler B P, Elzas M S, Klir G J, Oren T I (eds) Methodology in systems modelling and simulation. North Holland Amsterdam p 57-91.

Enright W H 1978 Improving the efficiency of matrix operations in the numerical solution of stiff ordinary differential equations. ACM Transactions on Mathematical Software 4:127-136.

Enright W H, Hull T E 1976 Comparing numerical methods for the solution of stiff systems of ODES arising in chemistry. In: Lapidus L, Schiesser W E (eds) Numerical methods for differential systems : recent developments in algorithms, software, and applications. Academic Press, London p 45-66.

Evans L B, Seider W D 1976 The requirements of an advanced computing system. Chem Engng Prog 72:80-83.

Farabi H 1978 The cascade control of a partially simulated CSTR. Ph.D. thesis. The University of Aston, Birmingham.

Forsythe G E, Moler C B 1967 Computer solution of linear algebraic systems. Prentice-Hall, Englewood Cliffs.

Fowler J R, Harvey D J 1978 Dynamic simulation of a PVC process. Chem Engng Prog 74:61-66.

Franks R G E 1972 Modelling and simulation in chemical engineering. John Wiley and Sons, New York.

Franks R G E, Miller D N 1979 Computed temperature-time excursions at succeeding locations in an industrial reactor. Simulation 32:123-132.

Froberg C E 1972 Introduction to numerical analysis. Addison-Wesley Publishing Co, Massachusetts.

Gear C W 1971 Numeric initial value problems in ordinary differential equations. Prentice-Hall, Englewood Cliffs.

Goh P K C 1978 Simulation in chemical engineering. M.Sc. thesis. The University of Aston, Birmingham.

Harvey D J, Fowler J R 1976 Putting evaporators to work: dynamic process modelling of a quadruple effect evaporator. Chem Eng Prog 72:47-52.

Hindmarsh A C, Byrne G D 1976 Applications of EPIDSODE: an experimental package for the integration of ordinary differential equations. In: Lapidus L, Schiesser W E (eds) Numerical methods for differential systems: recent developments in algorithms, software and applications. Academic Press, London p 147-166.

Hon V M L, Chen C S, Marsaioli A 1979 Computer simulation of dynamic behaviour in vacuum evaporation of tomato paste. Transactions of ASAE 22:215-218.

ICL SLAM Manual 1974 International Computers Limited.

Ingham J, Dunn I J 1974 Digital simulation of stagewise processes with backmixing. Chemical Engineer No.286: 354-367.

Jacoby S L S, Kowalik J S 1980 Mathematical modelling with computers. Prentice-Hall, Englewood Cliffs.

Jennings A 1977 Matrix computation for engineers and scientists. John Wiley and Sons London.

Jenson V G, Jeffreys G V 1977 Mathematical methods in chemical engineering. Academic Press, London.

Johnson A I, Lozada A, Anvari M, Stephanopoulos G, Weng P 1975 Steady state and dynamic studies of a multi-product chemical plant. Can J Chem Engng 53:340-345.

Johnson A I, Barney J R 1976 Numerical solution of large systems of stiff ordinary differential equations in a modular simulation framework. In: Lapidus L, Schiesser W E (eds) Numerical methods for differential systems : recent developments in algorithms, software and applications. Academic Press, London p 97-124.

Kim C, Friedly J C 1974 Approximate dynamic modelling of large staged systems. Ind Eng Process Des Dev 13:177-181.

Kocak M C 1980a Computer program listings for ACES. Deposited with Dr B Gay at the Chemical Engineering Department, University of Aston, Birmingham.

Kocak M C 1980b Test outputs for ACES. As 1980a.

Koenig H E, Tokad Y, Kesavan H K, Hedges H G 1967 Analysis of discrete physical systems. McGraw-Hill Book Company New York.

Korn G A, Wait J W 1978 Digital continuous-system simulation. Prentice-Hall, Englewood Cliffs.

Lee W, Weekman V W (Jr) 1976 Advanced control practice in the chemical industry : a view from industry AIChE J 22:27-38.

Luyben W L 1973 Process modelling, simulation and control for chemical engineers. McGraw-Hill Book Company, New York.

Martin E N 1979 The modelling of injectors, pumps and compressors in process simulation. Transactions of the Institution of Measurement and Control. 1:67-73.

Maudsley D 1978 An approach to the mathematical modelling of dynamic systems. Measurement and Control 11:181-189.

Mitchell E E L, Gauthier J S 1976 Advanced continuous simulation language (ACSL). Simulation 26:72-78.

Motard R L, Shacham M, Rosen E M 1975 Steady state chemical process simulation. AIChE J 21:417-436.

Mukesh D 1980 Dynamics of a continuous stirred tank reactor for different reaction orders. Ph.D. thesis in preparation. The University of Aston, Birmingham.

Myers A L, Seider W D 1976 Introduction to chemical engineering and computer calculations. Prentice-Hall, Englewood Cliffs.

Ord-Smith R J, Stephenson J 1975 Computer simulation of continuous systems. Cambridge University Press, Cambridge.

Oren T I 1977 Software for simulation of combined continuous and discrete systems : a state of the art review. Simulation 28:33-45.

Overturf B W, Reklaitis G V, Woods J M 1978a GASP IV and the simulation of batch/semi-continuous operations: single train process. Ind Eng Process Des Dev 13:161-165.

Overturf B W, Reklaitis G V, Woods J M 1978b GASP IV and the simulation of batch/semi-continuous operations : parallel train process. Ind Eng Process Des Dev 13:166-175.

Perry R H, Chilton C H 1973 Chemical Engineers' Handbook. McGraw-Hill Kogakusha Limited, Tokyo.

Ponton J W, Johnson A I, Browne P 1972 Application of a modular computer simulation system to the control of a reactor train. Can J Chem Engng 50:275-280.

Pritsker A A B 1974 The GASP IV simulation language. John Wiley and Sons New York.

Pritsker A A B, Pegden C D 1979 Introduction to simulation and SLAM. Wiley and Sons New York.

Ramirez W F 1976 Process Simulation. Lexington Books Toronto.

Robinson E R 1975 Time dependent chemical processes. Applied Science Publishers London.

Rosenbrock H H, Storey C 1970 Mathematics of dynamical systems. Nelson and Sons Limited London.

Schaal M, Aschner F S 1978 Dynamic behaviour of thin film sea water evaporators. Desalination 25:233-252.

SCI Simulation Software Committee 1967. The SCI continuous system simulation language (CSSL). Simulation 9:281-303.

Seinfeld J H, Lapidus L, Hwang M 1970 Review of numerical integration techniques for stiff ordinary differential equations. Ind Eng Fundamentals 9:266-275.

Shah M J 1976 Engineering simulation using small scientific computers. Prentice-Hall Englewood Cliffs.

Shan S N 1977 A CSMP example for chemical engineering. Simulation 29:88-92.

Shampine L F, Gear C W 1979 A user's view of solving stiff ordinary differential equations. SIAM Review 21:1-17.

Smith C L, Pike R W, Murrill P W 1970 Formulation and optimisation of mathematical models. International Textbook Company, Pennsylvania.

Speckhardt F H, Green W L 1976 A Guide to using CSMP - the continuous system modelling program. Prentice-Hall Englewood Cliffs.

Stainthorp F P, Benson R S 1974 Computer aided design of process control systems. Chemical Engineer No.289:531-535.

Stainthorp F P, Valdman B 1979 Strategies for the computer-aided design of process control systems. Paper presented at the Institution of Chemical Engineers Design 79 Symposium, Birmingham.

Stephenson R E 1971 Computer simulation for engineers. Harcourt Brace Jovanovich New York.

Stutzman L F, Deschard F, Morgan R, Koup T 1976 FAST: a translator for the solution of stiff and non-linear differential and algebraic equations. In: Lapidus L, Schiesser W E (eds) Numerical methods for differential systems : recent developments in algorithms, software and applications. Academic Press London p 125-146.

Tyreus B D, Luyben W L, Schiesser W E 1975 Stiffness in distillation models and the use of an implicit integration method to reduce computation times. Ind Eng Process Des Dev 14:427-433.

Wilkinson J H 1965 The algebraic eigenvalue problem. Clarendon Press Oxford.

Wilson J A, Smith W 1979 Dynamic modelling of counter-current mass transfer processes. Transactions of the Institute of Measurement and Control. 1:37-45.

ACES Library Variables

ACES variables are defined in Appendix 2 and the ACES arrays in Appendix 4 which also gives the names and definition of the various array elements. Array elements which are to be initialised are marked in Appendix 4. Cross references to the array elements are also given in the listings of most subprograms (Kocak 1980a).

The subprograms which make up the ACES library are listed according to classification in Appendix 3.

The COMMON blocks used are

YENI1 System variables

YENI2 STATE arrays

YENI3 SENSOR and CONTROLLER arrays

YENI4 MODULE arrays and reaction terms

YENI5 Physical properties and stream arrays

YENI6 Integration variables

YENI7 GEAR variables

YENI8 GEAR arrays

SLAMC1 Communication channels

The above mentioned variables are detailed in Appendix 2.