DYNAMIC SIMULATION OF CHEMICAL PROCESSES

Lakhbinder Singh

Doctor of Philosophy

The University of Aston in Birmingham

June 1991

SUMMARY

This thesis describes the design and implementation of an interactive dynamic simulator called DASP II. The starting point of this research has been an existing dynamic simulation package, DASP which will be referred to as DASPI throughout this thesis. DASP II is written in standard Fortran 77 and is implemented on universally available IBM-PC or compatible machines. It provides a means for the analysis and design of chemical processes. Industrial interest in dynamic simulation has increased due to the recent increase in concern over plant operability, resiliency and safety.

DASP II solves all model equations simultaneously and is therefore termed an equation oriented simulation package which allows solution of dynamic and steady state equations. The steady state can be used to initialise the dynamic simulation. A robust non-linear algebraic equation solver has been implemented for steady state solution. This has helped to make DASP II more robust than DASPI.

A graphical front end is used to generate the process flowsheet topology from a user constructed diagram of the process. A conversational interface is used to interrogate the user with the aid of database files to complete the topological information.

An original modelling formulation implemented in DASP II provides a simple mechanism for parameter switching which creates a more flexible simulation environment. The problem description generated is by a conversational procedure using database files. The model format used allows the same model equations to be used for dynamic and steady state solution.

All the useful features of DASP I are retained in DASP II. The program has been demonstrated and verified using a number of example problems. Significant improvements using the new NLAE solver have been shown. Topics requiring further research are described. The benefits of variable switching in models has been demonstrated with a literature problem.

Keywords

Dynamic simulation **Computer aided design**
Flowsheet generation **Equation oriented approach**
Non-linear algebraic equations

ACKNOWLEDGEMENTS

I would like to thank Dr John Fletcher for his supervision of this project. May I express my gratitude to the management at BP International, and in particular Mr Peter Banks for his encouragement, understanding and help. Thanks are also due to Mr Richard Bailey and Professor Ross Taylor for the many useful comments and suggestions and to my collegues at the Department of Chemical Engineering for the advice and moral support offered. I also wish to thank my wife, parents, brother, sister and family for their moral support and understanding.

Special thanks are also offered to Professor Paul Preece of the University of Swansea (formerly of Leeds University), for allowing me to use and modify PFG/PID for this work.

Funding for this project was received from the Science and Engineering Research Council and this is gratefully acknowledged.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

1.0 INTRODUCTION

This thesis is about dynamic simulation of chemical processes. The justification for this work is the severe problems encountered in accurately predicting the behaviour of chemical processes and the need to solve problems for safe design of chemical plant.

Dynamic simulation of chemical processes has become more significant over the last two decades as energy consumption has increased and resources have been depleted. During the same period the chemical industry has suffered several catastrophic incidents including Flixborough (Kletz (1988)), Chernobyl (Mould (1988)), Bhopal (Shrivastava (1987)), Seveso (Kletz (1988)) and Piper Alpha (Cook (1989)). As a result of these disasters there has been a general tightening in safety standards. Increasing public awareness of pollution and its effect on the environment has led to legislation for the production of less effluent to a higher purity standard.

These pressures have created financial difficulties for the industry. A great deal of capital cost is incurred in order to meet the increasingly stringent government legislations for pollution and safety (Anon (1984)), (Anon (1989)), (Stover (1985)). Operating costs have also been adversely affected by upward trends in raw materials and fuel. Great savings can, however, be made by improved reliability, increased efficiency, a greater recovery of usable by-products and also by incorporating a higher degree of energy integration. Chemical engineers can achieve some of the above savings by evaluating alternative process configurations. Process simulation provides one method for carrying out such studies. This also enables a better understanding of the process, hence allowing the engineers to improve the controllability of the process which in turn yields a safer and more pollution free plant. Until recently engineers relied mainly on experience for process engineering; however, as new process technology, such as the use of sub-sea slug catchers in the oil and gas industry (Sasnow (1989)), has rapidly advanced, only a small number of engineers have the required knowledge to gain the maximum benefits from these processes. Fortunately, during the same period there has been a vast increase in

computing and numerical methods technology which enables process simulation to predict process behaviour in a relatively small period of time.

The simulation of chemical processes can be defined as the use of mathematical models to mimic these processes. The models are represented by a set of differential and algebraic equations whose variables represent particular characteristics of the process. Steady state simulation produces time independent values of the variables, whereas, dynamic simulation generates the time dependent solutions of the equations. The models are usually solved using numerical methods on a digital computer. Extensive work has been undertaken on steady state simulation, which may be used to perform mass and energy balance calculations and preliminary optimization studies of steady state chemical processes. More recently it has been developed to enable an engineer to solve the dynamic equations of chemical plant systems. This can be used to determine whether a steady state will be reached and whether the process design is safe, efficient and environmentally clean. Since dynamic models contain differential equations in conjunction with algebraic equations which must be solved over a given period of time, a greater computing effort is required than that utilised for steady state simulation of an equivalent problem.

1.1 Dynamic Process Simulation Applications

Dynamic simulation provides predictions of the performance of chemical processes which were previously collected using pilot plant studies. The following are the more important applications of dynamic analysis:-

a) Start-up and Shut-down studies

Dynamic simulation can be extremely beneficial in finding safe and efficient process start-up and shut-down procedures. These must be known prior to initial start-up to ensure safe operation.

b) Process Control Strategies

Dynamic simulation over the last decade has proved to be a useful tool for process control design. It provides control engineers with the ability to rapidly investigate different control strategies to achieve the maximum control for most chemical processes. This includes evaluating control effectiveness by comparing advanced control schemes with conventional and even manual techniques. This allows for better understanding of problematic control applications. Once the most appropriate control scheme is chosen, the controllers can be tuned to achieve an optimum response to a given perturbation. By using simulation, the designers avoid having to rely on past experience alone. This allows a reduction in overdesign of the control scheme to allow for uncertainty.

c) Batch Processing

A demand for new chemicals has arisen on a smaller scale of production than in the bulk chemical industry. This demand has been met using batch processing. This enables often production of many different chemicals from a small number of plant units, such as paints and speciality products which are manufactured in a wide variety of formulations from the same equipment. Other products - notably certain polymers are difficult to produce in the required qualities by using continuous processes. While certain foods and drugs are made in volumes too small to justify continuous processes. Dynamic simulation can be used for analysis and control design in these unsteady processes.

d) Effects of process parameters

Dynamic simulation allows the study of the response of chemical plants to perturbations in feedstock, pressure, temperature and even climate. Consequently, optimum values of process parameters can be determined. An extreme example of this is in the assessment of the effect of changes in ambient temperature on a condenser. A change in ambient air temperature can alter the amount of condensation taking place in the condenser, this may significantly affect the complete distillation column and hence the product yields.

e) Hazard Analysis

Process engineers can use dynamic simulation to assess the safety problems associated with each process after the onset of perturbations and hence determine the limits of operation. Dynamic simulation enables the consequences of equipment failure to be understood and the necessary operations to prevent dangerous and unwanted situations can be designed.

f) Operability Studies

Highly integrated and complex processes can be simulated to study the effect of various process changes. The control requirements of the process can also be determined. This enables engineers to gain confidence with the process and work out emergency procedures.

g) Operator training

Dynamic simulation is becoming a method for training plant operators intensively within a short period of time. It offers hands on experience and exposure to a wide variety of process conditions, including both routine events and those rarely encountered on the job. This reduces the possibility of operator error through ignorance on the actual plant. Pathe (1986) estimated from insurance surveys that approximately 25% of losses from accidents in the petrochemical industry are the result of process upsets and human error. Simulations can provide a key link between operator training effort and a subsequent reduction in the number of plant trips and accidents.

It can be seen from the uses of dynamic simulation discussed above, that the possibilities for chemical process design have been greatly increased by the introduction of simulation. All the tasks now possible with dynamic simulation, were previously carried out on a full size chemical plant or else on a pilot plant, where scale up can cause different behaviour. Chemical engineers can also now analyse different process configurations and provide answers to "what would happen if ....." type questions in a sufficiently small response time. Consequently, chemical plants have been made more efficient due to the implementation of better equipment and control design and employment of greater energy integration. They have also become safer because of better understanding by engineers and operators. This shows the large economic incentive for applying dynamic simulation to chemical processes. Mix *et al* (1977) estimated that achievement of a ten per cent saving in the energy consumption would result in the USA alone saving 2×10^4 BTU's of energy per year, currently worth about $500 million per year.

Burchell (1989) discussed the application of dynamic simulation for operator training by BP on a new acetic acid and acetic anhydride plant. The acetyls process which involved both continuous and batch unit operations had only been demonstrated under laboratory conditions. As the new plant required advanced control schemes and utilised a Distributed Control System (DCS), it was decided that the process operations training would incorporate DCS training and plant operations training to ensure safe, effective and efficient operation of the plant. This replaced the orthodox method of 'on the job' training during commissioning. Burchell (1989) states that the benefits accrued by the application of the simulator included:-

a) A general reduction in training time.

b) An increase in effectiveness and efficiency of the training process.

c) Reduction in commissioning time.

d) Reduction in the frequency of the plant trips and plant down time.

e) An increase in the trainees confidence to effectively perform: cold and warm start-ups, production change and maintenance shut-downs.

It can be seen from the uses of dynamic simulation discussed above, that the possibilities for chemical process design have been greatly increased by the introduction of simulation. All the tasks now possible with dynamic simulation, were previously carried out on a full size chemical plant or else on a pilot plant, where scale up can cause different behaviour. Chemical engineers can also now analyse different process configurations and provide answers to "what would happen if ....." type questions in a sufficiently small response time. Consequently, chemical plants have been made more efficient due to the implementation of better equipment and control design and employment of greater energy integration. They have also become safer because of better understanding by engineers and operators. This shows the large economic incentive for applying dynamic simulation to chemical processes. Mix *et al* (1977) estimated that achievement of a ten per cent saving in the energy consumption would result in the USA alone saving 2×10^4 BTU's of energy per year, currently worth about $500 million per year.

Burchell (1989) discussed the application of dynamic simulation for operator training by BP on a new acetic acid and acetic anhydride plant. The acetyls process which involved both continuous and batch unit operations had only been demonstrated under laboratory conditions. As the new plant required advanced control schemes and utilised a Distributed Control System (DCS), it was decided that the process operations training would incorporate DCS training and plant operations training to ensure safe, effective and efficient operation of the plant. This replaced the orthodox method of 'on the job' training during commissioning. Burchell (1989) states that the benefits accrued by the application of the simulator included:-

a) A general reduction in training time.

b) An increase in effectiveness and efficiency of the training process.

c) Reduction in commissioning time.

d) Reduction in the frequency of the plant trips and plant down time.

e) An increase in the trainees confidence to effectively perform: cold and warm start-ups, production change and maintenance shut-downs.

The cost of the simulator has been estimated as 2-3 days production. The time saving by the company was much larger than this. A further example of the value of training simulators is illustrated by Pathe (1986). He described the key role played by a real time process simulator in obtaining "a near flawless and world record start-up" of C-I-L's Ammonia plant. The management confirmed the significant contribution made to the successful start-up by the extensive pre-training of the operating personnel using the simulator, which duplicated the operation of the plant in real time response. It was estimated that the training, involving more than 100 simulated trip situations, was equivalent to 10 years of on the job training and plant experience to acquire the same skills.

Alcock (1985) gives an example involving a hazard and operability study for revamping the water injection system of a North Sea platform. The performance of the redesigned system was evaluated using dynamic simulation to provide information for the hazard and operability study. The main issue of concern was the safe operation of the injection pumps which could be seriously damaged by a failure in the sea water supply. Although, it was unclear whether there would be sufficient liquid on the suction side to allow for safe shut down. The dynamic simulation indicated that the proposed control systems would perform satisfactorily.

Prokopakis and Seider (1983) applied dynamic simulation to azeotropic distillation using suitable algorithms to provide an efficient method for understanding this complex process. They concluded that the open loop response for dehydration of ethanol with benzene is unusual, with steep concentration and temperature fronts moving up and down the column. A combination of disturbances can diminish this movement, indicating that azeotropic distillation towers can be controlled by adjusting the decanter by pass fraction and reboiler heat duty.

These examples show that the economic and safety incentives for applying dynamic simulation to chemical engineering processes are large. Dynamic simulation aids the engineer in the decision making process by providing information that is difficult and time consuming to

obtain by other means. Consequently, dynamic simulation has a use at almost every stage in the life cycle of a chemical process plant.

A typical chemical process consists of a series of process units. A dynamic simulation of the whole process could be performed using a computer program specially written for the purpose in a "stand alone" manner. This would have to contain a model for each type of process unit. The effort of simulation is reduced by constructing a simulation package containing the unit models and the means of linking them together. This program can then be used repeatedly to solve different simulation problems. This development follows the pattern set for steady state simulation (Evans (1981)).

Ponton (1983) describes the simulation of a process by a simulator called DPS which required 2 to 3 man days of effort, whilst the same process required 6 man months of effort writing and using a stand alone package. Exxon (Slaver (1986)) found that stand alone systems were effective for problem solving but inefficient for running the variety of cases that were required in system studies. Slaver (1986) concluded that the primary requirement for system studies is that the simulation be sufficiently flexible to test a variety of cases.

During the 1950's analog computers were the major systems used for continuous simulations. The translation to digital simulation via hybrid simulators exposed many engineers to continuous system simulation languages (CSSL). CSSL's enable dynamic simulation problems to be posed in a mathematical format without a great deal of programming effort. The early CSSL's took the form of high level computer languages, such as FORTRAN. These subroutines did not provide the flexibility and convenience desired in most simulation studies and subsequently led to the Simulation Council Inc publishing the CSSL Standard (Anon.(1967)) which defines the program structure for future simulation language developments. A number of organisations developed CSSL's in line with the standard, including IBM which with its significant computer market penetration released the widely used CSMP (Speehart and Green (1976)). Other examples include ACSL (Mitchell (1978)), DARE-

P (Korn and Wait (1978)) and GASP (Pritsker (1974)) which was developed for discrete, continuous and combined simulations.]

Wood *et al* (1984), Cameron (1981) and Joglekar and Reklaitis (1984) have highlighted the major limitations prevalent with existing CSSL's for chemical engineering applications. Some of the limitations being:-

a) Lack of interaction with the simulation means the user has difficulty in extensive experimentation.
b) The inability to directly call external FORTRAN subroutines, such as physical property estimations.
c) The inability to calculate the steady state solution of the models by the solution of non-linear algebraic equations.
d) The rigid output format must be adhered to.
e) Computing run time is high.

The lack of features discussed above prompted chemical engineers to develop general purpose dynamic process simulators. These dynamic simulation systems, like steady state simulators, can be broadly classified as modular based systems and equation oriented systems (Westerberg *et al* (1979)).

1.2 Sequential Modular Approach

The majority of current steady state simulators are based on the sequential modular approach. As the early dynamic simulation systems evolved from flowsheeting packages they also followed this strategy. Unit operations are modelled with the equations housed in modules, which to aid solution may contain numerical methods particularly suited for these models, e.g. the use of continuation methods for distillation modelling (Kovach and Seider (1987)) which can converge from starting points further from the solution than Newton methods. As distillation systems are usually large, containing many variables the provision of a good enough starting

point for the Newton methods becomes difficult. Another advantage of the continuation methods over Newtons method is their ability to find multiple solutions.. It is assumed that the information flow in the models follow the material flow in the chemical plant and design variables are defined at all times. The simulator executive calls the unit modules in a predetermined sequence. Modules are structured to calculate derivatives of the output variables given the input variables, model parameters and the output variables at time t. The integrator then integrates the equations to generate the variables at the next time interval. Sequential modular programs include DYFLO (Franks (1972)), DYNSL (Paterson *et al* (1980)), DYSCO (Briggs (1974)) and FLOWPACK II (Aylot *et al* (1985)). However, this approach has inherent limitations that make it ineffective in dealing with large and complex processes. This is a result of its evolution from steady state simulators. One of the difficulties associated with modular dynamic simulators involves the degree of coupling between the equations in the various modules (Cameron (1981)). Sequential modular systems can be divided into two divisions depending on the coupling mechanism adopted in the simulator : uncoupled and coupled solution systems. In the uncoupled systems the equations from a particular module are solved independently of the other modules. The coupling only occurs at the end of predefined time intervals and not during each integration step. Coupled systems employ equation coupling at every step of integration, therefore all the differential equations are integrated in unison over the time interval. Optimisation studies and complex processes incorporating recycle loops also pose problems for this approach (Westerberg (1979)).

1.3 Equation Oriented Approach

To overcome the pitfalls existing in sequential simulators and the general advance in computing, researchers started to turn their attention to the equation oriented approach. Some of the subsequent simulators developed include DPS (Thambynayagom *et al* (1981)), SPEEDUP (Pantelides (1988)), BOSS (Joglekar and Reklaitis (1984)), QUASILIN (Smith (1985)) and ASCEND II (Kuru (1982)).

The main attribute of the equation oriented approach is the ability to solve all the model equations simultaneously. Thus, it overcomes the problem of coupling encountered in sequential modular simulators. All the model equations in the process are solved together at each time step. Two different variations of this type of simulator exist, the main difference being the method of generating the equation block for simultaneous solution. The first type involves the user assembling the model equations in a large system and the solver then acts upon this system, it is obvious this approach can be time consuming. The alternative development is the modularly organised equation based approach which entails the model equations being extracted from modules containing the equations for the required unit operations.

1.4 Difficulties Associated with the Dynamic Simulation of Chemical Processes

The development of dynamic simulation has been rather slow with most of the studies being confined to academic organisations and industrial attention being restricted to very few organisations. The main reason for this has been the high cost incurred in undertaking these studies. Consequently, it has been used only for the most important problems where answers were urgently required. The other difficulties met include:-

1.4.1 Limited user interactiveness

The major effort in a dynamic study is the definition of the complete and correct system description. Therefore, the simulator must provide a simple and efficient mechanism for input and output of the required information. Simulators attempt to provide this facility by using a variety of different mechanisms, including the use of a specially written simulator language. Others utilise text input files and a high level computer language. A mix of interactive facilities

and batch processing must be provided, so that operation of the program does not become too tedious for experienced users who can revert to the batch processing facility at will.

1.4.2 Stiff model equations

Chemical process models commonly contain dependent variables showing vastly different rates of change with the independent variable, which usually is time. This can lead to a "stiff" problem, where the ratio of maximum to minimum eigenvalues is large, typically greater than 1000 (Finlayson (1980)). Explicit integration methods are not generally suitable for stiff systems as they usually require very small time steps to achieve the necessary accuracy. The problem has been tackled with recently developed implicit integration techniques, such as Gears method (Gear (1971)), which show improved performance but are only suitable for computers because their highly iterative nature requires extensive computation. The problem can usually be reduced by converting differential equations, with much smaller time constants than other equations to algebraic equations, although extreme care must be taken to ensure that the resulting model is still valid.

1.4.3 Provision of a steady state

A consistent initial state has to be provided to commence the dynamic simulation. This can amount to solving the model equations with the derivatives equated to zero, yielding a set of non-linear algebraic equations from the original system of differential algebraic equations (DAE's). Consequently, it is the steady state or flowsheeting problem that must be solved. A great deal of research has generated a number of methods for the solution of non-linear algebraic equations (NLAEs), however, Newton Raphson or one of its variants remains the most popular technique in simulators (Sargent (1982)), as these methods are the most efficient solution techniques for NLAEs.

1.4.4 Handling of Discontinuities

A rapid change of state in a chemical process is represented by a discontinuity in a dynamic simulation study. This may result when a valve closes shut or a blowoff valve suddenly opens, this is termed an implicit discontinuity and requires the actual time of the switch to be located. Detecting and then stepping over the discontinuity without a great deal of difficulty poses a major problem for the dynamic simulator. Another type of discontinuity also exists and is caused by imposed changes at specified times. This type of discontinuity is called an explicit discontinuity.

1.4.5 Large number of equations

A large number of equations result if a comprehensive dynamic process study is carried out on a complete process or a large process unit, examples include multicomponent distillation columns which can yield as many as 3000 equations or distributed systems which yield variables varying in two different independent variables and result in partial differential equations. These large systems can pose difficulties for the numerical methods implemented in simulation systems. For instance the equation system must be re-initialised after a discontinuity, the likelihood of a failure increases if the system is large and in particular if the initial starting points are far from the solution. The sparsity of the problem generally increase with the size of the equation system. However, a mathematical model of a chemical process must only possess the degree of complexity required to answer questions that are posed. Excessive model complexity can be avoided by building the flowsheet from basic models. These models are validated by comparing the model simulation results with the actual plant performance. The models are then implemented in sections in order to construct the flowsheet. The resulting consolidated model is then fit for the purpose and not unnecessarily rigourous.

1.4.6 Sparse Equations Systems

Sparse matrices can be defined as matrices containing elements that are predominantly zeros occuring randomly. These matrices result when equations exhibit a dependency on only a small number of variables and can be manipulated to take advantage of the percentage and distribution of the zero elements. These systems commonly occur in chemical engineering problems.

During the solution of linear equations by standard linear equation solvers the structure of the sparse matrix is changed as new non-zero elements are generated by the elimination process. This process results in the loss of the matrix sparsity, which in turn increases the storage requirements and computation needed for obtaining the solution.

1.5 The DASP approach to simulation

DASP is an equation oriented simulation package developed at the University of Aston. The aim of the DASP project from its conception has been to provide a flexible and robust simulation package. To develop such a package the limitations discussed in Section 1.4 must be addressed and overcome.

Ogbonda (1987) attempted to overcome the following limitations occuring in the application of dynamic simulation to chemical processes : the efficient solution of sets of DAE's by the incorporation of DASSL - an implicit integrator. DASSL is also useful in the solution of stiff model equations. The inclusion of event processing which involves overcoming both implicit and explicit discontinuities. MA28 (Duff (1977)) has been implemented to handle sparse matrices.

Although, a library of chemical process models has been developed, the model structure implemented is inflexible allowing only predefined variables to be calculated from a model. This major limitation has been overcome in DASP II in which the models utilise the full potential of

the equation oriented approach and allow any variable in an equation to be defined as an unknown. The method implemented is a more efficient method than that implemented in other simulators such as SPEEDUP (Pantelides (1986)) and QUASILIN (Smith (1985)) as it allows a simple and rapid change in the model variable/parameter specification.

Another major issue that has been investigated is the lack of interactiveness in equation oriented dynamic simulation packages. A suitable interactive mechanism has been developed with the aid of an extremely useful process flowsheeting package PFG, which allows the generation of the process flowsheet topology information. An interface between a modified version of PFG and DASP II allows the automatic generation of topological information. The interface uses a conversational procedure as its interrogation method. A conversational environment is also used to generate the problem description including the simulation control parameters and the model variable/parameter information. A database system has been designed which enables all the model information to be stored in an efficient manner using a commercially available database manager thereby permitting its use on a universally available computer such as an IBM PC which has limited memory.

The robustness of DASP II has been significantly increased with the inclusion of CONLES - a non-linear algebraic equation solver. This contains three methods of solution, one each from the three classes of convergence available, i.e. local, expanded and global. Although, the methods offering local convergence properties need good initial variable estimates for solution, they are usually the most efficient algorithms. Whilst, those offering global convergence are the least efficient, they can converge from the worst initial estimates. Most simulators including DASP I have up until now utilised locally convergent methods. As non-linear algebraic equations are required to be solved during many stages of a simulation study a robust NLAE solver must be provided. This has been done in DASP II with the inclusion of CONLES.

Model validation is a problem common to all simulators. This problem can be overcome to some extent by model evolution where the model is built up in stages from basic equations. The model prediction is then assessed at each stage. Difficulties can also arise with validating the results generated by the simulators. For instance NLAEs generally possess multiple solutions and NLAE solvers can converge to different solutions depending on the choice of starting point. Consequently, the results must be thoroughly examined and infeasible solutions such as negative pressures and compositions or mole fractions greater than one can be rejected by inspection.

1.6 Scope of Thesis

Dynamic simulation of chemical processes with an equation oriented framework is discussed in this thesis.

This research project investigated dynamic simulation in particular the strategies adapted in DASP I and the limitations of this approach. The aim of this work was to develop a robust user friendly dynamic simulation system for simulating the transient behaviour of chemical processes and plant on a readily available computing medium such as IBM PC's or its compatibles. As noted earlier in this introduction, the equation oriented approach poses many interesting features that can be utilised to produce a comprehensive simulation package. However, it is also subject to many frailties that can seriously hinder its use in dynamic simulation.

These limitations are highlighted in this introduction and discussed more comprehensively in Chapter 2. The different methods used in simulation, their strengths and weaknesses are also addressed in Chapter 2. DASP I is an equation oriented dynamic simulation package developed at Aston University.

In Chapter 3 a description of the strategies implemented in DASP I, in comparison with other simulators, is given together with their limitations. Chapter 4 is devoted to the solution of non-linear algebraic equations which provides a steady state of the process which in turn can be

used as an initial starting point for dynamic simulation. The application of a graphical front end to automatically generate the process topology is described in Chapter 5. In Chapter 6 a strategy which allows the DASP II models to adopt a truly flexible equation oriented approach is outlined. The automatic generation of the problem definition using a database management system aiding interaction with the user is also described in Chapter 6. A collection of examples illustrating the features used and considered for implementation in DASP II are described in Chapter 7. The final chapter contains conclusions and recommendations for future work.

CHAPTER TWO

SIMULATOR DESIGN STRUCTURES

2.0 Introduction

A mathematical model of a chemical process is used to approximate the behaviour of the process, usually under defined conditions. Some simplifying assumptions are made in the model so that it can be solved relatively easily. Nevertheless a worthwhile balance must be achieved between the model and reality.

The model consists of a set of equations of various types, which relate the physical variables of the system being modelled. These are then solved to obtain design and performance predictions for the modelled chemical plant. Experimentation with these theoretical models can be used to overcome the expense associated with experimentation with real chemical plant. Prior to the availability of digital computers, simple mathematical models were often used to ensure solution. However, with more powerful computers readily available, more realistic models can be formulated. Mathematical models can be classified as either steady state or dynamic. Steady state models represent steady state processes which can be used for design or flowsheeting calculations. Dynamic models apply to transient processes which can be used to carry out performance predictions for chemical processes.

Chemical process simulators mimic real chemical processes by using numerical techniques on computers to solve validated mathematical models of the chemical processes. Different general purpose chemical process simulators are therefore, classified in the way the numerical techniques are used to solve the mathematical models. Two distinct approaches of simulation exist: sequential modular and equation oriented. The former solves one model at a time with some output variables being passed to the next model as input variables. Whilst the equation oriented solved all the model equations simultaneously.

These strategies which apply to steady state and dynamic simulation are discussed in Sections 2.3 and 2.4 respectively.

Most simulators can potentially carry out steady state, dynamic simulation and optimization within one package. Biegler (1989) lists several simulators demonstrating this feature. DASP I can undertake steady state and dynamic simulation.

2.1 Dynamic Simulation

Dynamic models are composed of a set of differential and algebraic equations (DAE) of the form

$$F\ (y',\ y,\ z,\ u,\ t) = \emptyset_m$$
$$g\ (y,\ z,\ u,\ t) = \emptyset_n \qquad 2.1$$

where

F is a vector of functions for the differential system

g is a vector of functions for the algebraic system

y is a vector of differential variables

u is a vector of control variables

t is the time

z is a vector of algebraic variables

$\emptyset_m$ is a null vector

$\emptyset_n$ is a null vector

The initial conditions $y_o = y\ (t_o)$ for the differential equations and $z_o = z\ (t_o)$ are required to commence the simulation from time t_o. The control variables u are functions of time and are specified between the start and final time of simulation.

2.2 Steady State Simulation

Steady state models yield sets of nonlinear and linear algebraic equations. The fundamental mathematical problem is the solution of these nonlinear algebraic equations of the form:

$$f(y, z, u, t) = \emptyset_{m+n} \qquad 2.2$$

It can be seen that the differential and algebraic system given in equation 2.1 generated by dynamic analysis is reduced to the steady state problem if the differential terms are eliminated, i.e. if the time variation is eliminated. Consequently, the steady state can be used as one set of initial conditions for the DAE system which requires a consistent and non-redundant initial state of the physical system. Initial estimates for the algebraic variables are also provided to aid the initialisation procedure.

2.3 Sequential Modular Approach

In the sequential modular approach all the units in the chemical process are represented by modules which describe the units. Each module contains mathematical models and can therefore be steady state or dynamic. These modules are simulation oriented or in other words behave like "black boxes" - generating outputs given the inputs. For dynamic simulation, variables are produced describing the output streams once the input variables, model parameters and the output variables at the previous time interval are given. Two different techniques exist within the dynamic modular simulation framework, with the main difference being the manner in which the equations are coupled from the different modules. The older simulators such as DYFLO (Franks (1972)), implement the uncoupled solution system, which solves the module equations independently from other modules. The integration through a time step is carried out sequentially over all the modules after which control is passed back to the executive routine, which in turn couples the variables. This procedure is repeated from the integration start time until the final simulation time. This method results in poor solution accuracy as outdated

variable values are used in subsequent calculations. DYFLO (Franks (1972)) employed this technique with explicit integration algorithms, whilst this integrator has been replaced by the implicit Euler method in DYFLO 2 (Franks (1982)). The alternative approach is to couple the equations at every step of the integration, so that the differential equations are integrated in unison over the time interval. This solution approach is adopted in DYNSL (Patterson and Rozsa, 1980).

The sequential modular approach was very popular during the early stages of the development of dynamic simulation because many organisations had expended a great deal of time and money on producing libraries of steady state models which could be easily converted to dynamic models. As the information flow in the program closely resembles the material flow in the system, process engineers found it easy to follow and error checking proved to be simple in the case of program failure. A great deal of research effort is still expended on simulators implementing this approach, because it allows numerical methods to be incorporated within the model. Hence, special techniques can be utilised for difficult problems. Examples of this include: Woodman (1989), who utilised continuation methods to solve complex azeotropic distillation problems. Kovach III and Sieder (1987) also attempted to solve this problem, using arc length continuation methods where the set of non-linear equations is reformulated as a set of ordinary differential equations with respect of the arc length of the path. This technique follows the actual curvature of the path, hence it is more reliable than the classical continuation methods which steps along the path by simply incrementing the homotopy parameter. However, several major problems seriously affect the efficiency of the sequential modular approach. A fundamental problem occurs with this approach when applied to dynamic simulation and involves the degree of coupling between the equations. The difficulty with uncoupled systems has been described earlier. In the coupled method, although the differential equations are solved simultaneously, coupling between the algebraic and dynamic variables is neglected. This results from different modules calculating the differential and algebraic equations separately, hence outdated variable values are used in subsequent calculations. Consequently, the correct solution for the equation system is not obtained. Another problem is encountered with flowsheets

containing recycles, these streams have to be guessed or "torn" and must then be converged. Therefore, complex flowsheets with multiple recycle streams require many iteration loops which can be time consuming to solve. As optimisation involves the addition of an outer loop on the process, it is evident that this will add to the degree of difficulty of solution (Westerberg (1981)). With steady state systems controlled simulation problems with one or more design specifications can be troublesome. These problems are usually solved by an iterative simulation in which the process is repeatedly simulated until the design specifications are met, these iterations are termed control loops. As the design specifications are usually very simple equations, the applications of control loops is an inefficient and costly mechanism for their solution. It is clear that the aforementioned problems can become severe and limiting for realistic applications which are usually complex in nature. Consequently, there has been a great deal of recent interest in the equation oriented approach, which is an alternative mechanism for the simulation of chemical processes.

2.4 The Equation Oriented Approach

In the equation oriented approach, the process model equations, connections and specifications are accumulated into one large system of equations which are then solved simultaneously. The physical property equations can also be included in the system of equations or handled separately. The physical property equations unnecessarily add to an already large system of equations which further exacerbates the problem of a large set of nonlinear algebraic equations. A substantial academic effort, including the work of Sargent and Westerberg (1964), Christensen and Rudd (1969), Westerberg (1979) and Gorczynski *et al* (1979), has been expended since the 1960's on equation based process simulators. This is a result of the general view that this approach can alleviate the inherent weaknesses commonly associated with the sequential modular approach. It is necessary for equation oriented simulators to assemble the model equations together as a system. The equations are then solved by the simulator. One example of this approach was the general purpose CSSL's which were used to dynamically

simulate chemical processes. Examples of this type of package include: CSMP (IBM (1970)), DSL (Anon. (1980)) and ACSL (Anon (1976)).

The use of continuous languages and their limitations in dynamic simulations have been discussed earlier. These weaknesses can be briefly highlighted as lack of interaction with the user, inability of accessing external code, inability to obtain the steady state, rigid output format, excessive computing time, excessive time taken to correctly formulate the system definition and the difficulty in debugging the code when errors occur. In practise other approaches have proved much easier to use. Consequently, they have only been utilised for problems in chemical engineering applications which necessitated the use of dynamic simulations.

For such complex problems, typically consisting of a few hundred variables, error free coding was seldom possible and inefficient storage meant that variable limits were easily exceeded.

To overcome the laborious task of coding the system of equations and the resulting problems, researchers such as Gorczynski *et al* (1979), Pantelides (1986) and Ogbonda (1987) considered the advantages of units being modelled in modules and subsequently developed techniques for using a modularly organised equation oriented approach. Unlike sequential modular unit models, the numerical methods were required to be kept invisible from the units to maintain simultaneity. Consequently, the modularly based equation oriented simulators usually contain a library of models, with each model containing the differential and algebraic equations describing a unit operation. The executive routines for these simulators are therefore sophisticated programs which collect the equations from the library of all the models in the flowsheet and utilise the most appropriate numerical techniques to ensure an efficient simulation. This technique is applied in QUASILIN (Gorczynski *et al* (1979)), SPEEDUP (Pantelides (1986)), SEQUEL (Stadtherr and Hilton (1982)), GOS-84 (Kohlert *et al* (1985) DASP (Ogbonda (1987)) and CHEOPE (Pagani *et al* (1989)). A distinction can be made between two general approaches to equation based flowsheeting. The first is the solution of the full problem using suitable numerical methods for nonlinear systems. The second involves partitioning the

equations into blocks which are then solved. Perkins and Sargent (1982) describe two mechanisms of partitioning, that can be used for simulation; the first method involves partitioning the full set of equations and variables into blocks which can be solved sequentially. This can reduce the size of the largest set of equations to be solved simultaneously. The second approach goes one step further, by tearing to reduce the size of the largest subproblem once partitioning has taken place. A number of variables are torn so that the remaining variables can be calculated by solving a sequence of single variable problems. A set of equations remains, equal in number to the torn variables, resulting in residuals which can enable the torn variables to be adjusted in an outer iteration loop. SPEEDUP (Pantelides (1986)) implemented the first decomposition scheme. GOS-84 (Kohlert (1985)) is another simulation system which utilises decomposition to aid convergence of the complete flowsheet. However, the majority of equation oriented simulators implement simultaneous linerization in the solution of a steady state for a problem. This approach is also attempted in different ways; Quasilin (Gorczynski *et al* (1979)) requires its models to be composed of linearised equations. The executive routine passes the unit parameters and the values of all the process variables to each unit module. Linearised equations are constructed from this information and returned to the executive. All the linearised equations are assembled from the modules and are solved by a modified form of Gaussian Elimination, generating a new set of values of the process variables. This whole process is repeated until convergence is obtained. Each unit module contains the necessary information required to linearise the original equations. This approach is very much like the simultaneous modular approach although the authors consider it an equation oriented approach.

In SEQUEL (Stadtherr and Hilton (1982)) this problem is overcome by utilising a set of equations in a fixed form.

The user must convert all model equations to the same form of one of the defined equations. The transformed model equations are then collected and solved by one of three available, non-linear algebraic solvers. Two of these are: Newton Raphson and a variant. The third is a hybrid method which consists of a first order method, used far away from the solution

because of its stability properties and a second order full Newton Raphson scheme nearer the solution to utilise its desirable speed. The user is required to specify a blending parameter and its rate of change. There is an option to use the Newton Raphson method throughout.

Pagani *et al* (1989) used another approach with their equation oriented flowsheeting package - CHEOPE. The modules are used to generate equation blocks for each process unit. The equations are then collected in a large system of equation blocks forming a quasi-tridiagonal Jacobian matrix. A Newton Raphson method is then used for solving the resulting equations.

Equation oriented simulators have the potential to overcome all the drawbacks associated with the sequential modular approach. However, industrial acceptance has been very limited because of major disadvantages existing with the equation oriented approach. One of the most serious problems is the complex executive routine that is required to generate the system equations from the necessary models. The executive routine must then solve the system of equations in the most efficient manner, and produce the results in a format that can be easily understood by the user. In order to achieve all the housekeeping efficiently the executive routine tends to be large and complex. The advent of cheap and powerful microcomputers with large storage capabilities has meant that the large and complex executive routines are not as limiting as they once were. The new generation of powerful computers also reduces the computational time of the advanced numerical methods that are used to increase the accuracy and efficiency of equation oriented simulations. These advanced numerical routines must be general purpose so that all the different types of model equations can be solved without failure. However, more reliable equation solvers have been developed (Sargent (1981), Shacham *et al* (1982), Pagani *et al* (1989)) which can handle the large, sparse system of nonlinear algebraic equations that arise. Whilst, a new generation of integration techniques have been formulated (Gear (1971), Holland and Liapus (1985)). These methods can automatically vary the step size and order of integration to handle stiff equations systems. Other researchers have worked on developing numerical methods such as sparse matrix packages (Stadtherr and Wood (1984) and Duff (1977)) which can be used to reduce the huge storage requirements demanded by the equation oriented approach. Another problem with equation oriented approach is the difficulty in setting up the

system definition. Whilst engineers encounter problems in following the information flow during an equation oriented simulation and assessing where the problem occurs when the simulation has failed. These problems are now less severe because equation oriented simulators have now generally adopted the modular organisation strategy. Simulators such as SPEEDUP (Anon (1988)), generally possess improved diagnostic facilities which allow for better error tracebacks.

2.5 Simultaneous Modular Approach

The simultaneous modular approach is considered as a subset of the equation oriented approach (Perkins (1982)), this is because the equations describing the entire process are solved simultaneously. However, the simultaneous modular approach contains several strategies different from the equation oriented and the sequential modular method. The model equations are contained in modules in a similar manner to the sequential modular approach. However, unlike the equation oriented approach the simultaneous modular approach uses two levels of calculation and is sometimes referred to as the two-tier method. The first is termed the module level which involves the representation and subsequent solution of simplified and usually linear models to generate an approximate Jacobian matrix. The second level computation is the flowsheet level which involves the simultaneous solution of the equations and some or all of the connection equations. The linear models generated in the module level contain coefficients which are calculated by perturbing the sequential modular models. Simultaneous modular simulators differ in the way the Jacobian matrix is generated in the module level computations, the numerical methods used for the flowsheet computations and the mechanism by which unit connecting streams are calculated. Examples of simultaneous module simulators include ASPEN (1979) and SIMMOD (Chen and Stadtherr (1985)). Joulia and Kohert (1985) have also experimented with the simultaneous modular approach by adding appropriate convergence modules to their sequential modular simulator. Jirapogphan (1980) attempted to utilise non-linear models in a simultaneous modular environment and concluded that the convergence was improved in steady state simulations and optimisation problems. Pierucci *et al* (1982) confirmed

the improved convergence obtained by using nonlinear reduced models. Chen and Stadtherr (1985) observed a considerable improvement in optimization problems by using the simultaneous approach over the sequential modular method - whilst Trevino-Lozano (1985) modified ASPEN to incorporate a simultaneous modular approach and called it ASPEN plus.

The simultaneous modular approach attempts to combine the best features of the sequential modular and equation oriented approaches, hence overcoming some of the deficiencies associated with the other methods. The main advantage over the sequential modular approach is that computationally intensive control loops used in design calculations and optimisation are no longer required. Existing sequential modular models can be used to generate the linearised model coefficients, which become redundant in the equation oriented methods.

2.6 Concluding Remarks

Extensive research carried out in process simulation over the last thirty years has meant there has been a considerable advance in chemical processes being more realistically simulated. Various reviews of process simulation have been undertaken during this period, highlighting studies that aided this general advance. Sargent (1967) discussed computer aided design encompassing steady state and dynamic simulation. Rosen (1980) reviewed the field of steady state chemical process simulation. This subject was discussed more comprehensively by Westerberg (1979). Whereas, Shacham *et al* (1982) discussed the application of the equation oriented approach to process flowsheeting. Cameron (1981) reviewed the field of dynamic simulation of chemical processes. Biegler (1989) discussed some general concepts adopted in process simulators. He also illustrated many commercially available chemical process simulators.

The sequential modular approach was adopted in the early chemical process simulators in the 1950's. This was because stand alone programs representing various unit operations were connected in a similar manner to the material flow in an actual process. The model equations are then organised in modules. The simulation progresses through each model in turn, enabling special purpose solution strategies to be constructed for each different unit type. There has been a reluctance in industry to switch from the sequential modular technique to the others available because of the large time and cost investment incurred in the development of model libraries. Although, a lot of effort has been expended on this method, a number of problems still remain unresolved. These include the excessive computational expense in the simulation of systems with nested iteration loops for recycles and design specifications which add further loops. In dynamic simulation an added problem occurs with the degree of coupling between the equations in the various modules.

The simultaneous methods can overcome most of the problems associated with the sequential modular approach. The simultaneous modular approach can utilise the existing sequential modular modules unlike the equation oriented approach. However, the use of

simplified models in the simultaneous modular approach exacerbates the problem of accuracy. As a great deal of work in numerical techniques has been carried out to increase the accuracy of simulation, the use of simplified models will go against this work.

The equation oriented approach like the simultaneous modular approach generally attempts to solve all model equations collectively hence overcoming the difficulties of design problems occurring during steady state simulation when using the sequential modular approach. Whilst, the problems of equation coupling in sequential modular approach is also overcome by using the simultaneous methods. As the equation oriented approach solves the full model equations and not linearised equations the flowsheet is more realistically represented.

However, the equation oriented approach has certain weaknesses including: seldom error free coding of the complete set of model equations representing the whole chemical process in one large block. This block is also difficult to debug. These weaknesses have been overcome to some extent by using a modular framework within the equation oriented approach. The resulting modularly organised equation oriented approach requires complex executive routines to generate the equation set from the chosen models.

Several methods have evolved from the methods already discussed. Ponton and Vasek (1986) describe a two level approach to dynamic simulation. Whereby the modelling of interactions between flows and pressures with small time constants can be modelled together with the models with large time constants which are encountered when modelling the details of composition changes and vapour liquid equilibria. This in effect is the problem of stiffness of differential equations and their method is an effective technique of overcoming the stiffness arising from the modelling of processes with widely different time constants. The equation oriented methods use well developed numerical methods to overcome this problem. Liu and Brosilow (1987) suggest the use of parallel processing in the modular simulation of dynamic systems by using suitable integration algorithms for each subsystem. This technique offers the possibility of computational speed by parallel processing of individual subsystems. The most

efficient integration algorithms for each subsystem can be selected, this also aids computational speed. However, difficulties can arise in co-ordinating the calculation of the different subsystems to ensure time synchronization. Coordination routines are therefore required to keep the dynamic modules in time synchronization.

The equation oriented approach within the modular framework is a method which overcomes the problems encountered with the other methods and gives a more accurate representation of the flowsheet. The problems still remaining with this method are the robust solution of the algebraic equation set during steady state solution and the effective definition of the problem. Once these problems have been solved this technique should provide an excellent mechanism for chemical process simulation, as it offers greater speed and flexibility for complex processes and controlled simulations than the sequential modular approach. The coupling problem encountered in the sequential modular approach is also overcome hence giving more representative results.

CHAPTER THREE

STRATEGIES AND STRUCTURE OF DASP I IN COMPARISON WITH OTHER APPROACHES

3.0 Introduction

DASP I was developed using the equation oriented approach, the reasons for selecting this method are described in Chapter 2. The equation oriented approach is, however, more difficult to implement than the sequential modular approach (Hlavacek, (1977)). One of the reasons for this added difficulty is that the executive routine for the equation oriented simulators is more complex, because of the extra duties performed by this executive routine. These duties include generating the equations from the chosen models and then controlling the solution of the resulting equation sets.

DASP I is structured modularly, in that the simulation options, such as dynamic and steady state simulations, event processing and the terminal section, are each controlled by a co-ordinating routine. The executive routine calls each of these co-ordinating routines when required. Consequently, new options can be added and removed. A model library is provided, with the models in a standard format. These models are lumped parameter models, which means that variables are relatively uniform throughout a control volume, when a process is divided into a number of control volumes of finite size or lumps. The model equations representing each unit are written in a format which allows these equations to be used both in steady state and dynamic simulation.

An event processor is present and allows events such as opening and closing of valves to be simulated. This enables the highly unsteady plant start-up and shut-down procedures to be mimiced. A change in flowsheet usually results when certain valves are opened or closed in a flowsheet. Hence, it is an essential feature for an effective dynamic simulator which can handle such highly transient operation and are very common in the batch processing facilities.

Dynamic simulators must provide facilities for disturbances to be entered to the system to determine the behaviour of the chemical processes. DASP provides the user with access to all the variables, parameters, simulation control information and the problem description during the simulation. This allows the user to view the simulation information at specified points during a run and modify the problem as and when required. Three different perturbation functions are provided, these are a step, a ramp and a sine wave.

The correct solution of model equations is one of the major tasks for a chemical process simulator. The equation oriented approach adopted in DASP I solves all the model equations simultaneously. Consequently robust numerical methods are required to solve this set of equations efficiently. Some well proven numerical techniques are utilised by DASP I to ensure the efficient simultaneous solution of the model equations. These are discussed in Section 3.1.

The information describing the process, supplied to most simulators is mostly numerical, difficulties can therefore arise when debugging is required. Different strategies are used in simulators to reduce problems in generating and debugging of the problem description information. DASP I requires the user to provide most of this information in data files. Others such as SPEEDUP (Pantalides (1986)) use special input languages, which the user must learn.

This chapter describes the basic features of DASP I and compares these features with those available in other research chemical process simulators to assess their capabilities and limitations. Hence the requirements of a useful simulation system can be established.

DASP I was developed on an ICL PERQ utilising a version of the UNIX operating system called PNX. It has however, been run on a number of machines including a DEC VAX and an IBM PC AT, with slight modifications. Ogbonda (1987) consequently described DASP I as highly portable.

3.1 Numerical Methods

In the equation oriented approach a set of equations must be solved simultaneously. Robust numerical methods are required to solve the equations where this is possible and indicate clearly any faults in the equations. With dynamic simulators the system equations must be integrated. The solution is not continuous but provided at intervals over the time of simulation. There must therefore be numerical integrators robust enough to do this reliably.

In steady state simulation a set of non-linear algebraic equations must be solved to obtain the steady state. These equations which include physical property approximations are solved simultaneously. As these equation sets can easily contain 2000 equations for a realistic chemical process, a method that is robust for large sets must be used for their solution. The solution of non-linear algebraic equations is discussed in Chapter 4.

3.1.1 The solution of differential equations

The differential and algebraic system that is solved is described in Section 2.1.

One of the major problems with solving differential equations is the large differences in the time constants of variables within an equation system. This type of system is called a stiff system. The difficulty arises in determining the correct step size, which must be small enough to prevent the system from going unstable, but large enough to solve the system in a reasonable amount of time. Two different types of numerical methods exist: explicit and implicit integration methods. With the explicit methods the unknown differential variable appears only on one side of the equation and can therefore be easily calculated. Examples of the explicit methods include the Euler and the Runge Kutta methods. On the other hand when the unknown variable appears on both sides of the solution equation the technique is called the implicit method.

Examples of this approach include the Adams-Bashforth and Gear's methods. Explicit methods are very inefficient for stiff systems because very small time steps must be used. The

implicit formulas are very useful for overcoming the stiffness problem. This is because the time step size chosen for the integration is subject only to the issue of accuracy and not stability, hence there is no constraint on the maximum step size used. A great deal of research in the field of implicit integration methods has yielded the semi-implicit Runge-Kutta methods (SIRK), and the backward differentiation formulas (BDF) implemented by Gear (Gear (1971)), as the most popular methods. Smith (1985) tested various integration techniques including two forms of BDF: EPISODE (Hindmarsh and Byrne (1975)) and GEAR (Gear (1971)) together with a SIRK method and some explicit methods. He concluded that the predictor corrector multistep methods were the most efficient. DASP I utilised DASSL (Petzold (1983)) as the integrator it implements is an efficient BDF method and has been successfully tested by several researchers.

3.2 Model Format

Many different strategies are used in dynamic simulation packages to formulate or layout the mathematical model equations representing the process units. ASCEND II (Kuru (1981)) and QUASILIN (Smith (1985)) both introduce a new variable for every differential equation. These new variables are termed velocity variables -V. This strategy increases the dimension of the system with an extra variable and an extra equation for each derivative term. The model equations (2.1) must be manipulated to state variable form:

$$\begin{aligned} X - V &= \emptyset_m \\ V - f(x,y) &= \emptyset_m \\ g(x,y) &= \emptyset_n \end{aligned} \qquad 3.1$$

DASP I implements another strategy for model simulation, which imposes fewer restrictions on the model. The equations can have more than one derivative term with variable coefficients. These equations do not need to be manipulated to be brought to state variable form as in other formulations. An important feature of this approach is that the formulation provides a consistent method of writing the equations which means there is no distinction between

differential and algebraic equations. The same model equations can also be used for both steady state and dynamic simulations by equating all the derivative terms to zero for steady state simulation.

The unit modules are written in standard FORTRAN 77 and are therefore compiled and linked into a library. However, DASP I must have the unknown variables in each equation predefined, hence if a different variable is required to be calculated from within the equation, the model must be rewritten. Therefore experimentation with the simulation may be difficult as recompilation of the models is required. This is a major drawback with the implementation in DASP I and is discussed further in Chapter 6.

3.3 Event Processing

Chemical engineering processes undergo behaviour which is highly transient during certain periods of operation. Events occur during these transient operations which include start-up and shut-down procedures, the attainment of emergency states such as bursting discs or blowoff valves being activated and plant trips. Events are defined as points in time beyond which there is an abrupt change in a particular state. Two types of events can occur the first being a time event, which occurs at a predefined time. This time of event is commonly used for describing the forcing function applied to a process. The second type of event occurs more frequently in chemical engineering processes, and is called the state event. This type of event occurs when a predefined state is attained. Examples of state events include the attainment of a particular temperature, pressure level or concentration within a process. Events are a form of discontinuity during the integration, with the time event being an explicit discontinuity whereas a state event is an implicit discontinuity. It is obvious that a state event is much harder to accommodate than a time event. This is because the simulator must detect whether a particular state has been attained and then locate the exact time of occurrence of this event.

3.3.1 Event Processing in DASP I

DASP I can handle events which have been predefined in a particular sequence. At present events occurring randomly cannot be accommodated. This is not a realistic representation of chemical processes and is an obvious limitation in DASP I.

A state event is located by determining if an unknown state variable has crossed a threshold value during each predictor evaluation during integration. Once this has occurred the variable coefficients are used to determine the time of the event using an iterative technique. Once an event has occurred control is returned back to the user. The user can then impose the required change to the flowsheet as required. This change can occur to a variable or parameter within a particular unit. This modification is carried interactively by issuing the unit number and the variable/parameter number. Another important modification can be made to the flowsheet. This can be in the form of disconnection or reconnection of a unit in the flowsheet. Units such as valves and drums can be removed from the flowsheet by specifying the unit number during interaction with the program. The user must then reconnect the loose streams to other units within the flowsheet. Reconnection of a unit is the exact opposite procedure to the disconnection. This facility is extremely useful in simulating start-up and shut-down procedures as well as batch processing facilities.

3.4 The Problem Description

The problem description is the data that is required for the definition of the chemical process that is to be simulated. It is a collection of information including the flowsheet or topology, the unit parameters the component information feed stream data and the simulation control data. The correct definition of the problem description is an essential factor in achieving a successful simulation. It is probably the most difficult operation for the user to perform.

Simulators achieve the generation of problem description by various methods. In DPS (Thambynayagom *et al* (1981)) the data input to the program takes place in an interactive session on the terminal. Although, it is very simple for non experts to use it may become tedious for regular users who become familiar with its operation, especially when it is used for simulation of any realistic chemical plant. SPEEDUP (Pantelides (1986)) uses a data base structure to enter all the input information including the problem description. However, it is limited in its interaction. The user is required to provide any models that do not exist in the standard library together with problem definition using a specially developed input language. This input language is based on the high level computer language, PASCAL. The data is provided in well defined sections which reflect the natural structure of the simulation problem. In each section the data must be provided according to the rules of the input language. The problem description that has been entered into the program is translated into FORTRAN. This FORTRAN source code is then compiled and linked with other code producing an executable program which is specific to one particular problem. Hence, if the problem description is modified in any way the code must be retranslated, compiled, linked and re-run. Even minor modifications, such as a change in value of a single parameter results in this retranslation procedure. It can be seen that experimentation with the simulation may be expensive both in time and computing costs. Another problem with this approach is that the user must first become familiar with the input language before using the package. QUASILIN (Smith (1985)) requires the user to supply an input file which defines the simulation problem to be solved and a control file which contains information defining the equation solving parameters. The input file is read by an input processor, which generates a program specific to the flowsheet and a data file containing information for the simulation. These two files are then used by the main QUASILIN executive together with the necessary units and physical properties from various libraries. Keywords are used in the input files to describe the problem. Another input file defines the graph that is required. Therefore this method also requires the user to have some knowledge of the keywords used in the input files.

DASP I uses a similar approach to QUASILIN for the problem description. This involves providing the description in ASCII files. However, several different files are used for providing the information required for the simulation. Although no keywords are required as in QUASILIN a pre-defined format must be used to enter the description

3.5 Concluding Remarks

DASP I has been developed as a portable semi interactive dynamic simulation package. It has a modular executive routine which is useful as it allows other features to be added or removed easily. The whole package including the models is written by using standard FORTRAN 77 and since most engineers are familiar with FORTRAN the models can be written and modified relatively easily.

The input data used to describe the problem is entered in ASCII data files in a standard format. As no translation of the problem description is required excessive computation expense is not incurred in setting up the files as in SPEEDUP. An added advantage is that knowledge of a special input language is not required. Unfortunately the user must know the correct set up for these files, as errors in the data file generation can lead to spurious results and/or failure. As a great deal of information is required, large processes may cause too many problems in setting up the correct definition.

Hence, if this process is automated the simulation in general becomes much simpler. A method for overcoming this limitation by automatically generating the correct input files is described in Chapters 5 and 6. The modelling formulation implemented in DASP I is an excellent method as it allows the same model to be used for steady state and dynamic simulations. However, the method currently used does not utilise the modelling formulation to its full potential, because the models must have the equations written with a predefined set of unknown variables which is rather inflexible for simulation experimentation. A new model layout and model solution strategy is described in Chapter 6 which overcomes this drawback.

An efficient integration package DASSL that has been tested by various researchers has been shown to be successfully used in DASP I by Ogbonda (1987), who tested it on several examples. The non-linear algebraic equations solvers implemented in DASP I are the Newton Raphson and Broyden methods. Both of these methods are locally converging methods, requiring good initial values to achieve convergence. As non-linear algebraic equations are required to be solved to obtain a steady state solution and also to solve the equations generated by the integrator, a robust solver must be used to assure convergence. There methods are discussed further in Chapter 4.

CHAPTER FOUR

THE SOLUTION OF NON-LINEAR EQUATIONS

4.0 Introduction

Models used to represent chemical engineering problems give rise to non-linear algebraic equations which must be solved using numerical methods to find the roots of the equations. A system of non-linear algebraic equations (NLAEs) must be solved to obtain a steady state solution for the flowsheet. This steady state solution is required in two instances, the first is when a solution is used as a design or flowsheeting calculation the other is when a steady state is needed as an initial starting point for an integration calculation. In DASP the differential models can be reduced to steady state models by equating all the derivatives to zero in the system.

Non-linear algebraic equations are generated during a dynamic run in DASP, as a step to solving the dynamic equations, as follows. In the method used, the derivative is approximated by a combination of the solution at the current and earlier times, resulting in a set of non-linear algebraic equations. These systems of equations are usually difficult to solve for several reasons. There may be several solutions to the system, not all physically feasible and some with complex number solutions, it is possible for NLAE solvers to converge to these infeasible solutions hence providing potentially incorrect results to the user. Another problem that commonly arises is that the functions are very non-linear and badly scaled. The non-linearity occurs because of the presence of products of variables, variables raised to powers other than one and also the existence of intrinsic functions such as logarithms for example, exponential functions involving temperature occur in detailed modelling of chemical kinetics. Bad scaling occurs when variables or coefficients differ in size by orders of magnitude. Scaling can be improved by transforming the coefficients of a system so that they do not have such different orders of magnitude. Such scaling reduces the possibility of arithmetic problems when a solution is sought. This can be illustrated by the following equation.

$$10^{20}\ x^2 + 3.10^{20}\ x + 2.10^{20} = 0 \qquad 4.1$$

which can be multiplied by 10^{-20} to give

$$x^2 + 3x + 2 = 0 \qquad 4.2$$

This type of scaling is called equation scaling. Another form of scaling can be used to simplify an equation, this is termed variable scaling and can be shown by equation (4.3).

$$10^{-20}\, x^2 + 3x + 2.10^{20} = 0 \qquad 4.3$$

which is scaled by substituting

$$x = 10^{20}\, z \qquad 4.4$$

to give

$$z^2 + 3z + 2 = 0$$

The coefficients of the system are redefined by scaling the variables and equations in the system. However, the actual structure of the system and the unscaled variable values will remain the same. The solutions will initially be in scaled units. The original units can be recovered. For realistic chemical processes the number of equations can be as large as 5000. Such sets of equations are usually sparse, any one equation containing only a few variables. This means that the coefficient matrix has a majority of zero elements. Duff (1977) states that "a sparse matrix or system is one in which advantage can be taken of the percentage and/or distribution of the zero elements". A robust NLAE solver must be present in a simulation package to ensure convergence.

Many different numerical methods are available for calculating the solution of a set of simultaneous NLAEs. They are based on successive approximations by applying a set of rules so the initial estimate of the solution, and, often also to estimates of partial derivatives of the equations with respect to the problem variables. The most common technique implemented in chemical engineering process simulators is the Newton Raphson method. This method is very efficient, in offering second order, or quadratic, convergence, when a good initial guess can be

supplied. However, it has several disadvantages associated with it. Consequently a great deal of work has been undertaken to find alternatives by a number of researchers. One of the most severe limitations is that the successive approximations may diverge away from the solution if the initial starting point is too far away from the solution. This can result in the method finding one of the physically infeasible solutions for the particular equation system. The method also fails to converge when the Jacobian matrix, i.e. the matrix of partial derivatives of the system, becomes singular. This means that the tangents at the solution are zero, hence the progression of the iterations breaks down. Methods involving the Jacobian matrix can become computationally intensive if the Jacobian matrix is evaluated and inverted at every iteration. It can be seen that for a system with 2000 variables and equations can result in a Jacobian matrix with 4 000 000 elements and is obvious that evaluation of such a large matrix would be computationally expensive. In fact, the sparse nature of the equations means that most of the terms are zero.

Researchers in this field have developed many methods which are claimed to be superior to the Newton Raphson method (Powell (1970), Westerberg and Director (1981)). However, extensive testing by many other workers, including Sargent (1981), Hiebert (1982) and More *et al* (1981) showed that the theoretical superiority was not in fact transferred to practical superiority. Broyden's method (Broyden (1969)) is one such technique developed to overcome the expense of Jacobian matrix evaluation at every step in the Newton Raphson method, by approximating the Jacobian at the first iteration and then updating this approximation at every iteration. Hence, it saves computational effort for large systems, but can offer only super linear convergence unlike Newton Raphson which possesses quadratic convergence (Conte and de Boor (1980)). It also requires the starting point to be relatively close to the solution to ensure convergence.

Other researchers have looked at methods of achieving convergence even when the initial starting point was not close to the solution. An example of this is the development of Powell's dog leg method (Powell (1970)). This method combines the steepest descent method, which

has good global properties and is used to predict the first part of iteration path, with Broyden's method which is considered to be an efficient locally convergent technique. Hence, the result is that the area of convergence is expanded. However, Cosnard (1975) stated that a test used to compare NLAE solvers showed that Powell's method failed to solve many problems. This was confirmed by Shacham (1985) who tested five different codes implementing different algorithms on chemical engineering problems and concluded that codes implementing Powell's algorithm were unpredictable and unreliable for these problems. Sargent (1981) discovered that Powell's method had problems with equation sets that possessed badly scaled variables and functions. He also reported that Powell's method was sensitive to random errors or noise in the function evaluation.

Non-linear algebraic equation solvers can be classified into three broad categories:

1. Locally convergent methods - hence requiring a good starting initial estimate to achieve convergence. This includes methods such as Newton Raphson and most of the Quasi-Newton techniques which includes the Broyden (Broyden (1969)) and Schubert (Schubert (1970)) methods.
2. Expanded region of convergence methods - these methods have a larger area of convergence than the locally convergent techniques. Examples of methods offering this type of convergence are Powell's dog leg method and Levenberg - Marquardt (Shacham (1986)) method which also overcomes the problem of singular Jacobian matrices.
3. Globally convergent methods - these methods are newer than the other methods but are usually more complex and are theoretically able to converge to the solution from any starting point. Examples include differential arc length homotopy methods (Seader (1985) and Burton and Morton (1987)).

A great deal of mathematical research has been undertaken with the latter class of methods because of the obvious attractions. The techniques developed are generically called continuation methods. As digital computing power has rapidly increased and unit computing costs have

fallen, the more efficient locally convergent methods are no longer the most effective to use. This is because more time has to be spent on initial analysis to produce a good initial estimate of the solution to ensure convergence. Instead, the globally convergent method can do this by computation. The basic idea behind continuation methods is to solve a series of problems as a parameter is slowly varied, using a locally convergent iterative technique for each problem, and the solution to the previous problem as a starting point for the current problem. This work has lead to the development of differential equation continuation formulation in various forms by Kubicek (1976), Georg (1981) and Rheinboldt (1982). Despite this work these continuation methods experience difficulty when the Jacobian matrix becomes singular and fail. Another advance by Chow *et al* (1979) was the development of probability one homotopy methods, which overcame the problem of singular Jacobian matrices as they were constructed not to have any singular points. Probability one refers to the supporting theory, which states that for most choices of a parameter vector in the homotopy map, there are no singular points and the method is globally convergent. These methods have been successfully tested on problems which caused difficulties for quasi Newton methods as in Dennis and Schnabel (1983). Morgan (1987) has demonstrated the use of this method in chemical engineering applications.

4.1 Variable Initialisation

From the preceding discussion it can be seen that the major problem in the solution of NLAEs is the provision of an initial starting point. Depending on the method implemented it can be easy or difficult to provide a sufficiently good initial estimate for convergence. For instance it is vitally important for the initial estimate to be close to the solution to achieve convergence when using locally convergent methods. Whereas with the expanded region methods the initial estimate does not have to be as close. However, these methods are less efficient than the locally convergent methods. This situation is more exaggerated in the globally convergent methods, where the starting value can theoretically be anywhere, although in practice this is usually not the case but the methods are computationally intensive. However, in chemical engineering applications and particularly in process simulators the most popular methods are those

possessing locally convergent properties. Consequently in simulators either a good initial estimate of the solution must be provided by the user or a method of automatically generating a good default initial estimate must be installed in the simulator. The former strategy is unreasonable, because a realistic model of a large chemical process has many hundreds or thousands of variables and supplying good initial estimates for these variables would be extremely difficult. Most simulators therefore provide a facility for some sort of default variable initialisation.

In SPEEDUP (Pantelides (1986)) the user declares types of variables; for example all the temperatures can be declared as type TEMPERATURE. All the variable types are then assigned upper and lower bounds and an initial starting point. This can be further augmented by giving estimates and bounds for specific variables for which the user can provide better estimates than those provided by the particular variable type. The bounds can represent physical bounds, such as non-negative pressures, or mathematical limits which can declare the domain of definition of functions. The numerical methods implemented in SPEEDUP are modified to ensure that bound violation does not occur. Although this technique is a good idea and has been successfully used by Paloschi (1982), it is very easy to overconstrain the system which then fails to converge. On the other hand it is possible that many solutions exist for the problem with some of them being infeasible within the bounds imposed on the variables.

Variable initialisation in QUASILIN (Hutchinson *et al* (1986)) is achieved by converting non-linear equations into linear ones for the first iteration. However, initial estimates can be specified for all the process variables if available. Some of the linearised equations require initial guess data to aid the generation of a good starting point. Unfortunately, accurate representation of the non-linear equations as linear equations can in some instances be very difficult. In ASCEND II (Westerberg and Bengamin (1985)), for each model the library creator decides on a minimal set of tear variables to be guessed leading to the calculation of the rest of the variables. Hence, with the stream variables known and the guesses for the tear variables, the output variables are determined much the same as in sequential modular simulators. This process is

carried out for each unit model within the flowsheet. However, this method requires an initialisation routine for each process unit.

Pagani *et al* (1989) implement a different strategy with their CHEOPE flowsheeting package, for variable initialisation. This scheme involves taking a single step of the steepest descent algorithm from default values of the variables. Then a Newton Raphson scheme is used for final convergence.

During the iterations the Jacobian matrix is transformed into a block triangular form, by for instance, setting all the recycle stream variables as constants. This approach is similar to the one used in ASCEND II which tears variables in a sequential fashion. However, this strategy is broadly similar to Powell's dog leg method with a steepest descent method used in conjunction with a Newton Raphson method. As already discussed Powell's method has been shown to possess unreliable convergence characteristics.

Westerberg and Benjamin (1985) discussed another technique for initialisation involving the evolution of the models from simple to final complex ones. The complete flowsheet is solved with simple models that for instance consider only material balance and then perhaps add the energy balance and recalculate the complete flowsheet. Although, Westerberg and Benjamin (1985) have found that this approach has worked from simple defaults and unit-by-unit initialisation it may not be as simple when there are units in the flowsheet that contain simultaneous heat and material balances. This method may also run into problems when the complexity of a large flowsheet is increased by the addition of say the energy balance, this may double the number of equations in the system and hence cause difficulties for the NLAE solver. A better scheme would be to evolve the flowsheet, in that several units may be solved first, then other units can be gradually added until the complete flowsheet is built up. In fact this strategy would be useful in most simulators as debugging is made much easier.

In DASP I (Ogbonda (1987)) the variable values must be estimated for each unknown. This has obvious limitations especially for large processes. Fortunately, DASP I uses a predefined list of variables for all the models hence a default strategy can be easily implemented.

4.2 Non-Linear Equation Solvers in Simulators

The Newton Raphson method is the most popular non-linear algebraic equation solver in chemical process simulators. CHEOPE (Pagani *et al* (1985)) implements such a modified method which reduces the step length in the Newton direction if constraints are violated or residual error increases. The Newton Raphson algorithm fails when the Jacobian becomes singular. This situation can arise when some equations become linearly dependent on the remaining ones. This solver attempts to overcome singularities by using a modified linear equation solver to solve only a subset of the linearly independent equations to obtain a suitable Newton direction. The discarded equations are re-included at the next step by the program. SEQUEL (Stadtherr and Hilton (1982)) utilises two Newton Raphson schemes; the first being the standard linearisation, the other utilises step size relaxation. A hybrid linearisation blending first and second order linearisations is also included. The non-linear terms are linearised by holding all but one of the variables in the term as constants, this selection was made at program development. This is a first order linearisation, and is transferred to second order or the Newton scheme by using a blending parameter. Hence, the first order is used far away from the solution and the second order close to the solution. The difficulty with the method is selecting the blending parameter and its rate of change. ASCEND II (Locke (1981)) also utilises a Newton Raphson method with flowsheet evolution. In SPEEDUP (Pantelides (1988)) the sparsity of the system is exploited by decomposing the problem by partitioning the equation set and variables in blocks. Then a NLAE solver is applied sequentially to each of the blocks. Three techniques are provided for solving the NLAEs, they are:- Newton, Schubert and Hybrid. They differ in the way the Jacobian approximation is obtained and maintained during solution. In the Schubert method the Jacobian is initialised at the first iteration by using finite-differences, whilst the other two methods generate the Jacobian using an automatic algebraic manipulation (AAM) method

which automatically generates analytical derivatives, and finite differences. The AAM is performed using symbolic operations in a recursive manner by applying the rules of differentiation. As system equations can be provided in procedures in addition to the normal equations, the form of these equations is not available for AAM. Hence, the partial derivatives are approximated numerically using finite differences or secant type approximations. During subsequent iterations the Schubert method implements a least change second update to all the non zero elements of the Jacobian. Newton's method uses the same method as the first iteration, whereas in the hybrid method the analytically available elements are recalculated and a least change secant update is applied to the remaining non zero elements. The decomposition of the flowsheet is a good method for reducing the sparsity of a problem but other sparse matrix methods are still required in the package to completely address the problem of sparse matrix manipulation. Also the vast majority of equations are usually in one large non-linear block. Consequently, the NLAE solver must still be powerful enough to handle these large blocks. Although, the AAM is an excellent method of providing most of the Jacobian, the locally convergent methods implemented in SPEEDUP must still have good initial starting estimates to ensure convergence.

Burton and Morton (1987) implemented a differential arc length continuation method in an equation oriented simulator. They were unable to solve some chemical engineering problems from arbitrary starting points. Problems can occur if the step length is not closely monitored and another path is followed or a reverse direction is followed. They concluded that there was some improvement over Newton's method but the greatest benefit was the potential of finding multiple solutions of an equation set. Also infeasible specifications can be identified and the limit of feasibility established.

DASP I currently implements Broyden's method and Newton's method to generate a steady state solution for a problem. The user sets a flag to define the NLAE solver that is used for the solution. As both of these methods are locally convergent techniques the starting

estimate must be close to the solution to ensure convergence. Consequently, more robust method must be provided to ensure convergence.

4.3 Non-Linear Algebraic Equation Solvers for DASP II

Earlier in this chapter it has been seen that the Newton method and its derivatives are the most frequently used NLAE solvers in simulators. Since these methods are locally convergent, good initial estimates must be provided to aid convergence. As discussed in Section 4.1, simulators must provide means by which a good starting estimate can be automatically generated. Although, various methods are utilised none of them guarantee a good starting point with global convergence. CONSOL (Morgan (1987)) and POLSYS (Watson (1987)) are programs that offer global convergence for polynomial systems, other equation types must be converted to polynomial form. These codes require no starting points and solve for all possible solutions. It can be seen that this method would be extremely valuable for providing a steady state solution which can be used to solve flowsheeting problems and as an initial starting point for dynamic simulation. This method is described further in 4.5.

Three different convergent strategies are possible, as described in 4.0 they are locally convergent, expanded region of convergence and globally convergent methods. However, methods with these properties are all prone to certain disadvantages and consequently it is difficult to say which method is preferable to the others. For instance, locally convergent methods offer the best speed of convergence but require the best starting estimates. The globally convergent methods does not require good starting points, but are the least efficient. CONLES (Shacham (1986), (1985)) is a solver that contains three methods, each offering convergence properties of one of three classes described earlier. This package is described in section 4.4.

4.4 The CONLES Non-Linear Algebraic Equation Solver

Hiebert (1982) evaluated several available codes for solving a set of various test functions. She concluded that such software should handle constraints such as non-negativity and simple

bounds on the variables. CONLES (Shacham (1986)) can handle two types of constraints. The first type is termed "Physical constraints" where the variables must be greater than or equal to zero only at the solution. The second type is called "Absolute constraints" where the variable values during the calculation cannot become less than zero. Shacham (1985) concluded that the step length restricted NR is effective in solving problems with absolute constraints.

CONLES contains three different methods and uses the strengths of each to create an efficient NLAE solver. For example it uses the speed and efficiency of the Newton Raphson method. When the Jacobian matrix becomes singular or even nearly singular the Levenberg Marquadt method is used. The third method is the continuation method which offers an extended region of convergence. Each of these methods is discussed in more detail.

4.4.1 Newton Raphson Method with step length restriction

This method linearises the set of NLAEs which are then solved by a suitable linear equation solver. The Newton Raphson uses an iterative sequence starting from the initial estimate X° and by solving the linear system.

$$J\ (X_m)\ \Delta X_m = -\ f(X_m) \qquad 4.9$$

where

$J\ (X_m)$ is the Jacobian matrix, evaluated at X_{mj};ΔX_m is the correction which is calculated by:

$$X_{m+1} = X_m + \lambda_m\ \Delta X_m$$

where

λ_m is the scalar step length.

If λ_m is set to 1 the method becomes the classical Newton Raphson. The step length restriction is used to prevent absolute constraints being violated. The λ_m is calculated from:

$$\lambda_m = \min_j \left[\frac{\alpha X_j}{\Delta X_j} \right] \qquad 4.10$$

where α is a number smaller than, but close to 1. j are the indicies of all the variables for which absolute constraints have been specified for which $\Delta x_j < 0$. α is used to ensure that an absolutely constrained variable does not become equal to zero.

The Newton Raphson method offers quadratic or second order convergence, i.e.

$$\lim_{M-\alpha} \left| \frac{Xm+1 - X^*}{Xm - X^*} \right| = K \qquad 4.11$$

where K is a constraint, X* is the solution and ||·|| is a suitable norm. Consequently, near the solution the number of correct significant digits of the variables is multiplied by two in every iteration.

However, Newton Raphson has only locally convergent properties and may fail if the initial estimate is not close to the solution. If the Jacobian matrix becomes singular the Newton Raphson method breaks down as equation 4.9 cannot be solved for ΔX_m. This problem can be overcome by combining this with the steepest descent method. The combined method is called the Levenberg-Marquardt method (Shacham (1985)) and is described in the next section.

4.4.2 The Modified Levenberg-Marquardt Algorithm

This algorithm solves a linear equation for the correction step

$$\left[\mu_m I + J(X_m)^T J(X_m) \right] \Delta X_m = - J(X_m)^T f(X_m) \qquad 4.12$$

where I is the unity matrix a diagonal matrix with positive elements and μm is a non-negative scalar. This scalar is used to control the direction and length of the correction vector. When μm is equal to zero the length of the correction vector and step length becomes the same as the Newton Raphson. When it becomes the dominating term, the direction is the steepest descent

direction and the length becomes very small. By selecting an appropriate μm the Levenberg Marquadt method converges to a local minimum of F (x). However, it is not certain that F (X) is a zero of f(x). Hence, a value of μm is selected so that the singularity in the Jacobian matrix is eliminated and that moving away from the near singular region at a reasonable rate is possible. In CONLES this scalar is selected so that:

$$\mu_m = \max \left| J (X_m) \right| \qquad 4.13$$

4.4.3 The Continuation Method

Methods using continuation or embedding methods are considered to have global convergence properties (Sieder (1985), Garcia and Zangwill (1987)). In these methods the function f (x) is embedded in homotopy functions G(x,t). This homotopy parameter is slowly changed to follow a path from a starting point G(x,1) = 0 where the solution X is known until G (X,0) = f(x), i.e.

$$G (X, 1) = f_o (X),$$
$$G (X, 0) = f (X) \qquad 4.14$$

and the equation

$$G (X, \theta) = 0 \qquad 4.15$$

is to be solved.

The homotopy satisfying (4.14) has the form:

$$G (X, \theta) = f (X) - \theta f (x_o), \quad X \varepsilon D, \theta \varepsilon [1, 0] \qquad 4.16$$

Equation 4.15 is itself a NLAE and must be solved by a NLAE solver. As the system path is tracked with the previous value providing a good starting point the Newton Raphson is an ideal method. A sequence of values for θ is generated by

$$\theta_o = 1$$

$$\theta_{kh} = \theta_k - \Delta\theta \quad ; \quad K = 1, 2, ..., p-1$$

$$\theta p = 0 \qquad 4.17$$

when $\Delta\theta = 1$, a single step is used to solve the system and the continuation method reduces to the Newton Raphson method. Otherwise a curve is generated along which the residual values must decrease.

4.5 CONSOL - Continuation Method

CONSOL (Morgan (1987)) is a continuation method which solves systems of polynomial equations. The advantage of this algorithm is that it finds all solutions, real and complex, for the sets of polynomial equations. HOMPACK (Watson *et al* (1986)) is a package that can solve many different types of equations and one of its options is POLSYS which is basically the same as CONSOL but with a different path tracker. However the advantage of this method over the classical NLAE solvers is that initial estimate values are not required to ensure convergence, hence overcoming the major drawback of the classical methods.

Most chemical engineering problems are in the form of a system of n polynomial equations - fn, in n unknowns - Xm or can be converted to such a form. The continuation method comprises two steps:

1. Definition of the continuation system h.

   For example the following continuation equation

   $$h(x, t) = x^2 + atx + [tb - (1 - t) q^2] \qquad 4.17b$$

   for the following equation:

   $$f(x) = x^2 + ax + b \qquad 4.17c$$

where

h is the continuation function

t is the continuation parameter

q is a complex constant

f is the non-linear function

2. Choosing a numerical method for tracking the continuation paths.

Which usually increments the continuation parameter from 0 to 1 and solves the resulting continuation equation at each increment. Equation 4.17b becomes:

$$h(x, 0) = x^2 - q^2 \qquad 4.17d$$

at $t = 0$, and

$$h(x, 0.1) = x^2 + 0.1\,a\,x + [0.1b - 0.9\,q^2] \qquad 4.17e$$

at $t = 0.1$, with the solution of 4.17d being used as a starting point for 4.17e.

The equation system to be solved is:

$$f_1(X_1, X_2, \ldots\ldots, X_n) = 0$$
$$f_2(X_1, X_2, \ldots\ldots, X_n) = 0$$
$$f_n(X_1, X_2, \ldots\ldots, X_n) = 0$$

or more simply

$$\underline{f}(\underline{X}) = 0 \qquad 4.18$$

Each of the polynomial functions are sums of terms. The general form of a term is:

$$\text{TERM} = \alpha X_1^{M1} X_2^{M2} \ldots\ldots X_n^{Mn}$$

where α is the coefficient of the term and can be a complex constant, real or zero. M_i are non negative integers. The degree of the term is $M_1 + M_2 + \ldots\ldots M_n$. The degree of an equation is

defined to be the largest degree of its terms. The total degree of the system is the product of the degree of the individual equations. The Jacobian is defined by:

$$J_{r,\,s}(X) = \frac{\partial f_r}{\partial X_s}(X) \qquad 4.19$$

If X* is the solution to f (X) = 0 , then X* is singular if det (J (X*)) = 0. A solution is termed geometrically isolated if no other solution exists within a region around the point. A start system g (X) = 0 is chosen, so that the deg (g_j) = dj for j = 1 to n.

$$g_j(X) = p_j^{dj}\, X_j^{dj} - q_j^{dj} \qquad 4.20$$

The homotopy map is defined as

$$h(X, t) = (1 - t)\,\gamma\, g(X) + t\, f(X) \qquad 4.21$$

with a randomly chosen complex number γ. Equation 4.18 may have solutions at infinity, which force some of the homotopy paths to diverge to infinity as t approaches 1. This system can be transformed into a new system, which under reasonable hypotheses can be proven to have no solutions at infinity and thus bounded paths. The transformation H (x,t) is defined by homogenising h (x,t) with a new variable X_n+1 and then defining X_n+1 as a linear equation in the other variables. Therefore the transformation becomes:

$$H_j(X_1, \ldots\ldots, X_{n+1}, t) = X_{n+1}^{dj}\; h_j(X_1/X_{n+1}, \ldots\ldots, X_n/X_{n+1}, t) \qquad 4.22$$

for j = 1, n and take

$$X_{n+1} = L(X_1, \ldots\ldots, X_n) = \alpha_1 X_1 + \ldots\ldots + \alpha_n X_n + \alpha_{n+1} \qquad 4.23$$

where the αi are randomly chosen real or complex numbers. When the transformation is used, paths that previously diverged converge to solutions with X_{n+1}= 0. Each solution to G (X) = 0 defines a continuation path for H (X,t) = 0 leading to a solution to F (X) = 0, hence there are d

continuation paths. As t is incremented the Newton Raphson method is used to solve the system h $(X, \Delta t) = 0$. Newton Raphson is effective in this situation as the system is solved at $t_i - \Delta t$.

Scaling can be critical to the success of this method as it affects the effective evaluation of the polynomial system (equation 4.18) and the solution of linear systems using finite precision arithmetic. A general purpose scalar is used to scale the variables and equations to centre the coefficients about unity and minimize the variation within the equations. Chemical engineering model equation systems possess extreme scaling of the variables and must therefore be addressed by the numerical methods used to solve these systems. As each path is tracked, the algorithm can be expensive in terms of computation, hence the algorithm is not recommended for systems greater than 10 equations with 10 unknowns. However most systems can be reduced to much smaller systems. Reduction will decrease the total degree of the system, consequently the system will be faster to solve and generally more reliable. However, the structure of the original equation system is transformed into a new structure. This procedure must be undertaken manually although a symbolic manipulation program REDUCE (Hearn (1983)) can be used to simplify the system of equations.

4.6 Concluding Remarks

The provision of a steady state is crucial in simulation both for steady state simulation and for dynamic simulation. For steady state simulation a steady state solution is calculated to provide the desired result for a particular plant configuration. In dynamic simulation the steady state is used for providing an initial starting point for the integration. During the integration a NLAE solver is also used to solve the NLAEs generated by the integrator at each step. Hence, a robust solver must be provided to ensure the solution of NLAEs.

The methods for solving NLAEs have various difficulties, including: convergence to one of many multiple solutions, some of which may be complex, the functions are usually very non-linear and badly scaled, the system is usually large and sparse and consequently an iterative

procedure is required. The methods have 3 types of convergence properties: locally convergent, expanded region and globally convergent. However, each of these methods have their disadvantages. The most frequently used methods in chemical engineering are those with locally convergent properties but they require good initial values and may encounter difficulties when the Jacobian matrix becomes singular. Hence, simulators implementing these methods usually employ an automatic method for generating initial estimates . As a completely successful scheme of automatically generating these initial values has not been developed, work (Pagani *et al* (1989)) is still being carried out to find a robust mechanism. The expanded region methods include Powell's method which has problems with badly scaled functions and is sensitive to random errors (Sargent (1981)). The globally convergent methods can converge to a solution when the initial estimate is further away from solution but are usually slow. CONLES (Shacham (1986)) is a program with three methods each with one of the convergence properties discussed earlier. The expanded region Levenberg Marquadt method, overcomes the problem of singular or near singular Jacobian matrices, whilst the continuation method is used if the Newton Raphson method fails. Shacham (1983) has successfully tested this algorithm with a series of chemical engineering problems. CONSOL (Morgan (1987)) is a continuation method that requires no initial values and finds all solutions for a set of polynominal equations. Hence, it could be used to generate initial estimates for the CONLES algorithm, CONSOL has problems with systems with more than 10 equations as it is computationally intensive. However, the system can usually be reduced manually so that fewer paths have to be tracked. Although the increasing power of computers has helped in the use of more complex methods, it would be potentially faster to use computers with parallel processing facilities such as transputers when all the paths are tracked.

CHAPTER FIVE

A GRAPHICAL METHOD FOR GENERATING THE PROCESS TOPOLOGY

5.0 Introduction

Chemical engineers use process flow diagrams (PFD's) to represent the flows of material between the process unit. These diagrams show the process units and how each flow is connected. PFD's summarise the process information for design and performance calculations. They also provide a visual means of communicating the process plant arrangement. The design of a control system will result in much more information about the plant instrumentation. This can be added to the PFD to form a Process and Instrumentation Diagram (PID). PFD's and PID's provide much of the information required for the simulation of chemical processes.

Each process unit in the real plant must be represented within the simulation. This is achieved using process modules within the simulator, also referred to as flowsheet units. Each flowsheet unit is a model representing a complete process unit or a part of a process unit.

In this work, a graphical program developed by Preece *et al* (1987) is used to provide PID and PFD information for the DASPII simulator. The program is modified to output data in a file suitable for use by DASPII. This file is further processed by an interface program BTOPOL which uses a database developed using BTRIEVE (Anon. (1986) and see Appendix D) and generates a dialogue with the user to complete the information needed by DASPII.

5.1 Methods of Topology description

The connections between the process units must be entered into the process simulator. This information is available in the PFD and PID diagrams. The information must be converted into the form required by each simulator for the simulation to be possible. A decision must be taken about each process unit as to which flowsheet unit or units will represent it. This modifies to PFD, introducing extra flows where a process unit is represented by more than one flowsheet unit. One way of achieving this step is to construct a block diagram illustrating the connectivity of the flowsheet units from the information on the PFD, or PID if the control system is also to be simulated. An example of a block diagram is shown in Figure 5.1. Such a block diagram is an aid to understanding, showing the decisions taken to modify the PFD or PID topology.

Different chemical process simulators need the process topology in a variety of formats. For instance, in SPEEDUP (Pantalides (1986)) the user must describe the topology in its special input language where each output from a unit must either be connected to an input of another unit or be declared as a product. Streams that terminate at a unit but not emanating from another unit are declared as feeds. Control signals are passed through special streams called connections. Once the flowsheet topology is created it is translated. During the translation SPEEDUP ensures that models exist for all the units declared in the flowsheet, stream input/output connections are compatible and that streams are not used more than once. This error checking is a useful feature of SPEEDUP although the user must be familiar with the input language. The user is also required to know the stream variable details, i.e. how many and which variables are passed in each stream, the number of streams in each model and the stream types to ensure compatible stream linkages. As this information is kept in different sections of the problem description the task of correctly constructing the flowsheet can be onerous, especially if the different sections of the problem description and the models have been written by somebody else.

In DASPI (Ogbonda (1987)) the flowsheet topology is described in one of the input files, the topology file. This is essentially numerical. Each flowsheet model has predefined stream connection types. The user is then required to define the units connected to each unit by its predefined stream connections. For instance in Figure 5.2 the connection information for unit 2 is as follows:

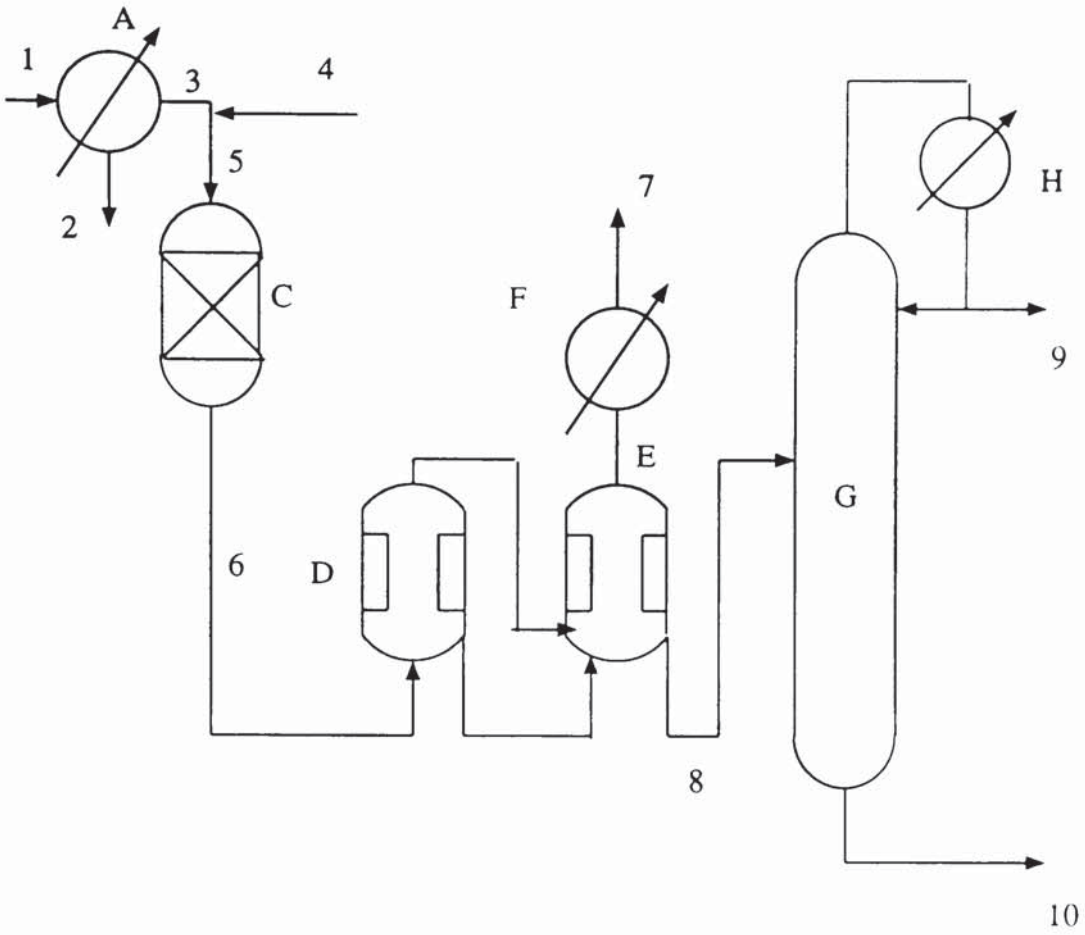
3 (number of connections)

1, 3, 4 (input 1, output 1, input streams)

The process topology information for DASP I is provided in an ASCII text file. No keywords are required in the file. As the data is mostly numerical, difficulties are encountered in deciphering an old flowsheet topology file without its associated block diagram. The stream details are also included in this file.

A similar method is used in SEQUEL (Stadtherr and Hilton (1982)) where the user creates an input file by a text editor. The data is prepared in a fixed format. As in DASP I most of the information supplied is numerical. The streams and units are assigned unique numerical codes which are then used to describe the flowsheet connections.

An intermediate method of flowsheet description to that used in SPEEDUP and DASP I is used in QUASILIN (Hutchinson (1986)). In the QUASILIN approach the flowsheet is described in a text input file as in DASP I but special keywords are used which are similar to the input language used in SPEEDUP. This input file is read by an input processor. The input processor generates a main program and unit calling routine which are specific to the problem, and a data file.



A – Partial Condenser
B – Mixer
C – Hydrolysis Reactor
D – Evaporator
E – Evaporator
F – Heat Exchanger
G – Distillation Column
H – Condenser
K – Splitter

Figure 5.1a Flowsheet for a section of an ethylene glycol plant

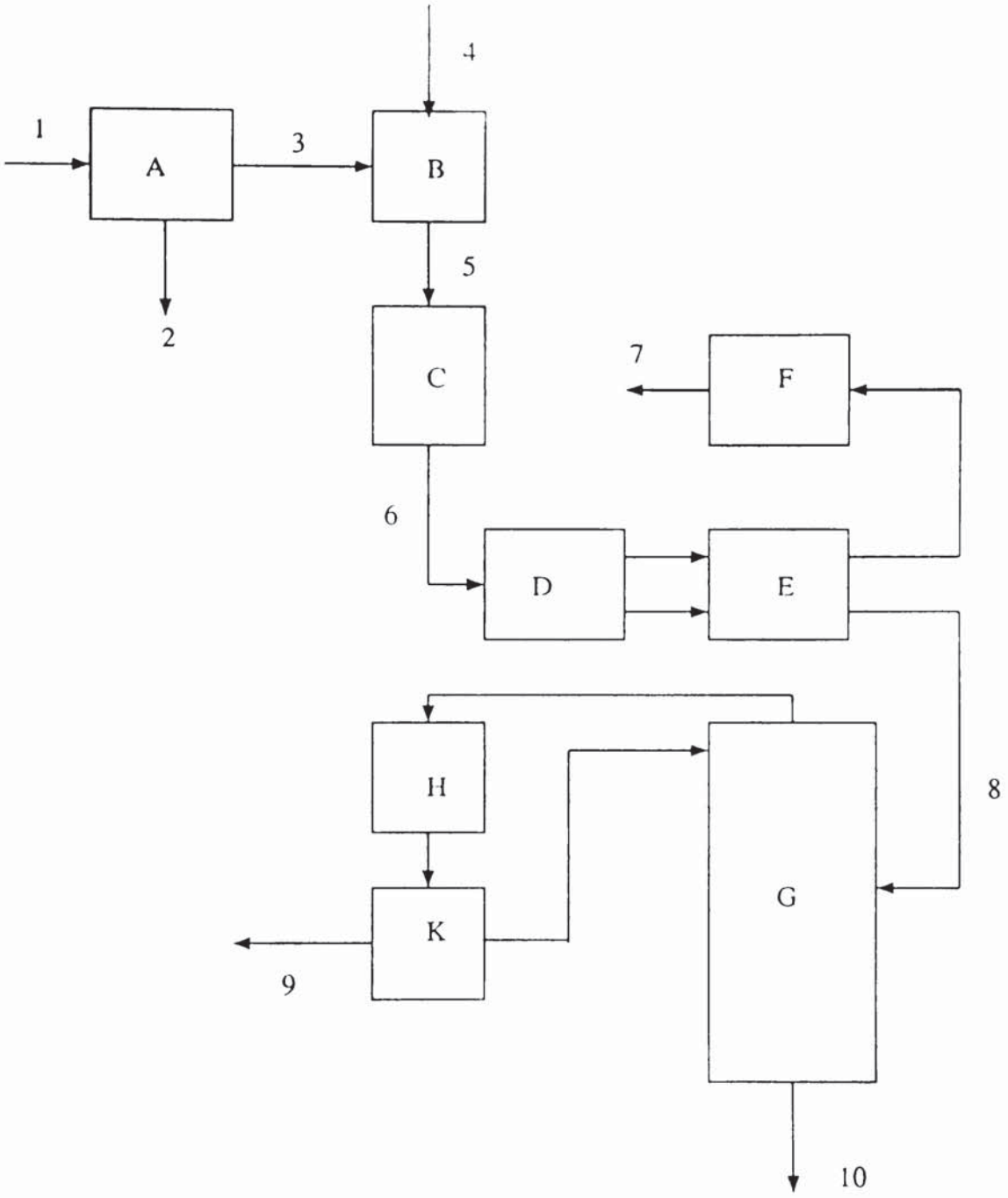


Figure 5.1b Block diagram for the section of ethylene glycol plant in figure 5.1 a

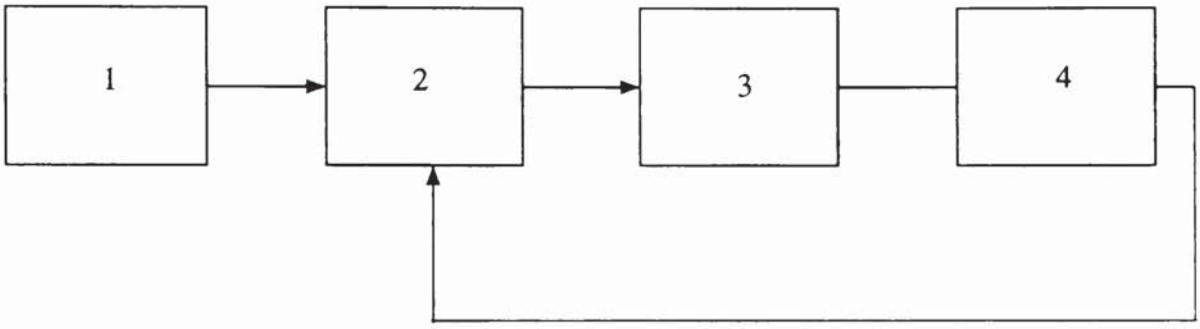


Figure 5.2 Connection of block modules

The data file contains the stream and unit information which is used by QUASILIN to perform the simulation.

Each of the methods used by the simulators described earlier, require the user to have prior knowledge of the stream details adopted in the library models to be used to simulate any flowsheet. This is required to ensure stream compatibility. Simulators utilising input processors also have the overhead of extensive translation times even for minor modifications although they can filter out errors occuring in the flowsheet, before the simulation run is commenced. Other simulators which use text files for the flowsheet topology input, require the user to know the file layout. This can result in errors during the generation of the topology file. This problem is increased as this type of simulator usually provides limited error checking.

An ideal simulator would help the user to construct the flowsheet topology. This can be achieved in at least two ways; the first method is a conversational procedure where the user responds to questions posed by the simulator, whilst the second method is a graphical flowsheet generator, which involves the user constructing a flowsheet using graphical techniques. Both of these methods create a flowsheet topology file in a suitable format for the simulator, hence overcoming the need for the user to be familiar with the topology file protocol.

Although, the conversational method takes the user through each step of the flowsheet generation, this method can become tedious for experienced users. Another advantage of the graphical flowsheet generation is that a diagram of the flowsheet is created which can be easily checked to ensure that the flowsheet generated matches the required flowsheet. The graphical flowsheet also offers a simple and rapid method of flowsheet comparison between different cases.

Various graphical methods of flowsheet topology generation for simulators have been developed. All of the methods involve the user selecting from a collection of predefined symbols representing different parts of the flowsheet. Some allow user definition of symbols.

Such symbols are known also as ikons or schematics by different authors. Some of these methods are described in section 5.2, where the suitability of their features is discussed in relation to the requirements of DASP II.

5.2 Methods for the graphical representation of the flowsheet

AGPSS (Singh and Carnahan (1981)) was one of the first interactive graphical interfaces to be linked to a simulation package. The data, including both picture and process information, are organised in a relational data structure. Fortran routines are used for addition, retrieval and deletion of the data. Although schematics of various unit operations are provided, new schematics can be produced by the use for new models. This may seem useful but can cause problems such as multiple definition of the same unit and the whole system becomes user specific.

Liang (1985) described another integrated simulation system which allows the generation of a graphical chemical process flowsheet. This system incorporates dynamic simulation in the form of DYNSIM. The graphical flowsheet generator must be used on graphics terminals hence universal usage is not possible. Keywords are used to generate the flowsheets. The unit schematics are chosen by using keywords then positioned using a graphics cursor. However, this method can become tedious when applied to the stream definition, which requires the line description keyword to be entered together with the source and sink unit numbers. A topology data file is produced. The whole package is integrated with the simulator and therefore allows a simulation to be executed from within the package. Flowsheet construction is undertaken by selecting the appropriate schematics which are then connected by using streams at assigned ports on the equipment schematics. The convergence criteria of stream parameters are entered at this stage. The program also calculates the enthalpy of the streams by calling the appropriate physical property routines. During a simulation the user can interrupt the calculation and change the values of parameters. All the data including the graphical flowsheet is stored in a database. This is an interesting concept but may be restrictive for experienced users who may not want to

reconstruct the whole flowsheet to slightly modify the topology for the simulator. As IDFP requires the user to learn the keywords to generate flowsheets, it negates some of the benefits of autogeneration of the topological information.

Naess and Loeken (1985) describe a front end which enables a graphical front end to be generated, which in turn provides the topological information for PROCESS (Brannock *et al* (1979)). A schematic program enables the user to build a library of graphical symbols which are then used to draw a flowsheet. A flowsheet extractor program combines the topological information with the respective simulator keywords and fills in the unit operation and stream identifiers. A database management system is used to handle the flowsheet data and the keywords to the process simulator. This front end produces a data input file which is then accessed by the simulator.

Other researchers such as Preece *et al* (1987) and Curtis *et al* (1981) have worked on production of stand alone graphical flowsheet generators. Curtis *et al* (1981) described a system which lays out the flowsheet drawings from the numerical connection data. This work is designed to provide a means for producing standardised output from flowsheeting studies. It can also be used as a visualization device for presenting alternative process networks generated by process synthesis programs. This mechanism of automatically generating the flowsheet has little significance in making it easier for a user to set up the topological information for a simulator. However, the aim of producing output from simulators for visual comparison would be useful in a complete simulation package.

A more promising graphical process flowsheet generator is discussed by Preece *et al* (1987). They have developed two 2-D graphics packages, the first is the graphical process flowsheet generator PFG and the other being a piping and instrumentation generator PIG. These packages enables the user to interactively construct 2-D process flowsheets and piping and instrumentation diagrams using ikon and menu driven input techniques. The PFG/PIG packages can be interfaced with simulation packages to automatically produce the topologicial

information for the simulators. As PFG/PIG automates the generation of the process topology information in a highly interactive manner they would be ideal for making DASPII into a more interactive and robust simulation system. Although, operation of simulation packages can be speeded up by using graphical flowsheets which automatically generates the topological information, some of the advantages are negated if the graphics package produces unnecessary flowsheet details. At present PFG produces the actual stream connection whilst the instrumentation information is added using PIG. This level of detail is not required during the creation of the process topology and one system combining PFG and PIG is sufficient. This version of PFG allows 18 types of process equipment.

In this section the general features available in some graphical process flow diagram generators are described. This is then used as a basis for formulating the requirements for a process flowsheet generator for DASPII, these are described in section 5.3.

5.3 Requirements for a process flowsheet generator for DASPII

The objectives of the graphical system required to generate the topological information for DASPII are:-

1. The package must be highly interactive, enabling the user to simply create a flowsheet from standard items of process equipment for which simulation models are available.
2. A method of selecting the models available within the DASPII model library.
3. The topological information must be automatically generated for DASPII from a graphical representation of the process.
4. The package should enable instrumentation to be included with other process equipment in one package and therefore be included in the topological information.
5. The package must be able to retrieve a partially or fully created flowsheet for modification.
6. A mechanism for modifying the topological information and the graphical flowsheet once the user has left the graphical front end and is in the simulation mode. This would

overcome the need for the user to return to the graphical front end when a small modification of the flowsheet is required once the user is in the simulation mode.

It is evident from the literature survey that PFG fulfills most of the requirements of a graphical flowsheet topology generator for DASPII. Although, PFG has some limitations, which are discussed in section 5.4, it is a good base from which a suitable graphical front end can be developed for DASP II.

5.4 The limitations of PFG as a flowsheet topology generator for DASPII

1. At present PFG generates the topological information for specific simulators. The PFG source code needs to be modified to supply the correct topological information for each simulator. A simpler method would involve dumping all the possible topological information into a standardised text file. Each simulator would then require a specific interface which would extract the relevant information in order to generate the required simulator topology input.
2. Most simulators require a flowsheet that consists of the process flow diagram and a basic process and instrumentation diagram. The usual PIG details illustrating the different controllers and valves on a flowsheet is not required.
3. Only 18 unit symbols are available in PFG to represent all possible units in a process flowsheet. As simulators usually have more than one model for each unit, there is no way of selecting the required model from those available in the simulator library.
4. PFG at present requires all modifications no matter how small to be made in the graphics mode. This means that the user needs to return to the graphics mode of PFG from the simulator even for minor modifications.
5. PFG does not differentiate between streams. Although, it allows streams of different thicknesses to be used in the flowsheet, there is no mechanism of defining different stream types, such as vapour, liquid, mixed phase etc.

An interface between PFG and DASPII has been developed, this overcomes the limitations described in this section. The interface is described in the next section together with the mechanisms used to overcome the limitations with PFG.

5.5 BTOPOL - The interface between PFG and DASPII

The main objective of BTOPOL is to create an efficient interface between PFG and DASPII, so that by utilising all the useful features of PFG and overcoming all its limitations an easy to use environment is created for generating the flowsheet topology for DASPII.

Overcoming the limitations of PFG

1) Modification of the source code to generate flowsheet topology

   As discussed in the previous section one of the major weaknesses of PFG is that the PFG source code has to be modified to generate the process topology information for each simulator. The best way of overcoming this limitation is to dump all the unit information and their connection data into a file. BTOPOL reads this data file and by interrogating the user, generates the required flowsheet topology file for DASPII. This required the modification of the PFG source code, so that all the topological information is dumped into a data file, see figure 5.3.

2) PFG has been modified so that a valve and controller symbol are available in PFD mode. This is in line with the DASPII requirement of a mixed PFD/PID. This allows the selection of general valves and controllers in the flowsheet hence enabling a flowsheet

topology to be generated for DASPII, without having to revert to PIG which now becomes redundant.

3) The limit of 18 symbols in PFG would be too restrictive for realistic chemical process representation. This limitation can be overcome in two ways : the first involves increasing the number of symbols so that each model isrepresented by a unique symbol. The alternative method involves constructing the flowsheet with the symbols that are currently available, these

symbols are used as generic flowsheet units and are defined more closely in BTOPOL.

The former method has certain weaknesses which would hinder its use as an effective topology generator. The main weakness would be the constant need to modify the PFG symbol list to match a usually growing simulator model library. This limitation would be avoided by the alternative method as the symbols are generic. Therefore this method is implemented in DASPII. The flowsheet is represented in PFG by using the generic symbols. A flowsheet topology file is then created in terms of the generic units. PFD also saves files in its own format. BTOPOL reads this file and by interrogating the user with the aid of database files via BTRIEVE (Anon (1976)), it creates a more detailed topology file for DASPII (see Figure 5.3). BTOPOL is also able to modify the files of the flowsheet saved by PFG to reflect changes made by the user. This is optional. Information of all the available DASPII library models for each generic PFG symbol is retrieved from the database files and displayed. The user then selects an appropriate library model to represent the flowsheet unit. If no library models are available for each unit the user has a number of options. These are as follows:

i. Remove the unit from the flowsheet. This will be discussed in (4).

ii. Select another model as a replacement, from the complete list of library models.

iii. To choose a model that is not currently in the library. A dummy model is written into the topology file. All of the relevant model information must then be added to

the topology file once the new model has been generated. This technique overcomes the need for PFG or BTOPOL to be modified as the DASPII model library is developed. The only updates required are to the database files, which can be made easily by the user.

4) PFG only allows modification to the flowsheet in the graphics mode. These flowsheet modifications can be required during various states of a simulation study. One such stage is when suitable library models must be selected for the generic units chosen in PFG (see 3). Two of the options available to the user could benefit from modifications to the flowsheet data saved by PFG other than by using PFG. For instance if a unit is to be represented by a different model type then the accompanying PFG - process flowsheet diagram description should be modified accordingly. This should also be the case if the unit is to be completely removed from the flowsheet. Another potential use of this feature is during an actual simulation run where parts of the flowsheet can be removed to overcome flowsheet latency (Ogbonda (1987)). It would also be extremely useful during a conceptual design study where variations of a process flowsheet are simulated to evaluate the best design. Therefore a copy of the existing flowsheet and modifications of this copy would allow a great deal of simulation experimentation with various flowsheet configurations.

The removal of a flowsheet unit has been implemented in DASPII during the construction of the flowsheet topology file via BTOPOL. This feature can be illustrated with the aid of Figure 5.5. The flowsheet represents a distillation column (unit 3) with its feed from mixer 2 (unit 2). Three feeds are mixed in mixer 1 (unit 1) with its output being one of the feeds of mixer 2. The fourth system feed being the second inlet stream of mixer 2. If for example mixer 2 needs to be removed from the flowsheet then feed 4 must either be added to mixer 1 or directly to the distillation column. BTOPOL prompts the user to select the sink unit for feed stream 4. Built-in rules allow the unit to be removed from the PFG flowsheet description and its feed stream (stream 11) is rerouted to the selected unit.

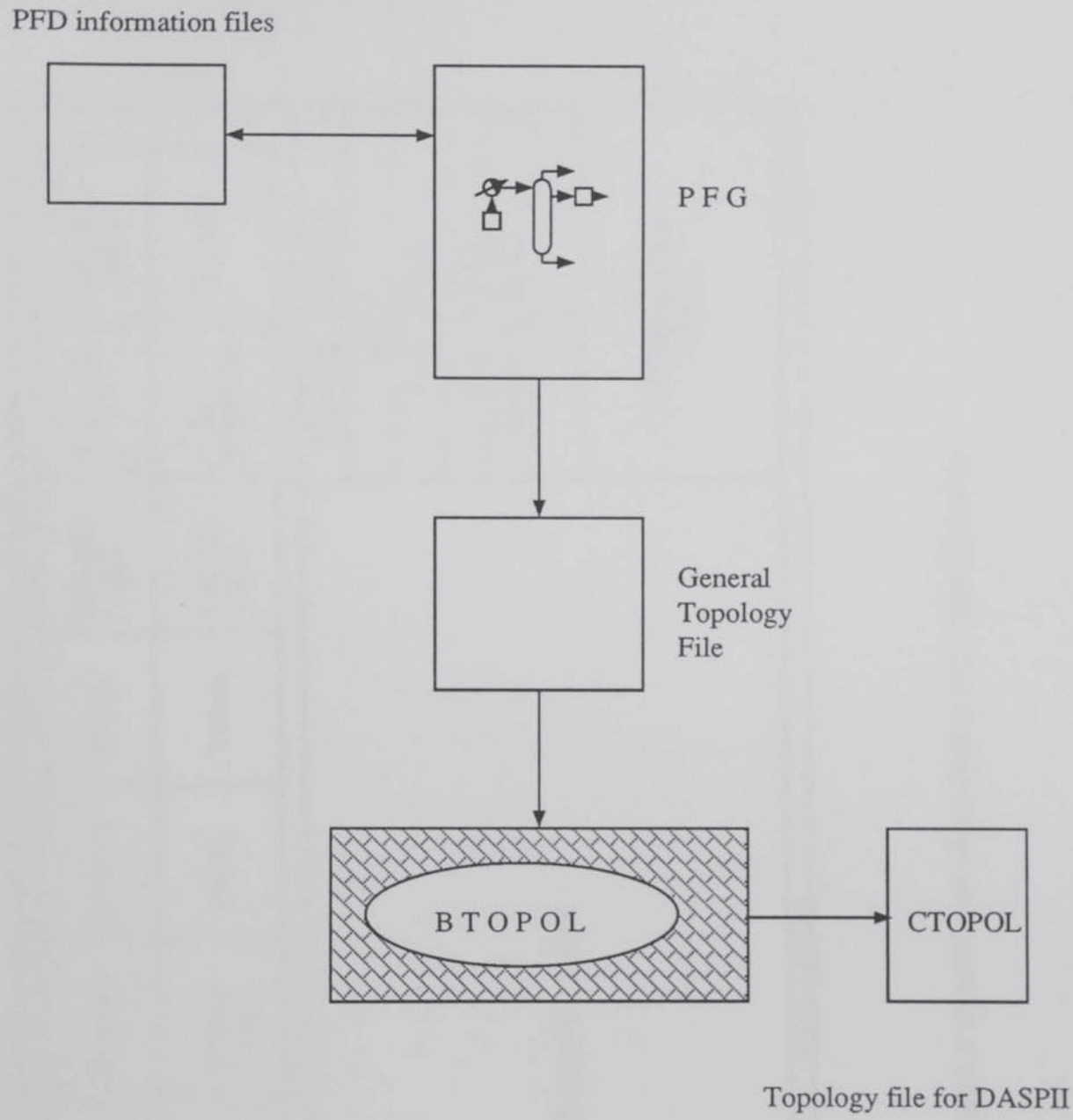


Figure 5.3 The generation of a topology file

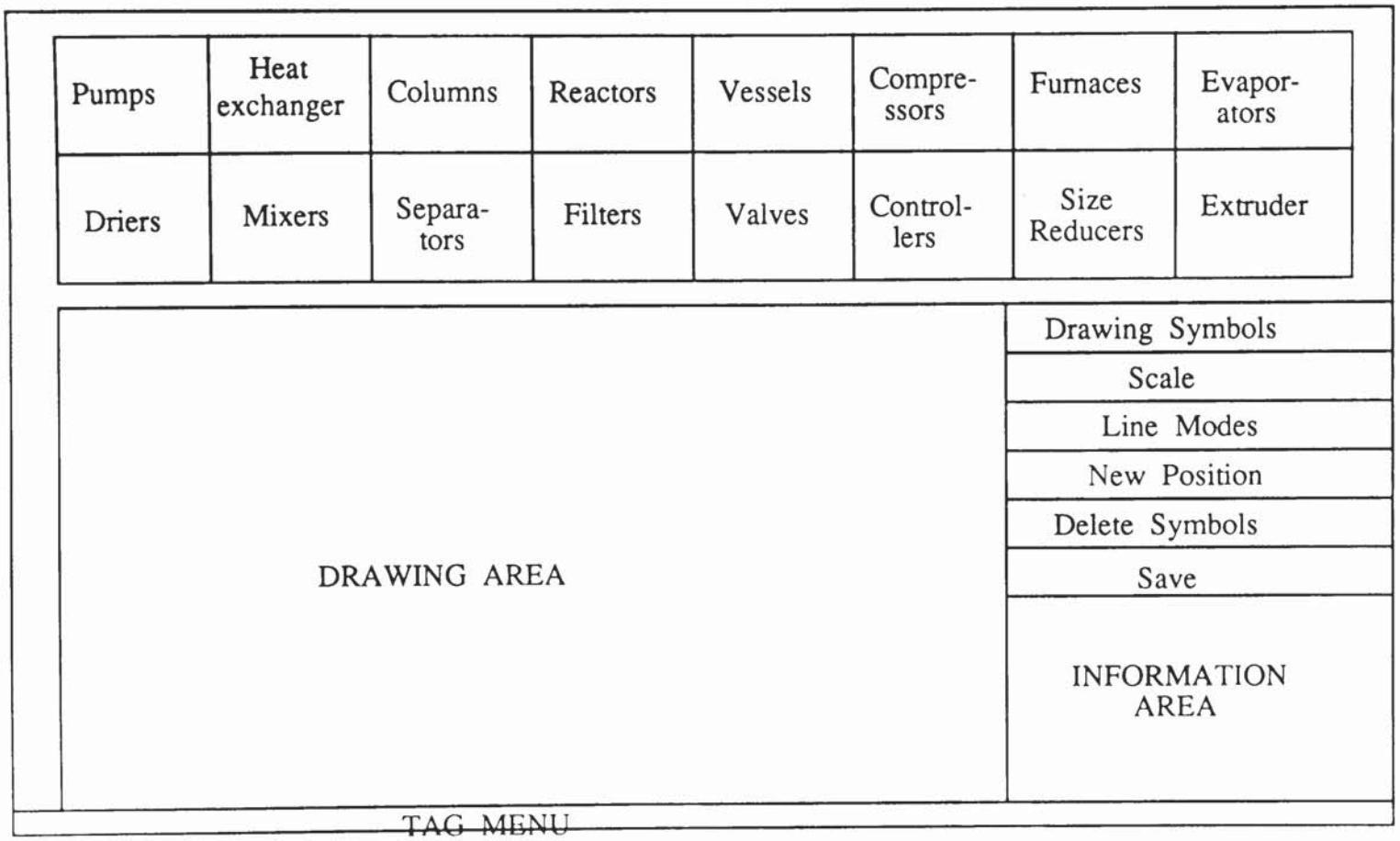
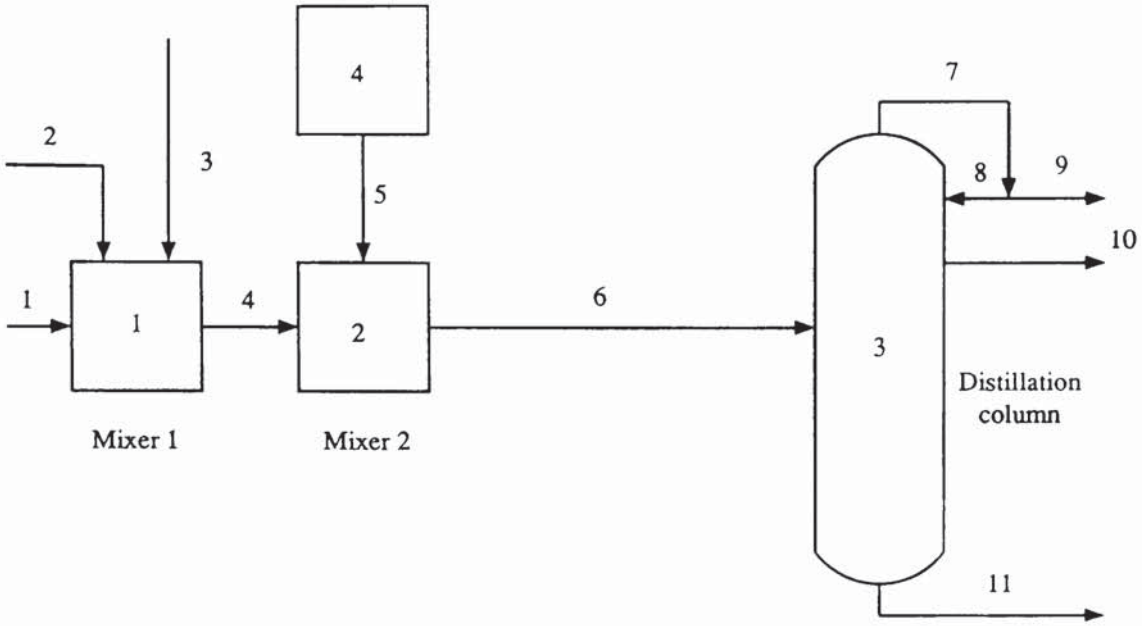


Figure 5.4 Symbols and Menu display in SYMBOLS menu



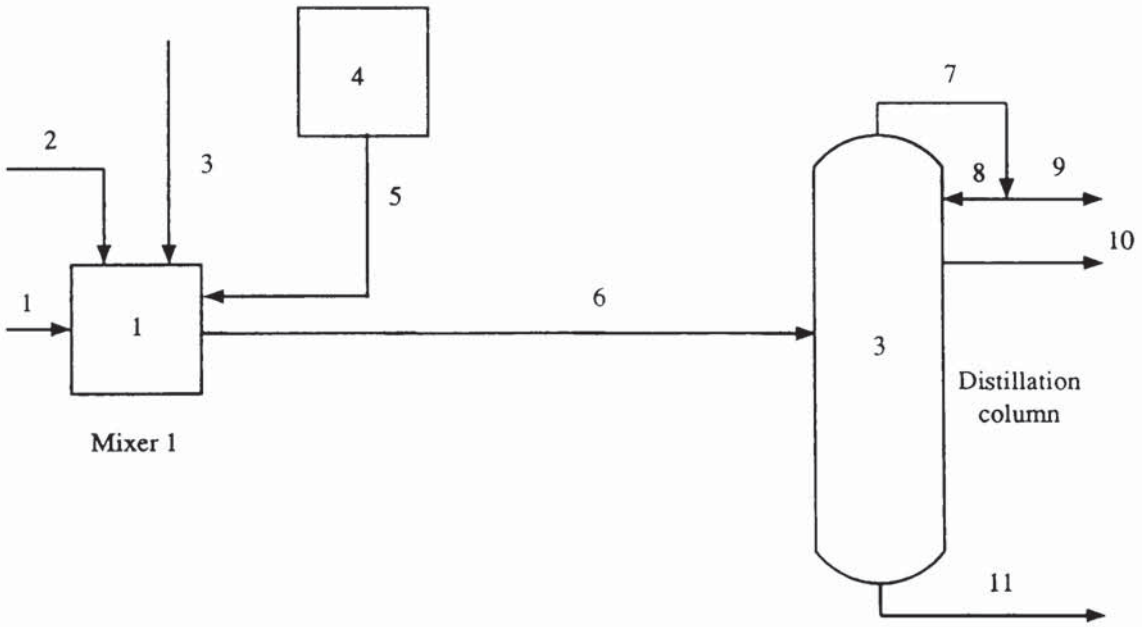


Figure 5.5 The removal of unit 2 from the flowsheet

In Figure 5.5 stream 11 becomes the fourth feed stream of mixer 1. The main rules implemented in BTOPOL to modify PFG flowsheet diagram description files are as follows:-

i. If the feed stream to the unit is at the vertical position as the exit stream from the unit then the end point of the feed stream is extended to meet the start point of the exit stream (see Figure 5.6).

ii. If the exit stream starts at a higher vertical position than the end point of the feed stream, the start point of the exit stream is moved to the vertical co-ordinate of the feed stream. The end point of the feed stream is extended to meet the start of the exit stream (see Figure 5.8).

iii If the exit stream starts at a lower vertical position than the end point of the feed stream, the start point of the exit stream is started at the vertical co-ordinate of feed stream. The end point of the feed stream is extended to meet the start point of the exit stream (see Figure 5.9).

iv. Streams with the same source and sink units as a result of unit removal are completely removed from the flowsheet. For example if unit 2 is removed in figure 5.6, stream 1 begins and ends with unit 1; as it has become redundant, stream 4 would be removed.

5) Stream Definition

DASPII requires each flowsheet stream to be clearly defined as belonging to a specific type (Ogbonda (1987)), for instance as liquid flow, vapour flow etc. Streams must also be assigned stages to enable differentiation. Although, PFG allows the use of streams of different thickness, there is no mechanism of stream definition such as that required by DASPII.

This stream information is obtained by BTOPOL by interrogating the user. The user shown the stream number and is then prompted to enter information such as the stream ta description and type. The data is then written in a suitable format to CTOPOL - the DASPII

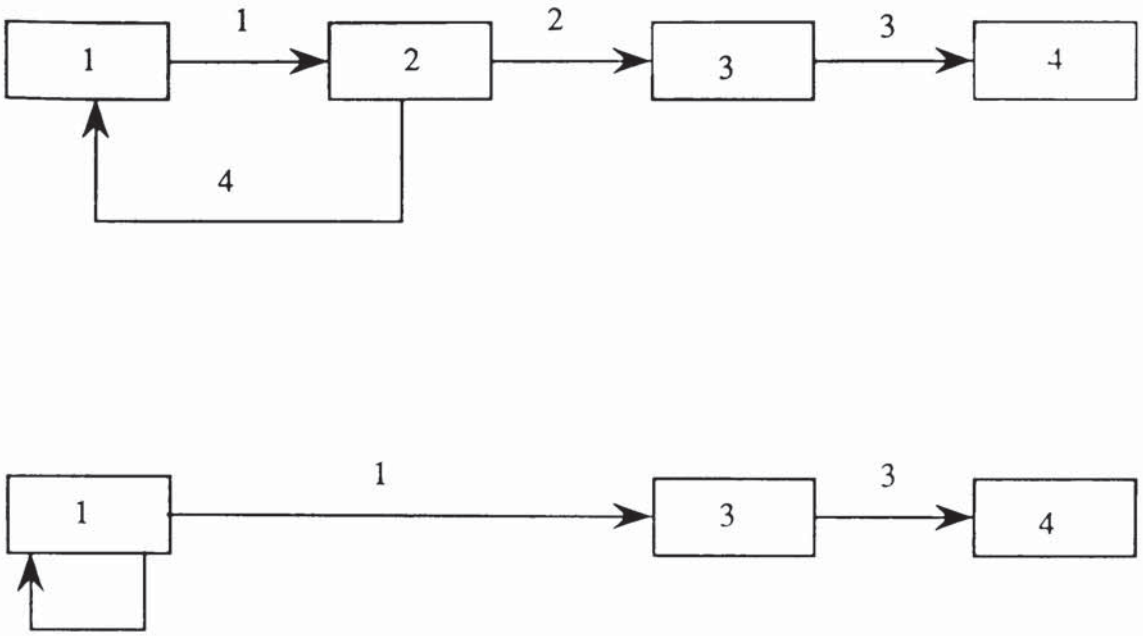


Figure 5.7 The removal of a recycle stream

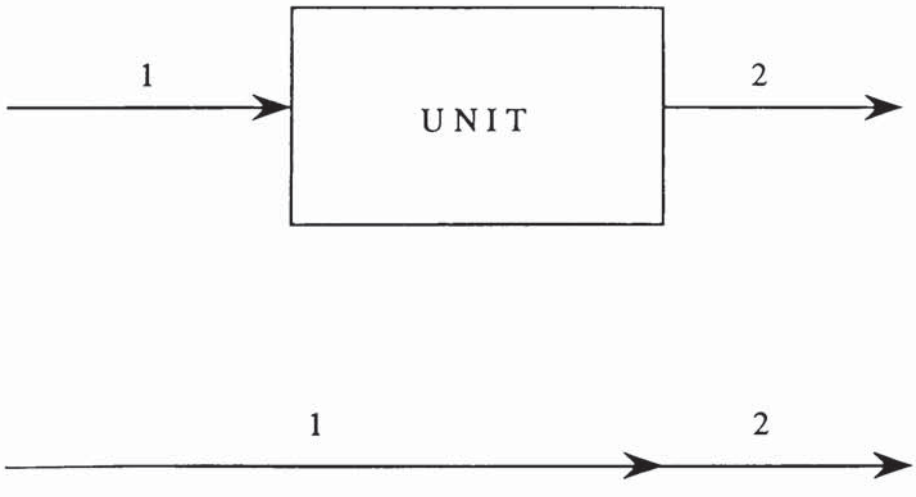


Figure 5.6 Connecting lines at the same vertical co-ordinets

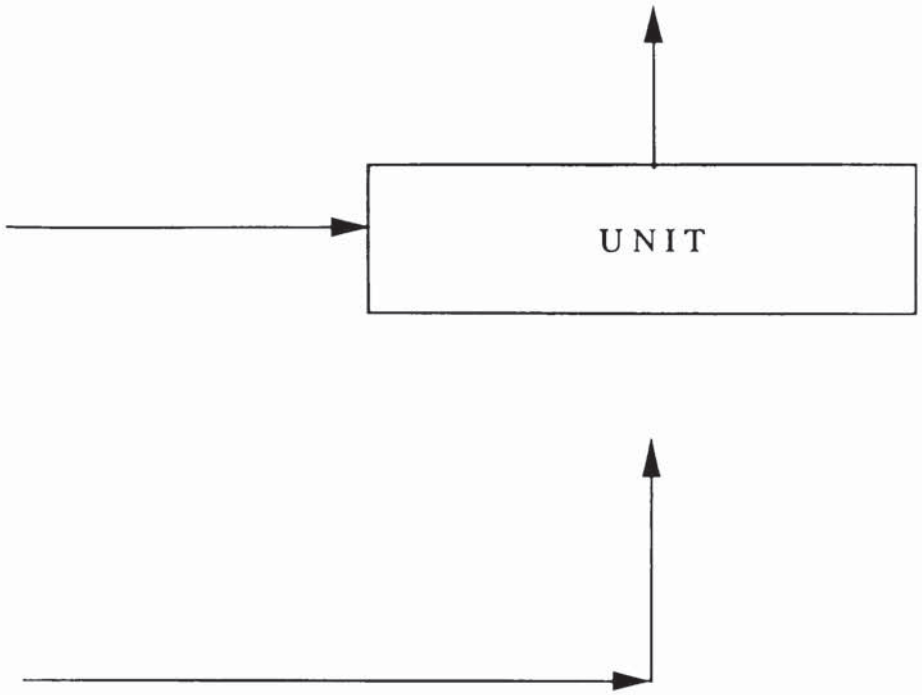


Figure 5.8 Connecting lines with an exit stream at a higher vertical position than the feed stream

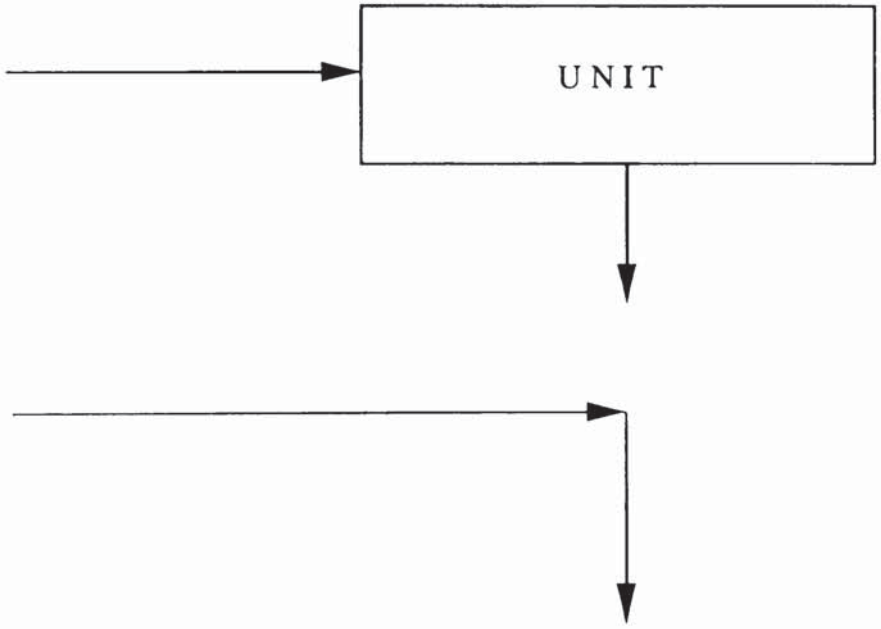


Figure 5.9 Connecting lines with an exit stream at a lower vertical position than the feed stream

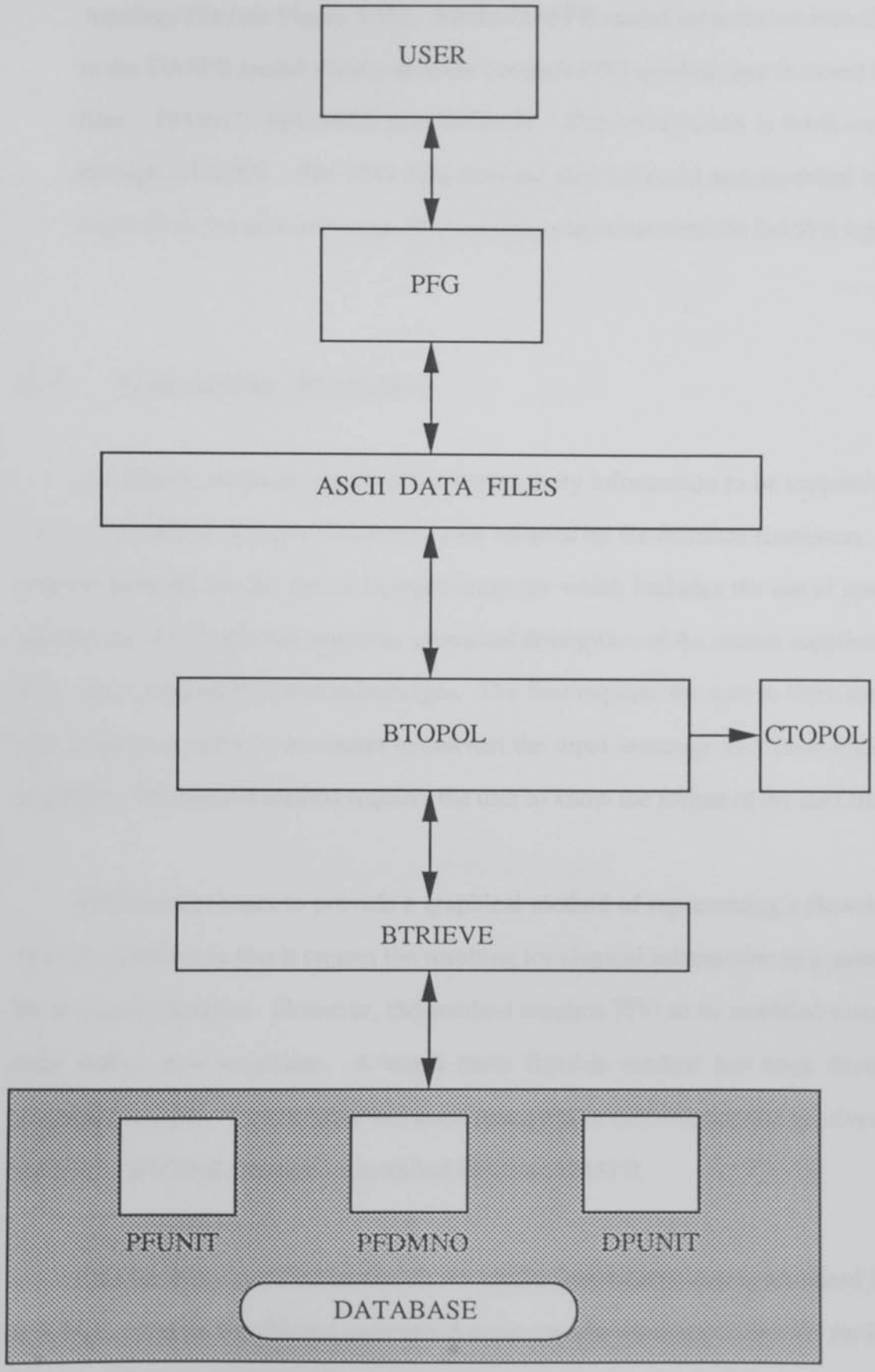


Figure 5.10 Structure of the graphics front end

topology file (see Figure 5.10). All the DASPII model information including the models in the DASPII model library suitable for each PFG symbol type is stored in the database files - PFUNIT, PFDMNO and DPUNIT. The information is retrieved by BTOPOL through BTOPOL. The PFD data files are also accessed and modified by BTOPOL as required by the user so that the PFD remains consistent with the DASPII topology file.

5.7 Concluding Remarks

All process simulators require the connectivity information to be supplied in a form that can be understood. Several mechanisms are adopted by the different simulators. The two most popular methods are the use of an input language which includes the use of special keywords and the use of a simple and normally numerical description of the stream supplied in ASCII data files. Each method has its disadvantages. The first requires the user to learn the language and also requires a software translater to convert the input language to a form understood by the simulator. The second method requires the user to know the format of the data file.

PFG enables users to provide a graphical method of representing a flowsheet. PFG can then be modified so that it creates the resulting topological information in a suitable format for the required simulator. However, this method requires PFG to be modified every time PFG is used with a new simulator. A much more flexible method has been developed for the PFG/DASPII link. This method has been described in this chapter and involves the use of an interface - BTOPOL, between a modified PFG and DASPII.

PFG has been modified so that all the topological information is deposited into a data file. BTOPOL accesses this file and generates a more specific topological data file for DASPII. It can also modify files used by PFG. This method overcomes the need for repeated modifications when PFG is linked with other simulators. PIG has also been merged with PFG so that excess information is not generated.

The use of the modified PFG for DASP has significant benefits. One of the most important benefits is the substantial increase in the usability of such a simulator. As DASP requires the topological information to be provided in a particular alpha-numerical format, the user was required to provide this information manually. However, by using PFG the user can visually create the flowsheet with errors and the required information is automatically generated. PFG is itself highly interactive using ikon and menu driven input techniques. PFG has required major changes to enable instrumentation to be drawn and included in the topological data for the front end. Another modification allows the generation of a topological data file that can be used by BTOPOL to generate a suitable topology data file for DASP.

BTOPOL - the link program between PFG and DASP uses a commercially available database management system BTRIEVE (Anon (1986)) and adopts an interactive conversational procedure to generate the topological information for DASPII. The PFG symbols are treated as unit types and BTOPOL enables the user to select appropriate models from the DASP library to represent the units in the flowsheet. This technique is more suitable as symbols are not required for each model as in AGPSS (Singh and Carnahan (1981)).

Since this would be user specific BTOPOL also enables the user to modify the flowsheet without reverting back to PFG graphics mode. This facility is further enhanced by the ability to modify the corresponding PFG graphical flowsheet from within BTOPOL, so that the two flowsheets are not out of step. Therefore this method of conversational flowsheet can be effectively used in other areas of DASPII.

Another useful feature of BTOPOL is the mechanism of automatically removing redundant lines which have the same sink and source as another line in the flowsheet. Whilst in PFG the redundant line is added to the line which has the same connections as the redundant line. Streams that have the same sink and source resulting from the removal of unit in a recycle loop are removed from the flowsheet.

No distinction is made between the lines in PFG and they are considered as just connections. This distinction is made in BTOPOL by interrogating the user who enters suitable codes for each line. The highly interactive system created by the implementation of a modified PFG and BTOPOL has resulted in an extremely usable system which promotes experimentation. As the topology data files for DASP are the same as before the user can still create them manually using a system editor or word processor.

CHAPTER SIX

MODELS AND THEIR SOLUTION

6.0 Introduction

The simulation of chemical processes using mathematical models is discussed in Chapter One. A suitable mathematical model must adequately represent all the aspects of the chemical system that are to be investigated. All of these aspects must be contained in the model without the inclusion of unnecessary detail. The consideration of what is essential and what is unnecessary will depend upon the requirement for which the simulation is undertaken. In general purpose simulation, the full range of requirements will not be known in advance. Therefore, a useful general purpose simulator will contain flexibility allowing different choices to be made by users with different needs. This flexibility can be provided by alternative models of the same process unit. This is inflexible because to change between models is a flowsheet change. In this work the flexibility is provided within process unit models to provide flexibility for the user without changing the flowsheet. This flexibility extends beyond simply changing the parameter set or specifications. Variables and parameters can be exchanged, allowing the user to specify a process variable without restructuring the problem.

The most common use of simulation involves standard unit models to be provided by model builders in simulator libraries (see Chapter Two), requiring the user to simply select the appropriate models for a particular flowsheet. Hence, to fully utilise the specification flexibility offered by the equation oriented approach various different methods of model design are used in simulators.

One method of avoiding including redundant equations is to evolve the model so that the complexity of the process model is increased progressively. This technique also enables the process model to be validated at every stage of development, thereby increasing confidence in the model. Models can be validated by comparing process observations with the predictions from the model under identical conditions or by comparing different versions. Such a stepwise

evolution is very easy if the changes are included in the unit models. The unit mathematical models are composed of sets of differential and algebraic equations, in which the variables represent particular characteristics of the process unit. Steady state simulation produces time independent solutions to their equations. The different strategies adopted for solving the model equations in process simulators are described in Chapter Two.

The equation oriented approach, which is implemented in DASP II, in theory enables the user to set any variables in the system as long as the problem is correctly defined. Implementation of this and other common difficulties occurring in model generation are discussed in Section 6.1. Modelling formulations and methods adopted for problem specification in various equation oriented simulators are assessed in Section 6.2. A method for defining the problem specification in DASPII which overcomes some of the major drawbacks associated with other methods is described in Section 6.3. An interactive method for generating the simulation control options is described in Section 6.4.

6.1 Model Building and Associated Difficulties

Simulators cannot be used to solve the model equations until the problem has been properly defined. If there are n equations and m system variables, (m-n) further variables must be arbitrarily specified in order to fully define the system. These extra specifications are termed degrees of freedom. In theory with the equation oriented approach any (m-n) system variables can be specified by the user with the remaining n variables being evaluated using the model equations. However, in reality this is not the case. If for instance all the variables are specified in a particular equation then the equation has become redundant. This results in a singular system which cannot be solved. This issue is further complicated during dynamic analysis which involves the incorporation of differential terms. In this case no differential term or differential variable (i.e. a variable which is included in a differential form anywhere in the system) can be specified as a known variable, as the system integrator will evaluate these variables. Another problem with dynamic systems has been discussed by Pantelides *et al*

(1987). This occurs during the solution of differential - algebraic equations (DAE's). They describe a classification method for DAE's according to their index, where they define the index of a system to be the minimum number of differentiations with respect to time that the system of equations have to undergo to be converted into a set of ordinary differential equations. Therefore the number of variables for which arbitrary initial values may be specified is less than the number of differential equations and similar numerical methods can be used to solve them. Problems of index greater than one arise when two or more state or differential variables are related. Examples of this occur in phase equilibrium processes, which commonly generate DAE systems of index 2 and greater. This usually occurs because the differential variables introduced by the material and energy balances are not independent, but related through the phase equilibrium processes. This problem of index greater than unity systems can be sometimes overcome by more rigourous models of the system. This provides an additional advantage for models with internal flexibility.

6.2 Modelling Strategies Implemented in Process Simulators

Three different strategies have been used for the formulation of the DAE problem in equation oriented simulators, they are as follows:-

i. Outer - inner method

This method simultaneously solves the differential equations, whilst the coupled algebraic equations are solved within an inner loop of the iteration process. This is therefore commonly referred to as the "outer-inner" convergence loop. This method was implemented in FLOWSIM V4.0 (Babcock (1981)) and can produce inaccurate simulation results because of this partial decoupling. It also renders the algebraic variables as local variables which therefore makes it impossible to allow them to be globally manipulated if this is required in design and optimization studies. In ASCEND (Kuru (1981)) and DYNAMIC QUASILIN (Smith (1985)) a velocity variable is introduced for each

differential equation, as shown in Chapter Three. This increases the dimension of the problem and also requires equations with more than one derivative term to be manipulated to state variable form.

ii. Linearly implicit DAE's

This formulation requires the derivatives to have coefficient values of one or zero. Hence all equations with variable coefficients must be manipulated to be brought to the correct form. BOSS (Joglekar and Reklaitis (1984)) is a simulator which adopts this formulation.

iii. The general format

This formulation allows more than one derivative term in any equation and derivatives to have variable coefficients. This approach does not have the same restrictions as the two former formulations. Other advantages offered by this approach include no distinction to be made between the algebraic and differential equations, introduction of new variables is not required and the model equations are not required in state variable format. This is the modelling strategy implemented in DASPII. In order to fully utilise the flexibility of the equation oriented approach, simulators have adopted many unique methods of enabling users to select the desired specification for each model.

For instance in QUASILIN (Hutchinison (1986)) various specification configurations are selected by the model builder for each model. These specification options are hard coded into the model. For a truly flexible system it is obvious that there will be a vast number of options for large models. This would require the user to have prior knowledge of the complete specification information for each option in order to select the desired specification. A similar method is implemented in DASPI where a special model parameter is selected by the user to define the variable specification details required for each model. Subsequently DASPI loses the benefits of using the general formulation strategy for the model equations.

In ASCEND II (Kuru (1981)) the interconnection between units and their equations and variables are represented by a GEV diagram. The GEV diagram is composed of Generators, Equation packets and Variable packets. A generator is a group of FORTRAN subroutines which

produces the partial derivatives and equation residuals. The generators undertake the calculations using the equations stored in equation packets which in turn uses the variables stored in the variable packets. A unit is a collection of generators. The user can change variable values and the variable specification by selecting the appropriate variable, and equation packets for each generator.

6.3 Selection Of Model Variable Specification

The objective of this part of the project was to enhance the model formulation strategy adopted in DASP II, to create a truly flexible simulation approach. This technique will also aid the user in the provision of a correct variable specification. This is achieved in two parts:

i) The model construction phase.

ii) An interactive front end which helps the user to correctly define the problem.

Each of these phases will be discussed in turn in this section.

6.3.1 Model Structure

The DASP II library models retain the same basic structure as in DASP I, in that the model is divided into several sections, each performing a separate function. The main modifications have occurred in the initialisation section, which initialises the model variables and parameters, and the function evaluation section which contains the model equations.

The initialisation section in the models has been modified to read a new data file - VCODES, containing variable and parameter codes. This enables the user to choose the known and unknown variables in each model. The variable and parameter values are passed to the model via data files in a similar way to DASPI.

The model builder writes the model equations using the general formulation strategy. One set of equations is required because the equations are independent of the specification. This is unlike DASPI, which required the model builder to write n sets of equations, where n was the number of variable specification options available in each model. Consequently, the new technique saves the model builder time and reduces the program size.

For each model, the model builder selects certain types of variables. These include:

i) Differential variables - Variables included in differential terms. During dynamic simulation these variables cannot be specified as known because this results in an equation system with a singular Jacobian matrix.

ii) Preset real parameters - These are real variables that are not allowed to be defined as unknowns such as liquid density. This can reduce the risk of structurally singular problems.

iii) Pre-defined unknown variables - These are variables which must always be defined as unknowns. An example of such variables is pressure in a phase equilibrium process. This can result in index greater than unity problems.

iv) Real parameters/variables - All the remaining real variables Any of these variables can be selected as knowns or unknowns.

This strategy can be shown using a proportional and integral controller as an example. The model equations are:-

$$ERROR = AXN * (SP - MV)$$

$$\frac{d(INTERR)}{dt} = \frac{(CONTOP-INTERR)}{TI}$$

$$CONTOP = GAIN * ERROR + INTERR$$

$$IF\ (MV < XI) \qquad MV = ZI$$

$$IF\ (MV > RI) \qquad MV = RI$$

IF (CONTOP < ZO) CONTOP = ZO

IF (CONTOP > RO) CONTOP = RO

Where

ERROR	-	the controller error
ZI	-	the zero of input signal
RI	-	range of input signal
ZO	-	zero of output signal
RO	-	range of output signal
N	-	controller action
TI	-	integral time
MV	-	measured variable
SP	-	set point
INTERR	-	integral of the error

The equation classification used for this controller is:-

i) Differential variables : INTERR

ii) Preset real parameters : ZI, RI, ZO, RO

iii) Predefined unknown variables : none

iv) Real variables : TI, AXN, SP, MV, CONTOP, ERROR, GAIN

The model builder inputs this information into database files, which are used in the interactive variable specification interface described in section 6.3.2. This is achieved by entering the data via XTRIEVE (Anon. (1986)) (see Figure 6.1) which is an interactive user interface to database files managed by BTRIEVE (Anon. (1986)).

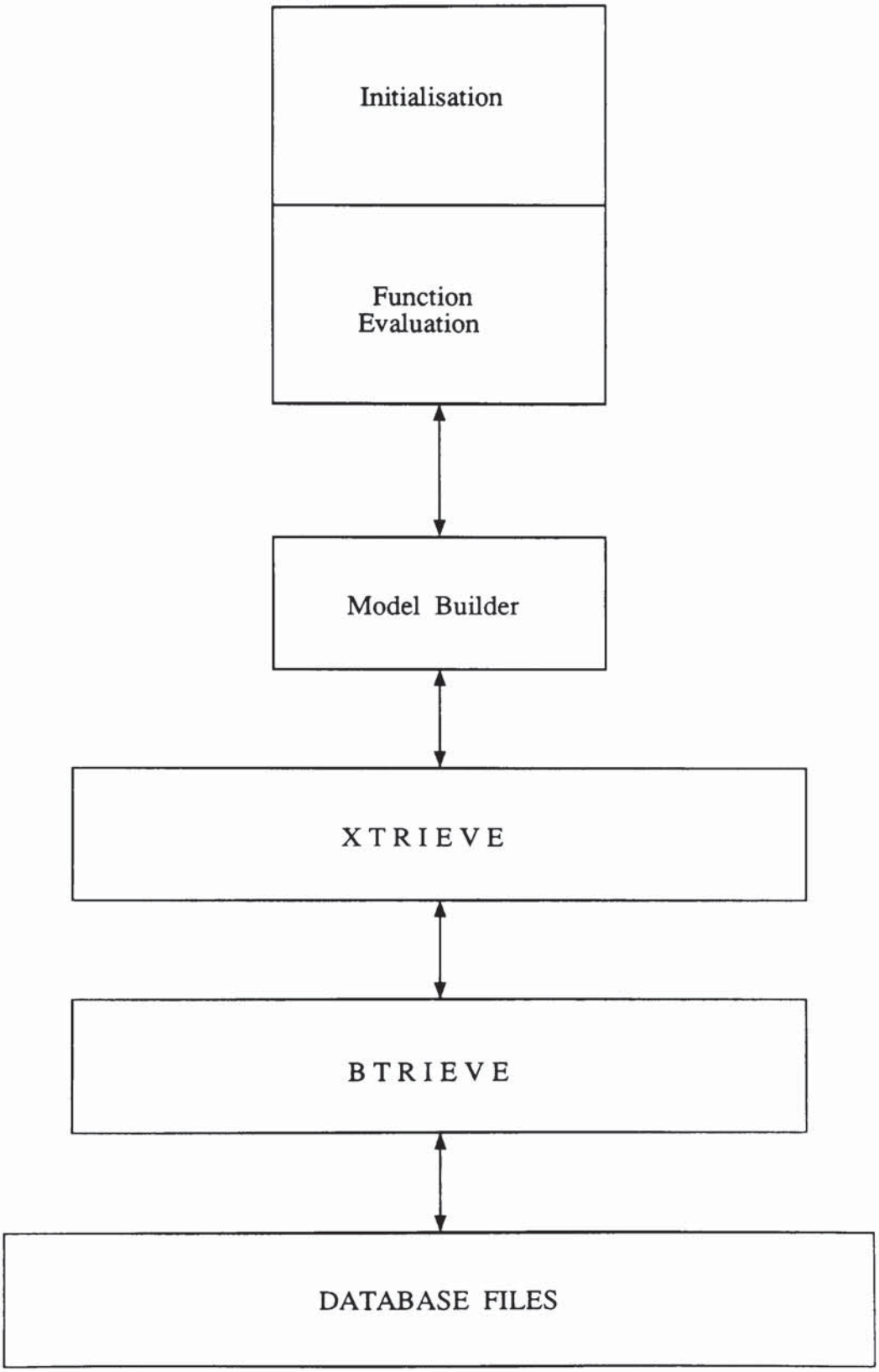


Figure 6.1 Model generation by the model builder

6.3.2 Interactive variable specification interface

An interactive interface - CUNIN has been developed to interrogate the user using the model information retrieved from database files via BTRIEVE. This information is provided by the model builder as described in Section 6.3.2. The interrogation is totally dependent on the flowsheet topology, therefore the interface accesses the DASPII topology file - CTOPOL which is created by BTOPOL the DASPII topology interface (see Chapter Five).

CUNIN prompts the user for values of the integer parameters for each model. This is done by retrieving the integer parameter list for the model from the database files and then appending this information to standard questions. The parameter values and their respective codes are written into the variable specification data file - CUNIT.

A similar technique is used for the variable specification definition, although this has increased complexity as a result of the user being required to select unknown and known variables for each model. Several rules have been applied to this procedure to overcome some of the modelling difficulties described in Section 6.1. The primary rule is to fulfill the degrees of freedom for the system. Another rule prevents the risk of structurally singular problems. Sections of the model can be switched on and off easily in order to modify the flowsheet that is being modelled. Once the user selects the known and unknown variables, their values are requested from the user again by the use of standard equations appended with data from the database files. The variable and parameter values are written in CUNIT. Their respective codes are also written into a variable specification coder file - VCODES (see Figure 6.2). These data files are accessed by the model initialization sections to initialise and define the equation system.

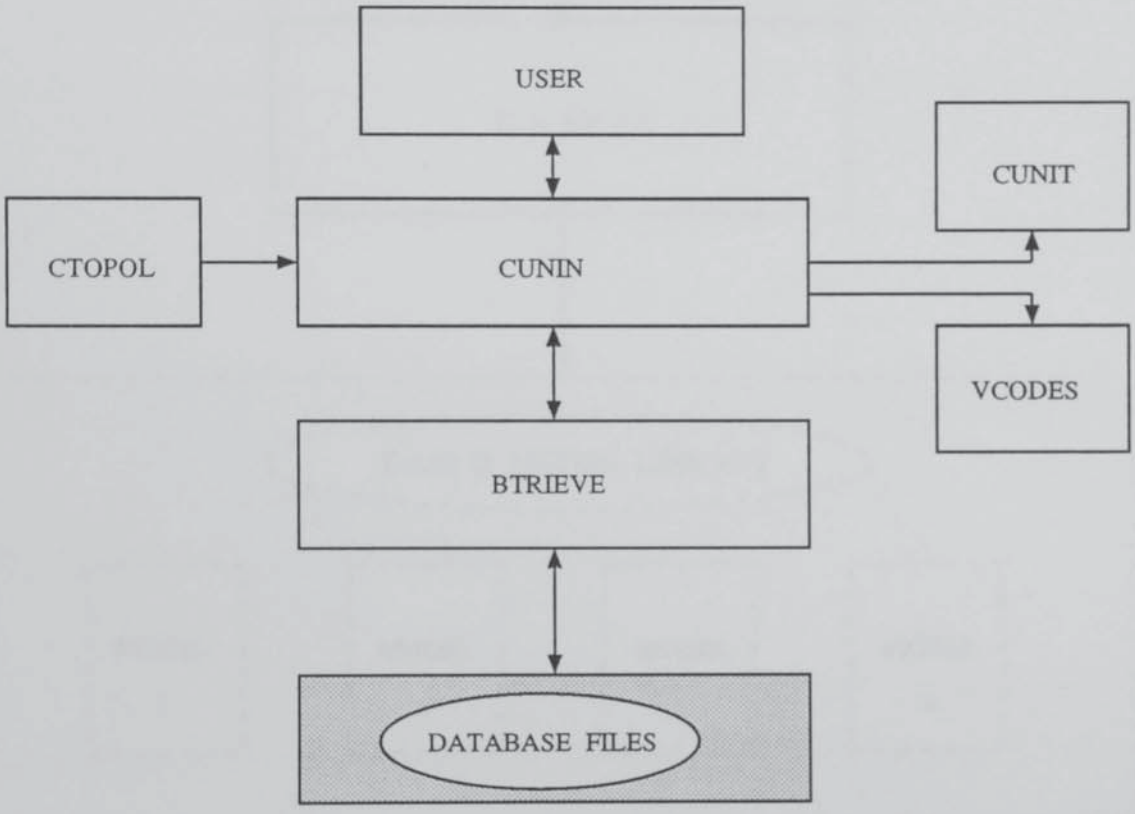


Figure 6.2 The generation of the variable specification files

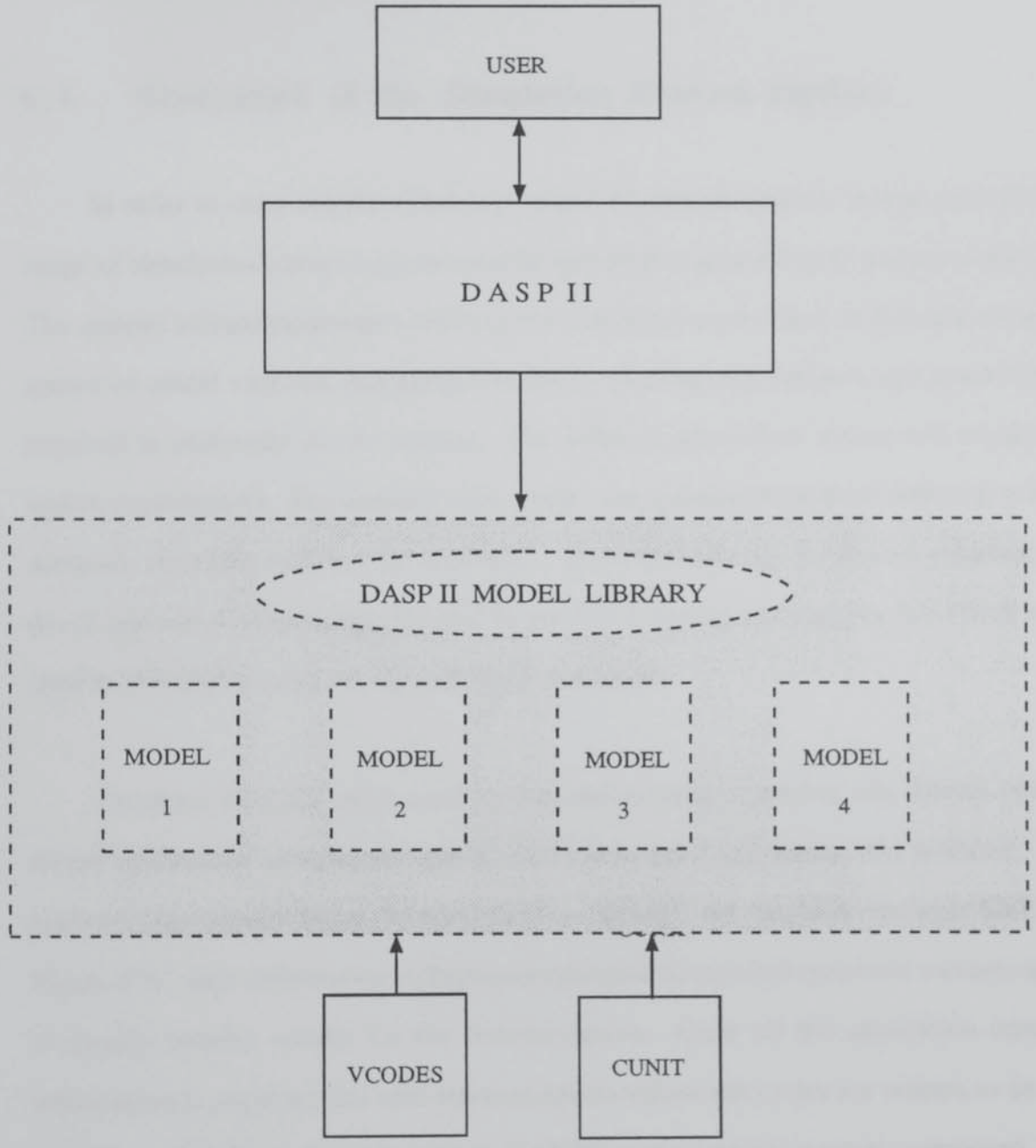


Figure 6.3 Construction of the equation system

The equation system for the flowsheet is constructed by DASP II. Each library model accesses CUNIT and VCODES during the initialization phase of the analysis to set up the equations, as shown in Figure 6.3.

6.4 Generation of the Simulation Control Options

In order to carry out the simulation with a flexible simulation system such as DASP II a range of simulation control options must be provided to give the user control of the simulation. The options include parameters defining the simulation model such as dynamic or steady state, source of model routines, numerical tolerances, plotting requirements and numerical methods required to undertake the simulation. The different simulation modes will require different option requirements. For instance with steady state simulation, options referring to integration methods available will not be required. Consequently, an intelligent interface has been developed which interrogates the user to produce a control options data file which can then be used by DASPII to carry out the necessary simulation.

Database files are again used to store information regarding the control options. The stored information includes the option, description, name and limits. The interface - CINDAN, retrieves information from the database files through the database manager BTRIEVE (see Figure 6.4). This information is displayed appended to standard questions and prompts the user to supply suitable values for the desired option. Once all the simulation control option information is supplied, the user selected option values and codes are written to an ASCII file CINDAT. This file is then accessed by DASPII once a simulation run is commened.

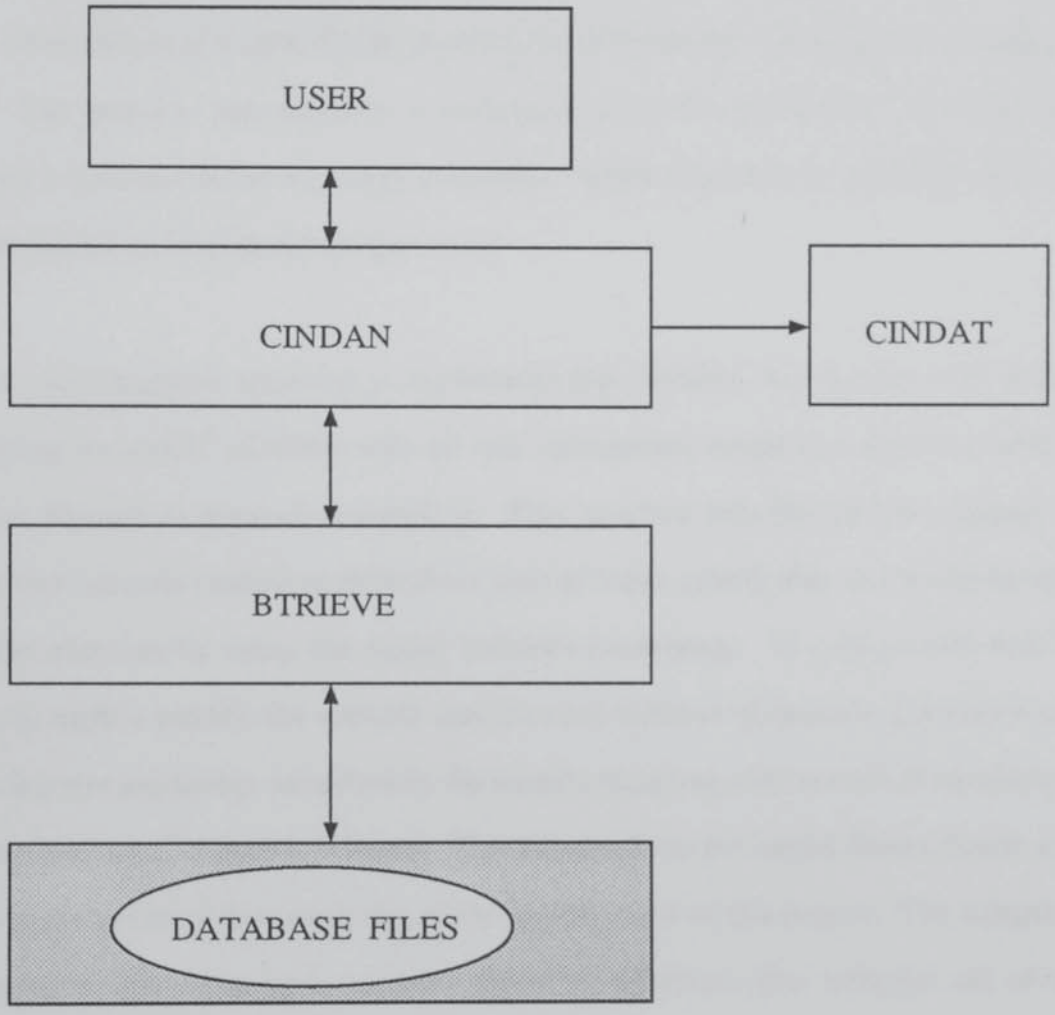


Figure 6.4 The generation of the simulation control options

6.5 Concluding Remarks

Most of the methods implemented in simulators for the definition of variable specification do not offer methods for aiding the user in the most difficult task of correctly defining the variable specification. SPEEDUP (Pantelides (1988)) offers a compromise, by having a section which allows the user to specify the problem but the variable values must be entered at a later stage. The problem also requires a translation once the values have been entered, which becomes a limiting factor for large problems. More importantly problems which result in singular Jacobian matrices are not prevented.

A more practical approach is implemented in DASPII, this involves the model builder classifying the model variables with all this information stored in a database which the user accesses through an interactive interface. This interface aids the user to correctly define the system and prevents modelling difficulties such as index greater than one problems and singular Jacobian matrices by using the model builder's knowledge. In conjunction with this is the ability to rapidly modify the variable specification without exhaustive translation procedures. These features are further enhanced by the models requiring only one set of equations no matter what variable specification is selected. This helps to keep the model library size to a minimum, hence enabling DASP II to meet one of the requirements of this project. The complete variable specification system is made compact by using database files with the aid of a database management system. This has yielded a system which is useful both for the model builder and more importantly the user who can use the flexibility and the power of an equation oriented simulator with a simple method of problem definition commonly associated with the sequential modular approach. Resulting in a system which provides an excellent method for experimenting with the flowsheet.

CHAPTER SEVEN

DEMONSTRATION OF DASP II

7.0 Introduction

The objective of this chapter is to demonstrate the features implemented in DASPII. As the aim of this project is to bring together various different sections of research to generate an interactive, flexible and robust simulation system, each of the new developments are illustrated with literature examples. Although each of the main aims of the project seem not to be related, they are in fact very closely linked. However, the main feature offering the most towards each of these aims will be tested to demonstrate the abilities of DASPII. The graphical representation of the process flowsheet and the subsequent generation of the topology information for DASPII will be used to illustrate the interactiveness of DASPII.

This is demonstrated in Section 7.1 where the topology information for a methanol mixer tank is generated from a graphical representation of its flowsheet. The methanol mixer tank is also used to illustrate the benefits of the new modelling and variable definition technique described in Chapter 6. This is shown in Section 7.2, where the variable definition is modified so that the system configuration is changed.

The robustness of the simulator has been improved by investigating the area of solution of non-linear algebraic equations. This is a vital area as systems of NLAEs are required to be solved repeatedly during many different operations. In Section 7.3 the performance of CONLES is compared with that of Broyden's method on a series of equations taken from literature.

7.1. The generation of a DASPII flowsheet topology input file for a methanol mixer tank problem with the aid of BTOPOL

A simple flowsheet of a methanol mixer tank (Figure 7.1) will be used to illustrate the interactive generation of the topology information for DASPII.

The system consists of a perfectly stirred tank with one water feed, two methanol solution feeds, one concentrated and the other dilute. The methanol solution feed streams are controlled by two-position actuator valves which can be fully on or fully off only. Control valves on the water feed stream and the tank outlet stream are used to regulate the flow of the respective streams. Two control loops manipulate these control valves to keep the level in the tank at 0.8 m and the mole fraction of methanol at 0.4 in the outlet stream. The tank acts as a perfect mixer with no heat transfer. The flowsheet generated by PFG for the system is shown by Figure 7.2 Control and instrumentation equipment is included in the diagram in the form of the composition and level controllers together with the inlet water feed valve and the outlet methanol valve. A heat exchanger is used to heat the dilute methanol feed. The symbols representing the units are shown in Table 7.1

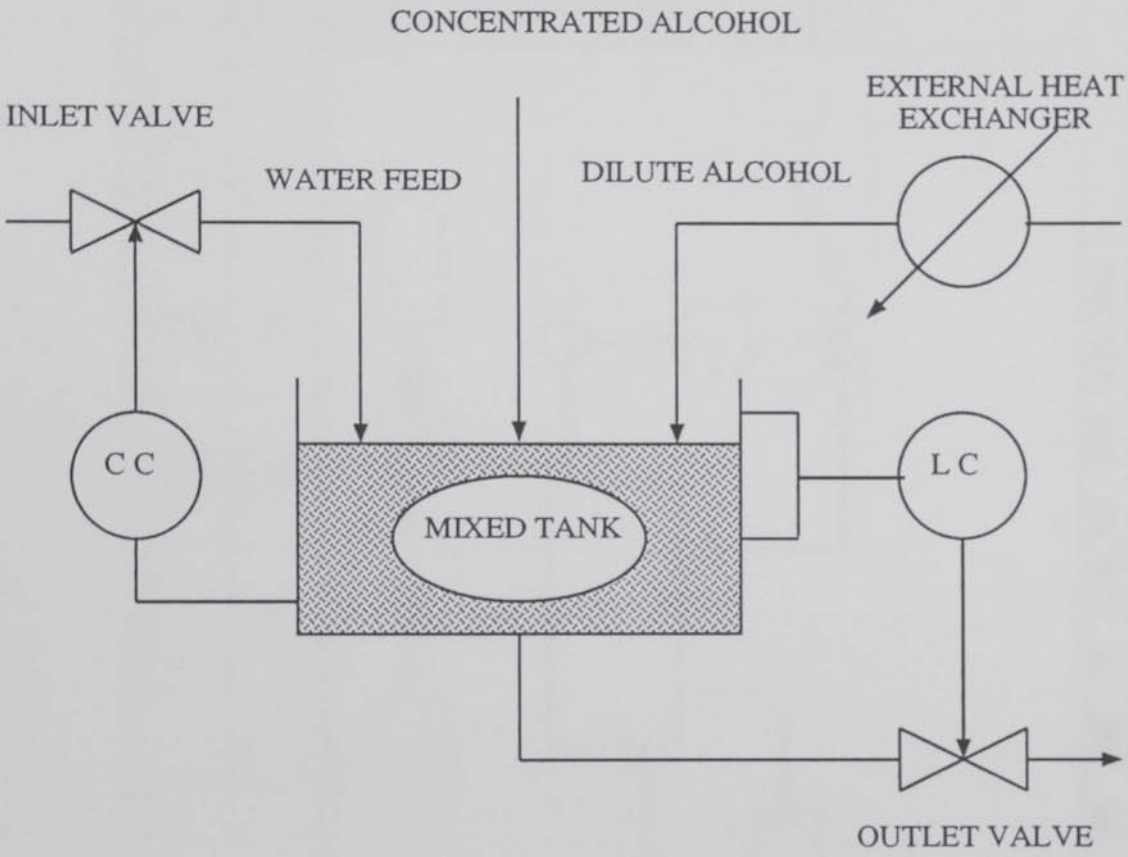


Figure 7.1 Simple flowsheet of methanol mixer problem

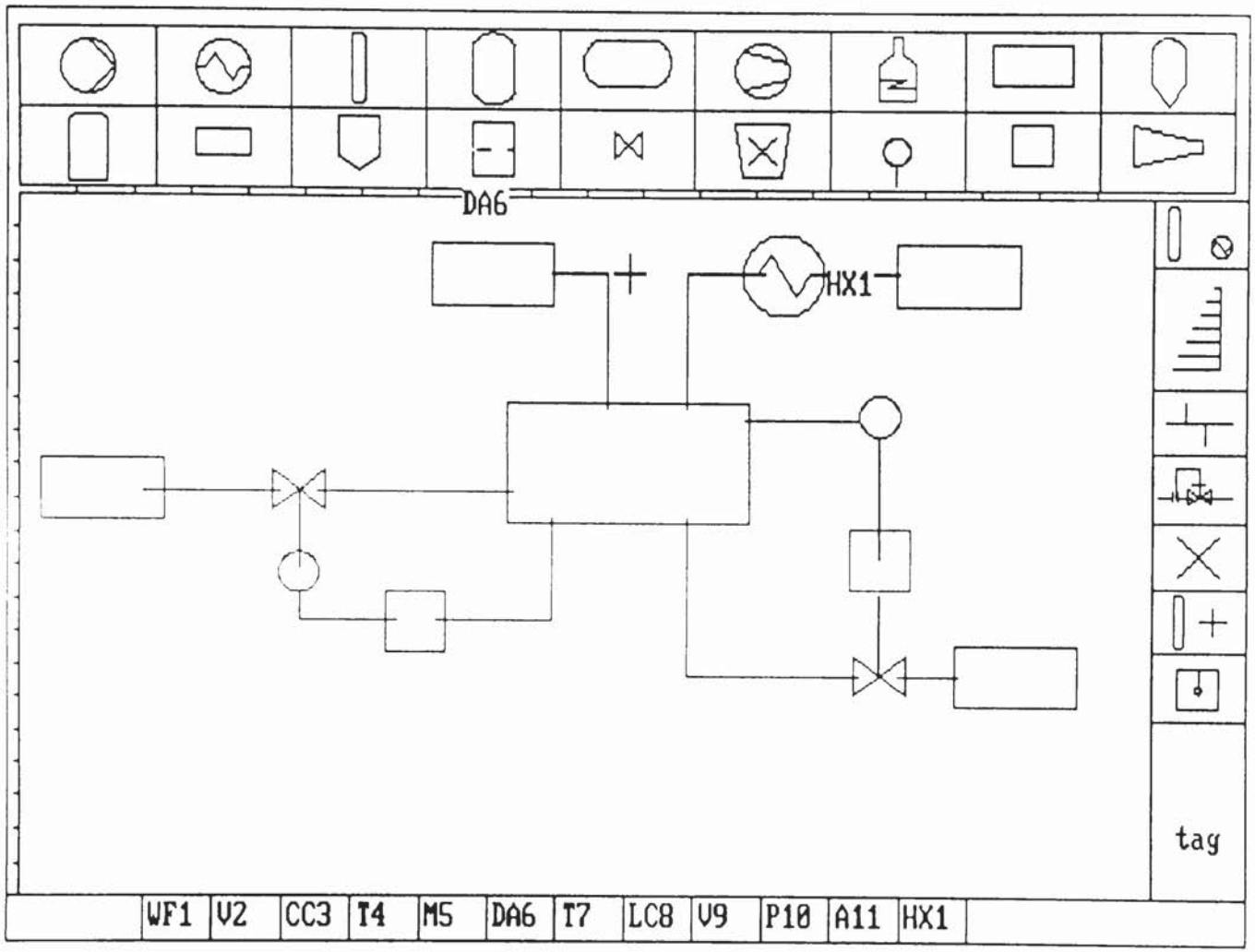


Figure 7.2 A PFG flowsheet of the methanol mixer tank system

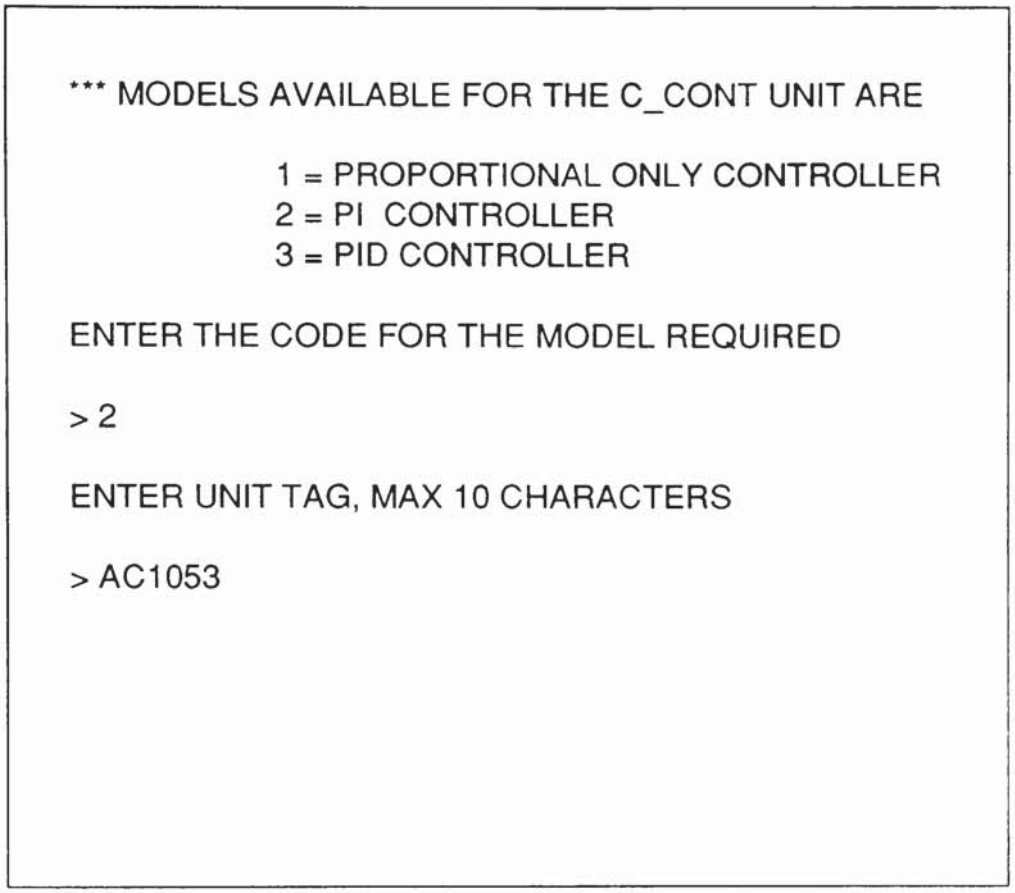
Unit Number	Equipment Tag Name	Equipment Name
1	WF1	Water feed
2	V2	Water feed valve
3	CC3	Composition controller
4	T4	Composition transmitter
5	M5	Mixer tank
6	DA6	Dilute methanol feeder
7	T7	Level transmitter
8	LC8	Level controller
9	V9	Outlet control valve
10	P10	Product sink
11	A11	Concentrated methanol feeder
12	DAH1	Dilute methanol heater

TABLE 7.1

The equipment list for the methanol mixer tank

The intermediate topology information file created by PFG is used by BTOPOL to interrogate the user. The file for the methanol mixer tank is given in Appendix A.

The purpose of the interrogation is to resolve the user's choice of the model type code assigned for each unit. The PFG unit code is used to retrieve information about the suitable model types from the database files. This information which includes the particular models available for each model type is displayed and the user is required to select the most appropriate model for the units in the flowsheet. An example of this is shown in Figure 7.3. There are other options available. These are to delete the PFG unit, and to add new PFG units, making appropriate changes to the topology.



THE MODELS AVAILABLE IN THE DASPII LIBRARY FOR PFG UNIT TYPE CONTROLLER ARE LISTED AND THE USER IS PROMPTED TO SELECT THE MOST SUITABLE.

Figure 7.3 Model selection for the units in the flowsheet

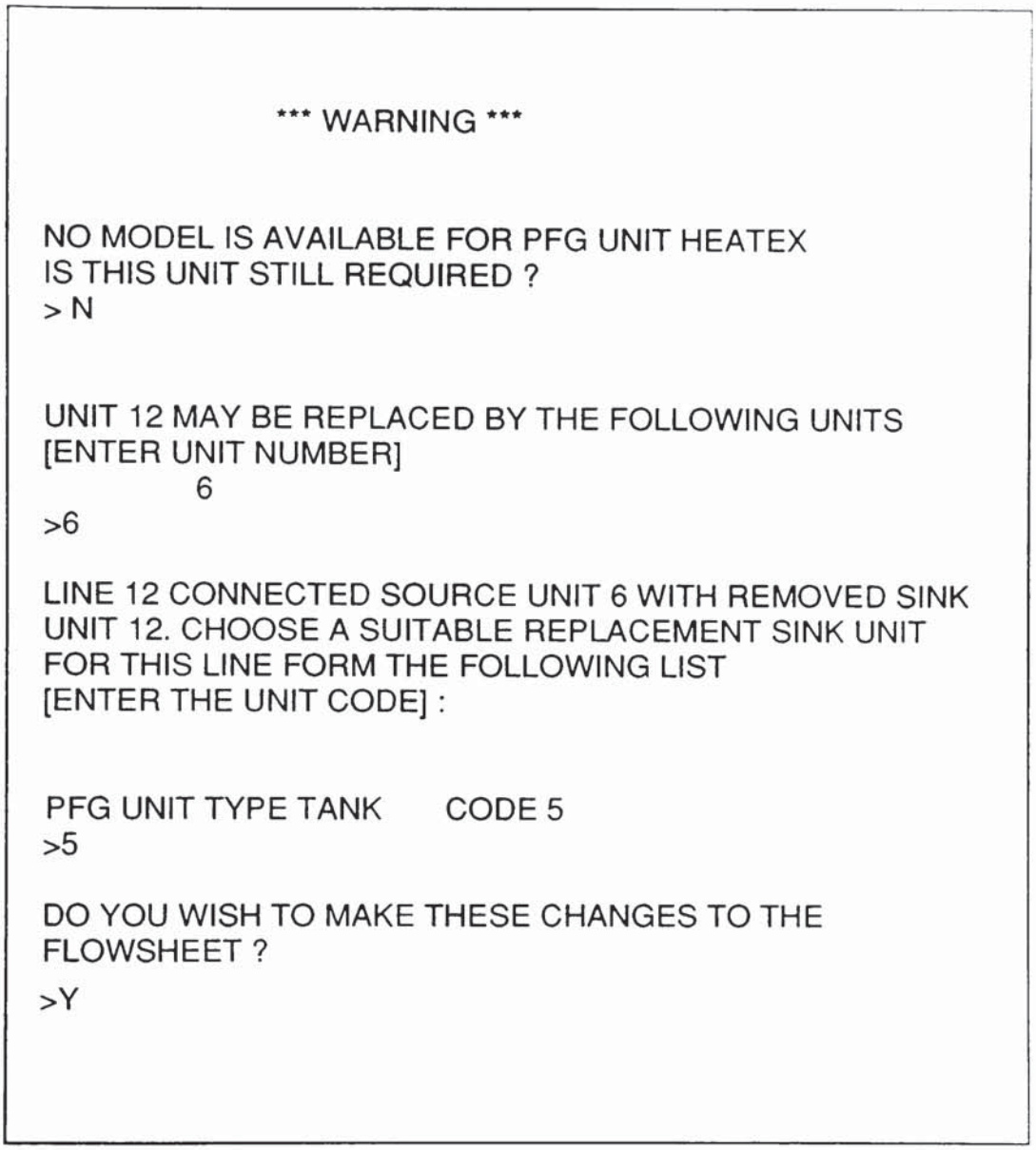


FIGURE 7.4 Selection of an alternative sink for an old stream

For the removal option the user is prompted to select an alternative sink unit for the old streams terminating at the unit (see Figure 7.4). The same changes are also made to the PFG file describing the block diagram. This gives the user the ability to modify the PFG diagram during an interactive text session to keep it consistent with the changed problem structure in the final topology file passed to DASPII. This is illustrated using the methanol mixer tank problem. This initially has a heat exchanger unit in the flowsheet. However, as there are no heat exchanger models in the DASPII library, the heat exchanger - unit 12 is removed resulting in the block diagram shown in Figure 7.5.

The topology information is completed by the user typing and labelling the streams as shown in Figure 7.6. All the user supplied information is merged with the information provided by PFG to produce a DASPII topology file. The topology file for the methanol mixer tank is shown in Appendix A.

7.2 Generation of the Problem Description information using CUNIN and CINDAN

In Section 7.2.1 the generation of the simulation control options file is undertaken for the methanol mixer problem with the aid of CINDAN which is described in Chapter 6. The next stage, shown in Section 7.2.2, is the creation of the problem description file for the methanol mixer problem using CUNIN, which is also described in Chapter 6.

7.2.1 Interactive generation of the simulation control information

A series of questions are posed to the user to produce a simulation control option file, as described in Chapter 6. Some of the questions depend on the options selected earlier during this procedure.

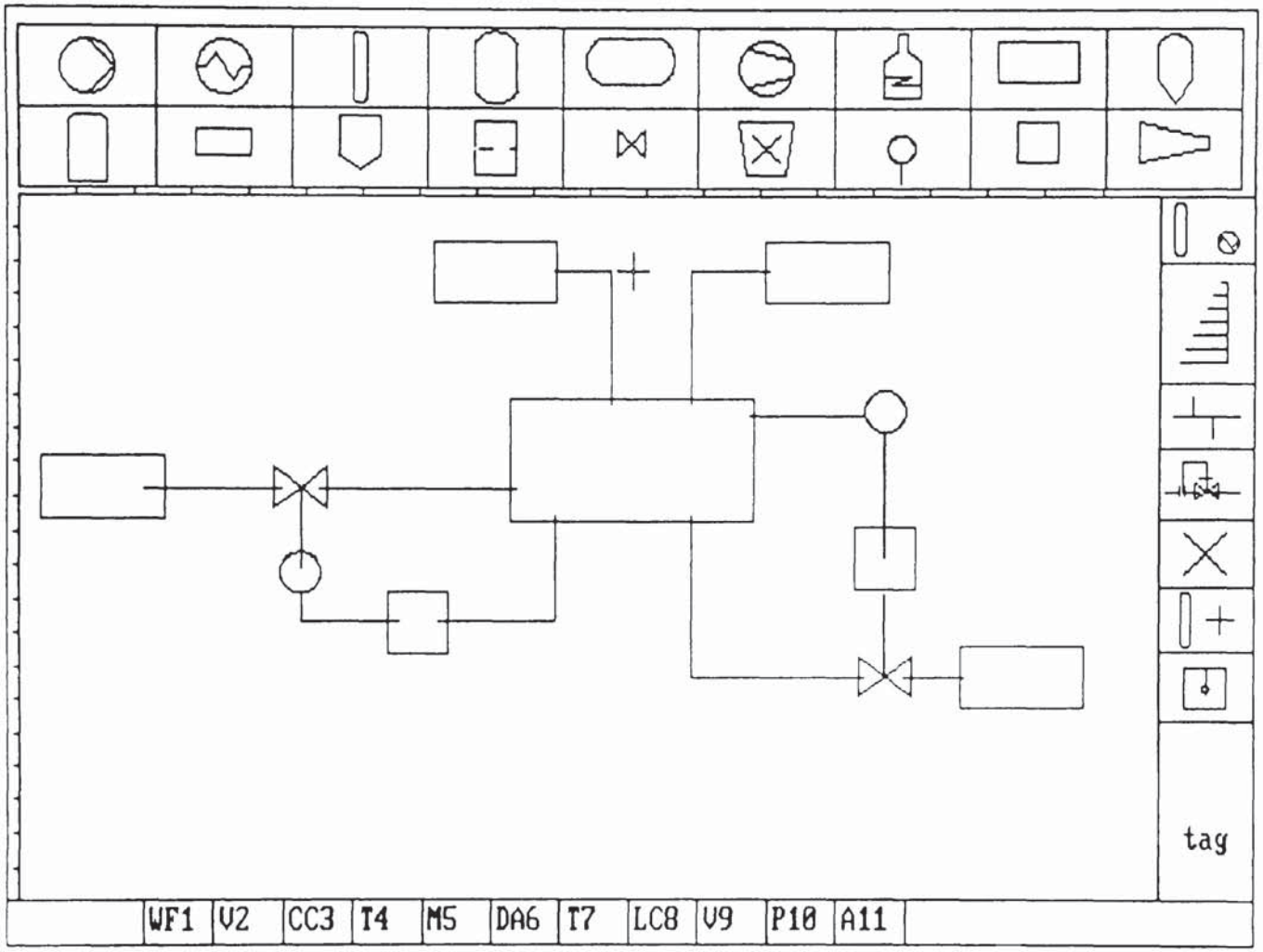
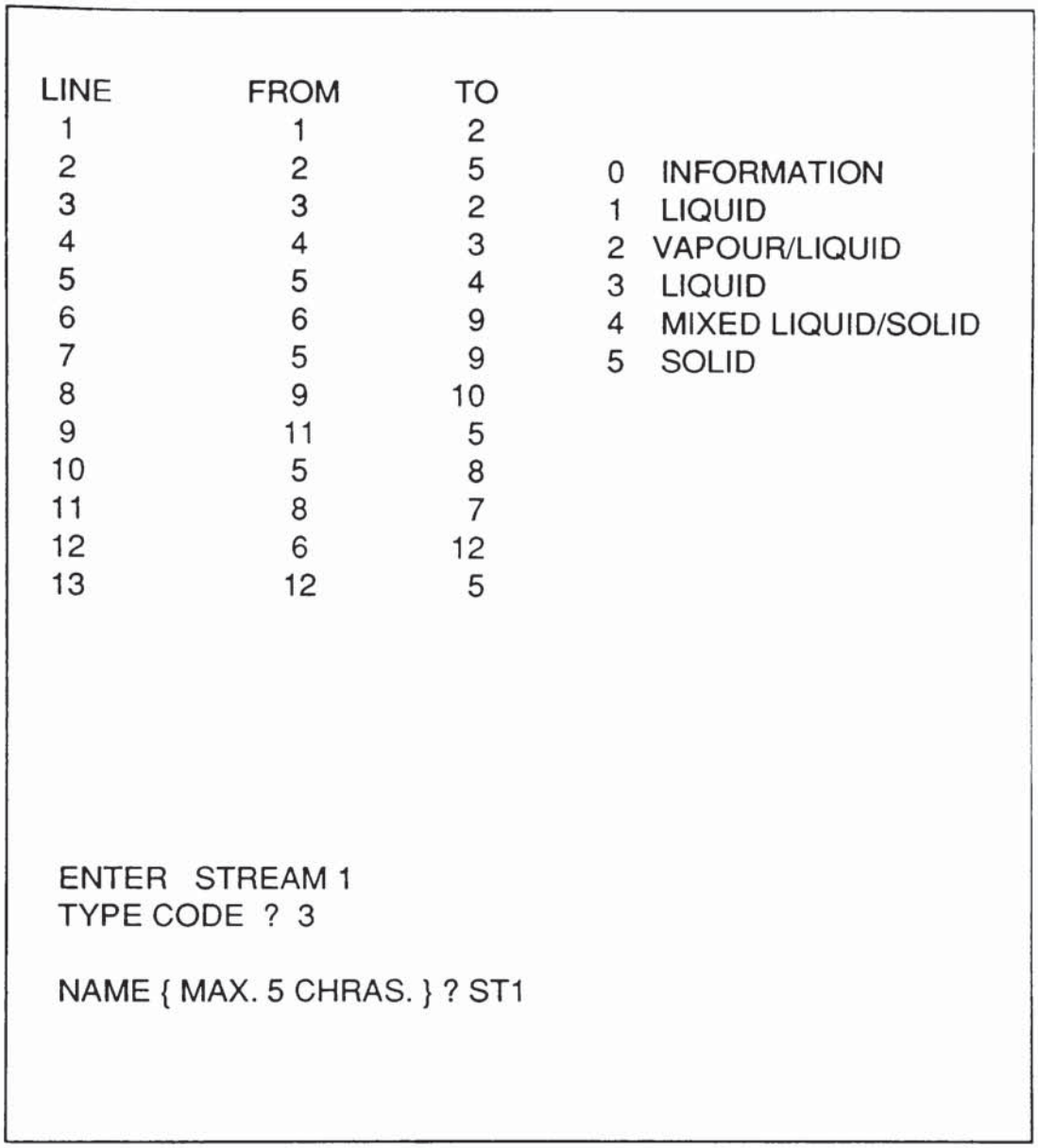
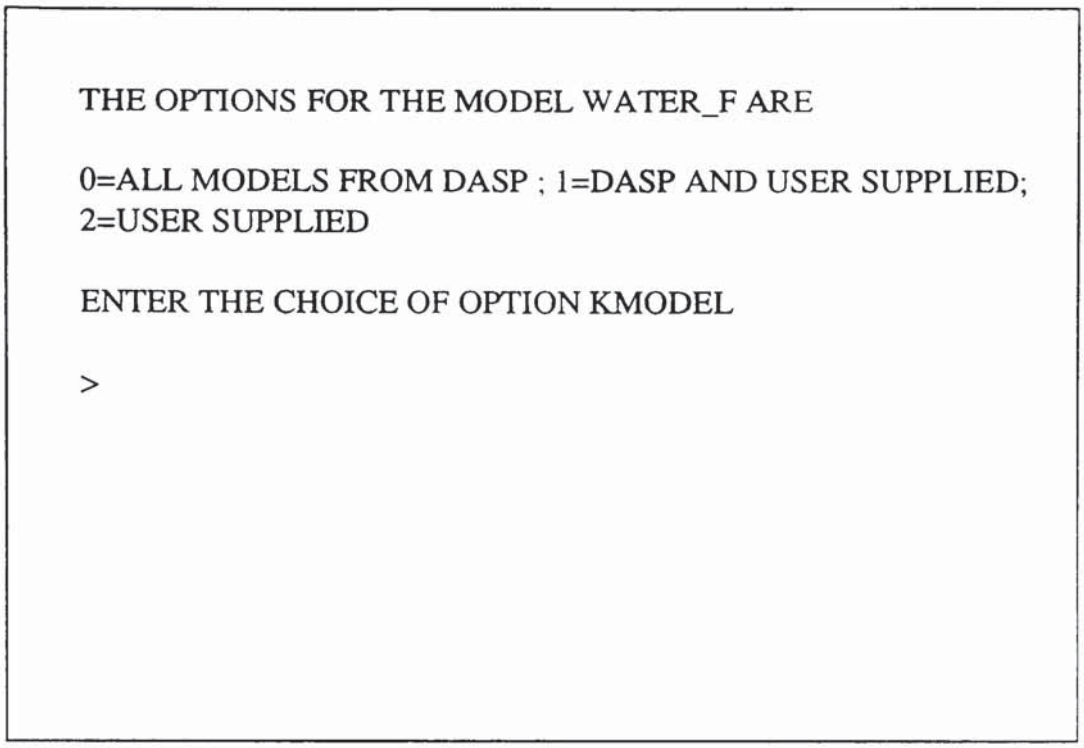


Figure 7.5 A PFG flowsheet of the methanol mixer tank system without a feed heat exchanger



THE SINK AND SOURCE OF EACH LINE IN THE FLOWSHEET IS DESCRIBED AND THE USER IS PROMPTED TO ENTER THE STREAM TYPE AND A STREAM TAG FOR EACH FLOWSHEET STREAM.

Figure 7.6 Stream typing and labelling



THE USER IS PROMPTED TO DESCRIBE THE SOURCE OF MODELS FOR THE FLOWSHEET SELECTED. THREE OPTIONS ARE AVAILABLE. THE INTERNAL OPTION NAME IS KMODEL

Figure 7.7 The standard prompt during the generation of the simulation control options

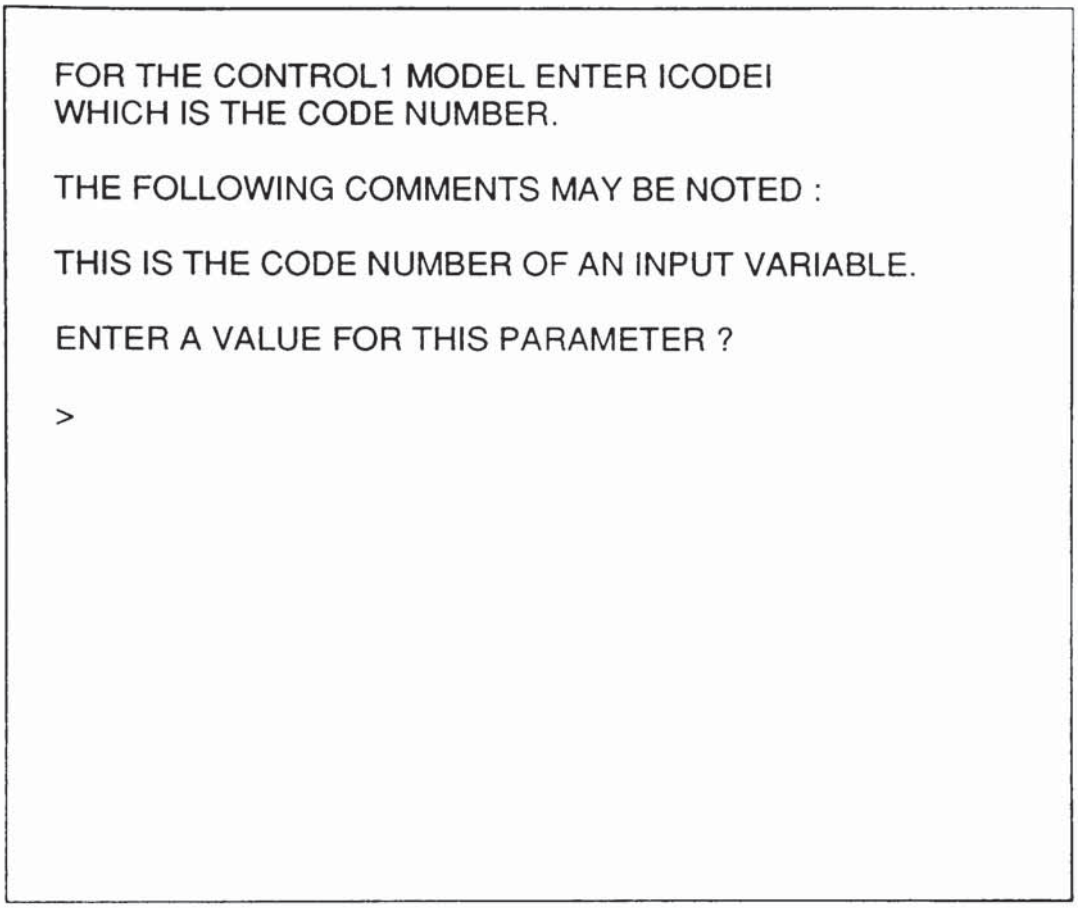


Figure 7.8 A typical integer parameter enquiry

```
*** DEFAULT UNKNOWN VARIABLES ARE ***

2 = PSIGN

PLEASE ENTER <ENTER> TO CONTINUE


DO YOU WISH TO KEEP THESE VARAIBLES
ENTER Y OR N ?
> N

HOW MANY  WOULD YOU LIKE TO CHANGE
{MAX 1} ?
>1

ENTER THE NUMBERS OF THE DEFAULT VARIABLES
YOU WISH TO CHANGE

CHOOSE 1 FROM

61  ZI          62  RI
63  ZO          64  RO
67  YO          68  AXN
69  GAIN        88  CMAN
71  TI          70  SP


>70
```

IN THIS EXAMPLE THE SET POINT OF A CONTROLLER HAS BEEN SELECTED AS THE UNKOWN VARIABLE INSTEAD OF THE CONTROL SIGNAL.

Figure 7.9 Selection of default real variables

The user prompts are standard questions appended with information from database files, an example of this is shown in Figure 7.7. A simulation control option file generated for the methanol mixer tank problem is shown in Appendix A.2.

7.2.2 Interactive generation of the variable specification

The generation of the variable specification for a system is carried out by CUNIN as described in Chapter 6. The process commences with the topology file (as created in Section 7.1) being read to determine the models required to simulate the system. The methanol mixer tank is used to illustrate the method adopted in DASPII for variable specification. This is followed by the retrieval of the integer parameter for each unit and displayed with the prompts requesting the user to provide suitable valves (see Figure 7.8). The next stage involves the selection of the known and unknown variables for the model All of the unknown variables allowed (see Chapter 6 for the variable classification) are displayed and the user is prompted to either select the defaults or choose from the list of real variables (see Figure 7.9).

The methanol mixer tank system is used to demonstrate variable specification switching. Two different variable specifications are used for this problem, in effect representing two different system configurations.

a) <u>The standard configuration - the controlled system</u>

Composition and level controllers are both active in this configuration (see Figure 7.1). This is achieved by declaring the controller set points as known variables. The controller outputs are declared as the unknown variables and are calculated by the controller models. The resulting variable specification file for this system is shown in Appendix A.2. The simulation results are shown graphically in Figures 7.10 to 7.13. It can be seen that the product

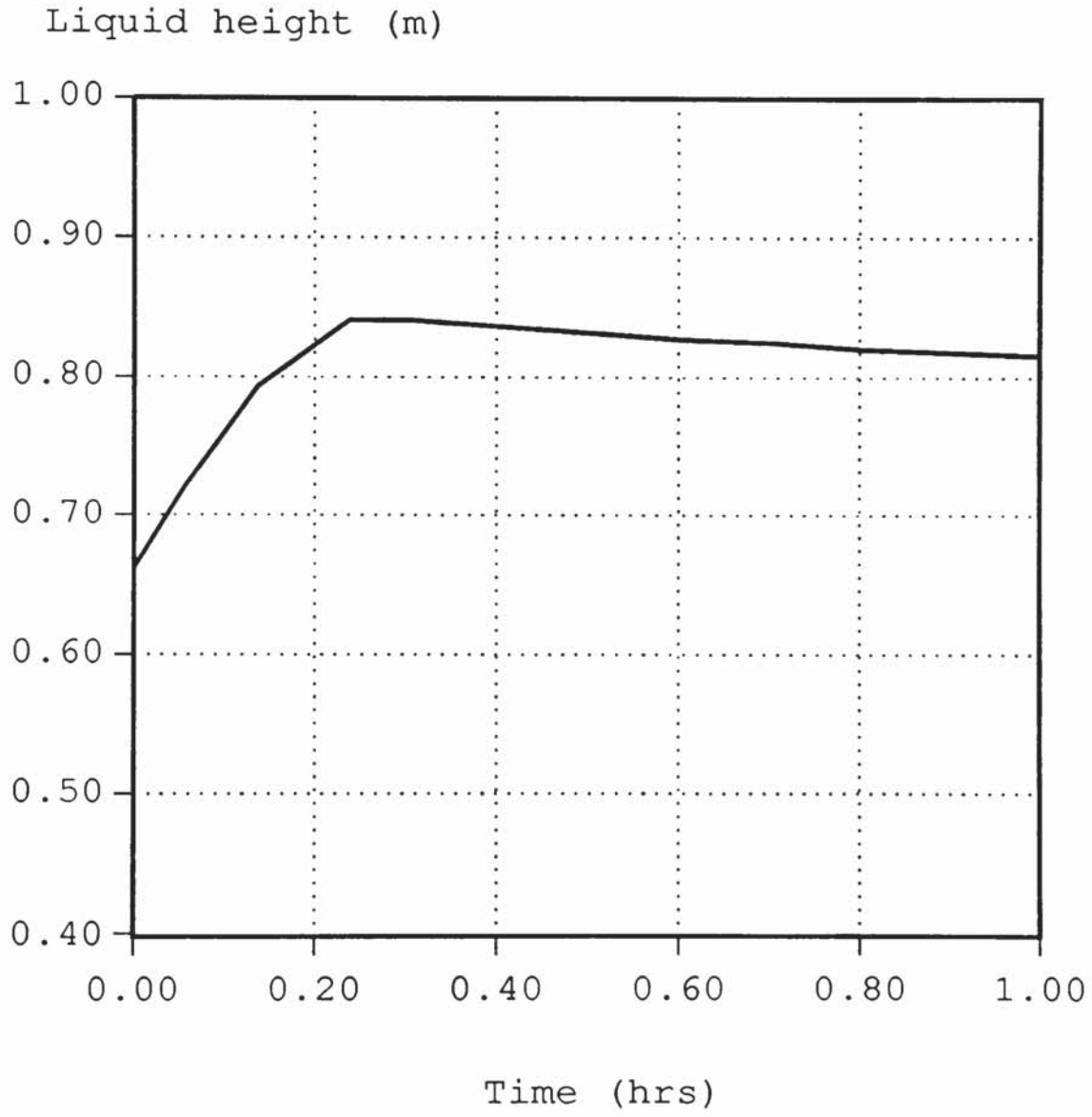


Figure 7.10 Tank level versus time for the controlled system

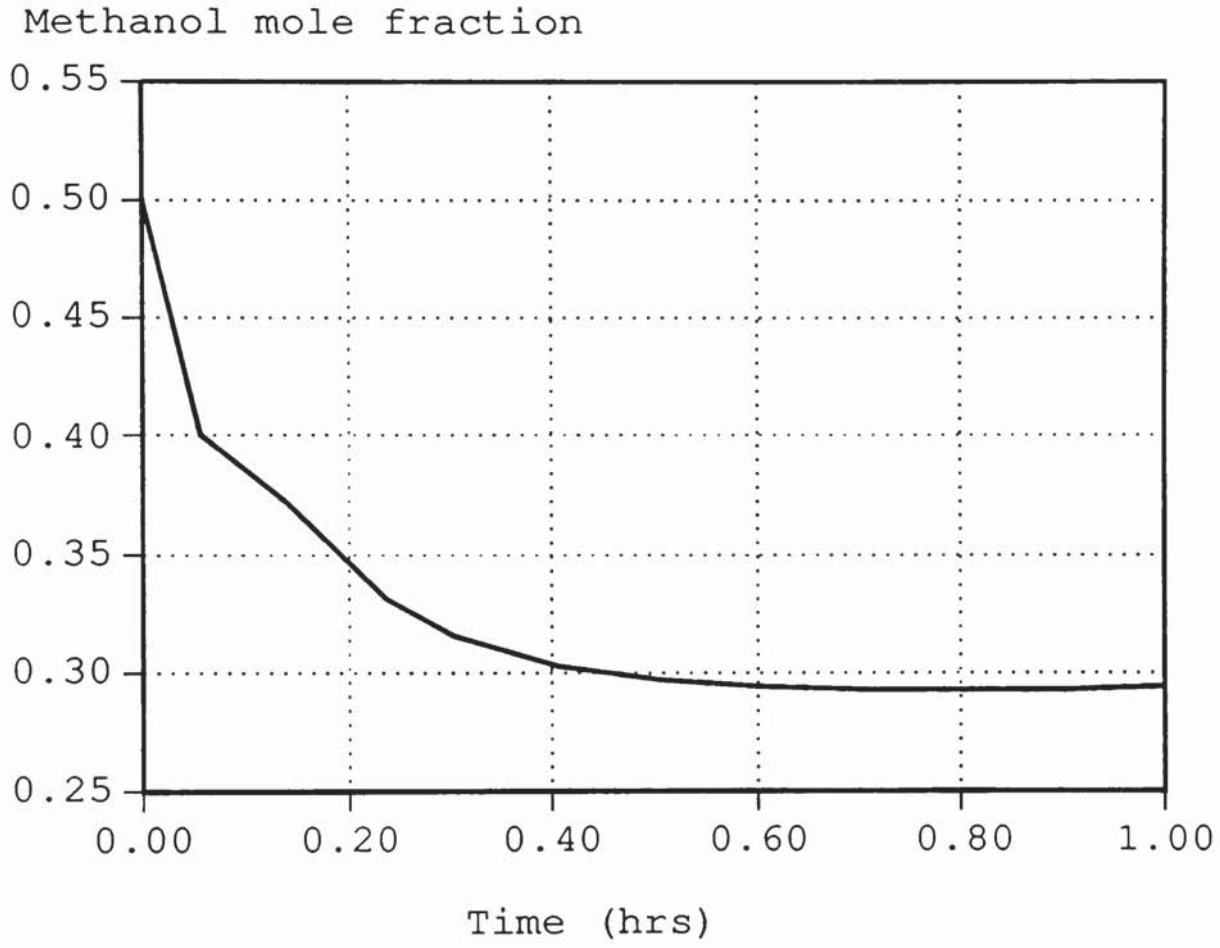


Figure 7.11 Methanol mole fraction in the tank versus time for the controlled system

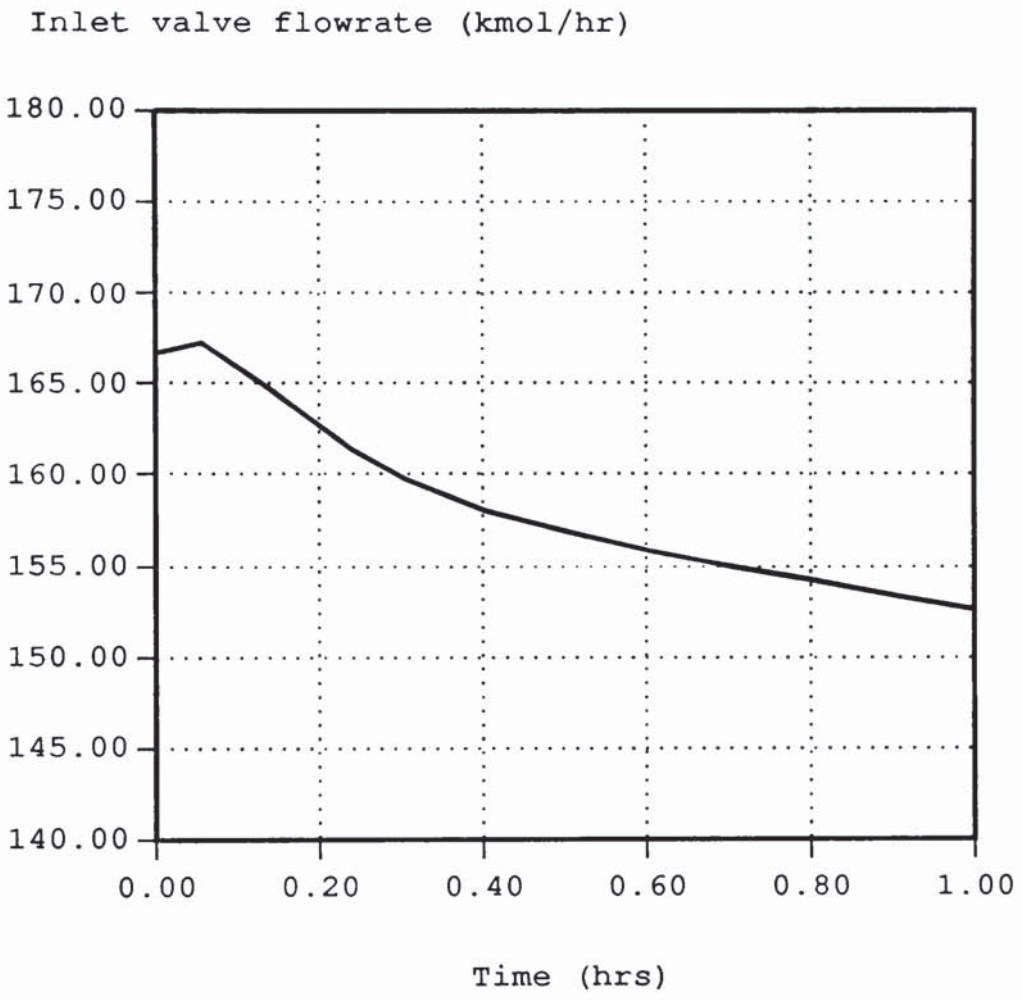


Figure 7.12 Inlet water flowrate versus time for the controlled system

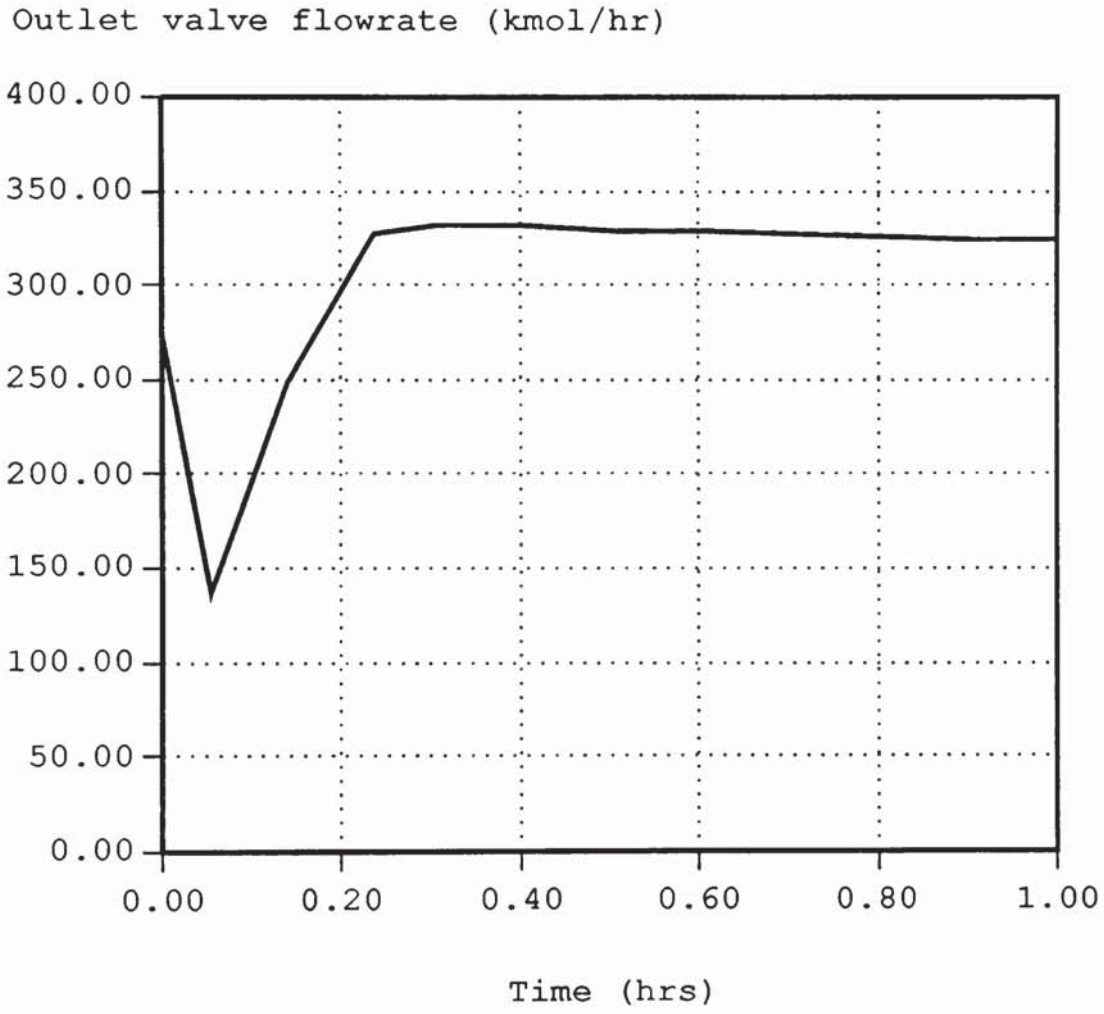


Figure 7.13 Liquid flowrate through tank outlet valve
for the controlled system

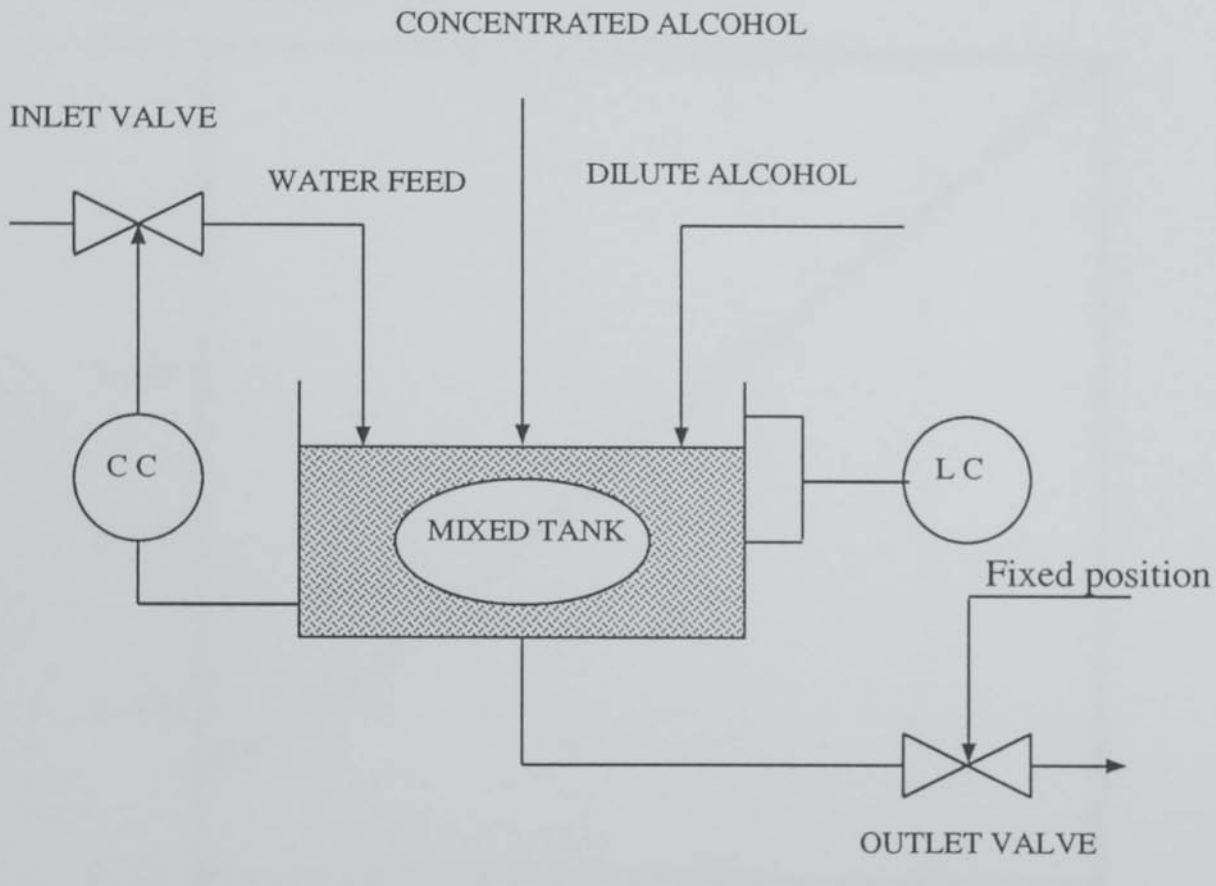


Figure 7.14 Flow diagram of the methanol mixer tank in the uncontrolled system

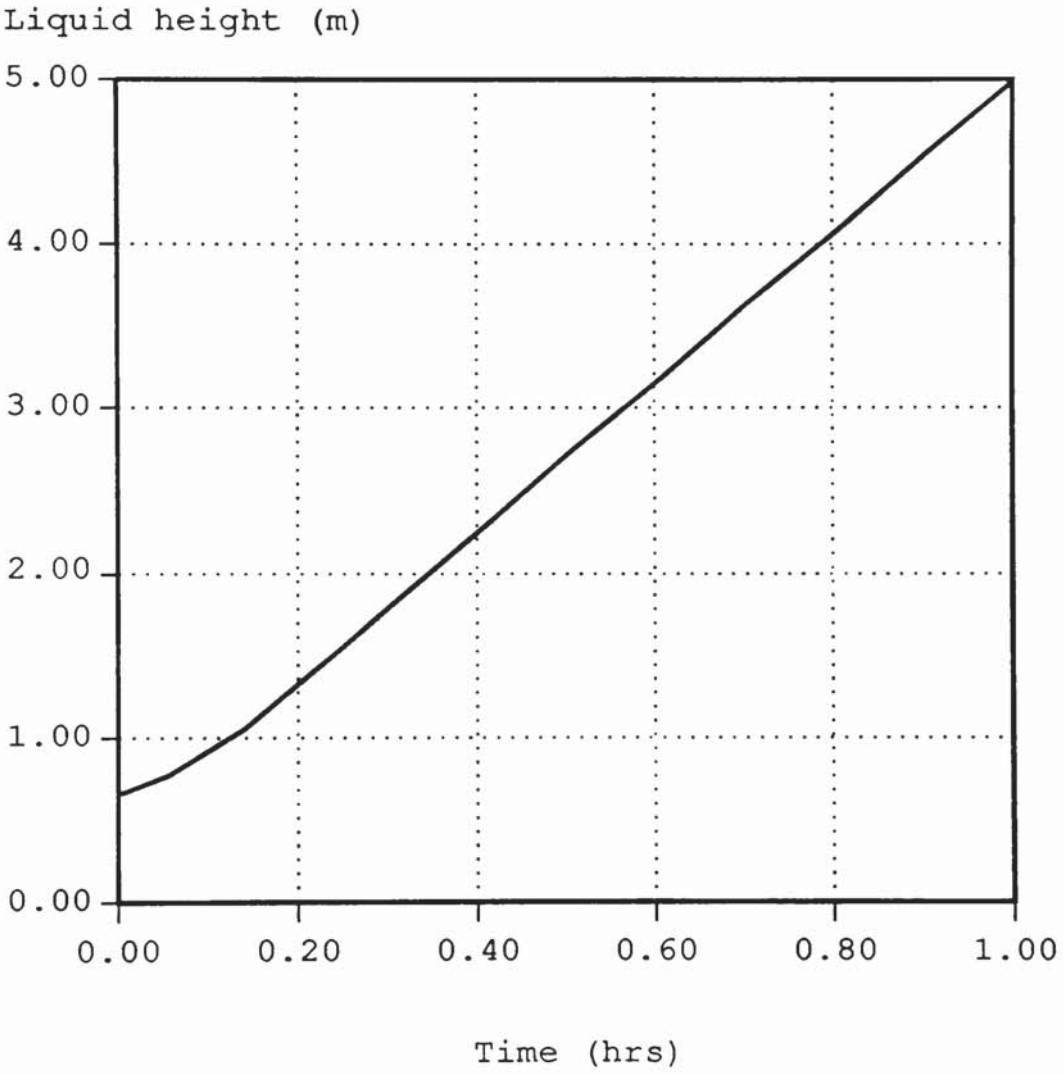


Figure 7.15 Tank liquid level versus time for the uncontrolled system

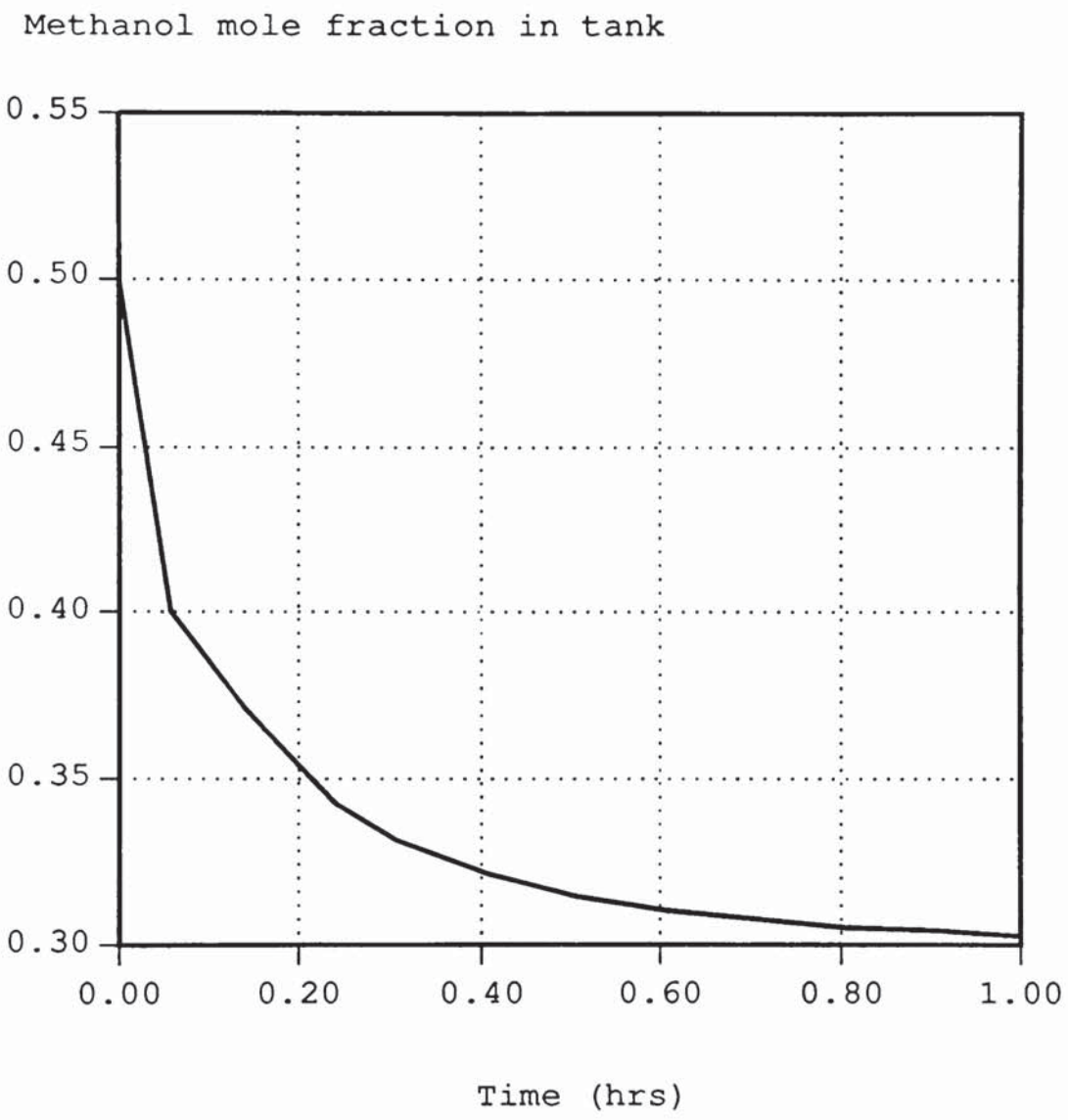


Figure 7.16 Methanol mole fraction in tank versus time for the uncontrolled system

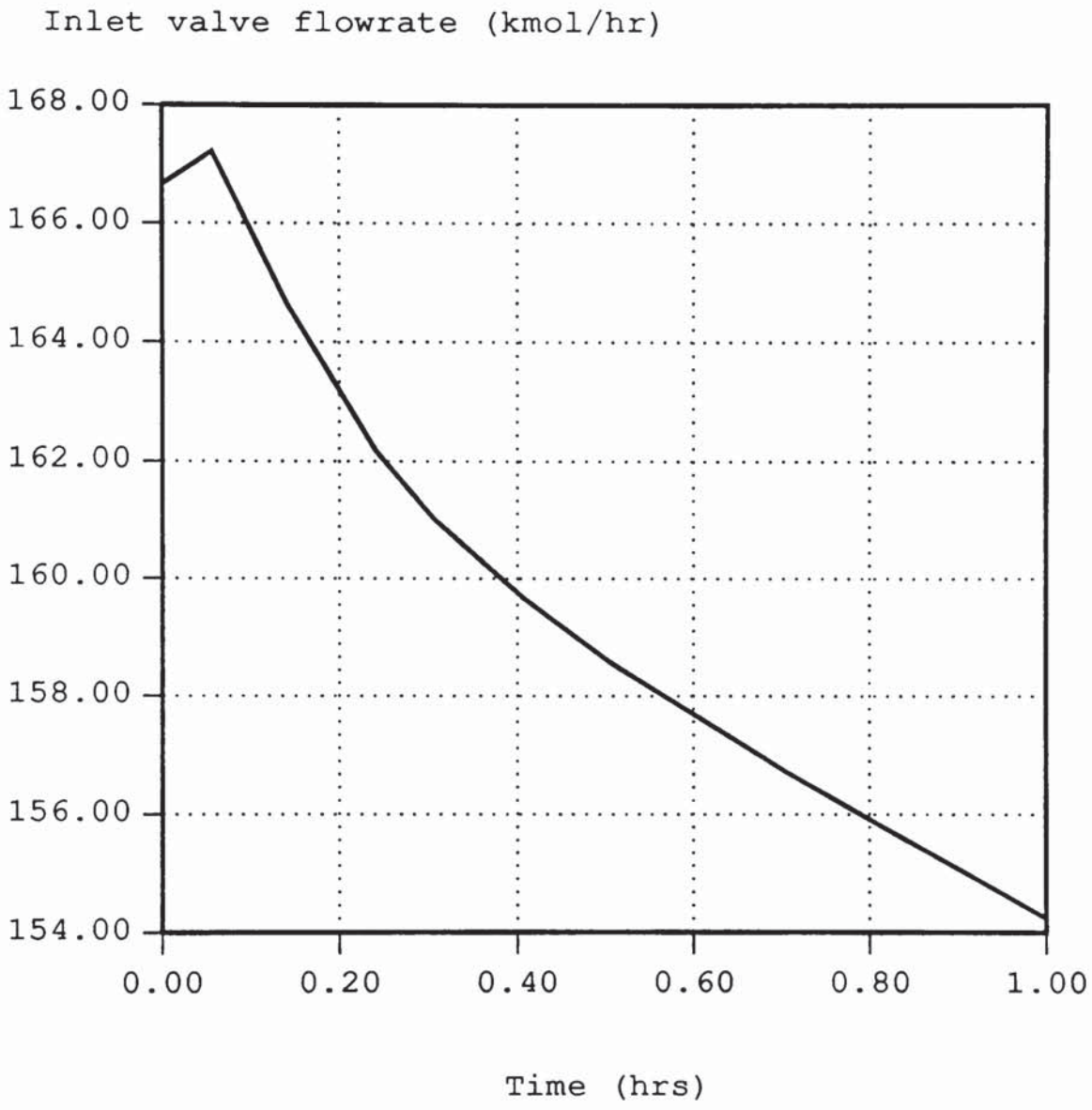


Figure 7.17 Water flowrate into the tank versus time
for the uncontrolled system

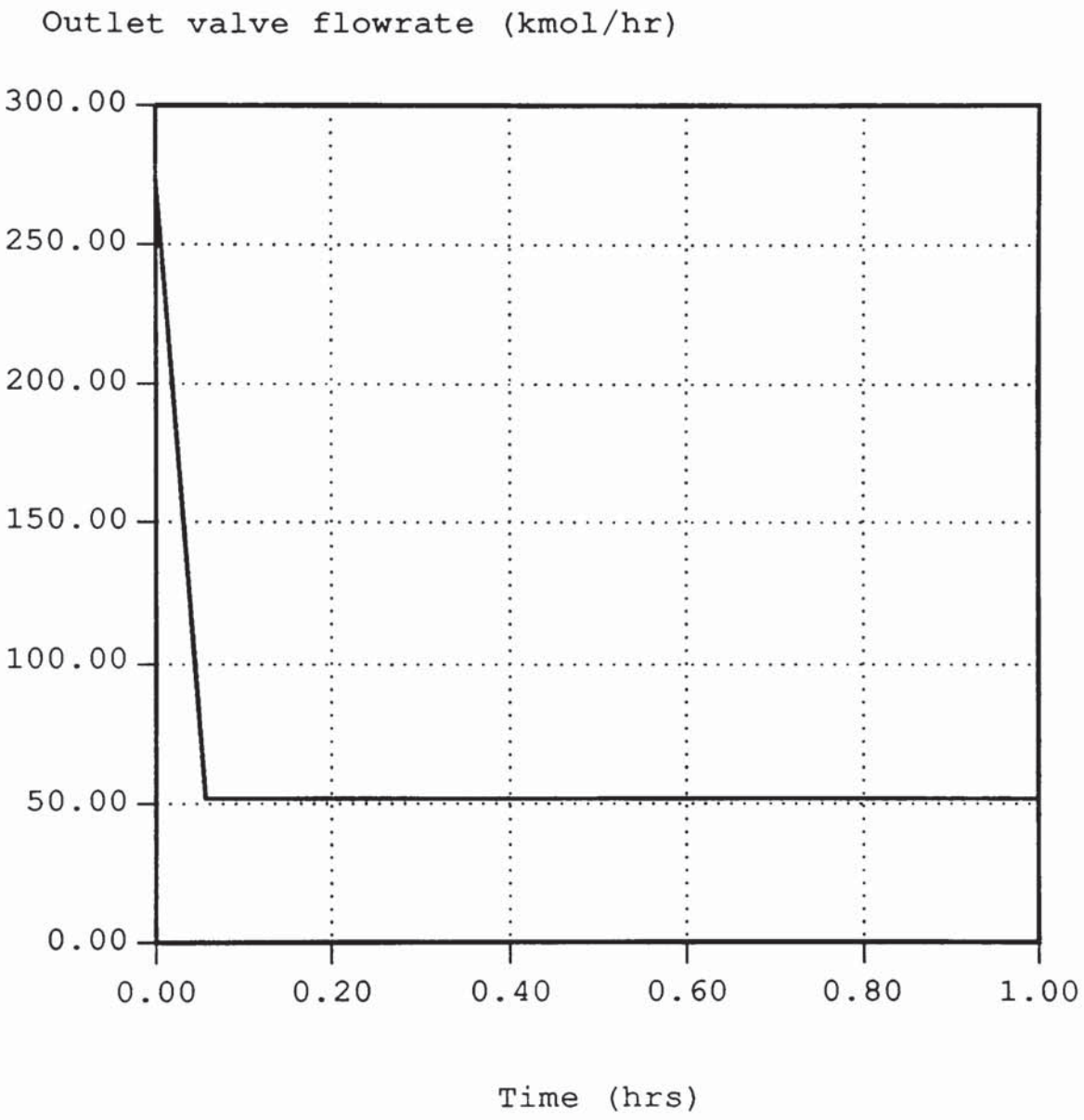


Figure 7.18 Liquid flowrate through the tank outlet valve versus time for the uncontrolled system

stream is suffering from underdamped level control, whilst the composition controller exerts more stable control over the water dilution stream. These results are consistent with the expected results and also with those obtained by Smith (1985). The different controller responses can be attributed to the widely differing gains for the two controllers. Figures 7.10 and 7.11 show that the level is controlled quite closely to the set point of 0.8 m, whilst, the composition controller offers poor performance in maintaining a methanol tank concentration of 0.4 mole fraction.

b) The alternative configuration - semi controlled system

In this system (see Figure 7.14) the controller output is fixed (known variable) by allowing the set point (unknown variable) of the controller to float. This can be achieved by producing another variable specification input file by using CUNIN or by simply modifying the input file used in Case A, resulting in the file shown in Appendix A.2. In can be seen from the graphical results shown in Figures 7.15 to 7.18 that the product flowrate is constant, whilst the level in the tank is varying and not controlled. On the other hand, the concentration controller is still attempting to maintain the correct methanol concentration by manipulating the water flow. The level controller is operating but not controlling as the set point in varing to maintain a constant output.

This example illustrates the ease with which the model variable specification can be modified in DASPII. This technique is available for any variables which are suitably described in the unit model.

7.3 The Solution of Non-linear algebraic equations

7.3.1 Comparison of CONLES with Broyden's method

Chapter Four describes the solution of non linear algebraic equations and its importance in the simulation of chemical processes. Several literature problems will be used to compare

CONLES, the new NLAE solver implemented in DASPII, with Broyden's method, the method used in DASPI.

Problem 1. Solution for reaction rate equations (Shacham (1986))

The equations in this problem represent the rate of different steps in a chemical reaction. The equation set is as follows:

$$r_1 = 1 - X_1 - K_1 X_1 X_6 + Kr_1 X_4$$

$$r_2 = 1 - X_2 - K_2 X_2 X_6 + Kr_2 X_5$$

$$r_3 = -X_3 + 2K_3 X_4 X_5$$

$$r_4 = K_1 X_1 X_6 - Kr_1 X_4 - K_3 X_4 X_5$$

$$r_5 = 1.5 (K_2 X_2 X_6 - Kr_2 X_5) - K_3 X_4 X_5$$

$$r_6 = 1 - X_4 - X_5 - X_6$$

where X_1 to X_6 are the quantities in moles of the different species present in this reaction; K_1, Kr_1, K_2, Kr_2 and K_3 are specified rate coefficients, and r_1 to r_6 are the reaction rates. At steady state the reaction rates are zero. The coefficients values are given in Appendix A.

This is a problem that has several real solutions although most of the solutions are infeasible. Two initial sets of estimates are used for this system, both are shown in Appendix A.1. Initial estimate set B is quite different to initial estimate set A, which is used by Shacham (1986). Set A is a perfectly valid initial estimate when solving this problem using the NLAE solver in standalone format. However, set B is more likely to be used in a simulation environment because similar types of variables are assigned with the same starting points, as providing individual initial estimates for a large set of variables would prove an unnecessary overhead. It can be seen from the results in Appendix A, that Broyden's method and CONLES both converge to solutions from the initial estimate Set A although the latter requires more iterations to converge. Broyden's method fails to converge after 500 iterations with initial estimate set B, whereas CONLES solves the problem after 85 iterations.

The results obtained from CONLES for this problem are different from those obtained by Shacham (1986, 1983) for this problem. The discrepancy between the results is quite considerable, the difference in one variable being as large as 550%. It is clear that either or both of the results are not correct or there is more than feasible solution. However, if Shacham's results (1986) are substituted in to the equations the residual values of the first and third equations are much larger than any realistic steady state convergence tolerance. Whereas, the results obtained by CONLES in this work yields residual values that can be classified as converged. It is likely that there is a typographical error in Shacham (1986) .

Problem 2. Combustion of propane in air (Hiebert (1982))

This chemical equilibrium problem represents the combustion of propane in air and consists of 10 equations. The equation set is as follows:-

$$f(1) = X_1 + X_4 - 3 = 0$$

$$f(2) = 2X_1 + X_2 + X_4 + X_7 + X_8 + X_9 + 2X_{10} - R = 0$$

$$f(3) = 2X_2 + 2X_5 + X_6 + X_7 - 8 = 0$$

$$f(4) = 2X_3 + X_5 - 4R = 0$$

$$f(5) = X_1 + X_5 - 0.193\, X_2\, X_4 = 0$$

$$f(6) = X_6\, (X_2)^{1/2} - 0.002597\, (X_2\, X_4\, X_5)^{1/2} = 0$$

$$f(7) = X_7\, (X_4)^{1/2} - 0.003448\, (X_1\, X_4\, X_5)^{1/2} = 0$$

$$f(8) = X_8\, X_4 - 1.799 \times 10^{-5}\, X_2\, X_5 = 0$$

$$f(9) = X_9\, X_4 - 2.155 \times 10^{-4}\, X_1\, (X_3\, X_5)^{1/2} = 0$$

$$f(10) = X_{10}\, X_4^2 - 3.846 \times 10^{-5}\, X_4^2\, X_5 = 0$$

where

$$X_S = \sum_{i=1}^{10} X_i$$

$\{X_i\}$ are the amounts of chemicals which must be non-negative at the solution.

R is the parameter of value 10

The functions in the equation set containing square root terms cannot be evaluated when any of the variables forming arguments of the square root terms are negative. This would yield complex solutions which are not detected by this type of solver. As in problem (1) two different starting points were used (see Appendix A.1). The first set is again taken from Shacham (1986) whilst the second is a more realistic one to use in a simulation environment. Broyden's method solved this problem from the first initial in the same number of iterations as CONLES, although the former method required fewer function and Jacobian evaluations. With the second starting set Broyden's method failed to converge in 500 iterations, whilst CONLES converged to the required solution well within this limit of iterations.

Problem 3. Chemical equilibrium problem (Hiebert (1982))

The following equations represent a chemical equilibrium system:-

$$f_1(X) = X_1 + X_2 + X_4 - 0.001$$

$$f_2(X) = X_5 + X_6 - 55.0$$

$$f_3(X) = X_1 + X_2 + X_3 + 2X_5 + X_6 - 110.001$$

$$f_4(X) = X_1 - 0.1X_2$$

$$f_5(X) = X_1 - 10^4 X_3 X_4$$

$$f_6(X) = X_5 - 55 \times 10^{14} X_3 X_6$$

Four of the equations are linear and the remaining equations are mildly non-linear, however the difficulty with this equation set arises with their scaling. The most extreme example of this is the sixth equation which has a coefficient for the second term of 55×10^{14}, whereas the coefficient for the first term has a value of 1.0. The solution requires all variables to have positive values. Four starting points, (Appendix A.1) were used for this problem and they

were the same as those used by Hiebert (1982). These were used as they represented most possible starting points. For instance one starting point is near the actual solution, the other three are uniform values similar to those that might be used in a simulator. The results in Appendix A show that both CONLES and Broyden's method solve the problem for all the starting points. These results differ slightly with those obtained by Hiebert, who failed to obtain solution with starting point A when using a Quasi-Newton method. However, Broyden's method solved the problem from the same starting point. In all these runs CONLES required a greater number of iterations and therefore function evaluations than Broyden's method. This means that badly scaled problems are well handled by Broyden and CONLES.

Problem 4. Eight variable problem (Ferrari's and Tronconi (1986))

Ferraris and Tronconi (1986) put forward this eight variable problem as a good test for NLAE solvers from bad starting estimates. The equations are as follows:-

$$f_1(X) = X_1$$

$$f_2(X) = X_2 - X_1^{1/2} - \exp(X_1) - 15$$

$$f_3(X) = X_3 - X_1 X_2 /100 - \sin(X_1) - 1.0$$

$$f_4(X) = X_4 - (X_2 + X_1 + X_3/2)^2 + 150$$

$$f_5(X) = X_5 - (X_4 - X_2)/(X_1 X_2 X_3)$$

$$f_6(X) = X_6 - X_5 X_1^{1/3} - \exp(X_5)$$

$$f_7(X) = X_7 - (X_1 - X_4^{1/2} - X_5^2).\exp(X_5)$$

$$f_8(X) = X_8 - X_1 - X_5 - X_6 - X_3$$

The starting points, shown in Appendix A, are intentionally selected far from the solution. This is further exacerbated close to the solution where the Jacobian becomes ill-conditioned as the exp terms are much greater than the remaining terms in these equations. Powell's method (1970) was unable to solve this problem in 600 iterations. A similar result was obtained with Broyden's method which also failed to converge after 500 iterations. Whereas Ferraris and Tronconi (1986) solved this problem in 38 iterations, with their algorithm which updates only the non zero, non constant elements in the Jacobian matrix. As shown in Appendix A, CONLES solved this problem in 267 iterations.

Problem 5. Partial oxidation of methane (Carnahan (1969))

Seven equations are used in this problem to represent the material and energy balances for the possible reactions between methane and oxygen. One equation is linear whilst the other equations show varying degrees of non linearity. Multiple solutions are possible for this problem with the feasible solution having positive mole fractions. The equation set is as follows:-

$$f_1(X) = 0.5X_1 + X_2 + 0.5X_3 - X_6/X_7$$

$$f_2(X) = X_3 + X_4 + 2X_5 - 2/X_7$$

$$f_3(X) = X_1 + X_2 + X_5 - 1/X_7$$

$$f_4(X) = -28837X_1 - 139009X_2 - 78213X_3 + 18927X_4 + 8427X_5 + 13492/X_7 - 10690X_6/X_7$$

$$f_5(X) = X_1 + X_2 + X_3 + X_4 + X_5 - 1$$

$$f_6(X) = 400X_1 X_4^3 - 1.7837 \times 10^5 X_3 X_5$$

$$f_7(X) = X_1 X_3 - 2.6058 X_2 X_4$$

Two starting points, originally proposed by Carnahan (1969), were used for this problem. Both starting points are shown with the results in Appendix A.1. The results show that from the first starting point both CONLES and Broyden's method easily solve the problem. Unlike the previous systems CONLES requires only 60% of iterations required by Broyden's method to solve the equation system. With the second starting point Broyden's method fails to converge in 500 iterations whilst CONLES is able to solve the equation set in 31 iterations. The second starting point is actually closer to the solution than the first starting point, this means that the solution path is not a simple function but very much non linear.

7.3.2 Solution of NLAEs for multiple solutions using CONSOL

Most chemical engineering problems yield NLAEs with multiple solutions, some of which are infeasible. However, all of the solvers used in chemical process simulators implement methods which are able to locate only a few of the real solutions depending on the starting points used. CONSOL will be used to evaluate the viability of determining all solutions to NLAEs. Several literature problems will be used to evaluate CONSOL.

Problem 1. Burning of fuel (Morgan (1987))

This equation set represents the combustion of fuel in a combustion chamber. The aim is to evaluate the composition of various components at a point during the combustion phase.

The coefficients of the system are derived from reaction constants and component conversion totals which are now related to temperature. The model equations are:-

$$f_1(X) = X_2 + 2X_6 + X_9 + 2X_{10} - T_H$$

$$f_2(X) = X_3 + X_8 - T_C$$

$$f_3(X) = X_1 + X_3 + 2X_5 + 2X_8 + X_9 + X_{10} - T_o$$

$$f_4(X) = X_4 + 2X_7 - T_N$$

$$f_5(X) = K_1 X_5 - X_1^2$$

$$f_6(X) = K_2 X_6 - X_2^2$$

$$f_7(X) = K_3 X_4 - X_4^2$$

$$f_8(X) = K_4 X_8 - X_1 X_3$$

$$f_9(X) = K_5 X_9 - X_1 X_2$$

$$f_{10}(X) = K_6 X_{10} - X_1 X_2^2$$

where T_o,T_H,T_c and T_N are totals, K_1 to K_7 are the reaction coefficients. The total degree of the system (see Chapter Four) is (2^5) or 96. This means there are 96 possible solutions for this equation system. However the equation set can be reduced so that the number of possible

solutions are reduced thereby reducing the computational effort required for solution. A possible reduced equation is:-

$$f_1^1(X) = a_{11} X_1 X^2 + a_{12} X_1 X_2 + a_{13} X_2^2 + a_{14} X_2 + a_{15}$$

$$f_2^1(X) = a_{21} X_1^3 + a_{22} X_1^2 X_2 + a_{23} X_1^2 + a_{24} X_1 X_2^2 + a_{25} X_1 X_2 + a_{26} X_1 + a_{27}$$

where a_{11} ..... a_{15} and a_{21} ..... a_{27} are parameters generated from the reduction of the original equation set. The new equation set has a total degree of 3 x 3 i.e. 9. These solutions are shown in Appendix A.4, where one of the solutions is complex and the remaining eight are real. Of these real solutions only one solution is feasible for this problem, the other solutions yield solutions with negative values. As these variables represents component compositions negative values are impossible.

Problem 2. Chemical Equilibrium problem (Hiebert (1982))

This problem has already been described in section 7.3.1. It is composed of the following six equations:-

$$f_1(X) = X_1 + X_2 + X_4 - 0.001$$
$$f_2(X) = X_5 + X_6 - 55.0$$
$$f_3(X) = X_1 + X_2 + X_3 + 2X_5 + X_6 - 110.001$$
$$f_4(X) = X_1 - 0.1 X_2$$
$$f_5(X) = X_1 - 10^4 X_3 X_4$$
$$f_6(X) = X_5 - 55 \times 10^{14} X_3 X_6$$

Hiebert (1982) and Shacham (1985) both attempted to solve this problem with various codes implementing a range of non-linear algebraic equation solving methods. They independently found that several methods including Powell's method and Brent's method

converged to a number of infeasible solutions depending on the initial starting point selected. CONSOL offers a technique for solving for all possible solutions to a problem without the requirement of starting point estimates. The equation set in its original form has a total degree of 2.2. Therefore this system has only four solutions and does not require system reduction. The four solutions obtained by CONSOL are shown in Appendix A.4. It can be seen that three of the solutions are real with the remaining one being a complex solution,. Two of the real solutions are indeed infeasible and may have been those located by Hiebert (1982) and Shacham (1985) test codes. The infeasibility of these solutions is due to of the existence of several negative variable values. The feasible real solution obtained by CONSOL was consistent with those obtained by Hiebert (1982) and Shacham (1985).

Problem 3. Solution of reaction rate equations (Shacham (1986))

This equation set which represents a chemical reaction system has been solved by CONLES and is problem number 1 in Section 7.3. The equation set is:-

$$f_1(X) = 1 - X_1 - K_1 X_1 X_6 + Kr_1 X_4$$

$$f_2(X) = 1 - X_2 - K_2 X_2 X_6 + Kr_2 X_5$$

$$f_3(X) = - X_3 + 2K_3 X_4 X_5$$

$$f_4(X) = K_1 X_1 X_6 - Kr_1 X_4 - K_3 X_4 X_5$$

$$f_5(X) = 1.5 (K_2 X_2 X_6 - Kr_2 X_5) - K_3 X_4 X_5$$

$$f_6(X) = 1 - X_4 - X_5 - X_6$$

The parameter vales K_1, Kr_1, K_2, Kr_2 and K_3 are specified in Appendix A.4, X_1 to X_6 are moles of different species present in the species. Multiple solutions have been found by Shacham (1986) for this solution. As discussed in Section 7.1 it appears that Shacham (1986) has made a mistake in either defining the equations or the parameter values, since the results obtained from CONLES are not consistent with those reported by Shacham (1986, 1983).

In the current form the total degree of this equation set is 2^5 or 32. Hence 32 possible solutions exist for this problem. Equation reduction to reduce the total degree of the system was not possible. Therefore the equation set was solved in its original form and 32 solutions were located.

Three of the 32 solutions are found to be real, whilst the remaining 29 solutions are complex. Of the three real solutions two are infeasible because of the existence of negative variable values. The remaining real solution is feasible and is consistent with the solution obtained with CONLES. The two infeasible real solutions are different from the two infeasible solutions obtained by Shacham (1986), further evidence that the equation set proposed by Shacham (1986) is not the set used to obtain the results reported.

7.4 CONCLUSION

A simulation problem has been successfully undertaken, starting from the generation of the topology information through to the actual simulation. The system considered is simple but representative of the type of problem generally simulated in chemical engineering process analysis. The simulation analysis commenced with the generation of the process topology with the aid of a modified PFG. The ability to modify the graphical flowsheet during the conversational procedure has been demonstrated. A problem variable definition was successfully generated using the interactive user interface. A similar procedure has been demonstrated for the creation of the simulation control option file. These files were then used with the models in the new model format which allows the same model equations to be used for different variable specifications. The standard methanol mixer tank problem, with both controllers active, generated the same results as those obtained by Smith (1985). As this was the original aim of this simulation exercise the results were considered to be satisfactory. However, this simulation was taken a step further by the modification of the variable specification, so that the level controller on the tank was effectively removed with the aid of

BTOPOL. The results obtained were consistent with expectations. This demonstrates the great flexibility present in DASPII to allow simulation of systems with different configurations in a simple and effective manner.

Non-linear algebraic equations problems from literature have been used to compare the solution methods present in DASPI, with a new method which offers local, expanded and global convergence, instead of just local convergence. The results illustrate the benefit of having a solution which offers an expanded region of convergence. Although the problems had fewer variables than those likely to be encountered in industrial problems, the problems had unusual characteristics which would be similar to those likely to exist in chemical engineering problems.

CONSOL found all possible solutions to the NLAE test problems without requiring the provision of any starting points. Both real and complex solutions were located, however the time taken to solve for all possible solutions was high. System reduction offers a technique for reducing the time taken to solve the problem by removing linear equations from the system consequently fewer solutions are required to be located.

These examples show that DASPII offers a flexible method for simulating chemical processes, with increased robustness due to the addition of CONLES, which offers an expanded region of convergence for non-linear algebraic equations. The methanol mixer example illustrates the efficiency and the robustness of the interactive front end which removes the need for the user to have detailed prior knowledge of the models available in the model library.

CHAPTER EIGHT

DISCUSSION AND FUTURE WORK

8.0 General Discussion

Dynamic and steady state simulation of chemical processes within an equation oriented environment has been examined in this work. Of the various approaches to chemical process simulation, the equation oriented approach overcomes most of the problems encountered by the other methods regarding design, optimisation, flow pressure modelling and coupling in dynamics. It also offers greater flexibility in the formulation of the model equations and their subsequent solution. Implementation of this method however requires the solution of several problems inherent in this approach. Ogbonda (1987) tackled some of these problems whilst designing DASPI (Fletcher and Ogbonda (1987)). The requirements fulfilled in the design of DASPI are the implementation of an efficient integration method which enables the simultaneous solution of sets of differential algebraic equations and the detection of discontinuities (Ogbonda (1987)). Ogbonda (1987) also developed a methodology for assembling the model equations from modules with the aid of an executive structure which controls the complete package.

Smith (1985) highlighted two main styles of use of simulation packages within the chemical industry. The first is the case where the simulation system is used by process plant based engineers with limited expert support. In this environment the complete package must be user friendly and offer a high degree of robustness. In the second situation simulation specialists provide a simulation service to process engineers within the company. The experts also provide models, in which the process engineer is only required to formulate his problem with these pre-prepared models. In this instance user friendliness is mainly needed in the simulation specification and analysis sections of the system. This is in line with the general view that equation oriented simulators are so user unfriendly that they verge on the hostile. The robustness of equation oriented simulators has also been questioned (Chen and Stadtherr (1985)) because of the need to solve large sets of non-linear equations simultaneously. The work described in this project has been undertaken to overcome the serious limitations described

above by creating an interactive and robust simulation package DASPII. This has been achieved by merging together various different ideas and are highlighted in this discussion. Consequently this work has been carried out to overcome the serious limitations described above, by creating an interactive and robust simulation package DASPII.

The use of a process flowsheet block diagram generator enables block diagrams to be created without the user requiring to have prior knowledge of special input languages, keywords or fixed format topology files as used in other equation oriented simulators. The technique of block diagram generation is more flexible than those implemented in other types of simulators. For instance unique icons are not used for particular models but are used in a generic manner, whereby a general icon is used to represent a set of similar types of models in the library. This does not overwhelm the user with too much information. Also confusion is avoided during the addition of new models, as new models added to the DASPII model library are classified in the appropriate generic model types and new icons need not be added to the flowsheet generator as in other simulation systems. The question and answer session reduces the risk of the user entering incorrect entries into the topology file. The generation of the process topology for DASPII is undertaken in a most efficient and user friendly manner.

It is well known (Chen and Stadtherr (1985)) that the equation oriented approach offers the most flexible technique of simulation. In principle any variable within the equation system may be calculated, unlike the modular methods which calculated with output variables for a unit given the inlet variables. However, with this added flexibility the user has a more difficult task of correctly specifying the problem. The interactive method implemented in DASPII assists the user in correctly defining the problem and executing the simulation.

The variable specification information for each model supplied by the model builder, is retrieved from the database. The information is used to interrogate the user in a question and answer session. As for the topology generation the need for the user to have knowledge of input languages, keywords or file formats is avoided. This interactive procedure also ensures

that the correct number of specifications have been fulfilled. Most importantly the user is prevented from incorrectly defining the problem, so that modelling difficulties such as singular systems do not arise, since the user has the model builders knowledge. Consequently in DASPII the enormous flexibility of the equation oriented approach is available in a truly user friendly environment, hence overcoming one of the major weaknesses prevalent in equation oriented simulators.

The flexibility of DASPII has been enhanced by the application of a model structure which enables almost any variable within the model to be calculated from one set of model equations. These equations can be used for both steady state and dynamic simulation without any modification. This together with the interactive variable specification definition allows the user to make changes to the specification in a simple manner. Unlike other simulators which implement special languages or keywords, no information translation is required, hence the turn around time for flowsheet modification is very low. This has been demonstrated with the methanol mixer tank problem. Single sets of equations in models, results in a compact and concise model library which reduces the time taken in coding and debugging.

Locally convergent NLAE solvers usually offer the most rapid convergence if they are given good starting points otherwise they fail to solve the problem. Solvers offering global convergence do not require good starting points but tend to be much slower.

It is ironic that most equation oriented simulators, which are required to solve very large sets of non-linear algebraic equations simultaneously employ NLAE solvers with only local convergence characteristics. Since the successful solution of NLAEs lies at the heart of steady state simulation and is vital in dynamic simulation where DAEs are integrated to form large sets of NLAEs, it is quite clear why the robustness of equation oriented simulators has until now been questioned. CONLES, the NLAE solution system that has been added to DASPII, offers three methods each having one of the three possible forms of convergence characteristics. CONLES has been compared with the method implemented in DASPI on a range of NLAE .

systems from different starting points. Although the test systems were much smaller than those likely to be encountered in an industrial chemical processes simulation, each of the test problems have difficulties associated with them which are of the type likely to occur in simulation problems. From these tests CONLES was found to converge to the correct solution, when the starting point was further away from the solution than was possible with Broyden's method. In simulators reasonable default starting points are usually provided for classes of variables. This means that a truly global equation solver that is computationally intensive is not vital. The conclusion of this assessment was that CONLES provides an expanded region of convergence that would be suitable for most chemical engineering simulations and it offers robustness and yet is still efficient in the most critical area of the simulator.

The means of providing an initial starting point for dynamic simulation were investigated. In most instances a steady state of the problem can be used as the starting point. Therefore the search for an initialisation technique was concentrated in the steady state area. Although CONLES can be used as the solver it can only locate a single starting point depending on its own initial estimate. Hence CONSOL was tested because of its ability to locate multiple solutions without the need of any initial estimates. However these benefits are available at the expense of considerable computational effort. This can be overcome to some extent by reducing the equation set by hand so that the total degree of the problem is reduced, hence less effort is needed to find the solutions. In practice this would not be a viable operation for a simulation system as it would require too much effort from the user. However equation reduction has been tackled by programs such as REDUCE (Hearn (1983)). This method would enable the simulation system to undertake the reduction and subsequently solve the problem. Since real and complex solutions are found the user is required to discount the infeasible solutions.

In DASPII the strengths of DASPI (Ogbonda (1987)) have been enhanced whilst the main weaknesses have been overcome to produce an equation oriented simulation system that is numerically accurate and efficient and yet is user friendly and robust.

8.1 Recommendations for future work

At present DASPII has been tested on small problems. In order to simulate realistic chemical processes the range of unit operation models need to be extended.

Some industrial chemical processes require process units to be modelled in a 'distributed' manner, which give rise to systems of partial differential equations. At present DASPII is unable to solve such systems and requires the manual conversion of the PDEs to sets of ODEs by using the method of lines. Process units such as detailed pipe models, heat exchangers, tubular reactors and packed tower models must be sometimes modelled as distributed units. DASP II needs to be developed so that the PDEs arising from the distributed system can be converted to ODEs without the user's intervention, thereby ensuring the robustness and interactiveness of package is not lost.

Further work is also required on the topology information generation. Although, PFG enables block diagrams to be represented graphically there are several weaknesses with this version of PFG. The main drawback being the limited space in which diagrams can be drawn, this can be overcome by two possible ways:

a) The first strategy creates the flowsheet on different levels. For instance in the top level of diagrams only the general chemical processes are represented. So that a top level diagram of a refinery flowsheet might show the crude storage, crude stabilisation, light component separation, heavy component separation and product storage as separate blocks on the block diagram. The next level down would show the diagram of one of the blocks in the top level. For the light component separation we would have for example: methane and hydrogen separation, ethane separation, propane separation, butane separation, and pentane separation. All represented as separate blocks. The third level down would be a detailed representation of the blocks in the second level. Therefore for the ethane separation we would have the deethaniser column, partial condenser, feed heat

exchangers, reboiler product pumps, reflux drum, side stream product drum, etc. Hence the flowsheet gets more detailed as the user goes further down the diagram levels.

b) The second approach is the development of a window type system as used in several commercial drawing packages such as Decwrite (Anon (1990)) where the computer screen represents a portion of the actual drawing area. Cursor movement will allow the user to pan the complete drawing area. Therefore a large and complex flowsheet can be drawn on one level starting at one point and then just building the flowsheet by moving the cursor up, down, left and right.

There are drawbacks with both of these techniques. In the first strategy the user is required to generate n diagrams, where n is the number of levels. However there is no limit on the size of diagram that can be generated. Although, the drawing area is increased in the second approach there is still limits on the size of diagram that can be generated, this is dependent on the actual size of the new drawing area.

DASPII enables the process topology to be altered during a simulation so that non functional units are in effect removed from the flowsheet. At present this is carried out by the user defining the latent units and modifying the stream connections with the aid of a conversational procedure. This can be done more interactively by allowing the user to modify a graphical block diagram. Since a graphical block diagram of the flowsheet is available in DASPII this development can be undertaken with minimal effort.

8.2 CONCLUSION

In this project a mechanism has been developed for gathering together recent algorithms and methods from various disciplines to design a robust and user friendly simulation system DASPII, which retains the strengths of the equation oriented approach. DASPII demonstrates that the ideas implemented work, therefore the objectives of this project are fulfilled. Hence a robust and user friendly equation oriented simulator has been created by the implementation of a new NLAE solver, a graphical topology information generator and an interactive front end which uses a commercial database management system. The flexibility of the system is enhanced by the model format implemented.

It is evident from this project that future simulators must possess the numerical accuracy of the equation oriented approach whilst having the usability of the sequential modular approach. A great deal more research effort must therefore be expended on the further developments of the usability and general robustness of the equation oriented simulators.

REFERENCES

REFERENCES

Alcock P. (1985), "Dynamic simulation as an engineering support tool". Process Engineering: September 1985.

Anon. (1967), "The continuous system simulation language (CSSL)". Simulation, Vol. 9, No 6: December 1967.

Anon. (1976), "Advanced continuous simulation language (ACSL)". User Guide/Reference Manual, Mitchell and Ganthcer Ass. Inc., Concord Mass. 1976.

Anon. (1980), "Standards for dynamic simulation language". Digest No 1980/17, Inst of Electrical Eng. London, 1980.

Anon. (1984), "Control of Industrial major accident hazard registries", Stat. Instrument No. 1902, HMSO, London, 1984.

Anon. (1988), "SPEED-UP - user guide". Prosys Tech. Cambridge, 1988.

Anon. (1990), "DecWrite User guide", Digital Equipment Corporation, 1990.

Aylot M.R., Ponton J.W. and Lott D.H. (1985), "Development of a dynamic flowsheeting program". IChemE Symposium Series No 92. pp 55-66, 1985.

Biegler L.T. (1989), "Chemical process simulation". Chemical Eng. Prog. Vol 85. No 10. 1989.

Briggs D.E. (1974), "DISCO - An interactive executive program for dynamic simulation and control of chemical processes". 78th National AIChE Meeting, Salt Lake City.

Broyden G.G. (1969), "A new method for solving non linear simultaneous equations". Comp J Vol 12 (1), 94 1969.

Burchell G. (1989), "Simulators for chemical operator training". Paper presented at the Nonsuch Branch IChemE Meeting, 1989.

Burton P.J. and Mortan W. (1987), "Differential arc length homotopy continuation in equation oriented simulation". CEF 87 XVIII Congress, 1987.

Cameron I.T. (1981), "Numerical solution of differential algebraic systems in process dynamics". PhD Thesis, University of London, 1981.

Carnaham M. and Stewart W.E. (1985), "Sensitivity analysis of initial value problems with mixed ODES and algebraic equations". Comput. Chem. Eng. Vol 9, No 4, pp 359-365.

Carvar M.B. (1978), "Efficient integration over discontinuities in ODE simulations". Math and Comput in simulation xx, pp 190-196, 1978.

Chen H.S. and Stadtherr M.A. (1981), "A modification of Powell's dogleg method for solving systems of nonlinear equations". Comput. Chem. Eng. 5, p 153.

Chen H.S. and Stadtherr M.A. (1985), "A simultaneous modular approach to process flowsheeting and optimization - Part I and II". AICheE J Vol 31, No 11, pp 1843-1881. November, 1985.

Chow S.N., Hallet-Paret J. and Yorke J.A. A homotopy method for locating all zeros of system at polynominals in functional differential equations and approximation at fixed points. Peitgen H.O. and Waltner H.O., edd; Springer-Verlag Lecture Notes in Math 730, New York, 1979. pp 228-237.

Christensen Y.S. and Rudd D.F. (1964) AIChEJ, Vol. 15. No. 94, 1964.

Cook J. (1989), "An accident waiting to happen". Unwin Hyman, 1989.

Cosnard N.Y. "A comparison of four methods for solving systems of nonlinear equations". Tech. Rep 75-248, Department of Computer Science, Cornell University, Ithaca, New York (1975).

Curtis B., Sood M.K. and Reklaitis J.G.V. "Computer aided flowsheet drawing I: Equipment layout". Computers and Chem. Engng 5 (4) (1981).

Dennis J.E. and Schnabel R. "Numerical methods for unconstrained optimization and non linear equations". Prentice-Hall, Englewood Cliffs, NJ, 1983.

Duff 15 (1977), "MA28 - A set of fortran subroutines for sparse unsymmetric linear equations". AERE Report R.8730, HMSO, London 1977.

Evans L.B. (1980), "Advances in process flowsheeting systems". in Mah R.S.H. and Seider W.D. (Eds), Foundations of computer-aided chemical process design, Proceeding of an International Conference, Henniker, New Hampshire, July 1980.

Ferraris G.B. and Tronconi E. (1986), "BUNLSI - A Fortran program for solution of systems of non linear algebraic equations". Comput Chem Eng Vol 10, No 2, pp 129-141.

Franks R.G.E. (1972), "Modelling and simulation in chemical engineering". John Wiley Interscience, New York, 1972.

Franks R.G.E. (1982), "DYFLO update: DYFLO2". 1982 Summer Computer Simulation Conference Denver, Colorado. July 19-21, 1982, pp 507-513.

Gear C.W. (1971c), "Simultaneous numerical solution of differential algebraic equations". IEEE Trans on Circuit Theory, CT - 18. No 1, pp 89-95, 1971.

Georg K. "Numerical integration at the Davidenko equation, in numerical solution of nonlinear equations". Allgover E., Glashoff K., Peitgen H.O. eds., Springer-Verlag Lecture Notes in Math. 878, New York, 1981.

Gerczynski E.W., Hutchinson H.P. and Wajih A.R.H. (1979), "Development of a modularly organized equation - oriented process simulator". Comput Chem Eng, Vol 3, pp 353-356.

Hearn A.C., "Reduce user's manual". Rand Publication CP78 (4/83), The Rand Corporation, Santa Monica, California, 1983.

Hiebert M.C., "An evaluation of mathematical software that solves systems of nonlinear equations". ACM Trans. Math. Softw., 8, 5-20 (1982).

Hindmarth A.C. and Byrne G.D. (1975), UCID - 30112, Lawrence Livermore Lab. Rept., California Univ.

Hlavacek V. (1977), "Analysis of a complex plant - steady state and transient behaviour". Comput Chem Eng. Vol 1, pp 75-100, 1977.

Holland C.D. and Liapis A.I. (1985), "Computer methods for solving dynamic separation problems". McGraw - Hill, New York.

Hutchinson H.P., Jackson D.J. and Morton W. (1986), "The development of an equation-oriented flowsheet simulation and optimization package I: The QUASILIN PROGRAM", Comput Chem Eng Vol 10, No 1 pp 19-29, 1986.

Jirapongphan S., "Simultaneous modular convergence concept in process flowsheet optimization". PhD Thesis. MIT, (1980).

Joglekar G.S. and Rekbaitis G.V. (1984), "BOSS - A simulator for batch and semi-continuous processes". Comput Chem Eng. Vol 8, No 6, pp 315-327, 1984.

Kletz. (1988), "Learning from accidents in Industry". Butterworth, 1988.

Kohlert W., Siegismund B., Hartmass K. and Vrba J. (1985), "Equation oriented simulation of technological systems". Coll. Czech Chem. Commun. Vol 50, 1985.

Korn D.A. and Wait J.V. (1978), "Digital continuous systems simulation". Prentice-Hall Inc, Inglewood Cliffs, NJ USA, 1978.

Kovach III J.W. and Seider W.D. (1987), "Hetrogeneous azeotropic distillation. Experimental and simulation results". AIChE J Vol 33, No 8, 1987.

Kubicek H. (1976), "Algorithm 502 - Dependence of solution of nonlinear systems on a parameter". ACM Trans Math Software, Vol 2, pp 98-107, 1976.

Kuru S. (1981), "Dynamic simulation with an equation-based flowsheeting system". PhD Thesis, Carnegie Mellon University, Pittsburgh.

Liang W.C. (1985), "An interactive dynamic flowsheet simulation program". PhD Thesis, Lehigh University.

Liu Y.C. and Brosilow C.B. (1987), "Simulation of large scale dynamic systems - I modular integration methods". Comput Chem Eng, Vol II, No 3. pp 241-253, 1987.

Locke M.H. (1981), "A CAD tool which accommodates an evolutionary strategy in engineering design calculations". PhD Thesis, Carnegie-Mellon University, Pittsburgh.

Mitchell E.E.L. (1978), "Advanced continuous simulation language (ACSL)". In numerical methods for differential equations and simulation, Bennett A.W. and Vichnevetsky (eds), IMACS, North Holland Publishing Company, pp 139-146, 1978.

Mix T., Dweck J. and Weinberg M. (1977), "The potential energy conversion in distillation". Paper present at 70th Annual AIChE Meeting, New York, 1977.

More J.J. and Cosnard M.H., "Brenth, a Fortran subroutine for the numerical solution of systems of nonlinear equations". ACM Trans. Math. Softw., 6, 240-251 (1980).

Morgan A.P. (1987), "Solving polynomial systems using continuation for scientific and engineering problems". Prentice-Hall, 1987.

Mould R.F. (1988), "Chernobyl the real story", Pergamon Press 1988.

Naess L. and Loeken R.A., "Graphical input to a process simulator". PSE 85 Cambridge in AIChE Symp. Ser. 92, 542, (1985).

Ogbonda J.E. (1987), "Dynamic simulation of chemical plant". PhD Thesis, University of Aston, Birmingham.

Pagani G., Arminio Mon forte A. and De Mitri A. (1989), "Improving an equation - oriented package". Comput Chem Eng, Vol 13, No 8 pp 931-945.

Paloshi J.R. (1982), "The numerical solution of nonlinear equations representing chemical processes". PhD thesis, University of London, 1982.

Pantelides C.C., "The consistent initialisation of differential-algebraic systems". Submitted for publication (1986).

Pantelides C.C., Gritsis D., Morison K.R. and Sargent R.W.H. (1987). "The mathematical modelling of transient systems using differential-algebraic equations". Proceedings of the XVIII Congress on the use of computers in Chemical Engineering. CEF '87, Giardini Naxes, Sicily, Italy. April 26-30, 1987.

Pantelides C.C. (1988), "SPEEDUP - Recent advances in process simulation". Comput Chem Eng, Vol 12, No 7, pp 745-755, 1988.

Pathe D.C. (1986), "Simulator a key to successful plant start-up". OGJ Report 7th April, 1986.

Patterson G.K. and Rozsa R.B. (1980), "DYNSYL - A general purpose dynamic simulator for chemical process". Comput. Chem Eng Vol 4, pp 1-20, 1980.

Perkins J.D. and Sargent R.W.H. (1982), "SPEED-UP - a computer program for steady state and dynamic simulation and design of chemical processes". AIChE Symp Ser. No 214, Vol 78 AIChE, NY.

Petzold L. (1983), "A description of DASSL: A differential-algebraic system solver". Scientific Computing with R Stephenman *et al* (eds), IMACS/North Holland Publ Co, 1983, pp 65-68.

Pierucci S.J., Ranzi E.M.and Biardi G.E. "Solution of recycle problems in a sequential modular approach". AIChE Journal, 28, 820, (1982).

Ponton J.W. (1983), Comput. Chem. Eng., Vol 7, No 1 pp 13-17

Ponton J.W. and Vasek V. (1986), "A two level approach to chemical plant and process simulation". Comput Chem Eng Vol 10, No 3, pp 277-286, 1986.

Powell M.J.D. (1970), "A hybrid method for nonlinear equations". In "Numerical methods for nonlinear algebraic equations". Rabinowitz P., ed, Gordon & Breach, London, 1970.

Preece P.E., Bader A.S.A.R., Davila Pernia J.L., Evans J.A.C. and Giller M.K. (1987), "The development of a 2-D graphical process flowsheet generator (PFG) and a piping and instrumentation generator (PIG)". Comput Chem Eng Vol 11, No 3, pp 279-289.

Pritsker A.A.B. and Hurst N.R. (1973), "GASP IV - a combined continuous - discrete Fortran based simulation language". Simulation, Sept pp 65-71, 1973.

Prokopakis C.J. and Seider W.D. (1983), "Dynamic simulation of azeotropic distillation towers". AIChE Vol 21, No 6, pp 1017-1029, 1983.

Reklaitis G.V., Curtis B. and Sood M.K. "Computer aided flowsheet drawing - I equipment layout". Comput chem Engng 5, 277, (1981).

Rheinboldt W.C., "On the computation of critical boundaries on equilibrium surfaces". SIAM J Numer. Anal. 19 (1982), pp 653-669.

Rosen E.M. (1980), "Steady state chemical process simulation - A state-of-the-art review". In computer applications to chemical engineering, R.G. Squires and G.V. Reklaitis (Eds), ACS Symp Ser, Vol 124, (3), 1980.

Sargent R.W.H. and Westerberg S.E. (1964), "SPEED-UP in chemical engineering design". Trans Inst Chem Engrs, 42, TI90, 1964.

Sargent R.W.H. (1981), "A review of methods for solving nonlinear algebraic equations". In R.S.H. Mah and W.D. Seider (eds). Foundations of computer-aided chemical process design, Vol 1, pp 27-76, Engineering Foundation NY.

Sasnow S. (1989), 'Well heads on the seabed". New Scientist 18 March, 1989.

Schubert L.K. (1970), "Modification of a Quasi-Newton method for non linear equations with a sparse jacobian". Math Comp Vol 24, pp 27-31.

Seader J.D. (1985), "Computer modelling of chemical processes". AIChE Monograph Ser, Vol 81, No 15, 1985.

Shacham M., Macchietto S., Stutzman L.F. and Babcock P. (1982). "Equation-oriented approach to process flowsheeting". Comput. Chem Eng, Vol 6, No 2, pp 79-95, 1982.

Shacham M., "Comparing software for the solution of nonlinear algebraic equations arising in chemical engineering". Comput. Chem. Eng., 9, 103-112 (1985).

Shacham M. (1986), "Numerical solution of constrained non linear algebraic equations". Int J Num Meth in Eng. Vol 23, pp 1455-1481, 1986.

Singh S.P. and Cranhan B., "An interactive process flowsheeting and simulation system based on relational data structures". Comput Chem Eng Vol 5, No 4, 1981.

Slaver E.R. (1986), "Dynamic simulation in 1991: An Exxon viewpoint". Proceedings of the 1986 Summer Computer Simulation Conference, pp 292-296.

Smith G.J. (1985), "Dynamic simulation of chemical processes". PhD Thesis, University of Cambridge 1985.

Speckhart F.H. and Green W.L. (1976), " A guide to using CSMP - the continuous system modelling program". Prentice Hall Inc., Englewood Cliffs NJ 1976.

Stadtherr M.A. and Hilton M. (1982b), "Development of a new equation-based process flowsheeting systems: numerical studies". In selected topics on computer-aided process design and analysis (eds R.S. Mah & G.V. Reklaitis) AIChe Symp Ser (1982).

Stadtherr M.A. and Wood E.S. (1984), Comput Chem Eng, Vol 8, No 9 1984.

Stover W. (1985), "The chemical industry after Bhopal"., Proceedings of a symposium, IBC Tech. Services, London, 1985.

Thambynayagom R.K.M., Wood R.K. and Winter P. (1981), "DPS - An engineers tool for dynamic process analysis". The Chemical Engineer, pp 58-65 1981.

Trevino-Lozano R.A., Evans L.B., Britt H.I. and Boston J.F. (1985), "Simultaneous modular process simulation and optimization". Symp. Ser. 92, pp 25-36.

Watson L.T., Billups S.C. and Morgan A.P., HOMPACK: "A suite of codes for globally convergent homotopy algorithms". Research Publication GMR - 5344, GM Research Laboratories, Wahgu, H I 48090, June, 1986.

Westerberg A.W. and Director S.W. (1978), "A modified least squares algorithm for solving sparse n*m sets of non linear equations". Comput Chem Eng, Vol 2, 1978.

Westerberg A.W., Hutchinson H.P., Motard R.L. and Winter P. (1979), "Process Flowsheeting". Cambridge University Press, Cambridge 1979.

Westerberg A.W. (1981), "Computer-aided design tools in chemical engineering process design". Proceedings of the IEEE, Vol 69, No 10, pp 1232-1239.

Westerberg A.W. and Bengamin D.R. (1985), "Thoughts on a future equation-oriented flowsheeting system". Comput Chem Eng, Vol 9, No 5, pp 517-526.

Wood R.K., Thambynayagom R.K.M., Noble R.G. and Sebastion D.J. (1984), "DPS - A digital simulation language for the process industries". Simulation pp 221-233.

Woodman M.R. (1989), "Simulation of three phase azeotropic ditillation columns", PhD Thesis, University of Cambridge, Cambridge.

Zangwill C.B. and Garcia C.B., "Pathways to solutions, fixed points and equilibria". Prentice-Hall, Englewood Cliffs, NJ (1981).

APPENDICES

LIST OF CONTENTS FOR THE APPENDICES

Page

A.1 Comparison of CONLES and Broyden's method

A.1 Problem 1 Solution for reaction rate equations (Shacham (1986))

The following parameters values were used in this problem:

$k_1 = 31.24$

$kr_1 = 2.062$

$k_2 = 0.273$

$kr_2 = 0.02$

$k_3 = 303.03$

1. CONLES method starting point 1 - successful solution

```
          EQUATION SOLVER AND MATRIX INFORMATION
          **************************************

Maximum number of iterations (KMAX):                    100
Equation solver method flag (MFALG):                     4
Jacobian availability flag (LJAC):                       F
Initial Jacobian flag (KJAC):                            0
-------------------------------------------------------------


          STATISTICS OF THE SIMULATION
          ============================

NO OF FUNCTION EVALUATIONS :        26
NO OF JACOBIAN EVALUATIONS :         3
NO OF ITERATIONS TAKEN     :         3
-------------------------------------------------------------


          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************


Total number of variables:                              6
Total number of equations:                              6
-------------------------------------------------------------

S/N   VARIABLE 1    VARIABLE 2    VARIABLE 3    VARIABLE 4
       ----          ----          ----          ----
---  ----------    -----------  -----------  -----------
```

```
1    .99000D+00     .50000D-01     .50000D-01     .99000D+00
2    .97424D+00     .98283D+00     .51513D-01     .93567D+00

S/N    VARIABLE 5     VARIABLE 6
----   ----------   ------------
1       .50000D-01     .00000D+00
2       .90840D-04     .64238D-01
==========================================================
```

2. Broyden's method starting point 1 - successful solution

```
          EQUATION SOLVER AND MATRIX INFORMATION
          **************************************

Maximum number of iterations (KMAX):                    100
Equation solver method flag (MFALG):                      2
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
----------------------------------------------------------


          STATISTICS OF THE SIMULATION
          ============================

NO OF FUNCTION EVALUATIONS :          11
NO OF JACOBIAN EVALUATIONS :           1
NO OF ITERATIONS TAKEN     :           2
----------------------------------------------------------


          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************


Total number of variables:                              6
Total number of equations:                              6
----------------------------------------------------------


S/N    VARIABLE 1     VARIABLE 2     VARIABLE 3     VARIABLE 4
         ----           ----           ----           ----
---    ------------   ------------   ------------   ------------
1     .99000D+00     .50000D-01     .50000D-01     .99000D+00
2     .97415D+00     .98277D+00     .51700D-01     .93567D+00
```

S/N	VARIABLE 5	VARIABLE 6	
1	.50000D-01	.00000D+00	
2	.78581D-04	.64250D-01	

3. CONLES method starting point 2 - successful solution

EQUATION SOLVER AND MATRIX INFORMATION

Maximum number of iterations (KMAX):	500
Equation solver method flag (MFALG):	4
Jacobian availability flag (LJAC):	F
Initial Jacobian flag (KJAC):	0

STATISTICS OF THE SIMULATION

NO OF FUNCTION EVALUATIONS :	676
NO OF JACOBIAN EVALUATIONS :	84
NO OF ITERATIONS TAKEN :	85

* R E S U L T S O F S I M U L A T I O N *

Total number of variables:	6
Total number of equations:	6

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.50000D+00	.50000D+00	.50000D+00	.50000D+00
2	.97424D+00	.98283D+00	.51513D-01	.93567D+00

S/N	VARIABLE 5	VARIABLE 6
1	.50000D+00	.50000D+00
2	.90840D-04	.64238D-01

4. Broyden's method starting point 2 - Failed run

EQUATION SOLVER AND MATRIX INFORMATION

Maximum number of iterations (KMAX):	500
Equation solver method flag (MFALG):	2
Jacobian availability flag (LJAC):	F
Initial Jacobian flag (KJAC):	0

STATISTICS OF THE SIMULATION

NO OF FUNCTION EVALUATIONS :	98
NO OF JACOBIAN EVALUATIONS :	8
NO OF ITERATIONS TAKEN :	35

* R E S U L T S O F S I M U L A T I O N *

Total number of variables:	6
Total number of equations:	6

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.50000D+00	.50000D+00	.50000D+00	.50000D+00

```
2        failed          failed          failed          failed


S/N    VARIABLE 5      VARIABLE 6
         ----            ----
---   ------------    ------------
1      .50000D+00      .50000D+00
2        failed          failed
==========================================================
```

A.1 Combustion of propane in air (Hiebert(1982))

1. Broyden's method starting point 1 - successful solution

```
        EQUATION SOLVER AND MATRIX INFORMATION
              ************************************

Maximum number of iterations (KMAX):                    100
Equation solver method flag (MFALG):                      2
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
-------------------------------------------------------------

          STATISTICS OF THE SIMULATION
                ============================

NO OF FUNCTION EVALUATIONS :        49
NO OF JACOBIAN EVALUATIONS :         3
NO OF ITERATIONS TAKEN     :        12
-------------------------------------------------------------

          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************

Total number of variables:                              10
Total number of equations:                              10
-------------------------------------------------------------

S/N  VARIABLE 1    VARIABLE 2    VARIABLE 3    VARIABLE 4
        ----          ----          ----          ----
--- ------------  ------------  ------------  ------------
1    .10000D+01    .10000D+01    .10000D+02    .10000D+01
2    .27561D+01    .38823D+01    .19968D+02    .18637D+00
3    .28812D+01    .39511D+01    .19984D+02    .11882D+00

S/N  VARIABLE 5    VARIABLE 6    VARIABLE 7    VARIABLE 8
        ----          ----          ----          ----
--- ------------  ------------  ------------  ------------
1    .10000D+01    .10000D+01    .00000D+00    .00000D+00
2    .47994D-01    .82038D-02    .99144D-01   -.16540D+00
3    .31408D-01    .46843D-02    .30241D-01    .14617D-01

S/N VARIABLE 9    VARIABLE 10
        ----          ----
--- ------------  ------------
1    .00000D+00    .00000D+00
2   -.31127D+00   -.41893D+00
3    .11430D+00    .42667D-02
=============================================================
```

3. Broyden's method starting point 2 - failed after 500 iterations

```
          EQUATION SOLVER AND MATRIX INFORMATION
                  ************************************

Maximum number of iterations (KMAX):                    500
Equation solver method flag (MFALG):                      2
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
-------------------------------------------------------------
```

4. CONLES method starting point 2 - successful solution

```
          EQUATION SOLVER AND MATRIX INFORMATION
                  ************************************

Maximum number of iterations (KMAX):                    500
Equation solver method flag (MFALG):                      4
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
-------------------------------------------------------------


          STATISTICS OF THE SIMULATION
                  ============================

NO OF FUNCTION EVALUATIONS :        656
NO OF JACOBIAN EVALUATIONS :         54
NO OF ITERATIONS TAKEN     :         58
-------------------------------------------------------------


          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************


Total number of variables:                          10
Total number of equations:                          10
-------------------------------------------------------------


S/N  VARIABLE 1    VARIABLE 2    VARIABLE 3    VARIABLE 4
       ----          ----          ----          ----
---  ------------  ------------  ------------  ------------
1    .10000D+02    .10000D+02    .10000D+02    .10000D+02
2    .86818D+01    .76718D+01    .10497D+02    .96182D+01
3    .75508D+01    .54145D+01    .10984D+02    .92192D+01
4    .28801D+01    .39507D+01    .19984D+02    .11989D+00
```

S/N	VARIABLE 5	VARIABLE 6	VARIABLE 7	VARIABLE 8
1	.10000D+02	.10000D+02	.10000D+02	.10000D+02
2	.10006D+02	.10269D+02	.91755D+01	.93569D+01
3	.99328D+01	.10992D+02	.84338D+01	.87854D+01
4	.31741D-01	.46846D-02	.30484D-01	.16088D-01

S/N	VARIABLE 9	VARIABLE 10
1	.10000D+02	.10000D+02
2	.93570D+01	.97285D+01
3	.87856D+01	.95300D+01
4	.12056D+00	.10438D-02

A.1 Problem 3 Chemical equilibrium problem (Hiebert (1986))

1. Broyden's method starting point 1 - successful solution

```
                EQUATION SOLVER AND MATRIX INFORMATION
                **************************************

Maximum number of iterations (KMAX):                        100
Equation solver method flag (MFALG):                          2
Jacobian availability flag (LJAC):                            F
Initial Jacobian flag (KJAC):                                 0
-------------------------------------------------------------


          STATISTICS OF THE SIMULATION
                ============================

NO OF FUNCTION EVALUATIONS :          49
NO OF JACOBIAN EVALUATIONS :           4
NO OF ITERATIONS TAKEN     :          16
-------------------------------------------------------------


          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************


Total number of variables:                                    6
Total number of equations:                                    6
-------------------------------------------------------------


S/N   VARIABLE 1    VARIABLE 2    VARIABLE 3    VARIABLE 4
        ----          ----          ----          ----
---  ------------  ------------  ------------  ------------
1     .00000D+00    .00000D+00    .00000D+00    .00000D+00
2     .82650D-04    .82650D-03    .89937D-04    .90850D-04


S/N   VARIABLE 5    VARIABLE 6
        ----          ----
---   ------------  ------------
1  .00000D+00    .00000D+00    .00000D+00
2  .00000D+00    .55000D+02   -.36420D-06
=============================================================
```

2. CONLES method starting point 1 - successful solution

EQUATION SOLVER AND MATRIX INFORMATION
****

Maximum number of iterations (KMAX): 500
Equation solver method flag (MFALG): 4
Jacobian availability flag (LJAC): F
Initial Jacobian flag (KJAC): 0

STATISTICS OF THE SIMULATION
==========

NO OF FUNCTION EVALUATIONS : 563
NO OF JACOBIAN EVALUATIONS : 68
NO OF ITERATIONS TAKEN : 76

* R E S U L T S O F S I M U L A T I O N *
****

Total number of variables: 6
Total number of equations: 6

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.00000D+00	.00000D+00	.00000D+00	.00000D+00
2	.72146D-02	.14019D-01	.41610D-01	.70767D-02
3	.71410D-02	.13864D-01	.41454D-01	.70323D-02
4	.42367D-02	.78393D-02	.33987D-01	.51463D-02
5	.82621D-04	.82621D-03	.90622D-04	.91171D-04

S/N	VARIABLE 5	VARIABLE 6
1	.00000D+00	.00000D+00
2	.55537D-01	.62500D-01
3	.60475D+00	.62108D-01
4	.22355D+02	.45453D-01
5	.55000D+02	.11035D-09

==========

3. Broyden's method starting point 2 - successful solution

EQUATION SOLVER AND MATRIX INFORMATION
****

```
Maximum number of iterations (KMAX):                    500
Equation solver method flag (MFALG):                      4
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
-------------------------------------------------------------

          STATISTICS OF THE SIMULATION
                ============================

NO OF FUNCTION EVALUATIONS :        571
NO OF JACOBIAN EVALUATIONS :         69
NO OF ITERATIONS TAKEN     :         77
-------------------------------------------------------------

          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************

Total number of variables:                              6
Total number of equations:                              6
-------------------------------------------------------------
```

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.00000D+00	.00000D+00	.00000D+00	.00000D+00
2	.72146D-02	.14019D-01	.41610D-01	.70767D-02
3	.71410D-02	.13864D-01	.41454D-01	.70323D-02
4	.49602D-02	.93192D-02	.36137D-01	.56467D-02
5	.82621D-04	.82621D-03	.90622D-04	.91171D-04

S/N	VARIABLE 5	VARIABLE 6
1	.00000D+00	.00000D+00
2	.55537D-01	.62500D-01
3	.60475D+00	.62108D-01
4	.16917D+02	.49872D-01
5	.55000D+02	.11035D-09

```
=============================================================
```

4. CONLES method starting point 2 - successful solution

```
          EQUATION SOLVER AND MATRIX INFORMATION
                *************************************

Maximum number of iterations (KMAX):                    100
Equation solver method flag (MFALG):                      2
Jacobian availability flag (LJAC):                        F
```

```
Initial Jacobian flag (KJAC):                                    0
-----------------------------------------------------------------

          STATISTICS OF THE SIMULATION
                ============================

NO OF FUNCTION EVALUATIONS :          61
NO OF JACOBIAN EVALUATIONS :           5
NO OF ITERATIONS TAKEN     :          20
-----------------------------------------------------------------

          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************

Total number of variables:                                       6
Total number of equations:                                       6
-----------------------------------------------------------------
```

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.00000D+00	.00000D+00	.00000D+00	.00000D+00
2	.82622D-04	.82622D-03	.90651D-04	.91154D-04
3	.82621D-04	.82621D-03	.90622D-04	.91171D-04

S/N	VARIABLE 5	VARIABLE 6
1	.00000D+00	.00000D+00
2	.55000D+02	.11960D-06
3	.55000D+02	.11037D-09

A.1 Problem 4 Eight variable problem (Ferraris and Tronconi (1986))

1. Broyden's method starting point 1 - solution not reached after 500 iterations

```
          EQUATION SOLVER AND MATRIX INFORMATION
                *************************************

Maximum number of iterations (KMAX):                    500
Equation solver method flag (MFALG):                      2
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
-------------------------------------------------------------


          STATISTICS OF THE SIMULATION
                ============================

NO OF FUNCTION EVALUATIONS :       1511
NO OF JACOBIAN EVALUATIONS :        101
NO OF ITERATIONS TAKEN     :        500
-------------------------------------------------------------


          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************


Total number of variables:                              8
Total number of equations:                              8
-------------------------------------------------------------
```

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.10000D+01	.20000D+02	.22000D+01	.10000D+03
2	.10000D+01	.18718D+02	.20287D+01	.27984D+03
3	.10000D+01	.18718D+02	.20287D+01	.27984D+03
4	.10000D+01	.18718D+02	.20287D+01	.27984D+03
5	.10000D+01	.18718D+02	.20287D+01	.27984D+03
6	.10000D+01	.18718D+02	.20287D+01	.27984D+03
7	.10000D+01	.18718D+02	.20287D+01	.27984D+03
8	.10000D+01	.18718D+02	.20287D+01	.27984D+03
9	.10000D+01	.18718D+02	.20287D+01	.27984D+03
10	.10000D+01	.18718D+02	.20287D+01	.27984D+03
11	.10000D+01	.18718D+02	.20287D+01	.27984D+03
12	.10000D+01	.18718D+02	.20287D+01	.27984D+03
13	.10000D+01	.18718D+02	.20287D+01	.27984D+03
14	.10000D+01	.18718D+02	.20287D+01	.27984D+03
15	.10000D+01	.18718D+02	.20287D+01	.27984D+03
16	.10000D+01	.18718D+02	.20287D+01	.27984D+03
17	.10000D+01	.18718D+02	.20287D+01	.27984D+03
18	.10000D+01	.18718D+02	.20287D+01	.27984D+03
19	.10000D+01	.18718D+02	.20287D+01	.27984D+03
20	.10000D+01	.18718D+02	.20287D+01	.27984D+03
21	.10000D+01	.18718D+02	.20287D+01	.27984D+03

22	.10000D+01	.18718D+02	.20287D+01	.27984D+03
23	.10000D+01	.18718D+02	.20287D+01	.27984D+03
24	.10000D+01	.18718D+02	.20287D+01	.27984D+03
25	.10000D+01	.18718D+02	.20287D+01	.27984D+03
26	.10000D+01	.18718D+02	.20287D+01	.27984D+03

S/N	VARIABLE 5	VARIABLE 6	VARIABLE 7	VARIABLE 8
1	.20000D+01	.80000D+01	-.60000D+02	.15000D+02
2	.68766D+01	.97616D+03	.48499D+07	.98606D+03
3	.68766D+01	.97616D+03	.68470D+07	.98606D+03
4	.68766D+01	.97616D+03	.88329D+07	.98606D+03
5	.68766D+01	.97616D+03	.10812D+08	.98606D+03
6	.68766D+01	.97616D+03	.12788D+08	.98606D+03
7	.68766D+01	.97616D+03	.14760D+08	.98606D+03
8	.68766D+01	.97616D+03	.16730D+08	.98606D+03
9	.68766D+01	.97616D+03	.18698D+08	.98606D+03
10	.68766D+01	.97616D+03	.20665D+08	.98606D+03
11	.68766D+01	.97616D+03	.22630D+08	.98606D+03
12	.68766D+01	.97616D+03	.24595D+08	.98606D+03
13	.68766D+01	.97616D+03	.26559D+08	.98606D+03
14	.68766D+01	.97616D+03	.28522D+08	.98606D+03
15	.68766D+01	.97616D+03	.30485D+08	.98606D+03
16	.68766D+01	.97616D+03	.32447D+08	.98606D+03
17	.68766D+01	.97616D+03	.34409D+08	.98606D+03
18	.68766D+01	.97616D+03	.36370D+08	.98606D+03
19	.68766D+01	.97616D+03	.38331D+08	.98606D+03
20	.68766D+01	.97616D+03	.40292D+08	.98606D+03
21	.68766D+01	.97616D+03	.42252D+08	.98606D+03
22	.68766D+01	.97616D+03	.44212D+08	.98606D+03
23	.68766D+01	.97616D+03	.46172D+08	.98606D+03
24	.68766D+01	.97616D+03	.48132D+08	.98606D+03
25	.68766D+01	.97616D+03	.50091D+08	.98606D+03
26	.68766D+01	.97616D+03	.52050D+08	.98606D+03

2. CONLES method starting point 1 - successful solution

EQUATION SOLVER AND MATRIX INFORMATION

Maximum number of iterations (KMAX):	500
Equation solver method flag (MFALG):	4
Jacobian availability flag (LJAC):	F
Initial Jacobian flag (KJAC):	0

STATISTICS OF THE SIMULATION

NO OF FUNCTION EVALUATIONS : 2488

NO OF JACOBIAN EVALUATIONS : 241
NO OF ITERATIONS TAKEN : 267

---

* R E S U L T S O F S I M U L A T I O N *

****

Total number of variables: 8
Total number of equations: 8

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.10000D+01	.20000D+02	.22000D+01	.10000D+03
2	.79903D+00	.18935D+02	.21090D+01	.10034D+03
3	.66182D+00	.18259D+02	.20215D+01	.10058D+03
4	.56020D+00	.17753D+02	.19396D+01	.10078D+03
5	.48071D+00	.17335D+02	.18614D+01	.10094D+03
6	.41804D+00	.16974D+02	.17873D+01	.10110D+03
7	.36953D+00	.16663D+02	.17184D+01	.10123D+03
8	.33619D+00	.16423D+02	.16618D+01	.10134D+03
9	.10009D+01	.18478D+02	.21294D+01	.26265D+03
10	.10033D+01	.18473D+02	.21297D+01	.26265D+03
11	.10061D+01	.18467D+02	.21302D+01	.26265D+03
12	.10092D+01	.18462D+02	.21310D+01	.26265D+03
13	.10118D+01	.18457D+02	.21317D+01	.26266D+03
14	.10146D+01	.18452D+02	.21326D+01	.26266D+03
15	.10174D+01	.18447D+02	.21335D+01	.26266D+03
16	.10203D+01	.18442D+02	.21345D+01	.26266D+03
17	.10232D+01	.18437D+02	.21355D+01	.26266D+03
18	.10163D+01	.18521D+02	.21035D+01	.26780D+03
19	.10114D+01	.18580D+02	.20811D+01	.27141D+03
20	.10080D+01	.18622D+02	.20654D+01	.27393D+03
21	.10056D+01	.18651D+02	.20544D+01	.27570D+03
22	.10039D+01	.18671D+02	.20467D+01	.27694D+03
23	.10027D+01	.18685D+02	.20413D+01	.27781D+03
24	.10019D+01	.18695D+02	.20375D+01	.27842D+03
25	.10011D+01	.18704D+02	.20339D+01	.27899D+03
26	.10000D+01	.18718D+02	.20287D+01	.27984D+03

S/N	VARIABLE 5	VARIABLE 6	VARIABLE 7	VARIABLE 8
1	.20000D+01	.80000D+01	-.60000D+02	.15000D+02
2	.21343D+01	.88590D+01	-.60025D+02	.14904D+02
3	.21932D+01	.93444D+01	-.60049D+02	.14855D+02
4	.21917D+01	.96698D+01	-.60074D+02	.14824D+02
5	.21536D+01	.99236D+01	-.60098D+02	.14799D+02
6	.21068D+01	.10139D+02	-.60122D+02	.14776D+02
7	.20689D+01	.10322D+02	-.60147D+02	.14756D+02

8	.20440D+01	.10461D+02	-.60168D+02	.14740D+02
9	.34529D+01	.22113D+02	-.35966D+03	.28710D+02
10	.34100D+01	.22126D+02	-.35967D+03	.28710D+02
11	.33693D+01	.22145D+02	-.35967D+03	.28709D+02
12	.33306D+01	.22169D+02	-.35968D+03	.28709D+02
13	.33010D+01	.22190D+02	-.35968D+03	.28708D+02
14	.32728D+01	.22211D+02	-.35968D+03	.28707D+02
15	.32458D+01	.22233D+02	-.35969D+03	.28707D+02
16	.32202D+01	.22255D+02	-.35969D+03	.28706D+02
17	.31959D+01	.22277D+02	-.35970D+03	.28705D+02
18	.42897D+01	.73489D+02	-.22793D+04	.80950D+02
19	.50605D+01	.16011D+03	-.63481D+04	.16830D+03
20	.56027D+01	.27492D+03	-.12637D+05	.28363D+03
21	.59836D+01	.40155D+03	-.20336D+05	.41061D+03
22	.62508D+01	.52378D+03	-.28320D+05	.53310D+03
23	.64382D+01	.63110D+03	-.35683D+05	.64059D+03
24	.65695D+01	.71851D+03	-.41879D+05	.72813D+03
25	.66923D+01	.81180D+03	-.48681D+05	.82153D+03
26	.68766D+01	.97616D+03	-.61080D+05	.98606D+03

A.1 Problem 5 Eight variable problem (Carnaham et al (1969))

1. CONLES method starting point 1 - successful solution

```
          EQUATION SOLVER AND MATRIX INFORMATION
                ************************************

Maximum number of iterations (KMAX):                    100
Equation solver method flag (MFALG):                      4
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
-------------------------------------------------------------


          STATISTICS OF THE SIMULATION
                ============================

NO OF FUNCTION EVALUATIONS :          62
NO OF JACOBIAN EVALUATIONS :           6
NO OF ITERATIONS TAKEN     :           6
-------------------------------------------------------------


          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************


Total number of variables:                              8
Total number of equations:                              8
-------------------------------------------------------------



S/N  VARIABLE 1    VARIABLE 2    VARIABLE 3    VARIABLE 4
       ----          ----          ----          ----
---  ------------  ------------  ------------  ------------
1    .50000D+00    .00000D+00    .00000D+00    .50000D+00
2    .32287D+00    .92235D-02    .46017D-01    .61817D+00

S/N  VARIABLE 5    VARIABLE 6    VARIABLE 7
       ----          ----          ----
---  ------------  ------------  ------------
1    .00000D+00    .50000D+00    .20000D+01
2    .37169D-02    .57672D+00    .29779D+01
=============================================================
```

2. Broyden's method starting point 1 - successful solution

```
          EQUATION SOLVER AND MATRIX INFORMATION
                  **********************************

Maximum number of iterations (KMAX):                        100
Equation solver method flag (MFALG):                          2
Jacobian availability flag (LJAC):                            F
Initial Jacobian flag (KJAC):                                 0
-------------------------------------------------------------

          STATISTICS OF THE SIMULATION
                  ============================

NO OF FUNCTION EVALUATIONS :          26
NO OF JACOBIAN EVALUATIONS :           2
NO OF ITERATIONS TAKEN     :           5
-------------------------------------------------------------

          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************

Total number of variables:                                    8
Total number of equations:                                    8
-------------------------------------------------------------


S/N VARIABLE 1    VARIABLE 2    VARIABLE 3    VARIABLE 4
        ----          ----          ----          ----
--- ------------  ------------  ------------  ------------
1    .50000D+00    .00000D+00    .00000D+00    .50000D+00
2    .32287D+00    .92235D-02    .46017D-01    .61817D+00

S/N VARIABLE 5    VARIABLE 6    VARIABLE 7
        ----          ----          ----
--- ------------  ------------  ------------
1    .00000D+00    .50000D+00    .20000D+01
2    .37169D-02    .57672D+00    .29779D+01
=============================================================
```

3. CONLES method starting point 2 - successful solution

```
          EQUATION SOLVER AND MATRIX INFORMATION
                  **********************************
```

```
Maximum number of iterations (KMAX):                    100
Equation solver method flag (MFALG):                      4
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
-------------------------------------------------------------
```

```
          STATISTICS OF THE SIMULATION
                ============================

NO OF FUNCTION EVALUATIONS :        295
NO OF JACOBIAN EVALUATIONS :         29
NO OF ITERATIONS TAKEN     :         31
-------------------------------------------------------------
```

```
          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************

Total number of variables:                              8
Total number of equations:                              8
-------------------------------------------------------------
```

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.22000D+00	.75000D-01	.10000D-02	.58000D+00
2	.28065D+00	.49038D-01	.10428D-01	.64264D+00
3	.32287D+00	.92235D-02	.46017D-01	.61817D+00

S/N	VARIABLE 5	VARIABLE 6	VARIABLE 7
1	.12500D+00	.43600D+00	.23500D+01
2	.17947D-01	.55258D+00	.28406D+01
3	.37169D-02	.57672D+00	.29779D+01

4. Broyden's method starting point 2 - diverging after 100 iterations

```
          EQUATION SOLVER AND MATRIX INFORMATION
                *************************************

Maximum number of iterations (KMAX):                    100
Equation solver method flag (MFALG):                      2
Jacobian availability flag (LJAC):                        F
Initial Jacobian flag (KJAC):                             0
```

STATISTICS OF THE SIMULATION
==========

NO OF FUNCTION EVALUATIONS : 311
NO OF JACOBIAN EVALUATIONS : 21
NO OF ITERATIONS TAKEN : 100

---

* R E S U L T S O F S I M U L A T I O N *
****

Total number of variables: 8
Total number of equations: 8

S/N	VARIABLE 1	VARIABLE 2	VARIABLE 3	VARIABLE 4
1	.22000D+00	.75000D-01	.10000D-02	.58000D+00
2	.11390D+05	-.19640D+06	.13734D+05	-.20727D+06
3	.18119D+03	.54426D+01	.28713D+02	.38916D+03
4	.95480D+01	.90135D+00	.49475D+01	.20083D+02
5	.86941D+01	.38766D+00	.54114D+00	.12475D+02

S/N	VARIABLE 5	VARIABLE 6	VARIABLE 7
1	.12500D+00	.43600D+00	.23500D+01
2	-.17451D+05	.36588D+05	.55617D+05
3	.49583D+03	.35341D+06	.53753D+06
4	.85370D+02	.82242D+07	.12508D+08
5	.41689D+01	.33191D+08	.50479D+08

==========

A.2 Typical data files generated by the interactive front end for DASP II

These data files are those generated for the methanol mixer tank problem (see sections 7.1and 7.2)

CINDAT - The simulation control options file

```
   --------    11    ---------------------
'PROBLEM NO 1, METHANOL-MIXER SYSTEM'    'L SINGH ' , 'EXAM-
PLE 1.' , '18-04-1990'
2  ,  0  ,  1  ,  0  ,  1  ,  0
(mode,kmodel,lcomp,ktol,lprnt,nonneg)
2  ,  0  ,  0  ,  0  ,  .FALSE.
(index,lterm,jfbs,kjac,ljac)
0  ,  1  ,  0  ,  0
0.0  ,  1.0  ,  1  ,  0
OUTPUT
4
2,5,7,9
EVENT INFORMATION
4
1 , 2 , 1 , 1
0.0 , 0.0 , 0.083 , 0.1667
1 , 5 , 1 , 1
0 , 0 , 0 , 0
0 , 21 , 0 , 0
0.0 , 0.4 , 0.0 , 0.0
0.00001,0.00001,0.00001,0.00001
1 , -1 , 1 , 1
'TIME EVENT 1','STATE EVENT','TIME EVENT 2','TIME EVENT 3'
```

COMDT - The components information file

```
PROBLEM NO 9 -  MIXED-TANK SYSTEM   GENERAL DATA
2 , 0 , 0 , 0
1 , 2
32 , 18
```

CUNIT - The variable and parameter specification file

```
PROBLEM NO 9, METHANOL-MIXER SYSTEM REF.
FEED1 DATA
5
21 , 22 , 44 , 50 , 51
0.0 , 1.0 , 166.67 , 2.5D5 , 293.0
VALVE1 DATA
1 , 44 , 44 , 0
-5000.0 , 5000.0  ,  0.0  ,  5555.56 , 0.0 , 166.667 , 1.0
166.67
CONTROL1 DATA
21  , 44 , 0
```

```
0.0 , 1.0 , -5000.0 , 5000.0 , 0.0 , -1.0 , 75.0 , 0.0 , 1.0
, 0.4
1.5,0.1
TRANSMIT DATA
21 , 21 , 0
0.0  , 1.0  , 0.0  , 1.0  ,  0.0  ,  0.0   ,  0.003  ,  1.0
0.5
MIXED TANK DATA
3 , 2 , 0
34.214  ,  1.7672 , 1.0D5 , 293.0
0.661  ,  0.5  ,  0.5
FEED2 DATA
4   21 , 22 , 44 , 51
0.0714 , 0.9286 , 70.0 , 293.0
CONTROL2 DATA   49 , 44 , 0
0.0 , 1.6 , -5000.0 , 5000.0 , 0.0 , -1.0 , 400.0 , 0.0 ,
1.0  , 0.8
-55.6,0.139
TRANSMIT2 DATA
44 , 44 , 0
-5000.0, 5000.0, -5000.0, 5555.56,  0.0,  0.0,  0.0015,  4.0
-55.6
VALVE2 DATA
0 , 44 , 44 , 0
-5000.0 , 5000.0  ,  0.0  ,  5555.56 , 0.0 , 273.712 , 1.0
273.712
OUTLET DATA
2
44 , 50
273.712 , 1.0D5
FEED3 DATA
4
21 , 22 ,44 , 51
0.9 , 0.1 , 100.0 , 293.0
```

CTOPOL - The topology description file

```
PROBLEM NO 9, TOPOLOGY DATA CREATED FOR MOD DASP LS15/5/90
  UNITS DATA
11      1 ,  2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10, 11      0 ,
11 , 2 , 7 , 9 , 0 , 2 , 7 , 11 , 0, 0
0 ,  4 , 1 , 1 , 1 , 0,  1 , 1 ,  4 , 0, 0
'FEED1' , 'VALVE1','CONTROL1','TRANSMIT1','MIXED-TANK' ,
'FEED2', 'CONTROL2' , 'TRANSMIT2' , 'VALVE2' , 'OUTLET',
'FEED3'   STREAMS DATA
12
1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12
1 , 2 , 3 , 4 , 5 , 6 , 5 , 7 , 8 , 5 , 9, 11
2 , 5 , 2 , 3 , 4 , 5 , 7 , 8 , 9 , 9 , 10, 5
3 , 3 , 0 , 0 , 0 , 3 , 0 , 0 , 0 , 3 , 3, 3
'S1', 'S2', 'S3', 'S4', 'S5', 'S6', 'S7', 'S8', 'S9', 'S10',
'S11', 'S12'
CONNECTION MATRIX FOR METHANOL-MIXER SYSTEM
UNIT NO 1 (FEED1)
1     2
UNIT NO 2 (VALVE1)
3     1  ,  5  ,  3
UNIT NO 3 (CONTROL1)
2     4 , 2
UNIT NO 4 (TRANSMIT1)
2     5  ,  3
UNIT NO 5 (MIXED TANK)
6     2  ,  6  ,  11 , 9 , 4 , 7
UNIT NO 6 (FEED2)
1     5
UNIT NO 7 (CONTROL2)
2     5 , 8
UNIT NO 8 (TRANSMIT2)
2     7  ,  9
UNIT NO 9 (VALVE2)
3     5  ,  10  ,  8
UNIT NO 10 (OUTLET)
1     9
UNIT NO 11 (FEED3)
1     5
```

A3 Dynamic simulation study of the methanol mixer tank system

This problem was used as the dynamic simulation example in Chapter 7. The process flow-sheet is illustrated in Figure 7.1. The system description is as follows :

Water feed control valve

Minimum flowrate	:	0.0	Kmol/hr
Maximum flowrate	:	1800.	Kmol/hr
Gain	:	1.0	

Composition controller

Reverse acting Proportional and Integral action control

Integral action time	:	4.0	hrs
Proportional constant	:	15.0	
Set point	:	0.4	

Composition transmitter

Proportional gain	:	5.	
Time constant	:	0.003	hrs

Mixer tank

Cross-sectional area	:	1.7672	m^2
Initial methanol mole fraction	:	0.5	
Density of mixture	:	28.29	$Kmol/m^3$

Concentrated methanol feed

Methanol mole fraction	:	0.9	
Flowrate	:	100	Kmol/hr

Dilute methanol feed

Methanol mole fraction	:	0.0714	
Flowrate	:	70	Kmol/hr

Level controller

Reverse acting Proportional and Integral action control

Proportional gain	:	400	
Integral action time	:	5	hrs

Outlet valve

Proportional gain	:	4	
Time constant	:	0.0015	hrs
Minimum flowrate	:	0.0	Kmol/hr
Maximum flowrate	:	2829.33	Kmol/hr

Events

i	Run with methanol solution feeds off unitl either mole fraction of methanol <= 0.4 or 25 minutes have elapsed.
ii	Run with concentrated stream on, dilute stream off for 5 minutes.
iii	Run with both solution feeds on for 10 minutes.

The results for the controlled system are shown in section A3.1, whilst the uncontrolled system simulation results are shown in A3.2

A3.1 The controlled methanol mixer system

```
     ==============================================
    *  Dynamic Analysis and Simulation Package    *
    *        DASP  -  Version 2.1 , 1990          *
     ==============================================

     TITLE:  PROBLEM NO 1, METHANOL-MIXER SYSTEM
-------------------------------------------------------------


          * U S E R   A N D   J O B   D E S C R I P T I O N
          *

*****************************************************
User of simulator:                          L SINGH
Name of Project:                            EXAMPLE 1.
Date of Execution:                          18-04-1990
-------------------------------------------------------------

          INTEGRATION PARAMETERS
```

```
          **********************
Initial time of simulation (T0) (HR):                .00000D+00
Final time of simulation (TFIN) (HR):                .10000D+01
Current time of simulation (TIME) (HR):              .00000D+00
Integrator method flag (MFDYN):                              1
DAE solution flag (KDAE):                                    0
-------------------------------------------------------------

          SIMULATION  OPTIONS              ******************
Mode of simulation (MODE):                                   2
Model routines option (KMODEL):                              0
Error tolerance option (KTOL):                               0
Nonnegativity option (NONNEG):                               0
Model discontinuity option (IDISCN):                         0
Mono/multicomponent option (LCOMP):                          1
Output level option (INDEX):                                 2
Output interval option (LPRNT):                              1
Plotting option (LPLOT):                                     0
Event processing option (LEVENT):                            1
Integration stepsize option (MDEFLT):                        0
Jacobian matrix structure option (JFBS):                     0
-------------------------------------------------------------

          I/O DATA FILES USED              ******************
General input data file (CINDAT):                    CINDAT9
Topology data file (CTOPOL):                         CTOPOL9
Output/Result data file (CRESLT):                    RES9.AN
Initial values data file (CUNIT):                    CUNIT9.N
DASP work file (CWORK):                              WKFILE.D
Components data file (COMDT):                        COMDT9
-------------------------------------------------------------

          VALUES OF ERROR TOLERANCES
          **************************
              .10000D-02          .00000D+00


          STATISTICS OF THE SIMULATION
          ============================

     NO OF FUNCTION EVALUATIONS :       1342
     NO OF JACOBIAN EVALUATIONS :         72
     NO OF STEPS TAKEN          :        160
     NO OF ERROR TEST FAILURES  :         21
-------------------------------------------------------------

          * R E S U L T S   O F   S I M U L A T I O N *
          *********************************************
```

* UNIT NO = 2 MODULE NAME = VALVE1 *

S/N	TIME HR	LRATE kgmol/hr
1	.00000D+00	.16667D+03
2	.54979D-01	.16724D+03
3	.13798D+00	.16469D+03
4	.23798D+00	.16132D+03
5	.30468D+00	.15976D+03
6	.40468D+00	.15809D+03
7	.50468D+00	.15687D+03
8	.60468D+00	.15587D+03
9	.70468D+00	.15500D+03
10	.80468D+00	.15418D+03
11	.90468D+00	.15339D+03
12	.10000D+01	.15266D+03

* UNIT NO = 5 MODULE NAME = MIXED-TANK *

S/N	TIME HR	LHGHT m	XCOMP kgmol/kgmol	XCOMP kgmol/kgmol
1	.00000D+00	.66100D+00	.50000D+00	.50000D+00
2	.54979D-01	.72066D+00	.40000D+00	.60000D+00
3	.13798D+00	.79298D+00	.37163D+00	.62837D+00
4	.23798D+00	.83995D+00	.33130D+00	.66870D+00
5	.30468D+00	.83978D+00	.31588D+00	.68412D+00
6	.40468D+00	.83535D+00	.30302D+00	.69698D+00
7	.50468D+00	.83105D+00	.29689D+00	.70311D+00
8	.60468D+00	.82732D+00	.29416D+00	.70584D+00
9	.70468D+00	.82406D+00	.29315D+00	.70685D+00
10	.80468D+00	.82116D+00	.29296D+00	.70704D+00
11	.90468D+00	.81858D+00	.29320D+00	.70680D+00
12	.10000D+01	.81636D+00	.29362D+00	.70638D+00

* UNIT NO = 7 MODULE NAME = CONTROL2 *

S/N	TIME HR	LRATE kgmol/hr	VARIABLE -	VARIABLE -
1	.00000D+00	-.55600D+02	.13900D+00	.00000D+00
2	.54979D-01	-.34015D+02	-.79340D-01	-.22790D+01
3	.13798D+00	-.62320D+01	-.70151D-02	-.34259D+01
4	.23798D+00	.13539D+02	.39953D-01	-.24421D+01
5	.30468D+00	.14549D+02	.39783D-01	-

```
.13640D+01          6      .40468D+00        .14280D+02
.35347D-01     .14132D+00           7      .50468D+00
.13886D+02     .31047D-01     .14674D+01           8
.60468D+00        .13562D+02     .27324D-01     .26329D+01
9      .70468D+00        .13282D+02     .24057D-01
.36589D+01          10     .80468D+00        .13026D+02
.21160D-01     .45619D+01          11     .90468D+00
.12786D+02     .18576D-01     .53556D+01          12
.10000D+01        .12567D+02     .16365D-01     .60210D+01
```

```
          * UNIT NO =     9      MODULE NAME = VALVE2        *
          ********************************************************
```

S/N	TIME HR	LRATE kgmol/hr
1	.00000D+00	.27371D+03
2	.54979D-01	.13658D+03
3	.13798D+00	.24812D+03
4	.23798D+00	.32761D+03
5	.30468D+00	.33190D+03
6	.40468D+00	.33086D+03
7	.50468D+00	.32928D+03
8	.60468D+00	.32798D+03
9	.70468D+00	.32685D+03
10	.80468D+00	.32583D+03
11	.90468D+00	.32487D+03
12	.10000D+01	.32399D+03

```
    ------------------------------------------------------------
----
```

A3.2 The uncontrolled methanol mixer system

```
 ==========================================
*  Dynamic Analysis and Simulation Package  *
*        DASP  -  Version 2.1 , 1990        *
 ==========================================
```

TITLE: PROBLEM NO 1, METHANOL-MIXER SYSTEM

---

* U S E R A N D J O B D E S C R I P T I O N

****

User of simulator:	L SINGH
Name of Project:	EXAMPLE 1.
Date of Execution:	18-04-1990

INTEGRATION PARAMETERS

****

Initial time of simulation (T0) (HR):	.00000D+00
Final time of simulation (TFIN) (HR):	.10000D+01
Current time of simulation (TIME) (HR):	.00000D+00
Integrator method flag (MFDYN):	1
DAE solution flag (KDAE):	0

SIMULATION OPTIONS

****

Mode of simulation (MODE):	2
Model routines option (KMODEL):	0
Error tolerance option (KTOL):	0
Nonnegativity option (NONNEG):	0
Model discontinuity option (IDISCN):	0
Mono/multicomponent option (LCOMP):	1
Output level option (INDEX):	2
Output interval option (LPRNT):	1
Plotting option (LPLOT):	0
Event processing option (LEVENT):	1
Integration stepsize option (MDEFLT):	0
Jacobian matrix structure option (JFBS):	0

I/O DATA FILES USED

****

General input data file (CINDAT):	CINDAT9
Topology data file (CTOPOL):	CTOPOL9
Output/Result data file (CRESLT):	RES9.AN
Initial values data file (CUNIT):	CUNIT9.N
DASP work file (CWORK):	WKFILE.D
Components data file (COMDT):	COMDT9

STATISTICS OF THE SIMULATION

==========

```
     NO OF FUNCTION EVALUATIONS :      1319
     NO OF JACOBIAN EVALUATIONS :        73
     NO OF STEPS TAKEN          :       136
     NO OF ERROR TEST FAILURES  :        23
-------------------------------------------------------------
```

* R E S U L T S O F S I M U L A T I O N *

****

* UNIT NO = 2 MODULE NAME = VALVE1 *

****

S/N	TIME HR	LRATE kgmol/hr
1	.00000D+00	.16667D+03
2	.56189D-01	.16722D+03
3	.13919D+00	.16465D+03
4	.23919D+00	.16219D+03
5	.30589D+00	.16103D+03
6	.40589D+00	.15968D+03
7	.50589D+00	.15858D+03
8	.60589D+00	.15761D+03
9	.70589D+00	.15670D+03
10	.80589D+00	.15584D+03
11	.90589D+00	.15500D+03
12	.10000D+01	.15423D+03

* UNIT NO = 5 MODULE NAME = MIXED-TANK *

****

S/N	TIME HR	LHGHT m	XCOMP kgmol/kgmol	XCOMP kgmol/kgmol
1	.00000D+00	.66100D+00	.50000D+00	.50000D+00
2	.56189D-00	.76780D+00	.40000D+00	.60000D+00
3	.13919D+00	.10434D+01	.37138D+00	.62862D+00
4	.23919D+00	.14989D+01	.34267D+00	.65733D+00
5	.30589D+00	.18080D+01	.33156D+00	.66844D+00
6	.40589D+00	.22695D+01	.32113D+00	.67887D+00
7	.50589D+00	.27289D+01	.31478D+00	.68522D+00
8	.60589D+00	.31867D+01	.31059D+00	.68941D+00
9	.70589D+00	.36429D+01	.30764D+00	.69236D+00
10	.80589D+00	.40976D+01	.30548D+00	.69452D+00
11	.90589D+00	.45510D+01	.30387D+00	.69613D+00
12	.10000D+01	.49764D+01	.30273D+00	.69727D+00

```
          * UNIT NO =    7     MODULE NAME = CONTROL2     *
          ********************************************************

      S/N      TIME           SP          VARIABLE
VARIABLE                   HR              -              -
-           -----  ----------  ------------  ------------
------------          1  .00000D+00   .80000D+00
.13900D+00    .00000D+00                    2  .56189D-01
.89921D+00   -.13141D+00   -.30380D+01                    3
.13919D+00   .11643D+01   -.12094D+00   -.72245D+01
4  .23919D+00   .16083D+01   -.10943D+00   -.11828D+02
5  .30589D+00   .17024D+01   -.10238D+00   -.14650D+02
6  .40589D+00   .16926D+01   -.92633D-01   -.18547D+02
7  .50589D+00   .16838D+01   -.83817D-01   -.22073D+02
8  .60589D+00   .16758D+01   -.75841D-01   -.25264D+02
9  .70589D+00   .16686D+01   -.68624D-01   -.28150D+02
10  .80589D+00   .16621D+01   -.62094D-01   -.30762D+02
11  .90589D+00   .16562D+01   -.56185D-01   -.33126D+02
12  .10000D+01   .16511D+01   -.51139D-01   -.35144D+02


          * UNIT NO =    9     MODULE NAME = VALVE2       *
          ********************************************************

      S/N      TIME          LRATE
               HR          kgmol/hr
     -----  ----------  ------------
        1  .00000D+00       .27371D+03
        2  .56189D-01       .51312D+02
        3  .13919D+00       .51312D+02
        4  .23919D+00       .51312D+02
        5  .30589D+00       .51312D+02
        6  .40589D+00       .51312D+02
        7  .50589D+00       .51312D+02
        8  .60589D+00       .51312D+02
        9  .70589D+00       .51312D+02
       10  .80589D+00       .51312D+02
       11  .90589D+00       .51312D+02
       12  .10000D+01       .51312D+02

------------------------------------------------------------
```

A.4 The location of multiple solutions of NLAEs using CONSOL

Each separate solution is tracked by unique paths. In this section the path statistics are given for the example problems in section 7.3.2.

A.4.1 Problem 1 - Burning of fuel (Morgan(1987))

The values of the parameters used in this example are as follows:

a11 = 0.194997e48	*a12 = 0.131824e23*	*a13 = 0.321383e22*
a14 = 1.0	*a15 = 3.000000e-5*	*a21 = 0.118048e51*
a22 = 0.115344e48	*a23 = 0.309912e26*	*a24 = -0.86209e47*
a25 = -0.87235e25	*a26 = -0.26249e21*	*a27 = -0.80000e-9*

Nine solutions exist for this problem.

```
  PATH NUMBER =     1

  FINAL VALUES FOR PATH

  PATH NUMBER =      1
TOTAL NUMBER OF STEPS =    284
NUMBER OF STEPS THAT SUCCEEDED =    258
NUMBER OF STEPS THAT FAILED    =     26
TOTAL CORRECTOR ITERATIONS ON PATH  =  611
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  895
PATH ARC LENGTH =  .3269D+01

  NUMPAT=     1
REAL SOLUTION
-------------------------------------------------
T  =   .100000000000000D+01
               REAL PART              IMAGINARY PART
X(     1)  =  -.3920D-24               .3582D-33
X(     2)  =  -.9290D-23               .2603D-13
NORM OF RESIDUAL =   .3980D-16
-------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .2206D+11       DET  =  .1887D+00 -.1350D+02


  PATH NUMBER =     2

  FINAL VALUES FOR PATH

  PATH NUMBER =      2
TOTAL NUMBER OF STEPS =    297
NUMBER OF STEPS THAT SUCCEEDED =    269
NUMBER OF STEPS THAT FAILED    =     28
```

```
TOTAL CORRECTOR ITERATIONS ON PATH  =  651
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  948
PATH ARC LENGTH =  .2833D+01

  NUMPAT=    2
REAL SOLUTION
-----------------------------------------------------
T  =   .100000000000000D+01
              REAL PART            IMAGINARY PART
X(    1)  =  -.1282D-24            .1188D-38
X(    2)  =  -.6478D-13            .6030D-27
NORM OF RESIDUAL =   .2967D-15
-----------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .1351D+12      DET  =  .5757D-01 -.2177D+01


  PATH NUMBER =    3

  FINAL VALUES FOR PATH

  PATH NUMBER =     3
TOTAL NUMBER OF STEPS =   241
NUMBER OF STEPS THAT SUCCEEDED =   223
NUMBER OF STEPS THAT FAILED    =    18
TOTAL CORRECTOR ITERATIONS ON PATH  =  477
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  718
PATH ARC LENGTH =  .3979D+01

  NUMPAT=    3
REAL SOLUTION
-----------------------------------------------------
T  =   .100000000000000D+01
              REAL PART            IMAGINARY PART
X(    1)  =   .1491D-14            .1647D-28
X(    2)  =   .3212D-18            .3555D-32
NORM OF RESIDUAL =   .1444D-14
-----------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .1256D+04      DET  = -.1716D+07 -.1614D+07


  PATH NUMBER =    4

  FINAL VALUES FOR PATH

  PATH NUMBER =     4
TOTAL NUMBER OF STEPS =   255
NUMBER OF STEPS THAT SUCCEEDED =   233
NUMBER OF STEPS THAT FAILED    =    22
TOTAL CORRECTOR ITERATIONS ON PATH  =  523
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  778
PATH ARC LENGTH =  .7932D+01

  NUMPAT=    4
```

```
REAL SOLUTION
-------------------------------------------------
T  =    .100000000000000D+01
                REAL PART              IMAGINARY PART
X(     1)  =    .1491D-14             -.2718D-29
X(     2)  =   -.3212D-18              .6045D-33
NORM OF RESIDUAL =    .1380D-14
-------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =   .1256D+04       DET  =   .1716D+07   .1614D+07


  PATH NUMBER =     5

  FINAL VALUES FOR PATH

  PATH NUMBER =      5
TOTAL NUMBER OF STEPS =    255
NUMBER OF STEPS THAT SUCCEEDED =    233
NUMBER OF STEPS THAT FAILED    =     22
TOTAL CORRECTOR ITERATIONS ON PATH  =  517
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  772
PATH ARC LENGTH =   .1832D+01

  NUMPAT=     5
REAL SOLUTION
-------------------------------------------------
T  =    .100000000000000D+01
REAL PART              IMAGINARY PART
X(     1)  =   -.1282D-24              .1468D-38
X(     2)  =    .6478D-13             -.7256D-27
NORM OF RESIDUAL =    .7828D-17
-------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =   .1359D+12       DET  =   .5071D-01 -.2197D+01


  PATH NUMBER =     6

  FINAL VALUES FOR PATH

  PATH NUMBER =      6
TOTAL NUMBER OF STEPS =    230
NUMBER OF STEPS THAT SUCCEEDED =    213
NUMBER OF STEPS THAT FAILED    =     17
TOTAL CORRECTOR ITERATIONS ON PATH  =  428
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  658
PATH ARC LENGTH =   .3105D+01

  NUMPAT=     6
REAL SOLUTION
-------------------------------------------------
T  =    .100000000000000D+01
                REAL PART              IMAGINARY PART
X(     1)  =   -.1491D-14              .1569D-21
```

```
X(    2)  =  -.1690D-25           -.3212D-18
NORM OF RESIDUAL =   .1282D-14
-------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .1293D+04      DET  = -.1660D+07  .1681D+07


  PATH NUMBER =    7

  FINAL VALUES FOR PATH

  PATH NUMBER =     7
TOTAL NUMBER OF STEPS =   288
NUMBER OF STEPS THAT SUCCEEDED =   261
NUMBER OF STEPS THAT FAILED    =    27
TOTAL CORRECTOR ITERATIONS ON PATH  =  634
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  922
PATH ARC LENGTH =  .2614D+01

  NUMPAT=    7
COMPLEX SOLUTION
-------------------------------------------------
T  =   .100000000000000D+01
               REAL PART           IMAGINARY PART
X(    1)  =   .0000D+00           .0000D+00
X(    2)  =  -.1938D+01           .5934D+00
NORM OF RESIDUAL =   .6879D-17
-------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .3147D+12      DET  = -.2332D-01  .9415D+00


  PATH NUMBER =    8

  FINAL VALUES FOR PATH

  PATH NUMBER =     8
TOTAL NUMBER OF STEPS =   276
NUMBER OF STEPS THAT SUCCEEDED =   252
NUMBER OF STEPS THAT FAILED    =    24
TOTAL CORRECTOR ITERATIONS ON PATH  =  582
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  858
PATH ARC LENGTH =  .7810D+01

  NUMPAT=    8
REAL SOLUTION
-------------------------------------------------
T  =   .100000000000000D+01
               REAL PART           IMAGINARY PART
X(    1)  =  -.1491D-14           -.1569D-21
X(    2)  =  -.1690D-25            .3212D-18
NORM OF RESIDUAL =   .4252D-15
-------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
```

```
COND =  .1293D+04      DET  =  .1661D+07 -.1681D+07


  PATH NUMBER =    9

  FINAL VALUES FOR PATH

  PATH NUMBER =     9
TOTAL NUMBER OF STEPS =   320
NUMBER OF STEPS THAT SUCCEEDED =   289
NUMBER OF STEPS THAT FAILED    =    31
TOTAL CORRECTOR ITERATIONS ON PATH  =  727
TOTAL LINEAR SYSTEMS SOLVED ON PATH = 1047
PATH ARC LENGTH =  .5532D+01

  NUMPAT=    9   REAL SOLUTION
-------------------------------------------------
T  =   .100000000000000D+01
               REAL PART           IMAGINARY PART
X(    1)  =  -.3920D-24          -.3582D-33
X(    2)  =  -.9290D-23          -.2603D-13
NORM OF RESIDUAL =   .2848D-15
-------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .2167D+11      DET  =  .4840D+00 -.1360D+02



  TOTAL CORRECTOR ITERATIONS  =      5150


  TOTAL LINEAR SYSTEMS SOLVED =      7596


  TOTAL ARC LENGTH  =  .3891D+02
```

A.4.2 Problem 2 - Chemical equilbrium problem (Hiebert(1982))

Four solutions exit for this problem.

```
  PATH NUMBER =    1

  FINAL VALUES FOR PATH

  PATH NUMBER =     1
TOTAL NUMBER OF STEPS =   167
NUMBER OF STEPS THAT SUCCEEDED =   148
NUMBER OF STEPS THAT FAILED    =    19
TOTAL CORRECTOR ITERATIONS ON PATH  =  422
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  589
PATH ARC LENGTH =  .4935D+03
```

```
  NUMPAT=    1
REAL SOLUTION
-------------------------------------------------------
T  =    .1000000000000000D+01
                REAL PART              IMAGINARY PART
X(     1)  =    .1000D-03              -.5530D-15
X(     2)  =    .1000D-02              -.5530D-14
X(     3)  =   -.1000D-03              -.5529D-14
X(     4)  =   -.1000D-03               .6082D-14
X(     5)  =    .5500D+02              -.1249D-13
X(     6)  =   -.1000D-09               .5529D-20
NORM OF RESIDUAL =    .3034D-09
-------------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =   .2666D+10      DET  =   .1370D+04   .1222D+04


  PATH NUMBER =     2

  FINAL VALUES FOR PATH

  PATH NUMBER =      2
TOTAL NUMBER OF STEPS =    112
NUMBER OF STEPS THAT SUCCEEDED =     98
NUMBER OF STEPS THAT FAILED    =     14
TOTAL CORRECTOR ITERATIONS ON PATH  =  256
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  368
PATH ARC LENGTH =   .9305D+02

  NUMPAT=    2
REAL SOLUTION
-------------------------------------------------------
T  =    .1000000000000000D+01
                REAL PART              IMAGINARY PART
X(     1)  =    .8264D-04              -.8344D-15
X(     2)  =    .8264D-03              -.8344D-14
X(     3)  =    .9091D-04              -.1010D-13
X(     4)  =    .9091D-04               .9179D-14
X(     5)  =    .5500D+02              -.1498D-14
X(     6)  =    .1100D-09               .1222D-19
NORM OF RESIDUAL =    .5435D-10  ----------------------------
------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =   .2026D+09      DET  = -.2749D+03 -.4024D+03


  PATH NUMBER =     3

  FINAL VALUES FOR PATH

  PATH NUMBER =      3
TOTAL NUMBER OF STEPS =     78
NUMBER OF STEPS THAT SUCCEEDED =     71
NUMBER OF STEPS THAT FAILED    =      7
TOTAL CORRECTOR ITERATIONS ON PATH  =  158
```

```
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  236
PATH ARC LENGTH =  .7016D+02

  NUMPAT=    3
COMPLEX SOLUTION
-----------------------------------------------------
T  =   .100000000000000D+01
REAL PART           IMAGINARY PART
X(    1)  =  -.1863D+14            .9159D+12
X(    2)  =  -.1863D+15            .9159D+13
X(    3)  =   .0000D+00            .0000D+00
X(    4)  =   .2049D+15           -.1007D+14
X(    5)  =   .2049D+15           -.1007D+14
X(    6)  =  -.2049D+15            .1007D+14
NORM OF RESIDUAL =   .2765D-12
-----------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .5795D+07      DET  =  .1266D+05 -.1102D+06


  PATH NUMBER =    4

  FINAL VALUES FOR PATH

  PATH NUMBER =     4
TOTAL NUMBER OF STEPS =    96
NUMBER OF STEPS THAT SUCCEEDED =    88
NUMBER OF STEPS THAT FAILED    =     8
TOTAL CORRECTOR ITERATIONS ON PATH  =  224
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  320
PATH ARC LENGTH =  .8145D+02

  NUMPAT=    4
REAL SOLUTION
-----------------------------------------------------
T  =   .100000000000000D+01
 REAL PART           IMAGINARY PART
X(    1)  =  -.1000D-09            .2746D-21
X(    2)  =  -.1000D-08            .2746D-20
X(    3)  =  -.1000D-10            .2745D-22
X(    4)  =   .1000D-02           -.5840D-19
X(    5)  =   .5500D+02           -.3655D-14
X(    6)  =  -.1000D-02           -.2745D-14
NORM OF RESIDUAL =   .4047D-12
-----------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .1141D+07      DET  =  .3179D+03  .5681D+03



  TOTAL CORRECTOR ITERATIONS  =      1060
```

```
  TOTAL LINEAR SYSTEMS SOLVED =      1513

  TOTAL ARC LENGTH  =  .7382D+03
```

A.4.3 Problem 3 - Solution of reaction rate equations (Shacham(1986))

Thity five solutions exit for this problem. However only the path statistics are given for the Real solutions.

```
  PATH NUMBER =    7

  FINAL VALUES FOR PATH

  PATH NUMBER =     7
TOTAL NUMBER OF STEPS =    63
NUMBER OF STEPS THAT SUCCEEDED =    56
NUMBER OF STEPS THAT FAILED    =     7
TOTAL CORRECTOR ITERATIONS ON PATH  =  131
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  194
PATH ARC LENGTH =  .6339D+01

  XNP1=    -.6232698204D-01     .4119090159D-01

  NUMPAT=    7
REAL SOLUTION
-----------------------------------------------------
T  =   .100000000000000D+01
REAL PART           IMAGINARY PART
X(    1)  =   .5195D-02          -.4592D-17
X(    2)  =   .3368D+00          -.3598D-15
X(    3)  =   .1990D+01          -.1831D-14
X(    4)  =  -.6411D-03           .6990D-18
X(    5)  =  -.5121D+01           .3656D-14
X(    6)  =   .6122D+01          -.4690D-14
NORM OF RESIDUAL =   .3896D-15
-----------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .3208D+04      DET  = -.2119D+00 -.2259D+00


  PATH NUMBER =    8

  FINAL VALUES FOR PATH

  PATH NUMBER =     8
TOTAL NUMBER OF STEPS =    55
NUMBER OF STEPS THAT SUCCEEDED =    50
NUMBER OF STEPS THAT FAILED    =     5
TOTAL CORRECTOR ITERATIONS ON PATH  =  107
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  162
PATH ARC LENGTH =  .8197D+01
```

```
  XNP1=      .1486315485D-01     -.1296089376D+00

  NUMPAT=    8
REAL SOLUTION
-----------------------------------------------------
T  =    .100000000000000D+01
REAL PART              IMAGINARY PART
X(    1)  =    .1030D+01              .2026D-15
X(    2)  =    .1020D+01              .2882D-15
X(    3)  =   -.6086D-01             -.1528D-16
X(    4)  =   -.1003D-03             -.2318D-19
X(    5)  =    .1001D+01              .2337D-15
X(    6)  =   -.9517D-03             -.2311D-18
NORM OF RESIDUAL =    .3275D-15
-----------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =   .1382D+03       DET  =   .5024D+00   .2542D+00


  PATH NUMBER =    11

  FINAL VALUES FOR PATH

  PATH NUMBER =     11
TOTAL NUMBER OF STEPS =     70
NUMBER OF STEPS THAT SUCCEEDED =     59
NUMBER OF STEPS THAT FAILED    =     11
TOTAL CORRECTOR ITERATIONS ON PATH  =  149
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  219
PATH ARC LENGTH =   .4978D+01

  XNP1=      .3926951297D-02      .1653040288D-02

  NUMPAT=    11
REAL SOLUTION
-----------------------------------------------------
T  =    .100000000000000D+01
REAL PART              IMAGINARY PART
X(    1)  =   -.5454D-01             -.1364D-15
X(    2)  =    .2970D+00             -.1191D-14
X(    3)  =    .2109D+01              .6771D-14
X(    4)  =   -.7703D+01             -.6985D-13
X(    5)  =   -.4518D-03              .2224D-17
X(    6)  =    .8703D+01              .7069D-13
NORM OF RESIDUAL =    .4102D-13
-----------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =   .9627D+05       DET  =   .1714D-06 -.4943D-05


  FINAL VALUES FOR PATH

  PATH NUMBER =     19
TOTAL NUMBER OF STEPS =     85
NUMBER OF STEPS THAT SUCCEEDED =     77
```

```
NUMBER OF STEPS THAT FAILED     =      8
TOTAL CORRECTOR ITERATIONS ON PATH  =  185
TOTAL LINEAR SYSTEMS SOLVED ON PATH =  270
PATH ARC LENGTH =  .9314D+02

  XNP1=    -.2642917378D-01    -.1619146299D-01

  NUMPAT=   19
REAL SOLUTION
 ------------------------------------------------------
T  =   .100000000000000D+01
REAL PART           IMAGINARY PART
X(    1)  =   .9742D+00             .3261D-15
X(    2)  =   .9828D+00             .3839D-15
X(    3)  =   .5151D-01             .3316D-16
X(    4)  =   .9357D+00             .4197D-15
X(    5)  =   .9084D-04             .5762D-19
X(    6)  =   .6424D-01             .3931D-16
NORM OF RESIDUAL =   .2344D-12
 ------------------------------------------------------

  CONDITION AND DETERMINANT OF DF(X) AT END OF PATH
COND =  .7104D+04      DET  =  .1004D-02 -.1350D-02


  TOTAL CORRECTOR ITERATIONS  =      5891

  TOTAL LINEAR SYSTEMS SOLVED =      8739

  TOTAL ARC LENGTH  =  .5494D+03
```

APPENDIX B

B.1 Introduction

Appendix B is a mini manual and illustrates how DASP II can be run on a 1MB PC.

B.2 Execution of PFG

The cursor throughout this version of PFG is moved by using the cursor keys. All selections are made by positioning the cursor at the desired location followed by a carriage return. PFG is resident in directory C:/USERS/PFG/PFGOBJ and is executed by entering PFG at the DOS prompt. The following steps are required to create a graphical flowsheet with PFG:

i) Enter a three character plant/process name when prompted. This name will be used as an extension for the files generated for this flowsheet.

ii) Enter the number of process units NUNITS comprising the flowsheet including input/output modules.

iii) For each unit enter the following:

   a. Unit tagname (maximum 3 characters)

   b. Unit description (maximum 12 characters)

iv) One the PFG graphics screen has displayed, selected PFD from the options menu.

v) For each unit in the flowsheet do the following operation:

   a. Select a tagname from the TAGNAME MENU

   b. Select the symbol from the SYMBOLS MENU

   c. Select the symbol size using the SCALE option from the OPTIONS MENU

d. Select the location of the symbol in the DRAWING SCREEN

vi) To draw lines, the LINE option is selected from the OPTIONS MENU. Once this selection has been made the LINE MENU replaces the SYMBOLS MENU.

vii) Select the LINE option from the OPTIONS MENU

viii) For each line, starting with line number 1, in the flowsheet the following procedure is required:

a. Select the starting point of the line ensuring that the starting point either touches or is embedded in the source symbol.

b. Select the end point of the line, again ensuring the end point either touches or is embedded in the sink symbol.

The lines can be composed of one or more segments. In this case the segments must be drawn in order starting from segment 1. The start point of segment 2 is the end point of segment 1, this procedure is repeated until the desired line is generated.

ix) Lines may be deleted by selecting the ERASE option from the OPTIONS MENU followed by the selection of the desired line.

x) To return to the PFD screen the PFG option is selected from the OPTIONS MENU.

xi) The graphical flowsheet can be saved by selecting the SAVE/EXIT option from the OIPTIONS MENU. At this point the program returns to the standard alphanumeric format.

B.3 Execution of the interactive front end - FRONT

FRONT the interactive front end generates the problem description for DASP together with the process topological data. This information is stored in a suite of a suite of ASCII files, which are then accessed by DASP. FRONT interacts with the user via a conversational procedure and the following steps are taken.

i) Are the simulation control options to be generated by using CINDAN.

ii) If yes then CINDAN is executed.

ii b) If no, the program requests the mode of operation and the componential information.

iii) Enquire is the topological information is to be generated for DASP.

iii b) If yes, BTOPOL is executed.

iv) Enquire is the model variable/parameter information is to be generated.

v) If yes, CUNIN is executed.

B.4 Execution of DASP

B.4.1 Introduction

The following files in C:/USERS/DASP/VER2 are needed to run DASPII:

1 PDASP.BAT

2 PDASP1.EXE

3 DASP.BAT

4 DASP.LINK

5 MESFLE.D

1 All the files generated by FRONT in Appendix B1 (i.e. CINDAT, CUNIT, CTOPOL, VCODES. D and COMDT if chemical species are present in the system).

2 "PDASP" is entered at the DOS prompt.

3 The topology data file name is entered. The program then prompts the user for the maximum lengths of the real and integer work spaces, CORE (*) and ICORE (*) arrays used by DASP. PDASP uses the topology data file to prepare two routines, the main program - DASP and the rubroutine GETSUB in the file MDASP.FOR. This routine is then compiled to produce MDASP.OBJ as the object file.

4 "MDASP" is entered at the DOS prompt. This links MDAPS.OBJ with the DASP programs and libraries creating the executabler file: DASP.EXE.

5 DASP is then executed by entering "DASP" at the DOS prompt.

In order to run the DASP program, the user should do the following:

1 Prepare the necessary data files as described in Appendix B.

2 The user must code the new routines in DASP format. These routines must then be compiled and linked with DASP library programs to produce the executable file, DASP.EXE.

The package is run by typing:

"DASP" <ENTER>

The system displays the following:

Welcome to DASP

(Dynamic Analysis and Simulation Package)

Version 1.1 1986, 1987

** Initialization Region **

Press < ENTER > to continue ...

The last message was produced by a routine that clears the screen if the < ENTER > key is depressed on the keyboard. A similar routine clears the screen automatically. The first option gives the user the opportunity to view the displayed message before it disappears.

B4.2 Initial Region

The following message will appear:

** Enter units used for input data **

1 = SI Units 2 = British Units

There are only two types of units supported, and the user can only use either of the two to input his data to DASP (see Appendix B8 for details of these units). However, if UM option is used, then it does not matter which of these is used as the user can describe the problem consistently in any units.

The user is prompted to enter the names of the data files and the result files. Note that if any typing errors are made, the user has up to 3 trials after which

execution is stopped. When any input data file name is read, it is opened for sequential access and the values in the file are read. If any errors are detected in the input data files, the execution is terminated. If scalar relative error tolerance option is to be used (KTOL = 0) then the user is prompted to enter the scalar error tolerance. If any discontinuity will occur in the model at a specified time, TSTOP, then the user is prompted for the value of TSTOP. If the user wants output to be given at specified output intervals, then the value of this interval is prompted. If the user wants output at specified time interval (in dynamic simulation option or after as number of iterations (in steady state simulation option, then he is prompted for the value of HOUT, the output time interval or NOUT, the number of iterations before output.

Afterwards the following is displayed:

*Enter interrupt/break flag (LINTRP) as follows **

0 = No break/interrupt

1 = Break after a given number of steps/iterations

2 = Break at every step/iteration

-1 - Interrupt/stop simulation now

If LINTRP = 0, then integration will proceed from initial time, TO to the final time, TFIN without any break, except when an event has occurred. If LINTRP = 1, then the user wants control returned to him via the MODIFY routine (explained in Chapter 8) after a given number of steps/iterations. In this case he is prompted for the value of NINTRP the number of iterations before break. For LINTRP = 2, this is equivalent to a break after every iteration or step. However, if LINTRP = -1 is entered, the simulation is abandoned and a message asking the user whether to stop or restart the simulation is displayed.

The plotting routine is called to initialize the plotting parameters, if plotting option is specified. Note that if the UM option is being executed, then the initial section of the subroutine USRSUB (i.e. JS = 00 is called to do any initializations specified by the user. To end the initialization process, the consistency of the values of the simulation variables are verified and if any inconsistency is detected, simulation is abandoned. Finally, the MODIFY routine is called. On exit from the MODIFY routine, simulation enters the dynamic region.

B4.3 Dynamic Region

This region carries out the numerical calculations of the steady state and dynamic simulation sessions. At the end of every step or iteration, control is passed to the Executive program to check which flags have been set. First it checks if any error has occurred (i..e. if IFLAG < 0), in which case the relevant error message is displayed and control is passed to the error analysis section of the Terminal Region. It then checks if the end of simulation has been reached (LEND=TRUE_, in which case control is passed to the Rerun/End simulation section of the Terminal Region. It then checks if the event flag, LEVENT and/or break flag, LINTRP has been set in which case the event processing interface routine, STESUB or the MODIFY routine is called to carry out the necessary procedure. Note that the user can abandon the simulation during this time by setting LINTRP to a negative value using option 10 of the MODIFY routine.

If none of these flags has been set, but the user has requested for output at specified time intervals or after a certain number of iterations, then the routine, OUTPUT is called to write the values of variables and derivatives to the work file, WKFILE and to the screen if LTERM=1. Finally, on return from any of the above routines, a check is made to find out if any errors were detected, in which case the relevant error message is displayed and control passed to the error analysis section

of the Terminal Region. Otherwise, if the user has set LINTRP to a negative value, signifying an interrupt, then the simulation is abandoned and the message

SIMULATION ABONDONED BY USER AT THE TIME.....

Select an option as follows:

0 = STOP

1 = RESTART

If 0 is selected, than all open files are closed after the results have been written to the result file and the simulation is abandoned. If 1 is selected, then the results are written to the results file, all open files closed, the pointers reset to zero and the simulation is restarted.

If no interrupts or errors were found, then the relevant simulation interface routine (DERIV for dynamic and ASSUB for steady state) is called to continue the simulation. This procedure is repeated until the final time, TFIN is reached and control is passed to the Terminal Region.

B4.4 Terminal Region

At the terminal region, if any errors have occurred typified by IFLAG < 0,m then using the absolute value of the flag, the relevant error messages are displayed on the screen and further messages from the message file,.MESFLE and execution is stopped. Otherwise, the following message appears:

****** TERMINAL REGION *****

Terminal Menu is

0 = Exit/End simulation

1 = Rerun present system after perturbation

2 = Run a new system with data in same files

3 = Run a new system using new data files

4 = Modify, plot or view variables/parameters

** Select an option number **

To exit, you must select option 0 above, in which case, the results are written to the result file and all files closed before stopping the simulation by returning control to the main program, accompanied by the display of the message.

NORMAL END OF SIMULATION

To view the latest results or plot the values of variables against time (if any plotting routines are available) select option 4, which calls the MODIFY routine. To rerun present system after a call to the MODIFY routine, select option 1. The user can restart the simulation starting from any step between the initial values and the latest variable values. Choosing option 2 means that the user wants to start a new simulation of a problem whose data are in the same files, placed just after the data for a previous simulation. In this case the result of the previous simulation is written to the result file, and an initialization of a new run will be started.

In option 3 the present simulation is ended and a reinitialization activated for a new run. Since all previous data files are closed, a new run must use new data files.

Depending on the option chosen, after a return from the Terminal Region, the Executive directs the simulation to the relevant region.

APPENDIX C

VARIABLE AND PARAMETER TYPES AND SYSTEM ERRORS

C.1 Introduction

This section presents the variable and parameter types defined in DASP. At present 100 variable and parameter types have been defined and each is given a unique positive code number and a name. In order to handle an equilibrium stage as a unit module, the vapor and liquid variable types are distinguished. Also the units which can be used to input and output the values of these variables and parameters are given. Two types of units have been defined, name SI and British Units. The user can only use one type of units for all the variables and parameters as there is no conversion between units. Note that the distinction between variables and parameters is arbitrary, as this may depend on the modelling assumptions made and what the model is intended to bt used for. Also, the code numbers of the integer type parameters are also given. This was intended to be used in conjunction with an input language.

C.2 VARIABLE TYPES

Variable Type	Variable Name	Code Number	SI Unit	British Unit
Vapor mole fractions	YCOMP	1-20	kgmole / kgmole	lbmole / lbmole
Liquid mole fractions	XCOMP	21-40	kgmole / kgmole	lbmole / lbmole
Vapor holdup	VHOLD	41	kgmol	lbmole
Liquid holdup	LHOLD	42	kgmole	lbmole
Total vapor flow rate	VRATE	43	kgmole/hr	lbmole/hr
Total liquid flow rate	LRATE	44	kgmole/hr	lbmole/hr
Vapor volume	VVOL	45	m3	ft3
Liquid volume	LVOL	46	m3	ft3
Vapor density	VDENS	47	kgmole/m3	lbmole/ft3

Variable Type	Variable Name	Code Number	SI Unit	British Unit
Liquid density	LDENS	48	kgmole/m3	1bmole/ft3
Liquid level	LHGHT	49	m	ft
Pressure	PRES	50	Pa	atm
Temperature	TEMP	51	K	°F
Vapor enthalph	VENTH	52	kJ/kgmole	Btu/1bmole
Liquid enthalph	LENTH	53	kJ/kgmole	Btu/1bmole
Percentage	PCENT	54	%	%
Pneumatic control signal	PSIGN	55	psig	psig
Electronic signal	ESIGN	56	mA	mA
Heat load	HTLOAD	57	kJ/hr	Btu/hr

Variable Type	Variable Name	Code Number	SI Unit	British Unit
Unspecified variable	VARIABLE	58-60	-	-
Zero of input	ZI	61	Same as the input variable	Same as the input variable
Range of input	RI	62	Same as the input' variable	Same as the input variable
Zero of output	ZO	63	Same as the output variable	Same as the output variable
Range of output	RO	64	Same as the output variable	Same as the output variable
Time constant	TAU	65	hrs	hrs
Zero offset of input	XO	66	Same as input	Same as input
Bias or manual reset	YO	67	Same as output variable	Same as output variable

Variable Type	Variable Name	Code Number	SI Unit	British Unit
Controller action +1 for direct -1 for reverse	AXN	68	-	-
System or process gain	GAIN	69	Unit of output / Unit of input	Unit of output / Unit of input
Set point of controller	SP	70	Same as input variable	Same as input variable
Integral time	TI	71	hrs	hrs
Derivative time	TD	72	hrs	hrs
Damping ratio	DAMP	73	-	-
Value constant	CV	74	kgmole/hr Pa2	lbmole/hratm2

Variable Type	Variable Name	Code Number	SI Unit	British Unit
Fractional valve opening	AV	75	-	-
Equal % value trim constant	ALFV	76	-	-
Individual/ composite heat transfer coefficient	HI0	77	kJ/hr m2K	Btu/hrft2°F
Overall heat transfer coefficient	UHTC	78	kJ/hr m2K	Btu/hrft2°F
Diameter	DIAM	79	m	ft
Area	AREA	80	m2	ft2
Efficiency	EFF	81	-	-
Ratio (eg. reflux or split fraction)	RATIO	82	-	-
Mass of metal, material etc	MASS	83	kg	1b

Variable Type	Variable Name	Code Number	SI Unit	British Unit
Molecular weight	MOLWT	84	-	-
Heat capacity	CP	85	kJ/kgmole-K	Btu/lbmole-°F
Latent heat of vaporization	HVAP	86	kJ/kgmole	Btu/lbmole
Heat of reaction	HTR	87		
Controller manual setting	CMAN	88	Same as output variable	Same as output variable
Number of revolutions per minute	RPM	89	1/min	1/min
Unspecified parameter	PARAMETER	90-100	-	-

C.3 INTEGER PARAMETERS

Parameter Name	Code Number	Description
MOPTN	1	The model option variable, which gives the various options based on the modelling assumptions,eg. isothermal or adiabatic option.
MPARAM	2	The parameter option variable, which gives the various options available for a specified parameter, eg. set point is constant or variable.
ICODEI	3	Code number of input variable
ICODEO	4	Code number of output variable
IDERIV	5	Derivative avaialability option, which has values as follows: = 1 derivatives of variable supplied, = 0 derivatives not avaialable
NIS	8	No of input streams
NR	10	No of reaction paths
JOPTN	11	Controller mode option
NC	12	No of comps in model

1 The amount of integer type workspace available for this problem is insufficient. The array ICROE should be increased as indicated in the accompanying message.

2 The amount of real type workshace available for this problem is insufficient. The array CORE should be increased as indicated in the accompanying message.

3 The number of variables/equations in the system to be solved should be greater than zero.

4 The erro tolerance(s) specified are too stringent. As a guide, they should be greater than the machine round-off number as given in the message above.

5 The local error test during the numerical calculation could not be satisfied, probably because a zero component was specified in both the relative (RTOL) and absolute (ATOL) error tolerances.

6 I/O error. An end of file occurred during an input/output operation. This normally happens when data is being read from a sequential file without first rewinding the file.

7 I/O error. The system was unable to access a file for input/outpur. Maybe the file does not exist or has not been properly connected or the access mode used was wront.

8 File connection erro. The file whose logical unit number is given above could not be connected for input/output. Either the file does not exist when it should or vice vise aor the mode of access is wront.

9 The initial derivatives of the state variables could not be computed by the code. This could happen when the inital approximations to the derivatives are not good enough or the derivatives consistent with the initial values given do not exist.

10 The specified number (KMAX) of iterations has been taken without convergence. You may have another go for anothe KMAX number of iterations.

11 The code has taken about 500 steps in the numerical integration towards the communication interval (TOUT) without reaching there. You may have another go for another 500 steps.

12 Incorrect input value(s) were detected which were outside the range of values for these variable(s).

13 Incorrect input. The permissible number of incorrect input trials has been exceeded. Bye.

14 The matrix of partial derivatives is singular. Maybe some of the equations are redunant or there is no solution to this problem or the solution is not unique.

15 The function values could not be computed by the code. Maybe some of the iterates./state variables values will cause a division by zero or overflow or uhderflow, etc.

16 The Jacobian values could not be computed by th4e code. Maybe some of the iterates/state variable values will cuase a division by zero or overlow or underflow etc.

17 Convergence problem. Reoeated error test failures occurred during the last test in the numerical itegration. This may be due to singualarity probnlem of the Jacobian matrix.

18 There was a multiple convergence test failures preceeded by multiple error test failurs on the last attempted step. It is possible that the prioblem is ill-posed or cannot be solved by this code or there is singularity/discontinuity in the prioblem.

19 The function/jacobian values were crudely approximnated for the last step and as a result there was repeated erro test failures for the last attempted step.

20 An error condition was reported but nother was done to remedy it before the code was called gain. You cannot continue the execution in this situation.

21 The value of a variable/parameter could not be locat4ed in a storage location. This may happen when there wront input data was given expecially during topology or variable code numbers input.

22 A dummy subroutine was called/referenced by the code. This option is not yeat available.

23 The code was unable to set up the sparse matrix structure. Maybe some of the module routines do not include this option. The dense matrix structure has been assumed but this may require more work spce than is available.

24 There is error in the sparse Jacobinan matrix sturcture which becam appraent when equation/variable numbers were referenced. You may restart the execution after correcting the mistake or use the dense matrix option.

25 The code has modified a user input value due to wrong input. Further computations make use of the new value.

26 No further merssage included for this type of error.

APPENDIX D

THE MANAGEMENT OF THE DATABASE INFORMATION USING XTRIEVE

D1 Introduction

XTRIEVE is a menu based file management program that enables rapid information management and analysis. It requires BTRIEVE - the record management program. The database is a collection of database files. Each piece of information is stored in a field. The files make up a record. Together, the fields and records make up a file of related information. Information can be retrieved from a database file by using an index. An index is a field in the file that is used to locate a specific entry or record. The information required in fields and the fields used as indexes called the file definition. The file definition includes a description of each field in the file. Each field has a name, a data type and a size. The data type tells XTRIEVE what type of information is stored in a field, such as a number or a name. The data size tells XTRIEVE the length of the data that will be stored in a field. After the database file has been defined, XTRIEVE stores the file definition in a data dictionary.

The data is entered and can be manipulated and displayed in a suitable format. This is done by creating a view for the data. A view is a collection of data that can be built up from one or more files. It allows a convenient mechanism for displaying the ata and its manipulation. However, in DASPII a single view is used for each database file.

D.2 Manipulation of data in DASPII using XTRIEVE

Throughout XTRIEVE, the menu selections are made by moving the cursor using the cursor keys or by entering the first letter of the desired option, followed by a carriage return. An escape key exits to the neu above the current menu.

The following procedure is undertaken to modify data in database files using XTRIEVE:

i) XTRIEVE is located in C:/SOFTW/XTRIEVE and is executed by entering XTRIEVE at the DOX prompt in the above directory.

ii) The program displays the main menu, from which the VIEW option is selected.

iii) The MANAGE option is then selected from the VIEW menu.

iv) The RECALL option is selected from the MANAGE option.

v) A list of view names are displayed which represent the files in the database. The views available for DASPII are shown in Table D1.

XTRIEVE VIEW	DASP Database files
CONTROL OPTIONS	CONTOP.DAT
DASP MODEL TYPES	DPUNIT.DAB
DASP MODELS FOR PFG	PFDMNO.DAB
DATA FOR UNITS	UNITS.DAB
INT PARAMETERS	INTPAR.DAB
MODEL DEFINITION	VARCOD.DAB
MODEL DETAILS	UNITDDF.DAB
MODEL INTEGERS	INTCOD.DAB
PFG UNITS	PFUNIT.DAB
VARIABLE DESCRIPTION	RLVAR.DAB

TABLE D1 XTRIEVE VIEWS
for the database files

When the required view is selected the field headings are displayed.

vi) An <ESCAPE> is used to return to the MANAGE menu, were BROWSE or EDIT is selected to either scan or edit the chosen database file. The file can be edited by moving the cursor between the fields using TABULATE to move forward and a SHIFT TABULATE to move backwards. The new field entry is written over the existing value following by a carriage RETURN. Once the editing session has been completed the program then prompts the user if the editing session is complete or if the changes are not required.

A new record is added by selected ADD in the MANAGE menu and the user enters the field values followed by a carriage return.

D3 The Database Dictionary

The file definition is stored in the dictionary. To review field and index definitions for a file, the following steps are required:

i) The DICTIONARY option is selected from the MAIN menu.

ii) This is followed by the selection of the SHOW option from the DICTIONARY menu - XTRIEVE displays a list of all the files in the dictionary.

iii) The desired file is selected, XTRIEVE displays the filed and index definitions for that file. It also displays the total number of records in the file and the total number of unique values tored in each index.

D4 The XTRIEVE set-up

XTRIEVE is located in C:/SOFTW/XTRIEVE. BTRIEVE the record manage must be executed before the execution of XTRIEVE. As BTRIEVE is a memory resident program is needs to be executated only once during a session. BTRIEVE is located in C:/SOFTW/BRIEVE. It is executed in the above directory by issuing BTRIEVE at the DOS prompt. Once the program is loaded it informs the user that BTRIEVE has been loaded.

After returning to the XTRIEVE directory the user enters XTRIEVE at the DOS prompt. The database files are stored in C:/USERS/DATA. Whilst the view files are stored in C:/USERS/XVIEW.