THE UNIVERSITY OF CALGARY

DYNAMIC SIMULATION OF MULTI-COMPONENT DISTILLATION

by

CRAIG G. MORRIS

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF CHEMICAL ENGINEERING

CALGARY, ALBERTA

FEBRUARY, 1980

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled, "Dynamic Simulation of Multi-component Distillation" submitted by Craig Morris in partial fulfillment of the requirements for the degree of Master of Science in Chemical Engineering.

W. Svrcek

Dr. W.Y. Svrcek (Supervisor)
Department of Chemical Engineering

P.R. Bishnoi

Dr. P.R. Bishnoi
Department of Chemical Engineering

John Slater

Dr. John Slater
Department of Computer Science

C. D. Atherton

Mr. C.D. Atherton
Chevron Canada Limited

J. J. Sparling

Mr. J.J. Sparling
Chevron Canada Limited

May 6/80

Abstract

A program has been developed for the dynamic simulation of large scale chemical processes. The simulator has a highly modular structure and an explicit integration routine which is capable of handling large systems of stiff ordinary differential equations. The use of an explicit method reduces computer storage requirements and greatly simplifies the writing of new modules.

Unit subroutines have been written which are capable of describing multicomponent distillation columns without resorting to conventional thermodynamic equilibrium calculations. The absence of these iterative routines reduces computation time and allows tray efficiencies to be handled in a more natural manner than is possible with Murphree efficiencies. The program also includes an advanced thermodynamic property package based on the Peng Robinson equation of state.

An industrial process consisting of two distillation columns with a total of seventy-five mass transfer stages, has been successfully simulated and the computed results have been compared with dynamic data collected from plant tests.

Acknowledgements

The author wishes to express his gratitude to Dr. W. Y. Svrcek for the guidance and support which made this project possible. Special thanks are also extended to Barry Rubin, Rakesh Mehra, John McRae and the many others who were always willing to offer their valuable ideas and assistance.

I am also particularly grateful to Mr. David Atherton and Mr. Joe Sparling of Chevron Standard Limited for their invaluable assistance in obtaining industrial data.

I would also like to acknowledge the financial support of Home Oil Company through their R.A. Brown Memorial Scholarship.

Table of Contents

List of Figures

List of Tables

Nomenclature

A	constant (used in Chapter 6 as area)
Ax	crossectional area
b	constant
B	constant
c	constant
Cp	heat capacity
d	constant
D	diameter
E	energy
Em	Murphree tray efficiency
$\bar{f}$	partial molar fugacity
h	time step
H	enthalpy
h_{ow}	height of liquid over the weir
h_{weir}	height of the weir
HTC	hydrauic time constant
$\bar{I}$	property package input vector
$\bar{J}$	Jacobian matrix
k	constant
K	constant (used in Chapter 6 as a mass transfer coefficient)
K_I	integral controller constant
K_P	proportional controller constant
L	molar liquid flow

m	constant
Mp	bubbles per unit time
N	moles
nc	number of components
$\overline{O}$	property package output vector
P	absolute pressure
P_c	absolute critical pressure
Q	heat flow
R	ideal gas constant
S	Richard Lanning Torrey method correction factor
t	time
T	absolute temperature
T_c	absolute critical temperature
t_c	overall bubble time constant
v	number of derivative evaluations
V	volume
w	acentric factor
W	weighting factor
X	liquid mole fraction
y	dependent variable
Y	vapour mole fraction
α	Peng Robinson constant
Θ	angle between derivative vectors
ℓ	molar vapour flow rate
ρ	molar density

ϕ_i constant used in approximation of fugacity

Subscripts and Superscripts

i index (used in Chapters 5 and 6 to denote component number)

j index

L liquid

M denotes Modified Euler terms

n index (used to denote time level in Chapter 3 and tray number in Chapter 6)

P bubble

o initial condition

′ time derivative

* ideal state

CHAPTER 1

1.1 Introduction

In recent years increased environmental consciousness and rapidly rising energy costs have resulted in a dramatic increase in the use of sophisticated control systems in the fluid processing industry. These complex systems can be difficult and costly to design and tune, especially when the dynamic behaviour of the process to be controlled is not well understood. Researchers have addressed themselves to this problem by developing a wide variety of dynamic simulation methods for fluid processes.

1.2 Dynamic Simulation

For the purposes of this thesis, dynamic simulation shall refer to the modelling of time varying physical processes. This involves the numerical integration of a system of differential equations:

$$\bar{y} = \int_{t_o}^{t} f(\bar{y}, t)\, dt \qquad (1.1)$$

A dynamic process simulator is a computer program which provides the framework and the numerical routines necessary to solve systems such as equation 1.1. Equation oriented simulators require that the entire system of algebraic and differential equations be provided in the program data set (Table 1.1). Modular simulators contain libraries of small specialized models which can be called and interconnected by means of the input data to form the desired overall system. A chemical engineering simulator would typically contain library routines for heat exchangers, tanks, reactors and other unit processes which could be assembled by the user to form a simulation of an entire plant. This thesis deals with the use of this type of simulator in the study of hydrocarbon distillation.

1.3 Project Rationale and Scope

The relatively primitive methods that have been used in dynamic simulators to evaluate thermodynamic properties, must surely have a detrimental effect on their results. This is particularly true for those processes which involve mass transfer. The light hydrocarbon industry is fortunate to have at its disposal, modern equations of state which very accurately predict multicomponent properties over all ranges of industrial interest. These property calculations have already been implemented in steady state programs

(Shah,1975), so it seemed to be a natural step to test their feasibility in a dynamic simulation program.

The initial work was done using the DYNSYS 2.0 simulation program and conventional distillation models, but the complexity of the property calculations and the size of the simulation necessary to simulate a full scale industrial problem made this approach untenable.

In succeeding phases of the work the DYNSYS 2.0 program was modified radically, new integration routines were introduced and a novel concept for distillation modelling was developed. The final phase of the work involved comparing the model to results taken from tests at an industrial plant.

CHAPTER 2

2.1 Modular Dynamic Simulation

Since the late sixties a large number of modular dynamic simulators have been developed for use in the fluid processing industry. A list of the more well known of these would include:

DYFLO - (Franks,1972)

DYNSYS 2.0 - (Barney,1975)

OSUSIM - (Koenig,1972)

REMUS - (Ham,1971)

PRODYC - (Ingels,1970)

The first two of these had by far the greatest impact on this project and while the others all have their own unique features (for instance PRODYC is written as a subset of the equation oriented simulation language, CSMP, and OSUSIM makes elaborate provisions for the solution of the steady state case and implicit recycle loops) they are all at least similiar in concept to DYNSYS. For this reason only DYFLO and DYNSYS will be discussed in detail.

2.2 DYFLO

This package consists of a collection of FORTRAN subroutines which provide the user with various process models, fluid property calculations, numerical integration methods and simple output routines. The user must write his own input and executive programs which call library routines in the appropriate order and with the appropriate arguments to create the simulation. This is in distinct contrast to other simulators which use fairly elaborate executive routines to read the input data and from that construct the calling sequence for the modules.

DYFLO communicates between routines principally by means of references to common block variables. The main process variable is the stream matrix, STRM(I,J), where each I represents a different stream of material and information flowing between process units. Typically a stream would have compositions, temperature, pressure and perhaps enthalpy stored as its J variables. When a process subroutine is called its header list contains the identification numbers of the streams flowing in and out of it, as well as any unit parameters such as vessel capacity, which might be needed. It is this stream and node concept which allows large simulations to be easily constructed from elementary models. It is a fundamental concept which appears in one

form or another in all modular simulators.

DYFLO only offers first, second and fourth order Runge Kutta integration methods and all of these have fixed time steps and no error estimates. The simplicity of these integration methods allows modules to be written in a very straightforward manner. The basic integration operation only requires the statement:

```
CALL INT(X,DX)
```

where DX is the calculated derivative and X is the solution value returned. A second routine INTI is called from the executive after all the derivative evaluations for a single time step have taken place, that is after all calls to INT have been completed. INTI must also be called at the beginning of the integration to establish the range of integration, the step size, the order of the integration and other such parameters. A flag is passed to the modules to indicate the first pass, which is done without integration to allow initialization of variables. An interesting feature of this integration routine is that it can be used independently of the rest of the simulator.

Although the primitive DYFLO system might not seem as desirable as the more elaborate programs available, this author can attest to the value of its great flexibility and simplicity. The flexibility stems from its very lack of a rigid executive and from the relatively independent manner in which its routines may be used. In as much as possible Franks has attempted to allow routines or groups of routines to be used independently of the rest of the package. As a result any engineer familiar with FORTRAN would have little difficulty in performing a simulation with DYFLO since there are few rules to remember other than basic FORTRAN. If, as is often the case, the user wishes to create his own modules, the explicit integration routines permit the modules to be written in a simple and brief manner.

A disadvantage of the method is that for even small changes in the process, it is usually necessary to change some of the code and recompile the program. Perhaps a more serious difficulty is the inability of its unsophisticated integration methods to cope with the stiff differential systems which will be discussed in Chapter 3.

2.3 DYNSYS 2.0

DYNSYS was first introduced in 1970 by A.I. Johnson and associates at the University of Western Ontario. In 1975 Barney incorporated the sophisticated Gear's integration package for stiff systems into the program and appended the 2.0 suffix to the name. Since that time it has been used in several research projects (Millares,1975; Pulido,1975; Hui,1977) with varying degrees of success.

Unlike DYFLO, the DYNSYS simulator has a sophisticated executive routine which controls the reading of data, the program flow and the printing of results. This requires a slightly more complex data structure to allow for the storage of the plant configuration and the necessary equipment parameters (Figure 2.1). The S(K,I,J) matrix is similiar to DYFLO's STRM matrix but it has another dimension to allow the stream variable from the previous time step to be stored. This is necessary to accomodate the predictor-corrector integration methods that DYNSYS uses. The plant structure is stored in the MP (Module Parameter) matrix. Each row of the matrix contains a unit identification number, a unit type number and a list of the streams flowing in and out of the unit. DYNSYS uses the convention that negative stream numbers indicate outflowing streams. The last parameter is an index to a block of extra storage,

EX(I), which can be used by those modules which require large amounts of input. The EP (Equipment Parameter) matrix is used to store input information such as vessel capacities or heat transfer coefficients that might be required in the module calculations.

During execution the program marches down the MP matrix row by row calling the routines designated by the type number. A parameter indicating which row of the MP and EP matrices to reference, is passed to the module by means of a common block so that the same routine may be called several times per time step in different contexts.

The 2.0 version of DYNSYS used the Gears technique for its numerical integration. While this has proved to be a powerful tool for the solution of stiff systems of differential equations, it complicates module writing and makes debugging difficult. The suitability of this technique for modular simulators is discussed further in the next chapter.

2.4 CRITERIA FOR A DYNAMIC SIMULATOR

When in the course of the project it became clear that DYNSYS would have to be extensively modified, it was decided to set forth certain criteria which would form a basis for evaluating the merits of proposed changes to the program:

1) In accordance with the nature of chemical plant simulations, the program must be able to handle large and possibly stiff systems efficiently.

2) In as much as possible the modular approach should be carried over into the executive. It should be relatively easy for the user to replace any routine with his own version to suit a particular circumstance. Input and output routines are particularly good examples of where this is desirable.

3) In the same spirit as above, all variables included in a common block should have some common purpose (i.e. all property values should be in "property" common blocks). This reduces the number of common blocks that individual routines need to reference.

4) Most applications of a dynamic simulator require the creation of at least some specialized modules. It is therefore important that new modules can be written and debugged in a straightforward and easy manner. Only a elementary knowledge of numerical methods should be necessary.

5) Modules should represent relatively elementary

processes. For instance, in the distillation problem which concerns this study, modules which represent individual trays rather than entire columns make better use of the advantages of the modular approach.

How closely these criteria can be met depends to large degree on the requirements of the numerical integration routine. Therefore discussion of the program structure will be deferred until numerical integration has been considered.

CHAPTER 3

The Numerical solution of Differential Equations

This discussion will be limited to the solution of first order ordinary nonlinear differential equations which have defined initial values. Partial differential equations which arise from process simulations are usually transformed into nonlinear ordinary differential equations by finite difference approximations. Higher order equations may always be rewritten as a system of first order equations and boundary value problems require special iterative methods which will not be considered. Despite these limitations the subject remains too broad to be covered in any depth in this thesis, so only a brief review of terms and a consideration of the special problems of stiff systems and the methods for dealing with them will be given.

3.1 Single Step Methods

These methods do not require any information prior to (t_n, y_n) to calculate y. The general single step equation is (Barney, 1975):

$$y_{n+1} = y_n + \sum_{i=1}^{v} W_i K_i \qquad (3.1)$$

where

$$K_i = hf(t_n + c_i h,\ y_n + \sum_{j=1}^{i-1} d_{ij} K_j)$$

When v, the number of derivative evaluations, is equal to one, equation 3.1 reduces to the simple Euler method:

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{3.2}$$

This is also the simplest method in the widely used family of Runge Kutta techniques. Typical of the higher order methods of this type is the fourth order Runge Kutta:

$$y_{n+1} = y_n + \frac{K_1 + 2K_2 + 2K_3 + K_4}{6} \tag{3.3}$$

where

$$K_1 = hf(t_n, y_n)$$

$$K_2 = hf(t_n + \frac{h}{2},\ y_n + \frac{K_1}{2})$$

$$K_3 = hf(t_n + \frac{h}{2},\ y_n + \frac{K_2}{2})$$

$$K_4 = hf(t_n + h,\ y_n + K_3)$$

3.2 Multi Step Methods

These methods require information from points preceding (t_n, y_n) in order to calculate y_{n+1}. Usually this information is used to generate the coefficients for an interpolating polynomial which is used in approximating the next point. The general formula is (Burden,1978):

$$y_{n+1} = \sum_{i=1}^{k} a_i y_{n+1-i} + h \sum_{i=0}^{k} B_i f(t_{n+1-i}, y_{n+1-i}) \qquad (3.4)$$

where a_i and B_i are constants.

A minor disadvantage of multi step methods is that they are not self starting, and therefore require auxiliary methods to generate the first n+1-k points.

3.3 Implicit and Explicit Methods

When the value of B_o in equation 3.4 is not equal to zero y_{n+1} appears on both sides of the equation and in general can only be solved for iteratively. Such methods are termed implicit while methods which allow y_{n+1} to be determined directly from previously calculated values are called explicit.

When implicit systems of equations must be solved some variation of the Newton Raphson iterative technique is frequently used. Although this usually results in rapid convergence, it requires that the Jacobian matrix of partial differentials be calculated. For example the three equation

system:

$$y'_1 = f_1$$

$$y'_2 = f_2$$

$$y'_3 = f_3 \qquad (3.5)$$

has as its Jacobian matrix:

$$\underline{J} = \begin{matrix} \frac{\delta f_1}{\delta y_1} & \frac{\delta f_1}{\delta y_2} & \frac{\delta f_1}{\delta y_3} \\ \frac{\delta f_2}{\delta y_1} & \frac{\delta f_2}{\delta y_2} & \frac{\delta f_2}{\delta y_3} \\ \frac{\delta f_3}{\delta y_1} & \frac{\delta f_3}{\delta y_2} & \frac{\delta f_3}{\delta y_3} \end{matrix} \qquad (3.6)$$

In large and complex systems evaluating this Jacobian can be a formidable task.

3.4 Predictor-Corrector Methods

Implicit methods require some estimate of the y_{n+1} values before their formulas can be applied. Normally an explicit formula is used to predict y_{n+1} while the implicit formula is used to correct this estimate; hence the name predictor-corrector. The number of corrector iterations depends on the method being used and the properties of the system to be solved. In simple systems it is common to take

only a single corrector pass, while in systems that tend to be unstable the corrector may be iterated on until it converges to within some error tolerance.

3.5 Accuracy

Virtually all numerical integration techniques are based either directly or indirectly on a Taylor's series expansion:

$$f(t) = f(t_o) + f'(t_o)(t-t_o) + \frac{f''(t_o)(t-t_o)^2}{2} + \ldots$$

$$\ldots + \frac{f^n(t_o)(t-t_o)^n}{n!} + R_n(t) \qquad (3.7)$$

where $R_n(t)$ represents the error which is incurred by truncating the n+1 terms. A numerical integration technique is said to be nth order if in a single step from t_o to t it generates a truncation error no larger than a nth order Taylor series would. (i.e. no larger than $R_n(t)$) Since this error is generated in a single time step it is called the local truncation error (L.T.E.), while the overall deviation from the true solution is called the global truncation error (G.T.E.). The global error is usually very difficult or impossible to numerically estimate, but for many types of integration methods the local truncation error is roughly proportional to $h^{1/n}$ where n is the order of the integra-

tion. This suggests that even the simple Euler's method could provide any desired accuracy if a small enough time step could be used. Unfortunately the use of very small time steps can result in round off errors during computations, but this is not a common problem in chemical engineering applications.

The strong dependence of error on integration order usually allows higher order methods to use much larger time steps; so much larger in fact that for many applications the higher order techniques are much more efficient than low order methods despite the greater amount of computation needed on each step.

3.5 Variable Step Sizes

If a simulation is to be useful there must be some method of controlling the error. The simplest approach is to perform the entire simulation with a fixed time step and then repeat it with the step size halved. If the user feels that the results for the two runs are significantly different, the step size is reduced and the procedure repeated.

Franks (1972) makes a strong argument for the utility of this method in engineering calculations. He claims that it is more practical to have an engineer exercise his judgement about what constitutes an acceptable error than to attempt to quantify this criteria in a program. He presents Figure 3.1 as an interesting example of where human judgement may be preferred to machine error control.

Despite the merits of this argument, the procedure is tedious since in complex simulations considerable time is spent comparing results and making repetitive runs. Therefore most advanced integration techniques have built in methods of estimating the local truncation error at each time step. If this error exceeds some absolute or relative maximum, the step size is reduced and the step is repeated. Besides convenience, automatic error control has the advantage of only taking small time steps when needed, rather than using the smallest necessary time step for the entire simulation.

3.6 Stiffness and Stability

A stiff differential equation is one which has an exact solution which contains a function of the form e^{at}. When "a" is large and negative the exponential part of the function decays rapidly to zero and as a result the numerical

integration method must use a very small time step in order to follow the solution. Consider the simplest stiff equation:

$$y' = ay$$

$$y(0) = b \tag{3.8}$$

Using Euler's method to integrate this results in the difference equation:

$$y_{i+1} = y_i + hy_i$$

$$= (1 + ha)y_i \tag{3.9}$$

so that

$$y_m = b(1 + ha)^m \tag{3.10}$$

When the sign of a is negative and the magnitude of h is much smaller than $|1/a|$, the y_m values will closely approximate the true solution:

$$y = b(e^{at}) \tag{3.11}$$

As the step size h is increased the accuracy of the approximation will decrease until $h = |1/a|$ is used and all resulting y_m will equal zero at times greater than zero. At larger values of h the sign of y_m will change on each step, but as long as h is less than $|2a|$ the solution y_m will

decay to zero at large values of m. If however h is larger than $|2a|$ the magnitude of y_m will grow exponentially while its sign continues to oscillate on each step.

If equation 3.8 was the only component in the system being integrated, its stiffness would cause no problem, since the very small step size required to approximate the curve accurately would be balanced by the very small range of integration necessary to follow the solution to steady state. However systems having more than one exponential term in their solutions can create difficulties when they have time constants with considerably different values. Even if the accuracy of the components with the small time constants is not important, the step size must be kept small enough to prevent errors in the approximation of these components from growing so large that they dominate the solution. Unfortunately the range of integration is usually determined by the components with large time constants since these require the longest time to approach steady state. The result is that a great many time steps are required to integrate the function and this of course requires excessive computer time. It is usually this situation which is referred to when the term "stiff system" is used.

It may be of benefit to consider a simple example which illustrates how stiffness can arise in process systems. The holdup of the two tanks depicted in Figure 3.2 can be described by the following differential equations:

$$\frac{dV_1}{dt} = F_1 - F_2$$

$$= F_1 - K_1 h_1$$

$$= F_1 - \frac{K_1 V_1}{Ax_1} \tag{3.12}$$

$$\frac{dV_2}{dt} = \frac{K_1 V_1}{Ax_1} - \frac{K_2 V_2}{Ax_2} \tag{3.13}$$

Let us assume the following initial values:

$$V_1 = 900$$

$$K_1 = 1000$$

$$Ax_1 = 1000$$

$$V_2 = 1$$

$$K_2 = 1000$$

$$Ax_2 = 1$$

$$F_1 = 1000$$

Then

$$\frac{dV_1}{dt} = 100$$

$$\frac{dV_2}{dt} = -100$$

If an Euler step of 0.1 was used, then after the first step: t=0.1, V_1=910 and V_2=-9. In order to avoid such physically unreasonable results a step size of about 0.001 would be required, but then V_1 would creep towards its final value very slowly. It is very possible that the rapidly varying volume of the small tank is of little interest in the overall simulation. It could for instance simply represent a vertical section of pipe in the line leading away from the large tank. In this simple case a solution would simply be to use a less detailed model which did not contain the second tank.

A somewhat less drastic approach is to assume that since the small tank approaches equilibrium rapidly, it may be considered to always be effectively at steady state. Equation 3.13 then reduces to

$$0 = \frac{K_1 V_1}{Ax_1} - \frac{K_2 V_2}{Ax_2}$$

$$V_2 = \frac{K_1 Ax_2}{K_2 Ax_1} V_1 \qquad (3.14)$$

This approach, termed the pseudo steady state method, is

very effective but it can generate incorrect results unless it is used with considerable care.

3.7 Stiff Integration Techniques

Over the last dozen years there has been a great proliferation of integration methods intended for use with stiff systems. These techniques have varied considerably in their effectiveness, practicality and flexibility. Many methods work well under certain circumstances but are inefficient or fail in other cases. As a result there has not been, to this author's knowledge, a comprehensive testing of a wide variety of stiff integration techniques over a broad range of conditions. In particular, no testing has been reported on large (>100 differential equations) stiff systems. However there have been several smaller testing programs and in the face of the great variety of techniques available, it has been necessary to limit consideration to those techniques included in these surveys. In view of the rapid advances in this field, only those comparisons done since 1970 have been considered.

3.7.1 Testing of Stiff Techniques

3.7.1.1 Seinfeld, Lapidus and Hwang (1970)
Lapidus and Seinfeld (1971)

Tests were done on four systems having from one to three differential equations and ranging from moderately to very stiff. Their conclusions were:

i) 4th Order Runge Kutta

Adams Predictor-Corrector

These methods were very accurate but were inefficient at handling stiff systems.

ii) Midpoint Method

This performed poorly.

iii) Treanor's Method (Treanor, 1966)

This stiff explicit method was usually, but not always, faster than the Runge Kutta routine. They felt it was only suitable for systems that had diagonally dominant Jacobians.

iv) Trapezoidal Rule

Calahan's Method (Calahan, 1968)

Liniger Willoughby Methods (Liniger, 1967)

These implicit methods were found to be superior

to the explicit methods in both efficiency and accuracy. The two methods by Liniger and Willoughby were preferred for very stiff systems. A trapezoidal rule with extrapolation was also tested but was found to be very inefficient.

3.7.1.2 Brandon (1974)

Brandon compared his own method with the methods tested by Lapidus and Seinfeld and found his method to be superior. He used two test examples.

3.7.1.3 Hronsky and Martens (1973)

Five tests were conducted on systems having from two to seven differential equations. Their results:

i) Runge Kutta Merson (Merson, 1957)

This was their choice for nonstiff systems.

ii) Runge Kutta Newton

This was fast, but since it is a form of the psuedo steady state approach, it requires manipulations of the original equations.

iii) Treanor's Method (Treanor, 1966)

This was their choice for large stiff systems since it did not need large Jacobian matrices.

iv) Liniger Willoughby (Liniger, 1967)

This was their choice for small stiff systems where the Runge Kutta Newton method was not applicable.

v) Backward Difference Formula (Brayton, 1972)

This is a modification of Gear's method by Brayton. Hronsky and Martens found it to be reliable, but inefficient.

3.7.1.4 Enright and Hull (1976)

These authors have been active in the field of integration technique testing for several years and have developed quite elaborate testing procedures. In this most recent paper on stiff techniques, they tested three backward difference methods, the original Gear's routine DIFSUB (Gear, 1971), a revised Gear's routine GEAR.REV3 (Hindmarsh, 1974), and EPISODE (Byrne, 1975), as well as a generalized Runge Kutta by Lawson (1972), the trapezoidal rule with extrapolation and a variable order second derivative multistep method called SDBASIC (Enright, 1974).

Only the backwards difference formulas and SDBASIC were considered to give acceptable performances. GEAR.REV3 was considered the best of these by a small margin, while SDBASIC was considerably slower than the others. However it was better than backwards difference formulas at solving problems which have eigenvalues of the Jacobian matrix close to the imaginary axis.

3.7.1.5 Barney (1975)

In the course of performing his revisions to DYNSYS, Barney tested a wide variety of integration methods on eleven different problems. The results of his testing are reproduced in Table 3.1. The different variations on Gear's routine resulted from implementing the basic method with various types of matrix solvers.

The GEAR-TRGB combination appeared to be the best overall routine for the systems tested, but the explicit Richard Lanning Torrey method (Richards et al, 1966) performed surprisingly well despite its tendency to give rather large errors in the small components. The Runge Kutta Merson method was clearly superior to all others in the limited testing of non-stiff systems (tests VIII - XI)

3.8 Integration Methods for Modular Simulators

The tests reviewed in the previous section suggest that the implicit backwards difference integration methods such as the Gear routines, are the best choice for stiff systems. On this basis Barney chose to install the GEAR-TRGB package in the DYNSYS 2.0 simulator and was able to demonstrate its power on a number of problems. However it is this author's opinion that the rather special nature of modular simulation was not adequately reflected in either the surveys or in Barney's choice of integration method.

The key difficulty in using an implicit method such as Gear's, is that it must be supplied with the system Jacobian matrix if it is to operate efficiently on stiff systems. In a modular simulator it is not possible to supply an analytical Jacobian for the entire system since the configuration of the problem is only established at execution time by the arrangement of the various modules. Numerical evaluation of the Jacobian would only be practical for very small systems.

Barney attempted to circumvent this difficulty by considering each module as an independent system for the purpose of corrector iteration. Unfortunately this scheme is only effective if stiff components do not arise from the interaction of modules. The simple tank problem mentioned

earlier could not be handled in this manner because the stiffness results from interactions between the tanks. Therefore the entire system would have to be written as a single module. This puts considerable limitations on the manner in which modules can be written. For the problem at hand it is desirable to construct the basic module as a mass transfer stage from which distillation, absorption and stripping columns of any configuration could be built. With DYNSYS 2.0 efficient operation could only be achieved if the entire tower, possibly including its control system, was written as a single module. Even when large self contained modules are written it is by no means obvious, especially to the casual user, when module interactions will occur which will lead to system stiffness and the resulting failure of the integration. The net result is a considerable loss in the inherent flexibility and simplicity of the modular system.

It is also of significance that the largest system used in the comparison of integration methods had only 33 equations. Very large systems can occur in process simulation (examples in this thesis will have over 550 ordinary non-linear differential equations), so the relative performance of a method as the system size increases is of considerable importance. All the implicit routines have matrix calcula-

tions whose overhead increases with the cube of the number of derivative evaluations. Since the overhead of explicit methods generally increases only linearly with system size, the performance of these techniques may improve relative to implicit methods in large simulations. However these gains may be offset by the great complexity of the derivative evaluations in process simulators which penalizes the small step size explicit methods much more than the implicit methods.

An example of the type of difficulty that can be experienced can be found in the work of Pulido (1975) in which DYNSYS 2.0 was used to reproduce a simulation of a simple distillation which Franks (1972) used as an example for the DYFLO simulator. Pulido reported an execution time of 4 minutes and 48 seconds for this simulation while on runs by this author on the same type of machine (a CDC Cyber 172) the DYFLO simulator required less than 15 seconds. While this example is not very stiff (it was not even necessary to invoke Franks' psuedo steady state assumptions), it does suggest that difficulties can be encountered with the DYNSYS 2.0 approach. It should be noted that in Pulido's work all the trays except the reboiler and condenser were represented by a single module.

The Gear system also makes the writing of modules considerably more complex in DYNSYS 2.0 than it is in DYFLO. The DYNSYS user must make provisions for supplying the module Jacobian, for doing different calculations on the predictor and corrector passes and for identifying corrector convergence. While none of this is particularly difficult, it does add considerable opportunity for error which coupled with the vagueness of the integrator error messages makes debugging very difficult. The result has been comments by many users (Pulido, 1975; Hui, 1977; Bush, 1978) on the high levels of overhead involved in the creation of new modules.

In light of these difficulties it was decided the use of Gear's method was not suitable for the current project and that alternate techniques would have to be found. To aid in the selection of these techniques the following criteria were established:

1) The routine should be simple to use and if possible should achieve the simplicity and flexibility of the DYFLO system.

2) There should be no requirement for Jacobian evaluations. This essentially eliminates implicit methods.

3) Since process models themselves, by the nature of the assumptions made in their formulation, are not extremely accurate, there is little benefit to be gained from highly accurate integration routines. This suggests lower order routines are probably satisfactory for most process simulations. However simplicity of operation would be enhanced by a system which retains automatic step sizing.

4) The method should be able to handle a wide variety of problems without failing. When a failure does occur sufficient information should be output to allow a module writer to determine where the problem lies.

5) The program must run in a reasonable amount of time for the given problem.

3.9 The Choice of Integration Methods

Based on the above criteria the choice of a method for nonstiff systems was obvious. The Runge Kutta Merson had been clearly superior in the tests surveyed and it was simple to use and program. It was also one of the first Runge Kutta methods to incorporate an effective approximation of the local truncation error. The method is a fourth order Runge Kutta with a fifth step being taken to determine the

error. The difference equations describing it are:

$$K_1 = hf(t_n , y_n)$$

$$K_2 = hf(t_n + \frac{h}{3} , y_n + \frac{K_1}{3})$$

$$K_3 = hf(t_n + \frac{h}{3} , y_n + \frac{K_1}{6} + \frac{K_2}{6})$$

$$K_4 = hf(t_n + \frac{h}{2} , y_n + \frac{k_1}{8} + \frac{3K_2}{8})$$

$$K_5 = hf(t_n + h , y_n + \frac{K_1}{2} - \frac{3K_3}{2} + 2K_4)$$

$$y_{n+1} = y_n + \frac{K_1 + 4K_4 + K_5}{6} \qquad (3.15)$$

where the local truncation error is estimated from:

$$L.T.E. = \frac{K_1 - 4.5K_3 + 4K_4 - 0.5K_5}{15} \qquad (3.16)$$

If the largest error is greater than the allowable error the step size is halved and the step is repeated. If the largest error is less than one tenth the allowable error the next step size is calculated from:

$$h_{new} = \frac{h_{old}}{1.1} \left(\frac{\text{allowable error}}{\text{largest error}}\right)^{\frac{1}{4}} \qquad (3.17)$$

The selection of a stiff technique was not as obvious. The first choice was to simply use the psuedo steady state technique in a manner similiar to Franks. Although this appeared to work well, there has been criticism of the method (Emanuel, 1967; Kolbrack, 1967; Snow, 1966) in which it has been pointed out that unless caution is used in making the assumptions serious errors can result. Since these errors may not be immediately obvious to the user, there is a certain risk involved in the use of the method. Also the criterium of simplicity in module writing is violated since some "juggling" must be done to determine when a component is becoming stiff.

The most promising alternative was the explicit Richard Lanning Torrey method (RLT) which had performed so well in Barney's tests. However this had no direct way of controlling truncation error, and while the main components of the simulation always seemed to remain fairly accurate, there was a certain uneasiness associated with its use. Nevertheless it was decided to incorporate it into the simulator and attempt to add a truncation error control.

The empirical observation that instability in numerical integration is accompanied by sudden reversals in the direction of the derivative vector, is used in the RLT technique to predict the onset of instability in a simple Euler's method. This is done by simply taking the dot product of the current and last valid derivative vectors. The cosine of the angle between the two vectors is calculated from simple trigonometry:

$$\cos\Theta = \frac{\bar{f}'_n \cdot \bar{f}'_{n+1}}{||\bar{f}'_n|| \cdot ||\bar{f}'_{n+1}||} \tag{3.18}$$

If this cosine is less than -1/8 the step is repeated with a new time step and solution vector calculated from:

$$h_{new} = h_{old} \times S$$

$$y^{new}_{n+1} = y_n + S(y^{old}_{n+1} - y_n)$$

$$S = \frac{\bar{f}_n \cdot (\bar{f}'_n - \bar{f}'_{n+1})}{||\bar{f}'_{n+1} - \bar{f}'_n||^2} \tag{3.19}$$

When the stability criterion is met a simple Euler step is taken with a step size equal to:

$$h = \text{beta} \times \frac{||\bar{y}_n||}{||\bar{f}'_n||} \tag{3.20}$$

where the n subscripts indicate the new valid values (i.e.

the n+1 values used in the stability calculation) Beta is a tunable parameter but 0.01 proved to be a satisfactory value for all the tests conducted.

Since the method is essentially Euler's method, there was little difficulty in adding an error estimation algorithm to it. The Euler method

$$y_{n+1} = y_n + hf(t_n , y_n) \tag{3.21}$$

is first order correct and so can be assumed to generate second order errors. The second order Modified Euler method:

$$y^M_{n+1} = y^M_n + \frac{h[f(t_n , y^M_n) + f(t_{n+1} , y^M_n + hf(t_n , y^M_n))]}{2} \tag{3.22}$$

only generates third order errors. The difference between the two methods will be approximately equal to the second order errors generated by the simple Euler method:

$$\begin{aligned} err &= |y^M_{n+1} - y_{n+1}| \\ &= |y^M_n + \frac{h[f(t_n , y^M_n) + f(t_{n+1} , y^M_n + hf(t_n , y^M_n))]}{2} \\ &\qquad - y_n - hf(t_n , y_n)| \end{aligned} \tag{3.23}$$

Assuming

$$y_n^M = y_n \tag{3.24}$$

then

$$err = \frac{h|f(t_{n+1}\ ,\ y_n + hf(t_n\ ,\ y_n)) - f(t_n\ ,\ y_n)|}{2} \tag{3.25}$$

Since the first term inside the absolute value operator is simply the derivative resulting from the last Euler estimate and thus the derivative to be used in the next Euler calculation, no additional function evaluations are necessary to obtain the error estimates. More detailed discussions of Euler error estimates may be found in Burden et al (1978) on pages 249 through 251 and in Franks (1972) on pages 49 and 50.

The largest calculated error is used in the following equation to estimate a new step size:

$$h_{new} = h_{old}\left(\frac{\text{error allowed}}{2 \text{ x largest error}}\right)^{\frac{1}{2}} \tag{3.26}$$

If the allowable error has been exceeded the step is repeated with this new step size, otherwise the minimum of this step size or the one predicted in equation 3.20 is used for the next step. If the stability criterion is not met the entire estimation is skipped.

A failing of the original RLT algorithm was the excessive execution times that resulted when a simulation approached steady state. The derivative vector would by definition become very small, until its value was mainly comprised of small random inaccuracies generated by the model calculations. As a result the direction of the derivative vector changed more or less randomly from step to step, which the routine perceived as the onset of instability and so reduced step sizes accordingly. The solution to this problem was to simply ignore the stability calculation if the norm of the derivative vector was less than an arbitrary value calculated from 1/2 beta times the solution vector norm. For situations where this permits too much steady state noise the value of beta could be reduced.

In addition to the RKM and RLT routines a simple Euler method was included for debugging purposes and in case it was required for use with a psuedo steady state approach.

3.10 Implementation of the Integration Package

In both DYFLO and DYNSYS the modules communicate with the integration methods by means of subroutine calls which pass the derivatives to the integrator and receive solutions by means of the header list. Even with the explicit DYFLO routines the placement of these call statements in the

module can be slightly confusing. Since the RLT routine cannot provide new estimates for the solution variables until all the derivative evaluations for the particular time step have been completed a more sensible and convenient method is to call the integration routine from the executive once per time step and let the modules communicate with it by means of a common block. In the new integration package the DIF common block was created to contain all the integration variables that a module would normally require access to. The variables are:

JSTART - a flag which has a value of 0 on the first pass only.

TIME - the independent variable.

H - the current step size.

NDEQ - the number of differential equations already evaluated on the current time step.

Y(600) - the solution vector array.

DY(600) - the derivative vector array.

A routine using the integration package would calculate the DY values from supplied values of Y, TIME and perhaps H. If more than one subroutine is calculating derivatives (as is the case in the process simulator) NDEQ is used as an index to the arrays. In any event, NDEQ is incremented by

the number of differential equations that the routine added to the system. The JSTART flag allows the routines to assign initial values to the Y vector on the first pass.

A second common block, DIF2, contains the variables which the executive routine needs to control the integration:

TMAX - finishing point for the integration.
(default = 1.0)

IEND - flag which is set to one when the integration is finished.
(default = 0)

ICC - when set to 0 this flag results in two derivative evaluations being done on each step before the integration is attempted.
(default = 0)

ICONV - flag set to 1 when integration step has failed and is being repeated with a smaller step size.
(default = 0)

HMIN - the smallest allowable step size.
(default = 10^{-10})

HMAX - the largest allowable step size.
(default = 0.1)

BETA - RLT tuning parameter.

(default = 0.01)

EPS - largest allowable local truncation error. (default = 0.001; on an absolute basis)

IORDER - selects integration method

1 = Euler

2 = RKM

3= RLT

(default = 3)

TPR - this is the interval between outputs. At the begining of the run and at every interval TPR after that the integration routine calls a subroutine called OUTPUT. Step sizes are adjusted to achieve proper matching with the output times.

MM - this is a counter which is used when the ICC flag is 0. It has a value of 1 on the first pass of the two pass series and a value of 0 on the second pass.

A key requirement of the RLT routine is that the values of the derivative vector depend solely on the independent variable and the solution vector values. It is particularly important that the derivative values not be a function of previously attempted steps. This can occasionally cause problems with simulations where modules react counter-

currently.

Consider the two units in Figure 3.3. It may be assumed that all streams entering or leaving a module influence the calculation of that modules derivatives, and in addition that the integrated values of those derivatives affect the values of all streams leaving the module. Thus if module A is executed first, the values in the stream going from B to A will represent the values of the previous, perhaps invalid, integration step. With the ICC parameter set to 0 this problem is avoided since an extra pass is made to ensure all streams reflect the new solution vector values. If desired, some modules could use the IDIF value to skip the calculation of derivatives on the first pass and the interpretation of the solution vector on the second. Although this approach is somewhat crude, it decreased run times on the stiff distillation problems studied in this project and eliminated the occasional failures of the integration routine caused by this problem.

A similiar difficulty can arise with the numerical differentiations in the calculations. Often the error produced by these calculations will increase significantly as the integration routine reduces the step size in order to achieve stability or error limits. In addition a discon-

tinuity in the solution (such as introducing a step change to some variable) creates havoc with numerical derivatives. If possible numerical derivatives should be avoided, but when this is not possible they should be calculated with special filtering routines to minimize the problems they create. Despite these potential problems, no trouble was experienced in using the RLT routine on DYFLO simulations containing crude numerical differentiations.

Since the error tolerance used in the integration package is an absolute error limit it is the users responsibiliy to scale the derivatives and solution vector variables appropriately. This usually consists of normalizing variables much larger than 1. This also has an effect on the RLT stability calculation since large elements will dominate the vector directions.

Both the RKM and the RLT routines will stop with a message should they be unable to achieve the desired tolerance at the minimum step size. With the RKM method this could be indicative of instabilities induced by a stiff system, but with the small minimum time steps used with the RLT method it usually means there is an error in the problem formulation or that some variable is improperly scaled. The RLT routine will also fail if it cannot meet its stability

requirements. This is invariably caused by the failure of the calculation to return the last valid set of derivatives as the step size is reduced. Except in the case of an extremely stiff system this indicates inconsistencies in the calculation. To aid in finding these problems the RLT routine prints out the derivative and solution vectors of the last valid step and the last (fatal) attempted step whenever it fails.

CHAPTER 4

4.1 Program Structure

The integration routine discussed in the previous chapter permitted the developement of a generalized integration package which accepts the simulation problem in the form of specially named subroutines. Each of these subroutines has an intended function which is relatively independent of the rest of the simulation:

DATA - an input and initialization routine which is executed only at the beginning of the simulation.

EXEC - a derivative evaluation routine which calculates the derivative vector as a function of the independent variable.

OUTPUT - a printing routine which produces output at desired intervals of the independent variable. (This was discussed in chapter 3)

SAVEP - this routine is called following each valid time step and is intended for the storage of variables for subsequent plotting.

END(ISW)- this routine is called at the end of the simulation and performs any final

processing required. If the ISW flag has its value changed from 0 the entire simulation is repeated.

Figure 4.1 illustrates the flow of the program through these routines. This structured approach makes the integration method and its effects on program flow completely transparent to the writer of a simulation, while still permitting virtually all the flexibility that would be possible by writing an entire executive. As a result the integration package serves as a convenient basis for problems ranging from simple equation oriented simulations requiring only a few lines per subroutine, up to and including very large modular simulators. One benefit of this flexibility is that it is particularly easy to write DYFLO simulations in a form acceptable to the package. This is further simplified by a small library of utility routines which includes a routine to mimic the action of the DYFLO INT routine. Other library routines aid in plotting variables, solving implicit equations and taking numerical derivatives.

4.2 The DYNSYS Process Segment

When this segment is combined with the integration package that was just discussed, the result is effectively a modified DYNSYS simulator. Although reconfigured

extensively, the basic input, output and module calling routines as well as the primary data structure would be familiar to a DYNSYS user. Although it is not feasible to discuss all the changes which were made the major modifications will be briefly reviewed.

4.3 Data Structure

The main process variables from DYNSYS 2.0 have been retained in a modified form in the common block MAT:

IM - Module pointer

number of the module currently being processed. Same usage as in DYNSYS 2.0

MP(100,10) - Module Parameter matrix

row length was reduced to 10 since the emphasis on more elemental modules requires fewer streams per module. This also led to the dropping of the extra parameter array EX.

- the convention of negative output stream numbers was dropped.

- the first element in the row is now the type number and the second element is the unit number.

EP(100,10) - Equipment Parameter matrix

S(175,14) - Stream matrix

- the explicit integration routine does not require the retention of previous time step values so the third dimension has been dropped.
- stream enthalpy is now stored in position 6 of the stream vector.

SIG(50) - SIGnal array

- in the original DYNSYS a full stream vector was used to carry a single signal variable. To increase storage efficiency this signal array was created.

In order to permit a large selection of components in the property package without greatly increasing the size of the stream matrix, the PROP common block was created. Its variables are:

NCOMP - the number of components used in the current simulation. At the present time this is limited to a maximum of 8, but this can be changed with redimensioning.

NCALL - a rarely used index to temporary storage in the property routine. (see chapter 5)

NCP(8) - this is a list of the components to be used in the current simulation. Fifteen

different components are presently available in the property routine (Table 4.1) and additional components can easily be added. The first NCOMP elements of the NCP array represent the components whose mole fractions are stored in the stream vector positions 7 to 6 + NCOMP.

The unit common block contains the parameters NE, NS and NSIG, which store the total number of units, streams and signal streams used in the simulation. The OUT common block stores a series of flags which specify the types of output to be generated.

4.4 I/O Routines

The I/O routines use the same style as the original DYNSYS, but permit free format inputs and have a number of features to aid in inputting the data and in controlling the appearance of the output. The most important new I/O feature is the restart capability.

Whenever the simulation is completed the program creates a data file which contains sufficient information to allow the program to resume the simulation at that point in time. Since the restart file is designed to be read and modified by the user between runs, adjustments can be made to the simulation as it proceeds. It also provides a useful debugging tool since it is created following any failure of the integration routine, thereby providing an accurate snapshot of the state of the simulation when if failed.

4.5 Module Writing

Module writing is significantly simpler in the new program than it was in DYNSYS 2.0. This is best illustrated by considering the coding of a simple module representing a proportional integral (PI) controller. The PI control algorithm can be written as:

$$\text{output signal} = K_p\left(\text{err} + K_I \int_0^t \text{err}\, dt\right) \tag{4.1}$$

where

$$\text{err} = \text{setpoint} - \text{measured value}$$

As in DYNSYS all modules have entry names of the form TYPEn where n is an integer. Thus the module might begin:

```
SUBROUTINE TYPE1
```

```
      COMMON /MAT/IM,MP(100,10),EP(100,10),
     & ,S(175,14),SIG(50)
      COMMON /DIF/ JSTART,TIME,NDEQ,Y(600),DY(600)
```

where the common blocks MAT and DIF communicate with the rest of the simulator. Normally these would be the only common blocks that a module would need to reference.

The MP array can be used to store the numbers of the input and output signal streams while the setpoint and the tuning parameters can be stored in the EP array. It is important to note that the first two positions of the MP array are reserved for unit identification.

```
      IN = MP(IM,3)
      IOUT = MP(IM,4)
      SETPT = EP(IM,1)
      XKD = EP(IM,2)
      XKI = EP(IM,3)
```

The error can now be calculated:

```
      ERROR = SETPT - SIG(IN)
```

If this is the first time step the initial value of the integrated variable must be set. On the first pass the parameter JSTART is equal to 0. Since the NDEQ parameter carries the value of the number of elements already used in

the solution array Y and the derivative array DY:

```
IF(JSTART.EQ.0)Y(NDEQ+1)=(SIG(IOUT)/XKP-ERROR)/XKI
```

The initial value was determined by rearranging equation 4.1 and solving for the integral term. The new output signal can now be calculated from the integrated value of Y:

```
SIG(IOUT)=XKP*(ERROR+XKI*Y(NDEQ+1))
```

and the new derivative value is calculated:

```
DY(NDEQ+1)=ERROR
```

All that remains is to advance NDEQ by the number of equations added by the module:

```
NDEQ=NDEQ+1
RETURN
END
```

Some slight additions to this code might be necessary to properly scale the derivatives and solution values to agree with the desired error limits. Scaling of all such values to near unity is advisable for consistency between modules.

CHAPTER 5

5.1 The Thermodynamic Property Package

It is common for chemical engineering process simulators to calculate thermodynamic properties from polynomial expressions. This has the advantage of allowing the user to include any substance for which there is sufficient data to generate the necessary coefficients. Unfortunately the results often do not accurately represent the physical system and this can result in serious difficulties with phase and reaction equilibrium calculations.

The light hydrocarbon industry is fortunate to have accurate equations of state at its disposal. These equations of state are parametric correlations capable of describing the pressure, temperature and specific volume behaviour of both liquid and vapour mixtures containing any combinations of the components of interest. Relatively simple thermodynamic calculations (Balzhiser et al, 1972, ch. 9) can be used with equations of state to predict mixture fugacity, enthalpy, entropy and heat capacity deviations from the ideal state. Since fugacities represent the chemical driving force for mass transfer and reaction processes, an equation of state can provide an accurate means of performing equilibrium calculations.

In recent years the two parameter equation of state has achieved widespread acceptance in the light hydrocarbon industry. These equations are relatively simple to implement and in the case of the Soave Redlick Kwong (SRK) and the Peng Robinson (PR) equations, they are very accurate across a wide range of conditions. For this project the Peng Robinson equation was chosen because it appears to exhibit slightly better performance in difficult situations. (Peng, 1976; Bishnoi, 1978) Due to the structural similiarity of the equations it would not be difficult to incorporate the SRK equation into the package.

The basic Peng Robinson equation is surprisingly simple considering its accuracy:

$$P = \frac{RT}{\bar{V} - b} - \frac{a}{\bar{V}(\bar{V} + b) + b(\bar{V} - b)} \qquad (5.1)$$

where the parameters are:

$$a = \frac{\alpha\ 0.45724 R^2 T_c^2}{P_c^2}$$

$$\alpha = (1 + K(1 - (\frac{T}{T_c})^{\frac{1}{2}}))$$

$$K = 0.37464 + 1.5422w - 0.2699w^2$$

$$b = \frac{0.0778RT_c}{P_c}$$

When multicomponent properties are being calculated, the mixture parameters are calculated from the pure component parameters with the following mixing rules:

$$a = \sum_{i=1}^{n} \sum_{j=1}^{n} X_i X_j a_{ij}$$

$$a_{ij} = (1 - \delta_{ij}) \ (a_i a_j)^{\frac{1}{2}}$$

$$b = \sum_{i=1}^{n} X_i b_i \qquad (5.2)$$

where δ_{ij} is a binary interaction coefficient characterizing the binary formed by component i and component j.

Equation 5.1 can be rewritten as:

$$Z^3 - (1 - B)Z^2 + (A - 3B^2 - 2B)Z - (AB - B^2 - B^3) = 0 \qquad (5.3)$$

where

$$A = \frac{aP}{R^2T^2}$$

$$B = \frac{bP}{RT}$$

$$Z = P\frac{\bar{V}}{RT}$$

In practice it is usually simpler to perform the thermodynamic calculations in terms of the compressibility factor Z. When equation 5.3 is solved for Z and three positive real roots occur the largest root is chosen for a vapour mixture while the smallest is chosen for a liquid mixture. In all cases the middle root is ignored.

This simulator only requires the calculation of fugacities, enthalpies, densities and heat capacities. The partial molar fugacity of a component i in a mixture can be calculated from: (Balzhiser et al, 1972, p373)

$$\ln\left(\frac{\bar{f}_i}{X_i P}\right) = \int_{P^*}^{P} \left(\frac{\bar{V}_i}{RT} - \frac{1}{P}\right) dP \tag{5.4}$$

Since the Peng Robinson equation is pressure explicit it is useful to rearrange 5.4 with the help of Maxwell relations to:

$$RT \ln\left(\frac{\bar{f}_i}{X_i P}\right) = \int_{V}^{\infty} \left[\left(\frac{\partial P}{\partial N_i}\right)_{T,V,N_j} - \frac{RT}{V}\right] dV - RT \ln\left(\frac{PV}{NRT}\right) \tag{5.5}$$

The partial derivatives can be evaluated using the Peng Robinson equation and then 5.5 can be integrated to yield:

$$\ln\left(\frac{\bar{f}_i}{X_i P}\right) = \frac{b_i}{b}(Z - 1) - \ln(Z - B)$$

$$- \frac{A}{2.828B}\left(\frac{2\sum_j x_j a_{ji}}{a} - \frac{b_i}{b}\right) \ln\left(\frac{Z + 2.414B}{Z - 0.414B}\right) \tag{5.6}$$

where the transformation to the Z variable was made to simplify the final equation.

In a similiar manner the thermodynamic equation:

$$H - H^* = RT(Z - 1) + \int_{\infty}^{V} \left[T\left(\frac{\partial P}{\partial T}\right)_V - P\right]dV \tag{5.7}$$

can be transformed using the PR equation of state into:

$$H - H^* = RT(Z - 1) + \frac{T\frac{da}{dT} - a}{2.828b} \ln\left(\frac{Z + 2.414B}{Z - 0.414B}\right) \tag{5.8}$$

Heat capacity deviations may be calculated from the temperature derivatives of the enthalpy expression. The ideal state enthalpies and heat capacities are calculated from correlations by Passut and Danner (1972).

Since the properties discussed can be calculated explicitly, implementation was straight forward. Phase equilibrium calculations present greater problems because these require the solution of a set of highly nonlinear implicit equations:

$$\bar{f}_i^V = \bar{f}_i^L \qquad i = 1,2,3, \ldots n$$

$$\sum_{i=1}^{n} X_i = 1$$

$$\sum_{i=1}^{n} Y_i = 1 \qquad (5.9)$$

Routines based on a conventional iterative approach taken from Evelein (1977) were written for the package, but these proved to be too inefficient and unreliable for use in dynamic simulation. To overcome this problem a new approach was taken to distillation simulation which eliminated the need for phase equilibrium routines. This approach will be detailed in the next chapter.

Although iterative property calculations are not necessary, the explicit calculations are still quite involved and may be required literally hundreds of thousands of times in a large and complex simulation. In order to reduce the time spent in the calculation of these basic properties, a routine was developed which linearizes the properties with respect to their partial derivatives. If the input to the property can be represented by a vector $\overline{I}$ where:

$$\overline{I} = X_i, T, P \qquad (5.10)$$

and where the output vector can be represented by:

$$\overline{O} = \overline{f}_i, \rho, Cp, H \qquad (5.11)$$

then it is possible to approximate $\bar{O}$ by

$$\bar{O} = \bar{O}' + \left(\frac{\partial \bar{O}'}{\partial \bar{I}'}\right) (\bar{I} - \bar{I}') \tag{5.12}$$

where $\bar{O}'$ is the exact solution at $\bar{I}'$.

Although the Jacobian of partial derivatives could be evaluated analytically, the potential savings in execution time were not considered to be worth the loss of generality and simplicity that was possible with numerical derivatives. The Jacobian is recalculated if the temperature changes by more than five degree Kelvin, the pressure changes by more than five percent or the following condition is not met

$$\frac{X_i - X_i^o}{1 + X_i^o} < 0.025 \tag{5.13}$$

This eliminates Jacobian re-evaluations for large percentage changes in small components. If these limits are set too high there will be a significant difference in the returned properties from before and after the Jacobian evaluation. This can cause serious difficulties with the numerical integration routine since it introduces discontinuities into the simulation. If the tolerances are set too low, the continual re-evaluation of the Jacobians will be more expensive than just evaluating the properties directly each time. The tolerances chosen are arbitrary but have given good perfor-

mance.

Because the Jacobians can require a substantial amount of storage provision was made for similiar streams to share Jacobians. When the NCALL parameter of the PROP common block has a value of 2 the values stored in the second position of the stream vectors serve as indexes to the Jacobian storage. Streams with the same index would use the same Jacobian. Since an excessive number of Jacobian evaluations will occur if the same Jacobian is used for streams which differ significantly this feature was not used in any of the simulations and should only be considered if core limitations are a problem.

CHAPTER 6

6.1 Process Models

During the course of this project a number of process models were developed for use with the modified simulator. (Table 6.1) No attempt will be made to discuss the simpler models which are based on well established principles, but the more unconventional approaches used in the distillation and heat transfer models will be discussed in detail.

6.2 Distillation Simulation

In this study distillation simulation is carried out by assembling various types of modules in a manner which approximates the physical situation. The most important of these modules is the counter current vapour-liquid mass transfer stage depicted in Figure 6.1. In all cases this module must determine the properties of the outgoing liquid and vapour streams, given the time dependent variables of the input stream and certain information about the characteristics of the tray.

The dynamic behavior of the stage is determined by the rates at which it accumulates material and energy. Assuming perfect mixing in both phases and the absense of chemical reactions, the mole balances can be written as:

$$\frac{dN_{i,n}}{dt} = L_{n+1}X_{i,n+1} + V_{n-1}Y_{i,n-1} - L_nX_{i,n} - V_nY_{i,n} \quad (6.1)$$

The corresponding energy balance is:

$$\frac{dE_n}{dt} = L_{n+1}H^L_{n+1} + V_{n-1}H^V_{n-1} + Q - L_nH^L_n - V_nH^V_n \quad (6.2)$$

In order to use these equations to determine the output stream variables, assumptions must be made, and it is in these assumptions that the model developed in this study differs significantly from the conventional equilibrium model. In order to put this approach in perspective, it will be of benefit to review the development of a simple conventional model.

The Simple Equilibrium Model

To develop the simplest of these models requires that the following assumptions be made:

1) Assume the vapour leaving a stage is in thermodynamic equilibrium with the liquid on that stage.

- Although this assumption is never truly valid it is

often a reasonable approximation. In some cases however, particularly for absorption and stripping, it is grossly in error. Two methods are commonly used to circumvent this problem, the simplest of which is to use a ratio of simulated ideal trays to actual trays which roughly corresponds to the observed tower efficiency. (ie. a 20 tray tower that is approximately 50% efficient would be simulated with a model having only 10 trays)

A somewhat more sophisicated approach is the use of Murphee tray efficiencies. These are defined as:

$$Em = \frac{(Y_{i,n-1} - Y_{i,n})}{(Y_{i,n-1} - Y^{*}_{i,n})} \qquad (6.3)$$

where Y^{*} represents the composition of the vapour in equilibrium with the tray liquid. Although commonly used, these have little in the way of a theoretical basis.

2) Assume the vapour holdup is negligible.

- For the vast majority of situations this assumption is reasonable, but inaccuracies can occur in high pressure towers where the liquid vapour density ratio is small. For instance the density ratio in a column

operating at atmospheric pressure and room temperature would be of the order of 1000 to 1, while ratios of less than 10 to 1 are common in gas plant absorbers. Thus the vapour holdup in the gas plant absorber represents a much larger fraction of the total holdup than is the case with the atmospheric column.

3) Assume the total holdup on the plate is constant

- This assumption is quite reasonable for small excursions from steady state, particularly if it is the volumetric holdup which is held constant while the molar holdup floats with changes in the liquid density. Simonsmeier (1977) compared simulations which had large differences in the value of the assumed holdup and found only slight variations in the results.

4) Assume the total plate enthalpy does not change.

- This is applicable only if assumption (3) has been made and even then it may introduce considerable error if the liquid composition changes markedly during the course of the simulation.

Assumption (1) allows the composition of the vapour stream leaving the tray and the temperature of both output streams to be calculated from a bubble point temperature calculation. Since assumption (2) implies that the liquid composition is the same as the total holdup composition it may be determined from the integrated values of equation 6.1. Assumption (3) permits the writing of an overall mass balance as:

$$L_n = L_{n+1} + V_{n-1} - V_n \tag{6.4}$$

A second equation is necessary to solve for the two unknowns L_n and V_n This is provided by rewriting equation 6.2 with assumption (4):

$$L_{n+1}H^L_{n+1} + V_{n-1}H^V_{n-1} = L_nH^L_n + V_nH^V_n \tag{6.5}$$

Rearranging yields:

$$V_n = \frac{L_{n+1}H^L_{n+1} + V_{n-1}H^V_{n-1} - L_nH^L_n}{H^V_n} \tag{6.6}$$

Substituting (6.4) into (6.6) and rearranging gives:

$$V_n = \frac{L_{n+1}H^L_{n+1} + V_{n-1}H^V_{n-1} - (L_{n+1} + V_{n-1})H^L_n}{H^V_n - H^L_n} \tag{6.7}$$

There are now sufficient relations to define the system.

The normal calculation procedure is:

1) calculate the bubble temperature and vapour composition from the liquid composition and pressure.

2) determine the vapour and liquid enthalpies at the bubble temperature.

3) determine V_n from equation 6.7

4) determine L_n from equation 6.4

5) calculate derivatives from 6.1

6) perform a numerical integration to determine the liquid compositions at the new time level. Note that equation 6.2 is no longer a differential equation.

7) Go to step 1

Most distillation simulators use some variation of this simple model. For example it is possible to determine the liquid flow by integrating the following equation:

$$\frac{dL_n}{dt} = \frac{L_{n+1} + V_{n-1} - L_n - V_n}{HTC} \tag{6.8}$$

where HTC is the hydraulic time constant for the liquid on the tray. This allows the liquid holdup to float to some degree and this variation in holdup can be represented by:

$$\frac{dN_n}{dt} = L_{n+1} + V_{n-1} - L_n - V_n \tag{6.9}$$

Since the total energy holdup E_n is a product of the molar liquid enthalpy H_n^L and the total molar holdup N_n the energy derivative can be written as:

$$\frac{dE_n}{dt} = H_n^L \frac{dN_n}{dt} + N_n \frac{H_n^L}{dt} \tag{6.10}$$

By substituting equation 6.10 into 6.2 the vapour flow may be calculated from:

$$V_n = \frac{L_{n+1}H_{n+1}^L + V_{n-1}H_{n-1}^V + Q - L_nH_n^L - H_n^L \frac{dN_n}{dt} - N_n \frac{dH_n^L}{dt}}{H_n^V} \tag{6.11}$$

While dN_n/dt can be determined from equation 6.9 the enthalpy derivative must be determined by numerical differentiation. This technique, used by Svrcek (1967) and Distefano (1968), was considered a significant improvement

over the simple method, since it allowed the tray energy dynamics to be approximated in the model while still permitting the vapour flow to be calculated explicitly. It is possible to assume the numerical derivative is zero on some non important trays (Franks 1972) in which case those trays are effectively calculated by the simple model equation 6.7

A minor variation on these models arises with the introduction of a hydraulic correlation to calculate the liquid downflow. Typically the Francis weir formula is used, but Simonsmeier (1977) recommends the A.I.C.H.E. bubble cap formula.

A Non-Equilibrium Model

The inability of equilibrium models to handle absorbing and stripping problems, the tendency for numerical differentiation calculations to introduce instabilities into the integration, and in particular the large amounts of computation time required for phase equilibria calculations, made conventional distillation models impractical for this project.

Mass transfer on a distillation tray is a complex process taking place in a poorly defined mixture of vapour, liquid and froth, and despite considerable research (Bernard et al, 1966; Thorogood, 1963 ; Strand, 1963; D'Arcy,1978) these processes are not well understood. In the following pages an analysis of a very simple and idealized system will be used to suggest the form of a pseudo-empirical approach to the distillation problem.

Consider a small bubble of gas rising through a liquid and assume the rate of mass transfer of component i from the liquid to the bubble may be described by:

$$\frac{dN_{i,P}}{dt} = KA(\bar{f}^{L}_{i,n} - \bar{f}^{V}_{i,p}) \qquad (6.12)$$

The vapour fugacity may be assumed to be:

$$\bar{f}^{V}_{i} = \phi_i P_i \qquad (6.13)$$

The partial pressure may be evaluated from the ideal gas law:

$$P_i = \frac{N_{i,P} RT}{V_P} \qquad (6.14)$$

If the diameter of the bubble is D then equation 6.15 may be rewritten as:

$$\frac{dN_{i,P}}{dt} = K\pi D^2 \left(\bar{f}^L_{i,n} - \frac{6\phi_i N_{i,P} RT}{\pi D^3}\right) \tag{6.15}$$

If it is assumed that D remains constant with respect to time, equation 6.15 can be rearranged and integrated to:

$$\int_{N^o_{i,P}}^{N_{i,p}} \frac{dN_{i,P}}{Sa - SbN_{i,P}} = \int_0^t dt \tag{6.16}$$

where

$$Sa = K\pi D^2 \bar{f}^L_{i,n}$$

and

$$Sb = \frac{6K\phi_i RT}{D}$$

The lower integration limit $N^o_{i,p}$ represents the number of moles of component i present in the bubble when it enters the layer of liquid at time zero. The upper integration bound $N_{i,p}$ represents the number of moles of component i in the bubble after a contact time t. The result of this integration is:

$$N_{i,P} = \frac{Sa - (Sa - SbN^o_{i,P})\, e^{(-Sb\ t)}}{Sb} \tag{6.17}$$

If the total volume of vapour entering the liquid is V^o and it is all carried in bubbles of diameter D, then the number of bubbles flowing through the liquid per unit time will be:

$$M_P = \frac{6V^o}{\rho^{V^o} \pi D^3} \tag{6.18}$$

The number of moles of component i initially in a bubble will be:

$$N^o_{i,P} = \frac{V^o Y^o_i}{M_P} = \frac{Y^o_i \rho^{V^o} \pi D^3}{6} \tag{6.19}$$

and the number of moles in the bubble after a contact time t will be:

$$N_{i,P} = \frac{VY_i}{M_P} \tag{6.20}$$

Substituting these relations into (6.17) and rearranging results in:

$$VY_i = V^o\left[\frac{\bar{f}^L_i}{\rho^{V^o} \phi_i RT} - \left(\frac{\bar{f}^L_i}{\rho^{V^o} \phi_i RT} - Y^o_i\right) \exp\left(\frac{-Dt}{6K\phi_i RT}\right)\right] \tag{6.21}$$

This equation makes it possible to calculate both the composition and the total molar flow rate of the vapour leaving a liquid layer,as long as the inlet vapour flow and composition are known. However an assumption of constant volumetric flow rate through the liquid is implicit in the assumption of constant bubble diameter, and while the change in volumetric flowrate across a single tray would normally be small, it is advantageous to devise a means of overcoming this limitation.

One possible method is to divide the liquid into a large number of thin layers and then apply equation 6.21 to each layer in turn; with the output from one layer serving as the input to the next layer. Over the course of any one layer the amount of material transferred into the bubble would not be enough to significantly alter the internal pressure of the bubble and hence the mass transfer rates would not be affected. This is equivalent to readjusting the number of bubbles after each layer.

Since the rate of total mass transfer is simply the sum of the component equations 6.21, it may be assumed to exponentially decay with contact time and it is even possible to roughly estimate its time constant. This information can be used to greatly reduce the number of liquid layers

necessary to essentially eliminate the constant volume assumption.

Four liquid layers, with contact times of 1/30, 4/30, 9/30, and 16/30 of the total contact time, were used in this study. Although some attempt was made to compensate for the shape of the exponential curve, these intervals were chosen more or less arbitrarily. The total contact time, which can be calculated from

$$t_c = \frac{\rho_{n-1}^{V^o} \sum_{i=1}^{nc} N_i^o}{V_{n-1}\rho_n^L} \qquad (6.22)$$

is limited to maximum of 4.5 times the estimated time constant to ensure that the approximations are applied to the critical part of the exponential curve.

Implementation of the Distillation Model

The method just outlined provides a straight forward means of evaluating the flow and composition of the vapour leaving a tray. Due to the empirical nature of the method the constants K and D are amalgamated into a single user supplied parameter $K'=K/D$. By varying this parameter the model can be made to closely approximate anything from an equilibrium stage to a grossly inefficient stripper or

absorber. Vapour holdup is neglected, but if a situation required it, vapour 'tank' modules could be placed between the tray modules.

The liquid composition is determined by the integration of equation 6.1 and the liquid flow rate is calculated from the Francis weir formula:

$$L_n = 481.82 \rho_n^L D_w h_{ow}^{1.5} \tag{6.23}$$

where the height of liquid over the weir is calculated from:

$$h_{ow} = \frac{N_n}{\rho_n^L A_x} - h_{weir} \tag{6.24}$$

By rearranging equation 6.2 a differential equation in terms of the liquid temperature can be obtained:

$$\frac{dT_n}{dt} = \frac{L_{n+1} H_{n+1}^L + \ell_{n-1} H_{n-1}^V + Q - \ell_n H_n^V - H_n^L \left(\frac{dN_n}{dt} + L_n\right)}{Cp\ N_n} \tag{6.25}$$

The vapour outlet temperature could be determined by a heat transfer analogue to the mass transfer procedure, but for simplicity it has merely been set equal to the liquid temperature.

In its final form the model avoids most of the assumptions inherent in the equilibrium models, while enjoying a considerable advantage in efficiency as a result of not requiring the convergence of iterative phase equilibrium calculations.

6.5 The Reboiler Model

In this simulator the reboiler was represented by two modules; one for the heat transfer characteristics, and a second for the calandria. This second module is essentially an equilibrium stage with no vapour inlet stream, and this posed a problem for the new stage model, which requires an inlet vapour flow for its calculation. Use of a conventional equilibrium model for this one stage was attempted, but the results were not satisfactory.

The problem was resolved by the observation that the liquid and vapour outlet streams calculated by the new tray model were essentially at equilibrium at the liquid bubble point when large mass transfer parameters were used. This allows the bottoms stage to be handled as a normal stage with the vapour inlet composition being guessed by a Wilson equation prediction. The inlet flow rate is based on a tray energy balance similiar to that used in an equilibrium stage, but the passage through the liquid with high mass

transfer parameters trims both vapour composition and flow rate to more accurate values.

6.6 Heat Transfer Model

The same type of analysis which was used to develop the mass transfer model can be used in the creation of a heat transfer model. Let us consider a simple counter current, sensible heat transfer device such as the double pipe exchanger depicted in Figure 6.2. The problem is to predict the fluid outlet temperatures TA2 and TB2, given the inlet temperatures and the initial tube wall temperatures T_{t1} and T_{t2}. Several assumptions are made which facilitate the solution of the problem:

1) That there is sensible heat transfer only.
- elimination of this constraint will be necessary in the future development of condensor and evaporator modules.

2) That the fluid holdup is negligible.
- This results in the exchanger dynamics being controlled by the thermal capacity of the tube wall.
- A time lag module could be incorporated into the system if fluid delays were important.

3) That the tube metal has a linear temperature

profile.

- In reality the tube will have an exponential temperature profile, but the error introduced should be acceptable for most process simulations.
- When greater accuracy is required, several heat exchange modules could be joined together in series to provide a very good representation of an actual exchanger's profile.

4) That there is negligible heat transfer by conduction axially down the tube.

- This is justified by the small cross sectional area of the conduction path axially as compared to the radial path.

With these assumptions the module calculations are reduced to a relatively simple procedure:

1) Calculate the fluid outlet temperature as a function of the fluid properties, the inlet temperature, the inlet flow rate and of the tube wall temperature profile.

2) With all four fluid temperatures known, it is possible to calculate the net rate of heat transfer to the

tube wall at any position. It is therefore possible to calculate the rate of change of T_{t1} and T_{t2} with respect to time. The integration routine supplies the updated values of T_{t1} and T_{t2} which in turn establishes the new temperature profile.

The implementation of these procedures will now be considered in more detail.

For a differential distance down the device, the heat transfer to one of the fluids may be represented as:

$$Q = (T_t - T)UD\ dz \tag{6.26}$$

Noting that

$$dA = D\ dz \tag{6.27}$$

where A is the heat tranfer area, allows the generalization of equation 6.26 to

$$Q = (T_t - T)\ UdA \tag{6.28}$$

If T_t is assumed to be a linear function of A,

$$T_t = a + bA \tag{6.29}$$

where a and b are constants with respect to A. Equation 6.28 can now be written as:

$$Q = (a + bA - T)\ UdA \tag{6.30}$$

We know that this heat flow will cause a change in the fluid temperature as described by:

$$Q = Cp_f W_f dT \tag{6.31}$$

Note that this assumes only sensible heat transfer takes place. Equating 6.30 and 6.31 and rearranging yields:

$$\frac{dT}{dA} + BT = (a + bA)B \tag{6.32}$$

where

$$B = \frac{U}{Cp_f W_f}$$

As this is a linear first order differential equation, the general solution may be applied to yield:

$$T\ e^{BA} = a\ e^{BA} + \frac{b}{B}(BA - 1)\ e^{BA} + C \tag{6.33}$$

Using the boundary condition $T = T_o$ at $A = 0$ allows the evaluation of the integration constant:

$$C = T_o + \frac{b}{B} - a \tag{6.34}$$

Substituting (6.34) and B into (6.33) gives the final result:

$$T = (T_o - a + \frac{bCp_f W_f}{U})\ \exp(\frac{-UA}{Cp_f W_f}) + a + bA - \frac{bCp_f W_f}{U} \tag{6.35}$$

With this expression it is possible to evaluate the instantaneous fluid outlet temperatures for the two sides of the exchanger. This information allows the time derivative of the tube wall temperature to be calculated. For T_{t1} we can write:

$$\frac{Q}{A} = \frac{Cp_t W_t}{A} \frac{dT_{t1}}{dt}$$

$$= (T_{A1} - T_{t1})\ Ua + (T_{B2} - T_{t1})\ Ub \qquad (6.36)$$

It is important to note that all parameters must be referenced to the same area (i.e. tubeside or shellside)

This model served as a basis for a variety of heat transfer modules developed during the course of this project. A simple variation which assumes a constant fluid temperature on one side of the exchanger served to model the heat transfer characteristics of the column reboiler.

CHAPTER 7

7.1 Model Testing

One of the goals of this project was to use industrial rather than laboratory data for the testing of the models. This, however, presents a number of difficulties:

- the test system cannot be isolated from external disturbances as is possible in the lab. This can result in many test attempts being made before good data are collected.
- data gathering systems will almost always be more primitive in industrial plants than in specialized laboratory setups.
- Economic, environmental and safety concerns all limit the size of upset which can be tolerated in an industrial plant test.
- perhaps the most serious difficulty is in finding an industrial firm which will permit one of its plants to be used for such tests.

Offsetting these difficulties is the greatly increased utility of a model which has demonstrated its ability to handle in industrial problem. In our case, we were extremely fortunate to have the complete cooperation of Chevron Standard Limited in conducting dynamic response

tests on their Fort Saskatchewan, Alberta natural gas plant. The test system consisted of two fractionators operating in series, as shown in Figure 7.1. A feed stream consisting mainly of propane through hexane is fed to the first column where the propane is removed as overhead product. The remaining bottoms stream is used as feed to the second column which produces a butane overhead product and a C5+ condensate stream. The depropanizer has 43 mass transfer trays while the debutanizer has only 30. Both columns use hot oil reboilers and flooded air cooled overhead condensers.

What makes this plant particularly valuable for test purposes is that it is one of the few light hydrocarbon plants which has a process control computer. This machine and its on-line chromatographhs provide effortless and continuous monitoring of the test variables.

Although several tests were done only two sets of data were sufficiently free of outside disturbances to permit them to be modelled. During these tests all feed forward elements in the control system were switched off but local controllers were allowed to maintain their variables at the given setpoints.

The first test involved a composition pulse of 30 minutes duration. This was accomplished by pumping butane product from storage into the feed stream to raise the total butane content of the feed from 35 to 50 percent. Total feed rates were held constant. The second test consisted of a 10% step increase in hot oil flow to the reboiler of the depropanizer.

The first obstacle in modelling these tests was the sheer size of the problem. At least six components were necessary for the property calculations and the two towers had a combined total of 73 stages. A second problem involved achieving the same steady state the plant was at when the test began with the simulator. Several parameters such as heat transfer coefficients were not accurately known and had to be deduced by trial and error matching with steady state data. Since there were no provisions for performing steady state calculations with the new models, all changes had to be made by driving the simulation from one state to another dynamically.

When the first test was modelled it became apparent that the plant data lagged the simulation by approximately six minutes. This was apparently the result of the butane addition to the feed stream being done a considerable dis-

tance upstream of the depropanizer. A second problem was the tendancy of the model to react much quicker than the actual towers. This was largely resolved when the liquid holdup in the tray downcomers was taken into consideration. The results of the final simulation are compared to plant data points in Figures 7.2 through 7.10.

When the second test was modelled the lag problem did not occur, which supports the piping lag assumption for the first test. The results of the simulation and the plant test are compared in Figures 7.11 to 7.21.

For the most part the model was able to match the experimental data reasonably well. In many variables the majority of the error was a result of failing to exactly match the steady state values at the beginning of the run. Figure 7.6 shows this problem. It is also obvious from some of the results that the plant itself was not completely at steady state (see Figure 7.5) at the beginning of the tests. Considering these factors in light of instrument error tolerances, the model has performed considerably better than was originally hoped for.

The execution times for these simulations were on the order of 2 to 2.5 times real time, when performed on a Honeywell DPS Level 2 computer. Preliminary runs on a CDC Cyber 172 required execution times on the order of real time.

7.2 Model Testing of Control Systems

Although a comprehensive review of control systems could not be undertaken within the scope of this project, a few simulations of the Chevron depropanizer and its feed forward control system were run to demonstrate the potential ability of the program. Typical results from these runs are depicted in Figures 7.22 to 7.27 where the response of the depropanizer to a series of disturbances is depicted. First the feed C5+ concentration is increased from 35 to 50 percent for a 15 minute period (time 0 to 15). This is followed 45 minutes later by a ramped decrease of 33 percent in the feed flow rate over a 15 minute period (time 60 to 75) which is in turn followed by another 15 minute composition pulse identical to the first one. One of the controlled variables was the iC4 in the overhead, which is compared to its set point in figure 7.23. Since details of the control system can not be disclosed, these results cannot be discussed further in any detail.

CHAPTER 8

8.1 Conclusions

8.1.1 The flexibility and simplicity of modular simulation is seriously compromised by the use of implicit integration methods. Whenever possible explicit routines should be used in these applications.

8.1.2 Suitably modified the Richard Lanning Torrey method is capable of integrating quite stiff systems of differential equations with reasonable efficiency. This is true even for very large systems where integration methods which require the system Jacobian become inefficient.

8.1.3 A distillation tray model has been developed which does not require the assumption of stage equilibrium or the calculation of numerical derivatives.

8.1.4 The Peng-Robinson equation of state can be used with the above distillation model to produce a representation of the dynamic behavior of industrial distillation columns. Due to the complexity of the equation of state calculations, efficient algorithms are necessary if execution times are to be reasonable.

8.2 Recommendation For Future Study

The following areas should be considered for future study:

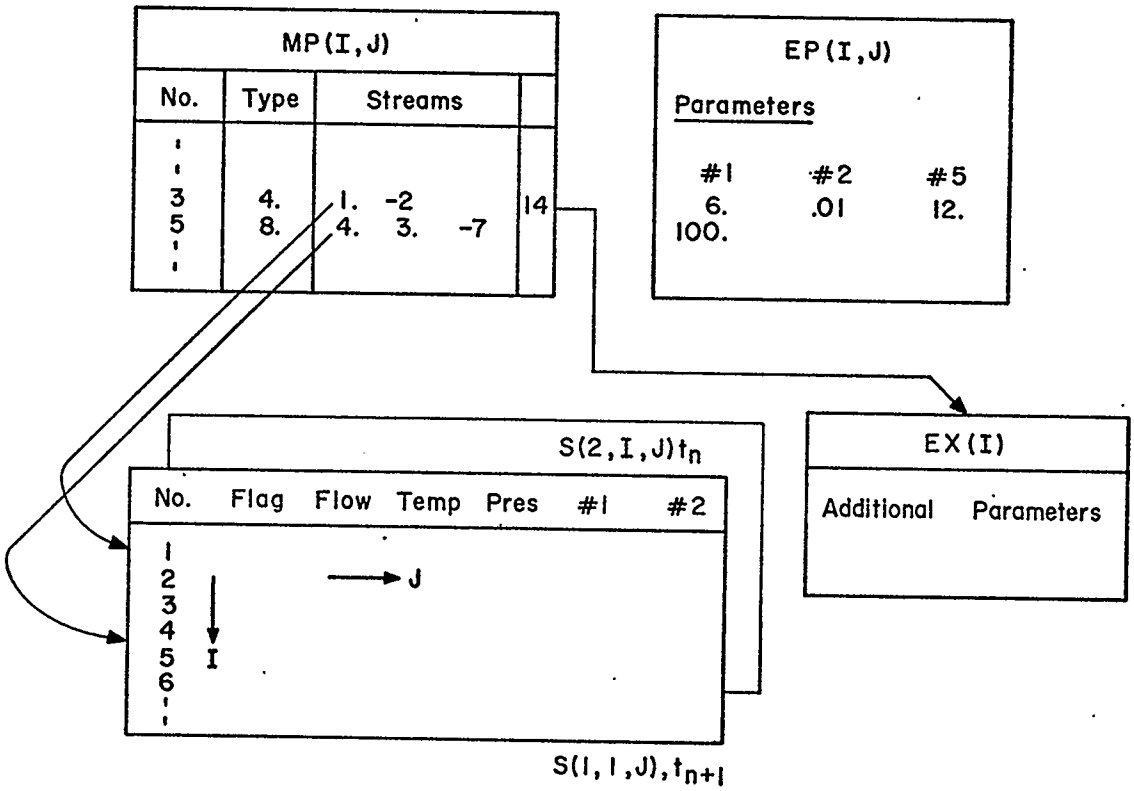
1) More complete testing of present models, in the form of further industrial tests, would be useful to establish the range of validity of the new models. Of particular interest would be dynamic tests on absorbers and strippers, since it is hoped that the new models would able to handle these with ease.

2) The main purpose of a simulator of this type is to permit evaluation of various types of control systems in a convenient manner. Although some runs have been done with the control system from Chevrons depropanizer no systematic study of control system performance could be undertaken within the scope of this project.

3) It would be interesting to evaluate the potential utility of the new distillation models in steady state calculations.

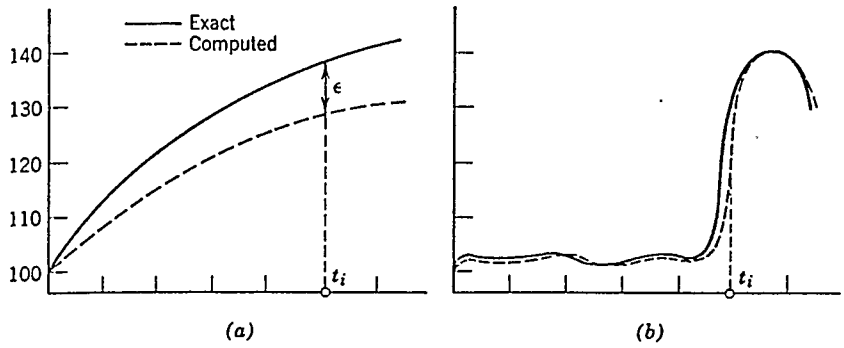
4) Further effort should be spent on increasing the efficiency of the property calculations. Several fairly simple modifications could be made which would make a substantial difference in the simulators overall execution time.

Figure 2.1



MAIN STORAGE UNITS FOR DYNSYS AND INFORMATION FLOW BETWEEN THEM

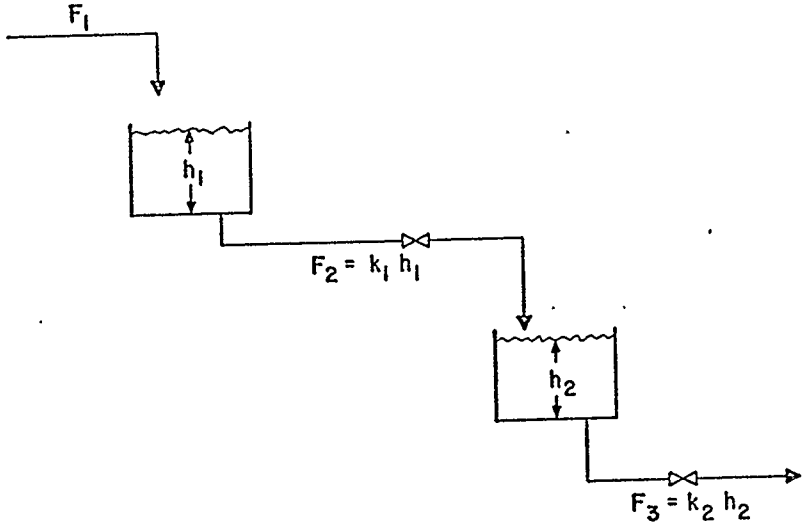
Figure 3.1



Two Cases Having a Maximum 10% Error

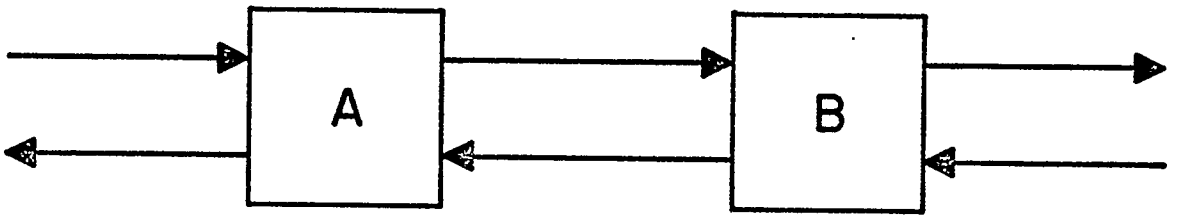
Types of Integration Errors

Figure 3.2



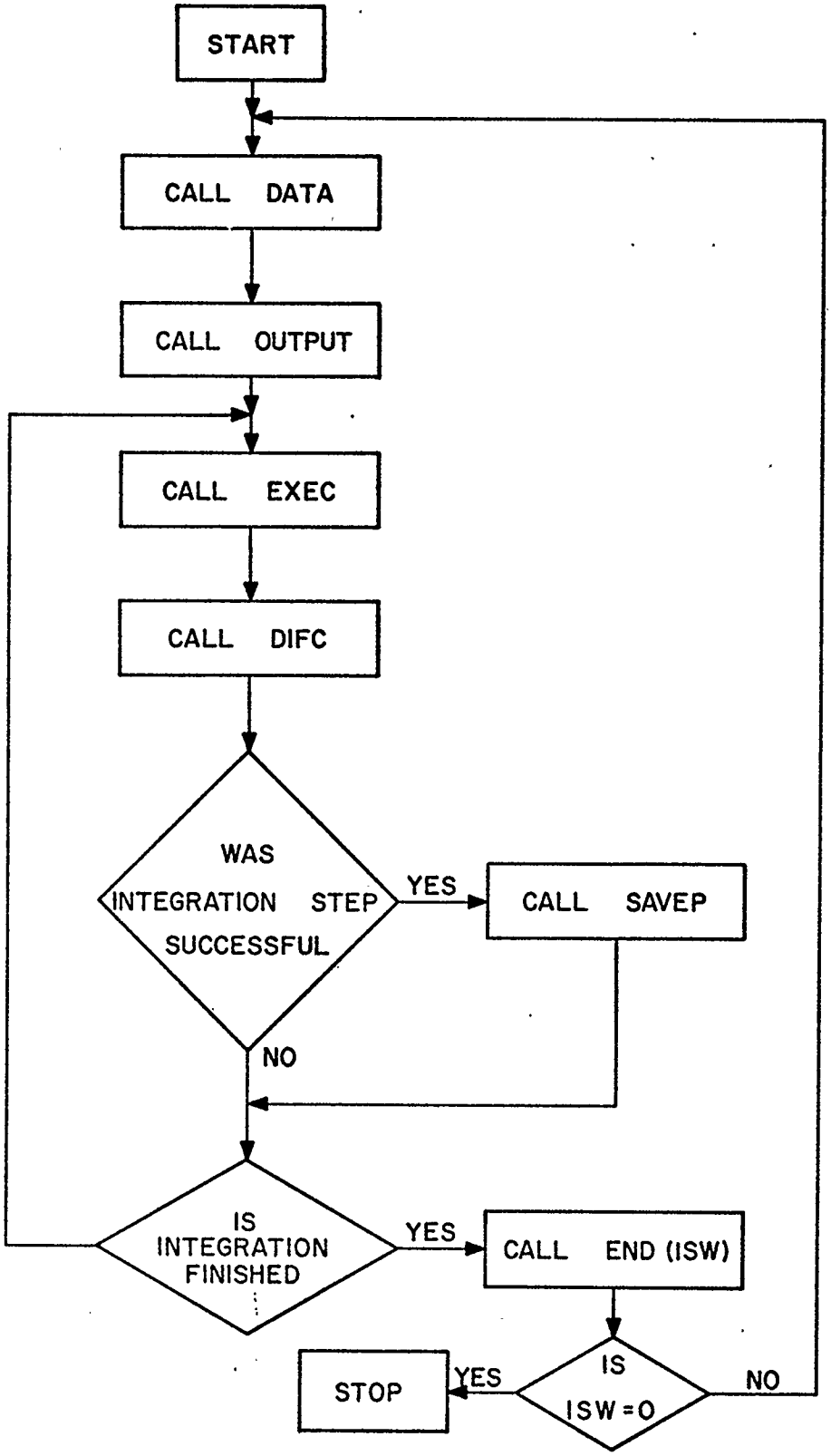
SIMPLIFIED TWO TANK FLOW SCHEME

Figure 3.3



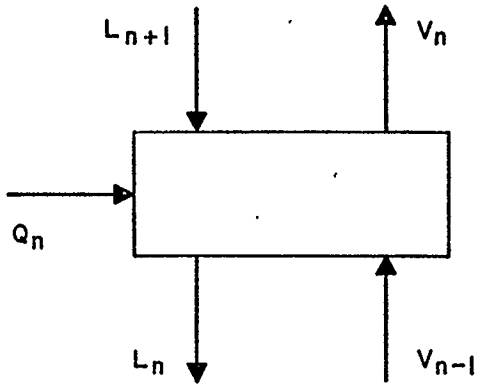
COUNTER CURRENT CONNECTION OF MODULES

Figure 4.1



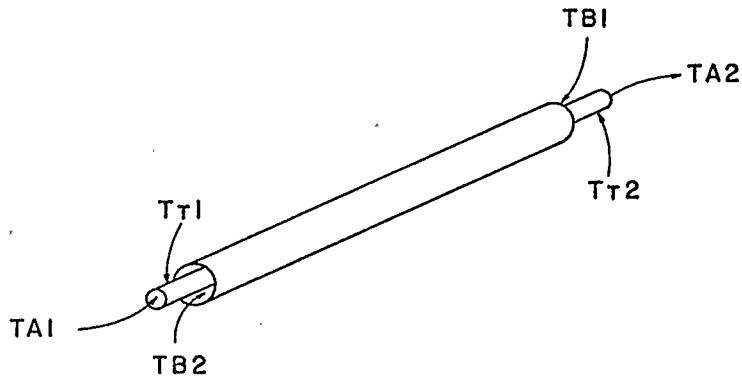
FLOW SHEET OF SEGMENT "MAIN"

Figure 6.1

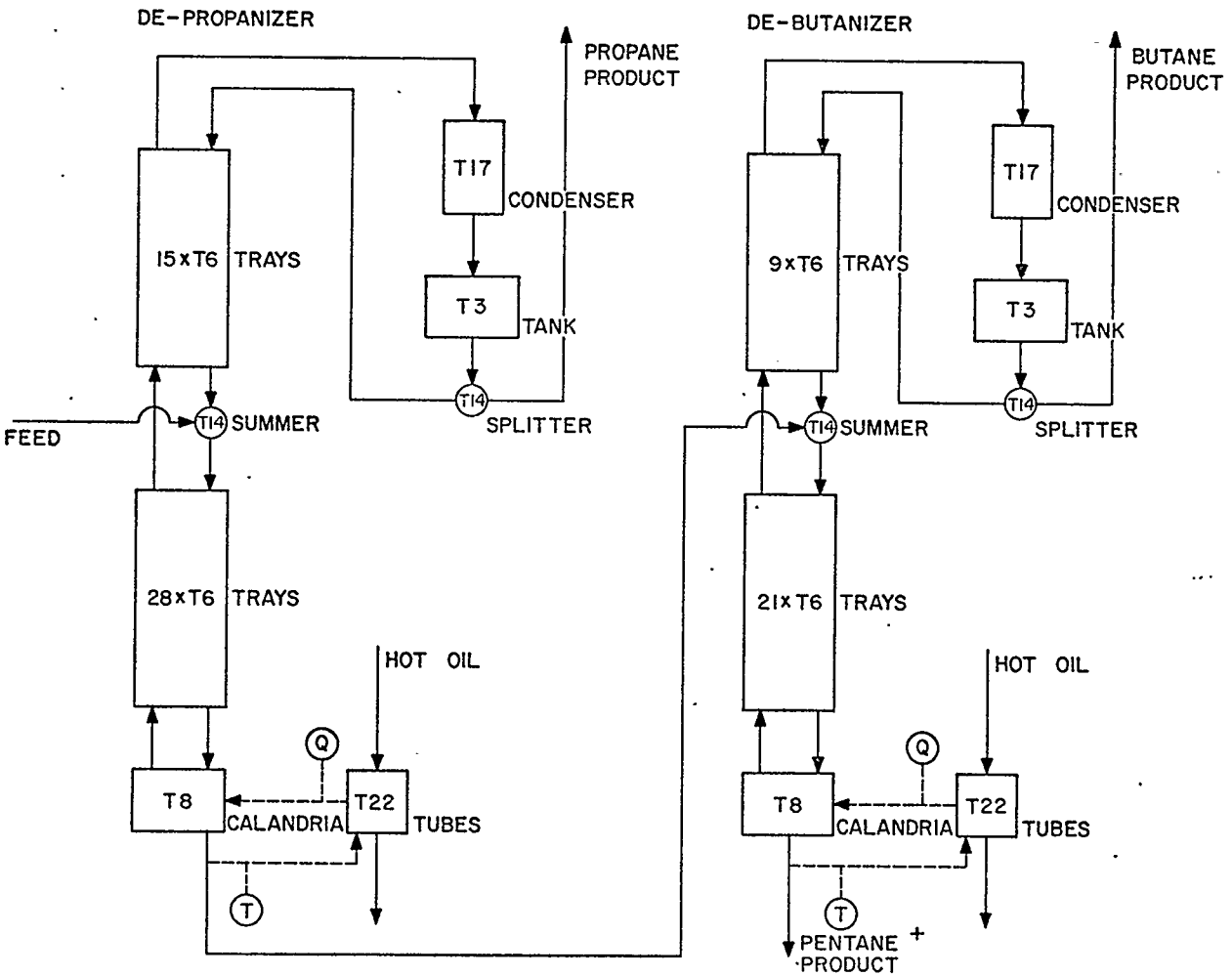


MASS TRANSFER STAGE "n"

Figure 6.2



DOUBLE PIPE HEAT EXCHANGER



BLOCK DIAGRAM OF TEST FRACTIONATION SYSTEM

Figure 7.1

Figure 7.2

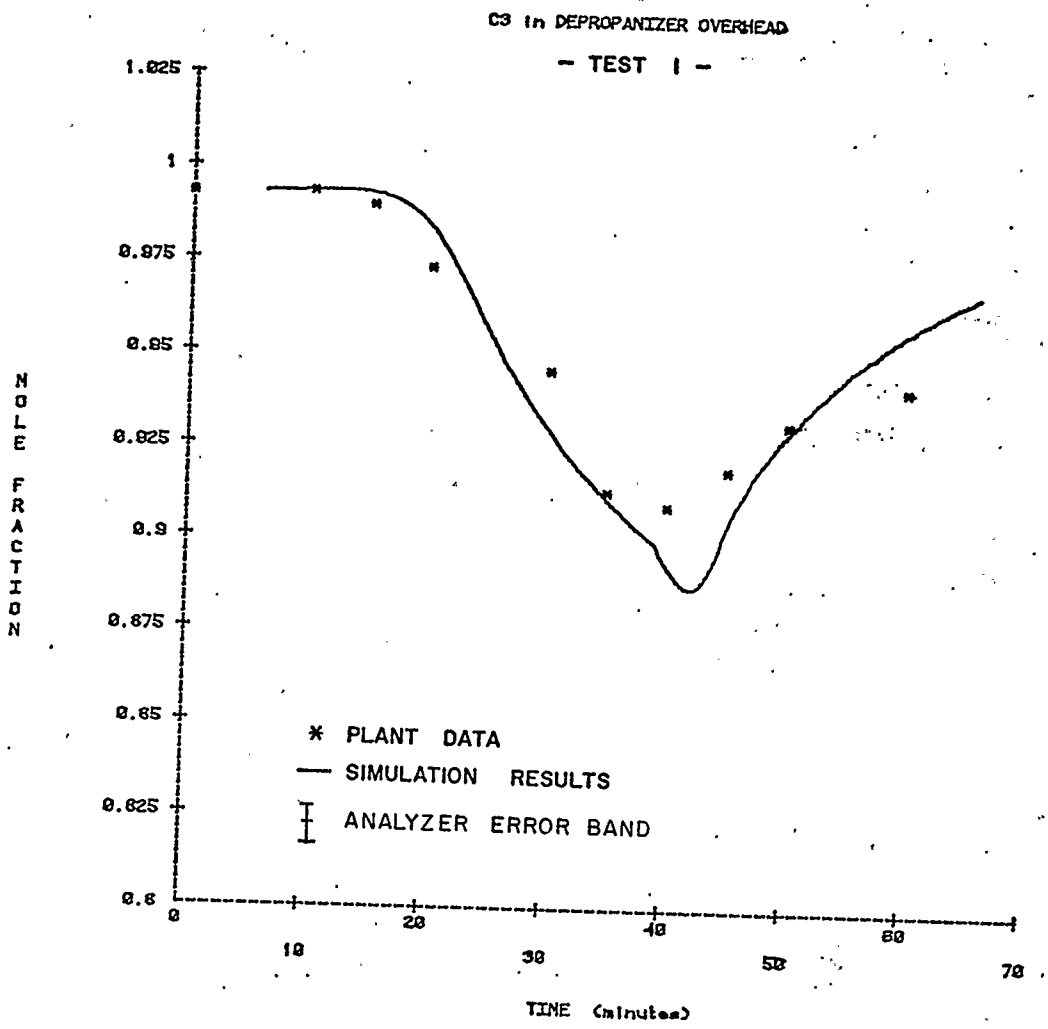

C3 in DEPROPANIZER OVERHEAD
- TEST I -
1.025
1
0.975
0.95
0.925
0.9
0.875
0.85
0.825
0.8
MOLE FRACTION
* PLANT DATA
SIMULATION RESULTS
ANALYZER ERROR BAND
0
10
20
30
40
50
60
70
TIME (minutes)

Figure 7.3

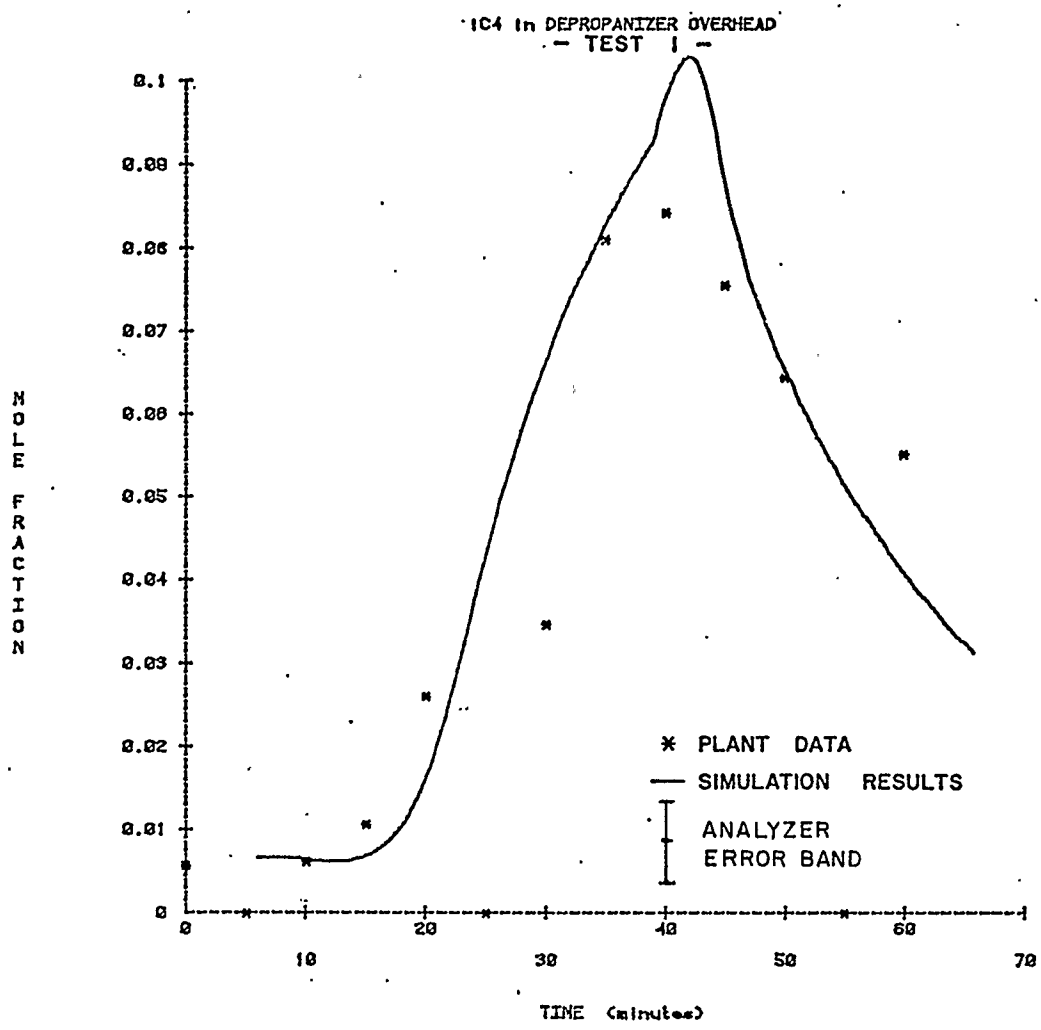

IC4 in DEPROPANIZER OVERHEAD
— TEST 1 —
MOLE FRACTION
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
10
20
30
40
50
60
70
TIME (minutes)
* PLANT DATA
— SIMULATION RESULTS
ANALYZER
ERROR BAND

Figure 7.4

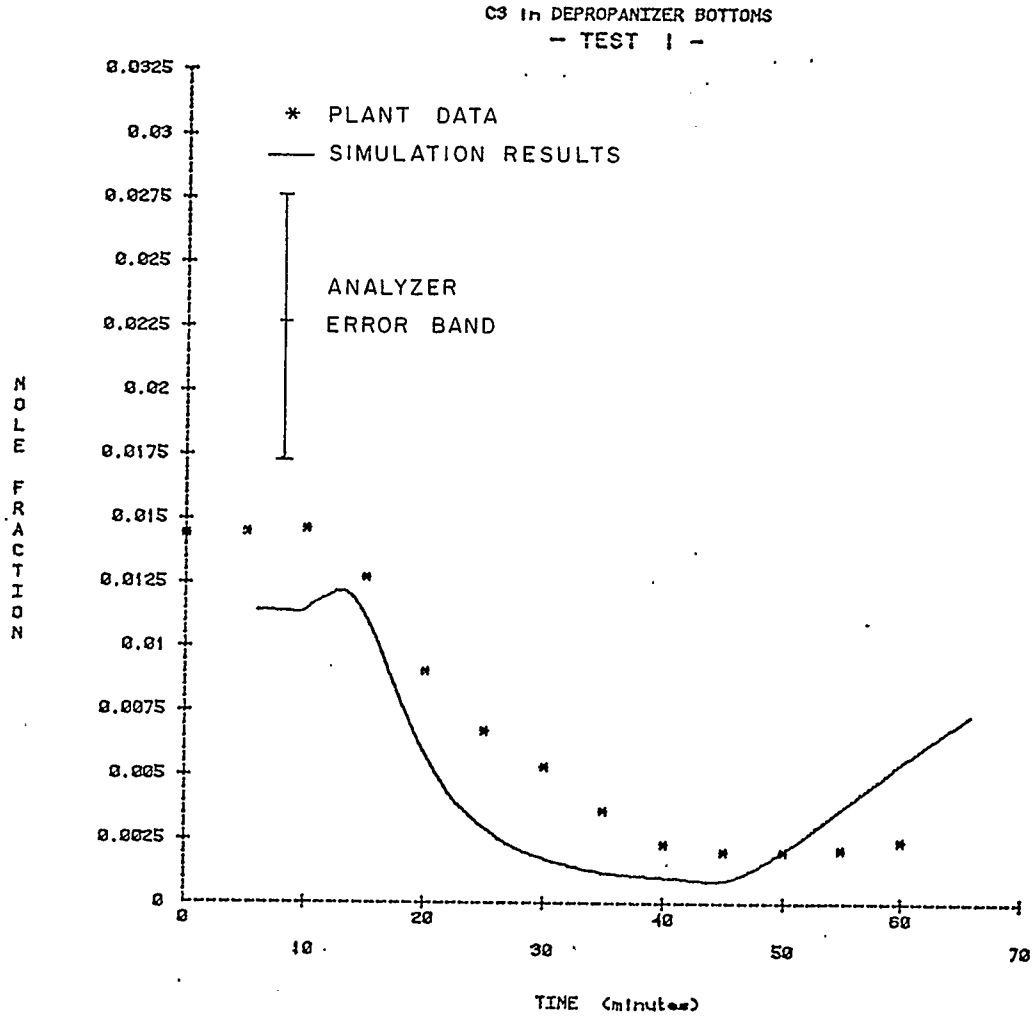

C3 in DEPROPANIZER BOTTOMS
— TEST 1 —
* PLANT DATA
—— SIMULATION RESULTS
ANALYZER
ERROR BAND
MOLE FRACTION
0.0325
0.03
0.0275
0.025
0.0225
0.02
0.0175
0.015
0.0125
0.01
0.0075
0.005
0.0025
0
0
10
20
30
40
50
60
70
TIME (minutes)

Figure 7.5

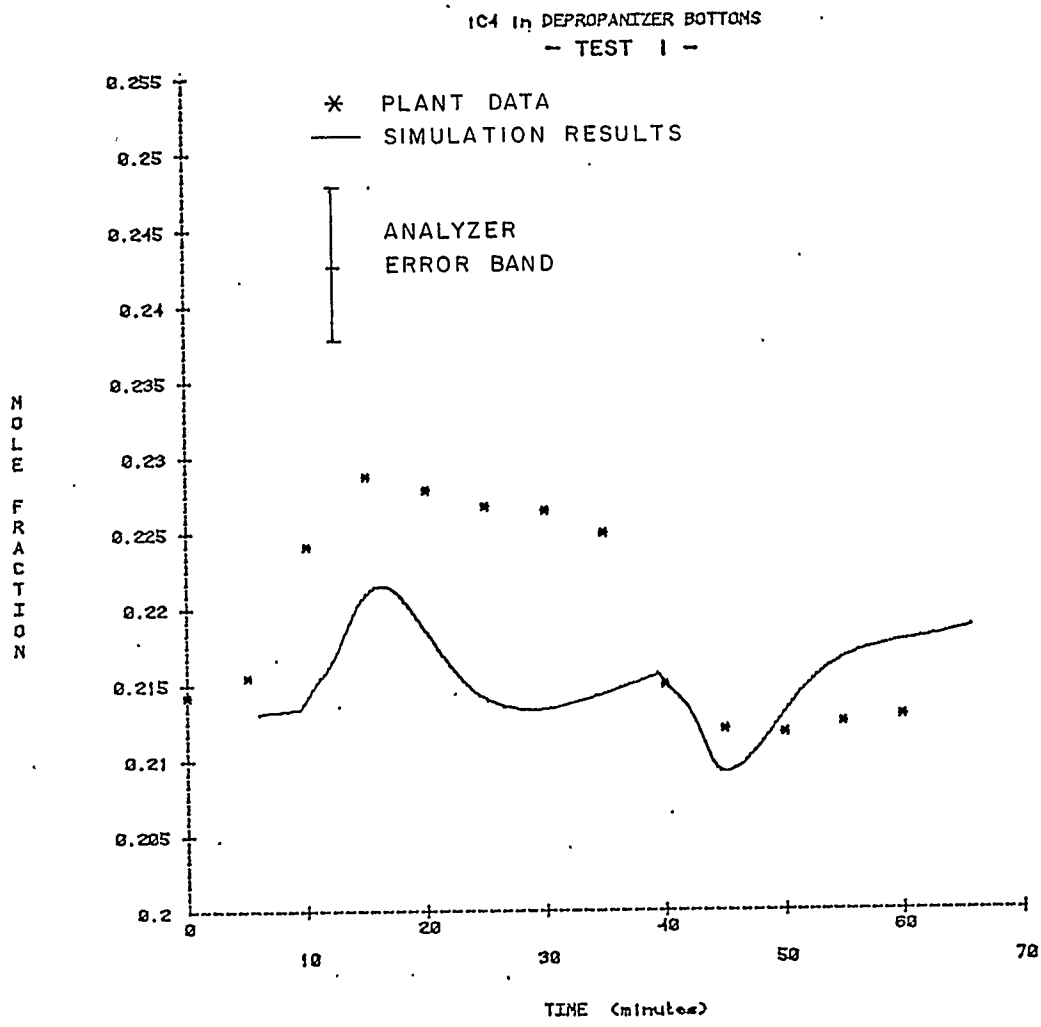

IC4 in DEPROPANIZER BOTTOMS
- TEST I -
* PLANT DATA
SIMULATION RESULTS
ANALYZER
ERROR BAND
MOLE FRACTION
0.255
0.25
0.245
0.24
0.235
0.23
0.225
0.22
0.215
0.21
0.205
0.2
0
10
20
30
40
50
60
70
TIME (minutes)

Figure 7.6

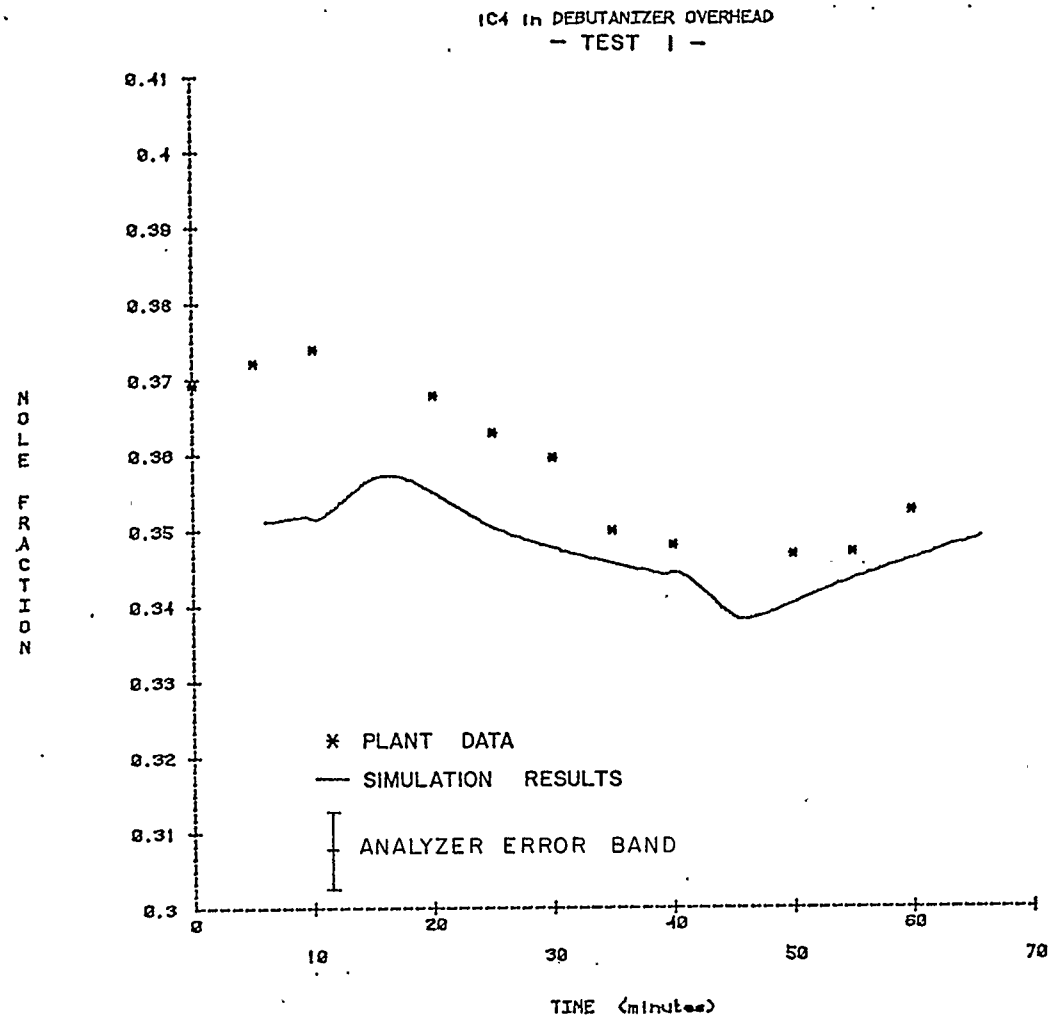

IC4 in DEBUTANIZER OVERHEAD
— TEST 1 —
MOLE FRACTION
0.41
0.4
0.39
0.38
0.37
0.36
0.35
0.34
0.33
0.32
0.31
0.3
0
10
20
30
40
50
60
70
TIME (minutes)
* PLANT DATA
SIMULATION RESULTS
ANALYZER ERROR BAND

Figure 7.7

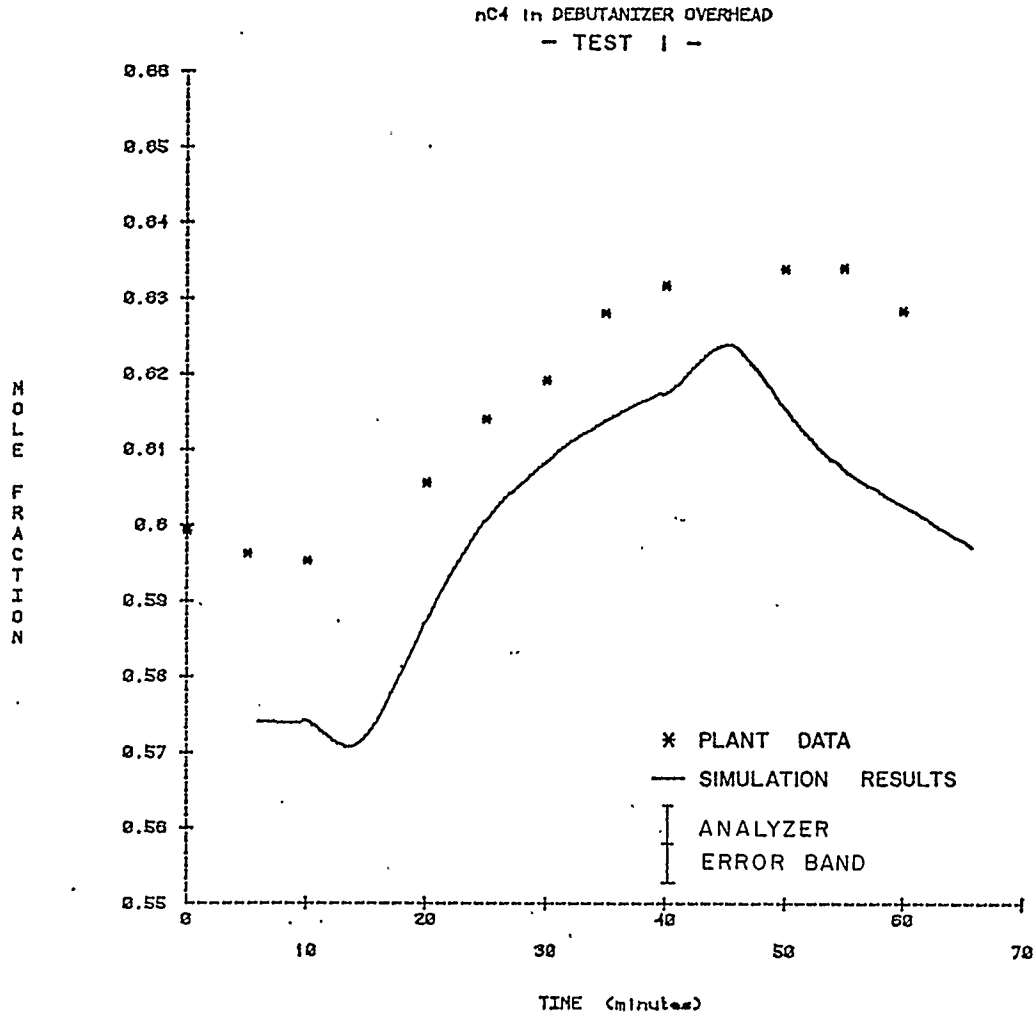

nC4 in DEBUTANIZER OVERHEAD
- TEST 1 -
0.66
0.65
0.64
0.63
0.62
0.61
0.6
0.59
0.58
0.57
0.56
0.55
MOLE FRACTION
0
10
20
30
40
50
60
70
TIME (minutes)
* PLANT DATA
SIMULATION RESULTS
ANALYZER
ERROR BAND

Figure 7.8

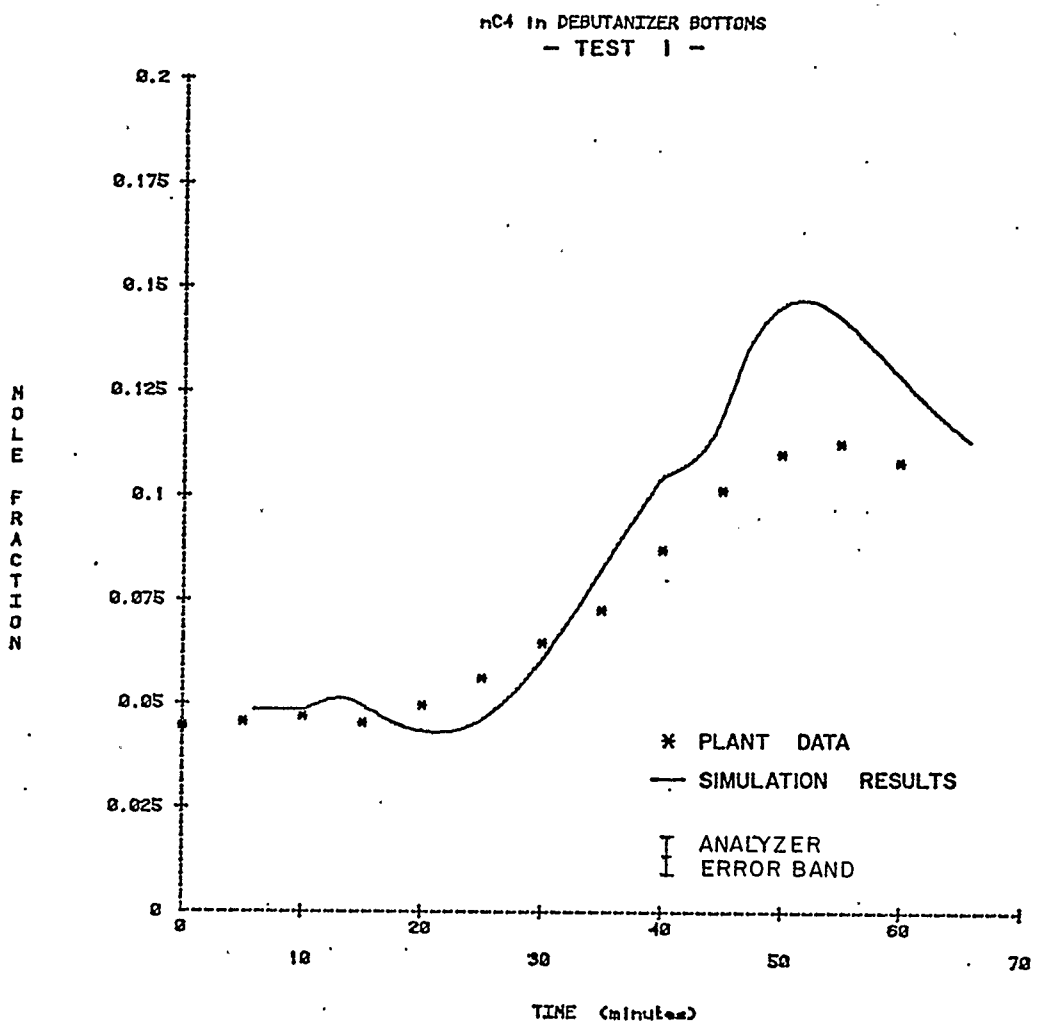

nC4 in DEBUTANIZER BOTTOMS
— TEST 1 —
MOLE FRACTION
0.2
0.175
0.15
0.125
0.1
0.075
0.05
0.025
0
0
10
20
30
40
50
60
70
TIME (minutes)
* PLANT DATA
SIMULATION RESULTS
ANALYZER
ERROR BAND

Figure 7.9

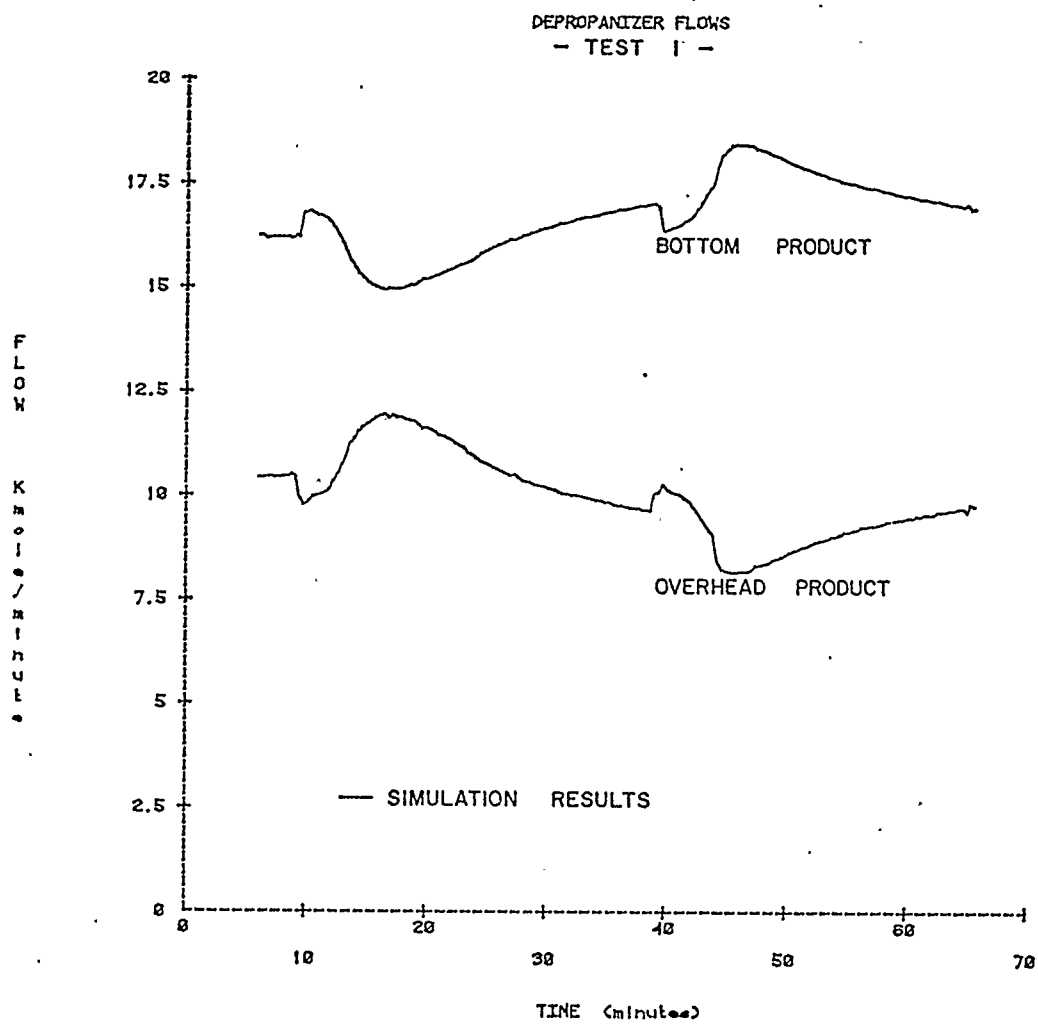

DEPROPANIZER FLOWS
- TEST I -
20
17.5
15
12.5
10
7.5
5
2.5
0
FLOW Kmole/minute
BOTTOM PRODUCT
OVERHEAD PRODUCT
SIMULATION RESULTS
0
10
20
30
40
50
60
70
TIME (minutes)

Figure 7.10

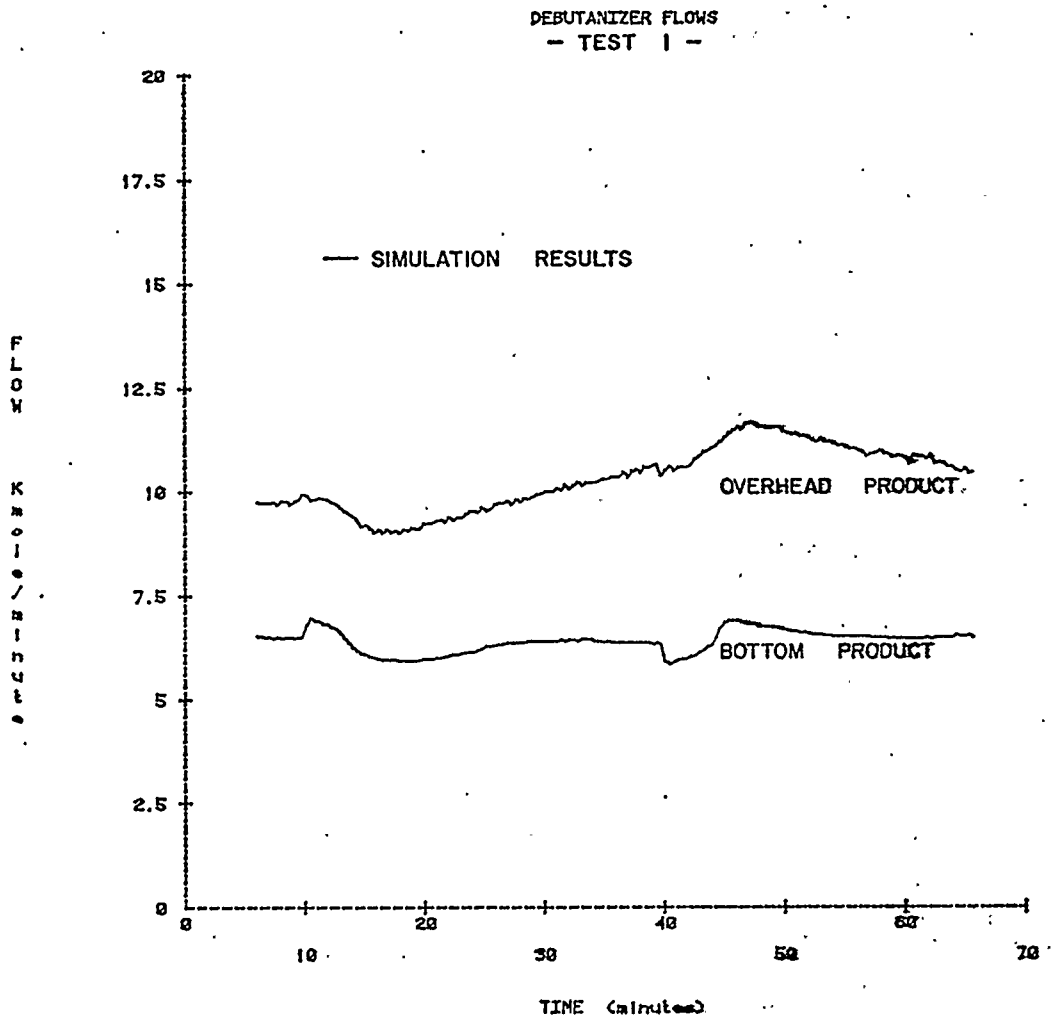

DEBUTANIZER FLOWS
— TEST 1 —
— SIMULATION RESULTS
OVERHEAD PRODUCT
BOTTOM PRODUCT
FLOW Kmole/minute
TIME (minutes)
20
17.5
15
12.5
10
7.5
5
2.5
0
0
20
40
60
10
30
50
70

Figure 7.11

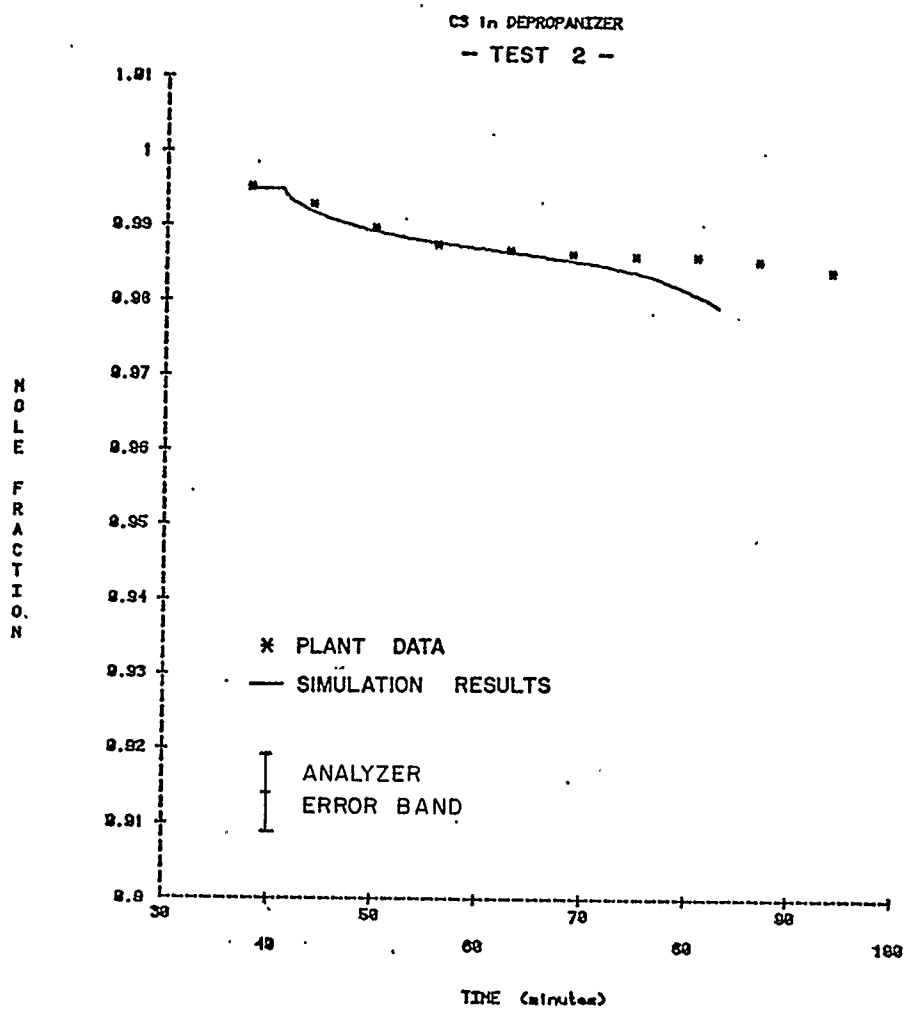

C3 in DEPROPANIZER
- TEST 2 -
1.01
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.9
MOLE FRACTION
* PLANT DATA
SIMULATION RESULTS
ANALYZER
ERROR BAND
30
40
50
60
70
80
90
100
TIME (minutes)

Figure 7.12

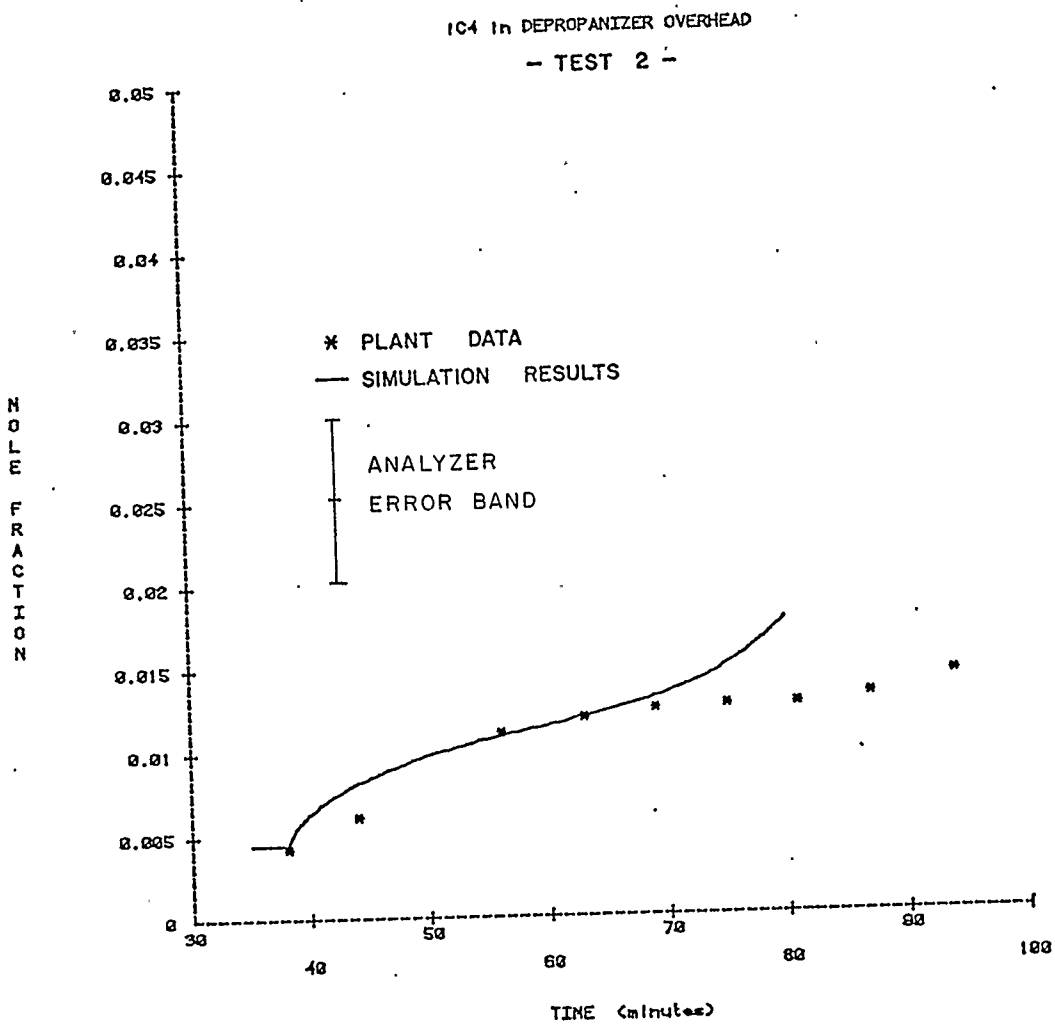

IC4 in DEPROPANIZER OVERHEAD
- TEST 2 -
* PLANT DATA
— SIMULATION RESULTS
ANALYZER
ERROR BAND
MOLE FRACTION
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
30
40
50
60
70
80
90
100
TIME (minutes)

Figure 7.13

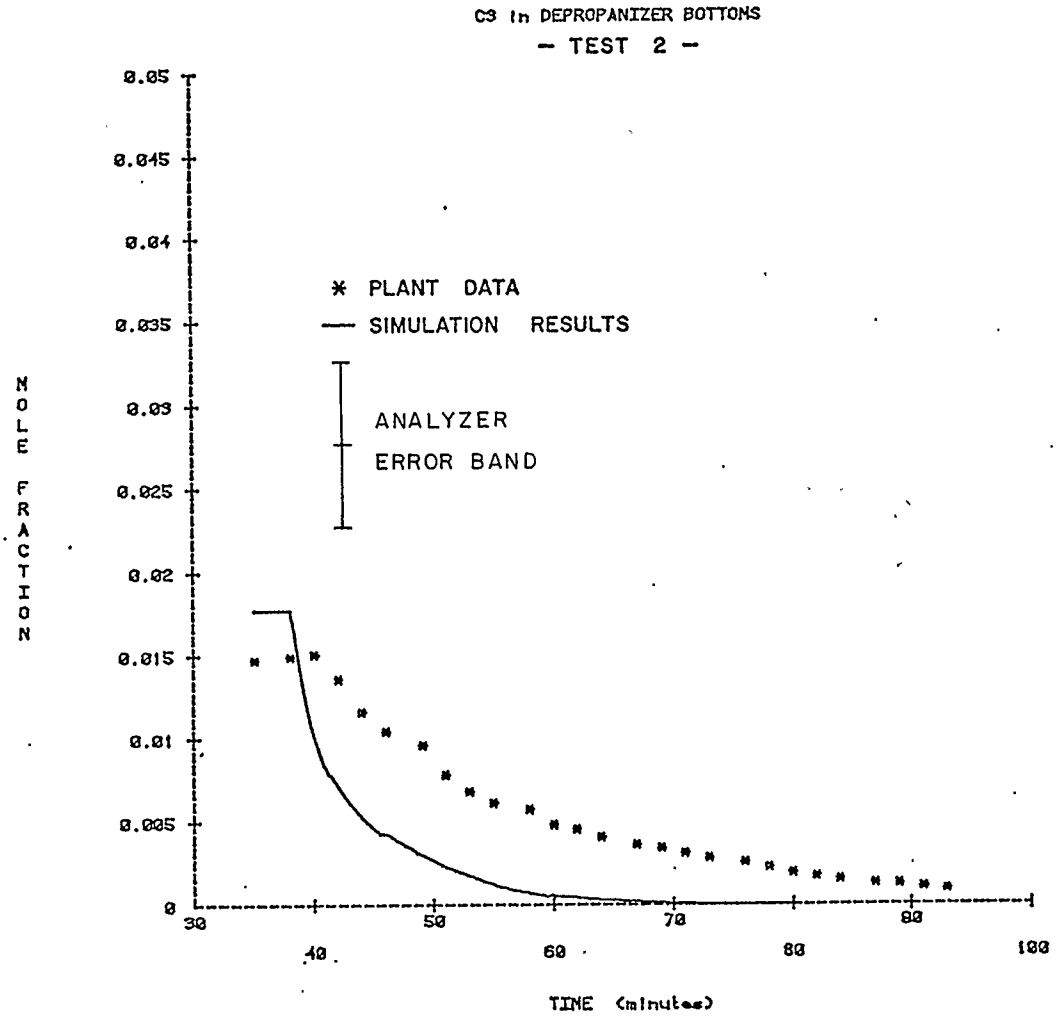

C3 in DEPROPANIZER BOTTOMS
— TEST 2 —
* PLANT DATA
— SIMULATION RESULTS
ANALYZER
ERROR BAND
MOLE FRACTION
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
30
40
50
60
70
80
90
100
TIME (minutes)

Figure 7.14

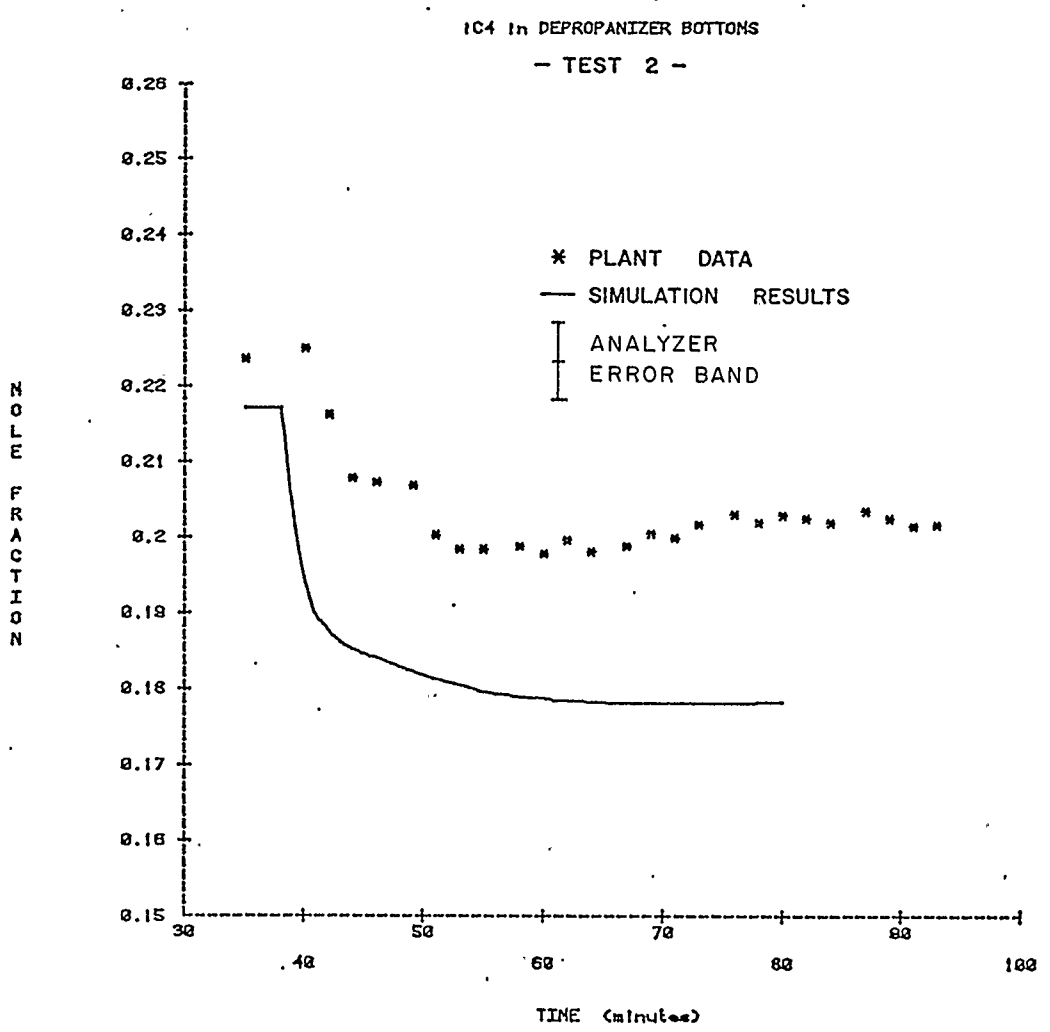

IC4 in DEPROPANIZER BOTTOMS
- TEST 2 -
* PLANT DATA
— SIMULATION RESULTS
ANALYZER
ERROR BAND
MOLE FRACTION
0.26
0.25
0.24
0.23
0.22
0.21
0.2
0.19
0.18
0.17
0.16
0.15
30
40
50
60
70
80
90
100
TIME (minutes)

Figure 7.15

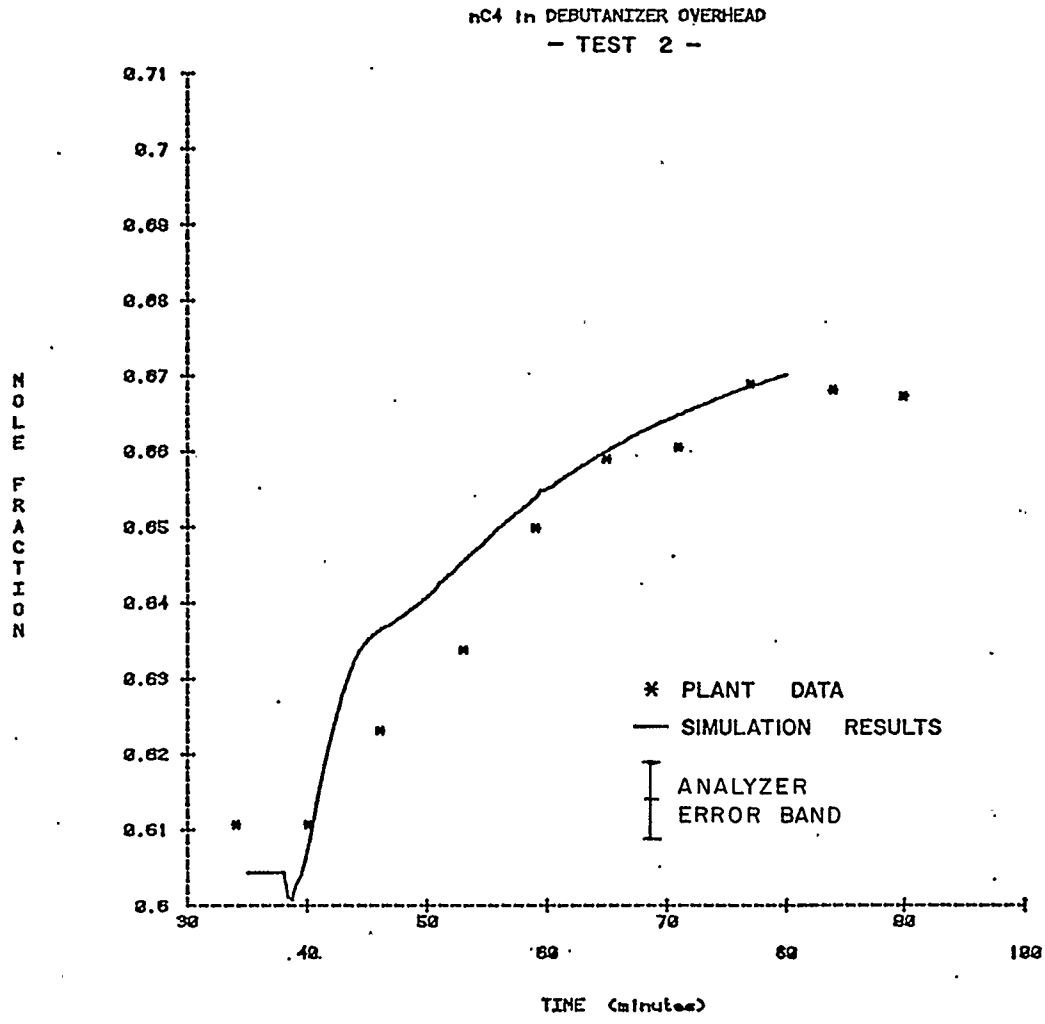

nC4 in DEBUTANIZER OVERHEAD
— TEST 2 —
0.71
0.7
0.69
0.68
0.67
0.66
0.65
0.64
0.63
0.62
0.61
0.6
MOLE FRACTION
30
40
50
60
70
80
90
100
TIME (minutes)
* PLANT DATA
— SIMULATION RESULTS
ANALYZER
ERROR BAND

Figure 7.16

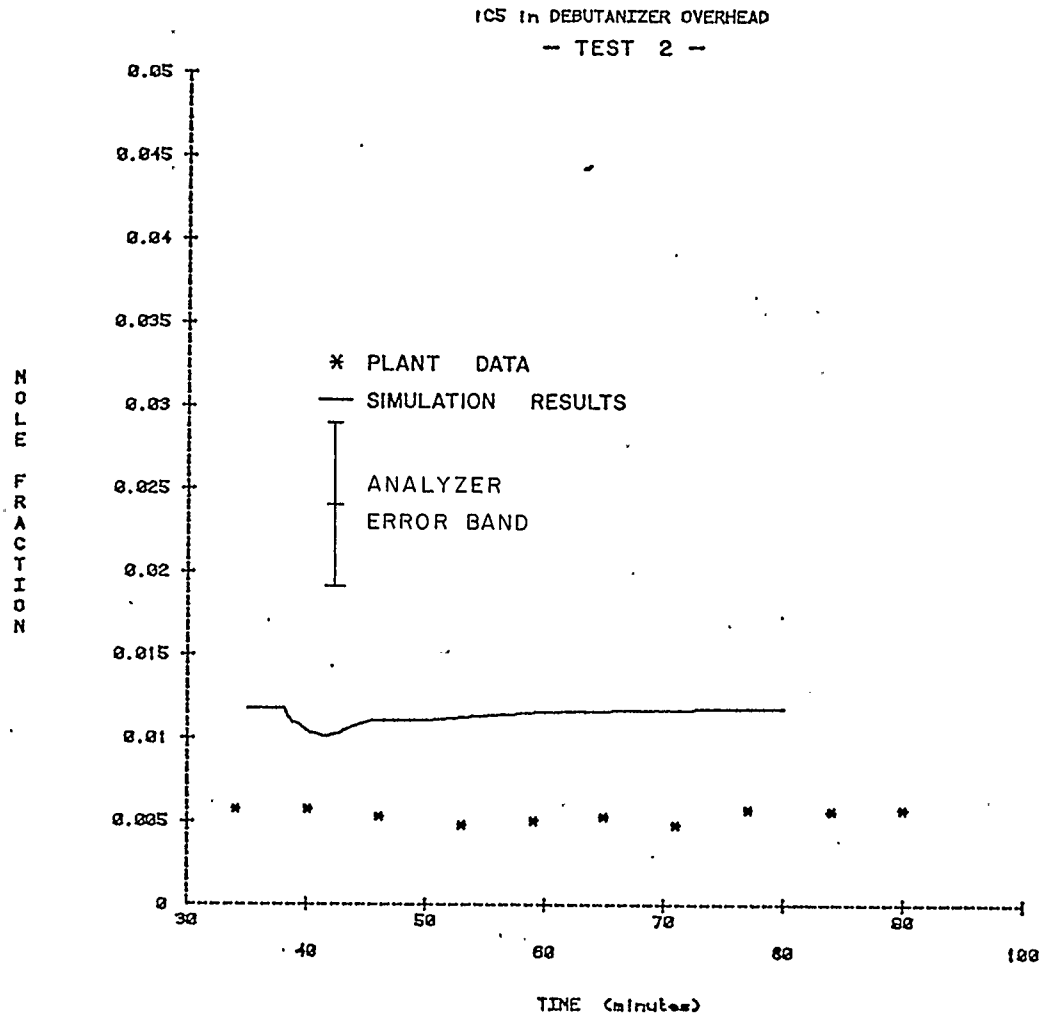

IC5 in DEBUTANIZER OVERHEAD
— TEST 2 —
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
MOLE FRACTION
* PLANT DATA
— SIMULATION RESULTS
ANALYZER
ERROR BAND
30
40
50
60
70
80
90
100
TIME (minutes)

Figure 7.17

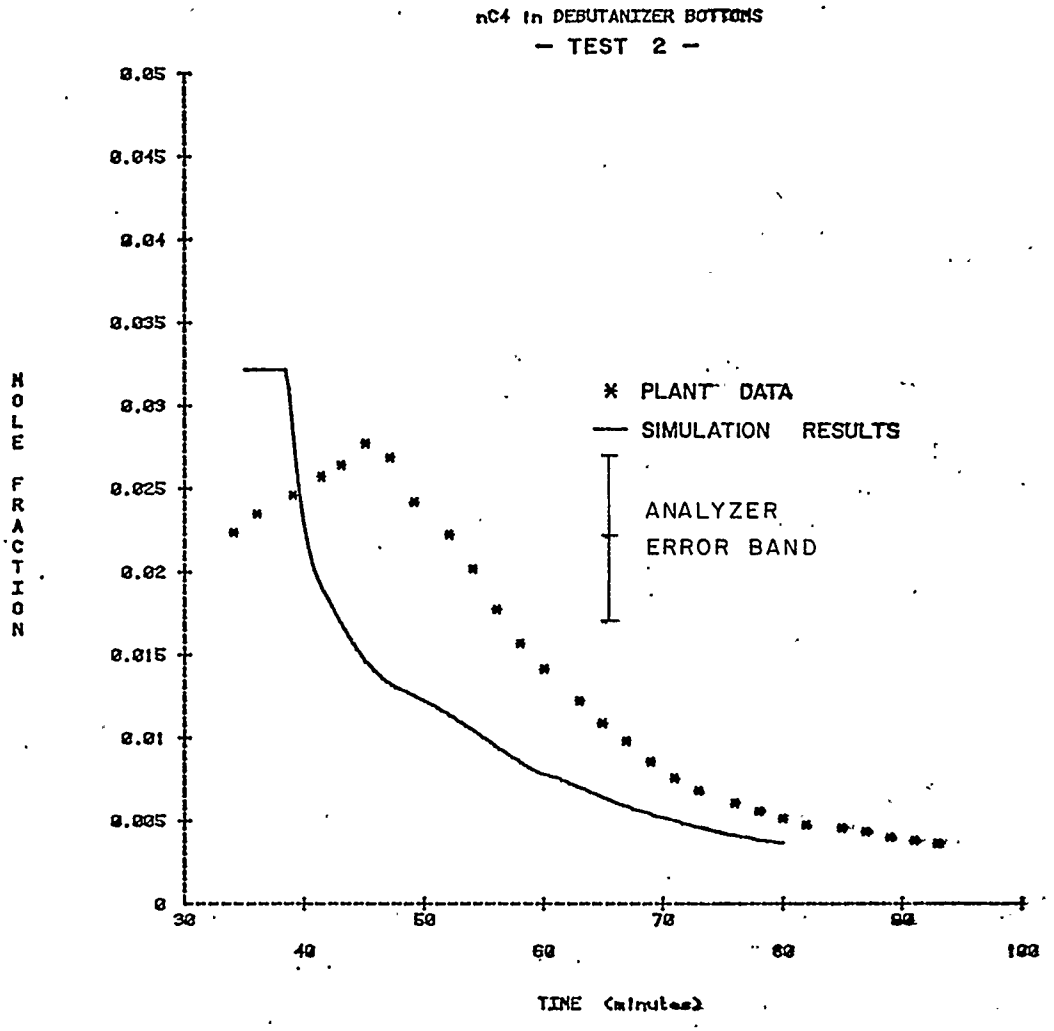

nC4 in DEBUTANIZER BOTTOMS
— TEST 2 —
MOLE FRACTION
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
30
40
50
60
70
80
90
100
TIME (minutes)
* PLANT DATA
SIMULATION RESULTS
ANALYZER
ERROR BAND

Figure 7.18

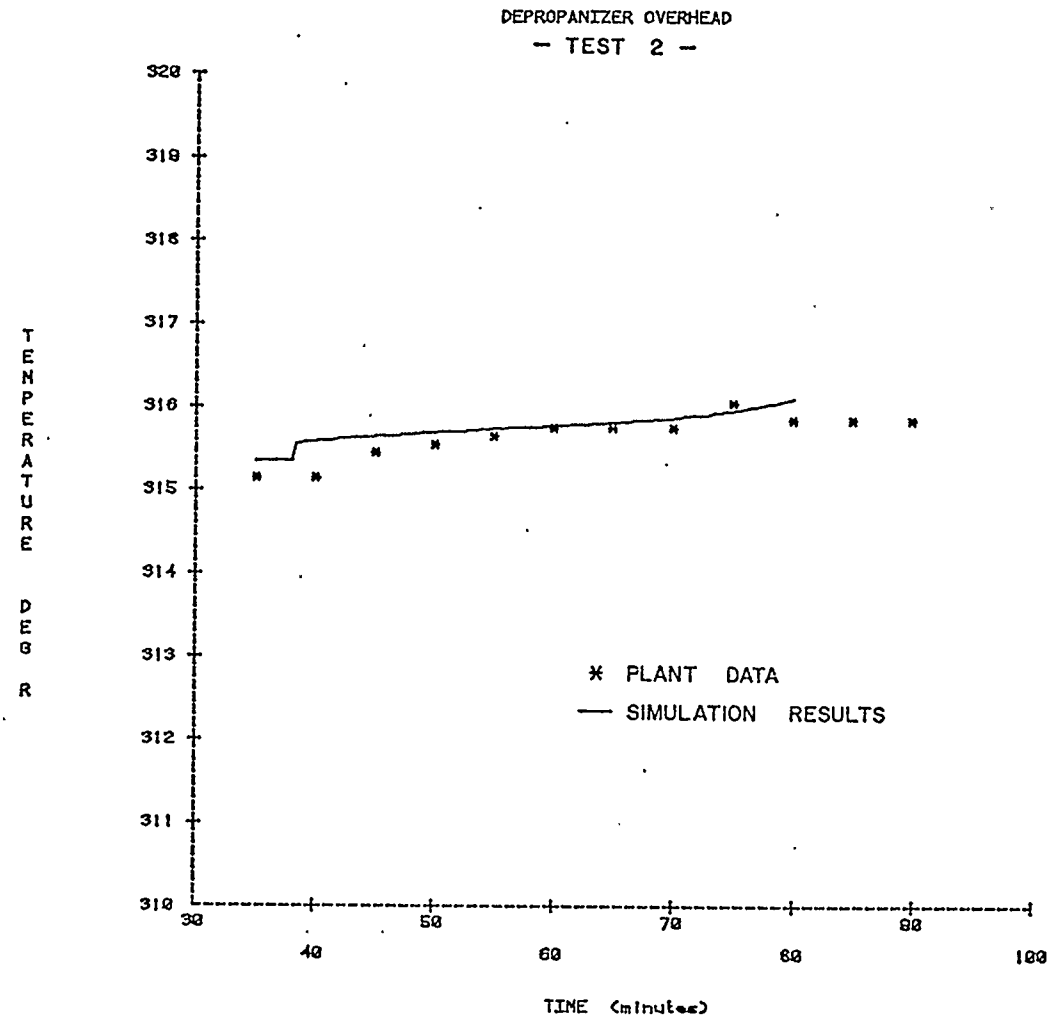

DEPROPANIZER OVERHEAD
— TEST 2 —
320
319
318
317
316
315
314
313
312
311
310
TEMPERATURE DEG R
* PLANT DATA
SIMULATION RESULTS
30
40
50
60
70
80
90
100
TIME (minutes)

Figure 7.19

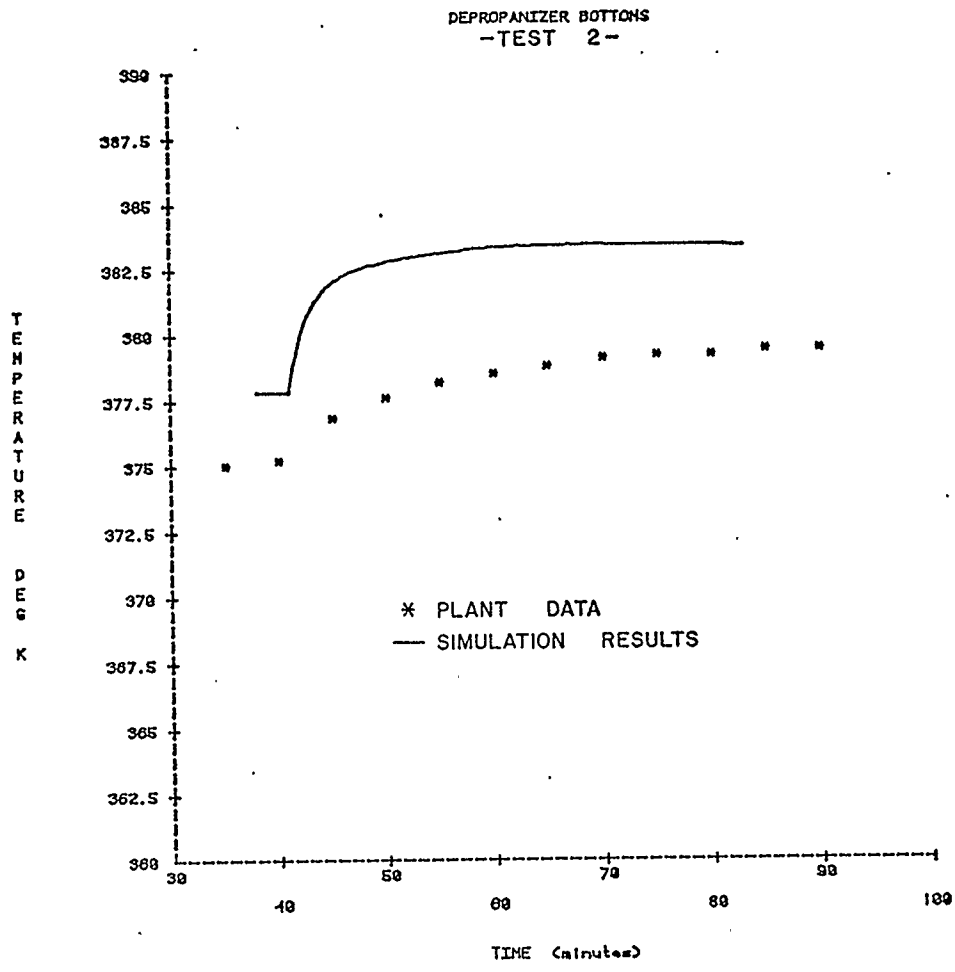

DEPROPANIZER BOTTOMS
-TEST 2-
390
387.5
385
382.5
380
377.5
375
372.5
370
367.5
365
362.5
360
TEMPERATURE DEG K
* PLANT DATA
— SIMULATION RESULTS
30
40
50
60
70
80
90
100
TIME (minutes)

Figure 7.20

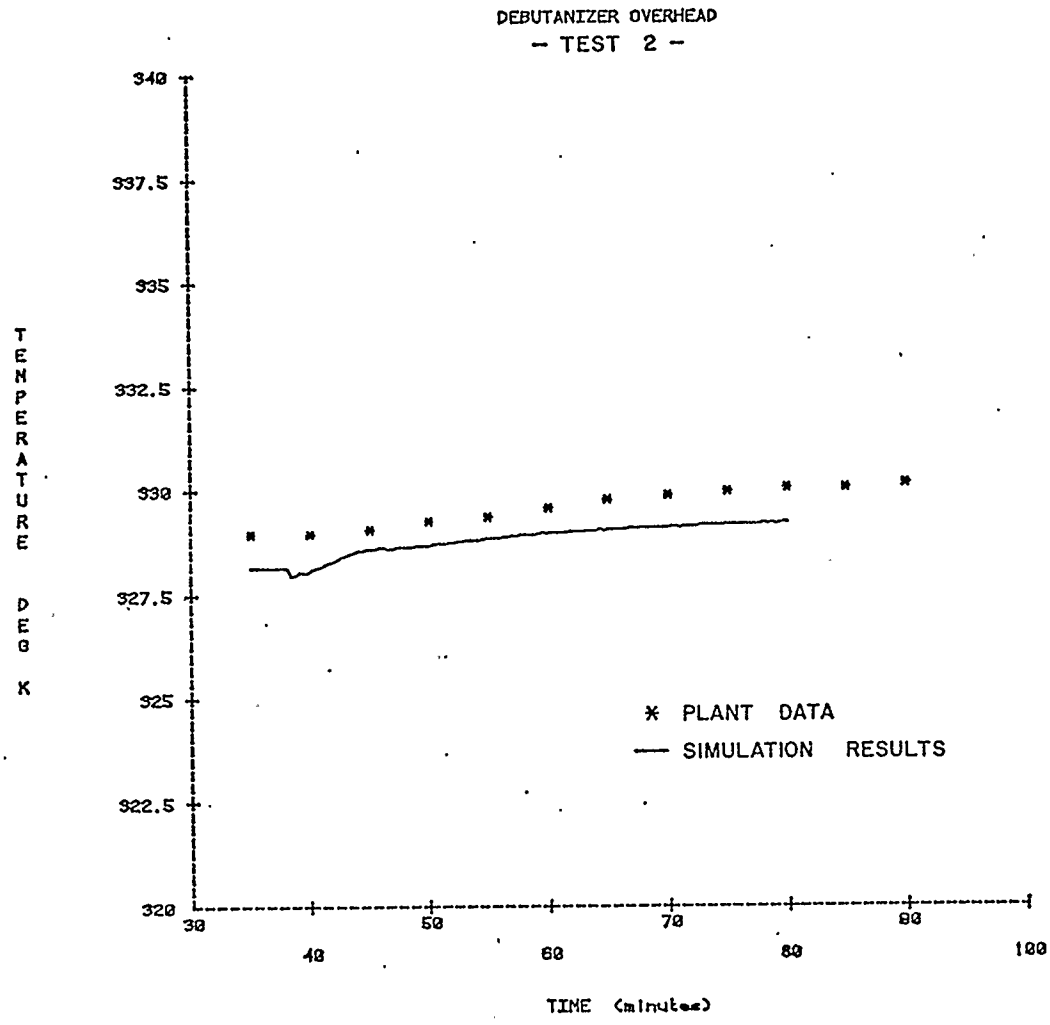

DEBUTANIZER OVERHEAD
- TEST 2 -
340
337.5
335
332.5
330
327.5
325
322.5
320
TEMPERATURE DEG K
30
40
50
60
70
80
90
100
TIME (minutes)
* PLANT DATA
— SIMULATION RESULTS

Figure 7.21

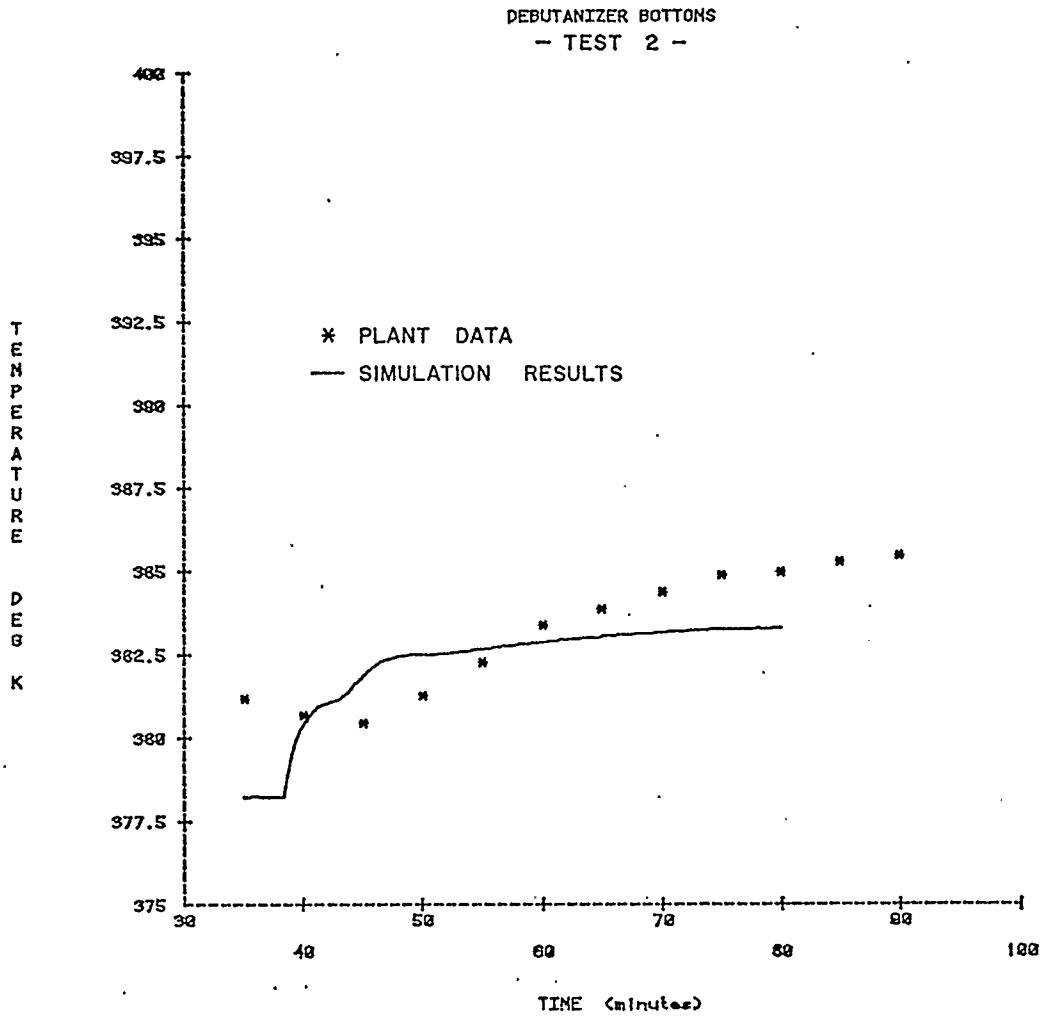

DEBUTANIZER BOTTONS
- TEST 2 -
400
397.5
395
392.5
390
387.5
385
382.5
380
377.5
375
TEMPERATURE DEG K
* PLANT DATA
SIMULATION RESULTS
30
40
50
60
70
80
90
100
TIME (minutes)

Figure 7.22

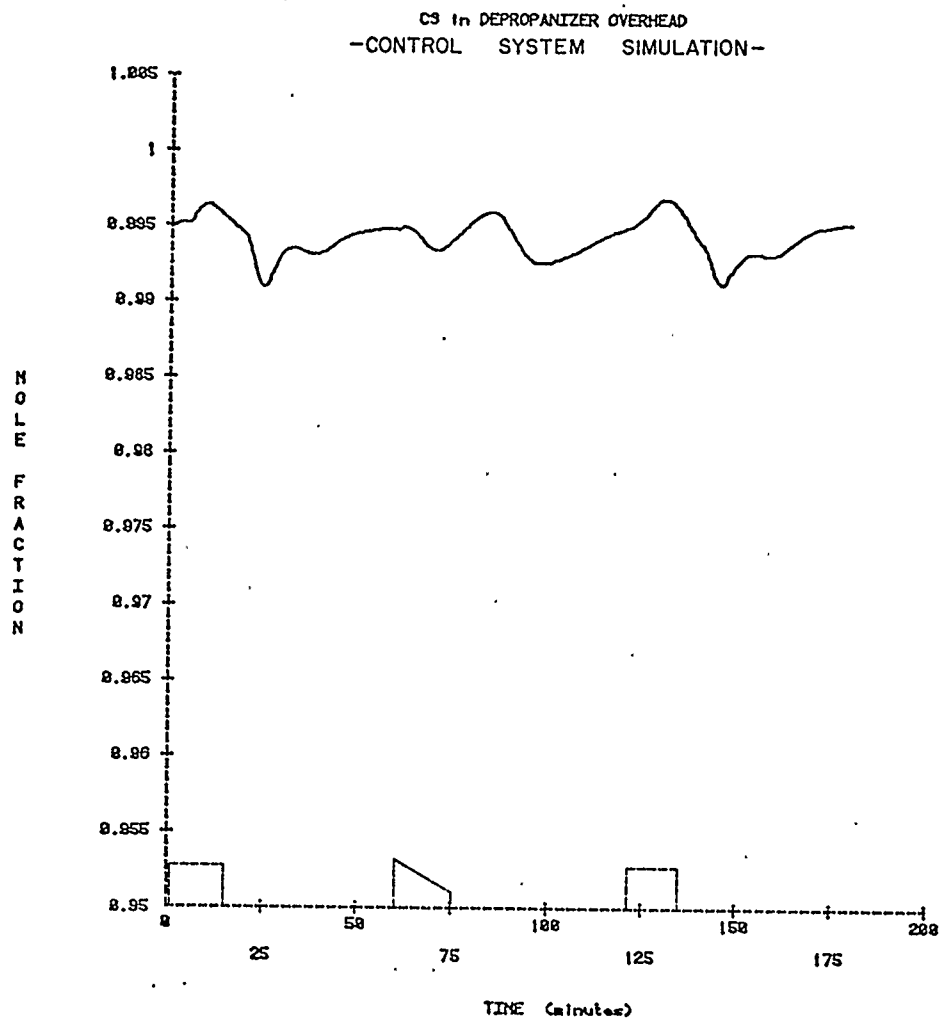

C3 in DEPROPANIZER OVERHEAD
-CONTROL SYSTEM SIMULATION-
1.005
1
0.995
0.99
0.985
0.98
0.975
0.97
0.965
0.96
0.955
0.95
MOLE FRACTION
0
25
50
75
100
125
150
175
200
TIME (minutes)

Figure 7.23

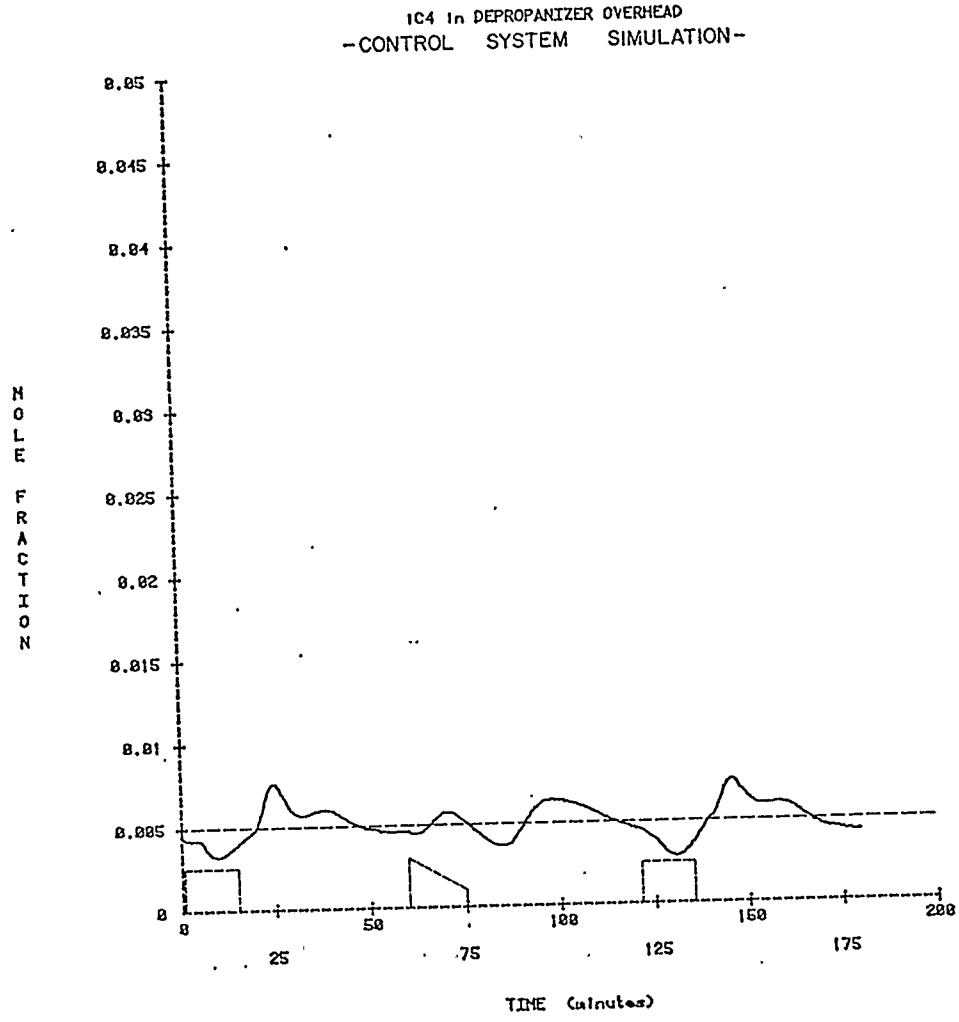

IC4 In DEPROPANIZER OVERHEAD
-CONTROL SYSTEM SIMULATION-
MOLE FRACTION
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
25
50
75
100
125
150
175
200
TIME (minutes)

Figure 7.24

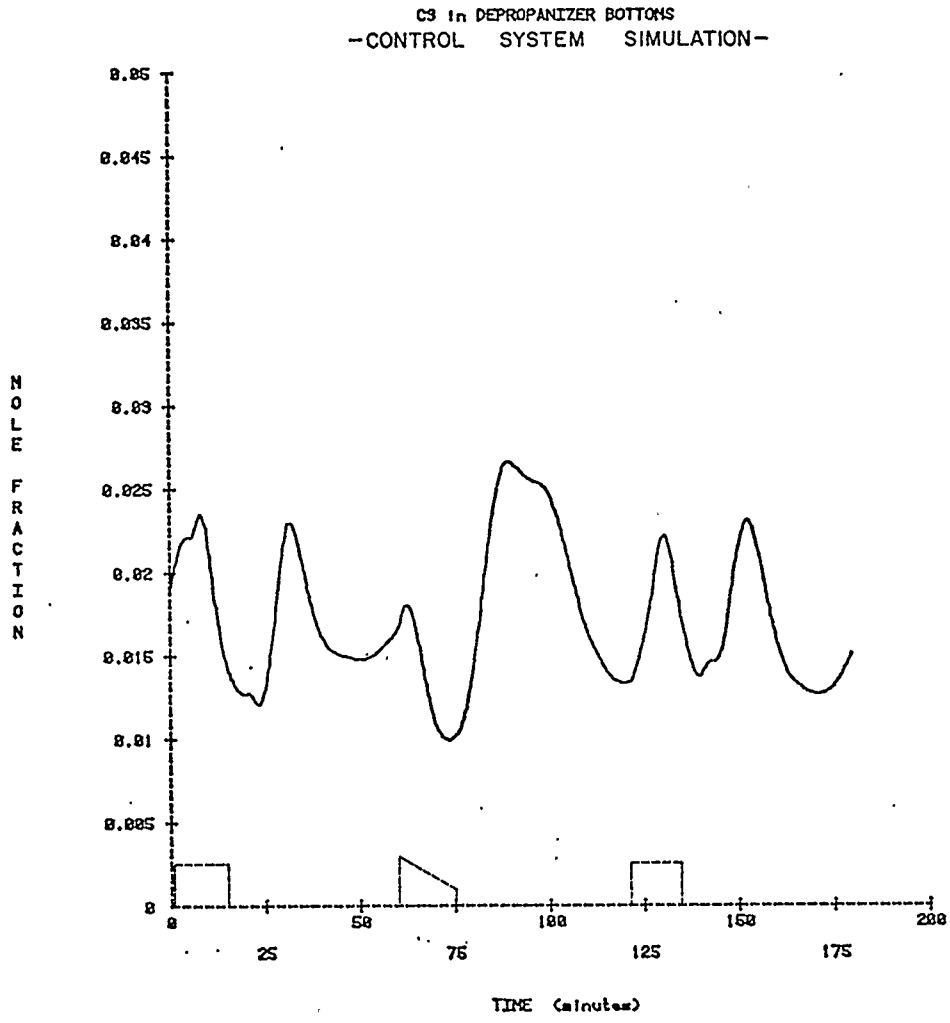

C3 in DEPROPANIZER BOTTOMS
-CONTROL SYSTEM SIMULATION-
MOLE FRACTION
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
50
100
150
200
25
75
125
175
TIME (minutes)

Figure 7.25

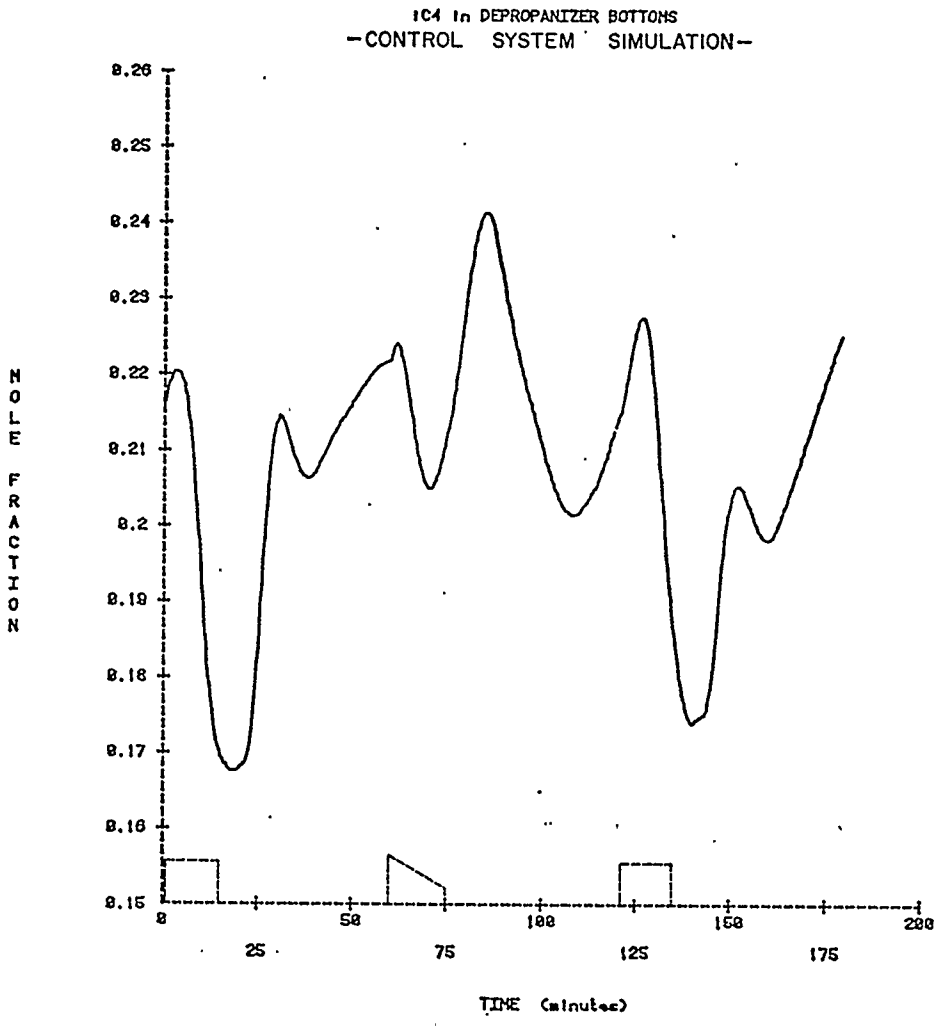

iC4 in DEPROPANIZER BOTTOMS
-CONTROL SYSTEM SIMULATION-
MOLE FRACTION
0.26
0.25
0.24
0.23
0.22
0.21
0.2
0.19
0.18
0.17
0.16
0.15
0
50
100
150
200
25
75
125
175
TIME (minutes)

Figure 7.26

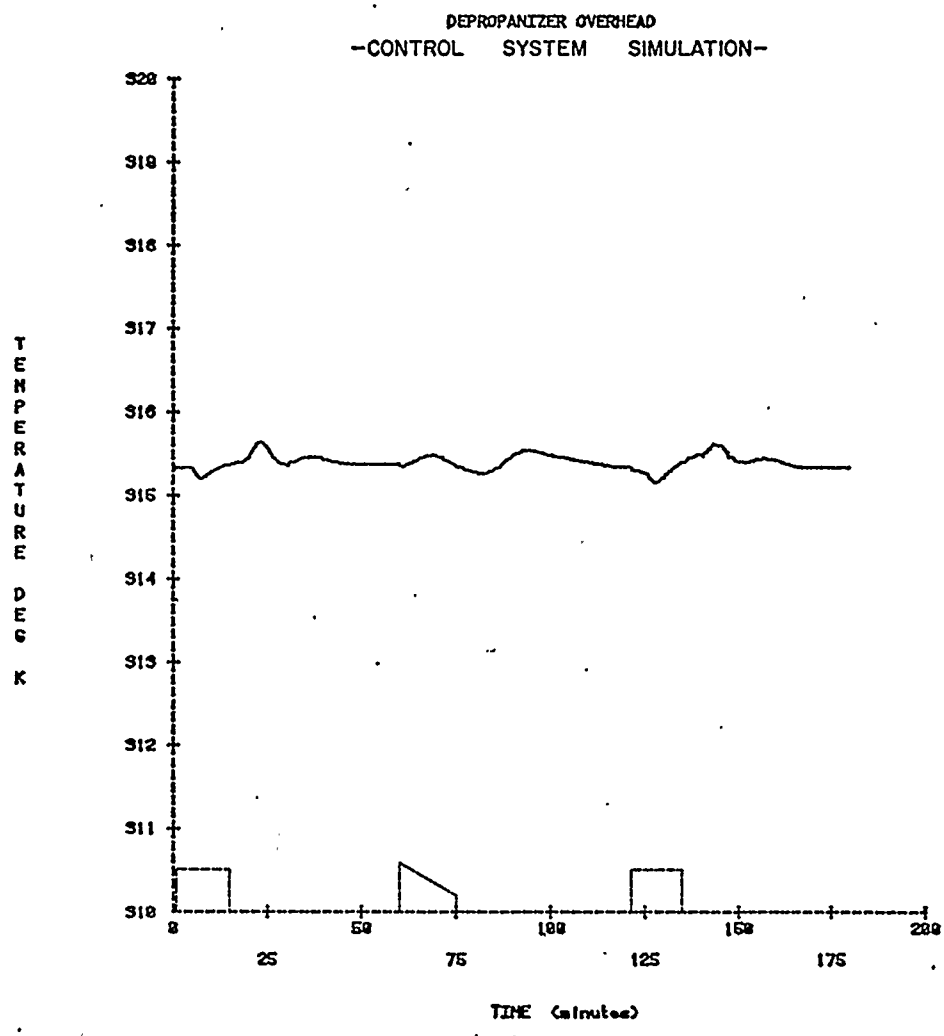

DEPROPANIZER OVERHEAD
-CONTROL SYSTEM SIMULATION-
320
319
318
317
316
315
314
313
312
311
310
TEMPERATURE DEG K
0
50
100
150
200
25
75
125
175
TIME (minutes)

Figure 7.27

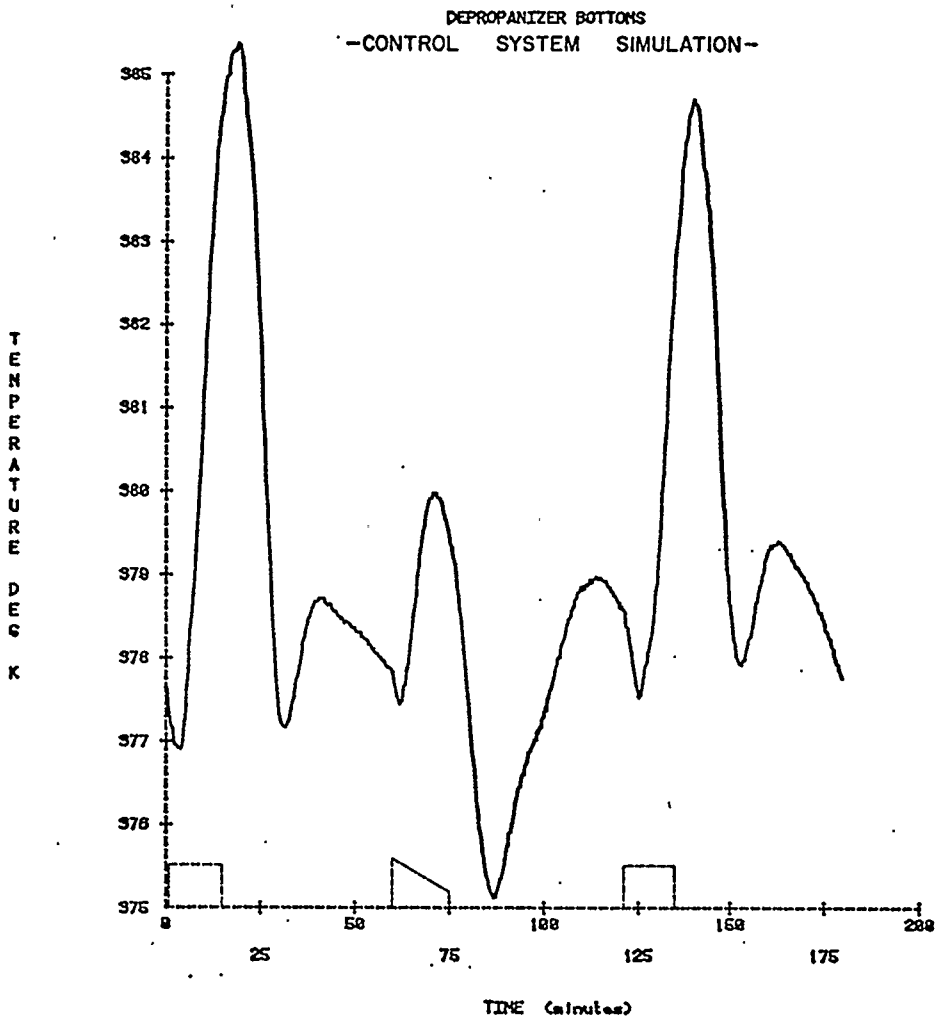

DEPROPANIZER BOTTOMS
-CONTROL SYSTEM SIMULATION-
TEMPERATURE DEG K
385
384
383
382
381
380
379
378
377
376
375
0
25
50
75
100
125
150
175
200
TIME (minutes)

Table 1.1

Equation-Oriented Executive Programs For Dynamic Simulation

(From Barney, 1975)

Acronym	Date	Institution Where Developed
MIMIC	1965	Wright Paterson AFB
CSMP	1967	IBM
IMP	1972	University of Connecticut

Table 3.1

TEST SYSTEM	METHOD EULER	STEP SIZE	ERROR	AMOS	ERROR	RKM	ERROR	GEAR NON-STIFF	ERROR
I	0.006	1.9×10^{-3}	9.50×10^{-4}	0.872	1.88×10^{-3}	0.301	3.72×10^{-4}	2.30	1.13×10^{-3}
II	71.7	1.9×10^{-6}	9.51×10^{-8}	866.0	1.01×10^{-3}	300.3	1.36×10^{-4}	2188.0	3.81×10^{-4}
III	0.325	1.9×10^{-3}	4.28×10^{-3}	2.06	4.29×10^{-3}	1.18	3.55×10^{-3}	4.90	4.05×10^{-3}
IV	1.47	1.9×10^{-3}	9.62×10^{-6}	9.09	5.65×10^{-4}	6.02	7.27×10^{-6}	36.4	2.89×10^{-5}
V	17.1	5.0×10^{-4}	3.91×10^{-5}	240.0	114.6 (4.70×10^{-3})	79.0	154.6 (5.21×10^{-4})	431.6	0.368 (3.95×10^{-4})
VI	U[1] 23.0 (0-10)	2.5×10^{-4} 1.0×10^{-3}	4.21×10^{-4}	>1500 101.0 (0-10)	9.95×10^{-3}	>1000 48.0 (0-10)	1.99×10^{-4}	1925.0	1.86×10^{-4}
VII	37.6	6.0×10^{-3}	4.55×10^{-3}	353.0	0.855[2]	167.8	7.94×10^{-2}	656.0	8.80×10^{-3}
VIII	0.530	2.0×10^{-3}	9.58×10^{-3}	0.059	6.69×10^{-3}	0.021	1.89×10^{-3}	0.101	8.15×10^{-4}
IX	1.04	5.0×10^{-3}	9.48×10^{-3}	0.107	9.62×10^{-3}	0.041	1.56×10^{-2}	0.213	6.67×10^{-3}
X	0.636	2.0×10^{-3}	9.75×10^{-3}	0.059	3.48 (1.81×10^{-4})	0.022	0.394 (6.05×10^{-4})	0.164	4.39×10^{-2}
XI	1.39	2.5×10^{-3}	9.45×10^{-3}	>320		0.263	4.87×10^{-4}	1.30	2.25×10^{-4}

[1] unstable

[2] only extremely small components inaccurate

TEST SYSTEM	METHOD RLT	ERROR	NIGRO	STEP SIZE	ERROR	TREANOR	ERROR	FOWLER WARTEN	ERROR
I	0.092	4.83×10^{-3}	0.144	1.8×10^{-3}	5.35×10^{-3}	0.384	2.24×10^{-4}	0.495	2.24×10^{-3}
II	0.068	5.03×10^{-4}	26.5	1.0×10^{-5}	2.15×10^{-2}	0.393	8.90×10^{-7}	49.1	3.98×10^{-2}
III	3.32	3.81×10^{-3}	0.498	1.8×10^{-3}	6.13×10^{-3}	1.38	5.09×10^{-4}	1.48	7.49×10^{-5}
IV	0.356	5.03×10^{-3}	0.817	5.0×10^{-3}	6.88×10^{-3}	6.90	1.85×10^{-3}	0.317	1.79×10^{-2}
V	0.078	343.0 (6.58×10^{-4})	18.5	1.0×10^{-3}	1.06×10^{-2}	0.505	3.56×10^{-3}	0.624	8.42×10^{-3}
VI	36.5	8.86[2]	464.0	1.0×10^{-3}	3.35×10^{-2}	U[1] 44.5 (0-10)	4.41×10^{-2}	117.0	2.48×10^{-2}
VII	3.58	2.03[2]	46.5	1.0×10^{-2}	1.73×10^{-2}	219.0	8.24×10^{-2}	75.2	1.41×10^{-2}
VIII	0.159	8.61×10^{-2}	1.24	2.0×10^{-3}	9.72×10^{-3}	0.927	1.97×10^{-5}	0.080	2.64×10^{-2}
IX	0.317	5.45×10^{-2}	2.02	4.0×10^{-3}	1.05×10^{-2}	2.69	3.13×10^{-9}	0.077	7.07×10^{-3}
X	0.118	0.630 (2.81×10^{-2})	1.36	2.5×10^{-3}	2.64×10^{-2}	2.36	2.46×10^{-4}	0.060	1.83×10^{-3}
XI	0.744	3.00×10^{-2}	5.65	1.0×10^{-2}	1.63×10^{-2}	5.07	4.06×10^{-5}	0.666	1.86×10^{-4}

[1] unstable

[2] large components reasonably accurate

Table 3.1 (continued)

TEST SYSTEM	METHOD KLOP DAVIS	ERROR	SAND SCH.	ERROR	BRANDON	ERROR	GEAR	ERROR
I	2.27	5.85×10^{-4}	0.274	1.21×10^{-2}	0.124	1.22×10^{-3}	0.139	1.07×10^{-3}
II	>1000 385.0 (0-10)	5.00×10^{-8}	0.276	7.10×10^{-4}	0.268	3.98×10^{-4}	0.196	1.01×10^{-4}
III	7.11	2.67×10^{-3}	0.788	4.64×10^{-2}	4.24	3.97×10^{-3}	0.318	3.77×10^{-3}
IV	39.2	5.90×10^{-6}	1.75	9.41×10^{-4}	1.11	5.66×10^{-4}	0.505	1.16×10^{-5}
V	845.0	0.267 (7.56×10^{-3})	0.117	1.27×10^{-4}	0.459	3.10×10^{-5}	0.168	4.57×10^{-5}
VI	>1000 (0-10)		>1000 626.0 (0-10)	6.83×10^{-3}	>1500 761.0 (0-10)	7.72×10^{-4}	81.7	8.47×10^{-3}
VII	>1000 (0-10)		442.0	0.133	299.0	1.86×10^{-2}	42.0	1.71×10^{-2}
VIII	0.760	0.206	0.721	0.204	0.294	0.308 (0.023)	0.078	3.23×10^{-3}
IX	0.788	0.335	0.830	0.316	0.546	9.68×10^{-3}	0.192	8.07×10^{-2}
X	0.473	0.936 (3.0×10^{-2})	0.455	0.870 (2.94×10^{-2})	0.313	8.13×10^{-3}	0.119	1.38 (2.74×10^{-3})
XI	29.1	3.78×10^{-2}	29.1	3.75×10^{-2}	2.44	1.77×10^{-4}	1.03	4.85×10^{-5}

TEST SYSTEM	METHOD GEAR MINV	GEAR DECOMP-SOLVE	GEAR TRGB	ERROR	IMP	ERROR
I	0.139	0.148	0.162	1.07×10^{-3}	0.139[1]	1.85×10^{-3}
II	0.196	0.216	0.235	1.01×10^{-4}	0.373[1]	2.84×10^{-4}
III	0.318	0.294	0.315	3.77×10^{-3}	3.72[1]	1.06×10^{-5}
IV	0.505	0.489	0.656	1.16×10^{-5}	0.967[1]	2.69×10^{-2}
V	0.168	0.165	0.182	4.57×10^{-5}	27.3[2]	3.91×10^{-5}
VI	81.7	20.0	7.58	8.47×10^{-3}	>1000	
VII	42.0	12.3	6.05	1.71×10^{-2}	30.2[2]	0.169
VIII	0.078	0.079	0.080	3.23×10^{-3}	0.258[1]	0.344
IX	0.192	0.197	0.292	8.07×10^{-2}	0.584[2]	2.37×10^{-3}
X	0.119	0.128	0.130	1.38 (2.74×10^{-3})	1.12[1]	5.27×10^{-3}
XI	1.03	0.897	0.980 0.753 (TRl)	4.85×10^{-5}	11.5[1]	0.967

[1]Crout elimination, variable order, variable or constant banded matrices

[2]Gauss-Seidel

Table 4.1

Components Available in the Property Package

1	Methane
2	Ethane
3	Propane
4	iso-Butane
5	normal Butane
6	iso-Pentane
7	normal Pentane
8	Hexane
9	Heptane
10	Octane
11	Nonane
12	Nitrogen
14	Carbon Dioxide
15	Hydrogen Sulphide

Table 6.1

Process Modules Used in Simulations

Type Number	Function
1	Simple Valve
2	Counter Current Heat Exchanger
3	Stirred Tank
4	P.I.D. Controller
6	Mass Transfer Stage
8	Reboiler Kettle
14	Tee-Junction
17	Total Overhead Condenser
22	Reboiler Heat Transfer Model

References

Balzhiser, R.E., Samuels, M.R. and Eliassen, J.D., 1972:
"Chemical Engineering Thermodynamics", Prentice Hall

Barney, J.R., 1975:
"Dynamic Simulation of Large Stiff Systems in a Modular Simulation Framework", Ph.D. Thesis, Dept. of Chemical Engineering, McMaster University

Bernard, J.D.T., and Sargent, R.W.H., 1966:
"The Hydrodynamics Performance of a Sieve Plate Distillation Column", Trans. Instn. Chem. Engrs. 44,T314

Bishnoi, P.R., 1978:
Private communication

Brandon, D.M., 1974:
"A New Single-Step Implicit Integration Algorithm with A-Stability and Improved Accuracy", Simulation 23, 1, 17

Brayton, R.K., Gustavson, F.G., and Hachtel, G.D., 1972:
"A New Efficient Algorithm for Solving Differential-Algebraic Systems Using Implicit Backward Differentiation Formulas", Proc. I.E.E.E. 60, 1, 98

Burden, R.L., Faires, J.D., and Reynolds, A.C., 1978:
"Numerical Analysis", Prindle, Weber and Schmidt

Bush, M., 1978:
Private communication

Byrne, G.D., and Hindmarsh, A.C., 1975:
"A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations", ACM Trans. on Math. Software 1,71

Calahan, D.A. 1968:
"A Stable, Accurate Method of Numerical Integration for Nonlinear Systems", Proc. I.E.E.E. (Letters), 56, 744

D'Arcy, D. 1978: "Analysis of Sieve Tray Froths Such as Occur in Heavy Water Production Plants", Atomic Energy of Canada Limited paper.

Distefano, G.P., 1968:
"Mathematical Modelling and Numerical Integration of Multicomponent Batch Distillation Equations", A.I.Ch.E. Journal 14,1,190.

Emanuel, G., 1967:
Comments on the paper "A Chemical Kinetics Computer Program for Homogeneous and Free-Radical Systems of Reactions", by R.H. Snow, J. Phys. Chem. 71, 4, 1161

Enright, W.H., 1974:
"Second Derivative Multistep Methods for Stiff Ordinary Differential Equations", SIAM J. Numer. Anal., 11, 321

Enright, W.H. and Hull, T.E. 1976: "Comparing Numerical Methods for the Solution of Stiff Systems of ODE's Arising in Chemistry", paper presented in "Numerical Methods for Differential Systems", edited by L. Lapidus and W.E. Schiesser, Academic Press.

Evelien, K. 1976:
"Prediction of Complex Phase Behaviour", M.Sc. Thesis, Dept. of Chemical Engineering, University of Calgary

Franks, R.G.E., 1972:
"Modeling and Simulation in Chemical Engineering", Wiley-Interscience.

Gear, C.W., 1971:
"Algorithm 407, DIFSUB for Solution of Ordinary Differential Equations", Comm. ACM 14, 3, 185

Ham, P.G., 1971:
"The Transient Analysis of Integrated Chemical Processes", Ph.D. Thesis, Dept. of Chemical Engineering, University of Pennsylvania

Hindmarsh, A.C., 1974:
"GEAR: Ordinary Differential Equation Solver", UCID-3001, Rev. 3, Lawerence Livermore Laboratory, University of California, Livermore

Hronsky, P. and Martens, H.R., 1973:
"Computer Techniques for Stiff Differential Equations", Proc. of the Summer Computer Simulation Conference, Montreal

Hui, A.T.K., 1977:
"Dynamics and Control of an Extractive Distillation Column", M.Sc. Thesis, Dept. of Chemical Engineering, McMaster University

Ingels, D.M. 1970:
"A System for Simulating Chemical Process Dynamics and Control", Ph.D. Thesis, Dept. of Chemical Engineering, University of Houston

Kolbrack, R., 1967:
"Computer Program for Chemical Kinetics", J. Phys. Chem., 71, 4, 1162

Koenig, D.M., 1972:
"OSUSIM: A Modular Approach to Dynamic Simulation", Ph.D. Thesis, Dept. of Chemical Engineering, Ohio State University

Lapidus, L. and Seinfeld, J.H., 1971:
"Numerical Solution of Ordinary Differential Equations", Academic Press.

Lawson, J.D. and Ehle, B.L. 1972:
"Improved Generalized Runge Kutta", Proceedings of the Canadian Computer Conference, Session 72

Liniger, W. and Willoughby, R.A., 1967: "Efficient Numerical Integration of Stiff Systems of Ordinary Differential Equations", IBM Research Report RL-1970

McCune, L.C. and Gallier, P.W. 1973:
"Digital Simulator: A Tool for the Analysis and Design of Distillation Controls", ISA Trans., 12, 3, 193

Merson, R.H., 1957:

"An Operational Method for the Study of Integration Processes", Proc. of Conference on Data Processing and Automatic Computing Machines, Weapons Research Establishment, Salisbury, Australia, June 3-8, p110

Millares, R., 1975:
M.E.Sc Thesis, Dept. of Chemical Engineering, The University of Western Ontario

Passut, C.A. and Danner, R.P., 1972:
"Correlation of Ideal Gas Entropy, Heat Capacity, and Entropy", Ind. Eng. Chem. Process Des. and Develop., 11, 4, 543

Peiser, A.M. and Grover, S.S., 1962:
"Dynamic Simulation of a Distillation Tower", Chem. Eng. Progress, 58, 9, 65

Peng, D.Y. and Robinson, D.B., 1976:
"A New Two-Constant Equation of State", Ind. Eng. Chem., Fundem., 15, 1, 59

Pulido, J., 1975:
"Application of DYNSYS 2.0 to the Dynamics Simulation of Distillation Columns", M.E.Sc. Thesis, Dept. of Chemical Engineering, University of Western Ontario

Rademaker, O., Rijnsdorp, J.E. and Maarleveld, A., 1975:
"Dynamics and Control of Continuous Distillation Units", Elseveir Scientific

Richards, P.I., Lanning, W.D. and Torrey, M.D., 1966:
"Numerical Integration of Highly-Damped Nonlinear Systems", SIAM Rev., 7, 3, 376

Seinfeld, J.H., Lapidus, L. and Hwang, M., 1970:
"Review of Numerical Integration Techniques for Stiff Ordinary Differential Equations", Ind. Eng. Chem., Fundem., 9, 2, 266

Shah, M.K., 1977:
"Development of a Procedure for Use of an Equation of State in Computations of Multicomponent Seperation Processes",

M.Sc. Thesis, Dept. of Chemical Engineering, University of Calgary

Simonsmeier, U.F., Bilec, R. and Wood, R.K. 1977:
"Rigorous Dynamic Models of a Binary Distillation Column", presented at the 27th Canadian Chemical Engineering Conference, Oct. 23-27, Calgary.

Snow, R.H., 1967:
Reply to comments on the paper "A Chemical Kinetics Program for Homogeneous and Free-Radical Systems of Reactions", J. Phys. Chem., 71, 4, 1162

Strand, C.P., 1963:
"Bubble Cap Tray Efficiencies", Chem. Eng. Progress, 59,4,58.

Svrcek, W.Y., 1967:
"dynamic Response of a Binary Distillation Column", Ph.D. Thesis, Dept. of Chemical Engineering, University of Alberta.

Tetlow, N.J., Groves, D.M. and Holland, C.D., 1967:
"A Generalized Model for the Dynamic Behaviour of a Distillation Column", A.I.Ch.E. Journal, 13, 3, 476

Thorogood, R.M., 1963:
"Mass Transfer Theory in Distillation", British Chem. Eng. 8,3,164.

Treanor, C.E., 1966:
"A Method for the Numerical Integration of Coupled First-Order Differential Equations with Greatly Different Time Constants", Math. Comp. 20, 39