DYNAMIC SIMULATION OF

LARGE STIFF SYSTEMS
DYNAMIC SIMULATION
OF LARGE STIFF SYSTEMS
IN A MODULAR SIMULATION FRAMEWORK

by

JAMES ROBERT BARNEY, B.A.Sc. M.A.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Doctor of Philosophy

McMaster University

May 1975

© James Robert Barney 1975
DOCTOR OF PHILOSOPHY (1975)  McMaster UNIVERSITY
(Chemical Engineering)  Hamilton, Ontario

TITLE: Dynamic Simulation of Large Stiff Systems in a
Modular Simulation Framework

AUTHOR: James Robert Barney, B.A.Sc. (University of
        Waterloo)  
         M.A.Sc. (University of
        Waterloo)

SUPERVISOR: Professor A.I. Johnson

NUMBER OF PAGES: xiv, 362
ABSTRACT

"DYNSYS" is a digital simulation package for modelling the dynamic behaviour and automatic control of complex industrial systems. An acronym for Dynamic Systems Simulator, it was developed in the late 1960s in the chemical engineering department at McMaster University. The underlying principle of "DYNSYS" is that of modularity, i.e., the user assembles mathematical models of process units and control devices to build the process whose transient behaviour is to be studied. A fundamental aspect of dynamic simulation is the numerical solution of ordinary differential equations (o.d.e.s). The original version of "DYNSYS" used a third-order Adams-Moulton-Shell routine; however, this is not sufficient to handle stiff systems, i.e., systems where the time constants differ greatly in magnitude. In chemical engineering, stiff o.d.e.s occur widely in reaction kinetics and to some extent in multistage systems.

Conventional numerical techniques are restricted by stability to using a very small step size resulting in large computer times. There have been many new numerical techniques published in the recent literature directed at
the efficient numerical solution of systems of stiff o.d.e.s. A literature survey of these has been made.

Numerical testing of several methods indicated Gear's method to be superior. It is a variable order, variable step, linear, multistep method.

Most stiff techniques are implicit and require a technique such as Newton-Raphson iteration to converge. Each iteration involves the solution of a system of linear algebraic equations (usually sparse) equal in size to the number of o.d.e.s. For a large stiff system, this requires considerable computer time. Various sparse linear equation solvers have been evaluated and that of Bending and Hutchison appears to be the most efficient. Their routine stores and operates on only the nonzero elements of the equations. When the equations are solved for the first time, a string of integers called the "operator list" is created which stores the particular solution process by Gaussian elimination. If the system is re-solved using the operator list, the amount of computer time required is greatly diminished. If the zero elements remain zero and the nonzero elements change, the same "operator list" can be used to solve the new system. This is essentially what occurs during numerical integration. The operator list could be set up on the first integration step and used on later steps to solve each new linear system.
Gears integration algorithm in conjunction with the Bending-Hutchison linear equation solve has been implemented into DYNYSYS version 2.0. An option for stiff systems with tridiagonal Jacobian matrix is also included. The procedure for writing modules is outlined.

Four small examples are presented to illustrate the new executive:

(1) The level control of a stirred tank system (nonstiff) with time delay.

(2) A network of 15 stirred tank reactors, stiff and nonstiff, 2 o.d.e.s per reactor.

(3) A tubular reactor with 222 stiff o.d.e.s resulting from the discretization of the partial differential equations.

(4) A tubular reactor with 49 stiff o.d.e.s with tri-diagonal Jacobian matrix.

A simulation of a fictitious chemical plant proposed by Williams and Otto is also described.
ACKNOWLEDGEMENTS

The study described herein was supervised by Dr. A.I. Johnson, Dean of Engineering Science at The University of Western Ontario. The author is grateful to him and to many of his associates for their assistance.

Discussion and correspondence with D.M. Brandon of Control Data Corporation was very helpful.

The author was aided by a National Research Council Scholarship, a Shell Canada Engineering Fellowship and a grant from Control Data Corporation.

The computational work was done on the CDC 6400 at McMaster University and the CDC CYBER 73 at The University of Western Ontario.

Thanks are also due to J. Pulido for the distillation column work in the Williams-Otto plant, G. Lusk for the drawings and Miss T. Wellon for the typing.
## TABLE OF CONTENTS

1. **INTRODUCTION TO DYNAMIC SIMULATION**
   1.1 Equation-Oriented Approach ........................................... 2
   1.2 Modular Approach .................................................. 4
   1.3 Thesis Objectives .................................................. 9

2. **NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS**
   2.1 Introduction ....................................................... 11
   2.2 Existence and Uniqueness Theorem .................................. 13
   2.3 Single-Step Methods .............................................. 14
   2.4 Multiple-Step Methods ............................................ 15
   2.5 Stability .......................................................... 17
   2.6 Stiffness .......................................................... 25
   2.7 Accuracy .......................................................... 29
   2.8 Accuracy Versus Stability ........................................ 35

3. **LITERATURE SURVEY OF NUMERICAL METHODS TO SOLVE STIFF SYSTEMS**
   3.1 Conventional Methods ............................................ 37
   3.2 Pseudo Steady State Approach ..................................... 38
   3.3 Stiff Techniques .................................................. 39
   3.4 Methods Used In Dynamic Simulation
       Executive Programs ............................................... 63
4. CONVERGENCE OF IMPPLICIT METHODS
   4.1 Jacobi Iteration ------------------------ 70
   4.2 Accelerated Iteration ------------------- 71
   4.3 Backward Iteration --------------------- 71
   4.4 Newton-Raphson Iteration ------------- 72
   4.5 Quasilinearization --------------------- 73
   4.6 Large Stiff Systems --------------------- 74

5. NUMERICAL SOLUTION OF SPARSE LINEAR ALGEBRAIC EQUATIONS
   5.1 Methods Tested For Solving Linear Algebraic Equations ------------------- 77
   5.2 Numerical Testing ---------------------- 88

6. NUMERICAL TESTING OF STIFF TECHNIQUES
   6.1 Testing By Other Workers -------------- 96
   6.2 Test Methods And Examples -------------- 98
   6.3 Test Results -------------------------- 102

7. THE DYNYSYS 2.0 EXECUTIVE PROGRAM
   7.1 Numerical Integration Of O.D.E.s ------ 111
   7.2 Corrector Convergence Of Stiff Equations -------------------------- 112
   7.3 Using the DYNYSYS 2.0 Executive Program ------- 115

8. SIMULATION EXAMPLES
   8.1 Level Control Example ------------------ 131
   8.2 Stirred Tank Reactor Network ----------- 144
9. THE WILLIAMS-OTTO PLANT SIMULATION
   9.1 The Reactor ........................................ 179
   9.2 The Heat Exchanger .................................. 186
   9.3 The Decanter .......................................... 190
   9.4 The Distillation Column .............................. 192
   9.5 Simulation Results ................................... 194
10. CONCLUSION
    10.1 Summary ............................................ 198
    10.2 Future Work ......................................... 200

APPENDIX A: GEAR'S METHOD .................................. 203
APPENDIX B: IMP ............................................... 209
APPENDIX C: SHANNON'S METHOD .............................. 216
APPENDIX D: TEST EXAMPLES .................................. 223
APPENDIX E: COMPUTER TIMINGS .............................. 235
APPENDIX F: THE DYNSYS DATA SET .......................... 238
APPENDIX G: TIME DELAY MODULE ............................ 245
APPENDIX H: WILLIAMS-OTTO PLANT LISTINGS ............... 252
APPENDIX I: PROGRAM LISTINGS ............................. 278
INDEX OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Process Flow Diagram</td>
<td>6</td>
</tr>
<tr>
<td>1.2</td>
<td>Dynamic Information Flow Diagram</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Typical Stability Boundary For</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Conventional Integration Technique</td>
<td>21</td>
</tr>
<tr>
<td>7.1</td>
<td>Skeleton Of Module For DYNYSYS 2.0</td>
<td>117</td>
</tr>
<tr>
<td>8.1</td>
<td>Process Flow Diagram-Level Control Example</td>
<td>132</td>
</tr>
<tr>
<td>8.2</td>
<td>Dynamic Information Flow Diagram - Level Control Example</td>
<td>132</td>
</tr>
<tr>
<td>8.3</td>
<td>Graph Of Tank Level Versus Time For Example #1</td>
<td>136</td>
</tr>
<tr>
<td>8.4</td>
<td>Listing Of Modules And Data Set For Example #1</td>
<td>137</td>
</tr>
<tr>
<td>8.5</td>
<td>Process Flow Diagram - Reactor Network Example</td>
<td>145</td>
</tr>
<tr>
<td>8.6</td>
<td>Dynamic Information Flow Diagram - Reactor Network Example</td>
<td>145</td>
</tr>
<tr>
<td>8.7</td>
<td>Listing Of Module And Data Set For Example #2</td>
<td>148</td>
</tr>
<tr>
<td>8.8</td>
<td>Graph Of Outlet Concentration Of A Versus Time For Example #2</td>
<td>152</td>
</tr>
<tr>
<td>8.9</td>
<td>Schematic Of Tubular Reactor</td>
<td>154</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>G.1</td>
<td>Listing Of Time/Delay Module</td>
<td>248</td>
</tr>
<tr>
<td>H.1</td>
<td>Listing Of Modules And Data Set For Williams-Ootto Plant</td>
<td>253</td>
</tr>
<tr>
<td>I.1</td>
<td>Listing Of DYNSYS 2.0 Executive</td>
<td>282</td>
</tr>
<tr>
<td>I.2</td>
<td>Skeleton Of Module For DYNSYS 2.1</td>
<td>317</td>
</tr>
<tr>
<td>I.3</td>
<td>Listing Of DYNSYS 2.1 Executive</td>
<td>321</td>
</tr>
</tbody>
</table>
# INDEX OF TABLES

1.1 Equation-Oriented Executive Programs For Dynamic Simulation ........................................ 3
1.2 Problem-Oriented Executive Programs For Dynamic Simulation ........................................ 5
3.1 Numerical Methods Used In Dynamic Simulation Packages .............................................. 64
5.1 Key To Linear Equation Solvers ................................................................. 91
5.2 Execution Times For 50 Linear Equations ......................................................... 92
5.3 Execution Times For 100 Linear Equations ......................................................... 93
5.4 Execution Times For 200 Linear Equations ......................................................... 94
5.5 Length Of TRGB Operator List ................................................................. 95
6.1 Execution Times And Errors For Conventional Methods ............................................. 106
6.2 Execution Times And Errors For Explicit Stiff Techniques ........................................ 107
6.3 Execution Times And Errors For Implicit Stiff Techniques ....................................... 108
6.4 Execution Times And Errors For Gear's Method And IMP ......................................... 109
7.1 New Parameters In The DYNSYS 2.0 Data Set .................................................. 116
7.2 Summary Of Module Variables Relating To DIFSUB .............................................. 122
8.1 Simulation Parameters For Reactor Network

Example ------------ 147
1. **INTRODUCTION TO DYNAMIC SIMULATION**

Dynamic simulation in this thesis will refer to the study, via digital computer simulation, of the dynamic behaviour of a process which is changing with time. In our context, the process will be all or part of a chemical plant.

There are many analysis and design problems that may require a flexible dynamic model for their solution, for example:

1. Difficulties in start up or shut down of a process.

2. Control strategies including finding suitable controller settings.

3. The influence of frequent disturbances or fluctuations in the process.

4. The response of the plant to equipment failure and testing of corrective action.

5. Operating strategies for multiproduct plants.

In addition there may be some steady state process problems whose solution can be obtained by driving a
dynamic model of the process to the steady state (thus avoiding numerical instabilities in a steady state iterative process, for example).

Various executive computer programs have been developed to handle the information transfer and calculations for performing a dynamic simulation.

1.1 Equation-Oriented Approach

The equation-oriented approach was developed first. Here, the user would write, possibly in some coded manner, all the algebraic and differential equations describing his process and the executive would then solve all the equations simultaneously over a period of time. There have been over thirty packages developed since 1955 (Franks, 1967). Table 1.1 lists some of the more common, recently developed programs: MIMIC (Northcott, 1967), GSMP (CSMP, 1967) and IMP (Brandon, 1972).
Table 1.1: Equation-Oriented Executive Programs For Dynamic Simulation

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Date</th>
<th>Full Name</th>
<th>Institution Where Developed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIMIC</td>
<td>1965</td>
<td></td>
<td>Wright-Patterson Air Force Base, Ohio</td>
</tr>
<tr>
<td>CSMP</td>
<td>1967</td>
<td>Continuous System Modeling Program</td>
<td>IBM</td>
</tr>
<tr>
<td>IMP</td>
<td>1972</td>
<td>Implicit Solution, Software System</td>
<td>University of Connecticut</td>
</tr>
</tbody>
</table>
1.2 Modular Approach

In the past few years, several executive programs have been developed, mostly by chemical engineers, which are more problem-oriented. A summary of these appears in Table 1.2: SWAPSO (Utsumi, 1969), KARDAZ (Brambilla et al., 1971; Kardasz, 1969; Kardasz and Molnar, 1971, 1974), DYNYSYS (Bobrow, Johnson, and Ponton, 1970, 1971), FLEX (Shern and Petty, 1970), PRODYC (Ingels and Mottard, 1970), REMUS (Ham, 1969), EARLYBIRD (Weaver, 1974), ACME (Loibl, Camp, and Wilkins, 1973), DYFLO (Franks, 1972a) and OSUSIM (Koenig, 1972). More information on these packages appears in Section 3.4.

These problem-oriented programs use variations of what is known as the modular approach. A real plant is made up of processing units such as reactors, heat exchangers, compressors, etc. with connecting lines of material flow. The flows, temperatures, concentrations and other variables in the process may be measured and, based on these measurements, controllers can activate control devices in an attempt to achieve desired conditions in the plant. Figure 1.1 illustrates a very simple mixing operation which is made up of a processing unit with controllers and control valves in a process flow diagram. Each of the interconnected units can be described by mathematical relationships relating inputs and outputs.
Table 1.2: Problem-Oriented Executive Programs For Dynamic Simulation

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Date</th>
<th>Full Name</th>
<th>Institution Where Developed</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWAPSO</td>
<td>1969</td>
<td>Stone and Webster All Purpose Simulator and Optimizer</td>
<td>Stone and Webster</td>
</tr>
<tr>
<td>KARDAZ</td>
<td>1969</td>
<td>Kardasz Program</td>
<td>University of Pisa</td>
</tr>
<tr>
<td>DYNSYS</td>
<td>1970</td>
<td>Dynamic Systems Simulator</td>
<td>McMaster University</td>
</tr>
<tr>
<td>FLEX</td>
<td>1970</td>
<td></td>
<td>Procter and Gamble</td>
</tr>
<tr>
<td>PRODYC</td>
<td>1970</td>
<td>A System for Simulating Chemical Process Dynamics and Control</td>
<td>University of Houston</td>
</tr>
<tr>
<td>REMUS</td>
<td>1970</td>
<td>Routine For Executive Multi-Unit Simulation</td>
<td>University of Pennsylvania</td>
</tr>
<tr>
<td>EARLYBIRD</td>
<td>1971</td>
<td></td>
<td>Tulane University</td>
</tr>
<tr>
<td>ACME</td>
<td>1972</td>
<td>Analyzer For Computer Modeling of Chemical Engineering Processes</td>
<td>University of Detroit</td>
</tr>
<tr>
<td>DYFLO</td>
<td>1972</td>
<td></td>
<td>DuPont</td>
</tr>
<tr>
<td>OSUSIM</td>
<td>1972</td>
<td>Ohio State University Simulator</td>
<td>Ohio State University</td>
</tr>
</tbody>
</table>
FIGURE I.1: PROCESS FLOW DIAGRAM

FIGURE I.2: DYNAMIC INFORMATION FLOW DIAGRAM
The assembly of the set of mathematical equations for simulating one of these process units, including controllers and control devices, in a computer program subroutine is called a unit computation or computation module.

Just as a chemical plant is made up of assemblies of the physical units, a mathematical model of a plant can be assembled as a network of modules, among which information will flow in a manner analogous to the material flow or control signals in the real plant. Figure 1.2 is a dynamic information flow diagram that corresponds to Figure 1.1. It can be seen that there is a very close similarity between the process flow diagram and the dynamic information flow diagram.

This building-block approach is very convenient for simulating systems which are very modular, i.e., which have many different pieces of physical equipment and where it is desired to study different combinations or configurations of these equipments and their effect on the dynamic behaviour. It is not as efficient computationally as the equation-oriented approach since there is more information transfer; however, it does allow the user to visualize his process more easily since each piece of equipment is usually represented by a corresponding module.
Several modules of different levels of sophistication can be developed for the same equipment. As a library of equipment and control modules becomes available, they may be readily used for the simulation of new plants, provided that the modules have a reasonable level of generality. This reduces the programming effort of new plant studies. A new simulation would require perhaps one third of the modules to be created, while the remaining two thirds could be taken from the library.

The modular approach deals with the real variables of the process, rather than transformed variables, and furthermore, the modules may be quite nonlinear in behaviour. This should encourage control studies by design and process engineers, even if they are unfamiliar with modern control systems theory and terminology.
1.3 Thesis Objectives

The objectives of this thesis are:

(1) to critically evaluate numerical techniques for the solution of systems of stiff ordinary differential equations such as those encountered in the analysis and design of complex chemical processes and to develop improved techniques for these situations.

(2) to critically analyze the DYNSYS package which is under development and to improve its approach for analyzing and designing complex systems.

(3) to demonstrate the application of an improved DYNSYS to several examples including a dynamic simulation of a realistic chemical process and to make recommendations for a library of unit computations.
2. **NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS**

A fundamental part of dynamic simulation is the numerical solution of ordinary differential equations (o.d.e.s). Partial differential equations can be broken down into o.d.e.s by discretizing one or more of the independent variables. Higher order o.d.e.s can usually be broken down into a set of first order equations. Thus we are primarily concerned with the numerical solution of a set of first order ordinary differential equations. These are usually initial value problems where each of the independent variables is specified at some initial time value (normally \( t = 0 \)), rather than boundary value problems where the independent variables are not all specified at the same time value. There are a large number of numerical integration techniques for solving these equations. This chapter discusses some aspects of these techniques.
2.1 Introduction

We are considering the system:

\[ \dot{y} = \frac{dy}{dt} = f(t,y) \quad (2.1.1) \]

with initial condition:

\[ y(t_0) = y_0 \quad (2.1.2) \]

where \( y \) may be scalar or vector.

\( f(t,y) \) represents any function of the dependent variable \( y \) and the independent variable \( t \).

\( t_0 \) represents the initial value of \( t \) and \( y_0 \) is the initial value of \( y(t) \) at \( t = t_0 \).

By a solution to the above system, we mean a curve in the \( y(t) \) versus \( t \) domain which passes through the point \( (t_0, y_0) \) and which satisfies (2.1.1). By a numerical solution, we mean a discrete set of values of \( y(t) \) called \( \{y_n\} \) corresponding to a discrete set of \( t \) values called \( \{t_n\} \) which approximate the equivalent continuous \( y(t) \) versus \( t \) curve.
To obtain these discrete sequences, we consider a finite set of points \( \{t_n\} \) which form a grid along the \( t \) coordinate. Each point in the sequence will be related to the previous point by the relationship:

\[
t_{n+1} = t_n + h_n \quad n = 0, 1, \ldots, N \quad (2.1.3)
\]

where \( h_n \) is the grid spacing.

Numerous techniques have been developed to solve o.d.e.s, the two principal types being single and multiple-step methods.
2.2 Existence and Uniqueness Theorem

Before we attempt to find a solution to (2.1.1) and (2.1.2), we must be assured that a unique solution does exist. The following two conditions are necessary and sufficient for a unique solution to exist in the time interval \([a, b]\) (Henrici, 1962)

1. \(f(t, y)\) is defined and continuous in the interval \(a \leq t \leq b\), \(-\infty < y < \infty\) where \(a\) and \(b\) are finite.

2. There exists a Lipschitz constant \(L\) such that for any \(t \in [a, b]\) and any two numbers \(u\) and \(v\):

\[
|f(t, u) - f(t, v)| \leq L |u - v| \quad (2.2.1)
\]

In the nonscalar case, vector norms replace absolute values.

The author is not aware of any equations resulting from chemical engineering applications where these conditions are not satisfied.
2.3 Single-Step Methods

Single-step methods do not require any information prior to \((t_n, y_n)\) to calculate \(y_{n+1}\). The most common single-step methods are the Euler and Runge-Kutta types. The simple Euler method is:

\[ y_{n+1} = y_n + h \cdot f(t_n, y_n) \quad (2.3.1) \]

Runge-Kutta methods use evaluations of \(f(t, y)\) within the interval \((t_n, y_n)\) and \((t_{n+1}, y_{n+1})\).

The general single-step equation is:

\[ y_{n+1} = y_n + \frac{1}{v} \sum_{i=1}^{v} \omega_i \cdot k_i \quad (2.3.2) \]

where \(\omega_i\) are weighting coefficients.

\(v\) is the number of \(f(t, y)\) substitutions and

\[ k_i = h \cdot f(t_n + c_i \cdot h, y_n + \sum_{j=1}^{i-1} c_i j \cdot k_j) \]

\[ c_1 = 0, \quad i=1, 2, \ldots, v \quad (2.3.3) \]

A typical fourth order formula is given by:

\[ y_{n+1} = y_n + \frac{1}{6} \left[ k_1 + 2k_2 + 2k_3 + k_4 \right] \]
where \( k_1 = h f(t_n, y_n) \)
\[
k_2 = h f(t_n + \frac{1}{2} h, y_n + \frac{1}{2} k_1)
\]
\[
k_3 = h f(t_n + \frac{1}{2} h, y_n + \frac{1}{2} k_2)
\]
\[
k_4 = h f(t_n + h, y_n + k_3)
\]

2.4 Multiple-Step Methods

A multiple-step or multistep method is one which uses information prior to \( y_n \) to calculate \( y_{n+1} \). The general linear, \( k \)-step differential-difference equation with constant coefficients is:

\[
y_{n+1} = \sum_{i=1}^{k} a_i y_{n+1-i} + h \sum_{i=0}^{k} b_i \dot{y}_{n+1-i} \tag{2.4.1}
\]

If \( b_o = 0 \), the resulting equation is called an explicit or predictor formula. However, if \( b_o \neq 0 \), the resulting equation is referred to as an implicit or corrector equation.

The normal mode of usage is to use an explicit form of (2.4.1) to get a first approximation to \( y_{n+1} \) (the predicted value) and then use an implicit form of (2.4.1) to improve on this value with \( \dot{y}_{n+1} \) evaluated using the predicted value of \( y_{n+1} \). The process may be repeated using the new value of \( \dot{y}_{n+1} \), calculated from the new
value of $y_{n+1}$. This may be done several times until $y_{n+1}$ converges to a fixed value. The term PECE means a predictor step followed by a derivative evaluation, then a corrector step and another derivative evaluation. For PE(CE)$^S$ the latter step is repeated $s$ times. These are also known as predictor-corrector methods although a single step method can also be a predictor-corrector.

A commonly used combination is the Adams-Bashforth predictor with the Adams-Moulton corrector equation. The third order equations are:

\[
\text{PREDICTOR: } y_{n+1} = y_n + \frac{h}{12} \left[ 23y_n - 16y_{n-1} + 5y_{n-2} \right]
\]

\[
\text{CORRECTOR: } y_{n+1} = y_n + \frac{h}{12} \left[ 5y_{n+1} + 8y_n - y_{n-1} \right]
\]
2.5 Stability

In the numerical solution of an o.d.e., a sequence of approximations $y_n$ to the true solution $y(t_n)$ is generated. The stability of a numerical method refers to the behaviour of the difference or accumulated error $y(t_n) - y_n$ as $n$ becomes large.

Stability analysis is usually performed on the scalar equation:

$$\frac{dy}{dt} = \lambda y \quad (2.5.1)$$

$$y(0) = 1 \quad (2.5.2)$$

with analytical solution:

$$y(t) = \exp(\lambda t) \quad (2.5.3)$$

For the numerical solution, if $h$ is constant

$$t = nh, \quad y_n = e^{nh\lambda} \quad n = 0, 1, 2, \ldots \quad (2.5.4)$$

or

$$y_{n+1} = e^{h\lambda} y_n \quad (2.5.5)$$
2.5.1 Stability Of Multistep Methods

Applying the general linear multistep method (2.4.1), we get the characteristic equation:

\[ \mu^k - \sum_{i=1}^{k} \alpha_i \mu^{k-i} - \kappa \lambda \sum_{i=0}^{k} \beta_i \mu^{k-i} = 0 \quad (2.5.6) \]

which has \( k \) characteristic roots \( \mu_i, i = 1, 2, \ldots, k \).

The numerical solution is thus:

\[ y_n = \sum_{i=1}^{k} d_i \mu_i^n \quad (2.5.7) \]

where \( d_1, \ldots, d_k \) are constants determined by the initial conditions.

One of the characteristic roots approximates the Taylor Series expansion of the true solution \( y = \exp(\lambda t) \) with a truncation error corresponding to the order \( p \) of the method. If we let this root be \( \mu_1 \), then as \( h \to 0 \)

\[ \mu_1 = \exp(h \lambda) + O(h^{p+1}) \quad (2.5.8) \]

This root, called the principal root, is the root which we wish to be represented in the numerical solution, since \( \mu_1^n \) approximates \( \exp(n \lambda) \). The other \( k-1 \) roots are extraneous and are as a result of the use of a difference equation of degree \( k \) to represent a first order
differential equation. The extraneous roots have no relation to the exact solution, but nevertheless are unavoidable.

The characteristic roots of (2.5.7) are the same as those of the difference equation for the error:

\[ \varepsilon_n = y_n - y(t_n) \]  \hspace{1cm} (2.5.9)

i.e.

\[ \varepsilon_n = \sum_{i=1}^{k} c_i \mu_i^N \]  \hspace{1cm} (2.5.10)

For a valid numerical solution we require that \( \varepsilon_n \) not grow with \( n \). A linear multistep method is called absolutely stable if \( |\mu_i| \leq 1 \), \( i = 2, \ldots, k \).

The critical problems of numerical stability in o.d.e.s are associated with inherently stable o.d.e.s \( (Re(\lambda) < 0) \) in which absolute stability is the important factor.

The value of \( h \lambda \) for which \( |\mu_i| = 1 \) and for which a small increase in \( h \lambda \) makes \( |\mu_i| > 1 \) is called the general stability boundary. Any method with a finite stability boundary is called conditionally stable, whereas any method with an infinite general stability boundary is called unconditionally stable or A-stable. The recent literature contains many new stability definitions (Lambert, 1973; Widlund, 1967; Cryer, 1973).
Most conventional methods have stability boundaries roughly like those in Figure 2.1. A-stability would require absolute stability for the entire left hand plane (Dahlquist, 1963).

Dahlquist (1963) has proven several important theorems for multistep methods:


2. The order of an A-stable implicit linear multistep method cannot exceed two.

3. The second order A-stable implicit linear multistep method with the smallest error constant is the trapezoidal rule:

\[ y_{n+1} = y_n + \frac{h}{2} (y_n + y_{n+1}) \quad (2.5.11) \]
FIGURE 2.1: TYPICAL STABILITY BOUNDARY FOR CONVENTIONAL INTEGRATION TECHNIQUE
2.5.2 Stability Of Runge-Kutta Methods

The difference equation resulting from an explicit Runge-Kutta method is:

\[ y_{n+1} = \left[ \frac{p}{i=0} \frac{(h\lambda)^i}{i!} + \sum_{i=p+1}^{v} a_i \frac{(h\lambda)^i}{i!} \right] y_n \]  \hspace{1cm} (2.5.12)

or

\[ y_{n+1} = \mu_1 y_n \]  \hspace{1cm} (2.5.13)

where \( p \) is the order of the method
\( v \) the number of derivative substitutions
\( a_i \) are constants depending on the specific formula
and \( \mu_1 \) is the characteristic root of the equation analogous to the principal root of a multistep method.

The numerical solution is thus:

\[ y_n = a_1 \mu_1^n \]  \hspace{1cm} (2.5.14)

and as for the linear multistep equation, we require for absolute stability \(|\mu_1| \leq 1\); this usually requires
\(|h\lambda| \leq k\) where \( k \) is of the order 1 to 10.

Note the difference in the basic causes of instability for the two types of solution. The difference equation for the multistep method yields extraneous roots.
which can cause instability if any one of them is greater than 1. However, for the Runge-Kutta technique, there are no extraneous roots and instability is caused only by taking a step size which is too large for the series to represent the characteristic root accurately.

2.5.3 Nonlinear Systems

The preceding stability analysis has been applied to the linear system \( \dot{y} = \lambda y \). There is no general stability theory for the nonlinear o.d.e. \( \dot{y} = f(t, y) \), but at any point in the integration, the equation can be linearized by a Taylor Series about that point, \((t_n, y_n)\):

\[
\dot{y} = f(t_n, y_n) + \left. \frac{\partial f}{\partial y} \right|_{(t_n, y_n)} (y - y_n) + O(h^2)
\]

Hildebrand (1956) has shown that the stability characteristics of the linearized o.d.e. are very similar to those of the original equation for small \( h \). Thus the stability analysis for the linear case represents the nonlinear problem at least locally.

The eigenvalues of (2.5.15) corresponding to \( \lambda \) of \( \dot{y} = \lambda y \) are the eigenvalues of the Jacobian matrix \( \frac{\partial f}{\partial y} \).

For the system \( \dot{y} = f(y) \) the Jacobian matrix is:
2.6 **Stiffness**

A stiff o.d.e. is one in which one component of the solution decays much faster than others. This may occur in a single o.d.e. or in a system of o.d.e.s. The term "stiff" probably arose from the study of mechanical or structural problems which had both soft and stiff spring constants (Steeper, 1970).

An example of a single stiff equation is:

\[
\frac{dy}{dt} = -200 \left[y - z(t)\right] + 3(t) \quad (2.6.1)
\]

\[
y(0) = 10 \quad (2.6.2)
\]

\[
z(t) = 10 - (10 + t)e^{-t} \quad (2.6.3)
\]

The analytical solution is:

\[
y(t) = z(t) + 10e^{-200t} \quad (2.6.4)
\]

It is more common, however, for this condition to occur in systems of equations. As an example:

\[
\begin{bmatrix}
-500.5 & 499.5 \\
499.5 & -500.5
\end{bmatrix}
\begin{bmatrix}
y \\
z
\end{bmatrix} = y \quad (2.6.5)
\]
\[ y(0) = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \]  

(2.6.6)

with analytical solution:

\[ y_1(t) = e^{-t} - e^{-1000t} \]  

(2.6.7)

\[ y_2(t) = e^{-t} + e^{-1000t} \]  

(2.6.8)

The eigenvalues of the matrix of coefficients are \( \lambda_1 = -1000 \) and \( \lambda_2 = -1 \). In terms of time constants, they are \( \tau_1 = 0.001 \) and \( \tau_2 = 1 \); the eigenvalues and time constants are negative reciprocals.

Both \( y_1 \) and \( y_2 \) have a rapidly decaying component, corresponding to \( \lambda_1 \), which very quickly becomes insignificant. After the brief initial phase of the solution in which the \( \lambda_1 \) component is not negligible, we would like to use a step size \( h \) which is determined only by the component of the solution corresponding to \( \lambda_2 \). However, stability considerations demand that both \( |h\lambda_1| \) and \( |h\lambda_2| \) be bounded for the entire integration, usually by a number from 1 to 10. As an example, Euler's method requires \( |h\lambda_1| = 2 \).

Although the component of the solution corresponding to \( \lambda_1 \) is of no practical interest, the criterion of absolute stability forces us to use an extremely small
value of \( h \) over the entire range of integration. As a result, the computation time necessary to integrate a highly stiff system can become excessive. It is desirable then, to use a method which is not so restrictive in step size, for example, a method that is A-stable.

The ratio of the largest to the smallest eigenvalue in absolute value is called the stiffness ratio. In (2.6.5), the stiffness ratio is 1000, a moderate value. In practice, ratios of the order \( 10^6 \) have been encountered, for example, the hydrogen-bromine reaction (Creighton, 1971).

Mathematically, for a stiff system, the real part of the largest eigenvalue is much less than zero and the absolute value of the ratio of the real part of the largest eigenvalue to the real part of the smallest eigenvalue is much greater than one.

\[
\text{i.e. } \Re(\lambda_{\text{max}}) \ll 0 \quad (2.6.9)
\]

\[
\left| \frac{\Re(\lambda_{\text{max}})}{\Re(\lambda_{\text{min}})} \right| \gg 1 \quad (2.6.10)
\]

One way of determining stiffness is to calculate the eigenvalues of the Jacobian matrix, but for all but the most simple cases, this is very tedious. In practice, an easy way to identify a stiff system is to use a
conventional method to integrate it. If an extremely small value is needed for the step size (say \( h < 0.01 \)) to avoid instability, the system is probably stiff and a special method should be used.

Stiff o.d.e.s occur commonly in reaction kinetics, control theory, circuit theory and to some extent in multistage systems. They are obtained in theoretical studies of reaction kinetic models of systems where the rate constants for the reactions involved are widely separated. Such systems occur in many fields of chemistry, particularly when radicals or quasistationary species are involved. The stiffness in control theory could, for instance, stem from the difference in time constants between the fast electrical control circuits and some slow mechanical devices. Stiffness in circuit theory often occurs in situations where transistors or other nonlinear semiconductors with very small time constants are connected to networks, often linear, with much larger time constants. Stiffness can arise in multistage systems, for example, in a distillation column because the time constants for the various components and plates may be very different and also since the time constant for the reboiler is large compared with the time constant of a plate.

Many chemical engineering systems give rise to stiff systems. Such situations occur in chemical reactors

2.7 Accuracy

In numerical integration there is usually an error associated with the method.

One source of error is the round-off error which is introduced by the machine due to the finite number of significant figures carried. Double precision arithmetic can be used, if it is felt round-off error is critical; however, it is very difficult to estimate and is usually neglected (Henrici, 1962).

The global truncation error of a method is the difference between the actual value and the calculated value:

\[ \text{G.T.E.} = y_{n}^{\text{ACT}} - y_{n}^{\text{CALC}} \]  

(2.7.1)

It is usually very difficult or impossible to calculate.
The local truncation error is the error committed in one integration step from $y_n$ to $y_{n+1}$. It is often of the form:

$$L.T.E. = Ch^{p+1} y^{(p+1)}(\xi) \quad (t_n < \xi < t_{n+1}) \quad (2.7.2)$$

If as $h \to 0$, the $L.T.E.$ becomes $o(h^{p+1})$, $p$ is called the order of the method.

The global truncation error is actually the accumulation of the local truncation errors. Because the global error is so difficult to calculate, the local truncation error is usually controlled. By keeping the local error below some specified value at each step, it is hoped that the global error is also kept under control. The step size can be chosen automatically to do this, if there is an estimate for the local error.

2.7.1 Single-step Methods

Among single-step methods, Euler's method has no error estimate, nor do most of the Runge-Kutta methods. Merson (1957) devised the first Runge-Kutta method with an error estimate:
\[ k_1 = hf(t_n, y_n) \]
\[ k_2 = hf(t_n + \frac{1}{3} h, y_n + \frac{1}{3} k_1) \]
\[ k_3 = hf(t_n + \frac{1}{3} h, y_n + \frac{1}{3} k_1 + \frac{1}{6} k_2) \]
\[ k_4 = hf(t_n + \frac{1}{2} h, y_n + \frac{1}{2} k_1 + \frac{1}{6} k_2 + \frac{1}{2} k_3) \]
\[ k_5 = hf(t_n + h, y_n + \frac{1}{2} k_1 + \frac{3}{2} k_3 + 2k_4) \]
\[ y_{n+1} = y_n + \frac{1}{6} (k_1 + 4k_4 + k_5) \quad (2.7.3) \]
\[ L.T.E. = \frac{1}{30} (2k_1 - 9k_3 + 8k_4 - k_5) \quad (2.7.4) \]

Other Runge-Kutta error estimators have also been developed. Lapidus and Seinfeld (1971) list a number of these.

2.7.2 Predictor-Corrector Methods

Predictor-corrector methods have very convenient local truncation error estimates, if the predictor and corrector are of the same order. For example, consider the third order Adams-Bashforth predictor, Adams-Moulton corrector (2.4.2).
predictor error = \frac{9}{24} h^4 y^{(4)}(\xi_p)
\text{where} t_{n-1} < \xi_p < t_{n+1} \quad (2.7.5)

corrector error = -\frac{1}{24} h^4 y^{(4)}(\xi_c)
\text{where} t_n < \xi_c < t_{n+1} \quad (2.7.6)

The exact value of \( y \) at \( t_{n+1} \) is given by:

\[
y(t_{n+1}) = y_{n+1}^{\text{PRED}} + \frac{9}{24} h^4 y^{(4)}(\xi_p) \quad (2.7.7)
\]

and

\[
y(t_{n+1}) = y_{n+1}^{\text{CORR}} - \frac{1}{24} h^4 y^{(4)}(\xi_c) \quad (2.7.8)
\]

Thus

\[
y_{n+1}^{\text{PRED}} + \frac{9}{24} h^4 y^{(4)}(\xi_p) = y_{n+1}^{\text{CORR}} - \frac{1}{24} h^4 y^{(4)}(\xi_c) \quad (2.7.9)
\]

Assuming the fourth derivative remains fairly constant over the interval:

\[
y_{n+1}^{\text{CORR}} - y_{n+1}^{\text{PRED}} = \frac{5}{12} h^4 y^{(4)}(\xi) \quad t_{n-1} < \xi < t_{n+1} \quad (2.7.10)
\]

thus predictor error = \frac{9}{10} \left( y_{n+1}^{\text{CORR}} - y_{n+1}^{\text{PRED}} \right) \quad (2.7.11)

corrector error = -\frac{1}{10} \left( y_{n+1}^{\text{CORR}} - y_{n+1}^{\text{PRED}} \right) \quad (2.7.12)
A possible PECE algorithm is:

1. calculate $y_{n+1}^{\text{PRED}}$

2. add predictor error (from previous step, assuming little change)

3. evaluate derivative

4. calculate $y_{n+1}^{\text{CORR}}$

5. add corrector error

6. evaluate derivative

2.7.3 Changing Step Size

Usually the local error is kept below some specified tolerance. If the error is too large (greater than tolerance) or too small (less than say $\frac{1}{10}$ of the tolerance), the step size is adjusted. One common algorithm is to double the step when the error is too small and halve the step when the error is too large.

Another scheme if the error varies as say, $h^4$

$$E = Ch^4 \quad (2.7.13)$$

Then if,

$$F_{\text{ACTUAL}} = C_{1}^{\text{OLD}} > \text{TOLERANCE} \quad (2.7.14)$$
If the desired error is say, half the tolerance,

$$E_{\text{DESIRED}} = 0.5 (\text{TOLERANCE}) = C_2 h_{\text{NEW}}^4$$ (2.7.15)

Assuming $C_1 \approx C_2$

$$\frac{E_{\text{ACTUAL}}}{E_{\text{DESIRED}}} = \left( \frac{h_{\text{OLD}}}{h_{\text{NEW}}} \right)^4$$ (2.7.16)

Thus the new step size is chosen as:

$$h_{\text{NEW}} = h_{\text{OLD}} \left[ \frac{E_{\text{DESIRED}}}{E_{\text{ACTUAL}}} \right]^\frac{1}{4}$$ (2.7.17)

$$= h_{\text{OLD}} \left[ \frac{0.5 (\text{TOLERANCE})}{E_{\text{ACTUAL}}} \right]^\frac{1}{4}$$ (2.7.18)

2.7.4 General Techniques

Usually relative error is controlled but if the independent variable becomes small (less than 1), it is better to control absolute error.

For a system of equations, the maximum error may be controlled. Gear (1971a) controls the Euclidean norm of the relative local truncation errors.

There are many other numerical integration techniques which may or may not have error estimators. An efficient and convenient local error estimate is a
desirable feature of any integration technique, assuming
a great deal of computation time is not required for estima-
tion and step-changing.

If the method does not have an estimate, one strategy is to
integrate the method using a step $h$ and then repeat the
integration for two steps each of $\frac{1}{2} h$.
By Richardson extrapolation, an expression for the local truncation error is (Carnahan, Luther and Wilkes, 1969):

$$L.T.E. = \frac{2^p (y_{n+1,2} - y_{n+1,1})}{2^p - 1} \quad (2.7.19)$$

where $p$ is the order of the method
$y_{n+1,1}$ is the value calculated by step $h$
$y_{n+1,2}$ is the value calculated by 2 steps of $\frac{h}{2}$

2.8 Accuracy Versus Stability

For a stable numerical solution, conventional methods for solving o.d.e.s require that $|h\lambda|$ for each eigenvalue be less than a number of the order 1 to 10 and hence the step size is restricted to be not much greater than the smallest time constant present. Ordinarily this restriction on $h$ does not control the step size since the accuracy requirement, or the prescribed bound on the local truncation error, usually forces the step size to satisfy an even more stringent restriction. However, for stiff
equations, the bound on the step size necessary to maintain stability can be three to five orders of magnitude smaller than that necessary to maintain accuracy (Walters, 1972), making it uneconomical to solve stiff systems with standard methods.

Also in stiff systems, some components may be so small that it is not necessary to simulate them accurately to get an accurate solution for the main components. What is required is that the solution be stable.

As a result, many numerical methods not subject to the above stability condition have been developed for solving stiff equations. A literature survey of these appears in the following chapter.
3. LITERATURE SURVEY OF NUMERICAL METHODS TO SOLVE STIFF SYSTEMS

The first mention in the literature of the stiffness problem was probably Curtiss and Hirschfelder (1952). Since then, the subject has received a great deal of attention, notably in the last five or ten years. This chapter surveys the methods which have been proposed to handle stiffness.

3.1 Conventional Methods

Conventional methods are restricted to a very small step size of the order $|\Delta| \less 1$ to 10. If a conventional method is to be chosen, it is reasonable to use one which requires the least work per time step, since they will use roughly the same step size and they will all be extremely accurate with such a small step. This method is the Euler method which has been recommended by Franks (1971, 1972b). However Euler's method will still use a great deal of computer time in solving stiff problems as shown in Chapter 6.
3.2 Pseudo Steady State Approach

Franks (1972a) suggests that the proper approach to stiff\ problems may be to simply eliminate those differential equations having small time constants and solve them as algebraic equations instead. This is generally known as the pseudo steady state approach.

Snow (1966) developed a computer program to calculate the product distribution in any homogeneous reaction mechanism. The program includes a numerical method to apply the steady state assumption when a mechanism involves intermediates present in low concentrations.

Brayton, Gustavson and Liniger (1966) analyze the transient behaviour of a transistor circuit. They reduce the original system of differential equations by setting some small resistors equal to zero, thereby eliminating the fast components. They point out that the new steady state differs from the original one. The error is roughly of the order of the ratio of the resistors neglected to those retained. For some problems this accuracy might be sufficient.

The pseudo steady state approach does introduce some large errors initially in some components. It is possible for these errors to be propagated throughout the solution so it is risky to use this procedure (Emanuel, 1967; Kolbrack, 1967; Snow, 1967; Gelin\nas, 1972). Also
with a complex nonlinear system, it may be very difficult or even impossible to eliminate the small time constants.

Perhaps future research will find the conditions under which the pseudo steady state approach is applicable and the amount of error incurred.

3.3 **Stiff Techniques**

In the past decade, many new sophisticated methods have been developed to overcome the instability problem.

The first major attempt to survey stiff techniques was Bjurel et al. (1970). There have been many review papers on a smaller scale (Lapidus and Seinfeld, 1971; Walters, 1971; Sigurdsson, 1970; Gear, 1969b; Steeper, 1970; Calahan, 1969; Hull, 1969).

This survey is not as detailed as that of Bjurel et al., but it extends the literature to early 1974, but since this is an area of current research, it should soon be obsolete.

Categorizing the methods is difficult since some methods would occur in more than one category; here we use several rough categories. Methods which are actually tested in Chapter 6 are explained in greater detail.
3.3.1 Extension Of Conventional Methods

Many authors have slightly extended the region of stability of conventional methods. Some of these are mentioned here.

Lawson (1966, 1967b) has developed fifth and sixth order Runge-Kutta formulae with extended ranges of stability.

Schoen (1971) develops fifth and sixth order predictors for use with Adams-Moulton correctors of the same order. The resulting PECE algorithms have larger regions of absolute stability than the Adams algorithms of corresponding order.

Emanuel (1964) examines the interaction between a stiff equation and several common integration procedures. He derives a convergence condition for the Runge-Kutta method which can be used to control the integration step size. He also introduces a technique for maximizing the step size and applies it to the generalized Adams predictor-corrector procedure.

Robertson (1966) combined linearity with a free parameter in the fourth order Simpson's Rule and the third order Adams-Moulton equation. For certain values of the parameter, he obtained stability regions superior
to those of either method (Sigurdsson, 1970).

3.3.2 Explicit Methods

Treanor (1966) has proposed a modified Runge-Kutta method. He observed that many stiff systems are of the form $\dot{y} = -\mathcal{P}(y-z)$, where $\mathcal{P}$ is a large number and $z$ is a slowly varying function of time. $z$ can be approximated by a power series in time containing unknown parameters which are determined in the course of the integration. In particular, the method assumes that the derivative can be approximated in any interval by:

$$\dot{y}_n = -(\mathcal{P}_n) y_n + (A_\xi)_n + (B_\xi)_n \dot{t} + (C_\xi)_n \dot{t}^2 \quad (3.3.1)$$

The four constants $A_\xi$, $B_\xi$, $C_\xi$ and $P_\xi$ are evaluated by determining the value of the derivative at four points in the interval $[t_n, t_n + \Delta t]$ and solving for the constants from evaluation of the above equation at the four points. The four points are $t_n$, $t_n + \frac{1}{2} \Delta t$, $t_n + \frac{1}{2} \Delta t$ and $t_n + \Delta t$.

The following algorithm is obtained:

$$y_{n+1}^{(1)} = y_n + \left( \frac{\Delta t}{2} \right) f_n$$

$$y_{n+1}^{(2)} = y_n + \left( \frac{\Delta t}{2} \right) f_n^{(1)}$$

$$y_{n+1}^{(3)} = y_n + \Delta t \left[ f_n^{(1)} + f_n^{(2)} + PhF_2 + \frac{f_{n+1}^{(2)}}{2} \phi \right]$$

$$\quad$$

$$y_{n+1} = y_n + \Delta t \left[ f_n^{(1)} + f_n^{(2)} + PhF_2 + \frac{f_{n+1}^{(2)}}{2} \phi \right]$$

$$\quad$$

$$y_{n+1} = y_n + \Delta t \left[ f_n^{(1)} + f_n^{(2)} + PhF_2 + \frac{f_{n+1}^{(2)}}{2} \phi \right]$$
\[ y_{n+1} = y_n + h f_n P_1 + h v_0 (P y_n + f_n) + h v_2 (P y_{n+1} + f_{n+1}) + h v_3 (P y_{n+1} + f_{n+1}) \]

where

\[ P_1 = \frac{e^{-Ph} - 1}{-Ph} \]
\[ P_2 = \frac{e^{-Ph} - 1 + Ph}{(Ph)^2} \]
\[ P_3 = \frac{e^{-Ph} - 1 + 2Ph - \frac{1}{2}(Ph)^2}{(Ph)^3} \]

and

\[ v_1 = -P_2 + 4P_3 \]
\[ v_2 = 2(P_2 - 2P_3) \]
\[ v_3 = 4P_3 - 3P_2 \]
\[ P_i = \frac{f_i^{(2)} - f_i^{(1)}}{y_i_{n+1} - y_i_{n+1}} \]

The value of \( P \) is taken to be the largest value of \( P_i \); if this value is negative, \( P \) is set to zero.

For small \( h \), the method is identical to the fourth order Runge-Kutta method (Eqn. 2.3.4).

The step size control is to bound \( \frac{y_{n+1} - y_n}{y_{n+1}} \) at each step between an upper and lower tolerance. Double-halving is used.
Jung (1967) has written a modified version of Treanor's method.

Lomax and Bailey (1967) have analyzed the method of Treanor and shown that it is efficient if there is only one large real negative eigenvalue or if all the large negative eigenvalues are close to each other and real. They also suggest some improvements of the method.

Richards, Lanning and Torrey (1965) devised a scheme for large stiff systems based on Euler's method. The method is based on two qualitative observations:

1. It is possible to take a large integration step even with Euler's method, if the value of $y$ is sufficiently close to the desired trajectory.

2. The onset of instability is indicated by sudden reversals of the derivative.

The algorithm uses Euler's method with step size $h = \beta \frac{\|v_n\|}{\|x_n\|}$, where the double bars indicate Euclidean norms. $\beta$ is a parameter that can be adjusted for stability. Essentially it limits the fractional change in the large components of $y$. $\beta = 0.01$ is a reasonable value for almost all problems.
If the derivatives change greatly; i.e., if

$$\cos \theta = \frac{\dot{\xi}_n \cdot \dot{\xi}_{n+1}}{||\dot{\xi}_n|| ||\dot{\xi}_{n+1}||} < -\frac{1}{s} \tag{3.3.4}$$

then the initial value of $y_{n+1}$ is abandoned and replaced by an interpolated value:

$$\dot{y}_{n+1} = \dot{y}_n + S(\Delta t)$$

$$y_{n+1}^{\text{NEW}} = y_n + S(y_{n+1} - y_n) \tag{3.3.5}$$

$$S = \frac{\dot{y}_n \cdot (\dot{y}_n - \dot{y}_{n+1})}{||\dot{y}_{n+1} - \dot{y}_n||^2}$$

Fowler and Warten (1967) derived an explicit second order method for large stiff systems, where the smallest time constant is real.

Consider the differential equation $\dot{y} = f(t,y)$ with solution $y_T$. At each step, let $y_C$ be the computed solution. $y_C$ is assumed to be the sum of two functions, $y_A$, an asymptotic part and $y_P$, a perturbation from the asymptote. $y_A$ is to be determined from present and past values of $y_T$. In practice of course, $y_T$ is not available and the approximations, $y_C$, obtained at the previous steps are used in the calculations instead of $y_T$. 
Assume \( y_A \) and \( y_{PE} \) are of the form:

\[
y_A(t + \xi) = y_A(t) + \xi \dot{y}_A(t)
\]

\[
y_{PE}(t + \xi) = y_{PE}(t) + e^{\lambda_P \xi - 1} y_{PE}(t)
\]

\[0 \leq \xi \leq h, \text{ thus:}\]

\[
y_C(t + \xi) = y_A(t + \xi) + y_{PE}(t + \xi)
\]

\[= y_A(t) + y_{PE}(t) + \xi \dot{y}_A(t) + e^{\lambda_P \xi - 1} y_{PE}(t)\]

Five conditions are imposed to determine the five constants \( y_A(t) \), \( y_{PE}(t) \), \( \dot{y}_A(t) \), \( \dot{y}_{PE}(t) \), \( \lambda_P \). The first three of these make the method second order exact and the last two determine the constants uniquely:

1. \( y_A(t) + y_{PE}(t) = y_T(t) \)
2. \( \dot{y}_A(t) + \dot{y}_{PE}(t) = \ddot{y}_T(t) \)
3. \( \lambda_P \dot{y}_{PE}(t) = \ddot{y}_T(t) \)
4. \( y_A(t) = y_T(t) \)
5. \( \dot{y}_A(t) = \frac{[y_T(t) - y_T(t - h_O)]}{h_O} \)

There are several possible modifications under certain conditions and a complex step size control.
Nigro (1969) uses multistep formulae of the form:
\[ y_{n+1} = \sum_{i=1}^{3} \alpha_i y_{n+1-i} + h \sum_{i=1}^{3} \beta_i y_{n+1-i} \] (3.3.8)

He constructs arbitrarily large stability intervals, \( h \lambda_{max} \), by choosing appropriate values of the \( \alpha \) and \( \beta \). By perturbing the \( \alpha \) and \( \beta \), larger relative stability intervals can be created.

The method is not self-starting; it needs three calculated points before it can take over. The step size is fixed, as there is not a good error or step changing analysis.

Lomax (1968) proposed some second order explicit Runge-Kutta methods which extended the region of stability considerably.

Pope (1963) proposes the explicit difference equation:
\[ y_{n+1} = y_n + h f_n + \frac{h^2}{2} \left[ \exp(hf_y) - 1 - hf_y \right] \] (3.3.9)

Kubicek (1975) has developed a complex one-step integration method, based on boundary value technique. The algorithm is nonlinear, explicit and second order. Kubicek also claims it is A-stable.

Chu and Berman (1974) describe an explicit complex single-step method. It is second order.
3.3.3 Implicit Multistep Methods

Klopfenstein and Davis (1971) present a study of a class of PECE algorithms consisting of an application of a predictor followed by application of one iteration of a pseudo Newton-Raphson method to a corrector.

They use algorithms of the form:

\[ p_{n+1} = y_n + h \left[ \alpha f_n + \beta f_{n-1} \right] \]

\[ c_{n+1} = y_n + h \left[ v f \left( t_{n+1} ; P_{n+1} \right) + u f_n \right] \tag{3.3.10} \]

\[ y_{n+1} = P_{n+1} + [\alpha I - v h J]^{-1} [c_{n+1} - P_{n+1}] \]

The \( \alpha, \beta, u, v \) and \( \alpha \) are real constants, \( I \) is the identity matrix and \( J \) approximates the Jacobian.

The algorithm with \( \alpha = 1 - \beta, v = 1 - u, u < \frac{1}{\beta} \) is first order with a first order error estimate:

\[ c_{n+1} = P_{n+1} - y_{n+1} + \frac{\beta + \frac{1}{\beta}}{\beta - u + 2} (c_{n+1} - P_{n+1}) \tag{3.3.11} \]

\( \alpha = 1 - u \) will shift the stability region so that the centre of an inscribed circle is at the origin.

The algorithm with \( \alpha = \frac{3}{2}, \beta = -\frac{1}{2}, v = u = \frac{1}{2} \) is second order with first order error estimate:
\[ e_{n+1} = p_{n+1} - y_{n+1} + \frac{5}{6} (c_{n+1} - p_{n+1}) \quad (3.3.12) \]

\[ \alpha = 0.71 \] will shift the stability region to the origin.

There are variants of the \( \theta \) method, known for many years:

\[ y_{n+1} = y_n + h [(1 - \theta)y_{n+1} + \theta \dot{y}_n] \quad (3.3.13) \]

Liniger and Willoughby (1970) and Brandon (1972) discuss methods of this type. See Appendix B for an analysis of Brandon's method. Zein and Hakimi (1971) use Liniger and Willoughby's method with Newton-Raphson iteration to solve circuit problems.

Sandberg and Shichman (1968), Shichman (1969) recommend taking one Newton-Raphson iteration of the implicit Euler method with \( y_n \) used as the estimate of \( y_{n+1} \):

\[ y_{n+1} = y_n + h \dot{y}_{n+1} \]

\[ y_{n+1} = y_n + [I - h \frac{\partial F}{\partial y}]^{-1} [h \dot{y}_n] \quad (3.3.14) \]

A rough error estimate and step control scheme is given. Sloate (1970) uses an extension of this method.

Hodgkins (1969) presents an algorithm based on the trapezoidal rule in conjunction with global extrapolation.
Lindberg (1971a,b) gives a simple smoothing procedure which damps out the oscillation of the trapezoidal rule applied to stiff systems.

The FACE program (Price, 1970) uses a simple, first order, one-step algorithm, known as a linear extrapolation method. There is provision for using a special integrator which permits fast portions of the problem to be run with a reduced step size. This is an approximation which assumes the inputs from the main portion of the problem are constant during a normal time step.

Curtiss and Hirschfelder (1952), Krogh (1971, 1973) and Gear (Appendix A) have recommended the numerical differentiation formulae for stiff systems. Cryer (1972) discusses a stability theorem for the formulae. Calahan (1971) also discusses Gear's method. Dill and Gear (1971) used an interactive computer graphics program to find stiffly stable methods of orders seven and eight.

Branin et al. (1971) have developed a version of the numerical differentiation algorithm different from Gear's which they claim is more stable than his version under certain conditions.

Klopfenstein (1971) generalizes the numerical differentiation formulae of orders one through six to increase the angular width of the wedge of stability at only modest cost in increased local truncation error.
Bickart and Piel (1973) derive some high order stiffly stable composite multistep methods.

Cooke (1972, 1973) analyzes stiff stability.


Brunner (1972) derives linear k-step methods \((k \geq 2)\) with constant coefficients by choosing as the second characteristic polynomial of the method, a Schur polynomial whose coefficients depend on a certain set of parameters. For the case \(k = 2\), the corresponding two-step methods are always A-stable.

Bjurel (1969, 1972) derives implicit and explicit Adams-like multistep formulae for equations of the type \(P\dot{y} = f(t, y)\) where \(P\) is a polynomial with constant coefficients and \(|\partial f/\partial y|\) is small compared with the roots of \(P\).

Ehle (1968) shows how one can obtain one-step methods of arbitrarily high order which are A-stable. These are n-stage implicit Runge-Kutta processes of order \(2n\) and a generalized class of linear one-step methods of the form:

\[
y_{n+1} = y_n + \sum_{i=1}^{j} \alpha_{i,j} h^i \left[ (-1)^{i+1} y_{n+1} + y_i \right] \\
\]

\((j=1,2,3,\ldots)\) (3.3.18)
Lindberg (1972) describes the package IMPEX which consists of two procedures for the treatment of the transient phase, and two algorithms, one of order two and one of order four, for computation of the smooth solution after the transient.

The Lipschitz constant at the starting point is estimated. From this estimate, a starting step size for the calculation of the transient phase is computed. Throughout the transient phase, a fourth order Runge-Kutta method is used. When the transients are negligible, the variation of the solution is examined in order to get a reasonable starting step size for the main program. The smooth solution is computed by the implicit midpoint rule with second order smoothing and fourth order extrapolation (Bulirsch and Stoer, 1966). The step size is controlled using an estimate of the global truncation error.

IMPEX is the only program found containing different algorithms for both the transient and smooth phases. However, in a practical simulation, there may be transients constantly arising from system disturbances. IMPEX assumes the transients arise only at the beginning of the simulation.

Zavorin and Khesina (1974) have developed three A-stable third order single-step methods.
Sloate (1971), Bickart, Burgess and Sloate (1971) and Sloate and Bickart (1973) use two simultaneous linear multistep difference equations to integrate stiff systems. In this way Dahlquist's limitation that no A-stable linear multistep formula can exceed order two is bypassed. They present a fourth order A-stable method.

Davison (1973) has devised a single-step implicit algorithm for integrating very large stiff differential equations of the type:

\[ \dot{y} = Ay + Bu + f(t, y) \quad (3.3.16) \]

where \( f(t, y) \) has a small Lipschitz constant.

Roe (1967) describes a four-step, predictor-corrector algorithm which is derived by fitting the formula to an exponential plus a quadratic. The method works best if the output versus time is exponential-like and degenerates to the Adams-Moulton method, if the response is not similar to an exponential.

Guderley and Hsu (1972) studied a predictor-corrector method for systems of the form:

\[ \dot{y} + Dy = Ay + Bt \quad (3.3.17) \]

where \( D \) is a diagonal matrix which may have some large
elements and the right-hand side is considered as non-stiff. The operator on the left is inverted and the right hand side is approximated by Lagrangian interpolation polynomials at grid points. The integration of the exponential functions is done analytically.

Horsett (1969) gives new finite difference formulae which are exact for the problem \( \dot{y} = P\dot{y} + Q(t) \) where \( P \) is a constant and \( Q(t) \) is a polynomial of degree \( q \). When \( P = 0 \), the method is identical with the Adams-Bashforth formulae.

Stineman (1965) used an implicit one-step method for the integration of first and second order equations of the type:

\[
\alpha(t) \dot{y} + y = b(t) \tag{3.3.18}
\]

\[
\alpha(t) \dot{y} + \alpha(t) \cdot \dot{y} + y = e(t)
\]

where the functions \( \alpha, b, \alpha, \dot{y} \) and \( e \) are locally approximated by linear functions.

Sarkany and Ball (1969) treat the solution of \( \dot{y} = -Py + f(t, y) \) where \( P \) is a diagonal matrix. A predictor-corrector method is used.

Spicer (1968, 1969) defines exact stability as the extraneous roots being constrained to be zero. He derives
predictor-corrector formulae having this property.

*Certaine* (1960) has developed an iterative multi-step method for solving systems of the form \( \dot{y} + Py = f(t,y) \) where \( P \) is a constant diagonal matrix with large positive elements and \( f(t,y) \) is a slowly varying function of time.

*Snider and Fleming* (1974) use aliasing to reduce the computation of finding an accurate trigonometric interpolation for a function with dominant high frequencies. The technique is applied to stiff differential equations, extending the applicability of the method of *Certaine* to systems with oscillatory forcing functions.

### 3.3.4 Runge-Kutta Methods

The basic Runge-Kutta equation is explicit:

\[
y_{n+1} = y_n + \sum_{i=1}^{v} w_i k_i
\]  
(3.3.19)

\[
k_i = hf(x_n + \sigma_i h, y_n + \sum_{j=1}^{i-1} \sigma_{ij} k_j)
\]  
\( \sigma_2 = 0, i = 1, 2, \ldots, v \)

*Butcher* (1964) introduced implicit Runge-Kutta methods. *Ehle* (1968) and *Gear* (1970) discuss these further:

\[
k_i = hf(x_n + \sigma_i h, y_n + \sum_{j=1}^{v} \sigma_{ij} k_j)
\]  
(3.3.20)

\[ k_i = hf(x_n + c_i h, y_n + \sum_{j=1}^{n} c_{ij} k_j) \quad \sum_{j=1}^{n} c_{ij} = 0, i = 1, 2, \ldots, n \]  

(3.3.21)

The implicit and semi-implicit forms are A-stable.

Chipman (1971) defines the concept of strong A-stability. He then introduces a class of strongly A-stable Runge-Kutta processes.

Axelsson (1972) gives formulae for a class of A-stable quadrature methods or equivalently for a certain implicit Runge-Kutta scheme. The methods are strongly A-stable.

Lopes and Phares (1965) and Lawson (1967a) have developed an A-stable Runge-Kutta method. The system must first be transformed by \[ y(t) = \exp(-At)z(t). \]

Stanton and Talukdar (1970) use a fourth order Runge-Kutta and a complex transformation of variables to increase the stability interval and permit a large step size.
3.3.5 Miscellaneous Methods

Jain (1972) gives some A-stable methods of order \(2n\), with variable coefficients based on Hermite interpolation polynomials, making use of \(n\) starting values.

Axelsson (1969) gives A-stable methods of a quadrature type:

\[
y_{n+1} = y_n + h \sum_{k=1}^{n} c_{ik} f(y_{k}, z) \quad (3.3.22)
\]

where \(c_{ik}\) are quadrature coefficients over the zeros of \(P_n - P_{n-1}(\nu=1)\) or \(P_n - P_{n-2}(\nu=2)\), where \(P_n\) is the Legendre polynomial orthogonal on \([0,1]\) and normalized such that \(P_n(1) = 1\).

Dahlquist (1969) devised a method based on local polynomial approximations for two stiff o.d.e.s of particular type. Oden (1971) gives further details of the method.

Watts and Shampine (1972) develop \(r\)-block implicit one-step methods which compute a block of \(r\) new values simultaneously with each set of application. A subclass of formulae is derived which is related to Newton-Cotes quadrature and it is shown that for block sizes \(r=1,2,\ldots\), these methods are A-stable, while those for \(r=3,4\) are not. They construct A-stable and strongly A-stable
formulae having arbitrarily high orders of accuracy.

Gilo and Grebe (1968) discuss multistep algorithms and the computation of optimum coefficients. They point out that each problem has its own optimum formula, and z-transform techniques are used to study this. This approach may not be practical for general purpose use, since the optimum formula would be difficult or impossible to determine, and the method is limited to linear equations.

Andrus (1967) uses a quasi-analytical approach which takes the inverse Laplace transform of the system equations, using a summation of the partial fractions. Some approximate techniques for handling nonlinearities are discussed, but this appears to be a limitation.

Liniger and Willoughby (1970) introduced the concept of exponential fitting. They proposed single-step methods of the form:

\[ y_{n+1} = \alpha_2 y_n + h^2 \left[ \beta_0 \ddot{y}_{n+1} + \beta_1 \ddot{y}_n \right] + h^2 \left[ \gamma_0 \dot{y}_{n+1} + \gamma_1 \dot{y}_n \right] \quad (3.3.23) \]

If the characteristic root is chosen by adjusting the free parameters to be \( \exp(q_0) \), then we say the method is exponentially fitted at \( q_0 \).
Liniger (1968) gives two easy-to-check conditions which together are sufficient and "almost necessary" for A-stability of linear multistep integration formulae.


Odeh and Liniger (1972) discuss unconditional fixed step size stability of linear multistep formulae.

Prothero and Robinson (1974) discuss the stability and accuracy of implicit one-step methods to stiff equations of the form:

\[ \dot{y} = \dot{y}(t) + \lambda(y - g(t)) \]

They describe some new stability properties and recommend a family of methods based on a compromise between accuracy and stability considerations.

Brunner (1974) uses recursive collocation. He approximates the solution on each subinterval by a linear combination of exponential functions which involve only the significant eigenvalues of the Jacobian. The unknown vectors are computed recursively by requiring they satisfy the given system at certain suitable points (collocation),
with the additional condition that the collection of these functions represent a continuous function satisfying the given initial conditions.

Liniger and Odeh (1972) discuss an algorithm wherein several low-order solutions are computed by repeated integration using a multistep method with parameters. By forming suitable linear combinations of such solutions, higher order solutions are obtained. If the parameters are properly chosen, the underlying solutions, and thus the higher order one, can be made A-stable and strongly damping with respect to the stiff components of the system.

Blue and Gummel (1970) discuss rational function approximations for the matrix exponential $\exp(h\lambda)$.

The use of Padé approximations for $\exp(h\lambda)$ is discussed by Calahan (1967), Junge (1968) and Osborne (1969) for linear systems.

Cooper (1969) gives a brief review of stiffness, and an iteration procedure likely to give convergence, both in multistep methods and in the steady state approach.

Miller (1964) gives a second order stiff equation and suggests a scheme for solving it. Cash (1972) extends this method and develops an entirely new algorithm for differential equations of order three and higher, to enable these equations to be integrated with a reasonable step length.
Miranker (1971) gives a description and analysis of a class of matricial difference schemes. This class of schemes is based in part on a generalization of the feature of classical numerical methods of being characterized by approximations at a single point in the complex plane.

The following four methods are restricted to linear systems or systems which can be linearized easily.

Mah, Michaelson and Sargent (1962) developed a stable linearization technique for solving the dynamic behaviour of multistage systems. The coefficients of the differential system are taken as constant over a short time interval. The exact solution of the linearized system is used to predict the values at the end of the interval and hence the coefficients for the next interval.

Buffam and Kropholler (1969) extend the concept of network combing to give the concentration history at all the nodes in a flow network whose nodes have exponential dynamic mixing characteristics with arbitrary time constants. They show that the transition matrix for the flow network is a special form of stochastic matrix whose power series has absolute convergence. The generalization results in a very stable method for integrating linear systems of o.d.e.s.

Davison (1968) has considered the problem of the numerical integration of large systems of constant
coefficient linear o.d.e.s. The poles and zeros of the solution are obtained and the solution constructed in terms of a sum of exponentials.

**Moretti's (1965)** technique requires nonlinear o.d.e.s to be reduced to linear form. The numerical solution then requires the evaluation of the complex eigenvalues of an n-th order matrix, thus limiting the analysis to fairly small systems. **DeGroat and Abbet (1965)** and **Maheshwari (1968)** have extended this approach.

The following four methods use state transition matrices.

**Liou (1966)** has devised a fast and accurate method. However, it is for linear systems only; one output is available and the problem must be formulated as a ratio of two polynomials in s (the Laplace operator). **Nichol (1970)** has a similar method, but simple nonlinearities can be handled.

**Giese (1967)** also uses state variable techniques and discusses their use in quasi-linear systems. He also discusses a combination approach for stiff equations, where this method is used for the fast portion of the problem and a conventional explicit numerical procedure is used for the slow portion.

**Oswald and Smith (1970)** also use state transition matrices in their method.
Lee (1967) uses matrix filters to separate the differential equations into slow and fast equations. These are then computed separately and the results are added together. It is limited to linear equations.

Allen and Fath (1969) propose techniques for systems which can be separated into a stiff linear part and a nonstiff linear part. Padé approximations are used.

Jain (1971) transforms the system by
\[ z(t) = \exp(At)y(t) \]
where \( A \) is a real square matrix and obtains modified Adams and linear multistep methods which are \( A \)-stable. Jain gives a working algorithm to obtain \( A \) and to calculate \( \exp(hA) \).

Shannon (1971) developed a technique using a changeable independent variable of integration. It can be used with any integration method. Appendix C contains an analysis of the technique.

Singular perturbation theory is applicable when the system can be written as:

\[
\begin{align*}
\epsilon \dot{y}_1 &= f_1(t, y_1, y_2) \\
\dot{y}_2 &= f_2(t, y_1, y_2)
\end{align*}
\]

(3.3.24)

where \( \epsilon \) is a small real parameter and the \( y_2 \) components are stable in the sense that if \( y_2 \) is fixed at
any time, then the $y_j$ components rapidly reach an equilibrium solution.

MacMillan (1968) proposes a method based upon this theory.

Miranker (1973) describes an equivalence between a subclass of stiff systems and differential equations subjected to singular perturbations. He uses the characterization of the solution of this class of equations in terms of boundary layers as a means of generating numerical procedures for solving the stiff equations. The numerical procedures have the desirable feature of improving with increasing stiffness.

Diperna (1971) describes a digital simulation method for interconnected continuous systems.

3.4 Methods Used In Dynamic-Simulation Executive Packages

As mentioned in Chapter 1, there have been several new problem-oriented executive packages developed in the past five years. Almost all of these programs contain an algorithm for the numerical solution of ordinary differential equations; i.e., the system model contains ordinary differential equations. Table 3.1 summarizes some of the numerical methods used.
Table 3.1: Numerical Methods Used in Dynamic Simulation Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Method Used To Integrate Differential Equations</th>
<th>Suited For Stiff Eqns.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIMIC</td>
<td>4th order Runge-Kutta</td>
<td>No</td>
</tr>
<tr>
<td>CSMP</td>
<td>various conventional methods, latest version contains Gear's method</td>
<td>Yes</td>
</tr>
<tr>
<td>IMP</td>
<td>Brandon's method</td>
<td>Yes</td>
</tr>
<tr>
<td>KARDAZ</td>
<td>explicit finite difference method used to solve a set of quasilinear hyperbolic partial differential equations</td>
<td>No</td>
</tr>
<tr>
<td>DYNSYS</td>
<td>Adams-Moulton-Shell</td>
<td>No</td>
</tr>
<tr>
<td>PRODYC</td>
<td>any method in CSMP</td>
<td>Yes</td>
</tr>
<tr>
<td>REMUS</td>
<td>16 conventional options: Simpson's Rule, various Runge-Kutta, Milne and Adams-Moulton formulae</td>
<td>No</td>
</tr>
<tr>
<td>ACME</td>
<td>2nd and 4th order Runge-Kutta, Euler's method</td>
<td>No</td>
</tr>
<tr>
<td>DYFLO</td>
<td>2nd and 4th order Runge-Kutta, Euler's method</td>
<td>No</td>
</tr>
<tr>
<td>OSUSIM</td>
<td>Euler predictor, trapezoidal rule as corrector</td>
<td>Yes</td>
</tr>
</tbody>
</table>
MIMIC (Northcott, 1967) uses the fourth-order Runge-Kutta method.

CSMP (CSMP, 1967) offers a variety of conventional methods, but the latest version contains Gear's method.

IMP (Appendix B) offers Brandon's method, a new A-stable method very suitable for stiff equations.

Franks (1971) has made a detailed review of the packages DYNNSYS, PRODYC, REMUS, DYFLO and OSUSIM.


PRODYC (Ingels and Motard, 1970) has a preprocessor which converts the system model into CSMP statements after which any of the methods available in CSMP (CSMP, 1967) can be used to solve the equations.


DYFLO (Franks, 1972) offers a choice of simple Euler or second or fourth order Runge-Kutta.

OSUSIM (Koenig, 1972) uses an Euler predictor and a trapezoidal rule as corrector. OSUSIM emphasizes the formation and solution of implicit and explicit algebraic equations.
ACME (Loibl, Camp and Wilkins, 1973) is a new modular package. It simulates both the steady state and dynamic behaviour of chemical engineering systems. It uses several options for solving o.d.e.s: second and fourth order Runge-Kutta and Euler's method.

The packages DYNSYS, REMUS, PRODYG, OSUSIM and ACME basically use the modular approach. All of these executives are highly structured and have their own input language which the user must master. The author has not noticed any particular advantage of any of these, but Franks claims DYFLO is the most convenient program from the user's point of view. DYFLO is not exactly modular; it is somewhere in between the modular approach and the equation-oriented approach. The user of DYFLO writes his own computer program using essentially a series of CALL statements to utilize a rather large number of service subroutines that Franks has provided. Culver (1972) used DYFLO to study distillation column dynamics and he attests to its convenience.

The IMP package requires a good deal of sophistication on the part of the user to set up the system equations for simulation and to select the best options for a particular study.

Kardasz (Brambilla et al., 1971; Kardasz, 1969; Kardasz and Molnar, 1971, 1974) has developed a package
for the simulation of the dynamic behaviour of chemical plants. It is quite different than the other packages just discussed. The language used is not FORTRAN but is based on SIMULA-67. It is structure-oriented according to the flowsheet of the plant to be simulated. The method of simulation is based on the concept of a cell model. Under the assumption of a one-dimensional model, a chemical plant composed of a set of devices can be represented mathematically by a set of quasilinear hyperbolic partial differential equations of first order. After appropriate discretization, each device is subdivided into a finite number of cells.

An explicit finite difference method is used to solve the equations.

Other attempts have been made to form dynamic simulation packages. Utsumi (1969) describes the SWAPSO package developed at Stone & Webster, but it has not been pursued (Crean, 1974). A package called EARLYBIRD has been developed at Tulane University but the dynamic simulation portion of it has been discontinued (Heaver, 1974). Proctor & Gamble (Shorn and Petty, 1970) has a modular package called FLEX. It uses Euler's method for numerical integration.

As seen in Table 3.1, only several of the packages contain a numerical method suitable for stiff o.d.e.s. The latest version of CSMP contains Gear's method.
IMP package contains Brandon's method. OSUSIM uses the trapezoidal rule which is A-stable. An equation-oriented package, FORSIM (Carver, 1972, 1973) developed at AECL uses Gear's and the Fowler-Warten method. Franks (1971, 1972b) recommends Euler's method for stiff systems. However, except for IMP, none of the packages are oriented to solving large systems of stiff o.d.e.s. In Chapter 6, we shall see a comparison of some of these techniques.
4. CONVERGENCE OF IMPLICIT METHODS

Of the methods developed to integrate stiff systems, implicit methods are usually preferred, but particular problems arise in solving the corrector or implicit equation, especially for large stiff systems.

Consider the general linear multistep formula:

\[ y_{n+1} = a_1 y_n + a_2 y_{n-1} + \ldots + a_k y_{n+1-k} \]
\[ + h [b_0 \dot{y}_{n+1} + b_1 \dot{y}_n + \ldots + b_k \dot{y}_{n+1-k}] \] (4.1)

or

\[ y_{n+1} = \sum_{i=1}^{k} a_i y_{n+1-i} + h \sum_{i=0}^{k} b_i \dot{y}_{n+1-i} \] (4.2)
\[ y_{n+1} = h B_0 y_{n+1} + \sum_{i=1}^{k} (a_i y_{n+1-i} + b_i \dot{y}_{n+1-i}) \] (4.3)

Since the last term is known, we can replace it by \( \omega_n \):

\[ y_{n+1} = h B_0 f(t_{n+1}, y_{n+1}) + \omega_n \] (4.4)
\[ y_{n+1} = h B_0 f(t_{n+1}, y_{n+1}) - \omega_n = 0 \] (4.5)
From the above equation we wish to determine $y_{n+1}$.

For a system of m o.d.e.s, this is equivalent to solving m nonlinear equations. Several common methods of doing this are outlined in this chapter, but this is an ongoing area of research and perhaps better methods will be found.

4.1 Jacobi Iteration

The obvious way of solving this is by repeated substitution:

$$y_{n+1} = hB_0 f(t_{n+1}, y_{n+1}) - w_n = 0 \quad (4.1.1)$$

until $y_{n+1}^{(e+1)}$ is equal to $y_{n+1}^{(e)}$ to the degree of accuracy desired.

This approach, called Jacobi iteration, is commonly used for nonstiff o.d.e.s but for stiff problems, it does not converge unless the step size is very small.
4.2 Accelerated Iteration

A modification of Jacobi iteration is:

\[
(1 + \alpha) y_{n+1}^{(s+1)} - h B_0 f(t_{n+1}, y_{n+1}^{(s)}) - \omega_n - a y_{n+1}^{(s)} = 0
\]  

(4.2.1)

where \( \alpha \) is an acceleration parameter.

This will increase the rate of convergence, but the step size condition for convergence is just as stringent as Jacobi iteration.

4.3 Backward Iteration

A backward iteration with much better convergence properties is:

\[
y_n^{(s+1)} - h B_0 f(t_{n+1}, y_{n+1}^{(s)}) - \omega_n = 0
\]  

(4.3.1)

however, this still requires the solution of nonlinear equations.
4.4 Newton-Raphson Iteration

The most commonly used method is Newton-Raphson iteration.

\[ g(y_{n+1}^{(e)}) = y_{n+1}^{(o)} - hB_o \frac{\partial f}{\partial y} (y_{n+1}^{(e)}, y_{n+1}) - \omega_n = 0 \]  \hspace{1cm} (4.4.1)

The Newton-Raphson formula is:

\[ y_{n+1} = y_{n+1}^{(o)} - \left[ \frac{\partial f}{\partial y} (y_{n+1}^{(e)}) \right]^{-1} g_{n+1} \]  \hspace{1cm} (4.4.2)

\[ y_{n+1} = y_{n+1}^{(o)} - \left[ 1 - hB_o \frac{\partial f}{\partial y} (y_{n+1}^{(e)}) \right]^{-1} \left[ y_{n+1}^{(o)} - hB_o f_{n+1}^{(e)} - \omega_n \right] \]  \hspace{1cm} (4.4.3)

The theoretical convergence condition is quite conservative, but in practice, the method converges for fairly large \( h \) and does not slow down the integration process.

\( y_{n+1}^{(o)} \) can be a predicted value or if no predictor equation is used, it can be taken as the previous value, \( y_n \).

Note that if the term to be inverted is singular, the step size \( h \) can always be reduced to make it non-singular.
4.5 Quasilinearization

The IMP package (Appendix B) uses a method called quasilinearization.

For a single equation:

\[ \dot{y} = f(t, y) \quad (4.5.1) \]

Expand the right hand side in a Taylor Series about the present point,

i.e.,

\[ \dot{y}_n = f(t_n, y_n) + \frac{\partial f}{\partial y} \bigg|_n (y - y_n) + \ldots \quad (4.5.2) \]

\[ \dot{y}_n = \frac{\partial f}{\partial y} \bigg|_n y + f(t_n, y_n) - \frac{\partial f}{\partial y} \bigg|_n y_n \quad (4.5.3) \]

\[ \dot{y}_n = Jy + B \quad (4.5.4) \]

For a system of equations, \( J \) is the Jacobian matrix, while \( B \) is called the augmented constant vector.

When the system is linearized in this way, we no longer have to solve a set of nonlinear equations but a set of linear equations at each time step.
4.6 Large Stiff Systems

Two useful methods for solving stiff corrector equations are Newton-Raphson and quasilinearization. Whichever of the two is used, a set of linear algebraic equations equal in size to the number of o.d.e.s must be solved.

Lapidus and Seinfeld (1971) recommend Jacobi or accelerated iteration if the stiffness ratio is of the order of 10. If it is greater than the order of 10, they recommend Newton-Raphson or backward iteration with the step size selected on the basis of the number of iterations desired per step.

In quasilinearization, the linear equations arise directly from the Taylor Series linearization of the system.

In Newton-Raphson iteration, we must evaluate the term:

\[
[I - h\beta_0 \left( \frac{a f}{\delta y} \right)_{n+1}]^{-1} \left[y_{n+1} - h\beta_0 f_{n+1} - \omega_n\right] \quad (4.6.1)
\]

In general this means solving a set of linear equations:

\[
AX = b \quad (4.6.2)
\]
where

\[ A = \left[ I - \hbar \frac{3f}{y} (e) \right] \quad (4.6.3) \]

\[ b = \left[ y_{n+1} - \hbar \alpha (e) \right] \quad (4.6.4) \]

and \( x = A^{-1} b \) \quad (4.6.5)

This set of linear equations must be solved for each corrector equation.

For large systems of O.D.E.s, the solution of this set of linear equations can require a large amount of computer time.

The matrix of coefficients is usually quite sparse; i.e., there are many zero elements. A method which takes advantage of the sparsity by storing and operating on only the nonzero elements will reduce the amount of computer time appreciably. Therefore we will examine ways of solving sparse sets of linear equations in the following chapter.
5. NUMERICAL SOLUTION OF SPARSE LINEAR ALGEBRAIC EQUATIONS

We saw in Chapter 4 that an implicit numerical integration technique for stiff o.d.e.s requires the solution of a set of linear algebraic equations equal in size to the number of o.d.e.s. For large systems of o.d.e.s, the solution of this set of linear equations can use a large amount of computer time.

The coefficient matrix of the linear equations is a function of the Jacobian matrix for the system. In general, the Jacobian matrix has many zeros so that using a method which stores and operates on only the nonzero elements will reduce the computer time greatly. Such methods are called sparse matrix techniques.

There are two major ways of solving linear equations: direct elimination techniques such as Gaussian elimination, and iterative methods such as Jacobi and Gauss-Seidel. Traditionally, iterative techniques have been used to handle sparse systems (Varga, 1962; Young, 1971); however, in recent years, elimination methods have been more commonly used (Reid, 1971; Rose and Willoughby, 1972; Westlake, 1968; Willoughby, 1969). Decomposition techniques (Himmelblau, 1973) where the system is broken into two or more subsystems have also been developed, but these require
extensive a priori analysis.

Several conventional methods and some sparse techniques were tested on a set of banded test examples. We will assume that our "sparse" equations contain at most 15 nonzeros per equation.

5.1 Methods Tested For Solving Linear Algebraic Equations

5.1.1 Conventional Methods

Five conventional methods were studied:

(1) MINV  An SSP matrix inversion algorithm (SSP, 1970)

(2) SOLVE  Gaussian elimination with no pivoting

(3) DECOMP-SOLVE  Gaussian elimination with partial pivoting (Forsythe and Moler, 1967)

(4) JACOBI  Jacobi iterative method (initial estimate of zero used)

(5) GAUSS-SEIDEL  Gauss-Seidel iterative method (initial estimate of zero used).
5.1.2 Key (1973)

Key (1973) has written a program called SIMULT based on Gauss-Jordan elimination with seven pivoting options:

(1) Simple Gauss-Jordan elimination. Diagonal elements are used in order as pivots.

(2) Gauss-Jordan partial pivoting. The largest coefficient in absolute value in each column in order is selected as pivot, provided this coefficient is not in a previously selected pivotal row.

(3) Gauss-Jordan full pivoting. The largest coefficient in the entire coefficient matrix that is not in a previously selected pivotal row or column is used.

(4) Minimum row-Minimum column. The coefficient matrix is searched to find the row with the least number of nonzero coefficients. Then for the nonzero elements in this row, the column with the most zeros is chosen as the pivotal column. If more than one row or column is chosen, the row or column with the smallest index is used.
(5) Minimum column-Minimum row. This is similar to the above except that the pivotal column is determined first.

(6) Maximum column-Minimum row. The column with the most nonzero elements is chosen, followed by the row with the least nonzero elements.

(7) Minimum of row entries times column entries. The pivot is chosen from the nonzero element with the smallest product of row and column entries.

The storage scheme is illustrated by the following example:

The matrix

\[ P = \begin{bmatrix}
3 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 1 \\
0 & 2 & 4 & 0 & 0 \\
1 & 0 & 0 & 2 & 0 \\
0 & 0 & 2 & 0 & 3
\end{bmatrix} \]  \hspace{1cm} (5.1.1)

is stored as

\[ A = \begin{bmatrix}
3 & 0 \\
8 & 1 \\
8 & 4 \\
1 & 2 \\
2 & 3
\end{bmatrix} \hspace{1cm} \text{and} \hspace{1cm} ICOL = \begin{bmatrix}
1 & 0 \\
2 & 5 \\
2 & 3 \\
1 & 4 \\
3 & 5
\end{bmatrix} \] \hspace{1cm} (5.1.2)
A contains the nonzero elements and ICOL the column index of the corresponding elements. The disadvantage of this is that one full row will make the A matrix as large as the original matrix.

5.1.3 IMP (1972)

IMP (Brandon, 1972) is a software system developed by D.M. Brandon at the University of Connecticut. It is now offered as a standard CDC software product through the Application Services Division of CDC. An object copy of IMP, Version 1.1, FTN version 4.0 compiler and optimization level 2 was used.

Five options with the IMP package were tested:

(1) Gauss-Seidel
(2) Gradient
(3) Crout elimination, fixed order, written for general sparse matrices
(4) Crout elimination, variable order, written for general sparse matrices
(5) Crout elimination, variable order, written for variable or constant banded matrices.
The Crout method is a variation of Gaussian elimination which uses less storage.

In addition, a program supplied by D.M. Brandon (Brandon, 1974c), which is a simplified version of the method (4) above, was used. It should run faster because there is none of the overhead associated with the IMP routines.

The storage scheme known as cumulative indexing is as follows:

The matrix

\[ P = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 \\ 0 & 2 & 4 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 3 \end{bmatrix} \] \hspace{1cm} (5.1.3)

is stored as

\[ A = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 2 \\ 4 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \quad JHA = \begin{bmatrix} 1 \\ 2 \\ 5 \\ 2 \\ 3 \\ 1 \\ 4 \\ 3 \end{bmatrix}, \quad ICUMA = \begin{bmatrix} 1 \\ 3 \\ 7 \\ 9 \end{bmatrix} \] \hspace{1cm} (5.1.4)

where \( A \) is the vector of nonzero entries of \( P \) by rows.
JHA is the vector of column numbers of nonzero elements for each row

ICUMA is a cumulative vector which separates JHA by rows.

5.1.4 Schappelle (1967)

Schappelle (1967) developed a program called LINEQ4 which is available from VIM, the CDC 6000 Series User Association.

The method first performs a systematic rearrangement, called pre-triangularization, of the rows and columns of the coefficient matrix. The equations then appear as:

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2
\end{bmatrix}
= 
\begin{bmatrix}
C_1 \\
C_2
\end{bmatrix}
\]  \hspace{1cm} (5.1.5)

where \(A\) is upper triangular, \(B\) and \(D\) are rectangular, and \(C\) is quasi lower triangular meaning no lead element of any row of the matrix is further to the right than the lead element of its preceding row.

\[
A X_1 + B X_2 = C_1 \\
C X_1 + D X_2 = C_2
\]  \hspace{1cm} (5.1.6)
solving for \( x_2 \)

\[
(D - CA^{-1}B)x_2 = (C_2 - CA^{-1}C_1) \quad (5.1.7)
\]

\( x_2 \) is solved by Gaussian elimination with full pivoting

\[
A x_1 = C_1 - Bx_2 \quad (5.1.8)
\]

\( x_1 \) is solved by back-substitution, since \( A \) is upper triangular.

The solution is corrected by iterating on the residuals.

The storage scheme is as illustrated below:

The matrix

\[
P = \begin{bmatrix}
  3 & 0 & 0 & 0 & 0 \\
  0 & 2 & 0 & 0 & 1 \\
  0 & 2 & 4 & 0 & 0 \\
  1 & 0 & 0 & 2 & 0 \\
  0 & 0 & 2 & 0 & 3
\end{bmatrix}
\quad (5.1.9)
\]
is stored as

\[
\begin{bmatrix}
1 & 1 & 3 \\
2 & 2 & 2 \\
2.5 & 1 & 4 \\
3 & 2 & 4 \\
3 & 3 & 4 \\
4 & 1 & 1 \\
4 & 4 & 2 \\
5 & 3 & 2 \\
5 & 5 & 3
\end{bmatrix}
\]

i.e., row number, column number, and value for each nonzero element in any order.

5.1.5 Bending-Hutchison (1973)

Bending and Hutchison (1973) developed the concept of an "operator list" to store the solution process by Gaussian elimination. The process consists of two stages, triangularization and back-substitution.

5.1.5.1 Triangularization

For the system

\[
\begin{bmatrix}
a_1 & 0 & a_9 & a_{11} \\
0 & a_6 & 0 & a_2 \\
a_4 & 0 & a_8 & 0 \\
a_5 & 0 & 0 & a_3
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{bmatrix}
= 
\begin{bmatrix}
a_7 \\
a_{10} \\
0 \\
0
\end{bmatrix}
\]  
(5.1.10)
(1) eliminate $a_2$, row 2 = row 2 - $\frac{a_2}{a_3}$ row 4

\[
\begin{bmatrix}
  a_1 & 0 & a_9 & a_{11} & a_7 \\
  a_{12} & a_8 & 0 & 0 & a_{10} \\
  a_4 & 0 & a_8 & 0 & 0 \\
  a_5 & 0 & 0 & a_3 & 0
\end{bmatrix}
\]

$a_{12} = 0 - \frac{a_2}{a_3} a_5 = -a_5 \frac{a_2}{a_3}$  \hspace{1cm} (5.1.11)

(2) eliminate $a_{11}$, row 1 = row 1 - $\frac{a_{11}}{a_3}$ row 4

\[
\begin{bmatrix}
  a_1 & 0 & a_9 & 0 & a_7 \\
  a_{12} & a_8 & 0 & 0 & a_{10} \\
  a_4 & 0 & a_8 & 0 & 0 \\
  a_5 & 0 & 0 & a_3 & 0
\end{bmatrix}
\]

$a_1^{**} = a_1 - a_5 \frac{a_{11}}{a_3}$  \hspace{1cm} (5.1.12)

(3) eliminate $a_8$, row 1 = row 1 - $\frac{a_8}{a_8}$ row 3

\[
\begin{bmatrix}
  a_1^{**} & 0 & 0 & 0 & a_7 \\
  a_{12} & a_8 & 0 & 0 & a_{10} \\
  a_4 & 0 & a_8 & 0 & 0 \\
  a_5 & 0 & 0 & a_3 & 0
\end{bmatrix}
\]

$a_1^{**} = a_1^{**} - a_4 \frac{a_8}{a_8}$  \hspace{1cm} (5.1.13)

The above process can be represented by a string of integers:
New element \[ a_i = - a_j \frac{a_k}{a_l} + k \text{i} \text{j} \] (5.1.14)

Element changed \[ a_i = a_i - a_j \frac{a_k}{a_l} + k \text{i} \text{j} \] (5.1.15)

For this example:

\[-2 \ 3 \ 12 \ 5 \ -11 \ 3 \ 1 \ 5 \ -9 \ 8 \ 1 \ 4 \ 0 \ 0 \] terminates string

5.1.5.2 Back-Substitution

\[
\begin{align*}
X_0 &= -1 \text{ (dummy variable)} \\
X_1 &= a_0 - a_1 X_0 / a_1 \\
X_2 &= (-a_{10} X_0 - a_{12} X_1) / a_6 \\
X_3 &= -a_4 X_1 / a_8 \\
X_4 &= -a_5 X_1 / a_3
\end{align*}
\]

This process can also be characterized by a string of integers:

\[ a_i = -a_j \frac{X_k}{a_l} + j \text{k} \text{i} \text{j} \] (5.1.17)

or

\[ a_i = (-a_j X_k - a_m X_n) / a_l + j \text{k} \text{m} \text{n} \text{i} \text{j} \] (5.1.18)

For this example:

\[ ? 0 \ -1 \ 1 \ 10 \ 0 \ 12 \ 1 \ -2 \ 6 \ 4 \ 1 \ -3 \ 8 \ 5 \ 1 \ -4 \ 3 \ 0 \ 0 \]
These two strings of integers form what is called the operator list, i.e., the particular solution process by Gaussian elimination. Subroutine TRGB solves the equations for the first time and creates the operator list which can be stored in core or on tape if the system is large. Subroutine TRGB2 can re-solve the system using only the operator list. If the zero elements remain zero and the nonzero elements change, the same operator list can be used to solve the new system. This is usually what occurs during numerical integration. The operator list could be set up on the first integration step by TRGB and used on later steps by TRGB2 to solve each new linear system. The operator list can change during the integration; however, in all examples tested, the original list could be used for the entire integration.

The pivot is chosen by the row and column each with the most zeros. The row is selected first. In case of ties, the largest element is chosen. The storage scheme is row number, column number, value.
5.2 Numerical Testing

Banded systems of 50, 100 and 200 linear equations with bandwidths 3, 5, 7, 9, 11 and 15 were used as test systems. The nonzero elements were generated randomly by rows from -10 to +10 using the FTN library random number generator RANF (Control Data, 1971). An initial seed of one was used. The right hand side was adjusted to yield a vector of ones as the solution.

The banded systems were used mainly for convenience; none of the algorithms tested takes advantage of this structure. However, with banded matrices, no new elements will be generated during the solution; thus the testing does not reflect how well the methods handle this problem. The Bending-Hutchison method, however, does handle this easily.

Execution times (See Appendix E) for the various methods are shown in Tables 5.2, 5.3 and 5.4. Table 5.1 provides a key to the methods.

The conventional methods are obviously unsuitable for these problems. Matrix inversion takes much time and Gaussian elimination is quite slow. The iterative methods, Jacobi iteration and Gauss-Seidel iteration both diverged from initial estimates of zero. None of the conventional techniques were able to solve 200 equations as this required more than the 49K of core available.
Of the sparse routines, many gave inaccurate answers. This may be attributed to the fact that the non-zero elements were generated over a wide range, producing badly conditioned systems. If instead the elements are generated from 1-2, these programs work quite well. The execution times were independent of the spread of the non-zero elements.

Of the seven pivoting options given by Key, the best was minimum-row, minimum-column which Key also recommends.

The Bending-Hutchison routine, TRGB2, gave the best performance of the sparse methods. This is almost to be expected since all of the work of determining the solution process has already been done by TRGB. The method is ideal for repeated solution of linear equations.

There is another program developed at IBM by Gustavson, Liniger and Willoughby (1970) in which the first run generates the FORTRAN code to solve the specific linear system. Further solutions of the same system structure can use the FORTRAN code already produced (Calahan, 1968a). This process has been implemented along with a modified version of Gear's method (Brayton, Gustavson and Hachtel (1972); Hachtel, Brayton and Gustavson, 1971) into the ECAP II Package (Branin et al., 1971).
The version of TRGB-TRGB2 used in the testing stored the operator list completely in core. For 200 equations, the method could not store the list for bandwidth of 11 or more. The list can be put into auxiliary storage however, but this will require more computer time to transfer the list or parts of it to and from core. If this is done, there would be no limit to the number of equations which can be solved. Table 5.5 gives the length of the operator list for the test systems.

As the bandwidth increases, one of the IMP options, method 17 of Table 5.1 becomes more competitive. Possibly as the bandwidth increases still further, the IMP method will become even faster than TRGB2. This opinion is held by Brandon (Brandon, 1974e).

Other algorithms not included in the numerical testing were a conjugate gradient program of Reid (1970), an elimination algorithm of Curtis and Reid (1977) and a program based on arc-graph structure (Rheinboldt, 1973).
Table 5.1: Key To Linear Equation Solvers

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CONVENTIONAL METHODS</td>
</tr>
<tr>
<td>1</td>
<td>(1) Matrix Inversion (MINV)</td>
</tr>
<tr>
<td>2</td>
<td>(2) Gaussian Elimination, No Pivoting (SOLVE)</td>
</tr>
<tr>
<td>3</td>
<td>(3) Gaussian Elimination, Partial Pivoting (DECOMP-SOLVE)</td>
</tr>
<tr>
<td>4</td>
<td>(4) Jacobi Iteration</td>
</tr>
<tr>
<td>5</td>
<td>(5) Gauss Seidel Iteration</td>
</tr>
<tr>
<td>6</td>
<td>KEY</td>
</tr>
<tr>
<td>7</td>
<td>(1) Simple Gauss-Jordan Elimination</td>
</tr>
<tr>
<td>8</td>
<td>(2) Gauss-Jordan Partial Pivoting</td>
</tr>
<tr>
<td>9</td>
<td>(3) Gauss-Jordan Full Pivoting</td>
</tr>
<tr>
<td>10</td>
<td>(4) Minimum Row-Minimum Column</td>
</tr>
<tr>
<td>11</td>
<td>(5) Minimum Column-Minimum Row</td>
</tr>
<tr>
<td>12</td>
<td>(6) Maximum Column-Minimum Row</td>
</tr>
<tr>
<td>13</td>
<td>(7) Minimum Of Row Entries Times Column Entries</td>
</tr>
<tr>
<td>14</td>
<td>IMP</td>
</tr>
<tr>
<td>15</td>
<td>(1) Gauss-Seidel</td>
</tr>
<tr>
<td>16</td>
<td>(2) Gradient</td>
</tr>
<tr>
<td>17</td>
<td>(3) Crout, Fixed Order, General Sparse</td>
</tr>
<tr>
<td>18</td>
<td>(4) Crout, Variable Order, General Sparse</td>
</tr>
<tr>
<td>19</td>
<td>(5) Crout, Variable Order, Variable Or Constant Band</td>
</tr>
<tr>
<td>20</td>
<td>MISCELLANEOUS</td>
</tr>
<tr>
<td>21</td>
<td>(1) Brandon, Crout, Variable Order, General Sparse</td>
</tr>
<tr>
<td>21</td>
<td>(2) LINEQ4</td>
</tr>
<tr>
<td>21</td>
<td>(3) TRGB</td>
</tr>
<tr>
<td>21</td>
<td>(4) TRGB2</td>
</tr>
</tbody>
</table>
Table 5.2: Execution Times for 50 Linear Equations

<table>
<thead>
<tr>
<th>NO.</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.17</td>
<td>5.15</td>
<td>5.13</td>
<td>5.15</td>
<td>5.18</td>
<td>5.20</td>
<td>5.16</td>
</tr>
<tr>
<td>2</td>
<td>0.838</td>
<td>0.850</td>
<td>0.816</td>
<td>0.833</td>
<td>0.819</td>
<td>0.818</td>
<td>0.818</td>
</tr>
<tr>
<td>3</td>
<td>0.790</td>
<td>0.791</td>
<td>0.780</td>
<td>0.779</td>
<td>0.755</td>
<td>0.764</td>
<td>0.756</td>
</tr>
<tr>
<td>4</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>5</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>6</td>
<td>0.647</td>
<td>0.798</td>
<td>0.935</td>
<td>1.09</td>
<td>1.27</td>
<td>1.44</td>
<td>1.62</td>
</tr>
<tr>
<td>7</td>
<td>0.748</td>
<td>0.986</td>
<td>1.23</td>
<td>1.38</td>
<td>1.64</td>
<td>1.91</td>
<td>2.14</td>
</tr>
<tr>
<td>8</td>
<td>0.523</td>
<td>1.79</td>
<td>2.30</td>
<td>3.27</td>
<td>5.05</td>
<td>5.39</td>
<td>5.15</td>
</tr>
<tr>
<td>9</td>
<td>0.215</td>
<td>0.327</td>
<td>0.495</td>
<td>0.691</td>
<td>1.2I</td>
<td>1.36SI</td>
<td>1.36SI</td>
</tr>
<tr>
<td>10</td>
<td>0.293</td>
<td>0.437</td>
<td>0.612</td>
<td>0.814</td>
<td>1.06</td>
<td>1.31SI</td>
<td>1.60</td>
</tr>
<tr>
<td>11</td>
<td>0.766</td>
<td>0.970</td>
<td>1.22</td>
<td>1.49</td>
<td>1.78</td>
<td>2.06</td>
<td>2.39</td>
</tr>
<tr>
<td>12</td>
<td>0.320</td>
<td>0.538</td>
<td>0.736</td>
<td>0.994</td>
<td>1.28</td>
<td>1.58SI</td>
<td>1.88</td>
</tr>
<tr>
<td>13</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>14</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>15</td>
<td>0.112</td>
<td>0.161</td>
<td>0.243</td>
<td>0.373</td>
<td>0.537</td>
<td>0.771</td>
<td>1.02</td>
</tr>
<tr>
<td>16</td>
<td>0.101</td>
<td>0.158</td>
<td>0.223</td>
<td>0.298</td>
<td>0.391</td>
<td>0.509</td>
<td>0.645</td>
</tr>
<tr>
<td>17</td>
<td>0.047</td>
<td>0.060</td>
<td>0.080</td>
<td>0.092</td>
<td>0.111</td>
<td>0.131</td>
<td>0.138</td>
</tr>
<tr>
<td>18</td>
<td>0.073</td>
<td>0.097</td>
<td>0.161</td>
<td>0.234</td>
<td>0.320</td>
<td>0.424</td>
<td>0.564</td>
</tr>
<tr>
<td>19</td>
<td>0.960I</td>
<td>1.35I</td>
<td>1.68I</td>
<td>2.04I</td>
<td>2.46I</td>
<td>2.97I</td>
<td>3.52I</td>
</tr>
<tr>
<td>20</td>
<td>0.185</td>
<td>0.268</td>
<td>0.346</td>
<td>0.453</td>
<td>0.550</td>
<td>0.674</td>
<td>0.797</td>
</tr>
<tr>
<td>21</td>
<td>0.012</td>
<td>0.023</td>
<td>0.033</td>
<td>0.049</td>
<td>0.065</td>
<td>0.086</td>
<td>0.106</td>
</tr>
</tbody>
</table>

I - inaccurate
SI - slightly inaccurate (1-3 figure accuracy)
DIV - diverged
Table 5.3: Execution Times For 100 Linear Equations

<table>
<thead>
<tr>
<th>NO.</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40.1</td>
<td>39.8</td>
<td>40.0</td>
<td>39.8</td>
<td>39.7</td>
<td>39.8</td>
<td>40.6</td>
</tr>
<tr>
<td>2</td>
<td>6.45</td>
<td>6.43</td>
<td>6.37</td>
<td>6.28</td>
<td>6.32</td>
<td>6.26</td>
<td>6.26</td>
</tr>
<tr>
<td>3</td>
<td>5.81</td>
<td>5.78</td>
<td>5.72</td>
<td>5.67</td>
<td>5.62</td>
<td>5.59</td>
<td>5.54</td>
</tr>
<tr>
<td>4</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>5</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>6</td>
<td>2.60</td>
<td>3.15</td>
<td>3.71</td>
<td>4.32</td>
<td>4.94</td>
<td>5.57</td>
<td>6.20</td>
</tr>
<tr>
<td>7</td>
<td>2.07</td>
<td>3.80</td>
<td>4.79</td>
<td>5.69</td>
<td>6.40</td>
<td>7.47</td>
<td>8.33</td>
</tr>
<tr>
<td>8</td>
<td>1.91</td>
<td>8.68</td>
<td>11.6</td>
<td>15.1</td>
<td>24.7</td>
<td>24.9</td>
<td>34.9</td>
</tr>
<tr>
<td>9</td>
<td>0.722 I</td>
<td>1.05 I</td>
<td>1.49 I</td>
<td>1.95 I</td>
<td>2.48 I</td>
<td>3.10 I</td>
<td>3.76 I</td>
</tr>
<tr>
<td>10</td>
<td>1.02 I</td>
<td>1.30 I</td>
<td>1.84 I</td>
<td>2.35 I</td>
<td>2.95 I</td>
<td>3.65 I</td>
<td>4.37 I</td>
</tr>
<tr>
<td>11</td>
<td>3.05</td>
<td>3.81</td>
<td>4.64</td>
<td>5.51</td>
<td>6.43</td>
<td>7.42</td>
<td>8.47</td>
</tr>
<tr>
<td>12</td>
<td>1.22</td>
<td>1.01</td>
<td>2.51</td>
<td>3.25</td>
<td>4.08</td>
<td>4.97</td>
<td>5.86</td>
</tr>
<tr>
<td>13</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>14</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>15</td>
<td>0.179</td>
<td>0.286</td>
<td>0.463</td>
<td>0.741</td>
<td>1.10</td>
<td>1.58</td>
<td>2.20</td>
</tr>
<tr>
<td>16</td>
<td>0.277</td>
<td>0.379</td>
<td>0.518</td>
<td>0.674</td>
<td>0.879</td>
<td>1.13</td>
<td>1.43</td>
</tr>
<tr>
<td>17</td>
<td>0.118</td>
<td>0.144</td>
<td>0.178</td>
<td>0.216</td>
<td>0.249</td>
<td>0.295</td>
<td>0.327</td>
</tr>
<tr>
<td>18</td>
<td>0.152</td>
<td>0.247</td>
<td>0.350</td>
<td>0.503</td>
<td>0.690</td>
<td>0.923</td>
<td>1.20</td>
</tr>
<tr>
<td>19</td>
<td>2.36 I</td>
<td>3.87 I</td>
<td>5.03 I</td>
<td>6.40 I</td>
<td>7.95 I</td>
<td>9.74 I</td>
<td>11.7 I</td>
</tr>
<tr>
<td>20</td>
<td>0.651</td>
<td>0.933</td>
<td>1.24</td>
<td>1.54</td>
<td>1.88</td>
<td>2.25</td>
<td>2.67</td>
</tr>
<tr>
<td>21</td>
<td>0.023</td>
<td>0.044</td>
<td>0.068</td>
<td>0.101</td>
<td>0.138</td>
<td>0.181</td>
<td>0.229</td>
</tr>
</tbody>
</table>

I = inaccurate  SI = slightly inaccurate (1-3 figure accuracy)  
DIV = diverged
Table 5.4: Execution Times For 200 Linear Equations

<table>
<thead>
<tr>
<th>No.</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10.4</td>
<td>12.5</td>
<td>14.8</td>
<td>17.1</td>
<td>19.4</td>
<td>21.8</td>
<td>24.2</td>
</tr>
<tr>
<td>7</td>
<td>6.17</td>
<td>15.5</td>
<td>20.0</td>
<td>23.3</td>
<td>26.4</td>
<td>29.9</td>
<td>33.8</td>
</tr>
<tr>
<td>8</td>
<td>6.64</td>
<td>33.5</td>
<td>60.3</td>
<td>95.2</td>
<td>110.0</td>
<td>147.7</td>
<td>148.2</td>
</tr>
<tr>
<td>9</td>
<td>2.60</td>
<td>3.57</td>
<td>4.70</td>
<td>5.98</td>
<td>7.43</td>
<td>9.09</td>
<td>10.8</td>
</tr>
<tr>
<td>10</td>
<td>3.85</td>
<td>4.85</td>
<td>6.05</td>
<td>7.51</td>
<td>8.97</td>
<td>10.7</td>
<td>12.7</td>
</tr>
<tr>
<td>11</td>
<td>12.2</td>
<td>15.0</td>
<td>18.0</td>
<td>21.1</td>
<td>24.4</td>
<td>27.7</td>
<td>31.1</td>
</tr>
<tr>
<td>12</td>
<td>4.56</td>
<td>6.59</td>
<td>8.81</td>
<td>11.2</td>
<td>13.7</td>
<td>16.5</td>
<td>19.3</td>
</tr>
<tr>
<td>13</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>14</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
<td>DIV.</td>
</tr>
<tr>
<td>15</td>
<td>0.293</td>
<td>0.530</td>
<td>0.897</td>
<td>1.46</td>
<td>2.22</td>
<td>3.24</td>
<td>4.52</td>
</tr>
<tr>
<td>16</td>
<td>0.816</td>
<td>1.02</td>
<td>1.28</td>
<td>1.62</td>
<td>2.04</td>
<td>2.56</td>
<td>3.17</td>
</tr>
<tr>
<td>17</td>
<td>0.315</td>
<td>0.372</td>
<td>0.440</td>
<td>0.506</td>
<td>0.577</td>
<td>0.650</td>
<td>0.743</td>
</tr>
<tr>
<td>18</td>
<td>0.417</td>
<td>0.587</td>
<td>0.824</td>
<td>1.15</td>
<td>1.53</td>
<td>2.00</td>
<td>2.59</td>
</tr>
<tr>
<td>19</td>
<td>7.26</td>
<td>13.2</td>
<td>17.5</td>
<td>22.7</td>
<td>28.5</td>
<td>35.1</td>
<td>42.3</td>
</tr>
<tr>
<td>20</td>
<td>2.45</td>
<td>3.47</td>
<td>4.51</td>
<td>5.59</td>
<td>T.L.</td>
<td>T.L.</td>
<td>T.L.</td>
</tr>
<tr>
<td>21</td>
<td>0.046</td>
<td>0.086</td>
<td>0.137</td>
<td>0.201</td>
<td>T.L.</td>
<td>T.L.</td>
<td>T.L.</td>
</tr>
</tbody>
</table>

1 - inaccurate  S1 - slightly inaccurate (1-3 figure accuracy)  T.L. - too large (requires more than 49K)  DIV - diverged
Table 5.5: Length of TRGB Operator List

<table>
<thead>
<tr>
<th>BANDWIDTH</th>
<th>50</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>600</td>
<td>1200</td>
<td>2400</td>
</tr>
<tr>
<td>5</td>
<td>1176</td>
<td>2376</td>
<td>4776</td>
</tr>
<tr>
<td>7</td>
<td>1928</td>
<td>3928</td>
<td>7928</td>
</tr>
<tr>
<td>9</td>
<td>2848</td>
<td>5848</td>
<td>11848</td>
</tr>
<tr>
<td>11</td>
<td>3928</td>
<td>8128</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>5160</td>
<td>10760</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>6536</td>
<td>13736</td>
<td>-</td>
</tr>
</tbody>
</table>
6. NUMERICAL TESTING OF STIFF TECHNIQUES

6.1 Testing By Other Workers

Several numerical comparisons of methods for solving stiff o.d.e.s have been made.


Lapidus and Seinfeld (1971), also Seinfeld, Lapidus and Hwang (1970), have compared fourth order Runge-Kutta, fourth order Adams, Treanor's method (Treanor, 1966), the modified midpoint rule, the trapezoidal rule with and without extrapolation, Calahan's method (Calahan, 1968b) and two methods of Liniger and Willoughby (1970). They found the last four implicit methods to be superior and roughly comparable in terms of accuracy and computing time. For highly stiff systems, they recommend one of Liniger and Willoughby's techniques. Four test examples were used.
Brandon (1972, 1974f) compares his own method with the stiff techniques compared by Lapidus and Seinfeld and finds his method to be superior. He uses two test examples.

Hronsky and Martens (1973) compare Runge-Kutta-Newton (Kreyszig, 1967) which is the pseudo steady state approach used with the fourth order Runge-Kutta method, Treanor's method (Treanor, 1966), a method of Liniger and Willoughby (1970), and the backward differentiation formula (Brayton et al., 1972) which is a modification of Gear's method (Appendix A). Based on five test examples, they recommend the methods in the order above, except if the number of equations is less than six, Liniger and Willoughby's method is recommended over that of Treanor.

Hull and Enright have been doing extensive comparisons of stiff techniques, but their final results are not yet available. Their preliminary testing indicated the three best methods to be Gear's method, trapezoidal rule with extrapolation and the second derivative multi-step method (Enright, 1974).

Several other comparisons appear in the literature but, unfortunately, most authors of new techniques compare their algorithm only with an inefficient conventional method such as fourth order Runge-Kutta or with a stiff technique very similar to theirs, to demonstrate a
superior modification.

6.2 Test Methods and Examples

The following methods were selected for numerical testing:

four conventional methods none of which are oriented to solving stiff systems:

(1) Euler
(2) Adams-Moulton-Shell (Shell, 1958)
(3) Runge-Kutta-Merson (Merson, 1957)
(4) Gear's nonstiff option (Appendix A)

four explicit stiff techniques:

(1) Richards, Lanning and Torrey (1965)
(2) Nigro (1969)
(3) Treanor (1966)
(4) Fowler and Warten (1967)

four implicit stiff techniques:

(1) Klopfenstein and Davis (1971)
(2) Sandberg and Shichman (1968)
(3) Brandon (1972, 1974f)
(4) Gear's stiff option (Appendix A).
The IMP package (Appendix B) containing Brandon's method was also tested. Obviously it is impractical to test all the methods surveyed in Chapter 3, but these provide a cross-section of promising explicit and implicit techniques. The explicit methods have the advantage of avoiding the solution of linear algebraic equations.

The four implicit methods used matrix inversion (MINV of Chapter 5) in solving the corrector. Gear's method was also used with two other linear equation solvers: DECOMP-SOLVE and TRGB-TRGB2 described in Chapter 5. TRGB was used only on the first time step to set up the operator list and TRGB2 used thereafter.

Eleven test systems were used (Appendix D). They can be divided into three groups: 

1. small stiff systems: I, II, III, IV, V
2. large stiff systems: VI, VII
3. nonstiff systems: VIII, IX, X, XI

Each of the fifteen test programs were tested on each example. Ideally the same method could handle both nonstiff and stiff methods efficiently; otherwise, two methods would have to be used.

The two complex eigenvalue systems III and VIII had small imaginary parts to avoid introducing the problem
of simulating rapidly oscillating components.

Most of the methods used had a local truncation error estimate: Gear, IMP, Brandon, Fowler-Warten, Klopfenstein-Davis, Sandberg-Shichman, Adams-Moulton-Shell and Merson. Gear's error control is described in Gear (1971a,b) and briefly in Appendix A. IMP uses a complex algorithm (Brandon, 1972, 1974f) to calculate the step size. In all of these methods, however, the maximum relative local error was kept below an upper tolerance and usually above a lower tolerance (except for Gear and IMP). The upper tolerance used was $10^{-3}$ for systems I-X and $10^{-6}$ for system XI. The lower tolerance was taken as one-tenth of the upper tolerance. Relative errors were used with the absolute error estimate divided by the maximum value of the corresponding independent variable. This maximum value had a minimum of 1.0, so that in effect, relative errors were used for large $y$ values and absolute errors for small $y$ values. Fowler-Warten had a complex error control scheme, but it was not used here. The last six methods mentioned above used the double-halving approach described in Section 2.7.3.

Three methods had no error estimate: Euler, Nigro and Treanor. Lapidus and Seinfeld (1971) recommend the following error control scheme for Treanor's method. If the relative change in $y$, $|y_{n+2} - y_n|/|y_{n+1}|$, is greater than
some upper tolerance, the step size is halved and the step repeated. If it is greater than some lower tolerance, the step is doubled. Tolerances of 0.05 and 0.005 were used. This strategy gave unstable results for systems V, VI and VII. It was altered for them to keeping

$$0.001 < \left| \frac{y_{n+1} - y_n}{y_{n+1} + R} \right| < 0.01$$

where $$R = 0.1$$ if $$y_{n+1} < 0.5$$

$$= 0.0$$ otherwise

Euler and Nigro used a fixed step size. The critical step size was used for the stiff examples, i.e., the step size just before the onset of instability. For the nonstiff examples, the step size was adjusted to keep the final error below 1%.

For Nigro's method, the parameters corresponding to $$h\lambda = 120$$ were used. Euler's method was used to start the integration. A step size of 0.01 of the Nigro step size was used to calculate the starting points. This was not included in the execution time.

The first order Klopfenstein-Davis algorithm tested better than the second order. Tests with various coefficients of $$\alpha, \beta, u, v$$ and $$\alpha$$ yielded very similar results so the values $$\alpha = 1.0, \beta = 0.0, u = 0.25, v = 0.75$$ and $$\alpha = 0.6667$$ were used. See Equation (3.3.10).
6.3 Test Results

Global errors are given at the end of the simulation with the exception of systems II, V, VI and VII. There, the error is given after 10% of the simulation, since it was expected several methods could require near infinite computer time to reach the end of the integration.

Analytical solutions are available on systems I, II, III, IV, VII and IX to calculate the error. For the remaining examples, V, VI, VII, X and XI, the Euler method with \( h = 0.00005 \) was used to get an accurate solution. This was arbitrarily taken as the "correct" solution. In each case with \( h = 0.0001 \), the results differed at worst in the fourth significant figure.

None of the examples came close to steady state, so an error taken at the end of the simulation will be reasonably indicative of the accuracy of the trajectory.

Table 6.1 shows the execution times and errors for the explicit methods and the Euler step size. Tables 6.2 and 6.3 show results for the explicit and implicit stiff methods respectively. Nigro's step size is also shown. Table 6.4 shows IMP and three different linear equation solvers in conjunction with Gear's method. Gear-MINV is repeated from Table 6.3. See Appendix E for a discussion on computer timings.
In general the conventional methods did not perform well on the stiff examples. As expected, Euler's method was the least inefficient, but obviously a method designed specifically for stiff systems is to be preferred. Some of the runs especially on systems II, V, VI and VII were a waste of computer time, as the methods were so inefficient. The conventional methods are not of great concern here except to note how well the stiff methods were able to solve the nonstiff examples by comparison. They do reasonably well with respect to execution time and error. Since the execution times are so small anyway, it would suffice to use a stiff method for both nonstiff and stiff examples.

Hull et al. (1972a) consider a method of Krogh (1969) very good for nonstiff systems, but they rate Gear's (nonstiff option) closely behind it. In our limited testing of nonstiff examples here, Runge-Kutta-Merson did very well. AMOS could not handle system XI, probably because of the extremely small tolerance demanded.

In general, the explicit stiff methods did not perform as well as the implicit methods. Of the stiff methods, Gear's was easily the best (Table 6.3). Brandon (1974d) has improved his error estimation procedure recently and it does much better, but it was not used in our testing. In particular, the latest version of the IMP package
integrates system VI and VII in 22.2 and 6.68 seconds respectively on the CDC 6600. These times would roughly be multiplied by three for comparable CYBER 73 times.

The Fowler-Warten method handled the complex eigenvalue systems III and VIII reasonably well in spite of the fact that it is intended for systems with the largest eigenvalue real.

Neither of the step-changing strategies used with Treanor's method were very efficient. Possibly a fixed step size would be a better strategy.

Some of the methods simulated only the extremely small components inaccurately especially in Systems V and X. In these cases, the second largest error is given in brackets.

Different methods of handling the corrector are illustrated in Table 6.4 with Gear's method. For less than about five equations, matrix inversion is fastest, then Gaussian elimination is preferable up to perhaps ten equations, but for large systems a sparse method such as the Bending-Hutchison algorithm is necessary, as shown by the results for Systems VI and VII. The difference in execution time is slight for the small examples.

System XI ran slightly faster with the tridiagonal option (0.753 seconds) rather than with TRGB-TRGB2 (0.980 seconds). The time savings will increase with the size of the system as shown in Section 8.4.
If we had to choose one method to handle all systems stiff and nonstiff, large and small, it would be Gear's method with stiff option and the Bending-Hutchison method to solve the corrector. It handles small stiff systems and nonstiff systems efficiently enough since little computer time is required anyway. However, the nonstiff option is also available in the same program and only a switch is required to change from one to the other, so that both stiff and nonstiff options are conveniently available. Different coefficients are used, but most of the code is similar for both options.
Table 6.1: Execution Times And Errors For Conventional Methods

<table>
<thead>
<tr>
<th>TEST SYSTEM</th>
<th>METHOD</th>
<th>STEP SIZE</th>
<th>ERROR</th>
<th>AMOS</th>
<th>ERROR</th>
<th>RKM</th>
<th>ERROR</th>
<th>GEAR NONSTIFF</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>EULER</td>
<td>1.9x10^-3</td>
<td>9.50x10^-4</td>
<td>0.872</td>
<td>1.88x10^-3</td>
<td>0.301</td>
<td>3.72x10^-4</td>
<td>2.30</td>
<td>1.13x10^-3</td>
</tr>
<tr>
<td>II</td>
<td>71.7</td>
<td>1.9x10^-6</td>
<td>9.51x10^-8</td>
<td>866.0</td>
<td>1.01x10^-3</td>
<td>300.3</td>
<td>1.36x10^-4</td>
<td>2188.0</td>
<td>3.81x10^-4</td>
</tr>
<tr>
<td>III</td>
<td>0.325</td>
<td>1.9x10^-3</td>
<td>4.28x10^-3</td>
<td>2.06</td>
<td>4.29x10^-3</td>
<td>1.18</td>
<td>3.55x10^-3</td>
<td>4.90</td>
<td>4.05x10^-3</td>
</tr>
<tr>
<td>IV</td>
<td>1.47</td>
<td>1.9x10^-3</td>
<td>9.62x10^-6</td>
<td>9.09</td>
<td>5.65x10^-4</td>
<td>6.02</td>
<td>7.27x10^-6</td>
<td>36.4</td>
<td>2.89x10^-5</td>
</tr>
<tr>
<td>V</td>
<td>17.1</td>
<td>5.0x10^-4</td>
<td>3.91x10^-5</td>
<td>240.0</td>
<td>114.6</td>
<td>79.0</td>
<td>154.6</td>
<td>431.6</td>
<td>0.368</td>
</tr>
<tr>
<td>VI</td>
<td>U^1</td>
<td>2.5x10^-4</td>
<td>4.21x10^-4</td>
<td>&gt;1500</td>
<td>9.95x10^-3</td>
<td>&gt;1000</td>
<td>154.6</td>
<td>431.6</td>
<td>(3.95x10^-4)</td>
</tr>
<tr>
<td>VII</td>
<td>23.0</td>
<td>1.9x10^-3</td>
<td>6.0x10^-3</td>
<td>4.55x10^-3</td>
<td>353.0</td>
<td>0.855²</td>
<td>167.8</td>
<td>7.94x10^-2</td>
<td>656.0</td>
</tr>
<tr>
<td>VIII</td>
<td>0.530</td>
<td>2.0x10^-3</td>
<td>9.58x10^-3</td>
<td>0.059</td>
<td>6.69x10^-3</td>
<td>0.021</td>
<td>1.89x10^-3</td>
<td>0.101</td>
<td>8.15x10^-4</td>
</tr>
<tr>
<td>IX</td>
<td>1.04</td>
<td>5.0x10^-3</td>
<td>9.48x10^-3</td>
<td>0.107</td>
<td>9.62x10^-3</td>
<td>0.041</td>
<td>1.56x10^-2</td>
<td>0.213</td>
<td>6.67x10^-3</td>
</tr>
<tr>
<td>X</td>
<td>0.636</td>
<td>2.0x10^-3</td>
<td>9.75x10^-3</td>
<td>0.059</td>
<td>3.48</td>
<td>0.022</td>
<td>0.394</td>
<td>0.164</td>
<td>4.39x10^-2</td>
</tr>
<tr>
<td>XI</td>
<td>1.39</td>
<td>2.5x10^-3</td>
<td>9.45x10^-3</td>
<td>&gt;320</td>
<td>0.263</td>
<td>4.87x10^-4</td>
<td>1.30</td>
<td>2.25x10^-4</td>
<td></td>
</tr>
</tbody>
</table>

¹unstable
²only extremely small components inaccurate
<table>
<thead>
<tr>
<th>TEST SYSTEM</th>
<th>METHOD</th>
<th>RLT</th>
<th>ERROR</th>
<th>NIGNO</th>
<th>STEP SIZE</th>
<th>ERROR</th>
<th>TREANOR</th>
<th>ERROR</th>
<th>FOWLER</th>
<th>WARTEN</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0.092</td>
<td>4.83x10^{-3}</td>
<td>0.144</td>
<td>1.8x10^{-3}</td>
<td>5.35x10^{-3}</td>
<td>0.384</td>
<td>2.24x10^{-4}</td>
<td>0.495</td>
<td>2.24x10^{-3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>0.068</td>
<td>5.03x10^{-4}</td>
<td>26.5</td>
<td>1.9x10^{-5}</td>
<td>2.15x10^{-2}</td>
<td>0.393</td>
<td>8.90x10^{-7}</td>
<td>49.1</td>
<td>3.98x10^{-2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>III</td>
<td>3.32</td>
<td>3.81x10^{-3}</td>
<td>0.498</td>
<td>1.8x10^{-3}</td>
<td>6.13x10^{-3}</td>
<td>1.38</td>
<td>5.0x10^{-4}</td>
<td>1.48</td>
<td>7.49x10^{-5}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IV</td>
<td>0.356</td>
<td>5.03x10^{-3}</td>
<td>0.817</td>
<td>5.0x10^{-3}</td>
<td>6.88x10^{-3}</td>
<td>6.90</td>
<td>1.85x10^{-3}</td>
<td>0.317</td>
<td>1.79x10^{-2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>0.078</td>
<td>343.0</td>
<td>18.5</td>
<td>1.0x10^{-3}</td>
<td>1.06x10^{-2}</td>
<td>0.505</td>
<td>3.56x10^{-4}</td>
<td>0.624</td>
<td>8.42x10^{-3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VI</td>
<td>36.5</td>
<td>8.86²</td>
<td>464.0</td>
<td>1.0x10^{-3}</td>
<td>3.35x10^{-2}</td>
<td>y¹</td>
<td>4.41x10^{-2}</td>
<td>117.0</td>
<td>2.48x10^{-2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VII</td>
<td>3.58</td>
<td>2.03²</td>
<td>46.5</td>
<td>1.0x10^{-2}</td>
<td>1.73x10^{-2}</td>
<td>219.0</td>
<td>8.24x10^{-2}</td>
<td>75.2</td>
<td>1.41x10^{-2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIII</td>
<td>-0.159</td>
<td>8.61x10^{-2}</td>
<td>1.24</td>
<td>2.0x10^{-3}</td>
<td>9.72x10^{-3}</td>
<td>0.927</td>
<td>1.97x10^{-5}</td>
<td>0.080</td>
<td>2.64x10^{-2}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td>0.317</td>
<td>5.45x10^{-2}</td>
<td>2.02</td>
<td>4.0x10^{-3}</td>
<td>1.05x10^{-2}</td>
<td>2.69</td>
<td>3.13x10^{-9}</td>
<td>0.977</td>
<td>7.07x10^{-3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>0.118</td>
<td>0.630</td>
<td>1.36</td>
<td>2.5x10^{-3}</td>
<td>2.64x10^{-2}</td>
<td>2.36</td>
<td>2.46x10^{-4}</td>
<td>0.060</td>
<td>1.83x10^{-3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XI</td>
<td>0.744</td>
<td>3.00x10^{-2}</td>
<td>5.65</td>
<td>1.0x10^{-2}</td>
<td>1.63x10^{-2}</td>
<td>5.07</td>
<td>4.06x10^{-5}</td>
<td>0.666</td>
<td>1.86x10^{-4}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹unstable
²large components reasonably accurate
<table>
<thead>
<tr>
<th>TEST SYSTEM</th>
<th>METHOD</th>
<th>KLOP</th>
<th>DAVIS</th>
<th>ERROR</th>
<th>SAND</th>
<th>SCH.</th>
<th>ERROR</th>
<th>BRANDON</th>
<th>ERROR</th>
<th>GEAR</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2.27</td>
<td>5.85x10^-4</td>
<td>0.274</td>
<td>1.21x10^-2</td>
<td>0.124</td>
<td>1.22x10^-3</td>
<td>0.139</td>
<td>1.07x10^-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>&gt;1000</td>
<td>5.00x10^-8</td>
<td>0.276</td>
<td>7.10x10^-4</td>
<td>0.260</td>
<td>3.98x10^-4</td>
<td>0.196</td>
<td>1.01x10^-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>III</td>
<td>7.11</td>
<td>2.67x10^-3</td>
<td>0.788</td>
<td>4.64x10^-2</td>
<td>4.24</td>
<td>3.97x10^-3</td>
<td>0.318</td>
<td>3.77x10^-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IV</td>
<td>39.2</td>
<td>5.90x10^-6</td>
<td>1.75</td>
<td>9.41x10^-4</td>
<td>1.11</td>
<td>5.66x10^-4</td>
<td>0.505</td>
<td>1.16x10^-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>845.0</td>
<td>0.267</td>
<td>0.117</td>
<td>1.27x10^-4</td>
<td>0.459</td>
<td>3.10x10^-5</td>
<td>0.168</td>
<td>4.57x10^-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VI</td>
<td>&gt;1000</td>
<td>626.0</td>
<td>761.0</td>
<td>6.83x10^-3</td>
<td>&gt;1500</td>
<td>7.72x10^-4</td>
<td>81.7</td>
<td>8.47x10^-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VII</td>
<td>&gt;1000</td>
<td>0.133</td>
<td>299.0</td>
<td>1.66x10^-2</td>
<td>42.0</td>
<td>1.71x10^-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIII</td>
<td>0.760</td>
<td>0.206</td>
<td>0.721</td>
<td>0.204</td>
<td>0.294</td>
<td>0.308 (0.023)</td>
<td>0.078</td>
<td>3.23x10^-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td>0.788</td>
<td>0.335</td>
<td>0.830</td>
<td>0.316</td>
<td>0.546</td>
<td>9.68x10^-3</td>
<td>0.192</td>
<td>8.07x10^-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>0.473</td>
<td>0.936</td>
<td>0.455</td>
<td>0.870</td>
<td>0.313</td>
<td>8.13x10^-3</td>
<td>0.119</td>
<td>1.38 (2.74x10^-3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XI</td>
<td>29.1</td>
<td>3.78x10^-2</td>
<td>29.1</td>
<td>3.75x10^-2</td>
<td>2.44</td>
<td>1.77x10^-4</td>
<td>1.03</td>
<td>4.85x10^-5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6.4: Execution Times And Errors For Gear's Method And IMP

<table>
<thead>
<tr>
<th>TEST SYSTEM</th>
<th>METHOD</th>
<th>GEAR MINIV</th>
<th>GEAR DECOMP-SOLVE</th>
<th>GEAR TRGB</th>
<th>ERROR</th>
<th>IMP</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td>0.139</td>
<td>0.148</td>
<td>0.162</td>
<td>1.07x10^-3</td>
<td>0.139</td>
<td>1.85x10^-3</td>
</tr>
<tr>
<td>II</td>
<td></td>
<td>0.196</td>
<td>0.216</td>
<td>0.235</td>
<td>1.01x10^-4</td>
<td>0.373</td>
<td>2.04x10^-4</td>
</tr>
<tr>
<td>III</td>
<td></td>
<td>0.318</td>
<td>0.294</td>
<td>0.315</td>
<td>3.77x10^-3</td>
<td>3.72</td>
<td>1.06x10^-5</td>
</tr>
<tr>
<td>IV</td>
<td></td>
<td>0.505</td>
<td>0.489</td>
<td>0.656</td>
<td>1.16x10^-5</td>
<td>0.967</td>
<td>2.69x10^-2</td>
</tr>
<tr>
<td>V</td>
<td></td>
<td>0.168</td>
<td>0.165</td>
<td>0.182</td>
<td>4.57x10^-5</td>
<td>27.3</td>
<td>3.91x10^-5</td>
</tr>
<tr>
<td>VI</td>
<td></td>
<td>81.7</td>
<td>20.0</td>
<td>7.58</td>
<td>8.47x10^-3</td>
<td>&gt;1000</td>
<td></td>
</tr>
<tr>
<td>VII</td>
<td></td>
<td>42.0</td>
<td>12.3</td>
<td>6.05</td>
<td>1.71x10^-2</td>
<td>30.2</td>
<td>0.169</td>
</tr>
<tr>
<td>VIII</td>
<td></td>
<td>0.078</td>
<td>0.079</td>
<td>0.080</td>
<td>3.23x10^-3</td>
<td>0.258</td>
<td>0.344</td>
</tr>
<tr>
<td>IX</td>
<td></td>
<td>0.192</td>
<td>0.197</td>
<td>0.292</td>
<td>8.07x10^-2</td>
<td>0.584</td>
<td>2.37x10^-3</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>0.119</td>
<td>0.128</td>
<td>0.130</td>
<td>1.38</td>
<td>1.12</td>
<td>5.27x10^-3</td>
</tr>
<tr>
<td>XI</td>
<td></td>
<td>1.03</td>
<td>0.687</td>
<td>0.753(TR1)</td>
<td>4.85x10^-5</td>
<td>11.5</td>
<td>0.967</td>
</tr>
</tbody>
</table>

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

2Gauss-Seidel
7. **THE DYNYSYS 2.0 EXECUTIVE PROGRAM**

We have tested several integration techniques for handling stiff ordinary differential equations. Of these, Gear's method appears to be the most effective. We have also tested various techniques for solving large systems of linear algebraic equations which arise in solving the implicit part of a stiff numerical integration method. The Bending-Hutchison approach was found to be particularly appropriate.

We would now like to implement these features into a dynamic simulation executive program. Either the equation-oriented or problem-oriented approaches can be used, but we believe the modular approach is especially suited to the engineer, since many engineering processes are modular in structure; i.e., they consist of many different pieces of equipment. It was decided to implement the new techniques into the DYNYSYS package. The resulting program is called DYNYSYS 2.0.

This chapter describes the new features of DYNYSYS 2.0 and how they affect the user. A listing of DYNYSYS 2.0 appears in Appendix I.
7.1 **Numerical Integration of O.D.E.s**

Both the stiff and nonstiff options of Gear's method are available; however, the same option must be used for the entire simulation.

A simulation will often contain both stiff and nonstiff modules. In this case the stiff option must be used and all modules will use the stiff coefficients of Gear's method. We saw in Chapter 6 that the stiff option handles nonstiff problems reasonably well. It is not possible to use the nonstiff and stiff coefficients in the same simulation as they have different error constants and the error and step-changing analysis could not be done.

The stiff option is assumed by the executive. If the user wishes to have the nonstiff system, i.e., the system is completely nonstiff, he must specify this through the data set. Section 2.6 describes how to identify a stiff system.

The algorithm used differs from Gear's original DIFSUB (Appendix A) because of the modular approach. A predictor step is taken through all of the modules followed by a corrector step. Up to three corrector iterations are made. The derivatives are re-evaluated by returning to the module. If convergence does not occur for any module, the integration step size is reduced to 1/4 of its present size and the step repeated for all modules. The factor 1/4
is given by Gear (1971c) and is empirical.

Another way of using the predictor-corrector would be to predict and correct for each module in turn. However, the manner actually chosen helps to converge any recycle in the system. As yet DYNSYS has no other way of handling recycle convergence. This is a possible item for future work.

After the corrector pass, the error analysis is done (Gear, 1971c; Appendix A) considering the equations from every module as one system.

7.2 Corrector Convergence of Stiff Equations

Gear's algorithm contains an option for both stiff and nonstiff equations. If the nonstiff option is used, the corrector equations will be solved by repeated iteration only. For the stiff option, Newton-Raphson iteration is generally used, but a special option for tridiagonal matrices (Carnahan, Luther and Wilkes, 1969) was included. This was done because tridiagonal systems arise often in chemical engineering in the discretization of partial differential equations and in countercurrent stagewise processes. A special algorithm for the tridiagonal system is much faster than using the Bending-Hutchison solver. See example #4 of Chapter 8.
In a system of nonstiff and stiff modules, all the modules will use the stiff coefficients of Gear's method to facilitate the error analysis, but the nonstiff modules can use Jacobi iteration of the corrector.

Each set of stiff p.d.e.s in each module will generate an operator list. All the lists are created on the first time step and are stored sequentially in core with 10% extra storage in case the list is re-created.

The routine contains a minimum pivot presently set at $10^{-6}$ which is reasonable for 14 figure single precision accuracy as on CDC machines. This figure may be altered through the data set for other computers (for example, perhaps $10^{-3}$ for single precision IBM computers). If, during the integration, one of the pivots falls below $10^{-6}$, the operator list will be re-calculated using different elements as pivots. It is assumed that the new operator list will not be more than 10% longer than the original one. Several tests were made where the operator list was re-calculated every time step. The greatest change in length was 5% longer than the original length; however, in every example tested, the original operator list was sufficient for the entire simulation. Different operator lists can be generated as the integration proceeds, in addition to the possibility of a pivot falling below $10^{-6}$, because at $\tau = 0$, many nonzero elements of the
Jacobian may actually be zero because of the initial conditions, their values may become large enough that they are available as pivots. The present executive does not, however, re-evaluate the operator list for this reason as it is unnecessary and would require too much computer time.

DIFSUB can use a numerical or analytical (user supplied) Jacobian matrix. It was decided to use an analytical Jacobian for DYNSYS as it is more efficient and it would be extremely difficult to compute a sparse numerical Jacobian and use it with the Bending-Hutchison routine.

Gear re-evaluated the Jacobian only if the corrector did not converge in three iterations or less or if a change in order was made. The present method in DYNSYS evaluates the Jacobian every time step to avoid storing all the Jacobians together in core at the same time since storage space is critical.

The DYNSYS version of DIFSUB has about the same execution speed as Gear's published version. The Jacobian is evaluated more often, but this is balanced by fewer corrector iterations being required.
7.3 Using the DYNSYS 2.0 Executive Program

7.3.1 New Parameters in the DYNSYS 2.0 Data Set

Several new data set parameters relating to the new integration method have been introduced. They are the smallest and largest permissible integration step size (HMIN, HMAX), the option, stiff or nonstiff to be used (NONSTIFF), the maximum permissible order of the integration (ORDER), the error tolerance (EPS), and the minimum permissible pivot element (MINPIVOT). These are summarized in Table 7.1. The other data set parameters are described in the DYNSYS Manual.

7.3.2 Writing Modules For DYNSYS 2.0

The user will usually write a module corresponding to each of his pieces of equipment. A general skeleton of such a module is shown in Figure 7.1. The user should study this and the examples in Chapter 8 before he writes his modules.

As shown in the skeleton, a module can be divided into five basic sections:

(1) calculate module parameters, stream input, initial conditions, ITER, etc.
Table 7.1: New Parameters In the DYNSYS 2.0 Data Set

<table>
<thead>
<tr>
<th>DATA INPUT</th>
<th>DEFAULT VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMIN</td>
<td>$HMIN=10^{-6}$</td>
<td>$HMIN$: smallest permissible step size to be used by DIFSUB</td>
</tr>
<tr>
<td>HMAX</td>
<td>$HMAX=0.05$</td>
<td>$HMAX$: largest permissible step size to be used by DIFSUB</td>
</tr>
</tbody>
</table>
| NONSTIFF   | ISTIFF=1      | ISTIFF=0: nonstiff option, i.e., nonstiff coefficients are used in Gear's method  
             |               | ISTIFF=1: stiff option |
| ORDER      | IORDER=6      | IORDER: maximum permissible order of integration method to be used  
             |               | IORDER cannot exceed 6 for stiff option or 7 for nonstiff option |
| TOLERANCE  | EPS=0.001     | EPS: Euclidean norm of relative local truncation errors |
| MINPIVOT   | XMIN=$10^{-6}$ | XMIN: minimum permissible value of pivot element in TRGB, TRGB2 |
FIGURE 7.11 SKELETON OF MODULE FOR DYNSYS 2.0

SUBROUTINE TYPE 2.0

COMMENTS DESCRIBING MODULE

THE FOLLOWING COMMON BLOCKS ARE OR MAY BE REQUIRED:
MAT, CON, PTAB, UNIT, GEAR2, MODULE, ROW, COLUMN, JACOB, SUBDI, DIAG, SUPERD

COMMON/MAT/MP(   ), EP(   ), S(   ), EX(   ) COMMON/CON/IG, NCOMP, NC5, H, NS, NPR, NPOL, Tmax, IORDER, NGRAPH
COMMON/PTAB/IGFLAG, PP(   ) COMMON/UNIT/IM, NMP
COMMON/GEAR2/EPS, TIME, KFLAG, JSTART, NBVMAX, ICONV, NOPPT, ISTITF
COMMON/MODULE/IDERY, ITER, ITRI, NZERO

NZERO IS NUMBER OF NONZERO ELEMENTS IN JACOBIAN

COMMON/ROW/IROW(NZERO) (NORMAL OPTION)
COMMON/COLUMN/JCOL(NZERO) (NORMAL OPTION)
COMMON/JACOB/XJACOB(NZERO) (NORMAL OPTION)

N IS NUMBER OF ODES

COMMON/SUBDI/A(N) (TRIDIAGONAL OPTION)
COMMON/DIAG/B(N) (TRIDIAGONAL OPTION)
COMMON/SUPERD/C(N) (TRIDIAGONAL OPTION)

******************************************************************************

SECTION #1 : PARAMETER CALCULATIONS
******************************************************************************

CALCULATE MODULE PARAMETERS : STREAM INPUT, INITIAL CONDITIONS,
VALUES OF ITER ETC.

INPUT STREAM INFORMATION IS OBTAINED FROM S(IG,   )
VALUE OF INDEPENDENT VARIABLE *(Y)* NEED ONLY BE SPECIFIED ON FIRST
INTEGRATION STEP (I.E. INITIAL CONDITIONS)
JSTART=0 ON FIRST INTEGRATION STEP
*CURRENT ORDER OF INTEGRATION TECHNIQUE ON LATER STEPS

IF ISTITF=0, NONSTIFF COEFFICIENTS WILL BE USED IN INTEGRATION ALGORITHM
(DIFSUB) FOR ALL MODULES,
JACOBIAN MATRIX IS NOT REQUIRED

IF ISTITF=1, STIFF COEFFICIENTS WILL BE USED FOR ALL MODULES
THEN, IF ITER=0 DIRECT ITERATION OF CORRECTOR WILL BE USED
(NONSTIFF MODULE)
JACOBIAN IS NOT REQUIRED
IF ITER=1 NEWTON-RAPHSON ITERATION OF CORRECTOR WILL
BE USED (STIFF MODULE)
JACOBIAN MATRIX MUST BE SUPPLIED
ITER MUST BE SPECIFIED 0 OR 1
IT IS NOT USED UNLESS ISTIFF=1

IF(IG.EQ.2) GO TO 2

SECTION #2 IS OMITTED FOR NONSTIFF MODULE

JACOBIAN MATRIX IS REQUIRED ONLY IF MODULE EQUATIONS ARE STIFF
CALCULATE JACOBIAN MATRIX ON CORRECTOR PASS ONLY
JACOBIAN NEED NOT BE VERY ACCURATE, AS IT IS USED ONLY FOR CONVERGENCE
OF CORRECTOR.
ONLY NONZERO ELEMENTS ARE CALCULATED, BUT ALL DIAGONAL ELEMENTS MUST
BE STORED WHETHER OR NOT THEY ARE ZERO (NORMAL OPTION)

NORMAL OPTION

IF JACOBIAN MATRIX IS NOT TRIDIAGONAL
ROW NUMBER, COLUMN NUMBER AND VALUE OF EACH NONZERO ELEMENT ARE STORED IN
ARRAYS IROW, JCOL AND XJACOB FROM COMMON BLOCKS ROW, COLUMN AND JACOB
RESPECTIVELY.
ELEMENTS MAY BE STORED IN ANY ORDER

NZERO = NUMBER OF NONZERO ELEMENTS IN JACOBIAN
INCLUDING ANY ZERO DIAGONAL ELEMENTS
IROW(I) = ROW NUMBER OF NONZERO ELEMENT I
JCOL(I) = COLUMN NUMBER OF NONZERO ELEMENT I
XJACOB(I) = VALUE OF NONZERO ELEMENT I

NZERO =
IROW(1) =
JCOL(1) =
XJACOB(1) =

TRIDIAGONAL OPTION

FOR TRIDIAGONAL JACOBIAN MATRIX, THE SUBDIAGONAL, DIAGONAL AND SUPERDIAGONAL
ELEMENT VALUES ARE STORED IN ARRAYS A, B AND C FROM COMMON BLOCKS SUBDI,
DIAG AND SUPERD RESPECTIVELY

N = NUMBER OF ODES
A - VALUES OF SUBDIAGONAL ELEMENTS ARE STORED IN A(2) . . . A(N)
    A(1) IS NOT USED
B - VALUES OF DIAGONAL ELEMENTS
C - VALUES OF SUPERDIAGONAL ELEMENTS ARE STORED IN C(1) . . . C(N-1)
    C(N) IS NOT USED

B(1)=
C(1)=
A(2)=
B(2)=
C(2)=
    .
    .
A(N-1)=
B(N-1)=
C(N-1)=
A(N)=
B(N)=
1 CONTINUE

IF TRIDIAGONAL OPTION IS BEING USED, ITRI MUST BE SET TO 1 HERE,
IF NORMAL OPTION IS USED ITRI MAY BE IGNORED
ITRI=1
2 CONTINUE

SECTION #3 : DERIVATIVE CALCULATION

CALCULATE DERivatives

DERY(1)=
    .
    .
DERY(N)=

SECTION #4 : CALL DIFSUB

CALL DIFSUB TO SOLVE ODES FOR MODULE
DIFSUB MAY BE CALLED MORE THAN ONCE FROM A MODULE FOR EXAMPLE IF THE
MODULE CONTAINS A SET OF STIFF O.D.E.S AND ANOTHER SET OF NONSTIFF
O.D.E.S

ARGUMENTS :  N - NUMBER OF ODES
             Y - INDEPENDENT VARIABLE
             DERY - DERIVATIVES
CALL DIFSUB(N,Y,DERY)

IF IDERY IS NOT ZERO, THE DERIVATIVES WILL BE RE-EVALUATED AND
RETURNED TO DIFSUB
ITRI MUST ALSO BE RESET IF IT IS 1

IF(IDERY NE 0) GO TO 1

---------------------------------------------------------------------------------------------------------

SECTION #5 : STREAM OUTPUT CALCULATION

---------------------------------------------------------------------------------------------------------

CALCULATE STREAM OUTPUT (STORED IN S(1, *, 1))

S(1, *) =
  
  
  
RETURN
END
(2) calculate Jacobian matrix (if o.d.e.s are stiff) on corrector pass only

(3) evaluate derivatives

(4) CALL DIFSUB to solve o.d.e.s for module

(5) calculate stream output.

There are several variables used in the skeleton module which relate to the integration. These are summarized in Table 7.2.

It is not necessary to supply a Jacobian matrix for the nonstiff modules. Setting the variable ITER to zero inside the module will cause direct iteration of the corrector. ITER should be set to one for stiff modules so that Newton-Raphson iteration will be used and the Jacobian of course must be supplied. If the module is known to be always stiff or nonstiff, ITER can be specified inside the module; otherwise, an equipment parameter can be used to specify the option.

It is possible to have more than one set of o.d.e.s in a module. This will present no problem as long as each set is defined before the call to DIFSUB.

It may be necessary to use a low value of HMAX, the maximum permissible integration step size, to keep the system under control. If too large a step is used, say
Table 7.2: Summary of Module Variables Relating To DIFSUB

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>values of subdiagonal elements of a tridiagonal Jacobian matrix are stored in A(2), ..., A(N)</td>
</tr>
<tr>
<td>B</td>
<td>values of diagonal elements of a tridiagonal Jacobian matrix are stored in B(1), ..., B(N)</td>
</tr>
<tr>
<td>C</td>
<td>values of superdiagonal elements of a tridiagonal Jacobian matrix are stored in C(1), ..., C(N-1)</td>
</tr>
<tr>
<td>DER1</td>
<td>derivative values</td>
</tr>
<tr>
<td>IDER1</td>
<td>IDER1 is tested immediately after call to DIFSUB</td>
</tr>
<tr>
<td></td>
<td>IDER1=0 : module execution is continued</td>
</tr>
<tr>
<td></td>
<td>IDER1=1 : derivatives are re-evaluated and returned to DIFSUB, ITR1 is reset to 1 if necessary</td>
</tr>
<tr>
<td>IRON</td>
<td>row numbers of nonzero Jacobian elements and zero diagonal elements in normal option for storing Jacobian</td>
</tr>
<tr>
<td>ITER</td>
<td>ITER is used only if ISTITF=1 (stiff option)</td>
</tr>
<tr>
<td></td>
<td>ITER=0 : direct iteration of corrector (ngnstiff module)</td>
</tr>
<tr>
<td></td>
<td>ITER=1 : Newton-Raphson iteration of corrector (stiff) module</td>
</tr>
<tr>
<td>ITR1</td>
<td>ITR1 specifies option for storing Jacobian</td>
</tr>
<tr>
<td></td>
<td>ITR1=0 : normal option</td>
</tr>
<tr>
<td></td>
<td>ITR1=1 : tridiagonal option</td>
</tr>
<tr>
<td></td>
<td>if ITR1=1, it must be reset to 1 each time DIFSUB is called</td>
</tr>
<tr>
<td>JCOL</td>
<td>column numbers of nonzero Jacobian elements and zero diagonal elements in normal option for storing Jacobian</td>
</tr>
<tr>
<td>N</td>
<td>number of ordinary differential equations in module</td>
</tr>
<tr>
<td>NYERO</td>
<td>number of nonzero elements in Jacobian for normal option (zero diagonal elements are included)</td>
</tr>
<tr>
<td>XJACOB</td>
<td>values of nonzero Jacobian elements in normal option</td>
</tr>
<tr>
<td>Y</td>
<td>independent variable</td>
</tr>
</tbody>
</table>
near steady state, the truncation error will be acceptable, but the corrector may not converge and thus the step size will be reduced by a factor of four. This may happen repeatedly if HMAX is too large, for both stiff and nonstiff systems.

Problems may also occur if the variables being simulated have widely varying values. If an important variable has a value of the order $10^{-6}$, then EPS, the error tolerance, must be set to about $10^{-9}$. This can use a great deal of computer time for the other variables. It may be necessary to scale the mathematical model so that there are no important variables with extremely small value.

DYNSYS 2.0 was generally tested with optimization level one of the CYBER 73 FTN compiler. Using the RUN compiler (a nonoptimizing compiler), erroneous results were obtained on some examples probably because of excessive round-off from the unoptimized code. Thus the user should ensure that an optimizing compiler should be used. Both DIFSUB and TRGB-TRGB2 should have at least twelve figure accuracy in the computer word length. Excessive round-off may affect the results, if a smaller word size is used.
7.3.3 Jacobian Evaluation

If the module o.d.e.s are stiff (Section 2.6), the Jacobian matrix must be supplied in the module. There are two ways of accomplishing this: the normal option for a general Jacobian matrix and a special option for a tridiagonal Jacobian. The Jacobian does not have to be extremely accurate as it is needed only for the convergence of the corrector; for example, if the Jacobian is diagonally dominant, it may suffice to use only the diagonal coefficients.

7.3.3.1 Normal Option

The row number, column number and value of each nonzero element are stored in arrays IROW, JCOL and XJACOB respectively. The elements may be stored in any order. For example the matrix:

\[
J = \begin{bmatrix}
1.0 & 2.0 \\
3.0 & 4.0 \\
\end{bmatrix}
\]  

(7.3.1)

would be stored as:
NZERO = 4
IROW(1) = 1
JCOL(1) = 1
XJACOB(1) = 1.0
IROW(2) = 1
JCOL(2) = 2
XJACOB(2) = 2.0
IROW(3) = 2
JCOL(3) = 1
XJACOB(3) = 3.0
IROW(4) = 2
JCOL(4) = 2
XJACOB(4) = 4.0

NZERO is the number of nonzero elements in the matrix. Note also that a Jacobian element must be created for all diagonal elements whether or not they are nonzero.

Figure 8.7 depicts a simple example of a module using the normal option.

7.3.3.2 Tridiagonal Option

The subdiagonal, diagonal and superdiagonal element values are stored in arrays A, B and C respectively. A(1) and C(N) are not used. For example:
\[
J = \begin{bmatrix}
1.0 & 2.0 \\
3.0 & 4.0 & 5.0 \\
6.0 & 7.0 & 8.0 \\
9.0 & 10.0 & 11.0 \\
12.0 & 13.0
\end{bmatrix}
\] (7.3.2)

is stored as

\begin{align*}
B(1) &= 1.0 & C(1) &= 2.0 \\
A(2) &= 3.0 & B(2) &= 4.0 & C(2) &= 5.0 \\
A(3) &= 6.0 & B(3) &= 7.0 & C(3) &= 6.0 \\
A(4) &= 9.0 & B(4) &= 10.0 & C(4) &= 11.0 \\
A(5) &= 12.0 & B(5) &= 13.0
\end{align*}

The variable \texttt{ITRI} must be set equal to 1 each time \texttt{DIFSUB} is called.

Figure 8.14 depicts a simple example of a module using the tridiagonal option.

### 7.3.4 Error Messages

\texttt{DYNSYS 2.0} prints several error messages related to the numerical integration. The program continues for messages 1-3, but execution stops for messages 4-8.
(1) TRUNCATION ERROR IS TOO LARGE

This message may occur in critical regions of the simulation where some variables may be changing rapidly. The step size is too large and will be adjusted automatically by the executive.

(2) THE CORRECTOR FAILED TO CONVERGE IN 3 ITERATIONS

DIFFERENTIAL EQUATIONS ___ TO ___

This message will be printed when the nonstiff option is used for a stiff system, or if the stiff option is used with direct iteration of the corrector for a stiff system. The stiff option should be used with Newton-Raphson iteration in either case. It may also occur if the problem is formulated incorrectly; i.e., there is an error in the differential equations or in the Jacobian, or the Jacobian is not accurate enough. The differential equations causing the problem are given. They are numbered in order of their occurrence in the simulation.

(3) THE CORRECTOR CANNOT BE SOLVED BECAUSE PW(__) IS A PIVOT ELEMENT WHOSE VALUE IS BELOW XMIN DIFFERENTIAL EQUATIONS ___ TO ___ OPERATOR LIST WILL BE RE-CREATED

This message can appear in using the stiff option with Newton-Raphson iteration, i.e., the Bending-Hutchison linear equation solver. If it occurs more than once or
or twice, it may help to decrease the variable XMIN through the data set. The formulation of the differential equations and Jacobian should be checked.

(4) THE MAXIMUM ORDER SPECIFIED IS OUT OF RANGE

For the nonstiff option, the maximum order must be in the range 1-7. The range is 1-6 for the stiff option. The default value is 6 for both options, but the user may specify a value through the data set.

(5) CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED FOR H.GT.HMIN DIFFERENTIAL EQUATIONS ___ TO ___

Message (2) will precede this one and the cause is the same, but this time the program will stop.

(6) THE STEP WAS TAKEN WITH H=HMIN BUT THE REQUESTED ERROR WAS NOT ACHIEVED

The user should try a larger value for the error tolerance or a smaller value of HMIN.

(7) THE REQUESTED ERROR IS SMALLER THAN CAN BE HANDLED FOR THIS PROBLEM

EPS has been specified equal to zero. As in (6), the user should try a larger tolerance.
(8) ERROR IN TRGB

This means that the Jacobian matrix has not been properly defined; i.e., there is a redundant or inconsistent equation. It may also occur if the arrays are dimensioned too small.
8. SIMULATION EXAMPLES

Four relatively simple examples were simulated with the new DYNNSYS 2.0 executive:

1. The level control of a simple stirred tank (nonstiff) with time delay.

2. A network of 15 stirred tank reactors, 2 o.d.e.s per reactor, some stiff and some nonstiff.

3. A tubular reactor with 222 stiff equations resulting from the discretization of partial differential equations.

4. A tubular reactor with 49 stiff o.d.e.s with tridiagonal Jacobian matrix.

The examples are not intended to be rigorously realistic, but they do demonstrate the types of problems which may be handled. A more complex simulation is described in Chapter 9. No system of units is employed since the examples are fictitious. The reader may assume any set of consistent units. Each example contains a listing of the modules and data set. A description of a data set is in Appendix F.
8.1 Level Control Example

This example is based on one in the DYNYSYS Manual (Bobrow et al., 1970). Figure 8.1 is the process flow diagram for a simple mixing tank equipped with a level controller.

Liquid flows into a stirred tank at a fluctuating rate. The tank outlet stream is equipped with a valve which is manipulated by a level controller connected to the tank.

The response of the system to a simultaneous step change in inlet flow and temperature is to be simulated. The controller will attempt to keep the tank level at a certain set point by manipulating the flow rate through the outlet valve. Available in the DYNYSYS library of modules are mathematical models for:

1. a perfectly mixed tank (STIR1)

2. a valve with a parabolic characteristic (VALV1)

3. a proportional-integral controller (CONT1)

4. a time delay (DELAY).

Figure 8.2 is the dynamic information flow diagram for the system. There is a time delay between the tank and the controller and another between the controller and
FIGURE 8.1: PROCESS FLOW DIAGRAM - LEVEL CONTROL EXAMPLE

FIGURE 8.2: DYNAMIC INFORMATION FLOW DIAGRAM - LEVEL CONTROL EXAMPLE
the valve but these can be represented together by one module.

STIR1 is the only differential module, i.e., the only one containing differential equations.

The unsteady state mass balance for each component $i$ can be written:

$$\frac{dM_i}{dt} = F_{i,in} - F_{i,out} \tag{8.1.1}$$

where $M_i$ = mass of component $i$ in the tank
$F_{i,in}$ = inlet mass flow of component $i$
$F_{i,out}$ = outlet mass flow of component $i$

in terms of the variables of DYNSYS

$$\frac{dM_i}{dt} = F_{in} \cdot x_{i,in} - F_{out} \cdot x_{i,out} \tag{8.1.2}$$

where $x_i$ = component $i$ mass fraction
$F$ = total mass flow.

This equation can be integrated to give the mass holdup of the component, $M_i$.

Summation over all components $i$ of $M_i$

$$M = \sum_{i=1}^{NCOMP} M_i \quad (NCOMP = \text{number of components}) \tag{8.1.3}$$
gives the total mass holdup in the tank. The mass fractions of each component in the vessel, and hence in the outlet stream, may be obtained by division:

\[ z_t = \frac{\frac{dW}{dt}}{M} \quad (8.1.4) \]

Similarly, the heat balance for the vessel:

\[ \frac{dH}{dt} = H_{in} - H_{out} \quad (8.1.5) \]

becomes:

\[ \frac{dH}{dt} = P_{in} \times T_{in} \times CP_{in} - P_{out} \times T_{out} \times CP_{out} \quad (8.1.6) \]

and \( H \) is obtained by integration.

Assuming constant heat capacity, the new vessel temperature is:

\[ T = \frac{H}{M \times CP} \quad (8.1.7) \]

The modules CQNTL and VALV are described in the DYN SYS Manual (Bobrow et al., 1970).

The time delay module, DELAY, uses the "bucket brigade" approach to simulate a delayed variable. The past times and stream information are stored in vectors
and the stream output from the delay is interpolated from these values. The module is described more fully in Appendix G.

A listing of the modules (except DELAY) and the data set is shown in Figure 8.4. A listing of DELAY appears in Appendix G. A graph of the tank level versus time is depicted in Figure 8.3. After a little more than two time units, the tank level returns to its set point of 1000.

The old version of DYNSYS required 20.8 seconds of execution time to simulate this example, while DYNSYS 2.0 took 5.9 seconds. In both runs, HMAX, the maximum permissible step size was limited to 0.01. However, with DYNSYS 2.0, HMAX had to be limited to 0.01 to get accurate results, while the AMOS version could use a maximum step of 0.95 and still be accurate. Possibly the Gear algorithm does not handle forcing functions as well as the Adams-Moulton-Shell routine. The Gear algorithm might function better if the order was limited to one or two when forcing functions are present.
Figure 8.3: Graph of tank level versus time for example #1
SURROUNTE TYPE 1

SURROUNTE VALVE 1

V-PORT (PARABOLIC) CONTROL VALVE

EQUIPMENT PARAMETERS

1 - NOT USED
2 - VALVE CONSTANT
3 - ACTION (DIRECT, REVERSE)

COMMON /UNIT/ IM*NMP
COMMON /MAT/ MP(4,5), EP(4,5), S(2,5,7)*EX(1)
COMMON /CON/ IG, NCOMP, NS, NPR, NPOL, TMAX, IORDER, NGRAPH

MI=MP(4,3)
MO=IABS(MP(4,4))

CHECK THAT THE VALVE SIGNAL IS IN THE RANGE 0-100
V=S(1,MI/3)
IF (V.LT.0.) V=0.
IF (V.GT.100.) V=100.

ACTION
A=EP(IM,3)
IF (A.LT.0.) S(1,MO/3)=EP(IM,2)*(100.-V)**2
IF (A.GE.0.) S(1,MO/3)=EP(IM,2)*V**2
RETURN
END
SUBROUTINE TYPE4

SUBROUTINE CONT1

PI OR ON/OFF CONTROLLER
SECOND OPTIONS ARE FOR ON/OFF CONTROLLER
IF EQUIPMENT PARAMETERS 4 AND 5 ARE LT.0, CONTROLLER IS ON/OFF

EQUIPMENT PARAMETERS

1 - CONTROLLED VARIABLE
2 - RANGE OR UPPER LIMIT
3 - SET POINT OR LOWER LIMIT
4 - PROPORTIONAL GAIN OR LT. 0
5 - INTEGRAL TIME OR LT. 0

IF, LT. - 10.0, OFF WHEN, GT. SETP

COMMON /UNIT/ IM*NMP
COMMON /MAT/ MP(4,5)*EP(4,5)*S(2,5,7)*EX(1)

P1=EP(IM,4)
P2=EP(IM,5)
IN=MP(IM,3)
IOUT=ABS(MP(IM,4))
K=EP(IM,1)
IF (P1,LT,0.0,AND.,P2,LE,0.0) GO TO 1
MEASURED VARIABLE AT CURRENT TIME
SIG1=S(1,IN,K)
MEASURED VARIABLE AT LAST TIME
SIG2=S(2,IN,K)
SCALE THE ERROR
SCALE=50.0/EP(IM,2)
PRESENT ERROR
ERR=(SIG1=EP(IM,3))*SCALE
LAST ERROR
OLDER=(SIG2=EP(IM,3))*SCALE

OUTPUT SIGNAL
S(1,IOUT,3)=P1*(ERR-OLDER)+P2*(ERR+OLDER)*0.5+H)*S(2,IOUT,3)
IF (S(1,IOUT,3),GT,100.0) S(1,IOUT,3)=100.0
IF (S(1,IOUT,3),LT,0.0) S(1,IOUT,3)=0.0
RETURN

1 CONTINUE
ON/OFF CONTROLLER
IF (IG*,EQ,1) RETURN
T=S(1,IN,K)
IF (P2,LT,-10.0) GO TO 2
IF (T,GE,EP(IM,2)) T=50.0
IF (T,LT,EP(IM,3)) T=0.0
GO TO 3
CONTINUE
IF (T,LE,EP(IM,2)) T=50.0
IF (T,GTE,EP(IM,2)) T=0.0

T04 1
T04 2
T04 3
T04 4
T04 5
T04 6
T04 7
T04 8
T04 9
T04 10
T04 11
T04 12
T04 13
T04 14
T04 15
T04 16
T04 17
T04 18
T04 19
T04 20
T04 21
T04 22
T04 23
T04 24
T04 25
T04 26
T04 27
T04 28
T04 29
T04 30
T04 31
T04 32
T04 33
T04 34
T04 35
T04 36
T04 37
T04 38
T04 39
T04 40
T04 41
T04 42
T04 43
T04 44
T04 45
T04 46
T04 47
T04 48
3 CONTINUE
S(1+IOUT+3)=T
RETURN
END
SURROUNTE TYPE 3

STIR01 - TYPE 3

GENERAL WELL MIXED, MULTIPLE INPUT/OUTPUT VESSEL
PRESSURE EFFECTS ARE CONSIDERED, ASSUMING IDEAL GAS
BEHAVIOUR IF A VOLUME IS SPECIFIED.
PARAMETERS:
-1 INITIAL HOLDUP
-2 VOLUME
-5 1ST. OUTPUT STREAM, USED INTERNALLY
3 AND 4 ARE NOT SPECIFIED BY USER.

COMMON /UNIT/ IM,NMP
COMMON /MAT/ MP(4.5),EP(4.5),S(2.5.7),EX(1)
COMMON /CON/ IC,NCMP,NC5,H,NE,NS,NPR,NPOL,TMAX,ORDER,NGRAPH
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NB,MAX,ICONV,NOPPT,ISTIFF
COMMON /PTAB/ IG,FLAG,PP(10.10)
COMMON /MODULE/ IDER,TET,ITRIA,INZ
DIMENSION Y(10), CMPT(10)
REAL M

USE DIRECT ITERATION IF STIFF OPTION IS USED (ITER=0)

ITER=0
NC=NCMP+1
VOL=EP(IM,2)
IF (JSTART,NE.0.,OR.,IG,EQ.1) GO TO 4

FIND THE FIRST OUTPUT STREAM
DO 1 J=1,NMP
JS=ABS(MP(IM,J))
IF (ABS(S(1,IS,2),GT.10.0)) GO TO 1
IF (MP(IM,J),LT.0.0) GO TO 2
1 CONTINUE

EP(IM,5)=J
NOUT=EP(IM,5)
IOUT=MP(IM,NOUT)

CALCULATE INITIAL HOLDUP IF NOT SPECIFIED.
IF (VOL,GT.0.0,AND.,EP(IM,1),LE.0.0) EP(IM,1)=VOL*SG(1,IS)
IF (ABS(S(1,IS,2),GT.10.0)) WRITE (6,15) MP(IM,1)

CALCULATE INITIAL CONDITIONS FOR VARIABLES TO BE INTEGRATED,
I.E., HEAT AND MASSES.

HEAT
Y(1)=EP(IM,1)*S(IG,IOUT,4)*CP(IG,IOUT)

MASSES
DO 3 K=1,NC5
K=K+4
Y(KK)=EP(IM,1)*S(IG,IOUT,K)
3 CONTINUE

CONTINUE
NOUT=EP(IM,5)
IOUT=MP(IM,NOUT)
C CALCULATE DERIVATIVES
5 CONTINUE
DO 6 J=1,NC1
CMPT(J)=0
6 CONTINUE
DO 9 J=3,NMP
IS=ABS(MP(IM+J))
IF (IS.EQ.0) GO TO 9
C DO NOT INCLUDE CONTROLLER SIGNALS IN MASS AND HEAT BALANCES
IF (ABS(S(1,IS+2)).GT.10.) GO TO 9
SIGN=IS/MP(IM+J)
C ALLOW HEAT INPUT STREAM WITH FLAG = 9
C S(1,3) CONTAINS HEAT RATE IN BTU/HR.
IF (ABS(S(1,IS+2)).LE.9999) GO TO 7
CMPT(1)*=CMPT(1)*S(1,IS+3)
GO TO 9
7 CONTINUE
C HEAT
CMPT(1)*=CMPT(1)*SIGN*S(IG+IS+4)*CP(IG+IS)*S(IG+IS+3)
C MASS BALANCES
DO 8 K=2,NC1
KK=K+4
CMPT(K)*=CMPT(K)*SIGN*S(IG+IS+KK)*S(IG+IS,3)
8 CONTINUE
9 CONTINUE
C PREDICT/CORRECT NEW VALUES
CALL DIFSUB (NC1,Y,CMPT)
C NORMALIZE MASS FRACTIONS
SUM=0.0
DO 10 I=2,NC1
SUM=SUM+Y(I)
10 CONTINUE
DO 11 K=6,NC5
S(1,IOUT+K)=Y(K-4)/SUM
11 CONTINUE
C THE NEW TEMPERATURE
S(1,IOUT+4)=Y(1)/(SUM*CP(1,IOUT))
IF (IDERY.NE.0) GO TO 5
C INSERT THE NEW VALUES IN THE OUTPUT STREAMS
DO 13 J=3,NMP
IS=MP(IM+J)
IF (IS.LE.0) GO TO 13
DO 12 K=4,NC5
S(1,IS+K)=S(1,IOUT+K)
12 CONTINUE
C FOR CONTROLLER OUTPUT, HOLD-UP STORED IN PLACE OF FLOW
IF (ABS(S(1,IS+2)).GT.10.) S(1,IS+3)=SUM
13 CONTINUE
IF (VOL.LE.0.0) RETURN
C PRESSURE
PRES=SUM*10.71*S(1,IOUT+4)/(VOL*MW(1,IOUT))
DO 14 I=3,NMP
J=IABS(MP(IM,I))
IF (J.EQ.0) GO TO 14
S(1,J,5)=PRES
14 CONTINUE
RETURN
C 15 FORMAT (1H0,14H ERROR EQ, NO.,13,22H ONLY OUTPUT IS SIGNAL)
END
**TEST OF SIMPLE LEVEL CONTROL SYSTEM WITH STEP CHANGE IN INLET FLOW RATE AND TEMPERATURE, TIME DELAY INCLUDED.**

<table>
<thead>
<tr>
<th>BEGIN</th>
<th>IN/OUT</th>
<th>3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOLERANCE</td>
<td>0.001</td>
<td></td>
</tr>
<tr>
<td>HMAX</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>LIBRARY</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>DELAY</td>
<td>9.0</td>
<td></td>
</tr>
<tr>
<td>PROCESS</td>
<td>CONT1</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1500.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td>DELAY</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-5.0</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-2.0</td>
</tr>
<tr>
<td></td>
<td>1000.</td>
<td></td>
</tr>
<tr>
<td>STIR1</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>VALV1</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-2.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>STREAMS</td>
<td>5.0</td>
</tr>
<tr>
<td>EXPPLICIT</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>22.36068</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>22.36068</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>PROPERTIES</td>
<td>-1.0</td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.2 Stirred Tank Reactor Network

Figures 8.5 and 8.6 represent the process flow diagram and the dynamic information flow diagram respectively for a network of CSTRs. The first order reversible reaction:

\[ \frac{k_1}{k_2} \]

occurs in each reactor.

The differential equations for the module are the mass balances for components \( A \) and \( B \):

\[
\frac{dy_1}{dt} = -(k_1 + \frac{1}{\tau})y_1 + k_2 y_2 + y_{10}/\tau
\]

\[
\frac{dy_2}{dt} = k_1 y_1 - (k_2 + \frac{1}{\tau})y_2 + y_{20}/\tau
\]

(8.2.1)

where \( y_1, y_2 \) = mass fraction components \( A \) and \( B \) respectively
\( \tau \) = reactor time constant (volume/volumetric flow rate)
\( k_1, k_2 \) = reaction rate constants
\( y_{10}, y_{20} \) = mass fraction of components \( A \) and \( B \) respectively in inlet stream
FIGURE 8.5: PROCESS FLOW DIAGRAM - REACTOR NETWORK EXAMPLE

FIGURE 8.6: DYNAMIC INFORMATION FLOW DIAGRAM - REACTOR NETWORK EXAMPLE
The Jacobian matrix of the system is:

\[
J = \begin{bmatrix}
-(k_1 + \frac{1}{\tau}) & + k_2 \\
+k_1 & -(k_2 + \frac{1}{\tau})
\end{bmatrix}
\]  \hspace{1cm} (8.2.2)

The eigenvalues are \(-\frac{1}{\tau}\) and \(-k_1 - k_2 - \frac{1}{\tau}\).

By choosing values of \(k_1, k_2\) and \(\tau\) we may create different stiffness ratios in each of the reactors. In practice the \(\kappa\)'s may vary because of different temperatures in each reactor. Table 8.1 shows the values which were used for the simulation. This creates a mixture of non-stiff and stiff modules. It was found experimentally that if the stiffness ratio is greater than approximately 3.5:1, Newton-Raphson iteration of the corrector should be used, but that below 3.5:1 the execution time is faster for direct iteration.

DYNYSYS 2.0 required 3.92 seconds of execution time. A listing of the module and its data set are given in Figure 8.7. A graph of the outlet concentration of \(A\) versus time for each reactor is shown in Figure 8.8. The curves are ordered from top to bottom in order of increasing module number. As the module number increases, so does the stiffness ratio as seen in Table 8.1. It can be seen in Figure 8.8 that the higher the stiffness ratio, the faster the reactor reaches steady state.
Table 8.1: Simulation Parameters For Reactor Network Example

<table>
<thead>
<tr>
<th>Module</th>
<th>Tau</th>
<th>K1/K2</th>
<th>K1</th>
<th>K2</th>
<th>Eig1</th>
<th>Eig2</th>
<th>Stiffness Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.1</td>
<td>0.26190</td>
<td>0.23810</td>
<td>-1.0</td>
<td>-1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.2</td>
<td>0.54545</td>
<td>0.45455</td>
<td>-1.0</td>
<td>-2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>1.3</td>
<td>0.84783</td>
<td>0.65217</td>
<td>-1.0</td>
<td>-2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>1.4</td>
<td>1.16667</td>
<td>0.83333</td>
<td>-1.0</td>
<td>-3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>1.5</td>
<td>5.40000</td>
<td>3.60000</td>
<td>-1.0</td>
<td>-10.0</td>
<td>10.0</td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
<td>1.6</td>
<td>30.15385</td>
<td>18.84615</td>
<td>-1.0</td>
<td>-50.0</td>
<td>50.0</td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
<td>1.7</td>
<td>62.33333</td>
<td>36.66667</td>
<td>-1.0</td>
<td>-100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>1.8</td>
<td>127.92857</td>
<td>71.07143</td>
<td>-1.0</td>
<td>-200.0</td>
<td>200.0</td>
</tr>
<tr>
<td>9</td>
<td>1.0</td>
<td>1.9</td>
<td>261.41379</td>
<td>137.56621</td>
<td>-1.0</td>
<td>-400.0</td>
<td>400.0</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>2.0</td>
<td>399.33333</td>
<td>199.66667</td>
<td>-1.0</td>
<td>-600.0</td>
<td>600.0</td>
</tr>
<tr>
<td>11</td>
<td>1.0</td>
<td>2.1</td>
<td>541.25806</td>
<td>257.74194</td>
<td>-1.0</td>
<td>-800.0</td>
<td>800.0</td>
</tr>
<tr>
<td>12</td>
<td>1.0</td>
<td>2.2</td>
<td>686.81250</td>
<td>312.18750</td>
<td>-1.0</td>
<td>-1000.0</td>
<td>1000.0</td>
</tr>
<tr>
<td>13</td>
<td>1.0</td>
<td>2.3</td>
<td>835.66667</td>
<td>363.33333</td>
<td>-1.0</td>
<td>-1200.0</td>
<td>1200.0</td>
</tr>
<tr>
<td>14</td>
<td>1.0</td>
<td>2.4</td>
<td>1058.11765</td>
<td>440.88235</td>
<td>-1.0</td>
<td>-1500.0</td>
<td>1500.0</td>
</tr>
<tr>
<td>15</td>
<td>1.0</td>
<td>2.5</td>
<td>1427.85714</td>
<td>571.14286</td>
<td>-1.0</td>
<td>-2000.0</td>
<td>2000.0</td>
</tr>
</tbody>
</table>
FIGURE 8.71 LISTING OF MODULE AND DATA SET FOR EXAMPLE #2
CSTR WITH FIRST ORDER REVERSIBLE REACTION (15 IN SERIES)

SUBROUTINE TYPE10
SUBROUTINE REAC1

THIS MODULE REPRESENTS A CSTR WITH A FIRST ORDER REVERSIBLE
REACTION, 1 INPUT STREAM, 1 OUTPUT STREAM, CONSTANT TEMPERATURE.

EQUIPMENT PARAMETERS

1 = VOL - REACTOR VOLUME - FT**3
2 = K1 - FORWARD REACTION RATE CONSTANT - MIN**-1
3 = K2 - BACKWARD REACTION RATE CONSTANT - MIN**-1
4 = ITER - ITER*10 USE DIRECT ITERATION WITH STIFF OPTION
   ITER=1 USE NEWTON-RAPHSON ITERATION WITH STIFF OPTION
   FOR NON-STIFF OPTION ITER IS NOT USED

COMMON /MAT/ MP(15,5),EP(15,5),S(2,16,7),EX(1)
COMMON /CON/ IG,NCOMP,NCS,NHE,NS,NPR,NPOL,TMAX,IORDER,NGRAPH
COMMON /PTAB/ IFLAG,PP(10,10)
COMMON /UNIT/ IM
COMMON /ROW/ IRV(4)
COMMON /COLUMN/ JCOL(4)
COMMON /JACOB/ XJACOB(4)
COMMON /MODULE/ IDERY,ITER,ITRI,NZERO
DIMENSION Y(2), DERY(2)
REAL K1,K2

CALCULATE MODULE PARAMETERS

GET VOLUME OF REACTOR FROM EP - VOL
VOL=EP(IM,1)

GET REACTION RATE CONSTANTS FROM EP - K1,K2
K1=EP(IM,2)
K2=EP(IM,3)

GET ITERATION OPTION FROM EP
ITER=EP(IM,4)

GET STREAM NUMBER OF INPUT STREAM FROM MP - IN
IN=MP(IM,3)

GET STREAM NUMBER OF OUTPUT STREAM FROM MP - IOUT
IOUT=IABS(MP(IM,4))

GET DEN SITY OF INPUT STREAM FROM PP - DENS
DENS=PP(1,2)

CALCULATE REACTOR TIME CONSTANT - TAU
TAU=VOL/IS(IG,IN,3)/DENS

GET INITIAL REACTOR CONCENTRATIONS FROM OUTPUT STREAM
Y(1)=S(IG,IOUT,6)
Y(2)=S(IG,IOUT,7)
CALCULATE JACOBIAN MATRIX ON CORRECTOR PASS

IF (IG.EQ.2) GO TO 1

NZERO IS NUMBER OF NON-ZERO ELEMENTS IN JACOBIAN
NZERO=4
IROW(1)=1
JCOL(1)=1
XJACOB(1)=(K1+1.0/TAU)
IROW(2)=2
JCOL(2)=1
XJACOB(2)=K1
IROW(3)=1
JCOL(3)=2
XJACOB(3)=K2
IROW(4)=2
JCOL(4)=2
XJACOB(4)=(K2+1.0/TAU)
CONTINUE

CALCULATE DERIVATIVES

MASS BALANCE FOR COMPONENT A (UNITS=MIN**2=1)
RATE OF CHANGE OF MASS FRACTION OF COMPONENT A
  = RATE OF INPUT OF A
  - RATE OF OUTPUT OF A
  + RATE AT WHICH B REACTS INTO A
  - RATE AT WHICH A REACTS INTO B
DERY(1)=(K1+1.0/TAU)*Y(1)+K2*Y(2)+S(IG,IN,6)/TAU

MASS BALANCE FOR COMPONENT B (UNITS=MIN**2=1)
RATE OF CHANGE OF MASS FRACTION OF COMPONENT B
  = RATE OF INPUT OF B
  - RATE OF OUTPUT OF B
  + RATE AT WHICH A REACTS INTO B
  - RATE AT WHICH B REACTS INTO A
DERY(2)=K1*Y(1)-(K2+1.0/TAU)*Y(2)+S(IG,IN,7)/TAU

CALL DIFSUB TO SOLVE ODES FOR MODULE

CALL DIFSUB (2,Y,DERY)
IF (IDERY.NE.0) GO TO 1

CALCULATE STREAM OUTPUT

NORMALIZE CONCENTRATIONS OF A AND B
SUM=Y(1)+Y(2)
Y(1)=Y(1)/SUM
Y(2)=Y(2)/SUM

PUT MASS FRACTIONS INTO OUTPUT STREAMS
S(1,IOUT+6)=Y(1)
S(1,IOUT+7)=Y(2)
RETURN
END
### Simulation of 15 CSTRs in Series

<table>
<thead>
<tr>
<th>Process</th>
<th>Reactor</th>
<th>Time (s)</th>
<th>Reaction Rate (mol/s)</th>
<th>Reaction Volume (L)</th>
<th>Temperature (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAC1</td>
<td>1.0</td>
<td>1.0</td>
<td>-2.0</td>
<td>10.0</td>
<td>-2.0</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>1.0</td>
<td>0.26190</td>
<td>0.23810</td>
<td>0.23810</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>1.0</td>
<td>10.0</td>
<td>0.54545</td>
<td>0.54545</td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td>1.0</td>
<td>0.84783</td>
<td>0.65217</td>
<td>0.65217</td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td>1.0</td>
<td>4.0</td>
<td>5.4</td>
<td>5.4</td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td>1.0</td>
<td>10.0</td>
<td>1.16667</td>
<td>1.16667</td>
</tr>
<tr>
<td></td>
<td>5.0</td>
<td>1.0</td>
<td>10.0</td>
<td>0.83333</td>
<td>0.83333</td>
</tr>
<tr>
<td></td>
<td>5.0</td>
<td>1.0</td>
<td>10.0</td>
<td>3.6</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>6.0</td>
<td>1.0</td>
<td>10.0</td>
<td>18.846</td>
<td>18.846</td>
</tr>
<tr>
<td></td>
<td>7.0</td>
<td>1.0</td>
<td>10.0</td>
<td>30.154</td>
<td>30.154</td>
</tr>
<tr>
<td></td>
<td>7.0</td>
<td>1.0</td>
<td>10.0</td>
<td>18.846</td>
<td>18.846</td>
</tr>
<tr>
<td></td>
<td>8.0</td>
<td>1.0</td>
<td>10.0</td>
<td>30.154</td>
<td>30.154</td>
</tr>
<tr>
<td></td>
<td>9.0</td>
<td>1.0</td>
<td>10.0</td>
<td>18.846</td>
<td>18.846</td>
</tr>
<tr>
<td></td>
<td>10.0</td>
<td>1.0</td>
<td>10.0</td>
<td>18.846</td>
<td>18.846</td>
</tr>
<tr>
<td></td>
<td>10.0</td>
<td>2.0</td>
<td>-12.0</td>
<td>199.67</td>
<td>199.67</td>
</tr>
<tr>
<td></td>
<td>10.0</td>
<td>3.0</td>
<td>10.0</td>
<td>257.74</td>
<td>257.74</td>
</tr>
<tr>
<td></td>
<td>11.0</td>
<td>10.0</td>
<td>-14.0</td>
<td>312.19</td>
<td>312.19</td>
</tr>
<tr>
<td></td>
<td>12.0</td>
<td>10.0</td>
<td>10.0</td>
<td>363.33</td>
<td>363.33</td>
</tr>
<tr>
<td>REACT</td>
<td>10.0</td>
<td>1058.12</td>
<td>440.88</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>---------</td>
<td>--------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STREAMS</td>
<td>16.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPlicit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
<td>624.0</td>
<td>60.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td>65.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>2.0</td>
<td>1.0</td>
<td>624.0</td>
<td>70.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td>75.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>1.0</td>
<td>624.0</td>
<td>80.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td>85.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>4.0</td>
<td>1.0</td>
<td>624.0</td>
<td>90.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td>95.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>1.0</td>
<td>624.0</td>
<td>100.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td>105.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>6.0</td>
<td>1.0</td>
<td>624.0</td>
<td>110.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td>115.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>1.0</td>
<td>624.0</td>
<td>120.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td>125.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>8.0</td>
<td>1.0</td>
<td>624.0</td>
<td>130.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td>135.0</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>9.0</td>
<td>1.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.0</td>
<td>1.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.0</td>
<td>1.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.0</td>
<td>1.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.0</td>
<td>1.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.0</td>
<td>1.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.0</td>
<td>1.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.0</td>
<td>1.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>624.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROPERTIES</td>
<td>-1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 8.8: GRAPH OF OUTLET CONCENTRATION OF A VERSUS TIME FOR EXAMPLE 2
8.3 Tubular Reactor Simulation

An exothermic tubular chemical reactor based on Chapter 7 of the IMP Manual (Brandon, 1972) was simulated. A diagram of the reactor is shown in Figure 8.8. The following reaction occurs within the reactor:

\[ \text{A} \xrightarrow{k_1} \text{B} \xrightarrow{k_2} \text{C} \]  \hspace{1cm} (8.3.1)

The partial differential equations describing the process are:

\[ \frac{\partial C_A}{\partial t} = -k_1 c^2 C_A^2 + D_1 \frac{\partial^2 C_A}{\partial z^2} - v \frac{\partial C_A}{\partial z} \]

\[ \frac{\partial C_B}{\partial t} = -k_2 c^2 C_B^2 + k_1 c^2 C_A^2 + D_2 \frac{\partial^2 C_B}{\partial z^2} - v \frac{\partial C_B}{\partial z} \]  \hspace{1cm} (8.3.2)

\[ \frac{\partial T}{\partial t} = k_1 c^2 C_A^2 H_1 + k_2 c^2 C_B^2 H_2 + D_3 \frac{\partial^2 T}{\partial z^2} - v \frac{\partial T}{\partial z} \]

Reaction, diffusion and convection terms are included in the mathematical model. The following boundary and initial conditions apply:

\[ C_A = 0 \text{ at } t = 0 \text{ for all } z \]
\[ C_B = 0 \text{ at } t = 0 \text{ for all } z \]
\[ T = 0 \text{ at } t = 0 \text{ for all } z \]
m = 11 MESH POINTS

\[ Z = 0 \]
\[ C_A = C_A^* \]
\[ C_B = C_B^* \]
\[ T = T^* \]

\[ Z = L \]
\[ \frac{3CA}{3z} = 0 \]
\[ \frac{3C_B}{3z} = 0 \]
\[ \frac{3T}{3z} = 0 \]

**FIGURE 8.9: SCHEMATIC OF TUBULAR REACTOR**
\[ C_A = C_A^0 \text{ at } z = 0 \text{ for all } t \]
\[ C_B = C_B^0 \text{ at } z = 0 \text{ for all } t \]
\[ T = T_0 \text{ at } z = L \text{ for all } t \]

\[ \frac{\partial x}{\partial z} = 0 \text{ at } z = 0 \text{ for all } t \]
\[ \frac{\partial C_A}{\partial z} = 0 \text{ at } z = L \text{ for all } t \]

\[ \frac{\partial T}{\partial z} = 0 \text{ at } z = 0 \text{ for all } t \]

\[ \frac{\partial C_B}{\partial z} = 0 \text{ at } z = L \text{ for all } t \]
\[ a_c \equiv \text{reaction two activation constant} \]

\[ H_1 \equiv \text{heat of reaction one} \]

\[ H_2 \equiv \text{heat of reaction two} \]

The values of the parameters for this simulation are:

\[ D_1 = 30.0 \quad H_1 = 1.0 \quad a_2 = 0.07 \]

\[ D_2 = 20.0 \quad H_2 = 50.0 \quad C_A^o = 10.0 \]

\[ D_3 = 90.0 \quad k_1 = 1.5 \quad C_B^o = 0.0 \quad (8.3.4) \]

\[ V = 100.0 \quad k_c = 0.00002 \quad T^o = 100.0 \]

\[ L = 100.0 \quad a_1 = 0.01 \quad t_{\text{max}} = 5.0 \]

To use DYNSYS, the set of partial differential equations must be broken down into a set of ordinary differential equations. To do this, the length of the reactor is divided into \(N\) mesh points as shown in Figure 8.9. Each mesh point will have 3 o.d.e.s (except for the inlet and outlet):

Replacing \( \frac{\partial^2 x}{\partial z^2} \) and \( \frac{\partial^2 x}{\partial z^2} \) by

\[ \frac{\partial x}{\partial z} = \frac{x|_{z} - x|_{z-\Delta z}}{\Delta z} \]  

(8.3.5)

\[ \frac{\partial^2 x}{\partial z^2} = \frac{x|_{z+\Delta z} - 2x|_{z} + x|_{z-\Delta z}}{(\Delta z)^2} \]
we obtain:

\[
\frac{dc_{A,i}}{dt} = -k_1 e^{-a_1 Ti} c_{A,i}^2 + D_1 \left[ \frac{c_{A,i}^{i+1} - 2c_{A,i} + c_{A,i}^{i-1}}{(\Delta z)^2} \right] - \nu \left[ \frac{c_{A,i}^{i+1} - c_{A,i}^i}{\Delta z} \right]
\]

\[
\frac{dc_{B,i}}{dt} = -k_2 e^{-a_2 Ti} c_{B,i} + \frac{k_1}{\varepsilon} e^{-a_1 Ti} c_{A,i}^2 + D_2 \left[ \frac{c_{B,i}^{i+1} - 2c_{B,i} + c_{B,i}^{i-1}}{(\Delta z)^2} \right] - \nu \left[ \frac{c_{B,i}^{i+1} - c_{B,i}^i}{\Delta z} \right]
\]

\[
\frac{dT_i}{dt} = k_1 e^{-a_1 Ti} c_{A,i}^2 H_1 + k_2 e^{-a_2 Ti} c_{B,i} H_2 + D_3 \left[ \frac{T_i^{i+1} - 2T_i + T_i^{i-1}}{(\Delta z)^2} \right] - \nu \left[ \frac{T_i^{i+1} - T_i^i}{\Delta z} \right]
\]

The equations are very stiff.

The equations at mesh point 1 are:

\[
\frac{dy_1}{dt} = -k_1 e^{-a_1 y_1^3} y_1^2 + D_1 \left[ \frac{y_1^{i-2} y_1^i + y_1^{i+1}}{(\Delta z)^2} \right] - \nu \left[ \frac{y_1^{i-2} y_1^i}{\Delta z} \right]
\]

\[
\frac{dy_2}{dt} = -k_2 e^{-a_2 y_2^3} y_2 + \frac{k_1}{\varepsilon} e^{-a_1 y_2^3} y_1^2 + D_2 \left[ \frac{y_2^{i-2} y_2^i + y_2^{i+1}}{(\Delta z)^2} \right] - \nu \left[ \frac{y_2^{i-2} y_2^i}{\Delta z} \right]
\]

\[
\frac{dy_3}{dt} = k_1 e^{-a_1 y_3^3} y_1^2 H_1 + k_2 e^{-a_2 y_3^3} y_2 H_2 + D_3 \left[ \frac{y_3^{i-2} y_3^i + y_3^{i+1}}{(\Delta z)^2} \right] - \nu \left[ \frac{y_3^{i-2} y_3^i}{\Delta z} \right]
\]
The equations at mesh points 2, 3, ..., M-2 are:

\[
\frac{dy_i}{dt} = -k_1 e^{a_1 y_{i+2}} y_i^2 + D_1 \left[ \frac{y_{i-1} - y_i - y_{i+1}}{(\Delta z)^2} \right] - \nu \left[ \frac{y_i - y_{i-2}}{\Delta z} \right]
\]

\[
\frac{dy_{i+1}}{dt} = -k_2 e^{a_2 y_{i+2}} y_{i+1} + k_1 e^{a_1 y_{i+2}} y_i^2 + D_2 \left[ \frac{y_{i-1} - y_i - y_{i+1}}{(\Delta z)^2} \right] - \nu \left[ \frac{y_i - y_{i+1}}{\Delta z} \right]
\] (8.3.8)

\[
\frac{dy_{i+2}}{dt} = -k_3 e^{a_3 y_{i+2}} y_i^2 H_1 + k_2 e^{a_2 y_{i+2}} y_{i+1}^2 + D_3 \left[ \frac{y_{i-1} - y_i - y_{i+1}}{(\Delta z)^2} \right] - \nu \left[ \frac{y_i - y_{i+1}}{\Delta z} \right]
\]

The equations at the last mesh point are (i=3M-5):

\[
\frac{dy_i}{dt} = -k_1 e^{a_1 y_{i+2}} y_i^2 + D_1 \left[ \frac{y_{i-1} - y_i}{(\Delta z)^2} \right]
\]

\[
\frac{dy_{i+1}}{dt} = -k_2 e^{a_2 y_{i+2}} + \frac{k_1}{e} e^{a_1 y_{i+2}} y_i^2 + D_2 \left[ \frac{y_{i-1} - y_i + y_{i+1}}{(\Delta z)^2} \right]
\] (8.3.9)

\[
\frac{dy_{i+2}}{dt} = k_1 e^{a_1 y_{i+2}} y_i^2 H_1 + k_2 e^{a_2 y_{i+2}} y_{i+1}^2 + D_3 \left[ \frac{y_{i-1} - y_i + y_{i+1}}{(\Delta z)^2} \right]
\]
where $y_1, y_4, y_7, \ldots, y_{3M-5}$ refer to $C_A$
$y_2, y_5, y_8, \ldots, y_{3M-4}$ refer to $C_B$
$y_3, y_6, y_9, \ldots, y_{3M-3}$ refer to $T$

76 mesh points were taken giving a total of 222 o.d.e.s.

The execution time for the simulation was 102.5 seconds while an equation-oriented version of Gear's method took 114.2 seconds. The IMP package required 577.7 seconds.

A listing of the module and its data set are given in Figure 8.10.

Graphs of the outlet values of $C_A$, $C_B$ and $T$ versus time are depicted in Figures 8.11, 8.12 and 8.13 respectively. No change in the outlet values is seen until about 0.6 time units. Steady state is reached after about 1.5. $C_A$ rises to a fairly sharp peak at about 1.0 and then falls to a steady state of about 0.226. The graphs of $C_B$ and $T$ are sigmoid in appearance.
SUBROUTINE TYPES

SUBROUTINE REAC3

THIS MODULE REPRESENTS A TUBULAR REACTOR WITH
REACTION 2C1-C2-C3
FEED CONCENTRATIONS MUST BE FAIRLY CONSTANT OTHERWISE ODES
MUST BE SOLVED FOR C3

EQUIPMENT PARAMETERS

1 - D1 - DIFFUSIVITY OF COMPONENT 1
2 - D2 - DIFFUSIVITY OF COMPONENT 2
3 - D3 - THERMAL DIFFUSIVITY
4 - V - FLUID VELOCITY
5 - L - LENGTH OF REACTOR
6 - H1 - HEAT OF REACTION 1
7 - H2 - HEAT OF REACTION 2
8 - K1 - REACTION 1 RATE CONSTANT
9 - K2 - REACTION 2 RATE CONSTANT
10 - A1 - REACTION 1 ACTIVATION CONSTANT
11 - A2 - REACTION 2 ACTIVATION CONSTANT
12 - M - NUMBER OF SECTIONS IN REACTOR

COMMON /MAT/ MP(1,5),EP(1,5),S(2,2,8),EX(7)
COMMON /CON/ IG,NCOMP,NCS,H,NE,NS,NPR,NPOL,TMAX,ORDER,NGRAPH
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBVMAX,ICONV,NOPPT,ISTIFF
COMMON /UNIT/ IM,NMP
COMMON /ROW/ IR(1030)
COMMON /COLUMN/ JCOL(1030)
COMMON /JACOB/ DFDY(1030)
COMMON /MODULE/ ID,ITER,ITRI,NZERO
REAL K1,K2,K1,K2,L
DIMENSION Y(222), DRY(222)

CALCULATE MODULE PARAMETERS

USE NEWTON-RAPHSON ITERATION WITH STIFF OPTION (ITER=1)

ITER=1
IF (JSTART,NE.,0.,OR.,IG,.,EQ.,1) GO TO 2
D1=EP(IM,1)
D2=EP(IM,2)
D3=EP(IM,3)
V=EP(IM,4)
L=EP(IM,5)
NEX=MP(IM,NMP+1)

FIGURE 8.10: LISTING OF MODULE AND DATA SET FOR EXAMPLE #3
TUBULAR REACTOR (222 STIFF ODES)
CALCULATE INITIAL CONDITIONS

N=3 M=3
IN=MP(IM+3)
IOUT=ABS(MP(IM+4))
IF=2
DO 1 I=1 IF,3
Y(I)=S(2*IOUT+3)*S(2*IOUT+6)
Y(I+1)=S(2*IOUT+3)*S(2*IOUT+7)
Y(I+2)=S(2*IOUT+4)
CONTINUE
1 CONTINUE
 C NZERO IS NUMBER OF NONZERO ELEMENTS IN JACOBIAN
NZERO=14*M-20
 C CALCULATE FEED CONDITIONS
 C C10=S(IG,IN,3)*S(IG,IN,6)
 C20=S(IG,IN,3)*S(IG,IN,7)
 T0=S(IG,IN,4)
 C CALCULATE JACOBIAN MATRIX ON CORRECTOR PASS
 C IF (IG.EQ.2) GO TO 4
 C MESH POINT 1
 K1=K1*EXP(A1*Y(3))
 K2=K2*EXP(A2*Y(3))
 IROW(1)=1
 JCOL(1)=1
 DFDY(1)=-2.0*K1*K2*Y(1)-2.0*D1D22-VDZ
 IROW(2)=1
 JCOL(2)=3
 DFDY(2)=-A1*K1*K2*Y(1)*Y(1)
 IROW(3)=1
 JCOL(3)=4
 DFDY(3)=D1D22
$\text{IROW(4)} = 2$
$\text{JCOL(4)} = 1$
$\text{DFDY(4)} = K1 * K* Y(1)$
$\text{IROW(5)} = 2$
$\text{JCOL(5)} = 2$
$\text{DFDY(5)} = K2 = 2.0 * D2DZ2 * VDZ$
$\text{IROW(6)} = 2$
$\text{JCOL(6)} = 3$
$\text{DFDY(6)} = A2 * K2 * Y(2) + 0.5 * A1 * K1 * K* Y(1) * Y(1)$
$\text{IROW(7)} = 2$
$\text{JCOL(7)} = 5$
$\text{DFDY(7)} = D2DZ2$
$\text{IROW(8)} = 3$
$\text{JCOL(8)} = 1$
$\text{DFDY(8)} = 2.0 * K1 * H1 * Y(1)$
$\text{IROW(9)} = 3$
$\text{JCOL(9)} = 2$
$\text{DFDY(9)} = K2 * H2$
$\text{IROW(10)} = 3$
$\text{JCOL(10)} = 3$
$\text{DFDY(10)} = A1 * K1 * H1 * Y(1) * Y(1) + A2 * K2 * H2 * Y(2) - 2.0 * D3DZ2 - VDZ$
$\text{IROW(11)} = 3$
$\text{JCOL(11)} = 6$
$\text{DFDY(11)} = D3DZ2$

\text{MESH POINTS 2,3,...,M=2}

J=1$
$\text{IF} = 1 + \text{M} = 44$
$\text{DO 3 I=12, IF+14}$
$J=J+3$
$K1 = K1 * EXP(K1 Y(J+2))$
$K2 = K2 * EXP(K2 Y(J+2))$
$\text{IROW(1)} = J$
$\text{JCOL(1)} = J+3$
$\text{DFDY(1)} = D1DZ2 * VDZ$
$\text{IROW(1+1)} = J$
$\text{JCOL(1+1)} = J$
$\text{DFDY(1+1)} = 2.0 * K1 * Y(J) - 2.0 * D1DZ2 * VDZ$
$\text{IROW(1+2)} = J$
$\text{JCOL(1+2)} = J+2$
$\text{DFDY(1+2)} = A1 * K1 * Y(J) * Y(J)$
$\text{IROW(1+3)} = J$
$\text{JCOL(1+3)} = J+3$
$\text{DFDY(1+3)} = D1DZ2$
$\text{IROW(1+4)} = J+1$
$\text{JCOL(1+4)} = J+2$
$\text{DFDY(1+4)} = D2DZ2 * VDZ$
$\text{IROW(1+5)} = J+1$
$\text{JCOL(1+5)} = J$
$\text{DFDY(1+5)} = K1 * Y(J)$
$\text{IROW(1+6)} = J+1$
JCOL(I*6)=J*1
DFDY(I*6)=K2K=2.0*D2DZ2=VDZ
IROW(I*7)=J*1
JCOL(I*7)=J*2
DFDY(I*7)=A2*K2K*Y(J+1)+0.5*A1*K1K*Y(J)*Y(J)
IROW(I*8)=J*1
JCOL(I*8)=J*4
DFDY(I*8)=D2DZ2
IROW(I*9)=J*2
JCOL(I*9)=J*1
DFDY(I*9)=D3DZ2+VDZ
IROW(I*10)=J*2
JCOL(I*10)=J
DFDY(I*10)=2.0*K1K*H1*Y(J)
IROW(I*11)=J*2
JCOL(I*11)=J+1
DFDY(I*11)=K2K*H2
IROW(I*12)=J+2
JCOL(I*12)=J+2
DFDY(I*12)=A1*K1K*H1*Y(J)*Y(J)+A2*K2K*H2*Y(J+1)-2.0*D3DZ2=VDZ
IROW(I*13)=J+2
JCOL(I*13)=J+5
DFDY(I*13)=D3DZ2
CONTINUE

MESH POINT M=1

I=14#H=30
J=3#H=5
K1K=K1*EXP(A1*Y(J+2))
K2K=K2*EXP(A2*Y(J+2))
IROW(I)=J
JCOL(I)=J-3
DFDY(I)=D1DZ2+VDZ
IROW(I+1)=J
JCOL(I+1)=J
DFDY(I+1)=2.0*K1K*Y(J)-D1DZ2=VDZ
IROW(I+2)=J
JCOL(I+2)=J+2
DFDY(I+2)=A1*K1K*Y(J)*Y(J)
IROW(I+3)=J+1
JCOL(I+3)=J-2
DFDY(I+3)=D2DZ2+VDZ
IROW(I+4)=J+1
JCOL(I+4)=J
DFDY(I+4)=K1K*Y(J)
IROW(I+5)=J+1
JCOL(I+5)=J+1
DFDY(I+5)=K2K=D2DZ2=VDZ
IROW(I+6)=J+1
JCOL(I+6)=J+2
DFDY(I+6)=A2*K2K*Y(J+1)+0.5*A1*K1K*Y(J)*Y(J)
IROW(I+7)=J+2
JCOL(I+7)=J+1
DFDY(I+7)=D3DZ2+VDZ
IROW(I+8)=J+2
JCOL(I+8)=J
DFDY(I+8)=2.0*H1*K1*KY(J)
IROW(I+9)=J+2
JCOL(I+9)=J+1
DFDY(I+9)=K2*K2
IROW(I+10)=J+2
JCOL(I+10)=J
CONTINUE

CALCULATE DERIVATIVES

MESH POINT 1

K1*K1*EXP(A1*Y(3))
K2*K2*EXP(A2*Y(3))

DERY(1)=K1*K1*Y(1)*Y(1)+D1DZ2*(Y(4)-2.0*Y(1)+C10)-VDZ*(Y(1)-C10)
DERY(2)=K2*K2*Y(1)+0.5*K1*K1*Y(2)+D2DZ2*(Y(5)-2.0*Y(2)+C20)-VDZ*(Y(2)-C20)

1(Y(2)-C20)
DERY(3)=K1*K1*Y(1)*Y(1)+K2*K2*Y(2)+D3DZ2*(Y(6)-2.0*Y(3)+T0)-VDZ*(Y(3)-T0)

MESH POINTS 2,3, ..., M-2

IF=3*IF
DO 5 I=4, I+1
K1*K1*EXP(A1*Y(I+1))
K2*K2*EXP(A2*Y(I+2))
DERY(I)=K1*K1*Y(I)*Y(I)+D1DZ2*(Y(I+3)-2.0*Y(I)+Y(I-3))-VDZ*(Y(I)-Y(I-1))

5 CONTINUE

MESH POINT M-1

IF=3*IF
K1*K1*EXP(A1*Y(I+1))
K2*K2*EXP(A2*Y(I+2))

DERY(I)=K1*K1*Y(I)*Y(I)+D1DZ2*(Y(I+3)-Y(I))-VDZ*(Y(I)-Y(I-3))

5 CONTINUE

MESH POINT M

IF=3*IF
K1*K1*EXP(A1*Y(I+1))
K2*K2*EXP(A2*Y(I+2))

DERY(I)=K1*K1*Y(I)*Y(I)+D1DZ2*(Y(I+3)-Y(I))-VDZ*(Y(I)-Y(I-3))

5 CONTINUE

CALL DIFSUB(N,Y,DERY)
IF (IDERY .NE. 0) GO TO 4

CALCULATE STREAM OUTPUT

C30 = S(1, IN3) * S(1, IN8)
C1 = Y(N-2)
C2 = Y(N-1)
C3 = C30 * 0.5 * (C10 - C1) + C20 - C2
SUM = C1 + C2 + C3
S(1, IOUT*4) = Y(N)
S(1, IOUT*6) = C1 / SUM
S(1, IOUT*7) = C2 / SUM
S(1, IOUT*8) = C3 / SUM
RETURN
END
## TUBULAR REACTOR SIMULATION

<table>
<thead>
<tr>
<th>BEGIN</th>
<th>TIME</th>
<th>5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMAK</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>IN/OUT</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>COMPS</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>LIBRARY</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>REAC3</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>PROCESS</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>REAC3</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30.0</td>
<td>-2.0</td>
</tr>
<tr>
<td>EXTRA</td>
<td>7.0</td>
<td>20.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>90.0</td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>STREAMS</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>EXPPLICIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>END</td>
<td>PROPERTIES</td>
<td>-1.0</td>
</tr>
<tr>
<td>END</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 8.11: GRAPH OF OUTLET CONCENTRATION OF A VERSUS TIME FOR EXAMPLE 9
Figure 8.12: Graph of outlet concentration of B versus time for example 9.
FIGURE 8.19: GRAPH OF OUTLET TEMPERATURE VERSUS TIME FOR EXAMPLE #3
8.4 Tubular Reactor Simulation With Tridiagonal Jacobian Matrix

An isothermal tubular reactor was simulated yielding a tridiagonal Jacobian matrix. The reaction \( A \xrightarrow{k} B \) occurs.

The partial differential equation describing the process is:

\[
\frac{\partial C_A}{\partial z} = -k C_A^2 + D_{AB} \frac{\partial^2 C_A}{\partial z^2} - \nu \frac{\partial C_A}{\partial z} \tag{8.4.1}
\]

The following boundary and initial conditions apply:

\[
\begin{align*}
C_A &= 0 \text{ at } t=0 \text{ for all } z \\
C_A &= C_A^0 \text{ at } z=0 \text{ for all } t \\
\frac{\partial C_A}{\partial z} &= 0 \text{ at } z=L \text{ for all } t
\end{align*}
\tag{8.4.2}
\]

where \( C_A \) = concentration of component \( A \) \\
\( t \) = time \\
\( z \) = space coordinate along reactor axis \\
\( C_A^0 \) = feed concentration of component \( A \) \\
\( L \) = reactor length \\
\( \nu \) = fluid velocity \\
\( D_{AB} \) = diffusivity of component \( A \) \\
\( k \) = reaction rate constant
The parameter values used are:

\[ D = 30.0 \]
\[ V = 100.0 \]
\[ L = 100.0 \]
\[ k = 10.0 \]
\[ C_A^o = 10.0 \]
\[ C_B^o = 0.0 \]
\[ t_{max} = 5.0 \] (8.4.3)

Dividing the reactor into \( M \) mesh points as in the previous example, each mesh point will have 1 o.d.e (except at the inlet and outlet)

\[ \frac{dC_{A,i}}{dt} = -kC_A^i + D_{AB}\left[ \frac{C_{A,i+1} - 2C_A^i + C_{A,i-1}}{(\Delta z)^2} \right] \]
\[ - V\left[ \frac{C_{A,i+1} - C_{A,i}}{\Delta z} \right] \] (8.4.4)

The equations are moderately stiff.

The equation at mesh point 1 is:

\[ \frac{dy_1}{dt} = -ky_1 + D_{AB}\left[ \frac{y_0 - 2y_1 + y_2}{(\Delta z)^2} \right] - V\left[ \frac{y_0 - y_1}{\Delta z} \right] \] (8.4.5)
The equations at mesh points 2, 3, ..., M-2 are:

\[
\frac{dy_i}{dt} = -ky_i^2 + D_{AB} \left[ \frac{y_{i+1} - 2y_i + y_{i-1}}{(\Delta Z)^2} \right] - \nu \left[ \frac{y_{i+1} - y_i}{\Delta Z} \right]
\] (8.4.6)

The equation at the last mesh point is:

\[
\frac{dy_n}{dt} = -ky_n^2 + D_{AB} \left[ \frac{y_{n-1} - y_n}{(\Delta Z)^2} \right]
\] (8.4.7)

where \( y \) refers to \( C_A \).

51 mesh points were taken giving a total of 49 o.d.e.s.

The execution time for the simulation was 4.04 seconds while an equation-oriented version of Gear's method took 3.92 seconds. An equation-oriented version using TRGB-TRGB2 required 6.78 seconds. Thus the tridiagonal option does permit time savings. The IMP package required 46.1 seconds.

A list of the module and its data set are given in Figure 8.14. A graph of the outlet value of \( C_B \) versus time is shown in Figure 8.15. No change in the outlet concentration is seen until about 0.5 time units. Steady state is reached at about 1.2. The graph is sigmoid in appearance.
SUBROUTINE TYPE6

SUBROUTINE REAC1

THIS MODULE REPRESENTS A TUBULAR REACTOR WITH REACTION C1->C2
FEED CONCENTRATIONS MUST BE FAIRLY CONSTANT OTHERWISE
ODES MUST BE SOLVED FOR C2
THE JACOBIAN MATRIX IS TRIDIAGONAL

EQUIPMENT PARAMETERS

1 - D = DIFFUSIVITY OF COMPONENT 1
2 - V = FLUID VELOCITY
3 - L = LENGTH OF REACTOR
4 - K = REACTION RATE CONSTANT
5 - M = NUMBER OF SECTIONS IN REACTOR

COMMON /MAT/ MP(1,5),EP(1,5),S(2,2,6),EX(7)
COMMON /CON/ IG,NCOMP,NCS,ME,NE,NS,NPR,NPOL,TMAX,ORDER,NGRAPH
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBMAX,ICONV,NOPPT,ISTIFF
COMMON /UNIT/ IN,NMP
COMMON /SUBDI/ A(49)
COMMON /DIAG/ B(49)
COMMON /SUPERD/ C(49)
COMMON /MODULE/ IDER,ITER,ITR,NZERO
DIMENSION Y(49), DERY(49)
REAL K,L

USE NEWTON-RAPHSON ITERATION FOR THIS MODULE

ITER=1

CALCULATE MODULE PARAMETERS

IF (JSTART.NE.0.OR.IG.EQ.1) GO TO 2
D=EP(IN,1)
V=EP(IN,2)
L=EP(IN,3)
K=EP(IN,4)
M=EP(IN,5)

DZ=L/FLOAT(M)
DZ2=DZ*DZ

IF (JSTART.NE.0.OR.IG.EQ.1) GO TO 2

CALCULATE INITIAL CONDITIONS
CONTINUE

MESH POINT M=1 = N

DERY(N) = K*Y(N) + Y(N) + DDZ2*(Y(N-1) - Y(N)) - VDZ*(Y(N) - Y(N-1))

CALL DIFSUB (N, Y, DERY)

IF (DERY.NE.0) GO TO 4

CALCULATE STREAM OUTPUT

C20 = S(1, IN, 3) + S(1, IN, 7)
C1 = Y(N)
C2 = C20 + C10 = C1
SUM = C1 + C2
S(1, IOUT, 6) = C1 / SUM
S(1, IOUT, 7) = C2 / SUM

RETURN

END
BEGIN
TIME 5.0
DELTAT 0.0001
IN/OUT 2.0
HMAX 1.0
COMPS 2.0
LIBRARY 1.0
REAC4 6.0
PROCESS
REAC4 1.0
1.0 -2.0
30.0 100.0 100.0 10.0 30.0
END
STREAMS
EXPLICIT 2.0
1.0 1.0 10.0 100.0 14.7
1.0 0.0
2.0 1.0 10.0 100.0 14.7
0.0 1.0
END
PROPERTIES -1.0
END
FIGURE 8.15: GRAPH OF OUTLET CONCENTRATION OF A VERSUS TIME FOR EXAMPLE 4
9. THE WILLIAMS-OTTO PLANT SIMULATION

The examples of the previous chapter were quite simple since they were only intended to illustrate different capabilities of the DYNSYS 2.0 package. In this chapter, we would like to show a simulation of a realistic chemical plant. The Williams-Otto plant (Williams and Otto, 1960; Williams, 1961) was chosen.

Around 1960, many computer control schemes had recently been proposed for use by the chemical and petroleum processing industries. There was no direct method of comparing the relative applicability of these schemes. For this reason, the Monsanto Chemical Company proposed this plant as a chemical processing model to those interested in computer control with the hope that it may serve as a basis for a direct comparison of the available computers and their relative capabilities. Every effort was made to have the model include as many as possible of the effects present in typical chemical manufacturing plants.

Williams assumed that a plant has already been designed and built for producing a chemical product P. However, considerable difficulty has been experienced with the stability of operation of the plant, and it is desired to investigate the feasibility of applying computer control
in an effort to achieve stable operation. At the same 
time, it is desired that the computer maintain an optimum 
balance of plant operating conditions to assure that the 
maximum return on investment is being obtained.

The plant consists of four major pieces of equipment, 
a stirred tank reactor, a heat exchanger, a decanter and a 
distillation column. Figure 9.1 depicts a simplified 
process flow diagram for the plant.

9.1 The Reactor

The reactor is a continuous stirred tank reactor in 
which the following reactions occur:

\[ A + B \xrightarrow{k_1} C \]
\[ B + C \xrightarrow{k_2} P + E \]
\[ P + C \xrightarrow{k_3} G \]

The desired product of the plant is \( P \), while \( G \) is 
a heavy oily waste material. Reactants \( A \) and \( B \) are fed to 
the reactor in pure form and there is a recycle stream 
containing \( A, B, C, E \) and \( P \).

Steam is available to heat up the reactor and 
cooling water is available to keep the temperature under 
control, once the reactor has been heated up to about 180°F.
FIGURE 9.1: SIMPLIFIED PROCESS FLOW DIAGRAM - WILLIAMS-OTTO PLANT

Diagram depicting a process flow with various flow rates and temperatures:
- \( F_p \) 4763 lb/hr
- \( F_d \) 39376 lb/hr
- \( F_s \) 87487 lb/hr
- \( F_g \) 92249 lb/hr
- \( F_r \) 98361 lb/hr
- \( F_c \) 3712 lb/hr
- \( F_l \) 4811 lb/hr

Temperature and flow details:
- Reactor: 180°F, 33350 lb/hr steam
- Decanter: 100°F, 98361 lb/hr
- Reaction Cooler: 100°F, 98361 lb/hr
- Cooling Water: 60°F, 53300 lb/hr
- Cooling Water: 90°F, 53300 lb/hr

Flow control and notation:
- Distillation Column
- Flow Proportioning Control
- FA 70°F, 14500 lb/hr
The following differential equations were used in the model:

mass balance on $A$:

$$\frac{dy_1}{dt} = \frac{1}{V_R} \left( F_A + F_L A_L - F_R y_1 \right) - k_1 y_1 y_2 \quad (3.1.1)$$

mass balance on $B$:

$$\frac{dy_2}{dt} = \frac{1}{V_R} \left( F_B + F_L B_L - F_R y_2 \right) - k_1 y_1 y_2 - k_2 y_2 y_3 \quad (3.1.2)$$

mass balance on $C$:

$$\frac{dy_3}{dt} = \frac{1}{V_R} \left( F_L C_L - F_R y_3 \right) + 2.0 k_1 y_1 y_2 - 2.0 k_2 y_2 y_3$$

$$- k_3 y_3 y_6 \quad (9.1.3)$$

mass balance on $E$:

$$\frac{dy_4}{dt} = \frac{1}{V_R} \left( F_L E_L - F_R y_4 \right) + 2.0 k_2 y_2 y_3 \quad (9.1.4)$$

mass balance on $G$:

$$\frac{dy_5}{dt} = \frac{1}{V_R} \left( F_L G_L - F_R y_5 \right) + 1.5 y_3 y_6 \quad (9.1.5)$$
mass balance on \( P \):

\[
\frac{dy_6}{dt} = \frac{1}{V_R} \left( F_L P_L - F_R Y_C + k_2 y_2 y_3 - 0.5 y_5 y_6 \right) \tag{9.1.6}
\]

reactor heat balance:

\[
\frac{dy_7}{dt} = \frac{1}{V_{C PR}} \left[ -2.0 k_1 y_1 y_2^2 H V_R - 3.0 k_2 y_2 y_6 H^2 V_R - 1.5 k_2 y_2 y_6 H V_R - h_{PR} (y_7 - y_8) + h_{CS} (T_S - y_7) - F_L C_{PR} (y_7 - T_L) - F_A C_{PR} (y_7 - T_A) - F_B C_{PR} (y_7 - T_B) \right] \tag{9.1.7}
\]

cooling coil heat balance:

\[
\frac{dy_8}{dt} = \frac{1}{V_{PW}} \left[ h_{PW} (y_7 - y_8) + F_{INC} (T_{INC} - y_8) \right] \tag{9.1.8}
\]

total reactor mass balance:

\[
\frac{dy_9}{dt} = F_A + F_B + F_L - F_R \tag{9.1.9}
\]

where

\[
k_7 = a_7 \exp[-b_7/(y_7 + 452.69)] \tag{9.1.10}
\]
where

\[ y_1 - y_6 \] = concentrations of \( A, B, C, E, G \) and \( P \) respectively in the reactor (mass fraction)

\[ T \] = reactor temperature (°F)

\[ T_0 \] = cooling coil temperature (°F)

\[ \nu_1, \nu_2 \] = reactor mass holdup (lb)

\[ \nu_3 \] = cooling coil mass holdup (lb)

\[ h, e \] = overall heat transfer coefficient of cooling and steam coils respectively (BTU/hr ft\(^2\) °F)

\[ a, d \] = heat transfer area of cooling and steam coils respectively (ft\(^2\))

\[ A_1, B_1, C_1, E_1, G_1, P_1 \] = recycle stream concentrations of \( A, B, C, E, G \) and \( P \) respectively (mass fraction)

\[ A_2, B_2, C_2, E_2, G_2, P_2 \] = flow rates of \( A \) feed stream, \( B \) feed stream, recycle stream, reactor exit stream and coolant respectively (lb/hr)

\[ C_{PR}, C_{PW} \] = heat capacities of reactor mixture (assumed same for all reactants) and coolant respectively (BTU/lb °F)

\[ k_1, k_2, k_3 \] = reaction rate coefficients (hr\(^{-1}\))

\[ H_1, H_2, H_3 \] = heats of reaction (BTU/lb)

\[ T_{INC} \] = inlet coolant temperature (°F)

\[ T_S \] = steam temperature (°F)
\( T_A, T_B, T_L \) \( \equiv \) temperatures of \( A \) feed stream, \( B \) feed stream and recycle stream respectively (°F)

\( a_c, b_c \) \( \equiv \) Arrhenius rate constants (hr\(^{-1}\), °R)

The cooling coil is modelled as a perfectly stirred tank. A dynamic equation for the steam coil is not included since we assume a steam trap removes condensate instantly and the temperature of the coil is thus constant at the saturated steam temperature (assumed 250°F).

\( V_R \) is assumed constant for the first seven equations and is neglected in the Jacobian; however, a total mass balance is done over the whole reactor in equation (9.1.9) in order to provide a holdup signal to the controller. A signal is also sent to a controller which keeps the ratio \( F_A/F_B \) constant.

An information flow diagram for the reactor is shown in Figure 9.2. An on-off controller turns off the steam flow when the temperature reaches a certain value. A controller operates on the cooling water flow after the reactor is heated. 

The reactor mass and heat balances are not stiff, but when the cooling coil heat balance is included, the model is stiff. Thus the module is stiff and a Jacobian matrix is supplied.
FIGURE 9.2: REACTOR INFORMATION FLOW DIAGRAM
There are four equipment parameters:

1. \( V_R \) - initial holdup in reactor (1b)
2. \( V_W \) - holdup in cooling coil (1b)
3. \( H W A H \) - product of overall cooling coil heat transfer coefficient and effective heat transfer area (BTU/°Fhr)
4. \( H S A S \) - product of overall steam coil heat transfer coefficient and effective heat transfer area (BTU/°Fhr)

9.2 The Heat Exchanger

A heat exchanger follows the reactor in order to stop the reaction and prevent an overproduction of the waste material \( G \). Both the tube and shell side of the exchanger are modelled as perfectly stirred tanks. It is assumed no reaction occurs in the heat exchanger.

The following differential equations were used in the model:

**Tube side heat balance:**

\[
\frac{dy_1}{dt} = \frac{1}{V_{HTCPR}} \left[ F_{CPR} (T_{INT} - y_1) + h_{HT} h_{LM} \right]
\]

(9.2.1)
where $T_{LM}$ is the log mean temperature difference:

$$T_{LM} = \frac{(T_{INT} - y_2) - (y_1 - T_{INS})}{\ln\left(\frac{T_{INT} - y_2}{y_1 - T_{INS}}\right)}$$  \hspace{1cm} (9.2.2)

shell side heat balance (coolant):

$$\frac{dy_2}{dt} = \frac{1}{V_{HS} C_{PW}} \left[ F_P C_P \left(T_{INS} - y_2\right) + h_u a_u T_{LM} \right]$$  \hspace{1cm} (9.2.3)

component mass balances:

$$\frac{dy_{i+2}}{dt} = \frac{F_T}{V_{HT}} \left(C_t - y_{i+1}\right) \quad i=1,6 \hspace{1cm} (9.2.4 - 9.2.9)$$

where

- $y_1$ = tube side temperature ($^\circ F$)
- $y_2$ = shell side temperature ($^\circ F$)
- $y_{A-P}$ = tube side concentrations of $A, B, C, E, G$ and $P$ respectively (mass fractions)
- $T_{INT}, T_{INS}$ = inlet tube and shell side temperatures respectively ($^\circ F$)
- $V_{HT}, V_{HS}$ = tube and shell side mass holdups respectively (lb)
- $C_{PR}, C_{PW}$ = tube and shell side heat capacities respectively (BTU/lb$^\circ F$)
- $F_T, F_S$ = tube and shell side flow rates respectively (lb/hr)
\( C_\text{i} \) \( \equiv \) inlet tube side concentrations of \( A, B, C, E, G \) and \( P \) respectively (mass fractions)

\( U_H \) \( \equiv \) overall heat transfer coefficient (BTU/°F ft\(^2\) hr)

\( a_H \) \( \equiv \) area of heat transfer (ft\(^2\))

The module is nonstiff. An information flow diagram for the heat exchanger appears in Figure 9.3. The flow of the cooling water is controlled by the outlet temperature of the heat exchanger.

There are five equipment parameters:

1. \( VHT \) - tube side mass holdup (lb)
2. \( VHS \) - shell side (coolant) mass holdup (lb)
3. \( AH \) - area of heat transfer (ft\(^2\))
4. \( HH \) - overall heat transfer coefficient (BTU/°F ft\(^2\) hr)
5. \( CPW \) - coolant heat capacity (BTU/lb°F)
FIGURE 9.3: HEAT EXCHANGER INFORMATION FLOW DIAGRAM

FIGURE 9.4: DECANTER INFORMATION FLOW DIAGRAM
9.3 The Decanter

The heavy oil waste material, \( C \), is insoluble at temperatures of less than 100°F and since it has a considerably higher specific gravity than the carrier stream in which it is suspended, it may be removed by settling. Thus a decanter follows the heat exchanger.

The top and bottom layers are both modelled as perfectly stirred tanks. The mass fraction of \( C \) in the reactant stream as a function of temperature is given.

The decanter temperature determines the mass fraction of \( C \) entering the top layer.

The following differential equations were used in the model:

**Total mass balance:**

\[
\frac{dy_1}{dt} = F_{IN} - F_{OUT} - F_{OUTB} \tag{9.3.1}
\]

**Total heat balance:**

\[
\frac{dy_2}{dt} = \frac{1}{y_1} \left[ F_{IN}y_{IN} - (F_{OUT} + F_{OUTB}) y_2 \right] \tag{9.3.2}
\]

**Component mass balances for the top layer:**

\[
\frac{dy_{i+2}}{dt} = F_{INT}y_{INT} - F_{OUT} \frac{y_{i+2}}{\sum_{i=1}^{6} y_{i+2}} \tag{9.3.3-9.3.8}
\]
mass balance for bottom:

\[
\frac{dy_c}{dx} = F_{out} - F_{in}
\]

(3.2.3)

where

- \( y_c \) = total mass holdup (lb)
- \( T \) = decanter temperature (°F)
- \( Y_{A}, Y_{B}, Y_{C}, Y_{E}, Y_{G}, Y_{P} \) = top layer holdups of \( A, B, C, E, G \) and \( P \), respectively (lb/hr)
- \( y_c \) = holdup of \( G \) in bottom layer (lb)
- \( T_{in} \) = inlet temperature (°F)
- \( F_{out}, F_{outB} \) = outlet flow rates from top and bottom layers respectively (lb/hr)
- \( F_{in}, F_{inB} \) = inlet flow rates to top and bottom layers respectively (lb/hr)
- \( F_{in} \) = total decanter inlet flow rate (lb/hr)

Figure 9.4 shows an information flow diagram for the decanter. The holdups in the top and bottom layers are controlled.

There are three equipment parameters:

1. \( HUPT \) - initial holdup in top layer (lb)
2. \( HUPS \) - initial holdup in bottom layer (lb)
3. \( LSEP \) - number of component to be separated from entering stream
9.4 The Distillation Column

Williams and Otto proposed to model the column as a stirred tank. Rather unexpectedly, this did not give satisfactory results. Pulido (1975) has created DYNSYS 2.0 modules of a distillation column, condenser, reboiler and reboiler drum. These are discussed in detail in his thesis.

Assumptions used were no heat of mixing, constant holdup on each plate and equilibrium on each plate at all times. The column module contains mass and energy balances about each plate. Bubble point calculations are done. The condenser is either a total or a partial condenser. The reboiler module handles the heat transfer calculations, while the reboiler drum module does the mass and energy balances for the reboiler. The azeotrope in the column was neglected for simplicity, but it can be implemented once a correlation for the liquid activity coefficients is found, such as the Wilson equation.

The column, condenser drum, reboiler and reboiler drum modules are modelled as being stiff and Jacobian matrices are supplied. The condenser module is algebraic.

An information flow diagram for the column appears in Figure 9.5. Since the column uses molar rather than mass quantities, a module CONV1 converts the mass flow and mass fractions to molar flow and mole fractions respectively before entering the column and does the opposite for the
FIGURE 9.5: DISTILLATION COLUMN INFORMATION FLOW DIAGRAM
bottoms stream leaving the column.

A module SPLT1 divides the bottoms flow, according to a given ratio, into two streams, one of which is recycled back to the reactor.

9.5 Simulation Results

A listing of all the modules and a possible data set are contained in Appendix H. The modules form the basis of a library of modules for DYNSYS 2.0. In all, 31 modules and 42 streams were required.

The reactor was started up from 70°F assuming a constant recycle stream. Figure 9.6 shows a graph of the reactor concentrations versus time. Figure 9.7 shows a graph of the reactor and exit heat exchanger temperatures versus time. Steady state was reached after about 0.7 hours.

The control scheme for the reactor kept the temperature at 180°F after some slight oscillation. The steam flow is cut off at 175°F and a proportional controller with a high gain controls the cooling water flow so that the temperature does not rise above 180°F. This is similar to the control scheme recommended by Williams. The steady state cooling water flow rate at 180°F with no steam flow was calculated to be 5115 lb/hr. The reactor concentrations at steady state were the same as given by Williams.
FIGURE 9.6: GRAPH OF REACTOR CONCENTRATIONS OF A, B, C, E, G AND P VERSUS TIME
Figure 9.7: Graph of reactor and exit heat exchanger temperatures versus time
A value of 5000.0 was used for $k_0a_0$. The value of 10000.0 given by Williams heats the reactor too fast and the temperature becomes out of control.

The heat exchanger outlet temperature was below 102°F at all times. Since the reactor temperature control works well and the heat exchanger cools the reactor outlet stream efficiently, no trouble was encountered in the decanter. The decanter temperature never rose above 100°F and all of the $C$ was settled out.

The distillation column results were handled by J. Pulido and a discussion of these may appear in his thesis (Pulido, 1975).

DYNSYS 2.0 performed the simulation quite well and is very convenient for simulating a modular plant such as this one. However, during the simulation, corrector convergence problems occurred often for the heat exchanger even with a supplied Jacobian. Possibly the model will have to be refined.

The Williams-Otto plant offers many possibilities for control system studies. More work may be done of this in the future. DYNSYS 2.0 is a very convenient tool for evaluating different control policies. Pulido (1975) has used DYNSYS 2.0 to evaluate various distillation column control schemes.
10. CONCLUSION

10.1 Summary

The numerical solution of ordinary differential equations is a fundamental aspect of dynamic simulation. The original version of DYNSYS used a third order Adams-Moulton-Shell routine; however, this is very inefficient for stiff systems where there is a wide variation of time constants. In chemical engineering, stiff o.d.e.s occur widely in reaction kinetics and to some extent in multi-stage systems. Conventional numerical techniques are restricted by stability to using a very small step size resulting in large computer times. There have been many new numerical techniques published in the recent literature directed at the efficient numerical solution of systems of stiff o.d.e.s. A literature survey of these has been made.

Most stiff techniques are implicit and require a technique such as Newton-Raphson iteration to converge. Each Newton-Raphson iteration involves the solution of a system of linear algebraic equations (usually sparse) equal in size to the number of o.d.e.s. For a large stiff system, this requires considerable computer time. Various
recently developed sparse linear equation solvers have been evaluated and that of Bending and Hutchison appears to be the most appropriate.

A comparison of several integration techniques has been done with a number of nonstiff and stiff test examples: Gear's method appears to be the most efficient. Gear's integration algorithm in conjunction with the Bending-Hutchison linear equation solver has been implemented into DYNSYS version 2.0. An option for stiff systems with tridiagonal Jacobian matrix is also included. Several simulation examples have been described including the simulation of the Williams-Otto plant. The package is a very powerful one, capable of the simultaneous numerical solution of hundreds of stiff o.d.e.s.

The modular approach has been chosen as the framework of our study, but certainly the results apply equally well to the equation-oriented approach. An equation-oriented program containing Gear's method and the Bending-Hutchison linear equation solver has been created and it is useful for simulating nonmodular systems.

An interactive version of the executive, where the output can be displayed graphically on a CRT screen, has also been developed.
10.2 Future Work

A version of the executive which keeps the operator lists and possibly other variables in auxiliary storage can easily be created. Segments of the operator list can be stored or read in, thus allowing much larger systems to be simulated. Thousands of o.d.e.s could then be solved, but the program would not be as fast because of the extra time required to store and access the operator lists. It would also make the program more machine dependent.

A tridiagonal option has been included, but other options might well be included, similar to those in the IMP package, such as options for handed matrices in general rather than just tridiagonal, or iterative options for problems with diagonally dominant Jacobians.

This thesis has concentrated on the numerical solution of stiff systems, particularly large ones. Another area which can be studied is the convergence of recycle in dynamic simulation (Koenig, 1972).

With the present package, it is reasonably difficult for the novice to learn how to write modules. Ways of simplifying the module structure for user convenience should be studied.

A library of modules could be developed to make future programming easier. Pulido (1975) and Millares (1975) are presently using DYNSYS 2.0 to simulate a commercial
process. The modules from their work and from the Williams-Otto plant simulation would form the basis of such a library. Additional modules could be added corresponding to the types of equipments used in chemical plants.

The package should be applied to other industrial processes.

A physical properties package could be added to the system.

A version of DYNSSYS which can use CRT graphical input in the form of an information flow diagram is being developed.

The applicability of the pseudo state approach is a possible area of research. We pointed out in Chapter 3 that it is a dangerous method to apply; however the circumstances under which it can be applied and the amount of error incurred are questions of interest in solving stiff systems.

All of the test examples in Appendix D are of the relaxation type; i.e. the fast components become essentially zero for a very short time. In a control system simulation with disturbances, the fast modes will be constantly re-introduced into the system. Whether Gear's method is the best numerical technique under such conditions should be investigated.
The study of numerical techniques for solving stiff o.d.e.s and for solving sparse linear algebraic equations is a currently active area of research and more efficient techniques than those used in DYNSYS 2.0 will undoubtedly be developed. Gear himself has a new, more efficient version of DIFSUB which will be available around the end of 1974 (Gear 1974). DYNSYS 2.0 will be modified to take this into account and the evolution of the package will include the incorporation of more efficient techniques.
APPENDIX A: GEAR'S METHOD

C.W. Gear (Gear, 1971a,b,c; 1972) has created a subroutine called DIFSUB for solving ordinary differential equations. It uses a variable-order, variable-step, linear, predictor-corrector algorithm.

There are options for nonstiff and stiff equations.

A.1 Nonstiff Option

The nonstiff option uses an Adams-Bashforth predictor and an Adams-Moulton corrector:

PREDICTOR EQUATION: \( y_{n+1} = y_n + h \sum_{i=1}^{\kappa} \beta_i y_{n+1-i} \) \( (A.1) \)
(Adams-Bashforth)

CORRECTOR EQUATION: \( y_{n+1} = y_n + h \sum_{i=0}^{\kappa} \beta_i^* y_{n+1-i} \) \( (A.2) \)
(Adams-Moulton)

The coefficients \( \beta_i \) and \( \beta_i^* \) may be found in Henrici (1962). The order may vary from 1-7.
A.2 Stiff Option

The stiff option uses a predictor and corrector based on the backward or numerical differentiation methods (Henrici, 1962):

\[
\text{PREDICTOR EQN.: } y_{n+1} = h \n_1 y_n + \sum_{i=1}^{k} \alpha_i y_{n+1-i}
\]  \hspace{1cm} (A.3)

\[
\text{CORRECTOR EQN.: } y_{n+1} = h \n_0 y_{n+1} + \sum_{i=1}^{k} \beta_i y_{n+1-i}
\]  \hspace{1cm} (A.4)

The corrector coefficient values are given in Gear (1971c). From 1st to 6th order is available.

The corrector equation has the property of stiff stability (Gear, 1969a; Figure A.1) for up to 6th order, i.e., it is accurate and stable near the origin and stable to the far left of the origin in the $\Re \lambda$ plane.

Accuracy is not important far to the left of the origin, since the components here are very small, but absolute stability is required. Around the origin, accuracy is essential for which we need relative or absolute stability. Gear's method is relatively stable in the rectangle shown. Elsewhere there are no requirements, since to the right of $\Re(\Re \lambda) = \alpha$ and above and below $\Im(\Re \lambda) = \pm \delta$, the function is varying too rapidly to be represented by points spaced a distance $h$, but is not decaying so fast that we can ignore such terms. For order 1-6, $p > -6.1$, $\delta < 0.5$ and $\alpha = 0$ (Gear 1969a).
FIGURE A.1: STIFF STABILITY
The Jacobian matrix is needed for the stiff option, as up to three Newton-Raphson iterations of the corrector are performed. The Jacobian may be user-supplied or evaluated numerically. It is re-evaluated only when there is a change in order or the corrector failed to converge.

Much of the algorithm is the same for the nonstiff and stiff option except that different coefficients are used.

A.3 General Features

Rather than storing back values of the independent variable and the first derivatives, the algorithm uses a variation of the Nordsieck vector (Nordsieck, 1962):

$$\mathbf{S}_n = \left[ y_n, \frac{h^2 \dot{y}_n}{2}, \ldots, \frac{h^p \dot{y}_n}{p!} \right]$$  \hspace{1cm} (A.5)

Gear expresses the predictor-corrector in the form:

**PREDICTOR EQN.:** \[ \mathbf{S}_{n+1}(\xi) = \mathbf{A} \mathbf{S}_n \]  \hspace{1cm} (A.6)

**CORRECTOR EQN.:** \[ \mathbf{S}_{n+1}(\xi) = \mathbf{S}_{n+1}(\xi\xi) + I \mathbf{A}^{\xi-1} \mathbf{F}(\mathbf{S}_{n+1},(\xi)) \]  \hspace{1cm} (A.7)

where \( \mathbf{A} \) is the Pascal triangle matrix:

$$\mathbf{A}_{\xi\xi} = \binom{\xi}{\xi}$$  \hspace{1cm} (A.8)

$$= 0 \text{ if } \xi > \xi$$
\( E(a) \) is the amount by which the differential equation is not satisfied locally by \( a \):

\[
E(a) = \left| \xi(t, a) - y \right| = h \xi(\tau, y) - h y
\]  

(A.8)

For direct iteration as in the nonstiff option

\[
\hat{x} = I \quad \text{(identity matrix)}
\]  

(A.10)

For Newton-Raphson iteration as in the stiff option

\[
\hat{x} = \left[ \begin{array}{cc}
\frac{2}{\tau} & -1 \\
-1 & \frac{1 - \frac{2}{\tau}}{\tau}
\end{array} \right]
\]  

(A.11)

where \( \frac{2}{\tau} \) is the Jacobian matrix

The coefficients \( \hat{x} \) are given in Gear (1971c). \( I \) is chosen to achieve stability and accuracy.

The same order is used on the predictor and corrector steps. The local truncation error is given by

\[
\xi_{k+1} = (k+1) + O(h^{k+2}) \quad \text{for a } k\text{th order step.}
\]

Values of the Nordsieck vector are given in Gear (1971c). The last component of the Nordsieck vector is \( \xi_{k+1}^{(k)}/k! \) and under suitable smoothness conditions, its backward difference yields an estimate of \( \xi_{k+1}^{(k+1)}/k! \). The error is controlled by keeping the Euclidean norm of the relative local truncation errors below a specified value.

The method is self-starting, using the first order formula to begin the integration. The order is variable.
and is chosen to maximize the step size with some heuristic controls. The algorithm begins with first order and increases quickly to about fourth order where it remains for the bulk of the integration. Then as steady state is approached, the order decreases back to first order.

The nomenclature was first introduced in Gear (1967). Ratliff (1968) does some computations with the method. Gear (1971d) has extended his program to handle large sparse sets of simultaneous differential and algebraic equations; however, the program is in machine language.

Several workers have recommended Gear's method for stiff systems. Nikolai (1973) tells of his experience with it. Walters (1972) provides a detailed discussion of the method. A version of DIFSUB is available in the IMSL library (IMSL, 1973).
APPENDIX B: IMP

D.M. Brandon (Brandon, 1972, 1973, 1974a), now with Control Data Corporation, has developed IMP (Implicit Solution Software System). IMP is a software system for the direct or iterative solution of large differential and/or algebraic systems. The package is equation-oriented and contains the author's own A-stable numerical integration technique and sparse matrix routines for solving the corrector equation. IMP is offered as a standard CDC software product through their Application Services Division. For non-CDC computers, the package is available from Brandon himself.

The IMP tests in this thesis were done with an object copy of version 1.1 compiled under FTN version 4.0, optimization level 2, Scope 3.4 operating system.

The IMP user must write his own user subroutine defining the system to be simulated and choosing from the many options available. The problem must be expressed in the form:

\[ \dot{y} = \frac{1}{2} y + b \]  

(3.0)
where $A$ is the Jacobian matrix
and $\beta$ is the augmented constant vector.

Both $A$ and $\beta$ may be functions of time or $y$. The
left hand side is set to zero for algebraic equations.
Partial differential equations are reduced to ordinary
differential equations by discretization of one or more
dependent variables.

B.1 Brandon's Integration Technique

Brandon's integration procedure (Brandon, 1974f) is
a single-step method:

$$y_{n+1} = y_n + h[(1 - \omega)\dot{y}_{n+1} + \omega \dot{y}_n] \tag{B.1}$$

For the equation:

$$\dot{y} = \lambda y \tag{B.2}$$

$$y_{n+1} = y_n e^{\lambda h} \tag{B.3}$$

Substitute (B.2) into (B.1)

$$y_{n+1} = y_n + h[(1 - \omega)\lambda y_{n+1} + \omega \lambda y_n] \tag{B.4}$$

$$y_{n+1}[1 - \hat{h}\lambda(1 - \omega)] = y_n[1 + \hat{h}\lambda] \tag{B.5}$$
\[ \frac{y_{n+1}}{y_n} = \frac{1 + \lambda \psi}{1 - \lambda (1 - \psi)} = e^{\lambda n} \]  \hspace{1cm} (B.6)

Solving for \( \psi \):

\[ \psi = -\frac{\frac{1}{\lambda}}{e^{\lambda n} - 1} \]  \hspace{1cm} (B.7)

Liniger and Willoughby (1970) introduced the concept of exponential fitting in that \( \psi \) can be chosen to make the integration exact for a given value of \( \lambda \).

For more than one equation, different values of \( \psi \) may be used:

\[ \psi_i = -\frac{1}{\lambda_i} - \frac{1}{e^{2 \lambda_i} - 1} \]  \hspace{1cm} (B.8)

There will not be an eigenvalue corresponding to each equation except for the system:

\[ \psi_i = \lambda_i \psi_i \hspace{0.5cm} i=1,2,\ldots,\nu \]  \hspace{1cm} (B.9)

Brandon chooses \( \psi \) to approximate the local gradient:

\[ z_i = \left( \frac{\partial y_i}{\partial y_i} \right) = \lambda \sum_{j=1}^{\nu} c_{ij} \left( \frac{y_{i+1}}{y_i} \right) \]  \hspace{1cm} (B.10)
For the single equation $\dot{y} = \lambda y$, this reduces to exponential fitting. Thus Brandon has extended the concept of exponential fitting to systems of equations.

Note that no predictor equation is used. The first value for the corrector iteration is taken as $y_n$. The method is A-stable for $0 \leq \omega \leq 0.5$.

There is a very conservative local truncation error estimate. Expanding $y_{n+1}$ in a Taylor Series:

$$y_{n+1} = y_n + h \dot{y}_n + \frac{h^2}{2!} \ddot{y}_n + \frac{h^3}{3!} \frac{\dddot{y}}{3} + \ldots \quad (B.11)$$

$$\omega y_{n+1} = \omega y_n + h \omega \dot{y}_n + \frac{h^2}{2!} \omega \ddot{y}_n + \frac{h^3}{3!} \omega \frac{\dddot{y}}{3} + \ldots \quad (B.12)$$

$$(1-\omega)y_{n+1} = (1-\omega)y_n + h(1-\omega)\dot{y}_n + \frac{h^2}{2!} (1-\omega)\ddot{y}_n + \frac{h^3}{3!} (1-\omega)\frac{\dddot{y}}{3} + \ldots \quad (B.13)$$

Adding (B.12) and (B.13)

$$y_{n+1} = y_n + h \omega \dot{y}_n + h(1-\omega)\ddot{y}_{n+1} + h(1-\omega)\dddot{y}_n + \frac{h^2}{2!} \dddot{y}_n + \frac{h^3}{3!} \frac{\dddot{y}}{3} + \ldots \quad (B.14)$$

The error term is:

$$E = -h(1-\omega)\ddot{y}_{n+1} + h(1-\omega)\dddot{y}_n + \frac{h^2}{2!} \dddot{y}_n + \frac{h^3}{3!} \frac{\dddot{y}}{3} + \ldots \quad (B.15)$$
Expanding \( y_{n+1} \) in a Taylor Series:

\[
y_{n+1} = y_n + \frac{h}{2} \dot{y}_n + \frac{h^2}{24} \ddot{y}_n + \cdots \tag{3.16}
\]

Substitute (B.16) into (B.15)

\[
E = h^2 \left( \omega - \frac{1}{2} \right) \dot{y}_n + h^3 \left( \frac{\omega}{2} - \frac{1}{3} \right) \ddot{y}_n \tag{3.17}
\]

Replace \( \ddot{y}_n \) by \( \frac{y_{n+1} - y_n}{h} \)

\[
E = \frac{h^2}{12} \left[ (\omega - 2) \ddot{y}_n + (\omega - 4) \ddot{y}_{n+1} \right] \tag{3.18}
\]

Brandon adjusts this term empirically to:

\[
E = \left| \frac{\frac{h^2 \ddot{E}}{\ddot{y}_{n+1} - \ddot{y}_n}}{\ddot{E} (\ddot{y}_{n+1} - \ddot{y}_n)} \right| \left( \frac{1}{\ddot{E}} \right)^2 + \frac{h(\ddot{y}_{n+1} - \ddot{y}_n)}{15(\ddot{y}_{n+1} + \ddot{y}_n)} \tag{3.19}
\]

where

\[
\ddot{E} = \left[ (1 + \ddot{e}_r) \ddot{y}_{n+1} - (1 - \ddot{e}_r) \ddot{y}_n \right] \tag{3.20}
\]

\[
\ddot{e}_r = \frac{\Sigma a_{i,\ddot{e}} y_i}{\Sigma a_{i,\ddot{e}}} \tag{3.21}
\]

\[
\ddot{e}_r = 3(1 - \frac{2\omega}{h}) \tag{3.22}
\]

\[
y_i = 4\delta \frac{|a_{i,\ddot{e}}|}{\Sigma |a_{j,\ddot{e}}|} + \delta \tag{3.23}
\]
\( \dot{x} \) is the augmented constant vector
\( z \) is restricted as:

\[-170 \leq z \leq -0.015 \]  \hspace{1cm} (3.24)

This is the error term for IMP version 1.1; however, Brandon has used in later versions a new error option (Brandon, 1974d). This consists of taking a double step and comparing the solution with two single steps. This approach has been used with Runge-Kutta methods (Carnahan, Luther and Wilkes, 1969).

B.2 Sparse Linear Equation Solution

Version 1.1 offers many options for solving sparse linear equations. These can occur in solving the corrector or the user can solve his own set of linear equations.

There are two iterative options: Gauss-Seidel double sweep and gradient iteration. Direct elimination using the Crout approach is available. In elimination, the user has options for ordering the system matrix, handling the fill-in and pivoting. Ordering is important, since the order in which the variables are eliminated affects the number of nonzero elements generated (fill-in). However, in normal applications problems, the natural order of the equations would be such that the bandwidth would be minimized.
The storage scheme used is cumulative indexing (section 5.1.3) and quasilinearization (section 4.5) is used in the corrector solution.

Future versions of IMP will include the following additions (Brandon 1974b):

(1) the conjugate gradient method
(2) special elimination routines for banded matrices
(3) special Gauss-Seidel routine for banded matrices
(4) special elimination routine for symmetric and banded symmetric matrices
(5) special conjugate gradient method for symmetric matrices
(6) full matrix elimination for smaller full matrix problems.
APPENDIX C: SHANNON'S METHOD

Shannon (1971) has developed a technique for the numerical integration of stiff, sensitive and multivalued equations.

For a single equation, the method is very simple.

\[
\frac{dy}{dt} = f(t, y) \quad (C.1)
\]

If \( f(t, y) \) is less than 1, \( t \) is the dependent variable or the variable of integration. However, if \( f(t, y) \) is or becomes greater than one, \( y \) is made the variable of integration.

i.e.

\[
\frac{dt}{dy} = \frac{1}{f(t, y)} \quad (C.2)
\]

This may be extended to systems of equations as follows.

For the system:

\[
\frac{dy_1}{dy_n} = f_1(y_1, \ldots, y_n) \\
\frac{dy_2}{dy_n} = f_2(y_1, \ldots, y_n) \\
\vdots \\
\frac{dy_n}{dy_n} = 1
\]

(C.3)
where \( u_\kappa \) is time, the normal variable of integration, if we form the matrix of all possible derivatives,

\[
D = \begin{bmatrix}
\frac{dy_1}{dy_1} & \frac{dy_1}{dy_2} & \cdots & \frac{dy_1}{dy_\kappa} \\
\frac{dy_2}{dy_1} & \frac{dy_2}{dy_2} & \cdots & \frac{dy_2}{dy_\kappa} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{dy_\kappa}{dy_1} & \frac{dy_\kappa}{dy_2} & \cdots & \frac{dy_\kappa}{dy_\kappa}
\end{bmatrix}
\]

(C.4)

It can be shown that there is always at least one column of \( D \) where the values of all derivatives are less than or equal to one in absolute value. If this is the \( \kappa \)th column, then \( u_\kappa \) should be taken as the variable of integration. If the integration proceeds with \( u_\kappa \) as the variable of integration and the largest element \( \frac{dy_i}{dy_\kappa} \) becomes greater than 1, then \( u_i \) will become the independent variable. The derivatives in column \( i \) will then be less than or equal to 1.

Thus the variable of integration is changed as the integration proceeds, so that all the derivatives with respect to that variable are less than or equal to 1. Any method of integration may be used.

Let us try this method on SYSTEM I (Appendix D).
the original system is:

\[
\frac{dy_1}{dt} = -500.5y_1 + 499.5y_2, \quad y_1(0) = 0
\]
\[
\frac{dy_2}{dt} = 499.5y_1 - 500.5y_2, \quad y_2(0) = 2
\]

with analytical solution:

\[
y_1 = e^{-t} - e^{-1000t}
\]
\[
y_2 = e^{-t} + e^{-1000t}
\]

making time $\tau$, the system becomes:

\[
\frac{dy_1}{d\tau} = -500.5y_1 + 499.5y_2, \quad y_1(0) = 0
\]
\[
\frac{dy_2}{d\tau} = 499.5y_1 - 500.5y_2, \quad y_2(0) = 2
\]
\[
\frac{dy_3}{d\tau} = 1, \quad y_3(0) = 0
\]

the matrix of derivatives is:

\[
P = \begin{bmatrix}
\frac{dy_1}{dy_3} & \frac{dy_1}{dy_2} & \frac{dy_1}{dy_3} \\
\frac{dy_2}{dy_3} & \frac{dy_2}{dy_2} & \frac{dy_2}{dy_3} \\
\frac{dy_3}{dy_3} & \frac{dy_3}{dy_2} & \frac{dy_3}{dy_3}
\end{bmatrix}
\]

(C.8)
With \( \gamma \) as the variable of integration, the Jacobian matrix is:

\[
J = \begin{bmatrix}
\frac{\partial x}{\partial \gamma} & \frac{\partial y}{\partial \gamma} \\
\frac{\partial z}{\partial \gamma} & \frac{\partial w}{\partial \gamma}
\end{bmatrix}
\]

At time 0, \( \gamma = 0 \).

The eigenvalues are -1, 1000 and 0.

Shannon’s method say to make \( \gamma \) the independent variable.
If we make $y_2$ the independent variable

\[
\frac{\partial y}{\partial y_2} = \frac{-423.5y_1 + 600.5y_2}{423.5y_1 - 600.5y_2} \quad \frac{\partial y}{\partial y_2} = 1 \\
\frac{\partial y}{\partial y_2} = \frac{423.5}{423.5y_1 - 600.5y_2}
\]

(C.15)

The Jacobian matrix is:

\[
\begin{bmatrix}
1000y_2 & -1000y_1 \\
\frac{(423.5y_1 - 600.5y_2)^2}{(423.5y_1 - 600.5y_2)^2} & \frac{-423.5}{(423.5y_1 - 600.5y_2)^2} \\
\frac{-423.5}{(423.5y_1 - 600.5y_2)^2} & \frac{600.5}{(423.5y_1 - 600.5y_2)^2}
\end{bmatrix}
\]

(C.16)

The eigenvalues are

\[
\frac{1000y_2}{(423.5y_1 - 600.5y_2)^2}, \quad 0, \quad 0
\]

at time $t$: $y_1, y_2 = 0$  \[\lambda = \frac{1000}{(1001)^2} = 3 \times 10^{-2}, \quad 0, \quad 0\]  (C.14)

as $t \to 0.01$  \[y_1, y_2 = 0 \quad \lambda = \frac{1000}{y} \to \infty, \quad 0, \quad 0\]  (C.15)
To see what happens when \( y_1 \) is the variable of integration

\[
\frac{dy_1}{dy_2} = 1
\]

\[
\frac{dy_2}{dy_1} = \frac{499.5y_1 - 500.6y_2}{-500.6y_1 + 499.5y_2}
\]

\[
\frac{dy_3}{dy_1} = \frac{1}{-500.6y_1 + 499.5y_2}
\]

The Jacobian matrix \( \mathbf{J}(y_1) \) is:

\[
\mathbf{J}(y_1) = \begin{bmatrix}
0 & 0 & 0 \\
\frac{-1000y_2}{(-500.6y_1 + 499.5y_2)^2} & \frac{1000y_1}{(-500.6y_1 + 499.5y_2)^2} & 0 \\
\frac{500.6}{(-500.6y_1 + 499.5y_2)^2} & \frac{-499.5}{(-500.6y_1 + 499.5y_2)^2} & 0 \\
\end{bmatrix}
\]

The eigenvalues are:

\[
\frac{1000y_1}{(-500.6y_1 + 499.5y_2)^2}, 0, 0.
\]

At time 0: \( y_1 = 0, y_2 = 0 \)

\( \lambda = 0, 0, 0 \) \hspace{1cm} (C.18)

as \( t \to 0.01 \) \( y_1 - y_2 = y + 0 \)

\( \lambda = \frac{1000}{y} \to \infty, 0, 0 \) \hspace{1cm} (C.19)
If $y_2$ is made the independent variable, the eigenvalues of the Jacobian at time 0 are 0, 0, $2 \times 10^{-3}$. However a very short time later (-0.01), the largest eigenvalue approaches infinity.

Similarly if $y_1$ is made the variable of integration, the largest eigenvalue also approaches infinity.

As $t \to 0.01$, $e^{-1000t} \to 0$, and $y_1' + y_2 = y_2' + y_2 = 0$

The matrix of derivatives becomes:

$$
\begin{bmatrix}
1 & 1 & -\gamma \\
-1 & 1 & -\gamma \\
1 & -1 & 1
\end{bmatrix} \to 
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
-\infty & -\infty & 1
\end{bmatrix} 
$$

As $t$ approaches 0.01 approximately, $y_3$ (time) should be made the variable of integration.

Thus for a very short period of time, the change of independent variable from $y_3$ to $y_3$ would speed up the integration, however, after that period we are back where we started with $y_3$ (time), again the independent variable.

Thus Shannon's method seems to be of limited effectiveness for handling stiff systems.
APPENDIX D: TEST EXAMPLES

Seven stiff examples were tested:

SYSTEM I: Moderately Stiff Linear Real Eigenvalue Example

II: Very Stiff Linear Real Eigenvalue Example

III: Linear Complex Eigenvalue Example

IV: Krogh's Example

V: Chemistry Example

VI: Polymer Example

VII: Stable Polymer Example

and four nonstiff examples:

SYSTEM VIII: Linear Nonstiff Complex Eigenvalue Example

IX: Krogh's Nonstiff Example

X: Nonlinear Reaction Example

XI: Gas Absorber Example
D.1 SYSTEM I: Moderately Stiff Linear Real Eigenvalue Example

We use the 2-dimensional linear system:

\[
\begin{cases}
\dot{z} = A_z y = \begin{bmatrix} -a & b \\ b & -a \end{bmatrix} y \\
y(0) = \begin{bmatrix} 0 \\ 2 \end{bmatrix}
\end{cases}
\]  \hspace{1cm} (D.1.1)

with analytical solution:

\[
\begin{align*}
y_1 &= e^{-(a-b)t} - e^{-(a+b)t} \\
y_2 &= e^{-(a-b)t} + e^{-(a+b)t}
\end{align*}
\]  \hspace{1cm} (D.1.2)

We choose \(a = 500.5\), \(b = 499.5\) to yield the moderately stiff example:

\[
\begin{align*}
y_1 &= -500.5 y_1 + 499.5 y_2, \quad y_1(0) = 0 \\
y_2 &= 499.5 y_1 - 500.5 y_2, \quad y_2(0) = 2
\end{align*}
\]  \hspace{1cm} (D.1.3)

with analytical solutions:

\[
\begin{align*}
y_1 &= e^{-t} - e^{-1000t} \\
y_2 &= e^{-t} + e^{-1000t}
\end{align*}
\]  \hspace{1cm} (D.1.4)

The range of integration is \([0,1]\)
The eigenvalues are -1 and -1000.
D.2 SYSTEM II: Very Stiff Linear Real Eigenvalue Example

As in SYSTEM I we choose \( a = 500000.5 \), \( b = 499999.5 \) to yield the very stiff example:

\[
\begin{align*}
\dot{y}_1 &= -500000.5y_1 + 499999.5y_2 \\
\dot{y}_2 &= 499999.5y_1 - 500000.5y_2
\end{align*}
\]

with analytical solution:

\[
\begin{align*}
y_x &= e^{-t} - e^{-1000000t} \\
y_e &= e^{-t} + e^{-1000000t}
\end{align*}
\] (D.2.2)

The range of integration is \([0,1]\)
The eigenvalues are \(-1\) and \(-1000000\).
The stiffness ratio of \(10^6\) is as large as has been found in practice.

D.3 SYSTEM III: Linear Complex Eigenvalue Example

The system:

\[
\begin{align*}
\dot{y}_1 &= -Ay_1 + By_2 + (A - B - 1)e^{-t} & y_1(0) &= 2 \\
\dot{y}_2 &= -By_1 + Ay_2 + (A + B - 1)e^{-t} & y_2(0) &= 2 \\
\dot{y}_3 &= -Cy_3 + Dy_4 + (C - D - 1)e^{-t} & y_3(0) &= 2 \\
\dot{y}_4 &= -Dy_3 - Cy_4 + (C + D - 1)e^{-t} & y_4(0) &= 2
\end{align*}
\] (D.3.1)
is used with analytical solution:

\[ y_1 = e^{-At} (\cos Bt + \sin Bt) + e^{-t}, \]
\[ y_2 = e^{-At} (\cos Bt - \sin Bt) + e^{-t}, \]  \hspace{1cm} \text{(3.3.2)}
\[ y_3 = e^{-Ct} (\cos Dt + \sin Dt) + e^{-t}, \]
\[ y_4 = e^{-Ct} (\cos Dt - \sin Dt) + e^{-t}. \]

We choose \( A = 1000, B = C = D = 1 \) to yield the
moderately stiff example:

\[ \begin{align*}
\dot{y}_1 &= -1000y_1 + y_2 + 998e^{-t}, & y_1(0) &= 2, \\
\dot{y}_2 &= -y_1 - 1000y_2 + 1000e^{-t}, & y_2(0) &= 2, \\
\dot{y}_3 &= -y_3 + y_4 - e^{-t}, & y_3(0) &= 2, \\
\dot{y}_4 &= -y_3 - y_4 + e^{-t}, & y_4(0) &= 2. 
\end{align*} \]  \hspace{1cm} \text{(3.3.3)}

with analytical solution:

\[ \begin{align*}
y_1 &= e^{-1000t} (\cos t + \sin t) + e^{-t}, \\
y_2 &= e^{-1000t} (\cos t - \sin t) + e^{-t}, \\
y_3 &= e^{-t} (\cos t + \sin t) + e^{-t}, \\
y_4 &= e^{-t} (\cos t - \sin t) + e^{-t}. \end{align*} \]  \hspace{1cm} \text{(3.3.4)}

The range of integration is \([0, 1]\).
The eigenvalues are \(-1000\pm i, -1\pm i\).
D.4 SYSTEM IV: Krogh's Example

A nonlinear example suggested by Krogh was taken from Gear (1971c). The system:

\[
\dot{z}_i = -\beta_i z_i + z_i^2 \quad i = 1, 2, 3, 4 \quad z_i(0) = -1
\]  

(D.4.1)

has analytical solution:

\[
z_i = \frac{\beta_i}{1 + \beta_i e^{-\beta_i t}} \quad \sigma_i = -(1 + \beta_i)
\]  

(D.4.2)

using the transformation \( \gamma = u \sigma \)

(D.4.3)

where

\[
u = \begin{bmatrix}
-1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 \\
1 & 1 & -1 & 1 \\
1 & 1 & 1 & -1
\end{bmatrix}
\]

(D.4.4)

we obtain the system:

\[
\dot{y} = -u \beta z u y + u z^2
\]  

(D.4.5)

where

\[
u \beta = \begin{bmatrix}
\beta_1 & & & \\
& \beta_2 & & \\
& & \beta_3 & \\
& & & \beta_4
\end{bmatrix}
\]  

(D.4.6)
Using $B_1 = 1000$, $B_2 = 500$, $B_3 = -10$ and $B_4 = 0.001$ we obtain:

$$C = \frac{1}{8} \begin{bmatrix}
-1790.001 & 1602.993 & 169.999 & 210.001 \\
1602.993 & -1790.001 & -210.001 & -169.999 \\
169.999 & -210.001 & -1790.001 & -1602.993 \\
210.001 & -169.999 & -1602.993 & -1790.001 \\
\end{bmatrix}$$

$$y = \frac{1}{4} \begin{bmatrix}
\nu_1 + \nu_2 + \nu_3 + \nu_4 + 2\nu_1\nu_2 + 2\nu_1\nu_3 + 2\nu_1\nu_4 - 2\nu_2\nu_3 - 2\nu_2\nu_4 - 2\nu_3\nu_4 \\
+ \frac{1}{2} \\
\nu_1 + \nu_2 + \nu_3 + \nu_4 + 2\nu_1\nu_2 + 2\nu_1\nu_3 + 2\nu_1\nu_4 - 2\nu_2\nu_3 - 2\nu_2\nu_4 - 2\nu_3\nu_4 \\
+ \frac{1}{2} \\
\nu_1 + \nu_2 + \nu_3 + \nu_4 + 2\nu_1\nu_2 + 2\nu_1\nu_3 + 2\nu_1\nu_4 - 2\nu_2\nu_3 - 2\nu_2\nu_4 - 2\nu_3\nu_4 \\
+ \frac{1}{2} \\
\end{bmatrix}$$

$$y(0) = \begin{bmatrix}
-1 \\
-1 \\
-1 \\
-1 \\
\end{bmatrix}$$

The range of integration is $[0, 5]$

The eigenvalues are time-variant. Initially they are $2\nu_2 - B_2$, i.e., $-1002, -802, 8$ and $-2.001$. As $t \to \infty$ the eigenvalues approach $B_4$ (if $B_4 < 0$) and $0$ (if $B_4 > 0$).
D.5 SYSTEM V: Chemistry Example

An example which arose in a chemistry problem is taken from Gear (1969a):

\[
\begin{align*}
\dot{y}_1 &= -0.013 y_1 - 1000 y_2 y_3, \quad y_1(0) = 1 \\
\dot{y}_2 &= -2500 y_2 y_3, \quad y_2(0) = 1 \\
\dot{y}_3 &= -0.013 y_3 - 1000 y_1 y_3 - 2500 y_3, \quad y_3(0) = 0
\end{align*}
\]

The range of integration is \([0, 50]\)
The eigenvalues are real and initially are about \(-3500, 0.017\) and 0.

D.6 SYSTEM VI: Polymer Example

A large stiff system of 33 equations representing a polymerization reaction was taken from Hull et al. (1972b):

\[
\begin{align*}
\dot{y}_1 &= k_{12} y_2^2 - k_{21} y_2 y_3 + \frac{2D}{h^2} [y_2(1+\frac{0.01h}{D})y_2] \\
\dot{y}_i &= k_{i,i+11} y_{i+1} + k_{i,i-11} y_i + \frac{D}{h^2} (y_{i-1} - 2y_i + y_{i+1}) i=2,3,...,10 \\
\dot{y}_{11} &= k_{11} y_{22} - k_{21} y_{11} y_{11} + \frac{2D}{h^2} (y_{10} - y_{11}) \\
\dot{y}_i &= 2k_{i,i-1} y_{i-1} + 2k_{i,i+1} y_{i+1} + k_{i,i+11} y_{i+11} \quad i=12,13,...,22 \\
\dot{y}_i &= 2k_{i,i-22} y_{i-22} - 2k_{i,i-2} y_{i-2} - 2k_{i,i+1} y_{i+1} \quad i=23,24,...,33
\end{align*}
\]
where
\[ k_1 = 0.70 \]
\[ k_2 = 1.35 \]
\[ k_3 = 3 \times 10^{-6} \]
\[ z = 1.15 \times 10^{-2} \]
\[ h = 0.00127 \]

Initial conditions \( y_i(0) = 0 \quad i = 1, \ldots, 11 \)
\[ = 9.0 \quad i = 12, 13, \ldots, 22 \]
\[ = 0.0 \quad i = 23, 24, \ldots, 33 \]

The range of integration was \([0, 100]\)

This example represents a physically unstable system in that some of the variables \( y_i, i = 23, 24, \ldots, 33 \) become larger and larger as integration proceeds. Mathematically this means that the eigenvalues corresponding to these variables are positive.

D.7 SYSTEM VII: Stable Polymer Example

The polymer example in 2.2.5 is an unstable system. If \( k_2 \) is changed to \( k_1 \) in equations for \( i = 12-22 \), the system becomes a stable fictitious one.
D.8 SYSTEM VIII: Linear Nonstiff Complex Eigenvalue

Example

The system:

\[ y_1 = -Ay_1 + By_2 \quad y_1(0) = 1 \]
\[ y_2 = -Bv_1 - Ay_2 \quad y_2(0) = 1 \]
\[ y_3 = -Cy_3 + Dy_4 \quad y_3(0) = 1 \]
\[ y_4 = -Dv_3 - Cy_4 \quad y_4(0) = 1 \]  \hspace{1cm} (D.8.1)

is used with analytical solution:

\[ y_1 = e^{-At}(\cos Bt + \sin Bt) \]
\[ y_2 = e^{-At}(\cos Bt - \sin Bt) \]  \hspace{1cm} (D.8.2)
\[ y_3 = e^{-Ct}(\cos Dt + \sin Dt) \]
\[ y_4 = e^{-Ct}(\cos Dt - \sin Dt) \]

We choose \( A = 0.5 \), \( B = 0.25 \), \( C = 0.25 \), \( D = 0.5 \) to yield:

\[ \dot{y}_1 = -0.5y_1 + 0.25y_2 \quad y_1(0)=1 \]
\[ \dot{y}_2 = -0.25y_1 - 0.5y_2 \quad y_2(0)=1 \]  \hspace{1cm} (D.8.3)
\[ \dot{y}_3 = -0.25y_3 + 0.5y_4 \quad y_3(0)=1 \]
\[ \dot{y}_4 = -0.5y_3 - 0.25y_4 \quad y_4(0)=1 \]
with analytical solution:

\[
y_1 = e^{-0.5t}(\cos0.25t + \sin0.25t) \\
y_2 = e^{-0.5t}(\cos0.25t - \sin0.25t) \\
y_3 = e^{-0.25t}(\cos0.5t + \sin0.5t) \\
y_4 = e^{-0.25t}(\cos0.5t - \sin0.5t)
\]

(D.8.4)

The range of integration is \([0, \delta]\)

The eigenvalues are \(-0.5 \pm 0.25i, -0.25 \pm 0.5i\).

D.9 SYSTEM IX: Krogh's Nonstiff Example

The equations of SYSTEM IV: Krogh's example are used but with \(\beta_1 = 0.2, \beta_2 = 0.2, \beta_3 = 0.5, \beta_4 = 0.4\) to obtain:

\[
y = \frac{1}{4}
\begin{bmatrix}
-1.0 & -0.4 & -0.2 & 0.0 \\
-0.4 & -1.0 & 0.0 & 0.2 \\
-0.2 & 0.0 & -1.0 & 0.4 \\
0.0 & 0.2 & 0.4 & -1.0
\end{bmatrix}
y
\]

\[
\begin{bmatrix}
y_1^2 + y_2^2 + y_3^2 + y_4^2 + 2y_1y_2 + 2y_1y_3 + 2y_1y_4 - 2y_2y_3 - 2y_2y_4 - 2y_3y_4 \\
" " " " + 2" - 2" - 2" + 2" + 2" - 2" \\
" " " " - 2" + 2" - 2" + 2" - 2" + 2" \\
" " " " - 2" - 2" + 2" - 2" + 2" + 2"
\end{bmatrix}
\]

(D.9.1)
The range of integration is [0, 10]

Here the step size is constrained only by accuracy not stability. The initial eigenvalues are -2.1, -2.2, -2.3 and -2.4. As \( r \to \infty \), all the eigenvalues approach zero.

3.10 SYSTEM X: Nonlinear Reaction Example

A system representing a nonlinear reaction was taken from Hull et al. (1972b):

\[
\begin{align*}
\dot{z}_1 &= -y_2 \\
\dot{y}_2 &= y_1 - y_2^2 \\
\dot{z}_2 &= y_2^2 \\
\end{align*}
\]

The integration range was [0, 10].
0.11 SYSTEM XI: Gas Absorber Example

A nonlinear system representing the dynamics of a gas absorber was taken from Lapidus and Seinfeld (1971):

\[ \dot{y}_1 = \left\{ \begin{array}{l}
-\left[ 40.6 + 86.7(M_2 + 0.36y_2) \right] y_2 \\
+ 66.7(M_2 + 0.36y_2) y_2 / 2 + 40.8 y_2 / 2 
\end{array} \right. 
\]

\[ \dot{y}_2 = \left\{ \begin{array}{l}
(40.8 y_{e-1} - 40.8 + 86.7(M_2 + 0.36y_2)) y_2 \\
+ 66.7(M_2 + 0.36y_2) y_2 / 2 + 40.8 y_2 / 2 
\end{array} \right. \quad e = 2, 3, 4, 5 \quad (0.11.1) 
\]

\[ \dot{y}_e = \left\{ \begin{array}{l}
(40.8 y_{e-1} - 40.8 + 86.7(M_2 + 0.36y_2)) y_2 \\
+ 66.7(M_2 + 0.36y_2) y_2 / 2 + 40.8 y_2 / 2 
\end{array} \right. \quad e \]

where

\[ y_1 = y_2 = 0 \]

\[ \alpha = (M_2 + 0.16y_2) + 7e \quad e = 1, 2, \ldots, 6 \quad (0.11.2) \]

with initial conditions:

\[ x_0 = \begin{bmatrix} -0.06404026 \\ -0.06199031 \\ -0.06366694 \\ -0.10041628 \\ -0.11306532 \\ -0.12043693 \end{bmatrix} \quad \Lambda = \begin{bmatrix} 0.7307350 \end{bmatrix} \]

The range of integration was \([0, 0.5]\).
APPENDIX E: COMPUTER TIMINGS

In Chapters 5 and 6, various numerical techniques for solving linear algebraic equations and ordinary differential equations were compared by their computer execution times for solving a set of test examples. Only the time to actually solve the problem was measured; time for input-output or problem set-up was not included.

All of the runs were made on the CDC Cyber 73 computer at the University of Western Ontario. The Fortran Extended (FTN), version 4.2 compiler was used with optimization level 1 and Scope 3.4.2 operating system. All of the IMP tests, however, used an object deck supplied by Control Data from FTN version 4.0, optimization level 2 and Scope 3.4 operating system. Some of the longer non-IMP runs were made with both optimization levels 1 and 2, but the difference in execution time was very slight (about 1%).

A rigorous comparison of numerical techniques for solving o.d.e.s should do more than just measure execution time. Hull et al. (1972a) also measure the number of derivative evaluations, the time required to evaluate derivatives and the remaining execution time called the overhead. Krogh (1973) discusses in depth, the aspects of
numerically testing a subroutine for the numerical solutions of O.D.E.s.

Execution time itself is not a perfect criteria for comparison as one program may be faster than another on one computer but slower with a different machine. This arises because of different relative computer times for the various arithmetic operations: addition, multiplication and division. For example on the CYBER 73 or CDC 6400, the times for multiplication and division are identical, whereas on other computers, division can be three or four times slower. For the CYBER 73, the ratio of clock pulses for addition, multiplication and division is 11:57:57 while it is 4:5:20 on the Cyber 76 (Brandon, 1974e). For the CYBER 73, this means that an algorithm requiring many divisions rather than multiplications will not appear any slower than another algorithm using few divisions and many multiplications. For example, the IMP algorithm uses fewer divisions than the TRGB algorithm, but this is not reflected in the CYBER 73 execution times. Ideally one should not measure execution time, but the numbers of the different arithmetic operations; these could then be converted into execution times for any desired computer. However, this is very inconvenient.

Another problem is the reproducibility of execution times particularly those below 0.5 seconds. Any program
taking less than 0.5 seconds was run several times to get an average. One program took 0.000, 0.019 and 0.038 on different runs. This corresponds to 0, 1 or 2 pulses of the clock almost on a random basis. If it was convenient, the portion of the program being timed was put in a DO loop and repeated several times to get a larger time which could then be divided by the number of repetitions.

If many other jobs are running in the computer at the same time, it is possible for a program to be taken from core and placed in a waiting queue by another job of higher priority. This may have a very slight effect on the execution time. Ideally the execution times should be measured with no other jobs in the system. This is not very practical; but as much as possible, the times were measured during periods of low usage.
APPENDIX F: THE DYNYSYS DATA SET

The data required for a simulation is inputted to the DYNYSYS program through the data set. This is described fully in the DYNYSYS Manual (Bobrow, Johnson and Ponton, 1971). We will give here, a brief description of a data set using Example #1 in Chapter 8 as an illustration.

Figure 8.2 contains the dynamic information flow diagram for the example and the data set is at the end of Figure 8.4.

There are four basic sections in a data set: the simulation data, the equipment data, the stream data and the physical properties data. All data is stored in 12 column fields. The first 12 columns contain only alphanumeric data and the numeric data goes into 5 F12 fields starting in column 13.
F.1 Simulation Data

The simulation data begins with the data word "BEGIN". All cards before this are comments and are ignored by DYNSYS.

For example #1

\begin{tabular}{|l|l|}
\hline
IN/OUT & 3.0  \\
\hline
HMAX  & 0.01  \\
\hline
TIME  & 3.0  \\
\hline
LIBRARY & 1.0  \\
\hline
DELAY & 9.0  \\
\hline
\end{tabular}

- **IN/OUT 3.0** sets the maximum number of streams entering or leaving any module to three (default = 5)
- **HMAX 0.01** sets the maximum integration set size to 0.01 (default = 0.05)
- **TIME 3.0** sets the time of simulation to 3.0 (default = 10.0)
- **LIBRARY 1.0** means that one new module will be defined immediately after this card. Most common modules such as **CONT1**, **STIR1** and **VALV1** are already in a library of module names stored inside DYNSYS.
- **DELAY 9.0** DELAY is the new module being defined. It will be accessed as SUBROUTINE TYPE9
Other possible simulation data words are:

<table>
<thead>
<tr>
<th>Word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPS</td>
<td>sets the number of components to X (default = 1)</td>
</tr>
<tr>
<td>DELTAT</td>
<td>sets the initial integration step size to X (default = 0.00001)</td>
</tr>
<tr>
<td>HMIN</td>
<td>sets the minimum integration step size to X (default = 0.000001)</td>
</tr>
<tr>
<td>TOLERANCE</td>
<td>sets the error tolerance for the integration to X (default = 0.001)</td>
</tr>
<tr>
<td>ORDER</td>
<td>sets the maximum permissible order of integration to X (default = 6)</td>
</tr>
<tr>
<td>NONSTIFF</td>
<td>causes the nonstiff option to be used (default = stiff option)</td>
</tr>
<tr>
<td>MINPivot</td>
<td>sets the minimum permissible pivot value for TRGB-TRGB2 to X</td>
</tr>
<tr>
<td>PRINTING</td>
<td>causes output to be printed every X time steps (default = 1)</td>
</tr>
<tr>
<td>LINEPLOT</td>
<td>causes output to be printed as a line-plot (in this case additional data must be given after physical properties data)</td>
</tr>
<tr>
<td>GRAPH</td>
<td>causes output to be printed as graph or graphs on CALCOMP plotter as well as Lineplot. LINEPLOT must also be used.</td>
</tr>
</tbody>
</table>
FUNCTION \(X\) causes \(X\) vectors of stream information to be read in as an input function. Stream data follows immediately.

The data word "PROCESS" signifies the beginning of the equipment data and thus the end of the simulation data.

F.2 Equipment Data

The data word "PROCESS" signifies the beginning of the equipment data. Here data for the various modules is written in their order of execution.

\[\text{CONT1} \quad 1.0\] the first module to be executed is \[\text{CONT1}\], referred to as module #1 in the dynamic information flow diagram (Figure 8.2).

After this, the entering and leaving stream numbers are given. A positive number indicates an input stream while a negative number indicates an output stream.

\[3.0 \quad -4.0\] stream 3.0 enters \text{CONT1} and stream 4 leaves it. This can be verified in Figure 8.2.
The following card gives the equipment parameters for the module. This is the data which describes the individual units.

3. 1500. 1000. 2. 3.

For example, the set point for the controller is the third parameter 1000. Up to five parameters may be given. If more are required, the data word "EXTRA" X must follow and the X extra parameters appear on the following data cards. The data set in Figure 8.10 gives an example of this. We continue in this way for all modules. The data word "END" signifies the end of the equipment data.

F.3 Stream Data

The material flows between equipment are represented by material streams. Information about each stream is stored in the following vector.

position 1 - stream number
2 - flag
3 - total mass flow
4 - temperature
5 - pressure
6 - mass fraction of component 1
7 - mass fraction of component 2
    etc.
The stream flag serves the dual purpose of identifying the stream type (flags with absolute value \(< 10\) are material flows, those \(\geq 10\) are information flows) and, if negative, suppresses printing of the stream vector in the output.

The user may choose his own system of units as long as he is consistent throughout. Usually English units are employed.

STREAMS 5. means that initial information data will be given for five streams.

EXPLICIT means that the data will be given explicitly.

The following cards give the data. For the first stream

1. 1. 1000. 100. 14.7

the stream number is 1. The stream is a material flow (the flag is less than 10). The flow rate, temperature and pressure are 1000., 100. and 14.7 respectively. There is only one component so its mass fraction is 1.0.

Data is given for five streams. The data word "END" signifies the end of the stream data.
F.4 Physical Properties Data

PROPERTIES -1. means that the physical properties of water are assumed.

The user may provide his own data if desired. See the DYNSYS Manual for details.

The data word "END" signifies the end of the physical properties data and in this case the end of the data set.
APPENDIX G: TIME DELAY MODULE

A module for approximating a fixed or variable time delay is available. Time delays may occur in pipelines between equipments or in instruments such as values and controllers.

Instrument lag is fixed whereas a pipeline delay is usually variable depending on the flow rate of the fluid passing through it. The variable delay is equal to the volume of the delaying apparatus divided by the flow rate through it.

The bucket brigade approach is used to simulate the stream delay. The past times and stream information are stored in vectors and the stream output from the delay is interpolated from these values. For a lengthy delay, the storage space is prohibitive, but later versions of the module may remedy this.

The delayed streams are stored in SX(I,J,K) where I represents the vector of past times, J represents different delays, and K represents the value of the stream vector.
Not all of the stream is necessarily delayed; a fraction of it may be bypassed. Nor is the entire stream vector necessarily delayed; the user may choose the first variable to be delayed and only this variable and all others after it will be delayed. The delay may also simulate a pump by setting the inlet flow rate equal to the output flow rate.

There are five equipment parameters:

1. **TLAG** - if the time delay is fixed, TLAG is the value of the delay. If the delay is variable, TLAG is the negative of the volume of the delaying apparatus or pipeline.

2. **BYP** - fraction of stream not delayed, i.e., bypassed.

3. **NV** - number of storage spaces used in the delay vector. An error message is given if NV is not large enough.

4. **N1** - first variable to be delayed
   - if \( N1 = 3 \), the entire stream is delayed
   - if \( N1 = 0.5 \), only the mass fractions are delayed.
(5) FLAG - if FLAG is greater than zero, the module simulates a pump; if FLAG = 0, the delay is normal.

A listing of the time delay module is given in Figure G.1.
FIGURE G.1: LISTING OF TIME DELAY MODULE

SUBROUTINE TYPE9

THIS MODULE REPRESENTS A FIXED OR VARIABLE TIME DELAY

EQUIPMENT PARAMETERS
1 - TFLAG = LENGTH OF TIME DELAY; IF TIME DELAY IS FIXED
- NEGATIVE OF VOLUME OF DELAYING EQUIPMENT OR PIPELINE;
- IF TIME DELAY IS VARIABLE
2 - BYP = FRACTION OF STREAM NOT DELAYED; I.E. BYPASSED
3 - NV = NUMBER OF STORAGE SPACES USED IN DELAY VECTOR
4 - N1 = FIRST VARIABLE TO BE DELAYED; 3 = TOTAL FLOW; 4 = TEMP., ETC
5 - FLAG = FLAG, GT, 0; OUTPUT FLOW CONTROL (PUMP)
     FLAG, EQ, 0; NORMAL DELAY

COMMON /MAT/ MP(45), EP(45), S(2,5,7), EX(1)
COMMON /UNIT/ IN, NMP
COMMON /CON/ IO, NCOMP, NC, H, NE, NS, NPR, NPOL, TMAX, IORDER, NGRAPH
COMMON /GEAR2/ EPS, TIME, KFLAG, JSTART, NBVMAX, ICONV, NODP, ISTIFF
COMMON /LAG/ ND, NSW
DIMENSION MC(10), SX(100, 18)
INTEGER OUT
DATA MC/10*1/

SX IS THE DELAY MATRIX
1ST VECTOR = DELAY AT DIFFERENT VALUES OF TIME
2ND VECTOR = DIFFERENT DELAYS
3RD VECTOR = VALUES OF STREAM VECTOR

IN = MP(IM, 3)
OUT = MP(IM, 4)

FIXED TIME DELAY
TLAG = EP(IM, 1)

VARIABLE TIME DELAY
IF (TLAG, LT, 0.0) TLAG = ABS(TLAG) / S(2, IM, 1)
BYP = EP(IM, 2)
NV = EP(IM, 3)
N1 = EP(IM, 4)

FLAG = 1; OUTPUT FLOW CONTROL = 0; NORMAL
IF (EP(IM, 5), GT, 0.0) S(1, IM, 2) = S(1, OUT, 2)

FOR FIRST PREDICTOR STEP, SET ALL INITIAL INFORMATION REQUIRED
BY SX = MATRIX; FOR ALL FOLLOWING PREDICTOR STEPS SKIP THE DELAY
IF (IG, LT, 2) GO TO 3
IF (JSTART .NE. 0) RETURN
ND=ND+1
SX(1:ND+1)=0.0
C
C SET INPUT STREAM VALUES IN 1ST VECTOR OF SX-MATRIX
C
DO 2 K=N1,NC4
SX(1:ND+K)=S(2:IN+K)
C
C SET THE REST OF VALUES IN STORES TO EXIT VALUE
C
DO 1 I=2:NV
SX(1:ND+K)=S(1:OUT+K)
C
C FILL TIME STORES INITIALLY WITH =1.0 VALUES
C
SX(1:ND+1)=-1.
1 CONTINUE
2 CONTINUE
RETURN
C
ND=ND+1
MC(ND)=MC(ND)+1
C
IF REPEAT THEN PICKED UP VALUES OF TIME STORED IN SX(:,1)
C
IF (NSW.EQ.0) GO TO 5
MC(ND)=MC(ND)+1
DO 4 I=2:NV
SX(I:ND+I)=SX(I,ND+8)
4 CONTINUE
C
NORMAL OPERATION: STORE A COPY OF TIME VECTOR FOR FUTURE REPEAT
THEN INCREMENT TIME VALUES AS TIME PROGRESS
C
5 MD=MC(ND)
IF (MD.GT.NV) MD=NV
DO 6 I=2:NV
SX(I:ND+I)=SX(I,ND+1)
6 CONTINUE
7 IF (MD.EQ.2) GO TO 8
SX(MD+ND+1)=SX(MD-1+ND+1)+H
MD=MD+1
GO TO 7
8 SX(2:ND+1)=H
C
C SHIFT VALUES IN THE STORES RIGHT STARTING WITH RIGHTMOST
C
IF (NSW.EQ.1) GO TO 11
NV1=NV-1
DO 10 K=N1,NC4
DO 9 I=1:NV1
II=NV+I+1
L=II-1
SX(II+ND*3)=SX(L+ND*3)
CONTINUE
CONTINUE
TRANSFER VALUE OF INLET TO 1ST STORAGE OF DELAY VECTOR
DO 12 K=N1,NC4
SX(1+ND*3)=S(1,IN*K)
CONTINUE
COMPARE STORED TIME VALUES WITH TIME LAB
DO 13 I=1,NV1
IF (SX(I+ND*3)-TLAG) 13,14,17
CONTINUE
IF NONE IS GREATER THAN TLAG, EXIT THE LAST VALUE
IF TIME VALUE EQUALS T AG, EXIT CORRESPONDING VALUES
I=NV
IF (SX(NV+ND*3),EQ.,-1.0) GO TO 14
PRINT 19
STOP
DO 15 K=N1,NC4
S(1.OUT*K)=SX(I+ND*K)+(1.,BYP)•S(1,IN*K)•BYP
CONTINUE
RETURN
IF SX(I) .GT. TLAG, CHECK IF SX(I+1) .GT. TLAG
IF SO, SET THE TIME AT SX(I) TO -1.
SX(I+ND*3)=-1.
I=I+1
L=I-1
KEEP TESTING UNTIL ONLY ONE IS LEFT
IF (SX(L+ND*3),GE.,TLAG) GO TO 16
INTERPOLATE FOR EXIT VALUE
A=TLAG-SX(L+ND*3)
B=SX(I+ND*3)-SX(L+ND*3)
DO 18 K=N1,NC4
U=SX(L+ND*K)+(A/B)•(SX(I+ND*K)-SX(L+ND*K))
S(1.OUT*K)=(1.,BYP)•U•BYP•S(1,IN*K)
CONTINUE
RETURN
FORMAT (20H ERROR IN TIME DELAYS/23H NV MUST BE MADE LARGER)
APPENDIX H: WILLIAMS-OTTO PLANT LISTINGS

Figure H.1 gives a listing of all the modules and the data set used in the Williams-Otto plant simulation. The distillation column modules are not included. The version of the executive used allowed for 10 equipment parameters.
FIGURE H.11 LISTING OF MODULES AND DATA SET FOR WILLIAMS-OTTO PLANT

SUBROUTINE TYPE10

SURROUTINE CSTR1

THIS MODULE REPRESENTS THE CONTINUOUS STIRRED TANK REACTOR
IN THE WILLIAMS-OTTO PLANT

MODULE IS STIFF

EQUIPMENT PARAMETERS

1 - VR  =  INITIAL HOLDUP IN REACTOR (LB)
2 - VW  =  HOLDUP IN COOLING COIL (LB)
3 - HWAM = OVERALL COOLING COIL HEAT TRANSFER COEFFICIENT
          * EFFECTIVE HEAT TRANSFER AREA (BTU/(DEGREE F*HR))
4 - HSAS = OVERALL STEAM COIL HEAT TRANSFER COEFFICIENT
          * EFFECTIVE HEAT TRANSFER AREA (BTU/(DEGREE F*HR))
5 - XLHC = LATENT HEAT OF CONDENSATION OF ENTERING STEAM (BTU/LB)

STREAMS

1 - INA  =  A FEED STREAM
2 - INB  =  B FEED STREAM
3 - INL  =  L RECYCLE STREAM
4 - INC  =  COOLING COIL INPUT STREAM
5 - INS  =  STEAM COIL INPUT STREAM
6 - IOUT =  PRODUCT STREAM
7 - IOUTC = COOLING COIL OUTPUT STREAM
8 - IOUTS = STEAM COIL OUTPUT STREAM
9 - ISIGL = LEVEL CONTROL SIGNAL
10 - ISIGB = B FLOW CONTROL SIGNAL (RATIO FA/FB)

NOMENCLATURE

FA  =  A FEED STREAM FLOW RATE (LB/HR)
FB  =  B FEED STREAM FLOW RATE (LB/HR)
FL  =  L RECYCLE STREAM FLOW RATE (LB/HR)
FR  =  PRODUCT STREAM FLOW RATE (LB/HR)
FW  =  COOLING COIL FLOW RATE (LB/HR)
TA  =  A FEED STREAM TEMPERATURE (DEGREES F)
TB  =  B FEED STREAM TEMPERATURE (DEGREES F)
TL  =  L RECYCLE STREAM TEMPERATURE (DEGREES F)
TINC = INLET COOLING COIL TEMPERATURE (DEGREES F)
TS  =  STEAM COIL TEMPERATURE (DEGREES F)
SFLW = STEAM FLOW RATE (LB/HR)
CPR  =  HEAT CAPACITY OF REACTOR MIXTURE (BTU/(LB*DEGREE F))
       (ASSUMED SAME FOR ALL COMPONENTS)
CPW  =  HEAT CAPACITY OF COOLANT (BTU/(LB*DEGREE F))
COMMON /MAT/MP(35*13),EP(35*10),S(2*45*11),EX(1)
COMMON /CON/ IG,NCOMP,NC5,H,NE,NS,NPR,NPOL,TMAX,IORDER,NGRAPH
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBMAX,ICONV,NOPPT,ISTIFF
COMMON /PTAB/ 16FLAG,PP(10*20)
COMMON /UNIT/ IM,NMP
COMMON /MODULE/ IDERY,ITER,ITRI,NZERO
COMMON /ROW/IROWN(1000)
COMMON /COLUMN/JCOL(1000)
COMMON /JACOB/XJACOB(1000)
COMMON /RETN/ K1,K2,K3,H1,H2,H3,B1,B2,B3
DIMENSION IROWS(33),JCOLS(33)
DIMENSION Y(9),DERY(9)
REAL K1,K2,K3
DATA IROWS/3*1,4*2,5*3,4*4*4*5,4*6,6*7,1*8,9/
DATA JCOLS/1*1,2*1,2*3,3*1,2*3,6*2,2*3,6*2,3*3,6*7,1*2,3,16,7*8,7*8,9/
DATA CPR/CPW/0.4,1.0/

CALCULATE MODULE PARAMETERS

IF(JSTART .NE. 0 .OR. IG .EQ. 1) GO TO 2
     VR=EP(IM+1)
     VV=EP(IM+2)
     HW=EP(IM+3)
     HSAS=EP(IM+4)
     XLHC=EP(IM+5)
     INA=MP(IM+3)
     INB=MP(IM+4)
     INL=MP(IM+5)
     INC=MP(IM+6)
     INS=MP(IM+7)
     IOUT=IABS(MP(IM+8))
     IOUTC=IABS(MP(IM+9))
     IOUTS=IABS(MP(IM+10))
     ISIGL=IABS(MP(IM+11))
     ISIGB=IABS(MP(IM+12))
     DO 31 I=1,6
         31 Y(I)=S(2,IOUT,I+5)
        Y(7)=S(2,IOUTC,S)
        Y(8)=S(2,IOUTS,S)
        Y(9)=VR
2 CONTINUE
C USE NEWTON-RAPHSON ITERATION WITH STIFF OPTION
   ITER=1
   FA=S(IG,INA+3)
   FB=S(IG,INB+3)
   FL=S(IG,INL+3)
   FR=S(IG,IOUT,3)
   FW=S(IG,INC+3)
   TA=S(IG,INA+4)
   TB=S(IG,INB+4)
   TL=S(IG,INL+4)
TINC=S(IG*INC*4)

CALCULATE JACOBIAN MATRIX ON CORRECTOR PASS

IF(IG.EQ.2) GO TO 1
NZERO=33
DO 81 I=1,33
IROW(I)=IROWS(I),
JCOL(I)=JCOLS(I)
81

CALL REACT TO GET REACTION RATE COEFFICIENTS AND HEATS OF REACTION

TEMP=Y(7)+459.69
CALL REACT(TEMP)
T1=K1*Y(1)+Y(2)*B1/(TEMP**2)
T2=K2*Y(2)+Y(3)*B2/(TEMP**2)
T3=K3*Y(3)+Y(6)*B3/(TEMP**2)
XJACOB(1)=FR/VR-K1*Y(2)
XJACOB(2)=K1*Y(1)
XJACOB(3)=T1
XJACOB(4)=K1*Y(2)
XJACOB(5)=FR/VR-K1*Y(1)-K2*Y(3)
XJACOB(6)=K2*Y(2)
XJACOB(7)=T1-T2
XJACOB(8)=Z0*K1*Y(2)
XJACOB(9)=2.0*K1*Y(1)-2.0*K2*Y(3)
XJACOB(10)=FR/VR-2.0*K2*Y(2)-K3*Y(6)
XJACOB(11)=K3*Y(3)
XJACOB(12)=2.0*T1-2.0*T2-T3
XJACOB(13)=2.0*K2*Y(3)
XJACOB(14)=2.0*K2*Y(2)
XJACOB(15)=FR/VR
XJACOB(16)=2.0*T2
XJACOB(17)=1.5*K3*Y(6)
XJACOB(18)=FR/VR
XJACOB(19)=1.5*K3*Y(3)
XJACOB(20)=1.5*T3
XJACOB(21)=K2*Y(3)
XJACOB(22)=K2*Y(2)-0.5*K3*Y(6)
XJACOB(23)=0.5*K3*Y(3)-FR/VR
XJACOB(24)=T2-0.5*T3
XJACOB(25)=2.0*K1*Y(2)*H1/CPR
XJACOB(26)=-(2.0*K1*Y(1)*H1+3.0*K2*Y(3)*H2)/CPR
XJACOB(27)=-(3.0*K2*Y(2)*H2+1.5*K3*Y(6)*H3)/CPR
XJACOB(28)=1.5*K3*Y(3)*H3/CPR
XJACOB(29)=-(2.0*T1+H1+3.0*T2+H2+1.5*T3+H3)/CPR
1=(HWAW+HSAS)/(VR*CPR)-(FL*FA+FB)/VR
XJACOB(30)=HWAW/(VR*CPR)
XJACOB(31)=HWAW/(VW*CPW)
XJACOB(32)=(HWAW*FW*CPW)/(VW*CPW)
XJACOB(33)=0.0
GO TO 12
1 CONTINUE

CALCULATE DERIVATIVES

CALL REACT TO GET REACTION RATE COEFFICIENTS AND HEATS OF REACTION
 TEMP=Y(7)*459.69
 CALL REACT(TMP)

12 CONTINUE
 S1=K1*Y(1)*Y(2)
 S2=K2*Y(2)*Y(3)
 S3=K3*Y(3)*Y(6)
 TS=S(IG,INS=4)

C IF STEAM FLOW IS ZERO, THERE IS NO HEAT TRANSFER
 IF(S(IG,INS=3).LT.0.001) TS=Y(7)

C MASS BALANCE
 DERY(1)=1.0/VR*(FA*FL*S(IG,INL=6)-FR*Y(1))-S1

C MASS BALANCE
 DERY(2)=1.0/VR*(FB*FL*S(IG,INL=7)-FR*Y(2))-S1-S2

C MASS BALANCE
 DERY(3)=1.0/VR*(FL*S(IG,INL=8)-FR*Y(3))*2.0*S1-2.0*S2-S3

C MASS BALANCE
 DERY(4)=1.0/VR*(FL*S(IG,INL=9)-FR*Y(4))*2.0*S2

C MASS BALANCE
 DERY(5)=1.0/VR*(FL*S(IG,INL=10)-FR*Y(5))*1.5*S3

C MASS BALANCE
 DERY(6)=1.0/VR*(FL*S(IG,INL=11)-FR*Y(6))-S2=0.5*S3

C TOTAL HEAT BALANCE
 DERY(7)=1.0/(VR*CPR)*(-2.0*S1*H1*VR-3.0*S2*H2*VR-1.5*S3*H3*VR
 1=HWA*+Y(7)-Y(8)+HSAS*(TS-Y(7))-FL*CPR*Y(7)-TL-FA*CPR*(Y(7)-TA)
 2=FB*CPR*(Y(7)-TB))

C HEAT BALANCE FOR COOLING COIL
 DERY(8)=1.0/(VA*CPR)*(HWA*(Y(7)-Y(8))+FW*CPW*(TINC-Y(8)))

C TOTAL MASS BALANCE
 DERY(9)=FA*FB*FL=FR
 CALL DIFSUB(9,Y,DERY)
 IF(IDER=NE,0) 60 TO 1

CALCULATE STREAM OUTPUT

PRODUCT STREAM

NORMALIZE MASS FRACTIONS
 SUM=0.0
 DO 55 I=1,6
 55 SUM=SUM+Y(I)
 DO 6 I=1,6
  Y(I)=Y(I)/SUM

S(1.IOU)T=5*I
 S(1.IOU)T+5)*Y(I)
 S(1.IOU)T+6)*Y(7)

COOLING COIL
 S(1.IOU)T+3)*S(1.IOU)T+3)
 S(1.IOU)T+4)*Y(8)
C STEAM COIL
IF(S(IG,INS,3) .LT. 0.001) GO TO 56
SFLOW=HSAS/XLHC*(S(IG,INS,4)-Y(7))
S(1*INS,3)=SFLOW
56 S(1*OUTS,3)=S(1*INS,3)
S(1*OUTS,4)=S(1*INS,4)
C CONTROL SIGNALS
S(1*ISIGL,3)=Y(9)
S(1*ISIGB,3)=FA/FB
RETURN
END
SUBROUTINE REACT(T)

THIS SUBROUTINE SUPPLIES REACTION RATE COEFFICIENTS AND HEATS OF REACTION OF THE WILLIAMS-OTTO PLANT REACTION SCHEME BELOW

\[
\begin{align*}
A + B &= C \\
C + B &= P + E \\
P + C &= G
\end{align*}
\]

T IS DEFINED IN DEGREES RANKINE

COMMON /RETNR/ K1,K2,K3,H1,H2,H3,B1,B2,B3
REAL K1,K2,K3

CONSTANTS OF ARRHENIUS EQUATION \( K = A \cdot \exp(-B/T) \)

DATA A1,A2,A3/5.9755E+09,2.5962E+12,9.6283E+15/
DATA B1,B2,B3/12000,0,15000,0,20000,0/

ARRHENIUS REACTION COEFFICIENTS

IF (T.LT.500.0) T=500.0
IF (T.GT.1000.0) T=1000.0
K1=A1*EXP(-B1/T)
K2=A2*EXP(-B2/T)
K3=A3*EXP(-B3/T)

HEATS OF REACTION

H1=-125.0
H2=-50.0
H3=-143.0
RETURN
END
SUBROUTINE TYPE11

SUBROUTINE EXCH1

THIS MODULE REPRESENTS A SHELL AND TUBE HEAT EXCHANGER.
BOTH THE SHELL AND TUBE SIDE ARE MODELLED AS WELL-STIRRED TANKS.
MODULE IS NONSTIFF

EQUPMENT PARAMETERS

1 - VHT = TUBE SIDE MASS HOLDUP (LB)
2 - VHS = SHELL SIDE (COOLANT) MASS HOLDUP (LB H2O)
3 - AH = AREA OF HEAT TRANSFER (FT**2)
4 - HH = OVERALL HEAT TRANSFER COEFFICIENT
   (BTU/(DEGREE F*FT**2*HR))
5 - CPW = COOLANT HEAT CAPACITY (BTU/(LB*DEGREE F))

STREAMS

1 - INT = INLET TUBE SIDE
2 - INS = INLET SHELL SIDE
3 - IOU_T = OUTLET TUBE SIDE
4 - IOU_S = OUTLET SHELL SIDE

NOMENCLATURE

FT = TUBE SIDE FLOW RATE (LB/HR)
FS = SHELL SIDE FLOW RATE (LB/HR)
TINT = INLET TUBE SIDE TEMPERATURE (DEGREES F)
TINS = INLET SHELL SIDE TEMPERATURE (DEGREES F)
CPR = TUBE SIDE HEAT CAPACITY (BTU/(LB*DEGREE F))
TLM = LOG MEAN TEMPERATURE

COMMON/MAT,MP(35,13),EP(35,10),S(2,45,11),EX(1)
COMMON /CON/ IG,NCOMP,NC5,H,NE,NS,NPR,NPOL,THAX,IORDER,NGRAPH
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBVMAX,ICNV,NOPPT,ISTIFF
COMMON /PTAB/ IGFLAG,PP(10,20)
COMMON /UNIT/ IM,NMP
COMMON /MODULE/ IDERY,ITER,ITRI,NZERO
DIMENSION Y(8),DERY(8)
DATA CPR,0.4/

CALCULATE MODULE PARAMETERS

IF(JSTART,NE.0.OR.IG.EQ.1) GO TO 2
VHT=EP(IM+1)
VHS=EP(IM+2)
AH=EP(IM+3)
HH=EP(IM+4)
CPW=EP(IM+5)
INT=MP(IM+3)
INS=MP(IM+4)
IOU T T = I A B S ( M P ( I M , 5 ) )
IOU TS = I A B S ( M P ( I M , 6 ) )
Y ( 1 ) = S ( I G , IOU TT , 4 )
Y ( 2 ) = S ( I G , IOU TS , 4 )
DO 9 I = 1 , N C O M P
9 Y ( I + 2 ) = S ( I G , IOU TT , I + 5 )
2 C O N T I N U E
C U S E D I R E C T I T E R A T I O N I F S T I F F O P T I O N I S U S E D
I T E R = 0
F T = S ( I G , I N T , 3 )
F S = S ( I G , I N S , 3 )
T I N T = S ( I G , I N T , 4 )
T I N S = S ( I G , I N S , 4 )
1 C O N T I N U E
C C A L C U L A T E D E R I V A T I V E S
C C A L C U L A T E L O G M E A N T E M P E R A T U R E
T L M = ( ( T I N T - Y ( 2 ) ) / ( Y ( 1 ) - T I N S ) ) / A L O G ( ( T I N T - Y ( 2 ) ) / ( Y ( 1 ) - T I N S ) )
C H E A T B A L A N C E ( T U B E S I D E )
D E R Y ( 1 ) = 1 . 0 / ( V H T * C P R ) * ( F T * C P R * ( T I N T - Y ( 1 ) ) - M H * A H * T L M )
C H E A T B A L A N C E ( S H E L L S I D E )
D E R Y ( 2 ) = 1 . 0 / ( V H S * C P W ) * ( F S * C P W * ( T I N S - Y ( 2 ) ) + M H * A H * T L M )
C C O M P O N E N T M A S S B A L A N C E S ( T U B E S I D E )
D O 1 0 I = 1 , N C O M P
1 0 D E R Y ( I + 2 ) = F T / V H T * ( S ( I G , I N T , I + 5 ) - Y ( I + 2 ) )
N = 2 + N C O M P
C A L L D I F S U B ( N , Y , D E R Y )
I F ( I D E R Y , N E , 0 ) G O T O 1
C C A L C U L A T E S T R E A M O U T P U T
S ( 1 , I O U T T , 4 ) = Y ( 1 )
S ( 1 , I O U T S , 4 ) = Y ( 2 )
S ( 1 , I O U T S , 3 ) = S ( 1 , I N S , 3 )
S U M = 0 . 0
D O 1 5 I = 3 , N
1 5 S U M = S U M + Y ( I )
C P U T N O R M A L I Z E D M A S S F R A C T I O N S I N T O O U T L E T T U B E S I D E S T R E A M
D O 1 6 I = 3 , N
Y ( I ) = Y ( I ) / S U M
1 6 S ( 1 , I O U T T , I + 3 ) = Y ( I )
R E T U R N
E N D
SUBROUTINE TYPE12
SUBROUTINE 'DCAN1'

THIS MODULE REPRESENTS A DECANTER
ONE COMPONENT IS SEPARATED BY GRAVITY FROM THE ENTERING STREAM
BOTH TOP AND BOTTOM LAYERS ARE MODELLED AS WELL-STIRRED TANKS
FUNCTION XS GIVES WEIGHT FRACTION OF SEPARATED COMPONENT ENTERING
TOP LAYER OF DECANTER AS FUNCTION OF TEMPERATURE
CONTROL SIGNALS FOR THE HOLDUP IN TOP AND BOTTOM LAYERS ARE
EACH SENT TO CONTROLLERS
MODULE IS NONSTIFF

EQUIPMENT PARAMETERS

1 = MUPT = INITIAL HOLDUP IN TOP LAYER (LB)
2 = MUPB = INITIAL HOLDUP IN BOTTOM LAYER (LB)
3 = ISEP = NUMBER OF COMPONENT TO BE SEPARATED FROM ENTERING STREAM STREAMS

1 = IN = INLET
2 = IOUTT = OUTLET FROM TOP OF DECANTER
3 = IOUTB = OUTLET FROM BOTTOM OF DECANTER
4 = ISIGT = LEVEL CONTROL SIGNAL (TOP)
5 = ISIGB = LEVEL CONTROL SIGNAL (BOTTOM)

NOMENCLATURE

FIN = TOTAL INLET FLOW (LB/HR)
FIN = INLET FLOW ENTERING TOP LAYER (LB/HR)
FINB = INLET FLOW ENTERING BOTTOM LAYER (LB/HR)
FOUT = TOTAL OUTLET FLOW (LB/HR)
FOUTT = OUTLET FLOW FROM TOP LAYER (LB/HR)
FOUTB = OUTLET FLOW FROM BOTTOM LAYER (LB/HR)
FT = MASS FRACTIONS ENTERING TOP LAYER
FEC = MASS FRACTIONS IN TOP LAYER
WFG = MASS FRACTION OF SEPARATED COMPONENT ENTERING TOP LAYER
TEMP = DECANTER TEMPERATURE (DEGREES F)

COMMON/ HAT/ HP (35×13) , EP (35×10) , S (2×45×1) , EX (1)
COMMON/ CON/ G , NCOMP , NC , H , NE , NS , NP , INPOL , Tmax , IORDER , NGRAAP
COMMON/ GEAR2/ EPS , TIME , KLAG , JSTART , NBYMAX , ICONV , NCONP , ISTIFF
COMMON/ PTAB/ IFLAG , PP (10×20)
COMMON/ UNIT/ IM , HMP
COMMON/ MODUL/ IDER , ITER , ITRI , NZERO
DIMENSION Y (9) , DERY (9) , FT (6) , FEC (6)

CALCULATE MODULE PARAMETERS

IF ( JSTART .NE. 0 .OR. 18 .EQ. 1) GO TO 2
MUPT = EP ( IM+1 )
HUPB=EP(I+2)
ISEP=EP(I+3)
IN=MP(I+4)
IOUTT=IABS(MP(I+5))
IOUTB=IABS(MP(I+6))
ISIGT=IABS(MP(I+7))
ISIGB=IABS(MP(I+7))
Y(1)=HUPB
Y(2)=S(I+IOUTT,4)
N=NCOMP+2
DO 4 I=3,N
4 Y(I)=HUPB*S(I+IOUTB,3)
Y(N+1)=HUPB
2 CONTINUE
USE DIRECT ITERATION IF STIFF OPTION IS USED
ITER=0
1 CONTINUE

CALCULATE DERIVATIVES

FIN=S(I+IN,3)
FOUTT=S(I+IOUTT,3)
FOUTB=S(I+IOUTB,3)

TOTAL MASS BALANCE
DERY(1)=FIN=FOUTT=FOUTB

TOTAL HEAT BALANCE
DERY(2)=1.0/Y(1)*FIN*S(I+IN,4)-(FOUTT+FOUTB)*Y(2)
TEMP=Y(2)

FUNCTION X6 DETERMINES MASS FRACTION OF SEPARATED COMPONENT
ENTERING TOP LAYER
WFG=X6(TEMP)

WFG CANNOT BE GREATER THAN ENTERING MASS FRACTION
IF(WFG,GT,S(I+IN,ISEP+5)) WFG=S(I+IN,ISEP+5)
FIN=FIN*(1.0-S(I+IN,ISEP+5))/(1.0-WFG)
FINB=FIN*FIN

101 FT(1)=FIN=S(I+IN,1+5)/FIN
FT(ISEP)=WFG
SUM=0.0
NF=NCOMP+2
DO 3 I=3,NF
3 SUM=SUM+Y(I)
DO 14 I=1,NCOMP
14 FEC(I)=Y(I+2)/SUM

COMPONENT MASS BALANCES FOR TOP LAYER
DO 20 I=3,NF
20 DERY(I)=FIN*FT(I-2)-FOUTT*FEC(I-2)

MASS BALANCE FOR BOTTOM LAYER
N=NCOMP+3
DERY(N)=FINB=FOUTB
CALL/DIFSUB(N,Y,DERY)
IF(IDERY,NE,0) GO TO 1
CALCULATE STREAM OUTPUT

BOTTOM OUTLET STREAM
S(1:IOUTB:4)=TEMP

TOP OUTLET STREAM
S(1:IOUTT:4)=TEMP

SUM=0.0
DO 15 I=1:NCOMP
15 SUM=SUM+Y(I+2)
DO 16 I=1:NCOMP
16 S(1:IOUTT:5)=Y(I+2)/SUM

CONTROL SIGNALS
S(1:ISIGT:3)=Y(1)
S(1:ISIGB:3)=Y(9)
RETURN
END
FUNCTION XG(T)

THIS FUNCTION CALCULATES THE WEIGHT FRACTION OF SEPARATED
COMPONENT (g IN THE WILLIAMS-OTTO PLANT) ENTERING THE TOP LAYER
OF DECANTER AS A FUNCTION OF TEMPERATURE

T IS TEMPERATURE IN DEGREES F

IF (T.LE.100.0) GO TO 1
IF (T.GT.120.0) 2,3,4
1 XG=0.0
RETURN
2 XG=0.005769*0.1335*(0.01*T-1.1)*0.7396*(0.01*T-1.1)**2
   IF (XG.LT.0.0) XG=0.0
   RETURN
3 XG=0.050865
   RETURN
4 XG=0.0894*0.3125*(0.01*T-1.24)-0.9*(0.01*T-1.24)**2
   IF (T.GT.128.0) PRINT 10, T
10 FORMAT (23H DECANTER TEMPERATURE =,F6.1,42H REQUIRES EXTRAPOLATION
   10 OF SOLUBILITY DATA)
RETURN
END

S(I6+IOUT+3)=SUM
S(I6+IOUT+4)=S(I6+IN+4)
S(I6+IOUT+5)=S(I6+IN+5)
DO 15 I=1,NCOMP
15 S(I6+IOUT+I+5)=x(I)
RETURN
75 PRINT 20
20 FORMAT(37H CONVOL EQUIPMENT PARAMETER NE.1 OR.2)
STOP
END
SUBROUTINE TYPE14

SUBROUTINE SPLIT1

THIS MODULE SPLITS AN INCOMING STREAM INTO 2 EXIT STREAMS
ACCORDING TO A GIVEN RATIO

EQUIPMENT PARAMETERS

1 = RATIO = DESIRED FLOW RATIO OF FIRST EXIT STREAM
    TO ENTERING STREAM

COMMON /MAT/ MP (35,13), EP (35,10), S (2, 45, 11), EX (1)
COMMON /CON/ IG, NCOMP, NC5, HNE, NS, NPR, NPOL, TMAX, IORDER, NGRAPH
COMMON /UNIT/ IM, NMP

RATIO = EP (IM, 1)
IN = MP (IM, 3)
IOUT1 = IABS (MP (IM, 4))
IOUT2 = IABS (MP (IM, 5))
S (1, IOUT1, 3) = S (IG, IN, 3) * RATIO
S (1, IOUT2, 3) = S (IG, IN, 3) - S (1, IOUT1, 3)
DO 1 I = 4, NC5
   S (1, IOUT1, I) = S (IG, IN, I)
1   S (1, IOUT2, I) = S (IG, IN, I)
RETURN
END
SUBROUTINE TYPE1

SUBROUTINE VALV1

V=PORT (PARABOLIC) CONTROL VALVE

EQUIPMENT PARAMETERS

1 = NOT USED
2 = VALVE CONSTANT
3 = ACTION (==DIRECT,==REVERSE)

COMMON /UNIT/ IM,NMP
COMMON /MAT/ MP(35,13),EP(35,10),S(2,45,11),EX(1)
COMMON /CON/ IG,NCOMP,NCS,H,NE,NS,NPR,NPLOG,TMAX,IORDER,NGRAPH

MI=MP(IM,3)
MO=IABS(MP(IM,4))

C CHECK THAT THE VALVE SIGNAL IS IN THE RANGE 0-100
V=S(1,HI,3)
IF (V.LT.0.) V=0.
IF (V.GT.100.) V=100.

ACTION
A=EP(IM,3)
IF (A.LT.0.) S(1,MO,3)=EP(IM,2)*((100.-V)**2
IF (A.GE.0.) S(1,MO,3)=EP(IM,2)*V**2

RETURN
END
SUBROUTINE TYPE4
COMMON /UNIT/ IM,NMP
COMMON /MAT/ HP(35:13),EP(35:10),S(2:45:11),EX(1)
COMMON /CNTR/ INCOMP,NC5,H,NE,NS,NPR,NPOL,TMAX,IORDER,NGRAPH
COMMON /CNTR/ NCTR
DIMENSION SIGMA(10)
INTEGER OUT
DATA SIGMA/10,0.0/

CONTROLLER MODULE  BY R. MILLARES  NOV 1974

THIS CONTROLLER HAS FOUR OPTIONS: PROPORTIONAL, PROPORTIONAL AND INTEGRAL, PROPORTIONAL, INTEGRAL PLUS DERIVATIVE ACTIONS

THE FOLLOWING PARAMETERS ARE REQUIRED:
1. EP(IM,1)= VARIABLE TO CONTROL
2. EP(IM,2)= CONTROLLER UPPER LIMIT OF OPERATION (INPUT VAR)
3. EP(IM,3)= CONTROLLER LOWER LIMIT
4. EP(IM,4)= SET POINT, FOR VARIABLE S.P. SET THIS TO 0, AND ASSIGN STREAM NUMBER ON MP= ARRAY, AND STREAM FLAG= 12
5. EP(IM,5)= TYPE OF CONTROLLER
   1 = PROPORTIONAL
   2 = PI
   3 = PD
   4 = PID
   5 = ON-OFF (=EP(IM,6),GT,SETP,OFF,LT,SETP)
6. EP(IM,6)= DESIRED OUTPUT SIGNAL FOR ERROR= 0.
7. EP(IM,7)= GAIN, SEND(*) IF DIRECT ACTION IS DESIRED OR (-) FOR REVERSE ACTION
8. EP(IM,8)= INTEGRAL CONSTANT (RESET)
9. EP(IM,9)= DERIVATIVE CONSTANT
10. EP(IM,10)= TRACE OPTION (=1)

NTR=EP(IM,10)

IF(NTR.EQ.1) PRINT 500
FORMAT(5X,"ENTERING CONTROLLER")

IN = HP(IM,3)
OUT = HP(IM,4)
K = EP(IM,1)
N = EP(IM,5)
IF(IN. GT.4) GO TO 200
IF(IG.NE.1) RETURN
NCTR = NCTR+1
RANGE = EP(IM+2)-EP(IM+3)
IF (OUT.LT.0) GO TO 16
SETP = EP(IM+4)*50./RANGE
OLDERR = (S(2,IN+K)*50./RANGE)-SETP
GO TO 16

16 IN2 = -OUT
OUT = -MP(IM+5)
IF (S(1,IN+2).NE.11.) GO TO 15
IN = IN2
IN2 = MP(IM+3)

15 SEFP = S(1,IN2+3)
OLDERR = (S(2,IN+K)*50./RANGE)-S(2,IN2+3)

18 CONTINUE
S(1,IN+K) = S(2,IN+K)

30 ERR = (S(1,IN+K)*50./RANGE)-SEFP
PD = 0.
PI = 0.
GO TO (25,26,27,28)

27 PD = EP(IM+9)*(ERR-OLDERR)/H
GO TO 25

26 SIGMA(NCTR) = SIGMA(NCTR)+(H*(ERR-OLDERR)/2.)
PI = SIGMA(NCTR)/EP(IM+8)

25 S(1,OUT+3) = EP(IM+6)+(EP(IM+7)*(ERR+PI*PD))
RETURN

200 CONTINUE
IF (IG.EQ.1) RETURN
T=S(1,IN+K).
IF (N.EQ.6) GO TO 201
IF (T.LE.EP(IM+4)) T=EP(IM+6)
IF (T.LT.EP(IM+4)) T=0.0
GO TO 202

201 CONTINUE
IF (T.LE.EP(IM+4)) T=EP(IM+6)
IF (T.GT.EP(IM+4)) T=0.0

202 S(1,OUT+3)=T
RETURN
END
### SIMULATION OF WILLIAMS-OTTO PLANT

<table>
<thead>
<tr>
<th>TIME</th>
<th>COMPS</th>
<th>HMIN</th>
<th>IN/OUT</th>
<th>PRINTING</th>
<th>ORDER</th>
<th>LIBRARY</th>
<th>RORM2</th>
<th>REBO2</th>
<th>COLM1</th>
<th>COND1</th>
<th>PROCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>6.0</td>
<td>0.00000001</td>
<td>10.0</td>
<td>50.0</td>
<td>2.0</td>
<td>4.0</td>
<td>21.0</td>
<td>22.0</td>
<td>27.0</td>
<td>17.0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONT1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>-7.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>4.6720</td>
<td>4.6080</td>
<td>4.6400</td>
<td>2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.0</td>
<td>5.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VALV1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>-14.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>38.3844</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONT1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14.0</td>
<td>-8.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>175.0</td>
<td>6.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VALV1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>-9.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONT1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>14.0</td>
<td>-11.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.0</td>
<td>181.0</td>
<td>175.0</td>
<td>180.0</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.0</td>
<td>100.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VALV1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0</td>
<td>-12.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>51.148618</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONT1</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0</td>
<td>-5.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALV1</td>
<td>3.0</td>
<td>0.50</td>
<td>0.37</td>
<td>0.43</td>
<td>9863.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>60.0</td>
<td>0.1</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8.0</td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.0</td>
<td>-2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>13.34</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTR1</td>
<td>9.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>2.0</td>
<td>3.0</td>
<td>12.0</td>
<td>9.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14.0</td>
<td>-13.0</td>
<td>-10.0</td>
<td>-0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4640.0</td>
<td>105.08</td>
<td>9450.0</td>
<td>9450.0</td>
<td>9450.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONT1</td>
<td>10.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.0</td>
<td>-15.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALV1</td>
<td>11.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.0</td>
<td>-16.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>61.40</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXCH1</td>
<td>12.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14.0</td>
<td>16.0</td>
<td>-18.0</td>
<td>-17.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>274.0</td>
<td>95.0</td>
<td>365.0</td>
<td>82.5</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONT1</td>
<td>13.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>22.0</td>
<td>-23.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>10481.3</td>
<td>109483.8</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>53.0</td>
<td>982.22</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALV1</td>
<td>14.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23.0</td>
<td>-24.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>36.90</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONT1</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>19.0</td>
<td>-20.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>3241.35</td>
<td>2932.65</td>
<td>3087.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50.0</td>
<td>2.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALV1</td>
<td>16.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20.0</td>
<td>-21.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>1486.8</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCAN1</td>
<td>17.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.0</td>
<td>-24.0</td>
<td>-21.0</td>
<td>-22.0</td>
<td>-19.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4895.22</td>
<td>3087.0</td>
<td>5.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONV1</td>
<td>REBO2</td>
<td>RDRM2</td>
<td>COLM1</td>
<td>COND1</td>
<td>STIR1</td>
<td>CONT1</td>
<td>VALV1</td>
<td>CONT1</td>
<td>VALV1</td>
<td>CONT1</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>18.0</td>
<td>-25.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>24.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>19.0</td>
<td>39.0</td>
<td>38.0</td>
<td>40.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>7682500.0</td>
<td>15750.0</td>
<td>120.0</td>
<td>0.5</td>
<td>13.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14.357</td>
<td>-3818.33</td>
<td>227.21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20.0</td>
<td>35.0</td>
<td>40.0</td>
<td>-38.0</td>
<td>-34.0</td>
<td>-36.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>21.0</td>
<td>25.0</td>
<td>30.0</td>
<td>34.0</td>
<td>-35.0</td>
<td>-26.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>-28.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10.0</td>
<td>5.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>9.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>22.0</td>
<td>26.0</td>
<td>-27.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23.0</td>
<td>27.0</td>
<td>-30.0</td>
<td>-31.0</td>
<td>-33.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>40.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>24.0</td>
<td>31.0</td>
<td>-32.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>3.0</td>
<td>65.0</td>
<td>39.0</td>
<td>50.0</td>
<td>4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50.0</td>
<td>1.5</td>
<td>0.5</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>25.0</td>
<td>32.0</td>
<td>-33.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>26.0</td>
<td>28.0</td>
<td>-29.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td>110.0</td>
<td>8.0</td>
<td>95.0</td>
<td>2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40.0</td>
<td>1.5</td>
<td>0.5</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>27.0</td>
<td>29.0</td>
<td>-30.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>28.0</td>
<td>36.0</td>
<td>-37.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>100.0</td>
<td>60.0</td>
<td>80.0</td>
<td>4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VALV1</td>
<td>50.0</td>
<td>1.5</td>
<td>0.5</td>
<td>0.1</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>29.0</td>
<td>-38.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.2644</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONV1</td>
<td>36.0</td>
<td>38.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPLIT1</td>
<td>31.0</td>
<td>41.0</td>
<td>-3.0</td>
<td>-42.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td>42.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STREAMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPLICIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
<td>14500</td>
<td>70.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>2.0</td>
<td>3335</td>
<td>70.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>3.0</td>
<td>48111</td>
<td>100.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>5.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>6.0</td>
<td>4640</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>7.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>8.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>9.0</td>
<td>951</td>
<td>250.0</td>
<td>29.825</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>10.0</td>
<td>951.877</td>
<td>250.0</td>
<td>29.825</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>11.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>11.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>60.0</td>
<td>14.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>13.0</td>
<td>1.0</td>
<td>0.0</td>
<td>62.0</td>
<td>14.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>14.0</td>
<td>1.0</td>
<td>95961.0</td>
<td>70.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30303</td>
<td>0.69697</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>15.0</td>
<td>11.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>16.0</td>
<td>1.0</td>
<td>153500.0</td>
<td>60.0</td>
<td>14.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>17.0</td>
<td>1.0</td>
<td>153500.0</td>
<td>61.0</td>
<td>14.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.0</td>
<td>1.0</td>
<td>95961.0</td>
<td>62.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30303</td>
<td>0.69697</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>19.0</td>
<td>11.0</td>
<td>3087.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>20.0</td>
<td>11.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21.0</td>
<td>1.0</td>
<td>3712.0</td>
<td>62.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>22.0</td>
<td>11.0</td>
<td>9982.22</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.0</td>
<td>11.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.0</td>
<td>1.0</td>
<td>92249.0</td>
<td>62.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30303</td>
<td>0.69697</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>25.0</td>
<td>1.0</td>
<td>649.0</td>
<td>100.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1261</td>
<td>0.3992</td>
<td>0.0255</td>
<td>0.3615</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0877</td>
<td>26.0</td>
<td>1.0</td>
<td>649.0</td>
<td>94.0</td>
<td>270.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1261</td>
<td>0.3992</td>
<td>0.0255</td>
<td>0.3615</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0877</td>
<td>27.0</td>
<td>1.0</td>
<td>649.0</td>
<td>89.0</td>
<td>242.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1261</td>
<td>0.3992</td>
<td>0.0255</td>
<td>0.3615</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0877</td>
<td>28.0</td>
<td>11.0</td>
<td>0.0</td>
<td>80.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29.0</td>
<td>11.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.0</td>
<td>1.0</td>
<td>649.0</td>
<td>89.0</td>
<td>242.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1261</td>
<td>0.3992</td>
<td>0.0255</td>
<td>0.3615</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0877</td>
<td>31.0</td>
<td>11.0</td>
<td>50.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32.0</td>
<td>11.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33.0</td>
<td>1.0</td>
<td>33.0</td>
<td>89.0</td>
<td>242.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1261</td>
<td>0.3992</td>
<td>0.0255</td>
<td>0.3615</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0877</td>
<td>34.0</td>
<td>1.0</td>
<td>649.0</td>
<td>121.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1261</td>
<td>0.3992</td>
<td>0.0255</td>
<td>0.3615</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0877</td>
<td>35.0</td>
<td>1.0</td>
<td>1265.0</td>
<td>121.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1261</td>
<td>0.3992</td>
<td>0.0255</td>
<td>0.3615</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0877</td>
<td>36.0</td>
<td>11.0</td>
<td>80.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37.0</td>
<td>11.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38.0</td>
<td>1.0</td>
<td>616.0</td>
<td>121.0</td>
<td>350.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1261</td>
<td>0.3992</td>
<td>0.0255</td>
<td>0.3615</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0877</td>
<td>39.0</td>
<td>9.0</td>
<td>487.0</td>
<td>344.35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.0</td>
<td>9.0</td>
<td>7682500.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41.0</td>
<td>1.0</td>
<td>87487.0</td>
<td>100.0</td>
<td>44.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1329111754</td>
<td>0.4209654006</td>
<td>0.268382730</td>
<td>0.3811766320</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0381085190</td>
<td>42.0</td>
<td>1.0</td>
<td>39376.0</td>
<td>100.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1329111754</td>
<td>0.4209654006</td>
<td>0.268382730</td>
<td>0.3811766320</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0381085190</td>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROPERTIES</td>
<td>20.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.0</td>
<td>60.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62.24</td>
<td>123916.59</td>
<td>2096.2</td>
<td>1.0</td>
<td>14250.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62.24</td>
<td>123916.59</td>
<td>2096.2</td>
<td>1.0</td>
<td>14250.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX 1: PROGRAM LISTINGS

A listing is given in Figure I.1 of the DYNSYS 2.0 executive program except for SUBROUTINES TRGB and TRGB2. These routines were made available to the author for research purposes, but are considered proprietary by their creators. Copies of these may be obtained from:

MR. H.P. HUTCHISON
LECTURER
DEPT. OF CHEMICAL ENGINEERING
CAMBRIDGE UNIVERSITY
CAMBRIDGE, ENGLAND

A fee will probably be required from commercial users.

I.1 The DYNSYS 2.1 Executive Program

Since version 2.0 is not completely available, another version of DYNSYS, version 2.1, was created to have a sparse matrix oriented package which was completely available to the public. Key's subroutine SIMULT was used in place of TRGB. Version 2.1 is not as efficient as 2.0 since Key's program is not designed for repeated solution of linear equations. The new version, however, does illustrate the usage of DYNSYS and can be keypunched and
used by anyone.

There are some slight differences in the writings of the modules. A skeleton module is given (Figure 1.2) and also a listing of the executive and a sample module (Figure 1.3) which corresponds to the second example of Chapter 8 (Figure 8.7), a CSTR with first order reversible reaction. The only executive subroutines given in Figure 1.3 are DYN1, DYN2, DIFSUB, SIMULT and RITE. These replace subroutines DYN1, DYN2, DIFSUB, TRGB and TRGB2 in DYNSYS 2.0. Subroutines GET, FETCH, PROPS, TEMP, TRIDAG and GRAPH are identical with those in DYNSYS 2.0. The mainline program and subroutine OUTPUT are identical with DYNSYS 2.0 except for the COMMON/GEAR2/.

The differences in writing of modules are as follows. The Jacobian is stored as described in Section 5.1.2.

COMMON/ROW/IROW(4) is not required.

COMMON/COLUMN/JCOL(4) and COMMON/JACOB/XJACOB(4) are replaced by

COMMON/COLUMN/JCOL(10,2) and COMMON/JACOB(10,2)

i.e. the column numbers and nonzero elements are now stored as two dimensional arrays. Note, however, that the row dimension of JCOL and XJACOB must be the same as...
COMMON/MODULE/IDERY,ITER,ITRI,NZERO is replaced by COMMON/MODULE/IDERY,ITER,ITRI,MC,IPIVOT

NZERO is no longer required, but the user must supply, in SECTION #2 of the skeleton, values of MC and IPIVOT. MC is the maximum number of columns in the XJACOB matrix where the nonzero Jacobian elements are stored. This value can become larger than the maximum number of nonzeros in any Jacobian row since additional nonzero elements can be created during the solution. MC should be constant for any module and may have to be determined by trial and error. It is also a function of the pivot option. If MC is too small, the program will stop and print the message

MC SHOULD BE AT LEAST

MC should then be set to at least the number in the blank.

IPIVOT is the pivot option (from 1-7) used for the module. The options are described in the skeleton and in section 5.1.2.

The remainder of the module is written exactly as for DYNSYS 2.0. The only change in the data set is that MINPIVOT is not used.
The difference in execution time between version 2.0 and 2.1 will depend on the size of the system. For very small examples the times will be nearly the same. For large systems, Chapter 5 gives execution times for SIMULT and TRGB-TRGB2 for 50, 100 and 200 equations. For example for 50 equations, bandwidth 9, the best option of Key's method took 0.691 seconds while TRGB2 took 0.049 seconds. Since the linear equations must be solved at each Newton-Raphson iteration of the corrector, the time difference could become substantial for a long simulation with many time steps. For 500 time steps with an average of two Newton-Raphson iterations per step, the execution time would differ by over a minute.
FIGURE I.11: LISTING OF DYNSY 2.0 EXECUTIVE

```plaintext
PROGRAM DYNSY (INPUT, OUTPUT, PUNCH, TAPE5=INPUT, TAPE6=OUTPUT, TAPE7=PULLCH)

COMMON /MAT/ MP (35, 13), EP (35, 10), S (2, 45, 11), EX (1)
COMMON /CON/ IG, NCOMP, NC5, H, NE, NS, NPR, NPOL, TMAX, IOORDER,NGRAPH
COMMON /PLT/ NPLOTS, PLOTI, PLOTS (20, 4), PLOTT
COMMON /LAG/ NSX, NSW
COMMON /FTN/ FN (1, 1), NFN
COMMON /GEAR1/ Y (8, 222), SAVE (10, 222), ERROR (222), YMAX (222), DERY1 (222), DERY2 (222)

1 CALL DYN1
2 PLOTT = 0.0
3 DO 3 I = 1, 222
4 DERY2 (I) = 0.0
5 YMAX (I) = 1.0
6 CONTINUE
7 JSTART = 0
8 NBVMAX = 0
9 OTIME = TIME
10 NSW = 0
11 IF (NPOL GT 0) GO TO 4
12 GO TO 5
13 READ (5, 25) X
14 NPLOTS = X
15 READ (5, 25) PLOTT
16 READ (5, 25) ((PLOTS (I, J), J = 1, 4), I = 1, NPLOTS)
17 PRINT 25, ((PLOTS (I, J), J = 1, 4), I = 1, NPLOTS)
18 CONTINUE
19 SET UP INITIAL VALUES FOR GRAPH VECTOR COUNT
20 NV = 0
21 NT = 0
22 CALL OUTPUT
23 KPRINT = 0
24 CONTINUE
25 TIME = OTIME + H
26 KFLAG = 1
27 ICONV = 0
28 KPRINT = KPRINT + 1
29 IG = 2
30 CALL DYN2
31 IG = 1
32 CALL DYN2
33 IF (NPOL GT 0) GO TO 7
34 IF (KPRINT LT NPR) GO TO 8
35 CONTINUE
36 CALL OUTPUT
```

---

The code listing is for a program named DYNSY 2.0, which appears to be related to numerical computations or simulations. The program includes various common blocks for defining parameters and variables, and it seems to be designed to handle data for plotting and other graphical outputs. The code includes sections for reading data, setting initial values, and performing computations which are typical in scientific computing environments.
KPRINT=0
CONTINUE
NSW=1
IF (ICONV.EQ.1.OR.KFLAG.NE.1) GO TO 6
NSW=0
OTIME=TIME

UPDATE STREAM MATRIX

DO 10 I=1*NS
   DO 9 J=3*NC5
      S(2*I+J)=S(1*I+J)
   CONTINUE
9	CONTINUE
10 CONTINUE
   IF (TIME.LE.TMAX) GO TO 6
   WRITE (6*24)
   CALL OUTPUT
   IF (NGRAPH.NE.1) GO TO 11
   CALL GRAPH (NPLOTS)
11 CONTINUE
   CALL GET (N*X,0)
   IF (N.EQ.3HEND) GO TO 19
   IF (N.EQ.6HREPEAT) GO TO 1
   IF (N.EQ.8HCONTINUE) GO TO 12
   PRINT 20
STOP
12 H=0.00001
13 CALL GET (N*X,1)
   IF (N.EQ.4HTIME) TMAX=X
   IF (N.EQ.8HLINEPLOT) NPOL=X
   IF (N.EQ.5HGRAPH) NGRAPH=X
   IF (N.EQ.6HALTER) GO TO 14
   IF (N.EQ.6HALTERE) GO TO 16
   IF (N.EQ.8HFUNCTION) GO TO 17
   IF (N.EQ.3HEND) GO TO 2
   GO TO 13
14 NOS=X
READ 21, (S(1*NOS+J),J=1,5)
PRINT 23, (S(1*NOS+J),J=1,5)
READ 22, (S(1*NOS+J),J=6,NC5)
PRINT 23, (S(1*NOS+J),J=1,NC5)
DO 15 J=1,NC5
   S(2*NOS+J)=S(1*NOS+J)
15 CONTINUE
GO TO 13
16 NOEP=X
READ 21, (EP(NOEP+J),J=1,5)
PRINT 23, (EP(NOEP+J),J=1,5)
GO TO 13
17 NFN=X
DO 18 I=1,NFN
   READ 21, (FN(I,J),J=1,5)
READ 22, (FN(I,J), J=6,NC5)
CONTINUE
PRINT 23, ((FN(I,J), J=1,NC5), I=1,NFN)
GO TO 13
STOP

C
20 FORMAT ('41H LAST CARD MUST BE END.REPEAT OR CONTINUE')
21 FORMAT (12X,'5F12.2')
22 FORMAT (12X,'4F12.2')
23 FORMAT (1X,'9F10.3')
24 FORMAT ('720X,'23H '***TMAX APPROACHED***')
25 FORMAT ('4F12.4')
END
SUBROUTINE DYN1

C DATA-LOADING AND PRINTING
C COMMON /UNIT/ IM=NMP
C COMMON /MAT/ MP(35:13), EP(35:10), S(2:45:11), EX(1)
C COMMON /CON/ IG, NCOMP, NC5, H5, NS, NSP, NPOL, TMAX, IORDER, NGRAPH
C COMMON /GEAR/ EPS, TIME, KFLAG, JSTART, NBVMAX, ICONV, NOPPT, ISTITF
C COMMON /GEAR3/ HMIN, HMAX, XMIN, NOMES
C DIMENSION AM(100)
C DIMENSION ITAG(30)
C DATA ITAG(5)/10, 5HVAL1, 5HVAL2, 5HSTIR1, 5HCONT1, 5HSETL1, 6HMXPFL1, 5HSETTL1, 1.6HVSS1, 5HDELAY, 5HCSTR1, 5HECH1, 5HDCAN1, 5HCONV1, 5HSPLT1, 16*6H
C
2 / WRITE (6, 27)
C
C DEFAULT VALUES
C TIME=0.0
C MAXNE=35
C NPR=1
C TMAX=10.0
C NCOMP=1
C NMP=5
C H=0.00001
C HMIN=0.000001
C HMAX=0.05
C EPS=0.001
C NPOL=0
C NGRAPH=0
C IORDER=6
C ISTITF=1
C NOMES=0
C XMIN=1.0E-06
C
C READ AND COPY TITLE
C READ (5, 29) (AM(I), I=1, 8)
C WRITE (6, 28) (AM(I), I=1, 8)
C IF (AM(1), NE, 5HBEGIN) GO TO 1
C
C READ INITIAL DATA
C CALL GET (N*X, 1)
C IF (N*EQ, 5HCOMPS) NCOMP=X
C IF (N*EQ, 6HMIN/OUT) NMP=X
C IF (N*EQ, 6HDELTAT) H=X
C IF (N*EQ, 4HMIN) HMIN=X
C IF (N*EQ, 4HMAX) HMAX=X
C IF (N*EQ, 4HTIME) TMAX=X
C IF (N*EQ, 8HPRINTING) NPR=X
C IF (N*EQ, 9HTOLERANCE) EPS=X
C IF (N*EQ, 5HORDER) IORDER=X
C IF (N*EQ, 8HLINEPLOT) NPOL=1
C IF (N*EQ, 5HGRAPH) NGRAPH=1
C IF (N*EQ, 8HNONSTIFF) ISTITF=0
C IF (N*EQ, 8HMNONMESSAGE) NOMES=1
C IF (N*EQ, 6HMNPIVOT) XMIN=X
C IF (N*EQ, 7HLIBRARY) GO TO 3
IF (N, EQ., 7HPROCESS) GO TO 6.
IF (N, EQ., 8HFUNCTION) GO TO 5
GO TO 2

3
NL1B=X
DO 4 I=1,NL1B
CALL GET (N,X*0)
J=X
ITAG(J)=N
CONTINUE
GO TO 2

5
NFX=X
NC5=NCOMP+5
READ 39, ((FN(I,J), J=1, NC5), I=1,NFX)
PRINT 39, ((FN(I,J), J=1, NC5), I=1,NFX)
GO TO 2

6
CONTINUE
NMP=NVF+2
MAXNMP=NMP+1
NC5=NCOMP+5
WRITE (6*30)
DO 9 I=1,MAXNMP
DO 7 J=1,MAXNMP
MP(I,J)=0
CONTINUE
GO TO 8

7
CONTINUE
DO 8 J=1,10
EP(I,J)=0.0
CONTINUE

8
CONTINUE
NE=0
NEXT=1
TIME=0.0
CALL GET (NM,X*0)

10
IF (NM, EQ., 3HEND) GO TO 16
DO 12 I=1,30.
IF (NM, EQ., ITAG(I)) GO TO 13
CONTINUE
WRITE (6*31) NM
I=0

12
CONTINUE
WRITE (6*31) NM
I=0

13
NT=I
NE=NE+1
READ (5*32) (AM(I), I=1,NMP)
DO 14 J=1,NMP
MP(NE,J)=AM(J)
CONTINUE
MP(NE,1)=X
MP(NE,2)=NT
WRITE (6*33) (MP(NE,J), J=1,NMP)
READ (5*32) (EP(NE,I), I=1,10)
WRITE (6*34) (EP(NE,I), I=1,10)
CALL GET (NM,X*0)
IF (NM, NE, SHEXTRA) GO TO 15
NNX=X
NMX=NX+NX=1
READ (5,32) (EX(J), J=NX,NMX)
WRITE (6,35) (EX(J), J=NX,NMX)
MP(NE,NMP+1)=NX
NX=NX+NX
CALL GET (NM,X)=1
IF (NM,NE,THRESERVE) GO TO 11
NX=X
NX=NX+NX
GO TO 10
15 CALL FETCH (7HSTREAMS,NS)
WRITE (6,35) NS
DO 18 I=1,NS
S(1,I,1)=0
S(1,I,2)=1.0
S(1,I,3)=0.0
S(1,I,4)=530.0
S(1,I,5)=14.7
DO 17 J=6,NC5
S(1,I,J)=0
17 CONTINUE
18 CONTINUE
19 CALL GET (NM,X)=1
IF (NM,NE,HXPLICIT) GO TO 20
IF (NM,NE,3HEND) GO TO 25
IF (NM,NE,4HSPEC) GO TO 22
GO TO 19
20 READ (5,36) IT,(AM(I), I=1,5)
IF (IT,NE,3HEND) GO TO 25
READ (5,32) (AM(I), I=6,NC5)
N=AM(1)
DO 21 I=1,NC5
S(1,N+I)=AM(I)
21 CONTINUE
GO TO 20
22 N1=X
READ (5,32) (AM(I), I=1,NC5)
N2=AM(1)
DO 24 I=N1,N2
DO 23 J=2,NC5
S(1,I,J)=AM(J)
23 CONTINUE
S(1,I+1)=I
24 CONTINUE
GO TO 19
25 DO 26 I=1,NS
K=S(I,I+2)
WRITE (6,38) I,K,(S(I,I+J), J=3,5)
WRITE (6,37) (S(I,I+J), J=6,NC5)
DO 26 J=1,NC5
26 S(2,I,J)=S(I,I+J)
XX=PROPS(0,0)
RETURN

27 FORMAT (1HM1)
28 FORMAT (1X,8A10)
29 FORMAT (8A10)
30 FORMAT (1HM1)
31 FORMAT (18H ***FAIL*** NAME :A12*8H NOT SET)
32 FORMAT (12X,5F12.0)
33 FORMAT (/9H UNIT =I3,9H= TYPE =I3,1H=I12S)
34 FORMAT (11X,5F14.5)
35 FORMAT (1H1,I6*8H STREAMS)
36 FORMAT (A3*9X,5F12.0)
37 FORMAT (8X*4F14.5)
38 FORMAT (11H STREAM =I3*4H =I3,1HM*3F14.5)
39 FORMAT (12X,5F12.2)
END
SURROUNGE GET (NAME*,X*,IFG*)
READ (5*6) NAME*,X*
  IF (NAME*,NE.*10H) GO TO 2
  IF (X*EQ.*0.000 OR X*EQ.*0.) GO TO 1
  WRITE (6*7)
  GO TO 1
CONTINUE
IF (X*LE.*0.) GO TO 3
IF (IFG*) 3*4.*5
WRITE (6*9) NAME*
RETURN
WRITE (6*8) NAME*,N
RETURN
WRITE (6*9) NAME*,X*
RETURN
FORMAT (A10*,F14.0)
FORMAT (A8H ---WARN--- DATA SKIPPED WHILE READING KEYS)
SUBROUTINE FETCH (IWORD, 1)
READ (5, 2) IWORD, AI
IF (IWORD .NE. IWORD) GO TO 1
I = AI
RETURN
C
FORMAT (A10, F14.2)
END
SUBROUTINE DYN2
COMMON /UNIT/ IM,NMP
COMMON /MAT/ MP(35,10),EP(35,10),S(2,45,11),EX(1)
COMMON /CON/ IG,NCOMP,NCS,H,NE,NS,NNP,NPOL,TMAX,ORDER,NGRAPH
COMMON /BKV/ NBV
COMMON /CNTR/ NCTR
COMMON /LAG/ NSX,NSW
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBVMAX,ICONV,NOPPT,ISTIFF
NBV=0
NOPPT=0
NSX=0
NCTR=0
DO 31 IM=1,NE
IF (ICONV.EQ.1) RETURN
NTYPE=MP(IM,2)
GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23)
1,24,25,26,27,28,29,30)
NTYPE
1 CALL TYPE1
GO TO 31
2 CALL TYPE2
GO TO 31
3 CALL TYPE3
GO TO 31
4 CALL TYPE4
GO TO 31
5 CALL TYPE5
GO TO 31
6 CALL TYPE6
GO TO 31
7 CALL TYPE7
GO TO 31
8 CALL TYPE8
GO TO 31
9 CALL TYPE9
GO TO 31
10 CALL TYPE10
GO TO 31
11 CALL TYPE11
GO TO 31
12 CALL TYPE12
GO TO 31
13 CALL TYPE13
GO TO 31
14 CALL TYPE14
GO TO 31
15 CALL TYPE15
GO TO 31
16 CALL TYPE16
GO TO 31
17 CALL TYPE17
GO TO 31
18 CALL TYPE18

GO TO 31
19 CALL TYPE19
   GO TO 31
20 CALL TYPE20
   GO TO 31
21 CALL TYPE21
   GO TO 31
22 CALL TYPE22
   GO TO 31
23 CALL TYPE23
   GO TO 31
24 CALL TYPE24
   GO TO 31
25 CALL TYPE25
   GO TO 31
26 CALL TYPE26
   GO TO 31
27 CALL TYPE27
   GO TO 31
28 CALL TYPE28
   GO TO 31
29 CALL TYPE29
   GO TO 31
30 CALL TYPE30
31 CONTINUE
   RETURN
   END
SURROUTINE OUTPUT

C THIS ROUTINE PRINTS OUT THE RESULTS

COMMON /MAT/ MP(35,13),EP(35,10),S(2,45,11),EX(1)
COMMON /CON/ IG,NCOMP,NC5,H,HE,NS,NPR,NPOL,TMAX,IORDER,NGRAPH
COMMON /PLT/ NPLOTS,PLNOT,PLOTS(20,4),PLOTT
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBVMAX,ICONV,NOPPT,ISTIFF
COMMON /FTN/ FN(1,1),NFN
COMMON /FF/ TITLE(11),MPLSOT(20,2),PLTR(20,2)
COMMON /G/ NT,NV,TLT(602),V(602),VL(1,602)
DIMENSION SYMBOL(15),SMYMBOL(4),ALINE(125)
DATA SYMBOL/IHA,IHB,IHC,IM,HE,IF,MG,IMH,IME,IMK,IML,IMM,IMOUT
DATA SYMBOL/IHA,IHB,IHC,IM,HE,IF,MG,IMH,IME,IMK,IML,IMM,IMOUT
DATA SYMBOL/IHA,IHB,IHC,IM,HE,IF,MG,IMH,IME,IMK,IML,IMM,IMOUT
MP3,S,HCOMP5,S,HCOMP5,S,HCOMP5
IF (NPOLEN.0) GO TO 3
WRITE (6,14) TIME
WRITE (6,15)
DO 1 I=1,NS
IF (S(2,2>I),LT,0.0) GO TO 1
NN=S(2,1,I++)
WRITE (6,17) NN,(S(I,J),J=3,NC5)
CONTINUE
DATA SYMBOL/IHA,IHB,IHC,IM,HE,IF,MG,IMH,IME,IMK,IML,IMM,IMOUT
DO 2 I=1,NE
IF (MP(I,1),GT,0) GO TO 2
K=JABS(MP(I,1))
WRITE (6,16) K,(EP(I,J),J=1,5)
2 CONTINUE
RETURN
3 CONTINUE
IF (JSTART.NE.0) GO TO 6
C START PLOTTING RESULTS
WRITE (6,19)
DO 4 I=1,NPLOTS
N1=PLOTS(I,1)
N2=PLOTS(I,2)
MPLSOT(I,1)=N1
MPLSOT(I,2)=N2
PLTR(I,1)=PLSOT(I,3)
PLTR(I,2)=PLSOT(I,4)
WRITE (6,20) SYMBOL(I),N1,TITLE(N2),PLTS(I,3),PLTS(I,4)
4 CONTINUE
X=0.
DO 5 I=1,21
ALINE(I)=X
X=X+5.
5 CONTINUE
WRITE (6,21) (ALINE(I),I=1,21)
WRITE (6,22)
6 CONTINUE
C CALCULATE NUMBER OF LINES TO SKIP
XSPACE=(TIME-PLOTT)/PLOTT
NSPACE=XSPACE
XSPACE=XSPACE
IF ((XSPACE=XSPACE).GT.0.5) NSPACE=NSPACE+1
IF (JSTART.EQ.0) NSPACE=2
IF (NSPACE.LT.1) RETURN
C
SET UP ALINE
DO 7 1=2*120
ALINE(1)=SYMBOL1(3)
CONTINUE
ALINE(1)=SYMBOL1(1)
ALINE(101)=SYMBOL1(1)
PLTNT=TIME
NSPACE=NSPACE-1
IF (NSPACE.EQ.0) GO TO 9
DO 8 1=1+NSPACE
WRITE (6,18) (ALINE(J),J=1,101)
CONTINUE
C
SET UP NEW LINE()
N5=1
N6=NPLOTS
NV=NV+1
IF (NV.GT.602.OR.NV.LT.1) PRINT 24
DO 13 1=N5,N6
N1=PLOTS(I,1)
N2=PLOTS(I,2)
XN2=(S(N1,N2)-PLOTS(I,2))*100./((PLOTS(I,4)-PLOTS(I,3))
VPLT(I,NV)=XN2
XN3=XN2*N3
N3=N3
XXN3=N3
IF ((XXN3-XXN3).GT.0.5) N3=N3+1
IF (N3.LT.2.OR.N3.GT.100) GO TO 13
IF (ALINE(N3).NE.SYMBOL1(3)) GO TO 10
ALINE(N3)=SYMBOL1(I)
GO TO 13
DO 11 J=102,108
IF (ALINE(J).EQ.SYMBOL1(3)) GO TO 12
CONTINUE
J=109
ALINE(J)=ALINE(N3)
ALINE(J+1)=SYMBOL1(I)
ALINE(J+2)=SYMBOL1(4)
CONTINUE
WRITE (6,23) TIME,SYMBOL1(2),(ALINE(J),J=1,110)
NT=NT+1
IF (NT.GT.602.OR.NV.LT.1) PRINT 24
TPTOT(NT)=TPTOT
RETURN
C
FORMAT (/F20.5,F5.1,F5.1,F30.5,TIME=,F11.5,COMP1) 102
15X,F5.1,/) 103
FORMAT (61H STREAM TOT FLOW TEMP PRESS COMP 1 COMP 2) 104
16 FORMAT (1H0,5X,13,5X,6(3X,F12.5)) OUT 105
17 FORMAT (16,F12.3,2F8.1,9F9.5) OUT 106
18 FORMAT (19X,101A1) OUT 107
19 FORMAT (1H1,20X,6HSYMBOL,11X,6HSTREAM,23X,8Hvariable,19X,6MRANGES) OUT 108
20 FORMAT (1H0,22X,A2,13X,13,28X,A7,3X,Z(2X,F12.5)) OUT 109
21 FORMAT (///15X,21(1X,F4.0)) OUT 110
22 FORMAT (14X,5(1H.),21(5H..))) OUT 111
23 FORMAT (10X,F6.3,2X,112A1) OUT 112
24 FORMAT (1X,25H NT,NV,OR(N3 OUT OF RANGE)) OUT 113
END OUT 114

SUBROUTINE GRAPH (NPLOTS)
COMMON /UNIT/ IM*NHP
COMMON /CON/ IG,NCOMP,NC5,M,NE,NS,NPR,NPL,ITMAX,ORDER,NGRAPH
COMMON /G/ NT,NV,TPLOT(602),V(602),VLOT(1,602)
COMMON /FF/ TITLE(11),MPLOT(20,2),PLOT(20,2)
V(NT*1)=0.0
V(NT*2)=12.5
AAAA=X*AX IS ACTUAL INCHES
AAAA=10.
TPLOT(NT*1)=0.0
TPLOT(NT*2)=TMAX/AAAA
CALL PLOTS (60,24,10,75,2)
CALL PLOT (0.5,0.50,-3)
C
C SET UP AXIS
CALL AXIS3 (0.0,0.0,9HTIME (HRS)=-9,AAAA,0.0,TPLOT(NT*1),TPLOT(NT*2)
1.0,90.50)
CALL AXIS3 (0.0,0.0,6HV=AXIS,6,8.00,90.00,V(NT*1),V(NT*2),10,4.2)
C
C PLOT K=1 NPLOTS CURVES WITH DIFFERENT SYMBOLS
DO 2 K=1,NPLOTS
DO 1 J=1,NT
V(J)=VLOT(K,J)
1 CONTINUE
CALL LINE3 (TPLOT,V,NT,1,2,K,0)
2 CONTINUE
C
C WRITE HEADING
XD=11.5
YD=7.0
CALL SYMBOL (XD,YD,2,7HEADINGS,,0.7)
XD=10.5
YD=YD-0.3
DO 3 K=1,NPLOTS
C
C WRITE SYMBOL PEN UP
YD=YD+0.05
CALL SYMBOL (XD,YD,1,K,0.0,1)
C
C WRITE STREAM AND NUMBER
YD=YD+0.05
CALL SYMBOL (XD+0.3,YD,0.1,5HSTRM,,0.5)
CALL WHERE (P,Q,F)
STRMNO=MPLOT(K,1)
CALL NUMBER (P,Q,0.1,STRMNO,0.0,1)
CALL WHERE (P,Q,F)
C
C WRITE VARIABLE NAMES
CALL SYMBOL (P+0.3,Q,0.1,TITLE(MPLOT(K,2)),0.0,4)
CALL WHERE (P,Q,F)
C
C WRITE RANGES FOR VARIABLES
CALL NUMBER (P+0.3, Q+0.1; PLOT(K+1); 0.0; 2)
CALL WHERE (P+Q+F)
CALL SYMBOL (P+Q+0.1; .3H - .0; 0.3)
CALL WHERE (P+Q+F)
CALL NUMBER (P+Q+0.1; PLOT(K+2); 0.0; 2)
YD=YD-0.2
CONTINUE
CALL ENDPLT
RETURN
END
REAL FUNCTION PROPS (IG, IS)
COMMON /MAT/ MP (35, 13), EP (35, 10), S (2, 45, 11), EX (1)
COMMON /CON/ IU, NCOMP, NCST, NS, NPR, NPOL, TMAX, IORDER, NGRAPH
COMMON /PTAB/ IFLAG, PP (10, 20)
C ENTER HERE ON FIRST CALL
PROP = 0, 0
IFLAG = 0
CALL GET (NAME, X, 0)
NPP = X
IF (NPP) 1, 3, 5
C -VE, TREAT AS WATER, NO TEMP COEFF.
1 CONTINUE
DO 2 J = 1, NCOMP
DO 2 I = 1, 20
PP (I, J) = 0.
IF (I, J) EQ 1 PP (I, J) = 18.
IF (I, J) EQ 2 PP (I, J) = 1.
IF (I, J) EQ 16 PP (I, J) = 62, 4
2 CONTINUE
WRITE (6, 7)
RETURN
3 IFLAG = 1
C TREAT AS IG, MW AS AIR.
DO 4 I = 1, NCOMP
DO 4 J = 1, 20
PP (I, J) = 0.
IF (I, J) EQ 1 PP (I, J) = 29
IF (I, J) EQ 2 PP (I, J) = 26
4 CONTINUE
WRITE (6, 8)
RETURN
5 WRITE (6, 10) NPP
DO 6 I = 1, NCOMP
READ (5, 9) (PP (I, J) + J = 1, NPP)
WRITE (6, 11) (PP (I, J) + J = 1, NPP)
6 CONTINUE
RETURN
C
7 FORMAT (20H PROPERTIES AS WATER)
8 FORMAT (18H PROPERTIES AS AIR)
9 FORMAT (12X, 5F12.0)
10 FORMAT (15H PROPERTY TABLE, 14, 8H ENTRIES)
11 FORMAT (5F12.5)
END
FUNCTION CP (IG, IS)

COMMON /MAT/ MP(35, 13), EP(35, 10), S(2, 5, 11), EX(1)
COMMON /CON/ IU, NCOMP, NC5, M, NE, NS, NPR, NPOL, TMAX, IORDER, NGRAPH
COMMON /PTAB/ IGFLAG, PP(10, 20)

IF (IGFLAG .NE. 1) GO TO 1
CP = 0.26
RETURN

SUM = 0.0
T = S(IG, IS, 4) * PP(1, 20)
DO 2 I = 1, NCOMP
    SUM = SUM + S(IG, IS, I + 5) * ((PP(I, 5) * T * PP(I, 4)) * T * PP(I, 3)) * T * PP(I, 2))
2 CONTINUE
CP = SUM
RETURN
END
FUNCTION MW (IG, IS)
COMMON /MAT/ MP(35,13), EP(35,10), S(2*5,11), EX(1)
COMMON /CON/ IU, NCOMP, NC5, NE, NS, NPR, NPOL, THAX, IORDER, NGRAPH
COMMON /PTAB/ I0FLAG, PP(10*20)
DO 1 I=1, NCOMP
SUM = SUM + PP(I,1) * S(IG, IS, I*5)
1 CONTINUE
MW = SUM
RETURN
END
FUNCTION SG (IG, IS)
COMMON /MAT/ MP(35,13), EP(35,10), S(2*45,11), EX(1)
COMMON /CON/ IU, NCOMP, NC5, H, NE, NS, NPR, NPOL, TMAX, IORDER, NGRAPH
COMMON /PTAB/ IGFLAG, PP(10, 20)
IF (IGFLAG.NE.1) GO TO 2
SUM = 0.
DO 1 I = 1, NCOMP
SUM = SUM + PP(I, 16) * S(IG, IS, I + 5)
CONTINUE
1 PROPS = S(IG, IS, 5) / (10.71 * S(IG, IS, 4))
SG = PROPS * SUM
RETURN

2 SUM = 0.
T = S(IG, IS, 4)
DO 3 I = 1, NCOMP
SUM = SUM + S(IG, IS, I + 5) * (PP(I, 17) * PP(I, 18) * T)
CONTINUE
3 SG = SUM
RETURN
END
FUNCTION ENTHL (IG,IS)
COMMON /MAT/ MP(35+13),EP(35+10),S(2+5+11),EX(1)
COMMON /CON/ IU,NCOMP,NCS,NHE,NS,NPR,NPOL,TMAX,IORDER,NGRAPH
COMMON /PTAB/ IGFLAG,PP(10+20)
T=SIG(IS+4)*PP(1+20)
ML=0
DO 1 I=1,NCOMP
ML=ML+SIG(IS,I+5)*((1./4.*PP(I+5)*T+1./3.*PP(I+4)))*T+1./2.*PP(I+13)*T*PP(I,2))
CONTINUE
ENTHL=HL*T
RETURN
END
FUNCTION ENTHV (IG, IS)
COMMON /MAT/ MP(35,13), EP(35,10), S(2,45,11), EX(1)
COMMON /CON/ IU, NCOMP, NC5, H, NE, WS, NPR, NPOL, TMAX, IORDER, NGRAPH
COMMON /PTAB/ IFLAG, PP(10, 20)
T=S(IG, IS, 4) * PP(1, 20)
HV=0.
DO 1 I=1, NCOMP
HV=HV+S(IG, IS, I+5) * (((PP(I, 15) * T*PP(I, 14)) * T*PP(I, 13)) * T*PP(I, 12))
1 CONTINUE
ENTHV=HV
RETURN
END
FUNCTION EQUIL (IL, IV)
COMM CON/MAT/ MP(35,13), EP(35,10), S(2,45,11), EX(1)
COMM CON/ V, NCOMP, NS, H, NERROR, NORDER, NGRAPH
COMM CON/PTAB, IGFLAG, PP(10,20)
COMM CON/KVALU, HKK(10)

C
DUMMY STATEMENT
EQUIL=0.0
T=S(IG, IL+4)*PP(1,20)
IF (T.EQ.0.) T=100.
K=0
1 CONTINUE
K=K+1
IF (K.GE.50) PRINT 3, T, SUM, IL, IV
IF (K.GE.50) PRINT 4, (S(IG, IL, J)*S(IG, IV, J), J=1,10)
IF (K.GE.50) RETURN
DUM=KVAL(II, IV)
SUM=0.0
SUMY=0.0
DO 2 NC=6, NC5
N=NC5
S(IG, IV, NC)=HKK(N)*S(IG, IL, NC)
SUM=SUM*S(IG, IV, NC)
DY=S(IG, IV, NC)*PP(N,7)/(T*PP(N, 8))**2
SUMY=SUMY+DY
2 CONTINUE
YER=1.-SUM
T=T/YER/SUMY
S(IG, IL, 4)=T*PP(1,20)
IF (ABS(YER).GT.,0.01) GO TO 1
S(IG, IL, 4)=T-PP(1,20)
S(IG, IV, 4)=S(IG, IL, 4)
RETURN

C
3 FORMAT (** ITER, SUM, IL, IV **, 2E15.5, 215)
4 FORMAT (2E15.5)
END
REAL FUNCTION KVAL(IL,IV)
COMMON /MAT/ MP(35,13),EP(35,10),S(2,45,11),EX(1)
COMMON /CON/ IG,NCOMP,NC5,H,NE,NS,NPR,NPOL,THAX,ORDER,NGRAPH
COMMON /PTAB/ IFLAG,PP(10,20)
COMMON /KVALU/ HKK(10)
C
DUMMY STATEMENT
KVAL=0.0
T=S(IG,IL,4)*PP(I,20)
P=S(IG,IL,5)
DUM=ACTY(IL)
DO 1 I=1,NCOMP
HKK(I)=EXP(PP(I,6)*PP(I,7)/(T*PP(I,8)))*PP(I,9)/P
CONTINUE
RETURN
END
FUNCTION ACTY(IS)

COMMON /MAT/ MP(35,13), EP(35,10), S(2,45,11), EX(1)
COMMON /CON/ IU, NCOMP, NCS, H, NE, NS, NPR, NPO, TMAX, IORDER, NGRAPH.
COMMON /PTAB/ IFLAG, PP(10,20)

C DUMMY STATEMENT

ACTY=0.0

DO 1 I=1, NCOMP

PP(I+9)=1.

1 CONTINUE

RETURN

END
FUNCTION TEMPE (Q, MASS, N)
REAL: MASS
COMMON /MAT/ MP(35,13), EP(35,10), S(2,45,11), EX(1)
COMMON /CON/ IG, NCOMP, NC5, H, NE, NS, NPR, NPOL, TMAX, IORDER, NGRAPH
COMMON /PTAB/ IGFLAG, PP(10,20)
T= S(IG, N+4) * PP(1, 20)
IF (Q.LE.0., AND, MASS.LE.0.) GO TO 3
X1=0.
X2=0.
X3=0.
X4=0.
DO 1 J=1, NCOMP
  J=J+5
  X1=X1+S(IG, N+J) * PP(I, 2)
  X2=X2+S(IG, N+J) * PP(I, 3) / 2
  X3=X3+S(IG, N+J) * PP(I, 4) / 3
  X4=X4+S(IG, N+J) * PP(I, 5) / 4
1 CONTINUE
CONTINUE
F=Q-(((X4*T+X3)*T+X2)*T+X1)*T
DF=((4.*X4*T+3.*X3)*T+2.*X2)*T+X1
T2=T-DF/F
IF (ABS(T-T2) .LT. .01) GO TO 3
T=T2
GO TO 2
TEMPE=T-PP(1, 20)
RETURN
END
SUBROUTINE DIFSUB (N, YY, DERY)
COMMON /GEAR1/ Y(8,222), SAVE(10,222), ERROR(222), YMAX(222), DERY1(222)
12, DERY2(222)
COMMON /GEAR2/ EPS, TIME, KFLG, JSTART, NBVMAX, ICONV, NOPPT, ISTITF
COMMON /CON/ IG, NCOMP, NC5, N, NS, NPR, NPOL, TMAX, IORDER, NGRAF
COMMON /GEAR3/ HMIN, HMAX, XMIN, NOMES
COMMON /BKV/, NBV
COMMON /ROW/, IROW(1887)
COMMON /COLUMN/, JCOL(1887)
COMMON /JACOB/, PW(2109)
COMMON /SUBD/, ASUB(49)
COMMON /DIAG/, BDIA(49)
COMMON /SUPERD/, CSUP(49)
COMMON /MODULE/, IDERY, ITER, ITRI, NZERO
DIMENSION A(8), PERTST(7, 2, 3)
DIMENSION NOP(1000), NW(500), PWSAVE(200)
DIMENSION NOPS(1000)
DIMENSION YY(222), DERY(222)
DATA ITRI, IDERY/2*0/
DATA PERTST/2.0, 4.5, 7.3, 33, 10.42, 13.7, 17.15, 1.0, 2.0, 12.0, 24.0, 37.8/
DATA 1.53, 3.7, 0.8, 87.9, 7.3, 0.6, 0.9, 1.6, 12.5, 15.9, 1.0, 1.0, 12.0, 0.24, 0.37, 8/
DATA 29.5, 3.3, 1.0, 0.8, 87.9, 1.0, 1.0, 1.0, 0.5, 0.16, 7, 0.0, 133, 0.0, 826, 7, 1.0, 0.1/
DATA 30, 1.0, 2.0, 1.0, 0.3, 157, 0.0, 7, 0.0, 7, 0.0, 139/
DATA A(2)/1.0/
IF (IDER, EQ.1) GO TO 52
NBV1=NBV+1
NBV=NBV+N
IF (NBVMAX.LT.NBV) NBVMAX=NBV
DO 1 I=NBV1+NBV
II=I+NBV1+1
DERY(I)=DERY1(I)
1 CONTINUE
IF (IG.EQ.1) GO TO 26

C PREDICTOR SECTION
C
4 IRET=1
IF (JSTART.EQ.0) GO TO 4
DO 3 J=1, K
SAVE(J, I)=Y(J, I)
3 HOLD=H
NQOLD=Q
RACUM=1.0
IF (JSTART.GT.0) GO TO 23
GO TO 6
4 CONTINUE
N=1
N=N+1
DO 5 I=NBV1+NBV
II=I+NBV1+1
5
Y(1,I)=YY(II)
Y(2,I)=DERY1(I)*H
CONTINUE
HNEW=H
K=2
GO TO 2
IF (ISTIFF,EQ.0) GO TO 7
IF (NQ,GT,6,OR,NQ,LT,1) GO TO 8
GO TO (16,17,18,19,20,21), NQ
IF (NQ,GT,7,OR,NQ,LT,1) GO TO 8
GO TO (9,10,11,12,13,14,15), NQ
PRINT 84, NQ
STOP
C
C NONSTIFF COEFFICIENTS, ORDER 1-7
C
9
A(1)=1.0
GO TO 22
10
A(1)=-0.50000000
A(3)=-0.50000000
GO TO 22
11
A(1)=41666666666667
A(3)=75000000
A(4)=16666666666667
GO TO 22
12
A(1)=0.37500000
A(3)=0.916666666667
A(4)=0.3333333333333
A(5)=4166666666667E-01
GO TO 22
13
A(1)=0.3486111111111
A(3)=1.041666666667
A(4)=4861111111111
A(5)=1041666666667
A(6)=833333333333E-02
GO TO 22
14
A(1)=0.3298611111111
A(3)=1.141666666667
A(4)=625000000
A(5)=17708333333333
A(6)=250000000
A(7)=138888888888889E-02
GO TO 22
15
A(1)=0.31559193121693
A(3)=1.235000000
A(4)=75185185185
A(5)=2552083333333
A(6)=486111111111111E-01
A(7)=486111111111111E-02
A(8)=1984126984127E-03
GO TO 22
C
STIFF COEFFICIENTS, ORDER 1-6

A(1) = -1.00000000
GO TO 22

A(1) = 0.6666666666
A(3) = 0.333333333
GO TO 22

A(1) = 0.545454545454545
A(3) = A(1)
A(4) = -0.909090909091
GO TO 22

A(1) = 0.48000000
A(3) = 0.70000000
A(4) = 0.20000000
A(5) = 0.02000000
GO TO 22

A(1) = 0.43795620437956
A(3) = 0.8211678321169
A(4) = 0.31021897810219
A(5) = 0.54744525547445E-01
A(6) = 0.36496350364964E-02
GO TO 22

A(1) = 0.40816326360612
A(3) = 0.92063492063492
A(4) = 0.416666666666
A(5) = 0.99206349206349E-01
A(6) = 0.11904761904762E-01
A(7) = 0.56689342403628E-03

K = NQ + 1
IDOUB = K
MTYP = 2 - ISTIFF
ENQ2 = 0.5/FLOAT(NQ+1)
ENQ3 = 0.5/FLOAT(NQ+2)
ENQ1 = 0.5/FLOAT(NQ)
EU = PERTST(NQ,MTYP,2)*EPS)**2
E = PERTST(NQ,MTYP,1)*EPS)**2
EDNW = PERTST(NQ,MTYP,3)*EPS)**2
IF (EDNW.EQ.0) GO TO 83
IF (IRET.EQ.2) GO TO 75
IF (IRET.EQ.3) RETURN

CONTINUE
DO 24 J = 2, K
DO 24 J1 = J, K
J2 = J1 + J1 - 1
DO 24 I = NBV1, NBV
24 Y(J2+I) = Y(J2+I) + Y(J2+1, I)
DO 25 I = 1, N
II = I + NBV1 - 1
YY(I) = Y(1, II)
CONTINUE
RETURN

CONTINUE
CORRECTOR SECTION

DO 27 I=NBV1,NBV.
ERROR(I)=0.0
27 CONTINUE
IWEVAL=ISTIFF
L=1
28 CONTINUE

ITER=0 = DIRECT ITERATION OF CORRECTOR
ITER=1 = NEWTON-RAPHSON ITERATION OF CORRECTOR

IF (ITER.EQ.0 .OR. ISTITF.EQ.0) GO TO 41
R=A(I)*H
IF (ITRI.EQ.1) GO TO 43

TRGB OPTION

IF (IWEVAL.EQ.-1) GO TO 30
DO 29 I=1,NZERO
PW(I)=PW(I)*R
IF (IROW(I).EQ.JCOL(I)) PW(I)=1.0*PW(I)
29 CONTINUE
IWEVAL=-1
30 DO 31 I=1,N
IROW(I+NZERO)=I
JCOL(I+NZERO)=0
II=I+NBV1=1
PW(I+NZERO)=Y(2,II)-DERY1(II)*H
31 CONTINUE
NEM=NZERO+N
DO 32 I=1,NEM
PWSAVE(I)=PW(I)
32 CONTINUE
IF (JSTART.EQ.0) GO TO 34
NOP2=NOPS(NOPPT+2)
DO 33 I=1,NOP2
II=NOPPT+I
NOP(I)=NOPS(II)
33 CONTINUE
CALL TRGB2 (NOP,PW,XMIN,INTEST,NELEM)
IF (NELEM.EQ.0) GO TO 38
IF (NOMES.EQ.0) PRINT 85,NELEM,NBV1,NBV
34 CALL TRGB (NOP,NW,PW,JCOL,IROW,N,N,NEM,XMIN)
IF (JSTART.EQ.0) GO TO 35
NOP(2)=NOPS(NOPPT+2)
GO TO 36
35 CONTINUE

INCREASE LENGTH OF OPERATOR LIST BY 10 PERCENT IN CASE IT MUST BE RE-CREATED
C

NOP(2)=NOP(2)*NOP(2)/10
36  CONTINUE

NOP2 NOP(2)
DO 37 I=1,NOP2
II=I+NOP2
NOP2=II/NOP(2)
37  CONTINUE

DO 39 I=1,NEM
PW(I)=PWSAVE(I)
39  CONTINUE

NOP3 NOP(3)
DO 40 I=1,N
II+1+NBV1=I
SAVE(9,II)=PW(NOP3*I)
40  CONTINUE

GO TO 49

C

DIRECT ITERATION OF CORRECTOR

C

41  CONTINUE

DO 42 I=NBV1,NBV
SAVE(9,I)=Y(2,I)-DERY1(I)*H
42  CONTINUE

GO TO 49

C

TRIDIAGONAL OPTION

C

43  CONTINUE

IF (IWEVAL.EQ.-1) GO TO 46
DO 44 I=1,N
ASUB(I)=ASUB(I)*R
BDIAG(I)=BDIAG(I)*R
CSUP(I)=CSUP(I)*R
44  CONTINUE

DO 45 I=1,N
BDIAG(I)=1.0+BDIAG(I)
45  CONTINUE

IWEVAL=-1
46  CONTINUE

DO 47 I=NBV1,NBV
II=I-NBV1+1
DERY2(I)=Y(2,I)-DERY1(I)*H
47  CONTINUE

CALL TRIDAG (N,ASUB,BDIAG,CSUP,DERY2,YY,DERY,PWSAVE)

DO 48 I=1,N
SAVE(9,I)=YY(I)
48  CONTINUE

ITRI=0
49  NT=N

BND=EPS*ENQ3/FLOAT(N)
DO 50 I=NBV1, NBV
Y(1,I)=Y(1,I)+A(I)*SAVE(9,I)
Y(2,I)=Y(2,I)+SAVE(9,I)
ERROR(I)=ERROR(I)*SAVE(9,I)
IF (ABS(SAVE(9,I)) LE (BND*YMAX(I))) NT=NT+1
50 CONTINUE
DO 51 I=1*N
II=I+NBV1-1
YY(I)=Y(1,II)
51 CONTINUE
IF (NT LE 0) GO TO 56
IF (L EQ 3) GO TO 54
C
C RETURN TO MODULE TO GET NEW DERIVATIVE VALUES
C
IDERY=1
RETURN
52 IDERY=0
DO 53 I=NBV1, NBV
II=I-NBV1+1
DERY(I)=DERY(II)
53 CONTINUE
L=L+1
IF (L LT 4) GO TO 28
54 CONTINUE
ICONV=1
RACUM=RACUM+0.25
IF (NOWAE.EQ.0) PRINT 86, NBV1, NBV
IF (H LE (HMIN*1.00001)) GO TO 55
GO TO 80
55 PRINT 87, NBV1, NBV
STOP
56 CONTINUE
IF (ITRI.EQ.0 .AND. ITER.EQ.1) NOPPT=NOPPT+NOP(2)
IF (NBV, NE, NBVMAX) RETURN
C
C ERROR SECTION
C
D=0.0
DO 57 I=1, NBVMAX
D=D+ERROR(I)/YMAX(I)**2
57 CONTINUE
IF (D GT .E) GO TO 61
IF (K LT 3) GO TO 59
DO 58 J=3, K
DO 58 I=1, NBVMAX
58 Y(J,I)=Y(J,I)+A(J)*ERROR(I)
59 CONTINUE
IF (IDOUB.LE.1) GO TO 62
IDOUB=IDOUB+1
IF (IDOUB.GT.1) GO TO 77
DO 60 I=1, NBVMAX
SAVE(10*I) = ERROR(I)
CONTINUE
GO TO 77

KFLAG = KFLAG - 2
IF (NAMES.EQ.0) PRINT 88
IF (H.EQ.(MIN*1.00001)) GO TO 79
PR2 = (D/E)**ENQ2*1.2
PR3 = 1.0E+20
IF ((NQ.GE.IORDER).OR.(KFLAG.LE.-1)) GO TO 64
D = 0.0
DO 63 I = 1,NBVMAX
D = D + (ERROR(I) - SAVE(10,I))/YMAX(I)**2
CONTINUE
PR3 = (D/EUP)**ENQ3*1.4
PR1 = 1.0E+20
IF (NQ.LE.1) GO TO 66
D = 0.0
DO 65 I = 1,NBVMAX
D = D + (Y(K*I)/YMAX(I))**2
CONTINUE
PR1 = (D/EDWN)**ENQ1*1.3
CONTINUE
IF (PR2.LE.PR3) GO TO 72
IF (PR3.LT.PR1) GO TO 73
R = 1.0/AMAX1(PR1,1.0E-04)
NEWQ = NO - 1
69 I = 10
IF ((KFLAG.EQ.1).AND.(R.LT.1.1)) GO TO 77
IF (NEWQ.LE.NO) GO TO 70
DO 69 I = 1,NBVMAX
Y(NEWQ+1*I) = ERROR(I)*A(K)/FLOAT(K)
CONTINUE
K = NEWQ + 1
IF (KFLAG.EQ.1) GO TO 74
RACUM = RACUM*R
IRET = 3
GO TO 80
CONTINUE
71 IRET = 0
IF (NEWQ.EQ.NO) RETURN
Q = NEWQ
IRET = 3
GO TO 6
IF (PR2.GT.PR1) GO TO 67
NEWQ = NO
R = 1.0/AMAX1(PR2,1.0E-04)
GO TO 68
R = 1.0/AMAX1(PR3,1.0E-04)
NEWQ = NO + 1
GO TO 68
CONTINUE
R = AHIN1(R*HMAX/ABS(H))
H=H*R  
IF (NQ.EQ.NEWQ) GO TO 75  
NQ=NEWQ  
IRET=2  
GO TO 6  
75  
R1=1.0  
DO 76 J=2,K  
R1=R1*R  
DO 76 I=1,NBVMAX  
Y(J,I)=Y(J,I)*R1  
IDOUB=K  
77  
CONTINUE  
DO 78 I=1,NBVMAX  
YMAX(I)=AMAX1(YMAX(I),ABS(Y(1,I)))  
78  
CONTINUE  
JSTART=NQ  
RETURN  
79  
PRINT A9  
STOP  
80  
CONTINUE  
RACUM=AMAX1(ABS(HMIN/HOLD),RACUM)  
RACUM=AMIN(RACUM,ABS(HMAX/HOLD))  
H=HOLD*RACUM  
R1=1.0  
DO 81 J=2,K  
R1=R1*RACUM  
DO 81 I=1,NBVMAX  
Y(J,I)=SAVE(J,I)*R1  
DO 82 I=1,NBVMAX  
Y(I,I)=SAVE(I,I)  
82  
CONTINUE  
IDOUB=K  
IF (IRET1.EQ.3) GO TO 81  
RETURN  
83  
PRINT 90  
STOP  
C  
84  
FORMAT (41H THE MAXIMUM ORDER SPECIFIED IS TOO LARGE*,I5)  
85  
FORMAT (43H THE CORRECTOR CANNOT BE SOLVED BECAUSE PWY*,I4*,I4*7H ) ISDIF 403  
1 A PIVOT ELEMENT WHOSE VALUE IS BELOW XMIN/Y,23H DIFFERENTIAL EQUADIF 404  
2TIONS*,I4*,3H TO,I4*,/,33H OPERATOR LIST WILL BE RE-CREATED)  
86  
FORMAT (49H THE CORRECTOR FAILED TO CONVERGE IN 3 ITERATIONS*,23H DDIF 406  
1DIFFERENTIAL EQUATIONS*,I4*,3H TO,I4*)  
87  
FORMAT (58H CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED FOR M.GT.HOLDIF 408  
1MIN/Y,23H DIFFERENTIAL EQUATIONS*,I4*,3H TO,I4*)  
88  
FORMAT (30H TRUNCATION ERROR IS TOO LARGE)  
89  
FORMAT (72H THE STEP WAS TAKEN WITH M.MIN BUT THE REQUESTED ERRORDIF 411  
1 WAS NOT ACHIEVED)  
90  
FORMAT (68H THE REQUESTED ERROR IS SMALLER THAN CAN BE HANDLED FORDIF 413  
1 THIS PROBLEM)  
END
SUBROUTINE TRIDAG (N,A,B,C,D,V,BETA,GAMMA)

SUBROUTINE FOR SOLVING A SYSTEM OF N LINEAR SIMULTANEOUS EQUATIONS
HAVING A TRIDIAGONAL COEFFICIENT MATRIX.

SOURCE - APPLIED NUMERICAL METHODS, CARNAHAN LUTHER, & WILKES, 1969

ARGUMENTS

N - NUMBER OF EQUATIONS
A - SUBDIAGONAL COEFFICIENTS ARE STORED IN A(2)...A(N)
   A(1) IS NOT USED
B - DIAGONAL COEFFICIENTS
C - SUPERDIAGONAL COEFFICIENTS ARE STORED IN C(1)...C(N-1)
   C(N) IS NOT USED
D - RIGHT HAND SIDE VECTOR
V - SOLUTION VECTOR
BETA,GAMMA - INTERMEDIATE ARRAYS

DIMENSIONS - ALL ARGUMENTS EXCEPT N MUST BE DIMENSIONED AT LEAST
N IN CALLING PROGRAM

DIMENSION A(N), B(N), C(N), D(N), V(N), BETA(N), GAMMA(N)

COMPUTE INTERMEDIATE ARRAYS BETA AND GAMMA

BETA(1)=B(1)
GAMMA(1)=D(1)/BETA(1)
DO 1 I=2,N
   BETA(I)=B(I)-A(I)*C(I-1)/BETA(I-1)
   GAMMA(I)=(D(I)-A(I)*GAMMA(I-1))/BETA(I)
1 CONTINUE

COMPUTE FINAL SOLUTION VECTOR V

V(N)=GAMMA(N)
LAST=N-1
DO 2 K=LAST,N-1
   I=K-1
   V(I)=GAMMA(I)*C(I)*V(I+1)/BETA(I)
2 CONTINUE

RETURN
END
FIGURE 1.21 SKELETON OF MODULE FOR DYNSYS 2.1

SUBROUTINE TYPE2.1

COMMENTS DESCRIBING MODULE

THE FOLLOWING COMMON BLOCKS ARE OR MAY BE REQUIRED:
MAT,CON,PTAB,UNIT,GEAR2,MODULE,ROW,COLUMN,JACOB,SUBDI,DIAG,SUPERD

COMMON/MAT/MP(I,J),EP(I),S(I,J),EX(I)
COMMON/CON/IG,NCOMP,NC5,H,NE,NS,NPR,NPOL,TMAX,IORDER,NGRAPH
COMMON/PTAB/IGFLAG,PP(I)
COMMON/UNIT/IM,NMP
COMMON/GEAR2/EP,H,M,NP,MAX,ICOND,ISTIFF
COMMON/MODULE/IDERY,ITER,ITRI,MC,IPIVOT

THE ROW DIMENSION OF JCOL AND XJACOB MUST BE THE SAME AS IN DIFSUB
MC IS MAXIMUM NUMBER OF COLUMNS IN XJACOB AND JCOL

COMMON/COLUMN/JCOL(I,MC) (NORMAL OPTION)
COMMON/JACOB/XJACOB(I,MC) (NORMAL OPTION)

N IS NUMBER OF ODES

COMMON/SUBDI/A(N) (TRIDIAGONAL OPTION)
COMMON/DIAG/B(N) (TRIDIAGONAL OPTION)
COMMON/SUPERD/C(N) (TRIDIAGONAL OPTION)

******************************************************************************

SECTION #1 : PARAMETER CALCULATIONS

******************************************************************************

CALCULATE MODULE PARAMETERS : STREAM INPUT, INITIAL CONDITIONS,
VALUES OF ITER ETC.

INPUT STREAM INFORMATION IS OBTAINED FROM S(IG,I)
VALUE OF INDEPENDENT VARIABLE (Y) NEED ONLY BE SPECIFIED ON FIRST
INTEGRATION STEP (I.E., INITIAL CONDITIONS)
JSTART=0 ON FIRST INTEGRATION STEP
CURRENT ORDER OF INTEGRATION TECHNIQUE ON LATER STEPS

IF I STIFF=0, NONSTIFF COEFFICIENTS WILL BE USED IN INTEGRATION ALGORITHM
(DIFSUB) FOR ALL MODULES;
JACOBIAN MATRIX IS NOT REQUIRED

IF I STIFF=1, STIFF COEFFICIENTS WILL BE USED FOR ALL MODULES
THEN, IF I ITER=0 DIRECT ITERATION OF CORRECTOR WILL BE USED
(NONSTIFF MODULE)
JACOBIAN IS NOT REQUIRED

IF I ITER=1, NEWTON-RAPHSON ITERATION OF CORRECTOR WILL
BE USED (STIFF MODULE)
JACOBIAN MATRIX MUST BE SUPPLIED

ITER MUST BE SPECIFIED 0 OR 1
IT IS NOT USED UNLESS ISTIFF=1

IF(IG.EQ.2) GO TO 2

SECTION #2: JACOBIAN EVALUATION

SECTION #2 IS OMITTED FOR NONSTIFF MODULE

JACOBIAN MATRIX IS REQUIRED ONLY IF MODULE EQUATIONS ARE STIFF
CALCULATE JACOBIAN MATRIX ON CORRECTOR PASS ONLY
JACOBIAN NEED NOT BE VERY ACCURATE, AS IT IS USED ONLY FOR CONVERGENCE
OF CORRECTOR
ONLY NONZERO ELEMENTS ARE CALCULATED, BUT ALL DIAGONAL ELEMENTS MUST
BE STORED WHETHER OR NOT THEY ARE ZERO (NORMAL OPTION)

NORMAL OPTION

IF JACOBIAN MATRIX IS NOT TRIDIAGONAL, KEYS STORAGE SCHEME IS USED
COLUMN NUMBERS OF NONZERO ELEMENTS IN ROW I ARE STORED IN
(JCOL(I,J),J=1,NUMBER_OF_NONZEROS_IN_THAT_ROW)
SIMILARLY THE VALUES OF THE NONZERO ELEMENTS OF ROW I ARE
STORED IN XJACOB(I, )
REMEMBER TO STORE ALL DIAGONAL ELEMENTS EVEN IF THEY ARE ZERO

MC = MAXIMUM NUMBER OF COLUMNS IN XJACOB MATRIX
MC CAN BE LARGER THAN THE MAXIMUM NUMBER OF NONZERO ELEMENTS
IN ANY JACOBIAN ROW SINCE ADDITIONAL NONZERO ELEMENTS CAN BE
CREATED DURING THE SOLUTION. MC IS CONSTANT FOR ANY SET OF ODES
AND MAY HAVE TO BE DETERMINED BY TRIAL AND ERROR

IPIVOT = PIVOT OPTION USED IN SIMULT (1-7)
IPIVOT=1 : SIMPLE GAUSS-JORDAN ELIMINATION
IPIVOT=2 : GAUSS-JORDAN PARTIAL PIVOTING
IPIVOT=3 : GAUSS-JORDAN FULL PIVOTING
IPIVOT=4 : MINIMUM ROW-MINIMUM COLUMN
IPIVOT=5 : MINIMUM COLUMN-MINIMUM ROW
IPIVOT=6 : MAXIMUM COLUMN-MINIMUM ROW
IPIVOT=7 : MINIMUM OF ROW ENTRIES TIMES COLUMN ENTRIES

ICOL(I,J) = COLUMN NUMBER OF JTH NONZERO ELEMENT IN ROW I
XJACOB(I,J) = VALUE OF JTH NONZERO ELEMENT OF ROW I

MC=
IPIVOT=
IROW(1:1)=
XJACOB(1:1)=

"
TRIDIAGONAL OPTION

FOR TRIDIAGONAL JACOBIAN MATRIX, THE SUBDIAGONAL, DIAGONAL AND SUPERDIAGONAL ELEMENT VALUES ARE STORED IN ARRAYS A, B AND C FROM COMMON BLOCKS SUBDI, DIAG AND SUPERD RESPECTIVELY.

N = NUMBER OF ODES
A = VALUES OF SUBDIAGONAL ELEMENTS ARE STORED IN A(2) ... A(N)
    A(1) IS NOT USED
B = VALUES OF DIAGONAL ELEMENTS
C = VALUES OF SUPERDIAGONAL ELEMENTS ARE STORED IN C(1) ... C(N-1)
    C(N) IS NOT USED

B(1) =
C(1) =
A(2) =
B(2) =
C(2) =

A(N-1) =
B(N-1) =
C(N-1) =
A(N) =
B(N) =
1 CONTINUE

IF TRIDIAGONAL OPTION IS BEING USED, ITRI MUST BE SET TO 1 HERE.
IF NORMAL OPTION IS USED ITRI MAY BE IGNORED

ITRI = 1

2 CONTINUE

SECTION #3 : DERIVATIVE CALCULATION

CALCULATE DERIVATIVES

DERY(1) =
    :
    :
DERY(N) =
SECTION #4: CALL DIFSUB

CALL DIFSUB TO SOLVE ODES FOR MODULE
DIFSUB MAY BE CALLED MORE THAN ONCE FROM A MODULE FOR EXAMPLE IF THE
MODULE CONTAINS A SET OF STIFF O.D.E.S AND ANOTHER SET OF NONSTIFF
O.D.E.S

ARGUMENTS:
N - NUMBER OF ODES
Y - INDEPENDENT VARIABLE
DERY - DERIVATIVES

CALL DIFSUB(N,Y,DERY)

IF IDERY IS NOT ZERO, THE DERIVATIVES WILL BE RE-EVALUATED AND
RETURNED TO DIFSUB
ITRI MUST ALSO BE RESET IF IT IS 1

IF(IDERY.NE.0) GO TO 1

SECTION #5: STREAM OUTPUT CALCULATION

CALCULATE STREAM OUTPUT (STORED IN S(1, ))

S(1, )=

: :

RETURN
END
FIGURE I.3: LISTING OF DYNSYS 2.1 EXECUTIVE
(ROUTINES IDENTICAL OR NEARLY IDENTICAL WITH THOSE
IN DYNSYS 2.0 ARE NOT LISTED)

SUBROUTINE DYN1

DATA LOADING AND PRINTING
COMMON /UNIT/ IM*NMP
COMMON /MAT/ MP(15,5),EP(15,5),S(2,16,7),EX(1)
COMMON /CON/ IG,NCOMP,NC5,NX,NS,NPR,NPOL,TMAX,ORDER,NGRAPH
COMMON /FTN/ FN(1,1),NFN
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBVMAX,ICONV,ISTIFF
COMMON /GEAR3/ HMIN,HMAX,NOMES
DIMENSION AM(100)
DIMENSION ITAG(30)
DATA ITAG/SHVALV1,SHVALV2,SHSTIR1,SHCONT,SHSETL1,6HMXPLT1,5HSETT1,DY1111
1,6HVSS1,5HDELAY,5HSTR1,5HECH1,5HDAN1,5HCONV1,5HSPLT1,16*6H
2
WRITE (6,27)

DEFAULT VALUES
TIME=0.0
MAXNE=20
NPR=1
TMAX=10.0
NCOMP=1
NMP=5
M=0.00001
HMIN=0.000001
HMAX=0.05
EPS=0.001
NPOL=0
NGRAPH=0
ORDER=6
ISTIFF=1
NOMES=0

READ AND COPY TITLE
READ (5,29) (AM(I),I=1,8)
WRITE (6,28) (AM(I),I=1,8)
IF (AM(1).NE.5HBEGIN) GO TO 1

READ INITIAL DATA
CALL GET (N*X,1)
IF (N.EQ.5HCOMPS) NCOMP=X
IF (N.EQ.6HIN/OUT) NMP=X
IF (N.EQ.6HDELTAT) M=X
IF (N.EQ.4HMIN) HMIN=X
IF (N.EQ.4HMAX) HMAX=X
IF (N.EQ.4HTIME) TMAX=X
IF (N.EQ.8HPRINTING) NPR=X
IF (N.EQ.9HTOLERANCE) EPS=X
IF (N.EQ.5HORDER) ORDER=X
IF (N.EQ.8HLINEPLOT) NPOL=1
IF (N.EQ.5HGRAPH) NGRAPH=1
IF (N.EQ.8) N=0
IF (N.EQ.9) N=1
IF (N.EQ.7) N=2
IF (N.EQ.6) N=3
IF (N.EQ.5) N=4
IF (N.EQ.4) N=5
IF (N.EQ.3) N=6
IF (N.EQ.2) N=7
IF (N.EQ.1) N=8
IF (N.EQ.0) N=9

40 CALL GET (N,N)

10 CALL GET (N,N)

20 CALL GET (N,N)

30 CALL GET (N,N)

40 CALL GET (N,N)

50 CALL GET (N,N)

60 CALL GET (N,N)

70 CALL GET (N,N)

80 CALL GET (N,N)

90 CALL GET (N,N)

100 CALL GET (N,N)

110 CALL GET (N,N)

120 CALL GET (N,N)

130 CALL GET (N,N)

140 CALL GET (N,N)

150 CALL GET (N,N)

160 CALL GET (N,N)

170 CALL GET (N,N)

180 CALL GET (N,N)

190 CALL GET (N,N)

200 CALL GET (N,N)

210 CALL GET (N,N)

220 CALL GET (N,N)

230 CALL GET (N,N)

240 CALL GET (N,N)

250 CALL GET (N,N)

260 CALL GET (N,N)

270 CALL GET (N,N)

280 CALL GET (N,N)

290 CALL GET (N,N)

300 CALL GET (N,N)

310 CALL GET (N,N)

320 CALL GET (N,N)

330 CALL GET (N,N)

340 CALL GET (N,N)

350 CALL GET (N,N)

360 CALL GET (N,N)

370 CALL GET (N,N)

380 CALL GET (N,N)

390 CALL GET (N,N)

400 CALL GET (N,N)

410 CALL GET (N,N)

420 CALL GET (N,N)

430 CALL GET (N,N)

440 CALL GET (N,N)

450 CALL GET (N,N)

460 CALL GET (N,N)

470 CALL GET (N,N)

480 CALL GET (N,N)

490 CALL GET (N,N)

500 CALL GET (N,N)

510 CALL GET (N,N)

520 CALL GET (N,N)

530 CALL GET (N,N)

540 CALL GET (N,N)

550 CALL GET (N,N)

560 CALL GET (N,N)

570 CALL GET (N,N)

580 CALL GET (N,N)

590 CALL GET (N,N)

600 CALL GET (N,N)

610 CALL GET (N,N)

620 CALL GET (N,N)

630 CALL GET (N,N)

640 CALL GET (N,N)

650 CALL GET (N,N)

660 CALL GET (N,N)

670 CALL GET (N,N)

680 CALL GET (N,N)

690 CALL GET (N,N)

700 CALL GET (N,N)

710 CALL GET (N,N)

720 CALL GET (N,N)

730 CALL GET (N,N)

740 CALL GET (N,N)

750 CALL GET (N,N)

760 CALL GET (N,N)

770 CALL GET (N,N)

780 CALL GET (N,N)

790 CALL GET (N,N)

800 CALL GET (N,N)

810 CALL GET (N,N)

820 CALL GET (N,N)

830 CALL GET (N,N)

840 CALL GET (N,N)

850 CALL GET (N,N)

860 CALL GET (N,N)

870 CALL GET (N,N)

880 CALL GET (N,N)

890 CALL GET (N,N)

900 CALL GET (N,N)

910 CALL GET (N,N)

920 CALL GET (N,N)

930 CALL GET (N,N)

940 CALL GET (N,N)

950 CALL GET (N,N)

960 CALL GET (N,N)

970 CALL GET (N,N)

980 CALL GET (N,N)

990 CALL GET (N,N)
CALL GET (NM,X,0)
IF (NM,NE,5Hextra) GO TO 15
NNX=X
NNX=NNX*NEX=1
READ (5,32) (EX(J)+J=NNX,NNX)
WRITE (6,34) (EX(J)+J=NNX,NNX)
MP(NM,NMP+1)=NEX
NEX=NEX*NNX
CALL GET (NM,X,1)

15 IF (NM,NE,7HRESERVE) GO TO 11
NNX=X
NEX=NEX*NNX
GO TO 10

16 CALL FETCH (7HSTREAMS,NS)
WRITE (6,35) NS
DO 18 I=1,NS
S(I,I+1)=I
S(I,I+2)=1.0
S(I,I+3)=0.0
S(I,I+4)=530.0
S(I,I+5)=14.7
DO 17 J=6,NC5
S(I,I+J)=0
17 CONTINUE
18 CONTINUE
19 CALL GET (NM,X,-1)
IF (NM,NE,8HEXPLICIT) GO TO 20
IF (NM,NE,3HEHEND) GO TO 25
IF (NM,NE,4HSPEC) GO TO 22
GO TO 19
20 READ (5,36) IT,(AM(I),I=1,5)
IF (IT,NE,3HEHEND) GO TO 25
READ (5,32) (AM(I),I=6,NC5)
N=AM(1)
DO 21 I=1,NC5
S(I,N+1)=AM(I)
21 CONTINUE
GO TO 20

22 N1=X
READ (5,32) (AM(I),I=1,NC5)
N2=AM(1)
DO 24 I=N1,N2
DO 23 J=2,NC5
S(I,J)=AM(J)
23 CONTINUE
S(I,I+1)=I
24 CONTINUE
GO TO 19
25 DO 26 I=1,NS
K=S(I,I+2)
WRITE (6,38) I,K*(S(I,I+J),J=3,5)
WRITE (6,37) (S(I,I+J),J=6,NC5)
DO 26 J=1,NC5
26 S(2,I,J)=S(1,I,J)
XX=PROPS(0,0)
RETURN
C
27 FORMAT (1H1)
28 FORMAT (1X,8A10).
29 FORMAT (8A10)
30 FORMAT (1H1)
31 FORMAT (18H ***FAIL*** NAME *A12,8H NOT SET)
32 FORMAT (12X,5F12.0)
33 FORMAT (/9H ' UNIT -*I3,9H TYPE -*I3,1H-*12I5)
34 FORMAT (11X,5F14.5)
35 FORMAT (1H1,I6,8H STREAMS)
36 FORMAT (A3,9X,5F12.0)
37 FORMAT (8X,4F14.5)
38 FORMAT (11H STREAM -*I3,4H- (*I3,1H)*3F14.5)
39 FORMAT (12X,5F12.2)
END
SUBROUTINE DYN2
COMMON /UNIT/ IM,NMP
COMMON /MAT/ MP(15,5),EP(15,5),S(2,16,7),EX(1)
COMMON /CON/ IG,NCOMP,NC5,H,NE,NS,NPR,NPOL,TMAX,IOORDER,NGRAPH
COMMON /BKV/ NBV
COMMON /CNT/ NCTR
COMMON /LAG/ NSX,NSW
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBVMAX,ICONT,ISTIFF
NBV=0
NSX=0
NCTR=0
DO 31 IM=1,NE
IF (ICONT.EQ.1) RETURN
NTYPE=MP(IM,2)
GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23)
1 CALL TYPE1
   GO TO 31
2 CALL TYPE2
   GO TO 31
3 CALL TYPE3
   GO TO 31
4 CALL TYPE4
   GO TO 31
5 CALL TYPE5
   GO TO 31
6 CALL TYPE6
   GO TO 31
7 CALL TYPE7
   GO TO 31
8 CALL TYPE8
   GO TO 31
9 CALL TYPE9
   GO TO 31
10 CALL TYPE10
    GO TO 31
11 CALL TYPE11
    GO TO 31
12 CALL TYPE12
    GO TO 31
13 CALL TYPE13
    GO TO 31
14 CALL TYPE14
    GO TO 31
15 CALL TYPE15
    GO TO 31
16 CALL TYPE16
    GO TO 31
17 CALL TYPE17
    GO TO 31
18 CALL TYPE18
    GO TO 31
19 CALL TYPE19
   GO TO 31
20 CALL TYPE20
   GO TO 31
21 CALL TYPE21
   GO TO 31
22 CALL TYPE22
   GO TO 31
23 CALL TYPE23
   GO TO 31
24 CALL TYPE24
   GO TO 31
25 CALL TYPE25
   GO TO 31
26 CALL TYPE26
   GO TO 31
27 CALL TYPE27
   GO TO 31
28 CALL TYPE28
   GO TO 31
29 CALL TYPE29
   GO TO 31
30 CALL TYPE30
31 CONTINUE
   RETURN
   END
SURROUTINE DIFSUB (N,Y,DERY)
COMMON /GEAR1/ Y(8,222),SAVE(10,222),ERROR(222),YMAX(222),DERY1(222)
COMMON /GEAR2/ EPS,TIME,KFLAG,JSTART,NBMAX,ICOMV,ISTIFF
COMMON /CON/ IG,NCOMP,NC5,H,NE,NS,NPR,NPOL,TMAX,IOORDER,NGRAPH
COMMON /GEAR3/ HMIN,HMAX,NAMES
COMMON /BK/V,NBV
COMMON /COLUMN/ JCOL(10,10)
COMMON /JACOB/ PW(10,10)
COMMON /SUBDI/ ASUB(49)
COMMON /DIAG/ BDIAG(49)
COMMON /SUPERD/ CSUP(49)
COMMON /MODULE/ IDERY,ITER,ITRI,MC,IPIVOT
DIMENSION IEC(2), ICC(2)
DIMENSION PWSAVE(2,2), JCOLS(2,2)
DIMENSION A(8), PERTST(7,2,3)
DIMENSION YY(222), DERY(222)
DATA MR/10/
DATA ITRI,IDERY,2.0/
DATA PERTST/2.0,4.5,7.333,10.42,13.7,17.151,0,2.0,12.0,24.0,37.89/
1,53,33,70,08,87,97,3,0,6,0,9,167,12,5,15,98,1,0,1,0,1,0,12,0,24,0,37,8/
29,53,33,70,08,87,97,1,0,1,0,1,0,0,5,0,1667,0,0,4133,0,0,8267,1.0,1/
DATA 1,0,2,0,1,0,0,3157,0,0,7,0,7,0,0,139/
DATA A(2)=-1.0/
IF (IDERY,EQ,1) GO TO 47
NBV1=NBV+1
NBV=NBV+N
IF (NBVMAX,LT,NBV) NBVMAX=NBV
DO 1 I=NBV1,NBV
1 II=I-NBV1+1.
DERY1(I)=DERY(I)
CONTINUE
IF (IG,EQ,1) GO TO 26
C C
PREDECTOR SECTION
C C
IRET=1
IF (JSTART,EQ,0) GO TO 4
DO 3 I=NBV1,NBV
3 SAVE(J,1)=Y(J,1)
HOLD=H
NQLOD=NQ
RACUM=1.0
IF (JSTART,GT,0) GO TO 23
GO TO 6
CONTINUE
NQ=1
N=N+1
DO 5 I=NBV1,NBV
II=I-NBV1+1
5
Y(1,I)=YY(II)
Y(2,I)=DERY1(I)*H
5 CONTINUE
HNEW=H
K=2
GO TO 2
6 IF (ISTSF, EQ, 0) GO TO 7
IF (NQ, GT, 6, OR, NQ, LT, 1) GO TO 8
GO TO (16, 17, 18, 19, 20, 21), NQ
7 IF (NQ, GT, 7, OR, NQ, LT, 1) GO TO 8
GO TO (9, 10, 11, 12, 13, 14, 15), NQ
8 PRINT 79, NQ
STOP
C
C NONSTIFF COEFFICIENTS, ORDER 1=7
C
9 A(1)=-1.0
GO TO 22
10 A(1)=0.500000000
A(3)=0.500000000
GO TO 22
11 A(1)=0.416666666667
A(3)=0.750000000
A(4)=0.166666666667
GO TO 22
12 A(1)=0.375000000
A(3)=-0.916666666667
A(4)=-0.333333333333333
A(5)=0.416666666667E+01
GO TO 22
13 A(1)=0.348611111111111
A(3)=0.041666666667
A(4)=0.486111111111111
A(5)=0.104166666667
A(6)=0.833333333333333E+02
GO TO 22
14 A(1)=0.329861111111111
A(3)=1.141666666667
A(4)=0.625000000
A(5)=0.177083333333333
A(6)=0.025000000
A(7)=-0.1388888888888889E+02
GO TO 22
15 A(1)=0.31559193121693
A(3)=1.235000000
A(4)=0.75185185185
A(5)=0.255208333333333
A(6)=0.486111111111111E+01
A(7)=0.486111111111111E+02
A(8)=0.1984126984127E+03
GO TO 22
C
STIFF COEFFICIENTS, ORDER 1-6

16 A(1) = -1.00000000
GO TO 22

17 A(1) = 0.666666666666667
A(3) = -0.333333333333333
GO TO 22

18 A(1) = 0.545454545454545
A(3) = A(1)
A(4) = 0.9090909090909091E-01
GO TO 22

19 A(1) = 0.480000000
A(3) = 0.700000000
A(4) = 0.200000000
A(5) = 0.020000000
GO TO 22

20 A(1) = 0.43795620437956
A(3) = 0.82116788321169
A(4) = 0.310281897810219
A(5) = 0.54744525547445E+01
A(6) = 0.36496350364964E-02
GO TO 22

21 A(1) = 0.40816326530612
A(3) = 0.92063492063492
A(4) = 0.41666666666666
A(5) = 0.99206349206349E-01
A(6) = 0.11904761904762E-01
A(7) = 0.56689342403628E-03
GO TO 22

22 K = NQ+1
IDoub = K
MVT = 2 - ISTIFF
ENQ2 = 0.5/FLOAT(NQ+1)
ENQ3 = 0.5/FLOAT(NQ+2)
ENQ1 = 0.5/FLOAT(NQ)
EUP = (PERTST(NQ,MTYP,2)*EPS)**2
E = (PERTST(NQ,MTYP,1)*EPS)**2
EDWN = (PERTST(NQ,MTYP,3)*EPS)**2
IF (EDWN.EQ.0) GO TO 78
IF (IRET.EQ.2) GO TO 70
IF (IRET.EQ.3) RETURN

23 CONTINUE
DO 24 J = 2, K
DO 24 J1 = J, K
J2 = K - J1 + 1
DO 24 I = NBV1, NBV
24 Y(J2*I) = Y(J2*I) + Y(J2+1*I)
DO 25 I = 1, N
II = I + NBV1 - 1
YY(I) = Y(1*II)

25 CONTINUE
RETURN

26 CONTINUE
CORRECTOR SECTION

DO 27 I=NBV1,NBV
ERROR(I)=0.0
27 CONTINUE
IWEVAL=ISTIFF
L=1
28 CONTINUE

ITER=0 - DIRECT ITERATION OF CORRECTOR
ITER=1 - NEWTON-RAPHSON ITERATION OF CORRECTOR

IF (ITER.EQ.0 .OR. ISTIFF.EQ.0) GO TO 36
R=A(I).H
IF (ITRI.EQ.1) GO TO 38

SIMULT OPTION

IF (IWEVAL.EQ.-1) GO TO 31
DO 30 J=1,MC
DO 29 I=1,N
PM(I,J)=PM(I,J)*R
IF (JCOL(I,J).EQ.1) PM(I,J)=1.0*PM(I,J)
PWSAVE(I,J)=PW(I,J)
JCOLS(I,J)=JCOL(I,J)
29 CONTINUE
30 CONTINUE
IWEVAL=-1
31 DO 32 I=NBV1,NBV
II=I-NBV1+1
DERY2(II)=Y(2,I)-DERY1(I).H
32 CONTINUE
CALL SIMULT (N,MR,MC,PW,DERY2,JCOL,1.0E-10,ICC,IRC,IPIVOT)
DO 33 I=1,N
II=I-NBV1+1
SAVE(9,II)=PM(I,II)
33 CONTINUE

DO 35 J=1,MC
DO 34 I=1,N
PW(I,J)=PWSAVE(I,J)
JCOL(I,J)=JCOLS(I,J)
34 CONTINUE
35 CONTINUE
GO TO 44

DIRECT ITERATION OF CORRECTOR

36 CONTINUE
DO 37 I=NBV1,NBV
SAVE(9,I)=Y(2,I)-DERY1(I).H
37 CONTINUE
GO TO 44

C
C TRIDIAGONAL OPTION

38 CONTINUE
IF (IWEVAL.EQ.-1) GO TO 41
DO 39 I=1,N
ASUB(I)=ASUB(I)*R
BDIA(I)=BDIA(I)*R
CSUP(I)=CSUP(I)*R
39 CONTINUE
DO 40 I=1,N
BDIA(I)=B(I)*BDIA(I)
40 CONTINUE
IWEVAL=-1
41 CONTINUE
DO 42 I=NBV1, NBV
II=I-NBV1+1
DERY2(II)=Y(2,I)-DERY1(I)*H
42 CONTINUE
CALL TRIDAG (N,ASUB,BDIA,CSUP,DERY2,YY,DERY,PWSAVE)
DO 43 I=1,N
SAVE(9,I)=YY(I)
43 CONTINUE
ITRI=0
44 NT=N
BND= EPS*ENQ3/FLOAT(N)
DO 45 I=NBV1, NBV
Y(I,I)=Y(1,I)*A(I)*SAVE(9,I)
Y(2,I)=Y(2,I)-SAVE(9,I)
ERROR(I)=ERROR(I)*SAVE(9,I)
45 IF (ABS(SAVE(9,I)) .LE. (BND*YMAX(I))) NT=NT+1
CONTINUE
DO 46 I=1,N
II=I-NBV1+1
YY(I)=Y(I+II)
46 CONTINUE
IF (NT.LE.0) GO TO 51
IF (L.EQ.3) GO TO 49
C
C RETURN TO MODULE TO GET NEW DERIVATIVE VALUES

C
IDERY=1
RETURN

47 IDERY=0
DO 48 I=NBV1, NBV
II=I-NBV1+1
DERY(I)=DERY(I)
48 CONTINUE
L=L+1
IF (L.LT.4) GO TO 28
49 CONTINUE
ICONV=1
RACUM=RACUM*0.25
IF (HOMES.EQ.0).PRINT 80, NBV1,NBV
IF (H.LE.(HMIN+1.00001)) GO TO 50
GO TO 75
50 PRINT 81, NBV1,NBV
STOP
51 CONTINUE
IF (NBV.NE.NBVMAX) RETURN

C C ERROR SECTION
C
D=0.0
DO 52 I=1,NBVMAX
D=D+(ERROR(I)/YMAX(I))**2
52 CONTINUE
IF (D.GT.E) GO TO 56
IF (K.LT.3) GO TO 54
DO 53 J=3,K
DO 53 I=1,NBVMAX
Y(J,I)=Y(J,I)+A(J)*ERROR(I)
53 CONTINUE
IF (IDOUB.LE.1) GO TO 57
IDOUB=IDOUB-1
IF (IDOUB.GT.1) GO TO 72
DO 55 I=1,NBVMAX
SAVE(10,I)=ERROR(I)
55 CONTINUE
GO TO 72
56 KFLAG=KFLAG-2
IF (HOMES.EQ.0).PRINT 82
IF (H.LE.(HMIN+1.00001)) GO TO 74
57 PR2=(D/E)**ENQ2*1.2
PR3=1.0E+20
IF ((NO,GE,1,ORDER).OR.(KFLAG.LE.-1)) GO TO 59
D=0.0
DO 58 I=1,NBVMAX
D=D+(ERROR(I)-SAVE(10,I))/YMAX(I))**2
58 CONTINUE
PR3=(D/EUP)**ENQ3*1.4
59 PR1=1.0E+20
IF (NO.LE.1) GO TO 61
D=0.0
DO 60 I=1,NBVMAX
D=D+(Y(K+I)/YMAX(I))**2
60 CONTINUE
PR1=(D/EDWN)**ENQ1*1.3
61 CONTINUE
IF (PR2.LE.PR3) GO TO 67
IF (PR3.LT.PR1) GO TO 68
R=1.0/AMAX1(PR1,1.0E-04)
NEWQ=NO-1
63 IDOUB=10
64 IF ( (KFLAG EQ.1) .AND. (R LT 1.1)) GO TO 72
65 IF (NEWQ LE NQ) GO TO 65
66 DO 64 I=1, NBVMAX
67 Y (NEWQ+1,I) = ERROR(I) * A (K) / FLOAT(K)
68 CONTINUE
69 K=NEWQ+1
70 IF (KFLAG EQ.1) GO TO 69
71 RACUM = RACUM * R
72 IRET1 = 3
73 GO TO 75
74 CONTINUE
75 IRET1 = 0
76 IF (NEWQ EQ. NQ) RETURN
77 NQ = NEWQ
78 IRET = 3
79 GO TO 6
80 IF (PR2 GT PR1) GO TO 62
81 NEWQ = NQ
82 R = 1.0 / A MAX1 (PR2, 1.0E-04)
83 GO TO 69
84 R = 1.0 / A MAX1 (PR3, 1.0E-04)
85 NEWQ = NQ + 1
86 GO TO 63
87 CONTINUE
88 R = A MIN1 (R, HMAX / A BS(H))
89 H = H * R
90 IF (NQ EQ. NEWQ) GO TO 70
91 NQ = NEWQ
92 IRET = 2
93 GO TO 6
94 R1 = 1.0
95 DO 71 J = 2, K
96 R1 = R1 * R
97 DO 71 I = 1, NBVMAX
98 Y (J, I) = Y (J, I) * R1
99 IDOUB = K
100 CONTINUE
101 DO 73 I = 1, NBVMAX
102 YMAX (I) = A MAX1 (YMAX (I), A BS (Y (I, I)))
103 CONTINUE
104 JSTART = NQ
105 RETURN
106 PRINT A3
107 STOP
108 CONTINUE
109 RACUM = A MAX1 (ABS (HMIN / HOLD), RACUM)
110 RACUM = A MIN1 (RACUM, ABS (HMAX / HOLD))
111 H = HOLD * RACUM
112 R1 = 1.0
113 DO 76 J = 2, K
114 R1 = R1 * RACUM
115 STOP
DO 76 I=1,NBVMAX
76    Y(J,I)=SAVE(J,I)*B1
DO 77 I=1,NBVMAX
77    CONTINUE
    IDOUBK
    IF (IRET1.EQ.3) GO TO 66
    RETURN
78    PRINT 84
    STOP
C
79    FORMAT (41H THE MAXIMUM ORDER SPECIFIED IS TOO LARGE,i5)    DIF 365
80    FORMAT (49H THE CORRECTOR FAILED TO CONVERGE IN 3 ITERATIONS,/23H DIF 366
    1 DIFFERENTIAL EQUATIONS,i4,3H TO,i4)                DIF 367
81    FORMAT (58H CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED FOR M.GT.MDIF 368
    1MIN,/,23H DIFFERENTIAL EQUATIONS,i4,3H TO,i4)          DIF 369
82    FORMAT (30H TRUNCATION ERROR IS TOO LARGE)                  DIF 370
83    FORMAT (72H THE STEP WAS TAKEN WITH M=MMIN BUT THE REQUESTED ERROR DIF 371
    1 WAS NOT ACHIEVED)                                     DIF 372
84    FORMAT (68H THE REQUESTED ERROR IS SMALLER THAN CAN BE HANDLED FOR DIF 373
    THIS PROBLEM)                                           DIF 374
    END
SUBROUTINE SIMULT (NU, MR, MC, A, P, ICOLE, ZTEST, ICC, IRC, IPIVOT)

ARGUMENTS

NU : NUMBER OF Unknowns
MR : MAXIMUM ROW DIMENSION OF MATRIX A IN CALLING PROGRAM
MC : MAXIMUM NUMBER OF COLUMNS OF MATRIX A
A : MATRIX CONTAINING NONZERO ELEMENTS
P : RIGHT HAND SIDE VECTOR
ICOLE : RECORDS POSITION OF NONZERO ELEMENTS
ICC, IRC : DIMENSIONED IN CALLING PROGRAM AND USED ONLY IN SUBROUTINE SIMULT
IPIVOT : PIVOT OPTION 1-7
  1 : SIMPLEx GAUSS-JORDAN ELIMINATION
  2 : GAUSS-JORDAN PARTIAL PIVOTING
  3 : GAUSS-JORDAN FULL PIVOTING
  4 : MINIMUM ROW-MINIMUM COLUMN
  5 : MINIMUM COLUMN-MINIMUM ROW
  6 : MAXIMUM COLUMN-MAXIMUM ROW
  7-1 MINIMUM OF ROW ENTRIES TIMES COLUMN ENTRIES

SOLUTION VECTOR RETURNED IN FIRST COLUMN OF A : A(I, 1)

INTEGER PIVROW, PIVCOL, OPROW

DIMENSION IRC(NU), ICC(NU), A(MR, MC), ICOLE(MR, MC), P(NU)

INITIALIZE ROW ENTRY COUNTER
DO 1 I = 1, NU
  IRC(I) = 0
  CONTINUE
1

COUNT ROW AND COLUMN ENTRIES

JELE = 0
DO 3 I = 1, NU
  DO 2 J = 1, MC
    JCOL = ICOLE(J)
    IF (JCOL .EQ. 0) GO TO 3
    JELE = JELE + 1
    IRC(I) = JCOL
  2    CONTINUE
3

CONTINUE

JCOL = 0
JELEM = 0
DO 800 I = 1, NU
  800 IF (IRC(I) .GT. JCOL) JCOL = IRC(I)

CONTINUE

NORMALIZE PIVROW

X = A(PIVROW, IY)

IC = ICC(PIVROW)
DO 5 J = 1, IC

5
A(PIVROW+J)/A(PIVROW,J) /X  
CONTINUE  
A(PIVROW+Y) = 1.0  
P(PIVROW) = P(PIVROW) /X  
C  SELECT ROWS THAT CAN BE OPERATED ON  
DO 21 I = 1, NU  
IF (IRC(PIVCOL), EQ, 1) GO TO 22  
IF (I.EQ.PIVROW) GO TO 21  
IC = IABS(ICC(I))  
DO 20 J = 1, IC  
IF (ICOL(I+J) = PIVCOL) 20, 6, 21  
C IF YOU CAN GET TO THIS POINT OPROW CONTAINS PIVOTAL ELEMENT  
OPROW = I  
JKOP = 1  
JKPI = 1  
C C = A(OPROW, J)  
P(OPROW) = P(PIVROW) * C * P(OPROW)  
7 CONTINUE  
IF (ICOL(PIVROW, JKPI), EQ, 0) GO TO 21  
IF (ICOL(OPROW, JKOP), EQ, 0) GO TO 8  
IF (ICOL(PIVROW, JKPI) - ICOL(OPROW, JKOP)) = 12, 19  
C OPROW DOES NOT ContAIN THIS ELEMENT, ADD ELEMENT TO OPROW  
8 ICC(I) = ICC(I) + 1  
IF (ICC(I). LE. 0) ICC(I) = ICC(I) - 2  
II = IABS(ICC(I))  
C IF (II, GT, JCOL) JCOL = II  
IF (II, GT, MC) GO TO 9  
GO TO 10  
9 CONTINUE  
PRINT 47, II  
STOP  
10 CONTINUE  
JKL = JKOP + 1  
11 IX = I - 1  
A(OPROW, IX) = A(OPROW, IX)  
ICOL(OPROW, IX) = ICOL(OPROW, IX)  
II = IX  
IF (II, LE, JKL) GO TO 11  
A(OPROW, JKOP) = A(PIVROW, JKPI) * C  
ICOL(OPROW, JKOP) = ICOL(PIVROW, JKPI)  
IX = ICOL(OPROW, JKOP)  
IRC(IX) = IRC(IX) + 1  
GO TO 18  
C PIVROW AND OPROW CONTAIN THIS ELEMENT * SHIFT BOTH AND OPERATE ON  
C OPR  
12 IX = ICOL(OPROW, JKOP)  
IF (IX, EQ, PIVCOL) GO TO 13  
X = A(PIVROW, JKPI) * C * A(OPROW, JKOP)  
A(OPROW, JKOP) = X  
C TEST OPROW TO SEE IF ANY ELEMENTS WERE ELIMINATED OTHER THAN  
C THOSE IN THE PIVOTAL COLUMN  
A TEST = ABS(X) - Z TEST
IF (ATEST,GT,0,0) GO TO 18
13 IRC(I+I)=IRC(I+1)+1
C JETIM=JETIM+1
ICC(OROW)=ICC(OROW)+1
14 IF (ICC(OROW)) 15,14,16
15 CONTINUE
16 ICC(OROW)=ICC(OROW)+2
IX=IABS(ICC(OROW))
DO 17 NK=JKOP,IX
A(I,NK)=A(I,NK+1)
17 ICOL(I,NK)=ICOL(I,NK+1)
CONTINUE
IX=IX-1
ICOL(I,IX)=0
JKPI=JKPI+1
GO TO 7
18 JKPI=JKPI+1
C PIVROW DOES NOT CONTAIN THIS ELEMENT*SHIFT OPROW AND CONTINUE
19 JKOP=JKOP+1
GO TO 7
20 CONTINUE
21 CONTINUE
22 CONTINUE
C ELIMINATES PIVROW AND PIVCOL FROM BEING CONSIDERED AGAIN
23 ICC(PIVROW)=ICC(PIVROW)
24 IRC(PIVCOL)=IRC(PIVCOL)
LKJ=LKJ+1
IF (LKJ.ED.NU) GO TO 26
C UNSCRAMBLE AND STORE SOLUTION IN FIRST COLUMN OF A
25 DO 23 I=1,NU
II=ICOL(I+1)
A(I+1)=P(I)
CONTINUE
RETURN
C SUBROUTINE SINGLR
IB=1
26 DO 25 M=1,NU
II=ICC(M)
IF (II.ED.IB) IB=II
27 CONTINUE
WRITE (6,48) LKJ,PIVROW,PIVCOL,OROW
WRITE (6,49) (M,IRC(M),ICC(M)+1,NU)
WRITE (6,50)
CALL RITE (1,NU,IB,MR,MC,A,ICOL)
WRITE (6,51)
CALL RITE (2,NU,IB,MR,MC,A,ICOL)
STOP
28 CONTINUE
GO TO (27,28,30,33,36,40,44), IPIVOT
29 CONTINUE
PIVOT OPTION 1: SIMPLE GAUSS-JORDAN ELIMINATION

SURROUTINE PIVSEL
GAUSS ELIMINATION
PIVROW=LKJ
PIVCOL=LKJ
IY=1

END GAUSS ELIMINATION

RETURN

GO TO A

CONTINUE

PIVOT OPTION 2: GAUSS-JORDAN PARTIAL PIVOTING

SURROUTINE PIVSEL
GAUSS JORDAN-PARTIAL PIVOTING
PIVCOL=LKJ
ATEST=0
DO 29 I=1,NU
IC=ICC(I)
IF (IC.LE.0) GO TO 29
II=ICOL(I+1)
IF (II.GT.PIVCOL) GO TO 29
AA=ABS(A(I+1))
IF (AA.LE.ATEST) GO TO 29
ATEST=AA
PIVROW=I

CONTINUE

IY=1

END GAUSS JORDAN-PARTIAL PIVOTING

RETURN

GO TO A

CONTINUE

PIVOT OPTION 3: GAUSS-JORDAN FULL PIVOTING

SURROUTINE PIVSEL
GAUSS JORDAN FULL PIVOTING
ATEST=0
DO 32 I=1,NU
IC=ICC(I)
IF (IC.LE.0) GO TO 32
DO 31 J=1,IC
II=ICOL(I+J)
IR=IRC(I)
IF (IR.LE.0) GO TO 31
AA=ABS(A(I+J))
IF (AA.LE.ATEST) GO TO 31
ATEST=AA
PIVROW=I
PIVCOL=II
IY = J
31 CONTINUE
32 CONTINUE
C END GAUSS JORDAN FULL PIVOTING
C RETURN
GO TO 4
33 CONTINUE
C
C PIVOT OPTION 4: MINIMUM ROW - MINIMUM COLUMN
C
C SUBROUTINE PIVSEL
C SELECT FIRST MIN ROW THEN FIRST MIN COL
C SELECT ROW WITH MINIMUM ENTRIES
IK = 100000
DO 34 I = 1, NU
IC = ICC(I)
IF (IC .GE. IK OR IC .LE. 0) GO TO 34
PIVROW = I
IK = IC
34 CONTINUE
C SELECT SMALLEST AVAILABLE COLUMN FROM PIVROW
IK = 100000
IC = ICC(PIVROW)
DO 35 I = 1, IC
II = COL(PIVROW, I)
IR = IRC(I)
IF (IR .GE. IK OR IR .LE. 0) GO TO 35
PIVCOL = II
IK = IR
IY = I
35 CONTINUE
C END FIRST MIN ROW THEN FIRST MIN COL
C RETURN
GO TO 4
36 CONTINUE
C
C PIVOT OPTION 5: MINIMUM COLUMN - MINIMUM ROW
C
C SUBROUTINE PIVSEL
C SELECT FIRST MIN COL THEN FIRST MIN ROW
IK = 100000
DO 37 I = 1, NU
IR = IRC(I)
IF (IR .GE. IK OR IR .LE. 0) GO TO 37
PIVCOL = I
IK = IR
37 CONTINUE
IK = 100000
DO 39 I = 1, NU
IC = ICC(I)
IF (IC .LE. 0) GO TO 39
DO 38 J = 1, IC
IF (ICOL(I,J),LT,PIVCOL) GO TO 38
IF (ICOL(I,J),GT,PIVCOL.OR.IC,GE.IK) GO TO 39
IK=IC
PIVROW=I
IY=J
38 CONTINUE
39 CONTINUE
C END SELECT FIRST MIN COL THEN FIRST MIN ROW
C RETURN
GO TO 4
40 CONTINUE
C
C PIVOT OPTION 6 : MAXIMUM COLUMN-MINIMUM ROW
C
C SUBROUTINE PIVSEL
C SELECT FIRST MAX COL THEN FIRST MIN ROW
IK=1
DO 41 I=1,NU
IR=IRC(I)
IF (IR,LE.IK.OR.IR.LE.0) GO TO 41
PIVCOL=I
IK=IR
41 CONTINUE
DO 43 I=1,NU
IC=ICC(I)
IF (IC.LE.0) GO TO 43
DO 42 J=1,IC
IF (ICOL(I,J),LT,PIVCOL) GO TO 42
IF (ICOL(I,J),GT,PIVCOL.OR.IC,GE.IK) GO TO 43
IK=IC
PIVROW=I
IY=J
42 CONTINUE
43 CONTINUE
C END SELECT FIRST MAX COL THEN FIRST MIN ROW
C RETURN
GO TO 4
C CONTINUE
C
C PIVOT OPTION 7 : MINIMUM OF ROW ENTRIES TIMES COLUMN ENTRIES
C
C SUBROUTINE PIVSEL
C SELECT FIRST MIN(ROW*COL)
IK=100000
DO 46 I=1,NU
IC=ICC(I)
IF (IC.LE.0) GO TO 46
DO 45 J=1,IC
II=ICOL(I,J)
IR=IRC(II)
IF (IR.LE.0) GO TO 45
III=IC=IR
IF (III .GE. IK) GO TO 45
PIVROW = I
PIVCOL = II
IK = III
IY = J
45 CONTINUE
46 CONTINUE
C END SELECT FIRST MIN(ROW*COL)
C RETURN
GO TO 4
C
47 FORMAT (22H MC SHOULD BE AT LEAST .I3)
48 FORMAT (*1 MATRIX SINGULAR ,/ * NO. CYCLES COMPLETED .I5 ,/ P .ISM 325
1VROW = .I5 ,/ PIVCOL = .I5 ,/ OPROW = .I5)
49 FORMAT (*0 COL./ROW NO. .NO. COL. ENTRIES .NO. ROW ENTRIES */ (6(17 ,2 .ISIM 327
15)))
50 FORMAT (*1 COEFFICIENT MATRIX *)
51 FORMAT (*1 COLUMN IDENTIFICATION *)
END
SUBROUTINE RITE (IDUM, NR, NC, MR, MC, A, ICOL)
DIMENSION A(MR, MC), ICOL(MR, MC)
IPRINT=12
IF (IDUM.NE.1) IPRINT=30
IPR=IPRINT-1
DO 3, K=1, NC, IPRINT
MAX=K*IPR
IF (MAX.GT. NC) MAX=NC
IF (K.NE.1) WRITE (6, 7)
IF (IDUM.EQ.1) GO TO 2
WRITE (6, 5) (I, I=K, MAX)
DO 1 J=1, NR
WRITE (6, 8) J, (ICOL(J, I), I=K, MAX)
1 CONTINUE
GO TO 4
2 WRITE (6, 6) (I, I=K, MAX)
DO 3 J=1, NR
WRITE (6, 9) J, (A(J, I), I=K, MAX)
3 CONTINUE
4 CONTINUE
RETURN
C
5 FORMAT (6X, 30I4)
6 FORMAT (6X, 12I10)
7 FORMAT (1H1)
8 FORMAT (1X, 15, 30I4)
9 FORMAT (1X, 15, 12G10.3)
END
SUBROUTINE TYPE10

SUBROUTINE REAC1

THIS MODULE REPRESENTS A CSTR WITH A FIRST ORDER REVERSIBLE
REACTION, 1 INPUT STREAM, 1 OUTPUT STREAM, CONSTANT TEMPERATURE.

EQUIPMENT PARAMETERS

1 - REACTOR VOLUME - FT***3
2 - K1 - FORWARD REACTION RATE CONSTANT - MIN***1
3 - K2 - BACKWARD REACTION RATE CONSTANT - MIN***1
4 - ITER - ITER=0 - USE DIRECT ITERATION WITH STIFF OPTION
      ITER=1 - USE NEWTON-RAPHSON ITERATION WITH STIFF OPTION
      FOR NONSTIFF OPTION ITER IS NOT USED

COMMON /MAT, MP(15,5)*EP(15,5)*S(2,16,7)*EX(1)
COMMON /COM, IG,NCOM,NCS,H,NE,NS,NP0,NPOL,TMAX,IORDER,NGRAPH
COMMON /PTAB, IGFLAG,PP(10,10)
COMMON /UNIT, IM
COMMON /COLUMN, JCOL(10,10)
COMMON /JACOB, JACOB(10,10)
COMMON /MODULE, IDERY, ITR, ITRI, MC, IPIVOT
DIMENSION Y(2), DERY(2)
REAL K1,K2

CALCULATE MODULE PARAMETERS

GET VOLUME OF REACTOR FROM EP - VOL
VOL=EP(IM+1)

GET REACTION RATE CONSTANTS FROM EP - K1,K2
K1=EP(IM+2)
K2=EP(IM+3)

GET ITERATION OPTION FROM EP
ITER=EP(IM+4)

GET STREAM NUMBER OF INPUT STREAM FROM MP - IN
IN=MP(IM+3)

GET STREAM NUMBER OF OUTPUT STREAM FROM MP - IOUT
IOUT=IABS(MP(IM+4))

GET DENSITY OF INPUT STREAM FROM PP - DENS
DENS=PP(1,2)

CALCULATE REACTOR TIME CONSTANT - TAU
TAU=VOL/(S(IG,IN,3)/DENS)

GET INITIAL REACTOR CONCENTRATIONS FROM OUTPUT STREAM
Y(1)=S(IG,IOUT,6)
Y(2)=S(IG,IOUT,79)

CALCULATE JACOBIAN MATRIX ON CORRECTOR PASS

IF (IG,Eq.2) GO TO 1
MC=2
IPIVOT=3
JCOL(1,1)=1
JCOL(2,1)=1
JCOL(1,2)=2
JCOL(2,2)=2
XJACOB(1,1)=(K1+1.0/TAU)
XJACOB(2,1)=K1
XJACOB(1,2)=K2
XJACOB(2,2)=(K2+1.0/TAU)
CONTINUE

CALCULATE DERIVATIVES

MASS BALANCE FOR COMPONENT A (UNITS=MIN***1)
RATE OF CHANGE OF MASS FRACTION OF COMPONENT A
  RATE OF INPUT OF A
  RATE OF OUTPUT OF A
  RATE AT WHICH B REACTS INTO A
  RATE AT WHICH A REACTS INTO B
DERY(1)=-((K1+1.0/TAU)*Y(1)+K2*Y(2)+S(IN1,6)/TAU)
MASS BALANCE FOR COMPONENT B (UNITS=MIN***1)
RATE OF CHANGE OF MASS FRACTION OF COMPONENT B
  RATE OF INPUT OF B
  RATE OF OUTPUT OF B
  RATE AT WHICH A REACTS INTO B
  RATE AT WHICH B REACTS INTO A
DERY(2)=(K1*Y(1)-(K2+1.0/TAU)*Y(2)+S(IN1,7)/TAU)

CALL DIFSUB TO SOLVE ODES FOR MODULE

CALL DIFSUB (2,Y,DERY)
IF (IDERY.NE.0) GO TO 1

CALCULATE STREAM OUTPUT

NORMALIZE CONCENTRATIONS OF A AND B
SUM=Y(1)+Y(2)
Y(1)=Y(1)/SUM
Y(2)=Y(2)/SUM

PUT MASS FRACTIONS INTO OUTPUT STREAMS
S(1,IOUT6)=Y(1)
S(1,IOUT7)=Y(2)
RETURN
END
## Reactor Simulation - 2 CSTRs

BEGIN
TIME 1.0
MMAX 1.0
COMPS 2.0
LIBRARY 1.0
REAC1 10.0
PROCESS REAC1
  1.0 1.0
  1.0 10.0 4.90909 4.09091 1.0
REAC1
  7.0
  2.0 10.0
  2.0 686.81 312.19 1.0
END
STREAMS
EXPLICIT 3.0
  1.0 1.0 624.0 60.0 14.7
  1.0 2.0 624.0 70.0 14.7
  1.0 3.0 624.0 120.0 14.7
END
PROPERTIES -1.0
END
REFERENCES


Axelsson, O. 1969: A Class Of A-Stable Methods, BIT 9, 185

Axelsson, O. 1972: A Note on a Class Of Strongly A-Stable Methods, BIT 12, 1


Bjurel, G. 1972: Modified Linear Multistep Methods For A Class Of Stiff Ordinary Differential Equations, BIT 12, 142.


Brandon, D.M. 1973: Digital Simulation Of Continuous Systems And The New IMP Program, Simuletter/IV/3 (A.C.M.)

Brandon, D.M. 1974a: The Implementation And Use Of Sparse Matrix Techniques In General Simulation Programs, Comp. J. 17, 2, 165.

Brandon, D.M. 1974c: Private communication, March 22
Brandon, D.M. 1974d: Private communication, May 13
Brandon, D.M. 1974e: Private communication, June 24
Brandon, D.M. 1974f: A New Single-Step Implicit Integration Algorithm With A-Stability And Improved Accuracy, Simulation 23, 1, 17
Brunner, H. 1972: A Class Of A-Stable Two-Step Methods Based On Schur Polynomials, BIT 12, 468


Crean, J.P. 1974: Manager, Sales Administration, Stone and Webster Engineering Corporation, New York, private communication, Sept. 4


Cryer, C.W. 1972: On the Instability Of High Order Backward-Difference Multistep Methods, BIT 12, 17

Cryer, C.W. 1973: A New Class Of Highly-Stable Methods: A_0-Stable Methods, BIT 13, 153


Dahlquist, G. 1963: A Special Stability Problem For Linear Multistep Methods, BIT 3, 27


Davison, E.J. 1968: The Numerical Solution Of Large Systems Of Linear Differential Equations, A.I.Ch.E. J. 14, 1, 46
Davison, E.J. 1973: An Algorithm For The Computer Simulation Of Very Large Dynamic Systems, Automatica 9, 665


Dill, C. and Gear, C.W. 1971: A Graphical Search For Stiffly Stable Methods For Ordinary Differential Equations, J.A.C.M. 18, 1, 75


Distefano, G.P. 1968a: Mathematical Modeling And Numerical Integration Of Multicomponent Batch Distillation Equations, A.I.Ch.E. J. 14, 1, 190


Enright, W.H. 1974: Assistant Professor, Dept. Of Computer Science, Univ. Of Toronto, private communication, April 4


Franks, R.G.E. 1967: Mathematical Modeling In Chemical Engineering, Wiley


Gear, C.W. 1974: Professor, Dept. Of Computer Science, Univ. Of Illinois At Urbana-Champaign, Urbana, Illinois, private communication, May 14


Giese, C. 1967: State Variable Difference Methods For Digital Simulation, Simulation, 8, 5, 263


Henrici, P. 1962: Discrete Variable Methods In Ordinary Differential Equations, Wiley

Hildebrand, F.B. 1956: Introduction To Numerical Analysis, McGraw-Hill


Jung, H.P. 1968: Padé Integration Methods, Space Division Memo PIR 5540-82, General Electric Co., Syracuse, N.Y.


Kardasz, J.H. and Molnar, G. 1974: A SIMULA-Based Structure Oriented Language For The Dynamic Simulation Of Chemical Plants, Computer J. 17, 1, 28


Kubicek, M. 1975: A Nonlinear Explicit Second Order Algorithm For Integration Of Stiff Equations, Comm. A.C.M. (To Be Published)

Lambert, J.D. 1973: Computational Methods In Ordinary Differential Equations, Wiley


Pope, D.A. 1963: An Exponential Method Of Numerical Integration Of Ordinary Differential Equations, Comm. A.C.M. 6, 8, 491


Reid, J. K. (ed.) 1971: Large Sparse Sets Of Linear Equations, Academic Press

Rheinboldt, W.C. 1973: Programs For The Solution Of Large Sparse Matrix Problems Based On The Arc-Graph Structure, PB-224 810, NTIS

Richards, F.I., Lanning, W.D. and Torrey, M.D. 1965: Numerical Integration Of Large Highly-Damped, Nonlinear Systems, SIAM Rev. 7, 3, 376


Schappelle, R. 1967: LINEQ4, A FORTRAN Subroutine For The Solution Of Sparse Linear Equations Available Through CDC User Association VIM, VIM F4 GDC CSLNQ4

Schoen, K. 1971: Fifth And Sixth Order PECE Algorithms With Improved Stability Properties, SIAM J. Num. Anal. 8, 2, 244


Utsumi, T. 1969: Stone And Webster All Purpose Simulator And Optimizer (SWAPSO) And Its Applications, Paper Presented At The Conference On Applications Of Continuous Systems Simulations Languages, San Francisco
Varga, R.S. 1962: Matrix Iterative Analysis, Prentice-Hall

Walters, C.M. 1971: Numerical Integration Of Stiff Ordinary Differential Equations, AFWL-TR-71-6, NTIS

Walters, C.M. 1972: A Discussion Of Gears Implementation Of Adams And Stiff Methods For Solving Ordinary Differential Equations, AFWL-TR-72-170, NTIS

Watts, H.A. and Shampine, L.F. 1972: A-Stable Block Implicit One-Step Methods, BIT 12, 252


Widlund, O.B. 1967: A Note On Unconditionally Stable Linear Multistep Methods, BIT 7, 65

Williams, J. and Hoog, F. 1974: A Class Of A-Stable Advanced Multistep Methods, Math. Comp. 28, 125, 163


