

MATHEMATICAL SOFTWARE FOR  
GAS TRANSMISSION NETWORKS

by

TAT-SENG CHUA  
*~*

Submitted in fulfilment of the requirements for  
the degree of Doctor of Philosophy

The University of Leeds  
Department of Computer Studies

November, 1982

## ABSTRACT

This thesis is concerned with the development of numerical software for the simulation of gas transmission networks. This involves developing software for the solution of a large system of stiff differential/algebraic equations (DAE) containing frequent severe disturbances. The disturbances arise due to the varying consumer demands and the operation of network controlling devices such as the compressors. Special strategies are developed to solve the DAE system efficiently using a variable-step integrator. Two sets of strategies are devised; one for the implicit methods such as the semi-implicit Runge-Kutta method, and the other for the linearly implicit Rosenbrock-type method. Four integrators, based on different numerical methods, have been implemented and the performance of each one is compared with the British Gas network analysis program PAN, using a number of large, realistic transmission networks. The results demonstrate that the variable-step integrators are reliable and efficient.

An efficient sparse matrix decomposition scheme is developed to solve the large, sparse system of equations that arise during the integration of the DAE system. The decomposition scheme fully exploits the special structure of the coefficient matrix.

Lastly, for certain networks, the existing simulation programs fail to compute a feasible solution because of the interactions of the controlling devices in the network. To overcome this difficulty, the problem is formulated as a variational inequality model and solved numerically using an optimization routine from the NAG library (NAGFLIB(1982)). The reliability of the model is illustrated using three test networks.

## Acknowledgements

I would like to thank Peter Dew for his supervision, guidance and encouragement throughout this project. I would also like to thank Susan Nemes for her expert typing.

Special thanks are due to British Gas Corporation for its financial support, and to A.E. Fincham, N.J. Revel, J.R. Mallinson and N.H. Goodwin of its London Research Station for their help and advice during the project.

Finally, I would like to express my thanks to a very special person, Chew-Lan, for her encouragement and help in many ways.

## CONTENTS

	<u>Page</u>
Nomenclature	1
Chapter 1 : Introduction	4
Chapter 2 : A Survey of Numerical Techniques	13
Chapter 3 : The Solution of Linear Equations	36
Chapter 4 : The Design of the Variable-step Integrator	60
Chapter 5 : The Numerical Testings and Results	82
Chapter 6 : The Conflicting Constraints Problem	98
Chapter 7 : Conclusions	114
References	117
Appendix A : The Test Networks	126
Appendix B : The Mathematical Background to the PAN Program	132
Appendix C : Arithmetic Operation Counts	149
Appendix D : The Numerical Methods	152
Appendix E : The Implementation of the Variable-step Integrators	157

## Nomenclature

This thesis covers several different fields of study including the gas industry, the solution of differential/algebraic equations, the sparse matrix computation and numerical optimization. The repeated use of certain symbols is therefore unavoidable. To avoid complication, the symbols that appear in the appendices are not included here; they are explained as and when they occur. Furthermore, those symbols that are specific to a particular field of study and appear only in the background sections (for example, sections 3.4 and 6.5) are also excluded.

### Latin Characters

- A - cross-sectional area of the pipe (in section 2.1 only).
- $A(\underline{y})$  or  $A$  - the right-hand matrix of DAE system (1.3.1)
- $A_{ij}$  - the submatrices of matrix  $G$  as defined in eqn.(3.2.1).
- $A_p$  - submatrix corresponds to the machine nodes as defined in eqn.(3.3.1)
- $\underline{b}_1, \underline{b}_2$  - the known right-hand quantities of the linearised network equations given in eqns.(6.2.1)-(6.2.2)
- B - matrix with constant coefficients
- C - the matrix  $C$  defined in eqn.(3.3.2)
- d - the parameter of the Rosenbrock-type method
- $\underline{d}(t)$  - vector containing the demands from the network.
- $\underline{e}_n$  - actual local error of the numerical method at time  $t_n$
- E - the capacity matrix of the DAE system (1.3.1)
- EPS - user specified accuracy tolerance
- $\underline{f}(t, \underline{y})$  - nonlinear function in  $t$  and  $\underline{y}$
- $\underline{F}(\underline{y})$  - nonlinear algebraic equations in  $\underline{y}$
- $\underline{g}_n$  - the global error estimate at time  $t_n$
- $\underline{q}(t)$  - any function defined and bounded in some interval  $[0, T]$
- G - the iteration matrix of (1.3.1)
- h - the time stepsize
- $\bar{h}$  -  $= \lambda h$
- H - a functional of  $\underline{\Omega}$ ,  $\underline{\Phi}$  and  $\underline{Q}$  defined in eqn.(6.4.5)
- I - identity matrix

- $J(\underline{y})$  - the matrix  $\partial F(\underline{y})/\partial \underline{y}$   
 $k$  - order of the numerical method  
 $\underline{k}_i$  - the intermediate solution vector of the Runge-Kutta and Rosenbrock-type method  
 $K$  - the flow incidence matrix  
 $K_p$  -  $= L_p^{-T} L_p^{-1} K$   
 $K_I$  - machine inlet flow incidence matrix (eqn.(6.2.3))  
 $\underline{\ell}_n$  - local error estimate at time  $t_n$   
 $L, L_p$  - the Cholesky factors of the symmetric and positive-definite matrices  $A_{11}$  and  $A_p$  respectively  
 $m$  - number of iterations  
 $m_1, m_2, m_3$  - coefficients of the machine constraint equation (eqn.(6.5.1))  
 $M$  - molecular weight of gas (in section 2.1 only)  
 $M$  - general matrix  
 $M_1, M_2$  - submatrices containing the coefficients of the machine constraint equations (eqn.(3.2.1))  
 $\tilde{M}$  - the machine flow matrix defined in eqn.(3.3.3)  
 $n(\cdot)$  - number of non-zeros of a matrix or a vector  
 $N$  - size of matrix  $G$   
 $P$  - the pressure  
 $\underline{P}$  - the unknown pressures at free nodes  
 $\underline{q}$  - the flows through the pipes  
 $\underline{Q}$  - the unknown machine flows  
 $\hat{\underline{Q}}$  -  $= \underline{Q}_{\max} - \underline{Q}$ , defined in section 6.4  
 $r_c$  - rate of convergence of the modified-Newton iteration  
 $r(\bar{h})$  - stability polynomial (see eqn.(2.2.4))  
 $\underline{r}$  - known right-hand vector of the linear equations  
 $\underline{r}_1, \underline{r}_2, \underline{r}_3$  - the components of vector  $\underline{r}$   
 $R_A$  - absolute stability region of a numerical method  
 $t$  - the independent variable, the time  
 $\underline{u}_n(t)$  - local solution at time  $t > t_n$   
 $\underline{w}$  - iteration correction of the modified-Newton iteration  
 $x$  - distance along the pipe  
 $\underline{x}$  - the unknown vector of the linear equation (3.1.1)  
 $\underline{y}$  - the dependent variable  
 $\underline{Y}_n$  - computed solution at time  $t_n$   
 $Z$  - the off-diagonal matrix defined in eqn.(3.3.1)  
 $Z_{ij}$  - the submatrices defined in eqn.(6.2.1)-(6.2.2)

### Greek Characters

- $\beta$  - the parameter of a numerical method  
 $\theta$  - the parameter for the theta-method  
 $\lambda$  - the eigenvalue  
 $\gamma_1, \gamma_2, \gamma_3$  - length of vectors  $\underline{P}$ ,  $\underline{\pi}$  and  $\underline{Q}$  respectively  
 $\sigma_i$  - known right-hand quantity of the machine constraint equation  
 $\rho$  - density of gas  
 $\varepsilon_n$  - error estimate at time  $t_n$   
 $\underline{\phi}$  - switching functions for locating the discontinuities  
 $\underline{\pi}$  - unknown pressures at the machine nodes  
 $\underline{\Psi}(\underline{y})$  - the algebraic equations in the DAE system  
 $\underline{\Phi}$  - set of pressures at the machine outlet nodes (used in chapter 6 only)  
 $\underline{\Omega}$  - set of pressures at free nodes and the machine inlet nodes (used in chapter 6 only)  
 $\| \cdot \|$  - a suitable norm for measuring the size of a matrix or a vector  
 $(\cdot), \langle \cdot \rangle, \{ \cdot \}$  - inner products

## CHAPTER 1

## 1. INTRODUCTION

- 1.1 The Gas Transmission Network - the need for simulation
- 1.2 Collaboration with British Gas Corporation
- 1.3 The Mathematical Background to PAN
- 1.4 The Scope and Objectives of the Thesis
- 1.5 The Implementation of Leeds/PAN
- 1.6 The Survey of Contents



## CHAPTER 1

1.1 The Gas Transmission Network - the need for simulation

Natural gas has become an important alternative source of energy which accounts for nearly a fifth of the European Community's energy needs today (GASEMN(1981)). Large amounts of gas are consumed everyday to provide heating and energy for millions of homes and industries. Because natural gas is often discovered in remote locations far away from the centres of demand, there is an extensive transmission network to transport the gas to places where it is needed. The transmission network consists of a sequence of pipes with a number of special devices like compressors (used to boost the gas pressure so that the gas can be transmitted over a long distance). Other devices in the network include regulators and valves which are also used to control the pressure and flow. It is conventional to refer to devices that control the gas flow through the network as machines.

Massive pipeline construction in Great Britain started in the mid 1960s in the wake of discoveries of natural gas in the North Sea. To date, more than 2,900 miles of mainly 36 inch diameter steel pipes have been laid. These pipes are operating at pressures of between 35 to 70 bars. There are thirteen compressor stations in operation. The compressors are mainly of centrifugal type driven by gas turbine engines fueled by natural gas taken from the network. British Gas have plans to extend the network to about 4,000 miles of pipes and twenty compressor stations. A map showing the complexity of the British Gas national transmission network is given in fig.(A.7).

To design cost-effectively and operate such a large and complex transmission network, it is essential to simulate the gas transmission network using a computer. The simulation is used to assist the engineers during the design stage, for example, to check that a proposed design can cope with the anticipated demand. It is also used in the control of the network, for example, to check if a particular operating

policy is feasible in the light of the expected demand over the next few hours. The operation of the network invariably involves large dynamic variations caused by the time varying consumer demands and the switching on and off of machines. Furthermore for the high pressure network, it is also important to know the storage capability of the network so that it can be exploited in meeting consumer demands. To understand the behaviour of gas flow in the network, a thorough knowledge of the dynamic behaviour of the network is required which means that it is necessary to develop computer programs that can perform dynamic flow simulations.

### 1.2 Collaboration with British Gas Corporation

The research work reported in this thesis has been carried out in close collaboration with Mr. A.E. Fincham and the PAN team at the London Research Station of the British Gas Corporation. The scope and objectives of the research programme outlined in Section 1.4 arose out of discussions with British Gas and regular progress meetings were held to review the work.

Currently British Gas use a network analysis program called PAN (Program to Analyse Networks) (GOLDWATER(1976)) to carry out the dynamic flow simulations. This program has two distinct parts. The first part handles the I/O which includes a large number of facilities to assist the engineers in specifying the network. The second part performs the actual analysis of the network. It is the second part which is of interest in this thesis. British Gas are currently developing a replacement for this program and the purpose of this thesis is to investigate how the numerical methods used in PAN can be improved upon.

### 1.3 The Mathematical Background to PAN

To avoid giving too much detail at this stage, a full description of the mathematical background to PAN is given in Appendix B. The basic gas flow model used in PAN

is the one-dimensional, isothermal parabolic model which makes the assumptions that the gas flow temperature is constant and the inertia terms in the gas flow equations can be ignored (see Appendix B for further details). These assumptions are valid during the normal operation of the transmission network but would not be true in places where there is a large disturbance (say due to a pipe break). The PAN program has been tested extensively using actual network data during its initial implementation (GOLDWATER(1976)) and has been in general use within British Gas for about 10 years, we can therefore feel confident about the validity of the model employed. Hence it was decided early on (in conjunction with British Gas) that study should concentrate on improving the numerical methods used in PAN.

Applying the method of lines to the gas flow model over the whole network and adding in the models for the machines leads to a large system of differential/algebraic equations (DAE) of the form,

$$E \frac{dy}{dt} = A(y)y - \underline{d}(t) = \underline{f}(t,y) \quad (1.3.1)$$

where  $y$  is a vector denoting the unknown pressures at the nodes of the network and the gas flow through the machines;  $\underline{d}(t)$  is a vector containing the demands from the network. The matrix  $A$  is nonlinear in  $y$  with some of its elements depending on the operating characteristics of the machines; the matrix  $E$  is singular when there are machines present in the network. The details of how this system of DAE is derived can be found in Appendix B. For a typical national grid simulation, the resulting DAE system has 158 differential equations and 20 algebraic equations, however, because there is very little coupling within the network, the matrices  $E$  and  $A$  are very sparse.

An initial steady state solution is used to provide the initial conditions to the DAE system; this involves solving the algebraic equations,

$$A(\underline{y}_0) \underline{y}_0 = \underline{d}(t_0) \quad (1.3.2)$$

where  $t_0$  is the starting time of the simulation. This ensures that the initial condition is compatible with the DAE system.

The machines in the network are modelled by a set of inequality constraints (see Appendix B). For example, a compressor normally has constraints on the maximum outlet pressure, maximum horsepower and maximum compression ratio, where the later two constraints are nonlinear in  $\underline{y}$ . Following the model used in PAN, it is assumed that one of the constraints is actually equal to its extreme value and is referred to as the operating constraint. This gives rise to the algebraic equations in (1.3.1). For example, when a machine has its operating constraint on the maximum outlet pressure, the resulting algebraic equation is simply given by

$$P_0 - P_{\max} = 0 \quad (1.3.3)$$

where  $P_0$  is the machine outlet pressure and  $P_{\max}$  is its extreme value. The algebraic equation is only linear in this case; for machines operating on nonlinear operating constraints such as the compressor horsepower constraint, then the resulting algebraic equation will be nonlinear. The modified-Newton method (see section 2.2) is used to ensure the convergence of the nonlinear algebraic equations.

During the initial steady-state calculation, an arbitrary set of operating constraints for the machines is chosen, where the typical first choice is the maximum outlet pressure constraint if available. A solution is computed based on these constraints. The rest of the machine constraints are then checked; if any is violated, then the constraint which is violated by the greatest percentage are chosen as the new operating constraint of the machine. A new solution is found and the process is repeated until there are no violations. The same procedure is also employed to compute the dynamic solution at each time step. The switching of machine operating

constraints means that a new matrix  $A$  is required and a new problem is to be solved.

In addition, the demands from the network vary quite considerably throughout the simulation. These changes in demands are approximated using a step or linear function which means that the demand vector,  $\underline{d}(t)$ , is only piecewise continuous with respect to time. A typical demand profile is given in fig.(5.4).

The varying consumer demands generate large disturbances in the gas flow and cause the machine to switch operating constraints; these activities result in severe discontinuities in the derivatives of the solution. A survey of the literature on the numerical solution of such DAE system using a variable-step integrator is given in Chapter 2.

In the PAN program, a modification of the Crank-Nicolson method is used to integrate the differential equations within the DAE system; the resulting equations are then combined with the machine algebraic equations to solve for the unknown variables. The details of this solution scheme is given in section B.6. It is implemented in a constant step formulation and uses a block matrix partitioning method (GOLDWATER(1976)) to solve the linear, sparse system of equations at each time step. Although the resulting solution scheme using the Crank-Nicolson method is stable for any stepsize, it is well known that under certain conditions, spurious oscillations in the solution may occur.

#### 1.4 The Scope and Objectives of the Thesis

A major weakness in the current version of PAN is that it is based on a constant-step formulation which means that the user of the program has to provide a suitable time step. Since the choice of an appropriate time step is often a very difficult and time consuming task especially for large and complicated networks, it is obviously desirable to estimate the time step automatically within the program. Thus the main purpose of this thesis is to investigate the incorporation of a variable-step method in PAN. As the simulation usually involves severe disturbances in the gas flow due to the varying consumer demands and the operation of the machines, special techniques are needed to handle these changes efficiently. A detailed study on the best way of implementing the variable-step PAN will be carried out. A new method will also be devised for solving the large linear sparse system of equations at each time step.

For certain networks, PAN fails to find a solution, due to the difficulty of obtaining a feasible solution which satisfies all constraints associated with the network. Thus another aim of the study is to consider the development of an optimization model to overcome this problem. Such problems arise due to the interactions of the machines in the network and are referred to as the "conflicting constraints problem". A more detailed account of this problem and its possible solution is given in Chapter 6.

#### 1.5 The Implementation of Leeds/PAN

To provide a benchmark to measure the effect of any changes to the PAN program, it was necessary to implement the mathematical part of PAN at Leeds. Unfortunately, because of the portability and interface problems, it was not possible to mount PAN directly and a completely new version of the program had to be written.

The new version, referred to as Leeds/PAN, took about four months of intensive work to implement. It is

written in FORTRAN IV and consists of over 6,000 lines of code. It contains all the essential features of the original PAN program except the I/O facilities. Only a simple I/O routine was used for the study.

In Leeds/PAN, a constant-step theta method is employed to solve the DAE system (1.3.1). A new block matrix partitioning method is also used to solve the linear equations; this resulted in a much simpler solution scheme than the original implementation. The details of the new block matrix partitioning method will be given in Chapter 3.

Leeds/PAN has been tested against the original PAN program on a large number of test networks and similar results were obtained in each case. This ensured, as far as it was possible, that the Leeds/PAN program was free of programming errors and can safely be used as a basis for testing new implementations. A listing of the Leeds/PAN program along with some numerical results can be found in CHUA(1982).

## 1.6 The Survey of Contents

Briefly, the contents of the thesis are as follows. In chapter 2, a brief outline of the numerical techniques and computer programs developed during the last decade for carrying out the dynamic flow simulations is given. The chapter also surveys the numerical techniques that can be used to solve the stiff DAE system containing a large number of discontinuities efficiently.

Chapter 3 discusses the sparse matrix partitioning scheme developed for solving the linear systems of equations. The scheme is based on the extension of the block matrix partitioning method outlined in GEORGE(1974) and AZAR(1975). Three schemes are proposed; their efficiencies are compared using three test networks of different sizes and complexities.

The implementation of the variable-step program is discussed in chapter 4. Special techniques are described to handle efficiently the severe disturbances caused by the

abrupt changes in the gas flow and to solve the DAE system. The techniques used have wider applicability than the gas transmission systems since they only require that the model is parabolic in nature. Two sets of strategies are presented for implementing the variable-step program; one for the implicit numerical method such as the diagonally implicit Runge-Kutta methods, and the other for the Rosenbrock-type method.

Chapter 5 compares the numerical results obtained from the variable-step program with those obtained from the Leeds/PAN program. In this way, the accuracy and reliability of the results of the variable-step program are demonstrated using a number of large, realistic transmission networks.

In Chapter 6, the optimization model developed for resolving the "conflicting constraints problem" is discussed and analysed. The model is based on the dual extremum principles given in NOBLE(1972); it is tested using three test networks that exhibit the "conflicting constraints problem".

Lastly, the concluding remarks and recommendations for future developments are contained in chapter 7.



## CHAPTER 2

- 2. A SURVEY OF NUMERICAL TECHNIQUES
  - 2.1 Gas Transmission Network Analysis Programs
  - 2.2 Mathematical Background to Stiff Computation
    - 2.2.1 The Outline of a Stiff Variable-step Code
  - 2.3 A Survey of Stiff Integration methods
  - 2.4 The Handling of Discontinuities - A Survey
  - 2.5 The Solution of DAE System

## CHAPTER 2

2. A Survey of Numerical Techniques

This chapter surveys the numerical techniques that can be used for solving the DAE system arising from the simulation of gas transmission networks. In order to provide a general background to the solution of this problem, a brief outline of the numerical methods and computer programs developed for carrying out the gas transmission network simulation is also included.

2.1 Gas Transmission Network Analysis Programs

A large number of programs have been developed during the last decade to carry out the dynamic simulation of a gas transmission network. The majority of these programs are based on the isothermal model discussed in Appendix B (FINCHAM(1980)) and have been shown to predict the overall behaviour of the gas flow accurately under normal operating conditions. The major difference in the mathematical models underlying the various programs is the treatment of the inertia terms,  $\frac{M}{A} \frac{\partial q}{\partial t}$  and  $\frac{M^2}{2A} \frac{\partial}{\partial x} \left( \frac{q^2}{\rho} \right)$ , in the momentum equation (B.1.2). Some programs use a model which retains both terms while others neglect one (the second term) or both. By neglecting both the inertia terms, the model is parabolic in nature, otherwise it is hyperbolic.

The most popular method for approximating the gas flow equations is the finite difference technique. In this approach, the set of PDE's is discretised using the method of lines giving rise to a set of ordinary differential equations (ODE's) which is then solved using either an explicit or implicit integration method. Programs developed using explicit integration method include GOACHER(1970) and DISTEFANO(1970), while HEATH(1969), WYLIE(1971), PAN and the German network analysis program GANESI (SCHMIDT(1977)) use an implicit method. To illustrate the implicit approach and in particular, to highlight the techniques used in PAN, a

comparison between PAN and GANESI is carried out. GANESI has been chosen because it is similar to PAN in that it uses a constant-step implicit theta-type method to solve the differential equations and is widely used and well tested. The major differences between these programs are:

- i) GANESI uses a hyperbolic model which includes the first inertia term,  $\frac{M}{A} \frac{\partial q}{\partial t}$ ; whereas in PAN, a parabolic model is solved.
- ii) PAN uses essentially the Crank-Nicolson method to solve the differential equations; whereas in GANESI, the Crank-Nicolson and Backward Euler methods are used alternately.
- iii) In GANESI, the modified-Newton iterative method (see next section for details) is used to solve the non-linear algebraic equations; in PAN, however, these nonlinear equations are linearised and solved directly.

At first glance the most important difference between the two programs appears to be the basic gas flow model used. However, from the tests carried out in FINCHAM(1982) on the difference between PAN and GANESI, it was found that the inclusion of the inertia term,  $\frac{M}{A} \frac{\partial q}{\partial t}$ , in GANESI rarely makes any difference to the solution obtained under normal operating conditions. We shall therefore consider only the parabolic model in the rest of this thesis.

GANESI uses a modified-Newton method rather than simply linearising the equations as is done in PAN. This might generally be expected to lead to a more accurate solution, however, the need to perform several iterations per stepsize means that it is likely to be computationally more expensive. A more efficient implementation of the modified-Newton method will be discussed in Chapter 4.

Lastly, the mixed method which consists of Crank-Nicolson and backward Euler method is used in GANESI in an attempt to avoid the spurious oscillations that might result from the application of Crank-Nicolson method alone while achieving close to 2nd order accuracy. This point will be

taken into consideration in selecting the numerical methods in chapter 4.

Other techniques employed for numerically solving the gas flow equations include the method of characteristics and the finite element method. The method of characteristics (MOC) requires that the model concerned is hyperbolic (LISTER(1960)). It is explicit in nature and has a restriction on the time step to be less than the pipelength divided by the isothermal speed of sound. When implementing the MOC, the second inertia term  $\frac{M^2}{A} \frac{\partial}{\partial x} \left( \frac{q^2}{\rho} \right)$  is usually neglected which means that the characteristics lines are linear. A program successfully implementing this method is reported in WYLIE(1967). To overcome the restriction on the time step, STREETER(1970) proposed a method which combined the MOC with an implicit finite-difference method while in WYLIE(1974), the MOC is modified using a "inertia multiplier" technique suggested by YOW(1972).

The finite element method relies on rather more sophisticated functional analysis. The pipe is divided into several elements and the unknown pressures and flows are approximated by different polynomials over each element. A program based on this approach has been developed by RACHFORD(1974). In that implementation, both inertia terms are retained as important.

All the programs reported above are of constant-step in nature. To our knowledge, no program currently includes variable time step control though RACHFORD(1974) proposed it in the discussion following their paper.

## 2.2 Mathematical Background to Stiff Computation

The remaining part of the chapter is concerned with the numerical solution of a stiff DAE system which contains a number of discontinuities. This section provides the mathematical background to stiff computation and discusses the general design of a stiff variable-step integrator.

Consider the numerical solution of the system of differential equations

$$\underline{y}'(t) = \underline{f}(t, \underline{y}(t)), \quad t > t_0, \quad \underline{y}(t_0) = \underline{y}_0 \quad (2.2.1)$$

where  $\underline{f}(t, \underline{y})$  is assumed to be analytic in the neighbourhood of  $\underline{y}_0$  and  $\underline{y}'$  denotes differentiation with respect to  $t$ .

The system of differential equations (2.2.1) is said to be stiff in an interval  $I$  of  $t$  if, for all  $t \in I$ , the eigenvalues  $\lambda_i(t)$  of the Jacobian matrix  $\partial \underline{f} / \partial \underline{y}$  satisfy the following conditions (LAMBERT(1973))

- i) Real  $\lambda_i(t) < 0$ ,  $i = 1, 2, \dots, n$ ;
- ii)  $\max_t |\text{Real } \lambda_i(t)| \gg \min_t |\text{Real } \lambda_i(t)|$ .

As it is difficult to investigate the behaviour of a numerical method on a general stiff nonlinear problem, a linear model problem of the form below is normally considered

$$\underline{y}'(t) = B \underline{y}(t) \quad (2.2.2)$$

where the matrix  $B$  has constant coefficients and corresponds to an approximation to the Jacobian matrix of the nonlinear system (2.2.1) in the neighbourhood of the solution  $\underline{y}_0$ . It can be shown that it is sufficient to study only the stability of the simple test equation

$$y'(t) = \lambda y(t) \quad (2.2.3)$$

where the parameter  $\lambda$ , which can be real or complex, denotes an eigenvalue of the matrix  $B$ .

A one-step method applied to the test equation (2.2.3) gives the formula (see for example, LAMBERT(1973)),

$$y_{n+1} = r(\bar{h}) y_n \quad (2.2.4)$$

where  $\bar{h} = \lambda h$ ;  $y_{n+1}$ ,  $y_n$  are the computed solution at time  $t_{n+1}$  and  $t_n$  respectively with  $h = t_{n+1} - t_n$  and  $r(\bar{h})$  is a polynomial or rational function in  $\bar{h}$ . For example, when the Crank-Nicolson method is employed for solving (2.2.3),  $r(\bar{h})$  is given by

$$r(\bar{h}) = (1 + \frac{1}{2}\bar{h}) / (1 - \frac{1}{2}\bar{h}) \quad (2.2.5)$$

From eqn.(2.2.4), it can be deduced that if  $|r(\bar{h})| < 1$ , then the error (which includes the truncation as well as round-off error) in the computed solution will not grow in an unstable manner. This leads to the following definition

Def 2.2.1 - A one-step method is said to be absolutely Stable in a region  $R_A$  on the  $h\lambda$  - complex plane (see fig.(2.2.1)) if

$$|r(\bar{h})| < 1 \quad \forall \bar{h} \in R_A \quad (2.2.6)$$

The region  $R_A$  is known as the absolute stability region of the method.

For example, the classical fourth order Runge-Kutta method has a stability requirement of  $|\lambda h| < 2.8$  when  $\lambda$  is a negative real number. Hence for problem with large real negative  $\lambda$ , the time step must be kept small in order to ensure the stability of the method. It can be shown, however, that the Crank-Nicolson method does not have any restriction on the stepsize. Several definitions, which call for the method to possess some 'adequate' region of absolute stability, have been proposed.

Def.(2.2.2) (DAHLQUIST(1963)) - A numerical method is said to be A-stable if its region of absolute stability contains the whole of the left-hand, half-plane  $\text{Re } h\lambda < 0$ .

As a result of this definition, DAHLQUIST(1963) was able to prove that the maximum order of an A-stable linear multistep method is two. Owing to the restrictive nature of this result, the A-stability requirement is often relaxed, for example, GEAR(1969) introduces the idea of stiffly-stable formulae,

Def.(2.2.3) - A numerical method is said to be stiffly-stable if its region of absolute stability contains a region  $R = R_1 \cup R_2$  where (fig.(2.2.1)):

$$R_1 = \{\text{Real}(\lambda h) < D < 0\}$$

$$\text{and } R_2 = \{D \leq \text{Real}(\lambda h) \leq \alpha, \text{Im}(\lambda) \leq \theta\}$$

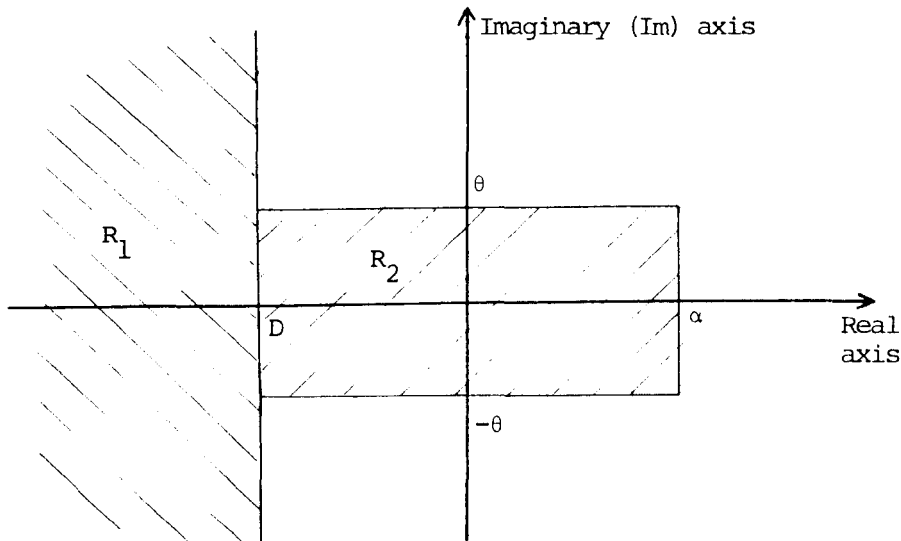


Fig. 2.2.1 - Complex  $h\lambda$ -phase

In general, A-stable methods which are not damped maximally as  $\lambda h \rightarrow -\infty$  are unsatisfactory for solving problems with large negative real eigenvalues, the concept of strongly A-stability is introduced.

Def.(2.2.4) - A one-step method is called strongly A-stable if it is A-stable and  $r(\bar{h}) \rightarrow 0$  (see eqn.(2.2.4)) as  $\bar{h} \rightarrow -\infty$ .

The stability properties discussed so far are based on the linear test problem (2.2.3), hence they give only limited guidance to the numerical behaviour of the method when it is applied to solve stiff nonlinear problems. In their work with large systems of stiff nonlinear equations, PROTHERO(1974) found that A-stability of a method was no guarantee that it would give stable solutions, and that the

accuracy of the solutions obtained often appeared to be unrelated to the order of the method used. Their analysis led to the introduction of a new stability concept.

Def.(2.2.5) - If the characteristics equation

$$y' = g' + \lambda \{y(t) - g(t)\} \quad (2.2.7)$$

is considered where  $g'(t)$  is any function defined and bounded in some interval  $[0, T]$ . A one-step method is termed S-stable if for any real positive constant  $\lambda_0$ , there exists a real positive  $h_0$  such that

$$\left\| \frac{y_{n+1} - g(t_{n+1})}{y_n - g(t_n)} \right\| = C_S < 1 \quad (2.2.8)$$

provided  $y_n \neq g(t_n)$ , for all  $0 < h < h_0$  where  $h = t_{n+1} - t_n$ , and all complex  $\lambda$  with  $\text{Re}(-\lambda) \geq \lambda_0$ , and  $t_n, t_{n+1} \in [0, T]$ .

Def.(2.2.6) - A S-stable one-step method is said to be strongly stable if  $C_S \rightarrow 0$  as  $\lambda h \rightarrow -\infty$ .

Other stability properties based on the theory of contractivity have also been considered for studying the behaviour of the numerical methods in solving stiff nonlinear problems. These include the concepts of G-stability (DAHLQUIST(1975)) for the linear multistep methods and B-stability (BUTCHER(1975)) for the implicit Runge-Kutta methods. In BUTCHER(1981), these two concepts are combined to give the algebraic stability for general linear methods. Details of these stability properties can be found, for example, in DAHLQUIST(1982).

The numerical methods which possess some of the properties discussed above are generally referred to as stiff integration methods and a survey of these methods is given in Section 2.3.



### 2.2.1 The Outline of a Stiff Variable-step Code

Modern general purpose stiff integration codes are usually implemented in the form of a variable-step (and possibly, variable-order) integrator, where the integration time step is varied automatically within the integrator in order to ensure that the computed solution satisfies a certain error criterion. Currently, most integrator control the local error which is the error introduced at the current step ignoring the error that propagates from previous steps. A formal definition of the local error at time  $t_{n+1}$  for the differential equations (2.2.1) is given by,

$$\underline{\ell}_{n+1} = \underline{y}_{n+1} - \underline{u}_n(t_{n+1}) \quad (2.2.9)$$

where  $\underline{y}_{n+1}$  is the computed solution at time  $t_{n+1}$  and  $\underline{u}_n(t)$  is the local solution which satisfies the differential equations

$$\underline{u}'_n(t) = \underline{f}(t, \underline{u}_n(t)), \text{ for } t \geq t_n \quad (2.2.10)$$

$$\text{and } \underline{u}_n(t_n) = \underline{y}_n$$

The global error on the other hand is the actual error incurred in the computed solution and is defined as

$$\underline{q}_{n+1} = \underline{y}_{n+1} - \underline{y}(t_{n+1}) \quad (2.2.11)$$

where  $\underline{y}(t)$  is the true solution of eqn.(2.2.1). A diagram depicting the meanings of the local and global error is given in Fig. (2.2.2).

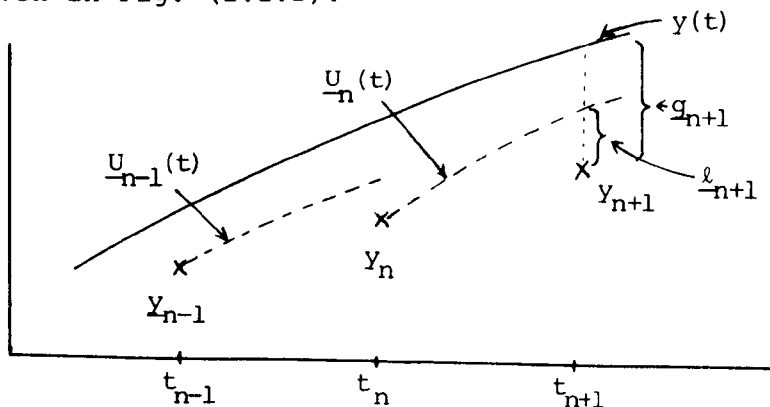


Fig. 2.2.2 - The Local and Global Errors

Ideally one would like to control the global error, but because this is computationally very expensive, it is normal to control it indirectly by ensuring that the local error is less than some user specified tolerances. A general outline of a variable-step code with local error control is given by,

REPEAT

    Compute the numerical solution at the new time step.  
    Estimate the local error.

    Accept the step if the local error is less than the required error tolerance, EPS; otherwise reduce the stepsize and repeat the above process.

    Estimate the new stepsize and increment the time step.

UNTIL the end condition is reached.

Most numerical methods which possess some of the stability properties discussed in the previous section are implicit in nature. These methods, with the exception of the Rosenbrock-type method, require the solution of at least one system of nonlinear algebraic equations at each time step. When the method is applied to solve eqn.(2.2.1), the system of algebraic equations has the form,

$$\underline{F}(\underline{y}) = \underline{y} - h\beta \underline{f}(\underline{y}) - \underline{\phi} = 0 \quad (2.2.12)$$

where  $\beta$  is a parameter depending on the particular integration method employed and  $\underline{\phi}$  is a combination of known function values and derivatives. Using the Newton-Raphson method to solve the above system of equations gives

$$[J(\underline{y}^{(m-1)})] \underline{w}^{(m)} = - \underline{F}(\underline{y}^{(m-1)}) \quad (2.2.13)$$

$m = 1, 2, 3, \dots$

where  $\underline{w}^{(m)} = \underline{y}^{(m)} - \underline{y}^{(m-1)}$

$$J(\underline{y}^{(m-1)}) = \partial \underline{F}(\underline{y}^{(m-1)}) / \partial \underline{y} = I - h\beta \partial \underline{f}(\underline{y}^{(m-1)}) / \partial \underline{y} \quad (2.2.14)$$

As it is both expensive and unnecessary to re-evaluate the matrix  $J$  for each iteration, a variant of the Newton-Raphson method is normally used. This method is referred to as the modified-Newton method and it takes the form

$$G \underline{w}^{(m)} = - \underline{F}(\underline{y}^{(m-1)}) \quad (2.2.15)$$

where  $G$  is an evaluation of  $\partial \underline{F}(\underline{y}) / \partial \underline{y}$  at some point  $(t, \underline{y})$  which may be any previous iteration or step. The matrix  $G$  is only updated when the rate of convergence of the iteration is slow. A measure of the rate of convergence is given by

$$r_c = \|\underline{w}^{(m)}\| / \|\underline{w}^{(m-1)}\| \quad (2.2.16)$$

and it is considered unsatisfactory, for example, when  $r_c > 0.5$ , although the exact choice of  $r_c$  is implementation dependent.

The iteration is continued until  $\|\underline{w}^{(m)}\|$  is less than  $\alpha \cdot \text{EPS}$ , for  $0 < \alpha \leq 1.0$ . If the iteration appears to be diverging (i.e.  $r_c > 1$ ) or the rate of convergence is considered unsatisfactory even after  $G$  has been updated, then the stepsize is also reduced.

At the end of the step, the local error is estimated. The step is accepted when the local error estimate satisfies the error tolerance. It is necessary to predict a stepsize that can be used for the next step and it can be found using the formula:

$$h_{\text{new}} = r_k h_{\text{old}} \quad (2.2.17)$$

where  $r_k = (\rho_k \cdot \text{EPS} / \|\underline{\ell}\|)^{\frac{1}{k+1}}$ ,  $0 < \rho_k < 1$ ;

$\|\underline{\ell}\|$  is the local error estimate measured in a suitable norm and  $k$  is the order of the integration method used. This formula is based on the assumption that the local error estimate is of order  $O(h^{k+1})$  and it corresponds to an optimal stepsize for a tolerance of  $\rho_k \cdot \text{EPS}$ . Most codes have further restriction on the size and frequency in which the stepsize can be increased. A good discussion of the implementation of a stiff integration method is given in DEW(1978).

### 2.3 A Survey of Stiff Integration Methods

The most widely used stiff integration method is the linear multistep method based on backward differentiation formulae. It was first proposed and implemented by GEAR(1971a, 1971b) and is generally referred to as the Gear's method. The method when applied to (2.2.1) can be written as:

$$\underline{y}_{n+1} = \sum_{i=1}^k \alpha_i \underline{y}_{n-i+1} + h \beta_0 f(t_{n+1}, \underline{y}_{n+1}) \quad (2.3.1)$$

where  $h = t_{n+1} - t_n$  and  $\underline{y}_{n+1}$  is the approximate solution at time  $t_{n+1}$ ;  $k$  is the order of the formula employed and  $\{\alpha_i\}$ ,  $\beta_0$  are known constants which depend on the order  $k$  (see GEAR(1971b)).

The method is A-stable for the first and second order formulae, but is only stiffly stable for higher order formulae of up to order 6. It is usually implemented in the form of a variable-step, variable-order integrator. Most implementations allow formulae of up to order 6 to be used. The details of the implementation can be found, for example, in DEW(1978). The usual implementation is to start the method from order 1 and build-up the order progressively as more information becomes available. Following GEAR(1971b), the method is often implemented using the Nordsieck vector of the form,

$$\underline{v}_n = [\underline{y}_n, h\underline{y}'_n, h^2\underline{y}''_n/2!, \dots, h^k\underline{y}_n^{(k)}/k!] \quad (2.3.2)$$

The use of Nordsieck vector enables the changing of step-size, the prediction of the solution and the local error estimation to be done easily. The modified-Newton method is employed to solve the implicit system of equations.

Many good implementations of the method exist, for example, HINDMARSH(1973), DEW(1978). The method has been shown by many authors (for example, ENRIGHT(1975)) to be very efficient and reliable for solving a wide range of stiff ODE's especially when high accuracy is required.

Most of the classical methods for solving stiff ODE's are based on single-step methods. The simplest one is the theta method which is widely used in practice. The method is defined as

$$\underline{y}_{n+1} = \underline{y}_n + h((1-\theta)\underline{f}_n + \theta\underline{f}_{n+1}) \quad (2.3.3)$$

$$\text{for } 0.5 \leq \theta \leq 1.0$$

when  $\theta = \frac{1}{2}$ , the method is the well known Crank-Nicolson method which is A-stable and second order. For  $\theta > \frac{1}{2}$ , the method is strongly A-stable, however, it is only first order. The method is particularly suitable in cases where only low accuracy solutions (say two to three significant figures) are required. It is usually implemented in a constant-step formulation for solving large scale practical problems (for example, PAN and GANESI). A variable-step version of the method has been implemented by HOPKINS(1976) for solving the ODE's arising from the solution of quasilinear PDE's. A similar version of the program with global error estimation is employed in PROTHERO(1977).

The class of semi- and fully-implicit Runge-Kutta (RK) method discussed in CASH(1982) are also suitable for this application. A q-stage fully implicit RK formula for solving eqn.(2.2.1) can be written

$$\underline{y}_{n+1} = \underline{y}_n + h \sum_{i=1}^q b_i \underline{k}_i \quad (2.3.4)$$

$$\underline{k}_i = \underline{f}(t_n + c_i h, \underline{y}_n + h \sum_{j=1}^q a_{ij} \underline{k}_j), \quad 1 \leq i \leq q.$$

Such formulae can be represented conveniently by the array

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1q} \\ \vdots & \vdots & & & \cdot \\ c_q & a_{q1} & a_{q2} & \cdots & a_{qq} \\ \hline & b_1 & b_2 & \cdots & b_q \end{array} \equiv \begin{array}{c|c} C & \underline{A}_R \\ \hline & \underline{b}^T \end{array} \quad (2.3.5)$$

The fully implicit RK method was introduced by BUTCHER(1964) where he showed that for all  $q$ , there exist a fully implicit  $q$ -stage RK formula of order  $2q$ . EHLE(1969) has shown that all of these maximal order formulae are A-stable and hence it is possible to derive A-stable fully implicit RK formulae of arbitrarily high order. The major drawback of this class of methods is that it is very difficult to implement the methods in an efficient manner. When implemented in the most obvious manner, a system of  $S$  equations using a  $q$ -stage method requires the solution of a simultaneous system of  $sq$  nonlinear equations. Although BUTCHER(1976), BICKART(1977) and VARAH(1979), and subsequent work carried out by BUTCHER(1979) on singly-implicit RK methods, have reduced this amount of work considerably, the methods are still not competitive compared with the Gear's method for solving general stiff problems (CASH(1982)).

A semi-implicit RK method is one whose defining matrix  $A_R$  (see eqn.(2.3.5)) is lower triangular, i.e.  $a_{ij}=0$  for  $j > i$ . This class of method was first considered by NØRSETT (1974) and further studied by CROUZEIX(1975) and ALEXANDER(1977). The maximum attainable order of a  $q$ -stage semi-implicit RK method is  $q+1$ , NØRSETT(1977). These methods are normally constructed so that the diagonal elements are all equal, i.e.  $a_{ii} = a$  for all  $i$ ; they were referred to as the diagonally-implicit RK (DIRK) method by ALEXANDER. The DIRK method has the computational advantage that when the modified-Newton scheme is employed to solve the  $k_i$ 's associated with the method, only one LU decomposition of the iteration matrix along with the solution of  $q$  nonlinear equations are required. Several strongly S-stable diagonally implicit Runge-Kutta methods are proposed and implemented in ALEXANDER(1977); the implementation uses Richardson's extrapolation to estimate the local error. The results given in that paper indicate that these methods can be competitive with Gear's method especially when low precision is required. In order to provide methods with efficient error estimation, a class of embedded DIRK methods have been

derived in CASH(1979). These methods have a lower order method embedded within them so that the error estimate can be obtained at virtually no extra cost. The numerical results given in CASH show some improvement over those given in ALEXANDER.

The other suitable class of methods is the Rosenbrock methods, first proposed by ROSENBROCK(1963) for solving autonomous systems of the form

$$\underline{y}'(t) = \underline{f}(\underline{y}(t)) \quad (2.3.6)$$

This class of method has the major computational advantage that it is only necessary to solve linear systems of algebraic equations but has the disadvantage that exact Jacobian matrix is required at each step. Various modifications to the method have been proposed by CALAHAN(1968), CASH(1976) and BUI(1979). Perhaps the most interesting variant of the method is the one given in STEIHAUG(1979) which requires only an approximate Jacobian matrix. These methods can be extended to the non-autonomous case as indicated in that paper. A q-stage method of STEIHAUG when applied to (2.3.6) gives,

$$\begin{aligned} \underline{y}_{n+1} &= \underline{y}_n + h \sum_{i=1}^q b_i \underline{k}_i \\ W_i \underline{k}_i &= \underline{f}(\underline{y}_n + h \sum_{j=1}^{i-1} a_{ij} \underline{k}_j) + hB \sum_{j=1}^{i-1} d_{ij} \underline{k}_j \end{aligned} \quad (2.3.7)$$

$$i = 1, 2, \dots, q$$

where  $W_i = (I - hd_{ii}B)$  and the matrix B is an approximation to the Jacobian matrix of (2.3.6). We shall only consider this variant of the Rosenbrock-type method.

The maximum attainable order of a q-stage Rosenbrock-type method is  $q+1$ , which is the same as the semi-implicit RK method. For efficiency reasons, the methods are constructed with  $d_{ii} = d$  for all  $i$  so that only one matrix  $(I - hdB)$  needs to be kept. A second order method with embedded error estimate is given in STEIHAUG along with some numerical results. Several strongly A-stable Rosenbrock-type methods with "built-

in" local error estimate are given in SCRATON(1981).

Although the Rosenbrock-type method proposed above is of the desired order for any arbitrary matrix B, in practice, it is still necessary to ensure that B is a "good enough" approximation to the Jacobian matrix in order to achieve the required stability characteristics of the method. This makes the method difficult to implement because there is no information to measure how good the approximation is. To overcome this problem, STEIHAUG uses the local error estimate to provide an indication on when the matrix B is to be updated; alternatively, SCRATON(1981) updates B after five steps. This is an area for further research.

For the Rosenbrock-type method (2.3.7), it is possible to derive a corresponding DIRK method of the form

$$\underline{y}_{n+1} = \underline{y}_n + h \sum_{i=1}^q b_i \underline{k}_i \quad (2.3.8)$$

$$\underline{k}_i = \underline{f}(\underline{y}_n + h \sum_{j=1}^{i-1} (a_{ij} + d_{ij}) \underline{k}_j + h d_i \underline{k}_i)$$

and use the above Rosenbrock-type method to provide an accurate initial prediction for the modified-Newton iteration.

#### 2.4 The Handling of Discontinuities - A Survey

Discontinuities of various kinds occur frequently in the simulation of physical systems. In a study on the handling of discontinuities in an ODE system, ELLISON(1981) points out that there are two types of events that cause discontinuities - one is the time event which happens at the known value of the independent variable t and the other is the state event which occurs when a given function of the dependent variables reaches a certain value. Time events can be handled simply by inspecting the calendar of events and adjusting the integration time step accordingly. The handling of state events, however, is not so straight-forward and it is in this area that much of the research effort is channelled.



The problems to be considered with the state events are the detection and location of the events. The detection is normally done by using switching functions (CARVER(1977)); one is defined for each state event and the  $i^{\text{th}}$  event is assumed to occur when

$$\phi_i(t, \underline{y}) = 0 \quad (2.4.1)$$

The occurrence of the event is readily detected if one of the  $\phi_i$ 's changes sign in the interval  $h$ . It is then located by computing a stepsize  $h^* < h$  which renders  $\phi_i(t+h^*, \underline{y}^*)$  zero.

Several different approaches for locating the state events have been reported in the literature. In the Runge-Kutta algorithm developed by HAY(1974), a separate interpolation algorithm is used for locating the events. HALIN(1976) uses Lie series methods to expand the variables as power series for integration, and points out that the switching functions can also be expanded, thus the discontinuities can be located by exactly the same method that the integration algorithm employs. CARVER(1977) transforms the switching functions into differential equations and appends them onto the original system. Gear's method is then used to solve the resulting differential system which means that the stored Nordsieck vector associated with the switching functions can be used directly to locate the events, no extra function evaluations is required in the process. The algorithm of CARVER(1977) assumes that the number of state events is small compared with the total number of equations, and hence the overhead associated with creating additional differential equations for the switching functions is negligible. To handle problems with frequent discontinuities, an improved version of the above algorithm is employed in CARVER(1978). Here a less comprehensive but simpler approach is used. Given that  $\phi_i(t_n)$  and  $\phi_i(t_n+h)$  have different signs, the solution at time  $t_n+h^*$  is predicted using the stored Nordsieck vector given in eqn.(2.3.2). The intermediate  $\phi_i$  value is then computed and the process is repeated until the required  $h^*$  has been found. The effectiveness of this technique is

demonstrated using a structural problem containing numerous discontinuities.

Once the event is located, the integration must be restarted. The restarting is easily done using a single-step method. However, it is less efficient on a multistep method such as the Gear's method because the method has to be restarted from order 1. In an attempt to overcome this difficulty, GEAR(1980) develops a Runge-Kutta-starter to generate enough information for a four-step multistep method to continue. This should make the multistep method more competitive for solving this type of problem especially when high accuracy solution is needed.

## 2.5 The Solution of DAE system

Many authors (for example GEAR(1971c), BROWN(1973), DEW(1978) and SINCOVEC(1979)) have written codes designed to handle the DAE system. These codes are based on the techniques employed in GEAR (1971c) and do not attempt to control the error in the components of the solution associated with the algebraic equations. It was not until recently that the particular problems associated with the solution of DAE system are recognised and discussed (GEAR(1981)). These problems include the determination of initial conditions, error estimation and stepsize selection. An excellent account on how the DAE system differs from an ODE system is given in PETZOLD(1981). This section summarises the work of PETZOLD and shows how a stiff integration method can be extended to a DAE system. The results are given for a linear DAE system; the extension to the nonlinear case will be considered in chapter 4.

The linear DAE system considered in this section is

$$E\mathbf{y}' = B\mathbf{y} + \mathbf{g}(t), \quad \mathbf{y}(t_0) = \mathbf{y}_0 \quad (2.5.1)$$

where matrices E and B can both be singular. The DAE system (2.5.1) can be transformed into canonical form by using linear transformation (SINCOVEC(1979)). The canonical subsystem

for which a unique solution exists can be written as

$$y_1'(t) = E_1 y_1(t) + g_1(t), \quad y_1(t_0) = y_{1,0} \quad (2.5.2a)$$

$$E_2 y_2'(t) = y_2(t) + g_2(t) \quad , \quad y_2(t_0) = y_{2,0} \quad (2.5.2b)$$

and matrix  $E_2$  has the property that there exists an integer  $m$  such that  $E_2^m = 0$ ,  $E_2^{m-1} \neq 0$ . The value of  $m$  is defined to be the nilpotency of the system. Standard ODE systems have nilpotency  $m=0$ .

PETZOLD points out that although nilpotency is a very important property of the DAE system, it is very difficult to determine in practice. Systems with nilpotency  $m \geq 3$  cannot be solved at all by the current ODE methods because changing the stepsize causes large error in the solution. Although systems with nilpotency  $m \leq 2$  can be solved, extensive modifications to the error estimate and strategies employed in the usual ODE code is needed. We shall therefore restrict our subsequent discussions to the solution of systems with nilpotency  $m \leq 2$ .

Consider a simple DAE system with nilpotency  $m=2$  given by,

$$y_2'(t) = y_1(t) + g_1(t) \quad , \quad y_1(t_0) = y_{1,0} \quad (2.5.3)$$

$$0 = y_2(t) + g_2(t) \quad , \quad y_2(t_0) = y_{2,0}$$

This system has the solution,

$$y_2(t) = -g_2(t)$$

$$y_1(t) = -g_1(t) - g_2'(t) \quad (2.5.4)$$

which shows clearly that the solution only depends on  $g_1$ ,  $g_2$  and  $g_2'$  at the current time, and not on the initial value or the past history of the  $g$ 's. Furthermore, the solution  $y_1(t)$  depends on the derivative of  $g_2(t)$ , which means that

it is discontinuous whenever  $g_2(t)$  is differentiable but not continuously differentiable. Numerical methods have a great deal of difficulty in dealing with this situation.

To gain further insight into the case when some derivatives of  $g(t)$  is discontinuous, consider the problem

$$\begin{aligned} y'_2(t) &= y_1(t) \\ 0 &= y_2(t) - g(t) \end{aligned} \quad (2.5.5)$$

where  $g(t)$  is given by  $g(t) = \begin{cases} 0 & t \leq 0 \\ ct & t > 0 \end{cases}$

Suppose  $y_1 \equiv y_2 \equiv 0$  for  $t < 0$ , and that the usual error estimate which is assumed to be proportional to  $\|y_{n+1} - y_n\|$  is employed. Now, take one step with the backward Euler method to advance the solution from  $t_{n-1}$  to  $t_n$ , with  $t_{n-1} < 0$  and  $t_n > 0$ . Then at time  $t_n$  we obtain

$$\begin{aligned} y_{2,n} &= g(t_n) &= ct_n \\ y_{1,n} &= (g(t_n) - g(t_{n-1}))/h_n &= ct_n/h_n \end{aligned}$$

where  $h_n = t_n - t_{n-1}$ . Now, as  $t_n$  approaches zero,  $h_n$  is bounded away from zero as long as  $t_{n-1} < 0$  is fixed.

So the error estimate

$$\epsilon_n \propto \begin{vmatrix} \|y_{1,n} - y_{1,n-1}\| \\ \|y_{2,n} - y_{2,n-1}\| \end{vmatrix} = \begin{vmatrix} \|ct_n/h_n\| \\ \|ct_n\| \end{vmatrix}$$

approaches zero as  $t_n \rightarrow 0$  and so for some  $t_n > 0$ , the step is accepted.

Since the step to  $t_n$  is accepted, the code will continue, taking another step to  $t_{n+1}$ . The computed solution and the corresponding error estimate are

$$y_{n+1} = \begin{pmatrix} c \\ ct_{n+1} \end{pmatrix}, \quad \underline{\epsilon}_{n+1} \propto \begin{pmatrix} c(1 - \frac{t_n}{h_n}) \\ ch_{n+1} \end{pmatrix}$$

These solutions are exactly correct but the first component of the error estimate is independent of  $h_{n+1}$ , so we cannot make the error estimate as small as we want by reducing  $h_{n+1}$ . Thus, for large enough  $c$ , the code will fail in this step even though the solution is exactly correct.

This example shows the distressing situation that the usual error estimates which are based on the difference between the predictor and corrector seem to bear little resemblance to the actual error incurred in the DAE systems. Furthermore the step control mechanism based on these estimates is likely to fail not on the step which spans the discontinuity, but on the next step afterwards. It is shown in PETZOLD that these difficulties with the error estimate are not limited to problems whose solution are discontinuous, but also to problems with severe changes in the solution.

To gain a better understanding of the source of the problem, an analysis was carried out in PETZOLD on the solution of (2.5.1) using the backward Euler method. In that analysis, it was shown that the actual local error in the solution at time  $t_{n+1}$  for the backward Euler method is,

$$\underline{\epsilon}_{n+1} = (E - h_{n+1}B)^{-1} E \left(\frac{h_{n+1}}{2}\right)^2 y''(\xi) \quad (2.5.6)$$

For problem (2.5.5), this is given by

$$\underline{\epsilon}_{n+1} = \begin{pmatrix} -\frac{h_{n+1}}{2} g''_{n+1}(\xi) \\ 0 \end{pmatrix} \quad (2.5.7)$$

which shows that the actual local error in the solution can be reduced by reducing  $h_{n+1}$  (recall, however, that the difference between the predictor and corrector is not reduced),

so in principle, if we know how to adjust the stepsize, the error in  $y_{n+1}$  can be controlled by locally adjusting  $h_{n+1}$ . The actual error, however, is only of order  $O(h_{n+1})$ , but not  $O(h_{n+1}^2)$  as assumed in the usual error estimate.

It is therefore necessary to derive an error estimate which would reflect the true magnitude of the error. In the solution of stiff problems, SACK-DAVIES (1977) noted that the usual error estimate usually overestimates the true error and he suggests an error estimate for the second derivative methods which is asymptotically correct as  $h \rightarrow 0$ , and is reliable and efficient for very stiff problems. The estimate has the form

$$\underline{\varepsilon}_{n+1} = \left\| W_n^{-1} c_n (y_{n+1}^c - y_{n+1}^p) \right\| \quad (2.5.8)$$

where  $W_n$  is the iteration matrix for the second derivative method and  $c_n (y_{n+1}^c - y_{n+1}^p)$  is the usual error estimate.

It is easily seen that (2.5.8) is similar to (2.5.6) if  $W_n$  is replaced by the iteration matrix for the backward Euler method. This leads to the use of the error estimate of the form

$$\underline{\varepsilon}_{n+1} = \left\| G^{-1} E c_n (y_{n+1}^c - y_{n+1}^p) \right\| \quad (2.5.9)$$

where  $G = E - h_{n+1} \beta B$ ,

and  $\beta$  depends on the integration method used. Since  $G$  is the iteration matrix of the method, an LU decomposition of the matrix is always available.

From eqn. (2.5.6), it can be seen that for the backward Euler method, eqn. (2.5.9) accurately estimates the actual error. It has been verified in PETZOLD that it also accurately reflects the behaviour of the error for all backward differentiation formulae. Lastly, it can also be deduced that this new error estimate is of the right order of magnitude, i.e. of order  $O(h^k)$ , where  $k$  is the order of the numerical method employed.

Even with a reliable error estimate, there are still several practical issues to be considered when implementing the numerical methods. These issues include the implementation of the modified-Newton method for solving the implicit nonlinear systems of equations and the strategies for step acceptance and stepsize selection. A detailed discussion of these issues will be given in section 4.4.

## CHAPTER 3

3. THE SOLUTION OF LINEAR EQUATIONS
  - 3.1 Introduction
  - 3.2 The Partitioning of Matrix G
  - 3.3 The Block Matrix Decomposition Schemes
  - 3.4 The Node Ordering Algorithm
  - 3.5 The Data Structure Set-up
  - 3.6 The Factorization Algorithms
  - 3.7 The Back-substitution and the Storage Requirement
  - 3.8 Comparison
  - 3.9 Summary and Conclusion



## CHAPTER 3

3.1 Introduction

This chapter discusses the solution of linear systems of equations arising from the integration of DAE system (1.3.1). The system has the form,

$$G \underline{x} = \underline{r} \quad (3.1.1)$$

where  $\underline{r}$  is the known right-hand vector for which the solution  $\underline{x}$  is required; and  $G$  is an  $N \times N$  matrix which has the general form,

$$G = E - h\beta A \quad (3.1.2)$$

where  $h$  is the current stepsize and  $\beta$  is a parameter depending on the particular integration method used. For example, when the theta method is used to solve (1.3.1),  $\beta$  is simply equal to  $\theta$  (see eqn. (D.1.3)). In general, the matrix  $G$  is very sparse and a large part of it is symmetric and positive-definite.

If the modified-Newton method is used to implement the numerical method for solving the DAE system, then the matrix  $G$  is simply the iteration matrix of the modified-Newton iteration. In this case, it is more appropriate to use a direct method for solving (3.1.1) since the same factorization of matrix  $G$  can be used to carry out many back-substitutions, hence the cost of factorizing matrix  $G$  averaging over all solutions may be negligible; the iterative methods offer no such advantage (BERESFORD(1980)). In choosing the direct method, it is assumed that there is enough storage to hold the matrix  $G$  in its appropriate factorized form. Research into the use of iterative methods based on conjugate gradient is being done in British Gas (GOODWIN(1982)) and shall not be covered here.

Typically, the solution of sparse linear systems of equations using direct methods can be divided into 3 distinct stages as follows:

- a) The Analysis Stage - this stage determines the appropriate partitioning of matrix  $G$  if necessary, and finds the suitable pivotal order in an attempt to minimise the fill-in and/or the number of arithmetic operations during the factorization and back-substitution stages;
- b) The Factorization Stage - this factorises the matrix  $G$  into the required factors using the pivotal order determined during the analysis stage;
- c) The Back-substitution Stage - it solves for the unknown vector  $\underline{x}$  from the stored factors of  $G$  and the  $\underline{r}$  vector.

The three stages are required successively more frequently when the modified-Newton method is employed in the solution process. The same partitioning and pivotal order determined during an initial analysis stage can be used throughout the simulation. Also the same factorization can be used to carry out many back substitutions. Because the time taken to perform the analysis stage is negligible compared with the time required for the whole simulation, it is sensible to choose a node ordering algorithm which would reduce the factorization and, especially, the back-substitution times.

The three stages will be considered separately. In the following section, the appropriate partitioning of matrix  $G$  is described. This is followed by a discussion on the suitable block matrix decomposition schemes in Section 3.3. Three decomposition schemes are proposed and the remainder of the chapter is concerned with the analysis and implementation of these schemes. Section 3.4 discusses a suitable node ordering algorithm and Section 3.5 outlines the setting up of the data structure. Their factorization and back-substitution processes are described in Sections 3.6 and 3.7 respectively. These two sections also give the theoretical arithmetic operation counts as well as the storage requirements for these processes. The operation counts include only the multiplicative operations (the multiplications and divisions) since the number

of subtractions and additions is about the same for all three cases. These operation counts are derived based on the results quoted in Appendix C. The comparison of the decomposition schemes using actual networks and the concluding remarks are given at the end of the chapter.

The following notations are used throughout this chapter. The number of non-zero components in  $S$  is denoted by  $n(S)$ , where  $S$  may either denote a matrix or a vector. The  $i^{\text{th}}$  row and  $i^{\text{th}}$  column of a given matrix  $M$  is denoted by  $M_i^R$  and  $M_i^C$  respectively; thus  $n(M_i^R)$  gives the number of non-zero components in the  $i^{\text{th}}$  row and  $\sum n(M_i^R) = \sum n(M_i^C) = n(M)$ . Lastly, the lower or upper half of a symmetrical, square matrix  $M$  is denoted by  $M^H$ .

### 3.2 The Partitioning of Matrix G

A large part of matrix  $G$  is symmetric and positive-definite (SPD), so it is essential to take this into consideration when solving the matrix. The advantage of working with SPD matrix has been discussed in DUFF(1980b). Generally, this property means that the diagonal elements of the matrix can be used as pivots during the factorization process without causing numerical instability (WILKINSON(1965)), hence a relatively simple pivotal selection algorithm which aims only at preserving the sparsity of the matrix can be used. A further advantage of dealing with this class of matrix is that the results of graph theory can be used in both the analysis and implementation of its factorization process (PARTER(1961), ROSE(1972)). This is particularly advantageous to network problems as there is an equivalence between the network and the graph associated with the resulting matrix.

The solution of linear equations with the above structure has been considered by AZAR(1975) in the simulation of electrical networks. In his approach, the coefficient matrix is partitioned in such a way that the SPD part of the matrix is factorized first so that its properties can be fully exploited. This idea has also been used in GOLDWATER(1976) in the implementation of the PAN program.

For a general gas transmission network with machines, the unknown variables to be solved are the pressures at the nodes of the network and the flows through the machines. It is convenient to refer to the inlet or outlet node of the machines as machine nodes and the remaining nodes of the network as free nodes. The symmetric and positive definite part of matrix  $G$  corresponds to the unknown pressures at the nodes of the network and the asymmetric part of matrix  $G$  corresponds to the unknown machine variables (i.e. the pressures at machine nodes and the machine flows). Thus the best way to partition the matrix  $G$  is to separate the variables at the free nodes from the machine variables. This leads to the following partitioning (see Section B.5 and eqn. (3.1.1))

$$\left( \begin{array}{c|c|c} A_{11} & A_{12} & O \\ \hline A_{21} & A_{22} & K \\ \hline O & M_1 & M_2 \end{array} \right) \begin{pmatrix} \underline{P} \\ \underline{\Pi} \\ \underline{Q} \end{pmatrix} = \begin{pmatrix} \underline{r}_1 \\ \underline{r}_2 \\ \underline{r}_3 \end{pmatrix} \quad (3.2.1)$$

The vector  $\underline{x}$  has been partitioned into:

- $\underline{P}$  - a vector of length  $\gamma_1$  denoting the pressures at the free nodes;
- $\underline{\Pi}$  - a vector of length  $\gamma_2$  denoting the pressures at the machine nodes and
- $\underline{Q}$  - a vector of length  $\gamma_3$  denoting the unknown machine flows, where  $\gamma_1 + \gamma_2 + \gamma_3 = N$ , the total length of vector  $\underline{x}$ .

The submatrices  $A_{ij}$  denote the connections within the network; the matrix formed by  $A_{ij}$ 's is symmetric and positive-definite with  $A_{21} = A_{12}^T$ . The matrix  $K$  is the flow incidence matrix as defined in eqn. (B.4.4) and submatrices  $M_1$ ,  $M_2$  contain the coefficients of the algebraic machine equations. The submatrices  $K$ ,  $M_1$  and  $M_2$  have at most two non-zero elements per row. For large network, the submatrices are very sparse and in general  $\gamma_1 \gg \gamma_2 \approx 2\gamma_3$  which means that the  $A_{ij}$  submatrices constitute a significant part of matrix  $G$ .

The above partitioning of unknown variables separates the machine variables from the rest of the variables so that

when there is a change in the machine operating constraints, only the machine part of the solution needs to be resolved. Furthermore, it results in a much more natural partitioning of matrix G compared with the one used in GOLDWATER(1976); a much simpler block matrix decomposition scheme can therefore be used to solve for the unknown variables.

### 3.3 The Block Matrix Decomposition Schemes

Several block matrix decomposition algorithms are discussed in GEORGE(1974) for the case when the coefficient matrix is positive-definite. This section extends the ideas given in that paper and GOLDWATER(1976) to decompose the matrix G.

For the partitioning of matrix G given in eqn. (3.2.1), the following block matrix decomposition scheme can be used,

$$G = \left( \begin{array}{c|c|c} L & O & O \\ \hline Z^T & I & O \\ \hline O & O & I \end{array} \right) \left( \begin{array}{c|c|c} L^T & Z & O \\ \hline O & A_p & K \\ \hline O & M_1 & M_2 \end{array} \right) \quad (3.3.1)$$

where L denotes the Cholesky factors of submatrix  $A_{11}$  with  $LL^T = A_{11}$ ,

$$Z = L^{-1}A_{12}$$

$$A_p = A_{22} - Z^TZ$$

and let

$$C = \left( \begin{array}{c|c} A_p & K \\ \hline M_1 & M_2 \end{array} \right) \quad (3.3.2)$$

Here and elsewhere in this thesis, it is understood that inverses are not computed explicitly; only the appropriate triangular factors are stored.

The above decomposition scheme is only efficient when  $\gamma_1 \gg (\gamma_2 + \gamma_3)$  so that the dimension of matrix C is

small compared with  $N$ . In this case the full matrix  $C$  is stored and a simple algorithm can be used to solve for the unknown variables. This decomposition scheme is referred to as decomposition  $D_1$ . It has been employed in the implementation of the Leeds/PAN program (see Section 1.5). The use of full matrix  $C$ , however, does not take into account the symmetric and positive-definite structure of submatrix  $A_p$ , and the sparseness of submatrices  $K$ ,  $M_1$  and  $M_2$ . For big networks with a large number of machines, this could give rise to storage problem and also results in rather more arithmetic operations than is strictly necessary.

With some added complexities, an alternative block matrix decomposition scheme (referred to as decomposition  $D_2$ ) can be derived,

$$G = \begin{pmatrix} L & O & O \\ Z^T & L_p & O \\ O & O & I \end{pmatrix} \begin{pmatrix} I & O & O \\ O & L_p^T & O \\ O & M_1 & I \end{pmatrix} \begin{pmatrix} L^T & Z & O \\ O & I & K_p \\ O & O & \tilde{M} \end{pmatrix} \quad (3.3.3)$$

$$\begin{aligned} \text{where } L_p L_p^T &= A_p \\ K_p &= L_p^{-T} L_p^{-1} K \\ \text{and } \tilde{M} &= M_2 - M_1 K_p \end{aligned}$$

Here the matrix  $C$  is further decomposed to take advantage of its underlying structure and the Cholesky factorization method is used to factorize the submatrix  $A_p$  into  $L_p L_p^T$ . Only a full matrix  $\tilde{M}$  is kept which has a dimension of  $\gamma_3 \times \gamma_3$ .

As pointed out in GEORGE(1974), if  $L$ ,  $A_{12}$  are sufficiently sparse compared with  $Z$ , operations could be saved in the factorization and back-substitution stages by using  $Z$  only implicitly through its definition. That is, during the factorization, we compute  $\tilde{A}_p$  as  $A_{12}^T (L^{-T} (L^{-1} A_{12}))$  rather than  $Z^T Z$ . During the back-substitution, we compute  $Z^T \underline{u}$  and  $Z \underline{v}$  as  $A_{12}^T (L^{-T} \underline{u})$  and  $L^{-1} (A_{12} \underline{v})$  respectively. As  $Z$  is denser than  $A_{12}$ , we can save storage and perhaps arithmetic

operation as well by using  $Z$  in this implicit manner. We shall denote decomposition  $D_2$  with  $Z$  used only implicitly as decomposition  $D_3$ .

The implementation and analysis of these three decomposition schemes will be considered in the following sections.

### 3.4 The Node Ordering Algorithm

#### 3.4.1 The Choice of Node Ordering Algorithm

The decomposition schemes derived in the previous section require the factorization of submatrix  $A_{11}$  and, for decomposition  $D_2$  and  $D_3$ , submatrix  $A_p$  using Cholesky factorization method. These submatrices are very sparse and are symmetric and positive-definite. When a sparse matrix is factorized, it usually suffers fill-in and it is well known that the order in which the matrix is factorized could affect the amount of fill-in considerably. It is therefore important that a judicious choice of the ordering is used in an attempt to minimise the fill-in and/or the amount of arithmetic required to solve the matrix.

The symmetric and positive-definite part of matrix  $G$  corresponds to the graph of the original network. The orderings of submatrices  $A_{11}$  and  $A_p$  are equivalent to the orderings of the free nodes and machine nodes respectively in the network. Before any discussion on the node ordering can be carried out, it is helpful to introduce some basic graph theory terminologies. For our purpose, a graph  $G = (X, E)$  consists of a finite nonempty set  $X$  of nodes together with a prescribed edge set  $E$  of unordered pairs of distinct nodes. Given  $x \in X$ , the adjacent set of  $x$  is defined as

$$\text{Adj}(x) = \{y \in X - \{x\} \mid \{x, y\} \in E\} \quad (3.4.1)$$

and the degree of a node  $x$ , denoted by  $\text{Deg}(x)$ , is simply the number  $|\text{Adj}(x)|$ , where  $|S|$  denotes the number of members

in the set  $S$ . The deficiency of a node  $x$ , denoted by  $\text{Def}(x)$ , is the set of all distinct pairs of  $\text{Adj}(x)$  which are not themselves adjacent, that is

$$\text{Def}(x) = \{\{y,z\} \mid y,z \in \text{Adj}(x), y \neq z, y \notin \text{Adj}(z)\} \quad (3.4.2)$$

To understand the relationship between the factorization of a symmetric positive-definite matrix and the elimination of its graph, consider a simple ordered graph as shown in fig.(3.4.1). Its corresponding matrix is given in fig.(3.4.1a) where the diagonal terms represent the nodes and the off-diagonal terms the edges.

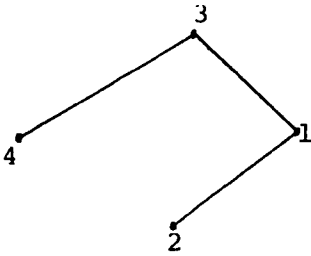


Fig. 3.4.1

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & 0 & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{pmatrix}$$

Fig.3.4.1a

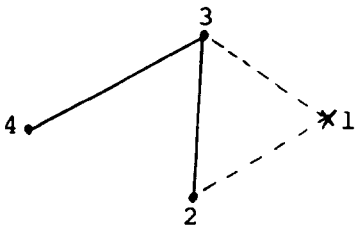


Fig. 3.4.2

$$A' = \begin{pmatrix} X & X & X & 0 \\ X & a'_{22} & a'_{23} & 0 \\ X & a'_{32} & a'_{33} & a'_{34} \\ 0 & 0 & a'_{43} & a'_{44} \end{pmatrix}$$

Fig. 3.4.2a

When the first row is eliminated from matrix  $A$  using Gaussian elimination, the resulting matrix  $A'$  has  $a'_{23}$  and  $a'_{32}$  fill-in terms. This is equivalent to performing the following on the original graph,

- i) deleting node 1 and its incident edges;
- ii) adding edges such that all nodes adjacent to node 1 are pairwise adjacent

as shown in fig.(3.4.2).



In general, if  $y$  is a node in graph  $G$ , the eliminated graph of  $G$  by  $y$ , denoted by  $G_y$  is the graph (ROSE(1972))

$$G_y = \{X - \{y\}, E(X - \{y\}) \cup \text{Def}(y)\} \quad (3.4.3)$$

This shows that when node  $y$  is eliminated from a given symmetric and positive-definite matrix, the number of fill-ins is equal to  $\text{Def}(y)$  and the amount of arithmetic required to eliminate  $y$  is proportional to  $\text{Deg}(y)$ . Various node ordering algorithms which seek to minimize one or both of these quantities during the factorization process are available. They are all heuristic in nature and the more widely used ones are the minimum degree algorithm, the minimum deficiency algorithm, the nested dissection algorithm and the variable-band method (see GEORGE(1981)).

The minimum degree algorithm (MDA) is also known as the second strategy of Tinney (TINNEY(1967)); as the name implies, it chooses, at each stage, the diagonal element with minimum degree as the node for elimination. Experience has shown that this algorithm is very efficient in finding low-fill orderings for a wide class of problems (GEORGE(1980)). The minimum deficiency algorithm (ROSE(1972)) selects as a pivot the node with minimum deficiency. It had been used extensively in the earlier work on power problems (for example, SATO(1963), AZAR(1975)) and is employed in PAN to reorder the network nodes. This ordering involves substantially more work than the MDA and experience has shown that it rarely produced a better ordering than the one produced by the MDA (GEORGE(1981)).

The nested dissection algorithm was first introduced by GEORGE(1973) and generalised by LIPTON(1979). Essentially, it seeks to identify a partitioning of the problem so that the coefficient matrix has the block matrix form similar to that of eqn.(3.2.1). Since the zero block in the matrix remains zero after factorization, the idea can be applied recursively to exploit the zeros of the matrix. This ordering produces an asymptotically optimal ordering for problems with regular grids (GEORGE(1973)) and can do better than the minimum

degree algorithm by a substantial margin on these problems (DUFF(1976)). However, its performance on network problem has been disappointing (BARRY(1978), ERISMAN(1980)).

The variable-band method permutes the matrix so that the nonzeros are near to the diagonal. The most successful algorithm for choosing the ordering of this kind is the Reverse Cuthill-Mckee algorithm with GIBBS(1976) starting point. As the fills are limited to within the bands, by ignoring the zeros within these bands, a very simple variable-band storage scheme (JENNINGS(1966)) can be used. This ordering and the storage scheme are very well suited for the parallel machines (DUFF(1980b)). While this technique is very competitive on problems in structural analysis where the sparsity pattern is very regular, it performs badly on the more irregular problem arising in networks (LEWIS(1980)).

Other ordering algorithms based on one way dissection and refined quotient tree (GEORGE(1981)) have also been proposed and applied successfully to structural problems. They are, however, not competitive on the network problems from the results quoted in LEWIS(1980).

From the above discussion, we conclude that the minimum degree algorithm is the most efficient ordering scheme for the network problems and has therefore been chosen for our simulation task. A further advantage of using this algorithm is that a number of efficient and reliable algorithms exist and hence it can be implemented quite easily.

### 3.4.2 The Detailed Algorithm

This section discusses the detailed implementation of the minimum degree algorithm for the ordering of the network nodes. To facilitate discussions, we shall treat the machine flow variable as an extra node in the network and refer to it as a 'flow node'. A simple test network given in fig.(3.4.3) is used to illustrate the elimination process.

The partitioning of matrix  $G$  in eqn.(3.2.1) means that submatrix  $A_{11}$  and hence the free nodes are eliminated first. The sparsity pattern of submatrix  $A_{11}$  corresponds to the graph of complete network minus the machine and flow nodes and their connections. This decomposes the graph into several components as shown in fig.(3.4.4). (This is generally true for most networks). Different network components are ordered independently using minimum degree algorithm and the resulting matrix is partitioned according to the components of the graph. The node ordering algorithm for reordering the free nodes is as follows:

STEP 1 Decompose the network into components by deleting the machine and flow nodes and their connections from the original network. Let  $G_1, G_2, \dots, G_m$  denote the sub-graphs representing the remaining  $m$  components. Initialise  $i = 1$ .

STEP 2 Let  $G_i = (X_i, E_i)$  and initialise the set  $S = \{0\}$ , Set  $DEG(x) = |Deg(x)|$  for  $x \in X_i$ .

STEP 3 Pick a node  $y \in X_i - S$  where

$$DEG(y) = \min_{x \in X_i - S} (DEG(x))$$

STEP 4 number node  $y$  next and update  $DEG(z)$  for all  $z \in Adj(y)$  and set  $S = (S \cup \{y\})$

STEP 5 if  $S \neq X_i$ , go to step 3

STEP 6 set  $i = i+1$

STEP 7 if  $i = m$  then stop, otherwise goto step 2.

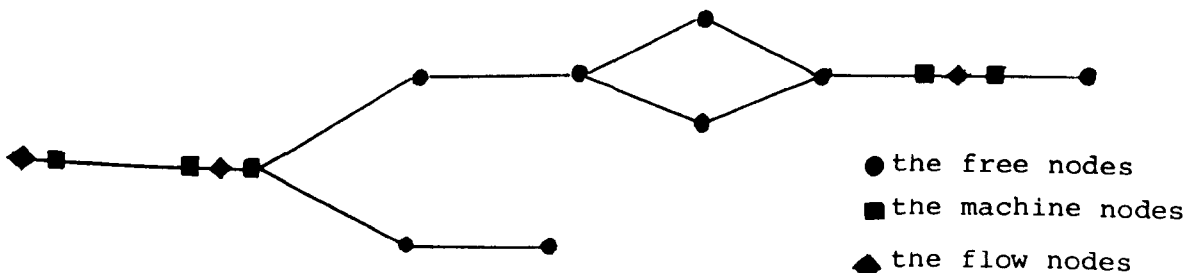


Fig. 3.4.3 - A Test Network

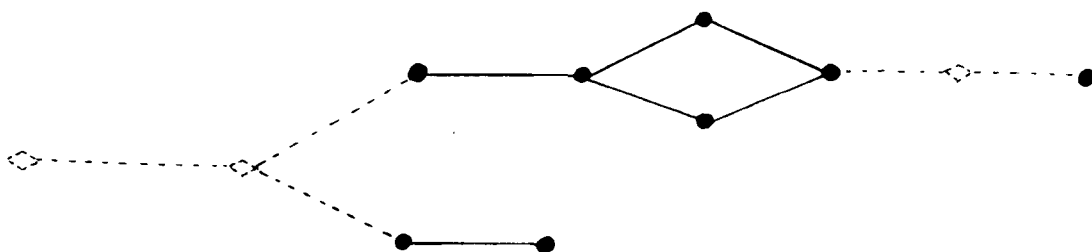


Fig. 3.4.4 - The Graph of Submatrix  $A_{11}$



Fig. 3.4.5 - The Graph of Submatrix  $A_p$

From (3.4.3), it can be easily deduced that the graph which results from the elimination of all free nodes with the flow nodes deleted is given in fig.(3.4.5). It can also be shown that the formation of submatrix  $A_p$  by block (i.e.  $A_p = A_{22} - A_{12}^T A_{11}^{-1} A_{12}$ ) in eqn.(3.3.1) is equivalent to a step by step elimination of the first  $\gamma_1$  rows of matrix  $G$  using Gaussian elimination. Hence the graph of submatrix  $A_p$  is the same as the graph shown in fig.(3.4.5). In practice, this graph can be generated quite easily since in general, two machine nodes will be connected in this graph if there exists a path between them through the set of free nodes without passing through a flow node. The same node ordering algorithm described above can also be used to order the machine nodes except in step 1, the network is decomposed by deleting the flow nodes only.

Finally, the flow nodes are simply numbered in machine order as their ordering will not affect the method in any way since the matrix  $\tilde{M}$  is stored in full. (Note that for decomposition  $D_1$ , the ordering of machine nodes is also immaterial).

### 3.5 The Data Structure Set-up

After the network nodes are ordered, the next stage of the solution is to set up the data structures for storing the nonzero elements of the submatrices involved in the numerical computation. It is necessary to set up the data structure for the final factorized form of these matrices since it is extremely difficult and inefficient to update the data structure during the actual numerical computation. The final factorized form of the matrices can be determined by the process called symbolic computation. As the name implies, it is the process of simulating the numerical computation of the matrices without actually referring to their numerical values. This can be carried out for the factorization of submatrices  $A_{11}$ ,  $A_p$  and the forming of submatrices  $Z$  and  $K_p$  in order to obtain in the zero/nonzero structures of their resulting matrices. Their corresponding data structures can then be set up before the actual computation begins.

The use of sophisticated ordering algorithm such as the minimum degree algorithm for ordering the matrix has the effect of scattering the nonzeros throughout the matrix, a complicated data structure is therefore needed to indicate the positions of all nonzeros within the matrix. A suitable data structure that can be used for this purpose is the 'compressed storage scheme' due to SHERMAN(1975). A detailed description of this storage scheme and a computer program for implementing it is given in GEORGE(1981).

Briefly, this storage scheme requires a set of indexing locations to indicate the position of the nonzeros (usually stored in columns) in the matrix. SHERMAN(1975) noted that it is common for sets of contiguous columns to have a similar structure, hence he proposed to store only the indexing information for the first column in each set so that it can be shared among all other similar columns. In this way, the indexing information is 'compressed' to avoid storing redundant information.

This storage scheme was originally proposed for storing the lower Cholesky factors,  $L$ , of a square non-singular matrix. Since the diagonal elements of  $L$  are non-zeros, they are normally stored in a separate vector so that only the indexing information for the off-diagonal elements are needed. In this way, it is necessary to use up to a maximum of  $(n(L) + N_L)$  indexing locations, where  $N_L$  is the dimension of matrix  $L$ . This is because  $2N_L$  column indicators are required in addition to those needed for the off-diagonal elements. If this storage scheme is used to hold a general matrix  $M$  (not necessarily square) with  $N_C$  columns, then up to a maximum of  $(n(M) + 2N_C)$  indexing locations are required.

The number of indexing locations quoted above is only the upper limit and it is rarely necessary to use more than  $n(L)$  (or  $n(M)$ ) locations; for large sparse matrices, only half of the amount is normally needed (GEORGE(1981), p.142) since most of the indexing information can be 'compressed'.

### 3.6 The Factorization Algorithms

We now provide the algorithms for the factorization of decomposition schemes  $D_1$ ,  $D_2$  and  $D_3$  outlined in Section 3.3.

#### A. Factorization using decomposition $D_1$

- i) Factor  $A_{11}$  into  $LL^T$  using Cholesky method of factorization.
- ii) Compute  $Z$  by solving  $LZ = A_{12}$  and overwrite it into the space occupied by  $A_{12}$ .
- iii) Compute  $A_p = A_{22} - Z^T Z$ ; only the upper half of matrix  $A_p$  is computed since it is also symmetric and positive-definite.
- iv) Set up matrix  $C$  as defined in eqn.(3.3.2) and factorize it into  $L_C U_C$  using full matrix factorization routine.

### B. Factorization using decomposition $D_2$

A more complicated algorithm is needed for decomposition  $D_2$ . Stages i) to iii) are identical to that of  $D_1$ . The remaining stages are:

- iv) Factorize  $A_p$  into  $L_p L_p^T$  using Cholesky factorization method.
- v) Compute  $K_p$  by a) solve for  $L_p H = K$ ; and b) solve  $L_p^T K_p = H$ .
- vi) Compute  $\tilde{M} = M_2 - M_1 K_p$  and factorize it into  $L_M U_M$ , again using full matrix factorization routine.

### C. Factorization Using decomposition $D_3$

For decomposition  $D_3$ , we do not wish to retain the off diagonal block  $Z$ . This implies that the matrix  $A_p$  has to be computed in the "asymmetric" way by  $(A_{22} - A_{12}^T (L^{-T} (L^{-1} A_{12})))$  without explicitly retaining  $Z$ . For convenience in counting the number of arithmetic operations later in this section, the process can be seen as consisting of 3 stages by:-

- a) compute  $Z$ ,
- b) Solve for  $L^T \hat{Z} = Z$  and
- c) compute  $A_p = A_{12}^T \hat{Z}$

Of course in practice,  $A_p$  is computed one row at a time and hence only  $\gamma_2$  temporary storage locations are needed. The algorithm is essentially the same as that of decomposition  $D_2$  except that stage ii) is now discarded and in stage iii),  $A_p$  is set up in the above asymmetric manner.

We can now compare the costs of computing the factorizations for decompositions  $D_1$ ,  $D_2$  and  $D_3$ , where the cost is defined to be the number of arithmetic operations (the multiplications and divisions) required.

#### Theorem 3.6.1

Let the costs of factorizing the matrix  $G$  by decomposition schemes  $D_1$ ,  $D_2$  and  $D_3$  be denoted by  $F_1$ ,  $F_2$  and  $F_3$  respectively. Let the cost of factorizing  $A_{11}$  be  $F_A$ , the costs of factorizing the matrix  $C$  in  $D_1$ , and the matrix  $\tilde{M}$  in  $D_2$  and  $D_3$  be  $F_C$  and  $F_M$  respectively. Then from the lemmas given in appendix C, we have that

$$F_1 = F_A + \sum_{i=1}^{\gamma_1} n(L_i^C) n(Z_i^R) + \sum_{i=1}^{\gamma_1} n(Z_i^R) (n(Z_i^R) + 1) / 2 + F_C \quad (3.6.1)$$

$$F_2 = F_A + \sum_{i=1}^{\gamma_1} n(L_i^C) n(Z_i^R) + \sum_{i=1}^{\gamma_1} n(Z_i^R) (n(Z_i^R) + 1) / 2 + F_R \quad (3.6.2)$$

$$F_3 = F_A + \sum_{i=1}^{\gamma_1} n(L_i^C) n(Z_i^R) + \sum_{i=1}^{\gamma_1} n(L_i^C) n(\hat{Z}_i^R) + \sum_{i=1}^{\gamma_1} n(A_{12i}^R) (\hat{Z}_i^R + 1) / 2 + F_R \quad (3.6.3)$$

where

$$F_R = \sum_{i=1}^{\gamma_2 - 1} (n(L_{P_i}^C) - 1) (n(L_{P_i}^C) + 2) / 2 + \sum_{i=1}^{\gamma_2} n(L_{P_i}^C) n(H_i^R) + \sum_{i=1}^{\gamma_2} n(L_{P_i}^R) n(K_{P_i}^R) + \sum_{i=1}^{\gamma_2} n(M_{1i}^C) n(K_{P_i}^R) + F_M + \gamma_2 (\text{sqrt}) \quad (3.6.4)$$

$$F_A = \sum_{i=1}^{\gamma_1 - 1} (n(L_i^C) - 1) (n(L_i^C) + 2) / 2 + \gamma_1 (\text{sqrt})$$

$$F_C = \frac{1}{3} (\gamma_2 + \gamma_3)^3 - \frac{1}{3} (\gamma_2 + \gamma_3)$$

$$F_M = \frac{1}{3} (\gamma_3)^3 - \frac{1}{3} \gamma_3$$

Proof: To illustrate the proof for these results, we shall only consider  $F_2$ ; the other two follow in a similar manner. In this factorization, we must first factorize  $A_{11}$  leading to a contribution of  $F_A$ . The calculations of  $Z = L^{-1} A_{12}$  and  $\tilde{A}_P = Z^T Z$  require  $\sum_{i=1}^{\gamma_1} n(L_i^C) n(Z_i^R)$  and  $\sum_{i=1}^{\gamma_1} n(Z_i^R) n(Z_i^R) + 1) / 2$  arithmetic operations respectively (by Lemmas C-3 and C-5).

In stage iv), factorizing matrix  $A_P$  into  $L_P L_P^T$  requires  $\sum_{i=1}^{\gamma_2 - 1} (n(L_{P_i}^C) - 1) (n(L_{P_i}^C) + 2) / 2$  arithmetic operations together with  $\gamma_2$  square roots by Lemma C-1. The setting up of matrix



$K_p$  in two steps in stage v) requires  $(\sum_{i=1}^{\gamma_2} n(L_{P_i}^C) n(H_i^R) + \sum_{i=1}^{\gamma_2} n(L_{P_i}^C) n(K_{P_i}^R))$  arithmetic operations by Lemma C-3. Lastly in stage vi), multiply  $M_i K_p$  to give matrix  $\tilde{M}$  requires  $\sum_{i=1}^{\gamma_2} n(M_{i_i}^C) n(K_{P_i}^R)$  arithmetic operations by Lemma C-4, and the cost of factorizing matrix  $\tilde{M}$  into  $L_M U_M$  using full matrix factorization routine is  $F_M$ . Collecting the terms together yields (3.6.2), concluding the proof.

From above, it is obvious that:

$$F_2 < F_1 \iff F_R < F_C \quad (3.6.5)$$

To say more about the results, it is necessary to know the actual sparsity patterns of the submatrices. A more thorough discussion will be given in section 3.8.

### 3.7 The Back-substitution and the Storage Requirement

For decompositions  $D_1$  and  $D_2$ , the back-substitution can be carried out as indicated below:

$$\begin{array}{l} \underline{D}_1 \\ L \tilde{r}_1 = \underline{r}_1 \end{array} \qquad \begin{array}{l} \underline{D}_2 \\ L \tilde{r}_1 = \underline{r}_1 \end{array} \quad (3.7.1)$$

$$\tilde{r}_2 = \underline{r}_2 - Z^T \tilde{r}_1 \qquad \tilde{r}_2 = \underline{r}_2 - Z^T \tilde{r}_1 \quad (3.7.2)$$

$$L_P L_P^T \tilde{r}_2 = \tilde{r}_2 \quad (3.7.3)$$

$$\tilde{r}_3 = \underline{r}_3 - M_1 \tilde{r}_2 \quad (3.7.4)$$

$$L_M U_M Q = \tilde{r}_3 \quad (3.7.5)$$

$$\underline{\Pi} = \tilde{r}_2 - K_P Q \quad (3.7.6)$$

$$\begin{array}{l} L_C U_C \begin{pmatrix} \underline{\Pi} \\ \underline{Q} \end{pmatrix} = \begin{pmatrix} \tilde{r}_2 \\ \tilde{r}_3 \end{pmatrix} \\ L^T P = \tilde{r}_1 - Z \underline{\Pi} \end{array} \qquad \begin{array}{l} L^T P = \tilde{r}_1 - Z \underline{\Pi} \end{array} \quad (3.7.7)$$

The back-substitution procedure for  $D_3$  is similar to that of  $D_2$  except in steps (3.7.2) and (3.7.7),  $L^{-1}A_{12}$  is used in place of matrix  $Z$ .

We assume throughout this section that the zeros in the vectors involved in the back-substitution are not exploited. The intermediate vectors  $\tilde{r}_1, \tilde{r}_2$  etc are only written for convenience, in a computer program the whole operation is executed by overwriting the old vectors into the new ones.

### Theorem 3.7.1

Let the costs of performing the back-substitution for decomposition schemes  $D_1, D_2$  and  $D_3$  be denoted by  $B_1, B_2$  and  $B_3$  respectively. From appendix C, we have

$$B_1 = 2n(L) + 2n(Z) + (\gamma_2 + \gamma_3)^2 \quad (3.7.8)$$

$$B_2 = 2n(L) + 2n(Z) + 2n(L_p) + n(M_1) + n(K_p) + \gamma_3^2 \quad (3.7.9)$$

$$B_3 = 4n(L) + 2n(A_{12}) + 2n(L_p) + n(M_1) + n(K_p) + \gamma_3^2 \quad (3.7.10)$$

**Proof:** We shall carry out the proof for (3.7.9) here; the proofs for (3.7.8) and (3.7.10) can be deduced in a similar manner. The costs of performing steps (3.7.1), (3.7.2) and (3.7.7) are  $2(n(L) + n(Z))$ . The cost of performing step (3.7.3) is  $2n(L_p)$ , and the costs of computing  $M_1 \tilde{r}_2$  and  $K_p \underline{q}$  are  $n(M_1) + n(K_p)$ . Lastly, since the matrix  $\tilde{M}$  is stored in full, the cost of carrying out the back-substitution in step (3.7.5) is  $\gamma_3^2$ . Collecting terms yields (3.7.9), hence proving the theorem.

### Corollary 3.7.2

$$B_2 < B_1 \iff 2n(L_p) + n(M_1) + n(K_p) + \gamma_3^2 < (\gamma_2 + \gamma_3)^2 \quad (3.7.11)$$

$$B_3 < B_2 \iff n(L) + n(A_{12}) < n(Z) \quad (3.7.12)$$

The corollary follows immediately from eqns.(3.7.8) to (3.7.10).

The storage requirements for the decomposition schemes can also be derived directly from the back-substitution processes. In this section, we consider only the storage required to store the matrices that appear in eqns.(3.7.1)-(3.7.7); and in the case of  $D_2$  and  $D_3$ , the additional indexing locations needed for the sparse matrices  $L_p$ ,  $M_1$  and  $K_p$ . The result vectors and other auxilliary stores are approximately the same for all three cases and need not be considered. Main storage locations are required to hold matrices  $L$ ,  $Z$  and  $C$  for decomposition  $D_1$ ; matrices  $L$ ,  $Z$ ,  $L_p$ ,  $K_p$ ,  $M_1$  and  $\tilde{M}$  for decomposition  $D_2$  and matrices  $L$ ,  $A_{12}$ ,  $L_p$ ,  $K_p$ ,  $M_1$  and  $\tilde{M}$  for decomposition  $D_3$ .

### Theorem 3.7.3

Let the amounts of storage required to store the non-zeros of the submatrices involved in decomposition schemes  $D_1$ ,  $D_2$  and  $D_3$  be  $S_1$ ,  $S_2$  and  $S_3$  respectively; and use  $S_I$  to denote the additional integer indexing locations needed for decomposition schemes  $D_2$  and  $D_3$ . We have the followings

$$S_1 = n(L) + n(Z) + (\gamma_2 + \gamma_3)^2 \quad (3.7.13)$$

$$S_2 = n(L) + n(Z) + n(L_p) + n(K_p) + n(M_1) + \gamma_3^2 + S_I \quad (3.7.14)$$

$$S_3 = n(L) + n(A_{12}) + n(L_p) + n(K_p) + n(M_1) + \gamma_3^2 + S_I \quad (3.7.15)$$

where

$$S_I \leq n(L_p) + n(K_p) + \gamma_2 + 4\gamma_3 \quad (3.7.16)$$

Proof: The expressions for  $S_1$ ,  $S_2$  and  $S_3$  can be easily deduced from the proof of theorem 3.7.1, only the expression for  $S_I$  is derived here. Since the compressed storage scheme is used to hold submatrices  $L_p$  and  $K_p$ , up to  $(n(L_p) + \gamma_2)$  plus  $(n(K_p) + 2\gamma_3)$  of integer locations are needed. The submatrix  $M_1$  contains at most  $2\gamma_3$  nonzeros and hence only  $2\gamma_3$  row

indicators are required. Collecting all terms together yields eqn.(3.7.16).

Because of the additional indexing locations,  $S_I$ , needed in decomposition schemes  $D_2$  and  $D_3$ , it is difficult to compare their storage requirements with those of decomposition  $D_1$  as the comparison is largely dependent on how the integer locations can be stored on a particular computer. Instead, a comparison between  $S_2$  and  $S_3$  is carried out.

#### Theorem 3.7.4

The storage requirement of decomposition  $D_3$  is always less than or equal to those of decomposition  $D_2$ .

Proof: By using lemma C-8 and the definition of submatrix  $Z$ , it follows that  $n(A_{12}) \leq n(Z)$ , hence from eqns.(3.7.14) and (3.7.15), we have the above result.

Lastly, to provide an indication on the effectiveness of the decomposition schemes in preserving the sparsity of the matrix, the amount of storage required to hold the original matrix  $G$  (see eqn.(3.2.1)) is also considered. To represent the matrix  $G$ , it is necessary to store at least the nonzeros of matrices  $A_{12}$ ,  $K$ ,  $M_1$ ,  $M_2$  and the lower or upper half of  $A_{11}$  and  $A_{22}$ . This is also the minimum amount of storage needed for solving (3.1.1) if an iterative method (see for example, JACOBS(1980)) were to be used.

#### Theorem 3.7.5

By ignoring the indexing locations needed to represent the sparsities of the submatrices, the minimum amount of storage required to hold the matrix  $G$  is given by

$$S_0 = n(A_{11}^H) + n(A_{12}) + n(A_{22}^H) + n(K) + n(M_1) + n(M_2) \quad (3.7.17)$$

Proof: The proof is similar to that of Theorem (3.7.3).

### 3.8 The Comparison

This section compares the efficiencies of the decomposition schemes using actual networks. Three networks of varied sizes and complexities are considered. The first is a small network shown in fig.(A.4). This network consists of 5 machines with the number of machine nodes greater than the number of free nodes (i.e.  $\gamma_2 > \gamma_1$ ); this is used to investigate the effect of the number of machines on the performance of the decomposition algorithms. The other two are the large and realistic British Gas transmission networks given in figs.(A.5) and (A.7). The second is a regional network consisting of 57 nodes and 15 machines and the third is the large scale national transmission grid with 158 nodes and 20 machines; they are used to test the efficiencies of the decomposition schemes in handling large, sparse matrices. The performance profile of the decomposition schemes on these networks are given in Table (3.8.1); the arithmetic operation counts and the storage requirements are evaluated based on the theorems developed in Sections 3.6 and 3.7.

Decomposition Network		$D_1$	$D_2$	$D_3$	Network Dimension
A.4	F	944	164+9 sqrt	174+9 sqrt	N = 21 $\gamma_1 = 7$ $\gamma_2 = 9$ $\gamma_3 = 5$ $S_3 = 51$ O
	B	230	117	135	
	S	213	86	86	
	$S_I$	-	64	64	
A.5	F	18534	1800+23 sqrt	1988+23 sqrt	N = 72 $\gamma_1 = 34$ $\gamma_2 = 23$ $\gamma_3 = 15$ $S_3 = 168$ O
	B	1686	661	701	
	S	1665	494	455	
	$S_I$	-	213	213	
A.7	F	59098	3770+36 sqrt	4216+36 sqrt	N = 178 $\gamma_1 = 122$ $\gamma_2 = 36$ $\gamma_3 = 20$ $S_3 = 404$ O
	B	3778	1300	1668	
	S	3457	918	872	
	$S_I$	-	277	277	

Table 3.8.1 - The Performance Profile of the Decomposition Schemes

Note - N is the size of the matrix to be solved and  $\gamma_1, \gamma_2, \gamma_3$  are network information as defined in Section 3.2. F, B, S,  $S_I$  and  $S_O$  are quantities as defined in Sections 3.6 and 3.7.

\* only decomposition  $D_1$  has been implemented.

From table (3.8.1), it is clear that decompositions  $D_2$  and  $D_3$  are very efficient compared with decomposition  $D_1$  on all the networks tested. These results clearly show that the added complexity of decompositions  $D_2$  and  $D_3$  pay off handsomely in both reducing the arithmetic operations and storage requirements. For decomposition  $D_3$ , very little saving in storage is obtained compared with  $D_2$  by storing matrix  $Z$  only implicitly while the arithmetic operations required in both the factorization and back-substitution processes are greatly increased. This is because very little fill-in occurs when forming  $Z$ , hence the 'throw-away' strategy employed in  $D_3$  is not suitable for this application.

From the table, it can be seen that the additional indexing locations,  $S_I$ , required in decompositions  $D_2$  and  $D_3$  are very large compared with the total storage requirement,  $S$ . This, however, is misleading since the value given in  $S_I$  is only a grossly overestimated upper limit. In practice, considerably less than this amount is needed especially for large networks since the compressed storage scheme is used to store the matrices (see section 3.5). Furthermore, these integer locations can be packed so very little storage is required.

A comparison between  $S_2$  and  $S_0$  in the above table reveals that decomposition  $D_2$  requires only 2 to 3 times the amount of storage needed for the original matrix  $G$  (this is also the minimum amount of storage needed for an iterative method). Hence the use of decomposition  $D_2$  will not result in a large increase in the storage requirement compared with the iterative method and can therefore be used efficiently for solving eqn. (3.1.1).

### 3.9 Summary and Conclusions

Three decomposition schemes are described and compared using actual test networks. Out of the three schemes considered, only decomposition  $D_1$  has been implemented and is employed in the Leeds/PAN and the variable-step PAN; the other two schemes have not been implemented because of the time factor.

From the results quoted in table (3.8.1), it is clear that decomposition  $D_1$  is very inefficient compared with decompositions  $D_2$  and  $D_3$  especially for large networks with many machines. Also, the 'throw-away' strategy employed in decomposition  $D_3$  has not been found to perform as well as expected compared with decomposition  $D_2$  in reducing the storage requirement. The best scheme arrived from this study is therefore decomposition  $D_2$ ; the results from the previous section indicate that it is a very efficient scheme.

## CHAPTER 4

- 4. THE DESIGN OF THE VARIABLE-STEP INTEGRATOR
  - 4.1 Introduction
  - 4.2 The Restart strategy
  - 4.3 The Selection of Numerical Method and Local Error Estimate
    - 4.3.1 The Choice of Numerical Method
    - 4.3.2 Local Error Estimates for the DAE Systems
    - 4.3.3 The Extension of the Rosenbrock-type Method to the Non-autonomous Systems
  - 4.4 The Strategies Used in the Integrator
    - 4.4.1 The Normal Phase
    - 4.4.2 The Restart Phase
  - 4.5 Further Enhancements



## CHAPTER 4

4.1 Introduction

This chapter discusses the design of the variable-step integrator for solving the DAE system arising from the simulation of gas transmission networks. Although the discussion aims specifically at the network problem, it relies only on the assumption that the DAE system arose from the discretization of a system of parabolic PDE's and that there are a number of severe discontinuities in the first and higher derivatives of the solution. The techniques described here can be employed to solve other similar systems of equations.

The properties of the DAE system (1.3.1) have been discussed in section 1.3. Briefly, the system contains frequent severe disturbances due to the varying consumer demands and the operations of machines in the network; these disturbances result in rapid changes in the solution. To solve a system of this kind using a standard variable-step integrator is computationally very expensive because extremely small time steps will be used immediately after a disturbance, in order to satisfy the required error tolerance. This means that for a number of applications, it is impractical to use a standard variable-step code because reliable solutions are normally required reasonably quickly. The author suspects that it was partly because of this difficulty that all the currently available network analysis programs surveyed in FINCHAM(1980) are constant-step in nature.

The purpose of this chapter is to describe a restart strategy that can be incorporated into a variable-step integrator for handling the severe changes in the gas flow efficiently. Other strategies are also discussed for handling the algebraic equations in the system. The implementation of a general variable-step integrator for solving systems of this kind is described in appendix E.

## 4.2 The Restart Strategy

To make the variable-step integration feasible for the general simulation of gas transmission network, it is essential to modify the variable-step algorithm so that it is detailed enough to produce accurate solutions for the slow dynamics, while not too time consuming for the rapid transients that result from the severe disturbances in the gas flow. This means that separate strategies are needed for handling the slow and rapid changing solutions.

In many applications, the simulation engineers are only interested in modelling the gas flow over a large time interval (in the gas industry, this is typically 24 hours) and are not concerned with accurately following the solution immediately after a severe disturbance in the gas flow. They do, however, require that after a specified time period, the effects of the disturbance are accurately modelled. Under these circumstances, it would be extremely inefficient to compute an accurate solution in this region, as is done when a standard variable-step integrator is used.

The strategies employed here is to use normal error control everywhere except in the region immediately after a severe disturbance in the gas flow where the error control is suspended. These regions are referred to as the normal phase and the restart phase respectively. By suspending the error control during the restart phase enables a larger time step to be used in this region than would otherwise be the case. A constant stepsize is used throughout the restart phase until the estimate of the global error indicates that it is safe to return to normal phase. This ensures that the rest of the solution satisfies the required accuracy tolerance.

The restart strategy relies heavily on the assumption that all the eigenvalues of the Jacobian matrix of the system are negative and hence the computed solution will eventually converge towards the true solution. This assumption is true for systems that arise from the discretization of parabolic PDE's. A theoretical analysis for the restart phase based on

a model problem is given below. The detailed strategies for the restart phase are given in section 4.4.

#### 4.2.1 The Model Problem

To illustrate the technique used for the restart phase, it is helpful to consider a model problem. The simplest one which reflects the behaviour of the gas transmission network problem is given by

$$y' = -\lambda y + d(t), \quad \lambda > 0 \quad (4.2.1)$$

$$\text{where } d(t) = \begin{cases} d_0 & t < t_0 \\ d_1 & t \geq t_0 \end{cases}$$

The function  $d(t)$  models the change in demand value and the parameter  $\lambda$  corresponds to an eigenvalue of the matrix  $E^{-1}A$  where  $E$  is assumed to be non-singular for the purpose of this analysis.

As a discontinuity in function  $d(t)$  occurs at time  $t_0$ , a restart phase is initiated at that time. Let the stepsize chosen for the restart phase be  $h$  and denote the computed solution, the true solution and the local solution after  $j$  steps by  $y_j$ ,  $y(t_j)$  and  $y_L(t_j)$  respectively. For simplicity, we shall assume that the global error at time  $t_0$  is negligible compared with the error introduced due to discontinuity, hence at time  $t_0$ ,

$$y_0 = y_L(t_0) = y(t_0) \quad (4.2.2)$$

This assumption is a reasonable one because the local error has been controlled up to time  $t_0$  and therefore will be much smaller than the error in the numerical solution immediately after the discontinuity.

The simplest method for integrating (4.2.1) is the theta method (see eqn.(2.3.3)). Applying the method to (4.2.1),

it follows after  $j$  steps from the start of a restart phase that

$$y_j = w^j y_0 + [1 - w^j] d_1/\lambda \quad (4.2.3)$$

$$\text{where } w = [1 - h(1-\theta)\lambda]/[1 + h\theta\lambda] \quad (4.2.4)$$

and  $\theta > 0.5$ .

From assumption (4.2.2), it can be shown that the true and local solutions at time  $t_j (=t_0 + jh)$  are given by

$$y(t_j) = y_0 e^{-j\lambda h} + [1 - e^{-j\lambda h}]d_1/\lambda \quad (4.2.5)$$

and

$$y_L(t_j) = y_0 w^{j-1} e^{-\lambda h} + [1 - w^{j-1} e^{-\lambda h}]d_1/\lambda \quad (4.2.6)$$

Hence the local and global error estimates at time  $t_j$  are

$$l_j = y_L(t_j) - y_j = (y_0 - d_1/\lambda)[w^{j-1}e^{-\lambda h} - w^j] \quad (4.2.7)$$

and

$$g_j = y(t_j) - y_j = (y_0 - d_1/\lambda)[e^{-j\lambda h} - w^j] \quad (4.2.8)$$

respectively (see fig. (4.2.1)).

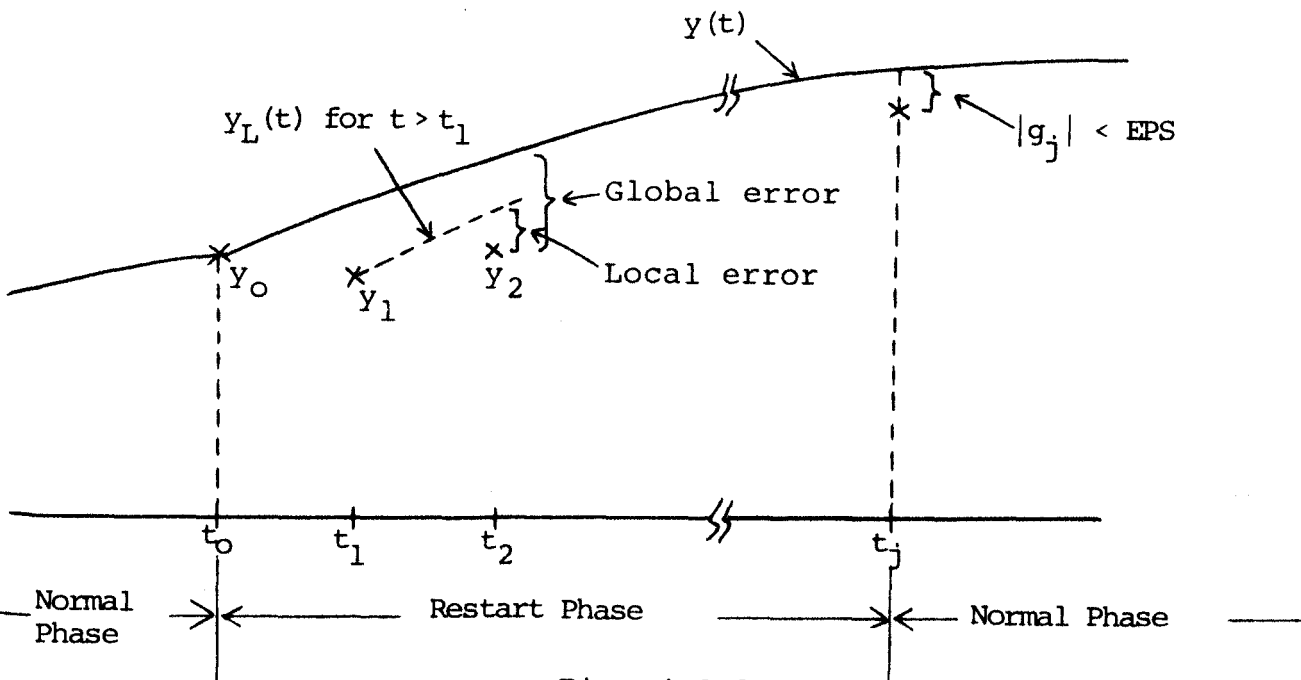


Fig. 4.2.1

For  $\theta > 0.5$ , it can be seen that the terms in the square brackets in eqns. (4.2.7) and (4.2.8) tend to zero as  $j$  tends to infinity. Hence  $|l_j|, |g_j| \rightarrow 0$  as  $j \rightarrow \infty$  as required.

This shows that the restart strategy will eventually terminate for the model problem. Before returning to the normal phase, it is necessary to ensure that the local solution is sufficiently close to the true solution so that it is meaningful to apply local error control. The difference between the local and true solutions after  $j$  steps of computation is given by

$$\text{DIFF} = y(t_j) - y_L(t_j) = e^{-\lambda h} g_{j-1} \quad (4.2.9)$$

Thus, the restart phase is terminated when  $|\text{DIFF}|$  is less than the local error tolerance EPS; that is when

$$|g_j| < \text{EPS} \quad (4.2.10)$$

Hence during the restart phase the global error is estimated after each step and normal error control is resumed when this estimate is less than the local error control.

Although this analysis has been carried out for the theta method, it naturally extends to L-stable methods and in particular, those given in Appendix D.

### 4.3 The Choice of Numerical Method and Local Error Estimate

A general outline of a variable-step code with local error control has been discussed in section 2.2. There are three aspects of the variable-step code that must be considered. The first is the selection of a suitable numerical method together with a formula for estimating the local error. The second is the design of a suitable strategy for selecting the stepsize, deciding when to update the Jacobian matrix associated with the DAE system and deciding when to accept a particular step. The third is the solution

of linear systems of equations which has been discussed in Chapter 3.

This section discusses the choice of the numerical method and local error estimate. The design of the strategies is discussed in the following section.

#### 4.3.1 The Choice of Numerical Method

The choice of the numerical method poses a problem because of the wide variety of methods that have been proposed over recent years. It is possible to restrict our choice to methods designed for stiff ODE's because it is known that the DAE system (1.3.1) is very stiff when there are relatively short pipes in the network.

For the gas transmission network problem, typically only low accuracy solutions of up to two significant figures are required because the initial data and the machine characteristics are only known approximately. The occurrence of disturbances due to the changes in consumer demand and the operations of machines means that there are many restart phases which imposes a severe restriction on the stepsize. Hence the numerical method selected should be single step with low order of accuracy so that restarting can be carried out easily and efficiently.

Multistep methods (e.g. Gear's method) are inappropriate because frequent restarting means that these methods are very inefficient. Although GEAR(1980) has recently proposed to use a Runge-Kutta-like starter to overcome this difficulty, this new approach is only useful when a high order formula (for example order 3 and above) is used, for orders less than 3, a fixed-order single step method is more efficient and easier to implement than the multistep formula.

Lastly, the numerical method chosen must also conserve the mass of the gas flow. Methods which do not possess this property, such as the Hopscotch method (GOURLAY(1970)), are not applicable.

Because of the restriction on the stepsize, the obvious choice of the method to be considered is the theta method which is widely used in practice. The second method considered is the 3-stage, second order, strongly S-stable embedded DIRK method discussed in CASH(1979). This method has a third order formula with a second order one embedded in it so that the error estimate can be obtained at virtually no extra cost; although it is the third order solution that is normally accepted at the end of the step, the error estimate is only second order and hence it is a second order process. The extension of these two methods to the DAE system is given in Appendix D.

Another class of method to be investigated is the Rosenbrock-type method, given in eqn.(2.3.7), that have appeared recently in the literature. This class of method is interesting because it is only necessary to solve linear systems of equations at each time step. The methods, however, are usually derived for the autonomous systems and hence it is necessary to extend these methods to the non-autonomous case. A suitable method that can be used for this application is the 2-stage, second order, strongly A-stable method given in SCRATON(1981); this formula has a "built-in" local error estimate. As a result of choosing this method, a second order, strongly A-stable DIRK method using the above Rosenbrock-type method as a predictor is also considered. The details of this pair of methods in their usual autonomous form are described in appendix D. The extension of these methods to the non-autonomous DAE system (1.3.1) is given in section 4.3.3.

All numerical methods chosen require the solution of linear systems of equations (3.1.1) with the coefficient matrix  $G$  given by eqn.(3.1.2). It is convenient to refer to matrix  $G$  as iteration matrix throughout this chapter.

#### 4.3.2 Local Error Estimates for DAE Systems

As the local error estimates for the numerical methods chosen are normally given for differential equations written

in normal form, it is necessary to extend these estimates in a DAE system.

A thorough discussion on the difficulties associated with the Solution of DAE system using ODE methods based on the work of PETZOLD(1981) is given in section 2.5. The main results arrived from that discussion, using the backward Euler method, are:

- i) the usual local error estimate which is assumed to be proportional to the difference between the predictor and corrector usually severely overestimates the actual local error incurred in the DAE system and
- ii) the actual local error is of order  $O(h^k)$  and not  $O(h^{k+1})$  as assumed in the usual error estimate, where  $k$  is the order of the numerical method employed.

Analysis carried out in section 2.5 shows that for the DAE system, the actual local error for the backward Euler method is given by

$$\underline{e} = G^{-1}E \left( \frac{h^2}{2} \underline{y}''(\xi) \right) \quad (4.3.1)$$

where  $h = t_{n+1} - t_n$  and  $t_n \leq \xi \leq t_{n+1}$ . Following the suggestion given by SACK-DAVIES(1977), the local error estimate of the form below is attempted,

$$G^{-1}E \underline{\ell} \quad (4.3.2)$$

where  $\underline{\ell}$  is the local error estimate that is normally used for an ODE system. It has been verified in PETZOLD(1981) that this new error estimate is of the right order of magnitude and that it accurately reflects the behaviour of the error for all backward differentiation formulae.

As the factorization of matrix  $G$  is available, the new local error estimate involves only an additional post multiplication by matrix  $E$  and a back-substitution. Extensive tests on the gas transmission network problem have shown that the new error estimate is much more accurate than the usual error estimate, and has been incorporated into the numerical methods described in appendix D.



### 4.3.3 The Extension of the Rosenbrock-type Method to the Non-autonomous Systems

This section discusses the extension of the Rosenbrock-type method and its associated DIRK method to the non-autonomous case, in the similar way as is done for the semi-implicit Runge-Kutta method. We first consider the extension of the Rosenbrock-type method (D.3.2) to the DAE system (1.3.1) using the approach suggested in STEIHAUG(1979). The method is given by

$$\begin{aligned} \underline{y}_{n+1} &= \underline{y}_n + h b_1 \underline{k}_1 + h b_2 \underline{k}_2 \\ G \underline{k}_1 &= \underline{f}(t_n, \underline{y}_n) \\ G \underline{k}_2 &= \underline{f}(t_n + \tau_2 h, \underline{y}_n + h a_{21} \underline{k}_1) + h d_{21} A \underline{k}_1 \end{aligned} \quad (4.3.3)$$

where  $G = E - hdA$ ,  $\tau_2 = a_{21}$  and all coefficients are as defined in section D.3. It can be shown that this formula is second order for any matrix  $A$  and is strongly  $A$ -stable when matrix  $A$  is updated at every step. However, the local error estimate given in eqn.(D.3.4) no longer predicts the actual local error of the method as accurately as in the autonomous case (PAINE (1982)). It can be shown that the actual local error of method (4.3.3), in terms of the 'elementary differentials', is of the form

$$e_{-n+1} = h^3 G^{-1} \left[ \left( \frac{1}{3} - d \right) \frac{\partial \underline{f}}{\partial \underline{y}} \frac{\partial \underline{f}}{\partial \underline{y}} \underline{f} - \left( \frac{1}{6} - \frac{d}{2} \right) \frac{\partial \underline{f}}{\partial \underline{y}} \frac{\partial \underline{f}}{\partial t} \right] + O(h^4) \quad (4.3.4)$$

and the error estimate (D.3.4) differs from the above by the amount  $|0.061 h^3 G^{-1} \frac{\partial \underline{f}}{\partial \underline{y}} \frac{\partial \underline{f}}{\partial t}|$ , which is negligible compared with the first term in eqn.(4.3.4) if  $\| \frac{\partial \underline{f}}{\partial t} \| \ll \| \frac{\partial \underline{f}}{\partial \underline{y}} \|$ .

The method (4.3.3) with the local error estimate (D.3.4) has been applied successfully for solving the gas transmission network problem and, so far, no particular difficulty has been encountered. The reason is that for the DAE system (1.3.1),  $\frac{\partial \underline{f}}{\partial t}$  is simply equal to  $\frac{d\underline{d}(t)}{dt}$ , where  $\underline{d}(t)$  is the vector containing the demands from the network. Since the demands are

approximated by either a step or a linear profile (see the demand profiles in appendix A),  $\frac{\partial \underline{f}}{\partial \underline{t}}$  is either equal to zero or a constant  $|s|$  respectively, where  $|s|$  is the slope of the linear profile and is usually less than 10 in magnitude (since for  $|s| > 10$ , a step profile is normally used by the engineers). In addition, as the DAE system is known to be stiff, hence  $\|\frac{\partial \underline{f}}{\partial \underline{y}}\|$  is large. Thus in this case, the error estimate (D.3.4) can still be used to provide an accurate prediction of the local error.

The DIRK method (D.4.1) using the above Rosenbrock-type method as a predictor can also be extended to the non-autonomous DAE system (1.3.1) as,

$$\underline{y}_{n+1} = \underline{y}_n + hb_1 \underline{k}_1 + hb_2 \underline{k}_2$$

$$E \underline{k}_1 = \underline{f}(t_n + c_1 h, \underline{y}_n + hd \underline{k}_1) \quad (4.3.5)$$

$$E \underline{k}_2 = \underline{f}(t_n + c_2 h, \underline{y}_n + h(a_{21} + d_{21}) \underline{k}_1 + hd \underline{k}_2)$$

where  $c_1 = d$ ,  $c_2 = a_{21} + d_{21} + d$  and all other coefficients are as defined in eqn.(D.3.3). The method can again be shown to be both second order and strongly A-stable. The local error estimate (D.3.4), although not as accurate as in the autonomous case, can still be shown to be a reliable one for the DAE system (1.3.1).

In summary, the Rosenbrock-type method (D.3.2) and its associated DIRK method can be extended quite naturally to the non-autonomous case without affecting the order and stability characteristics of the methods. However, the local error estimate which was originally derived based on an autonomous system will not, in general, be an accurate estimate of the local error of the method; it has been found to be an accurate one for the DAE system (1.3.1) because the system is only mildly nonlinear with respect to the independent variable  $t$ . Hence when the method is to be employed for solving a general non-autonomous system, either a better local error estimate should be found or the system be transformed into the autonomous form.

#### 4.4 The Strategies Used in the Integration

In this section, the strategies used in solving the DAE system (1.3.1) are discussed. The general approach is to use normal variable-step integration with local error control (referred to as normal phase) everywhere except in the region immediately after a disturbance in the gas flow (restart phase).

A number of the strategies described in this section have been determined after extensive numerical testing. This is common in the design of integrators where it is very difficult to determine and justify the strategies theoretically.

##### 4.4.1 The Normal Phase

Because the strategies for the implicit and Rosenbrock-type method are rather different, they are treated separately.

##### 4.4.1.1 Strategies for the Implicit Methods

An integrator based on implicit method requires the solution of systems of nonlinear algebraic equations at each time step. These equations are solved using the modified-Newton method discussed in section 2.2 with the iteration matrix  $G$  given in eqn.(3.1.2).

The usual strategy employed in most ODE codes when the iteration fails to converge satisfactorily is first to update the Jacobian matrix and if it still fails to converge satisfactorily, to reduce the stepsize. This strategy is successful for the ODE systems in normal form because by reducing the stepsize, a more accurate initial prediction for the Newton iteration is obtained and the conditioning of the iteration matrix is improved (as  $h \rightarrow 0$ , the iteration matrix tends to the identity matrix for an ODE system).

The situation, however, is quite different for a DAE system. As discussed in section 2.5, convergence of the

modified-Newton iteration is not ensured simply by reducing  $h$ . Furthermore, the iteration matrix tends towards the matrix  $E$  as the stepsize is reduced. Since the matrix  $E$  is singular, the iteration matrix will become more and more poorly conditioned as the stepsize is reduced which can cause the iteration to diverge. Hence the reduction of stepsize should only be carried out as a last resort.

In addition, the usual ODE strategy is unsatisfactory because for highly nonlinear systems, it is likely that after a few iterations, the solution will have changed so much that the iteration matrix becomes out-dated. Hence there is advantage in updating the iteration matrix even when it has already been updated at the current stepsize. The reduction of stepsize is considered only when repeated updating of iteration matrix fails to resolve the problem.

A new strategy is therefore needed which takes into account the nonlinearity of the problem without updating the iteration matrix too often. It has been found in practice that the best compromise is to update the iteration matrix for a maximum of three times in any one step before the stepsize reduction is considered. This simple modification has been found to be very efficient and robust for solving the gas transmission network problem.

Numerical experience in solving the gas transmission network problem has also revealed that the codes are very sensitive to changes in stepsize. Too frequent or too excessive change in stepsize usually causes the code to fail unnecessarily and should be avoided. Furthermore, the local error estimate (4.3.1) is only of order  $O(h^k)$  and not  $O(h^{k+1})$ , where  $k$  is the order of the method employed, hence a more conservative stepsize strategy should be used. For low order methods, it has been found to be more satisfactory to halve and double the stepsize rather than using the more formal formula of the form  $h_{\text{new}} = h_{\text{old}} (\text{EPS} / \|\underline{\ell}_{n+1}\|)^{1/k}$  for estimating the stepsize  $h$ .

In the light of the above discussions, two sets of strategies have been devised for the implicit methods. They are given by:

The strategies for the solution of implicit system of equations

- i) Predict the initial estimate for the modified-Newton iteration using the current stepsize  $h$ ;
- ii) carry out one modified-Newton iteration and compute the rate of convergence  $r_c$  (see Section 2.2.1);
- iii) if  $r_c > 0.5$   
 then if  $NMAT < 3$   
     then update the iteration matrix and go to step ii)  
     else halve the stepsize and go to step i);  
 where  $NMAT$  is the number of times the iteration matrix has been updated using the stepsize  $h$  at the current iteration;
- iv) the iteration is terminated when the iteration correction  $\|\underline{w}^{(m)}\|$  is less than the error tolerance; otherwise repeat step ii).

The strategies for step acceptance and stepsize selection

- i) if  $\|\underline{\ell}_{n+1}\| > EPS$ , the step is rejected and the stepsize is halved;
- ii) if  $0.15*EPS \leq \|\underline{\ell}_{n+1}\| \leq EPS$ , the step is accepted and the same stepsize is used for next step;
- iii) if  $\|\underline{\ell}_{n+1}\| < 0.15*EPS$ , the step is accepted; the stepsize is doubled provided that at least three successful steps have followed the last change in  $h$ ;

Other stepsize strategies have also been considered and the one given above was found to be the most efficient for solving the gas flow problem.

Lastly, the iteration matrix is also updated when there is a change in stepsize or when the same iteration matrix has been used for more than 25 steps.

4.4.1.2 The Strategies for the Rosenbrock-type Method

The advantage of a Rosenbrock-type method is that it is only necessary to solve linear systems of equations at each time step. However, the method is rather difficult to implement because it is difficult to decide when to update

the iteration matrix  $G$ . Since it is inefficient to recompute the iteration matrix after every time step, a strategy is required to decide when it is advantageous to update it. Several strategies have been attempted; the best of which is described below.

Numerical experience has shown that most of the iteration matrix updatings for the Rosenbrock-type method are carried out due to changing stepsize and step failure. With the discussions in section 4.4.1.1 in mind, a more conservative stepsize strategy is devised in an attempt to minimize the number of step failures and iteration matrix updatings. The same stepsize strategy as the one proposed in the previous section is used and the iteration matrix is updated when

- a) there is a change in stepsize;
- b) the same iteration matrix has been used for more than 15 steps or
- c)  $\| \underline{x}_{n+1} \| > 0.85 * \text{EPS}$

The above strategy for updating the iteration matrix results from extensive numerical testing and is based on those employed in STEIHAUG(1979) and SCRATON(1981).

Lastly, since the method is only linearly implicit, it is sometimes unsatisfactory for handling the nonlinear algebraic equations that may appear in the DAE system. The nonlinear algebraic equations arise when the machines are operating on nonlinear constraints such as the compressor horsepower constraint (see eqn.(B.3.2)). These equations can be solved satisfactorily by the method under normal operating condition. However, when there is a severe disturbance in the gas flow, then the method has difficulty in handling these equations; this is because a large number of iterations is normally needed to ensure the convergence of these equations.

Thus when implementing the method, it is also necessary to check for the convergence of the nonlinear algebraic equations at the end of each step. The  $i$ th nonlinear algebraic equation is considered to be converging satisfactorily when,

$$\text{ABS}(\psi_i(\underline{y})/\bar{y}_i) \leq 0.1 * \text{EPS} \quad (4.4.1)$$

where  $\bar{y}_i = \text{MAX}(1.0, |y_i|)$ ;  $\psi_i(\underline{y})$  is the  $i^{\text{th}}$  nonlinear algebraic equation in the DAE system and  $y_i$  is the corresponding component in the solution vector. If any of these equations have not converged satisfactorily, then the step is repeated using the implicit method. A suitable method that can be used for this purpose is the DIRK method (D.4.1). The problem of non-convergence often occurs during the first step of the restart phase when the solution is changing rapidly.

The implementation of this method into a variable-step integrator is discussed in appendix E.

#### 4.4.2 The Restart Phase

The strategies for the restart phase are independent of the particular numerical method that is used in the integration. They arise out of the analysis given in section 4.2 and numerical testing.

##### 4.4.2.1 The Detection and Location of the Disturbance

Before the restart phase can be initiated, the disturbance must be located. Any disturbances that occurs due to the changes in consumer demand are known at the beginning of the simulation, hence they can be handled quite easily as time events (see section 2.4). The switching of machine operating constraints, however, corresponds to the state events; they are more difficult to handle and an extension of the procedure outlined in CARVER(1978) is employed.

Associated with each type of machine is a set of constraints which defines the operating limits of the machine. These machines are modelled using the algorithm outlined in section 1.3. Briefly, the algorithm initially selects an appropriate set of operating constraints for the machines so that the rest of the machine constraints are satisfied. During the simulation, a machine switches its operating constraint if

one or more of its other constraints are violated, and the constraint which is violated by the greatest percentage is chosen as the new operating constraint for the machine. To detect and locate when the change in machine operating constraint takes place, the constraint functions for the machines are defined. A suitable constraint function takes the form,

$$\phi_m^j(t, \underline{y}) = \delta_m^j - C_m^j(t, \underline{y}) \quad \forall m, j \quad (4.4.2)$$

where  $C_m^j(t, \underline{y})$  is the value of constraint  $j$  for the machine  $m$  at time  $t$ , and  $\delta_m^j$  is its extreme value. For example, when machine  $m$  has constraints on the maximum outlet pressure and maximum flow, then the required constraint functions are given by

$$\phi_m^1 = P_{\max} - P_o \quad (\text{the maximum outlet pressure constraint})$$

$$\phi_m^2 = Q_{\max} - Q_m \quad (\text{the maximum flow constraint})$$

where  $P_o$  and  $Q_m$  are the outlet pressure and flow of machine  $m$  respectively, and  $P_{\max}$ ,  $Q_{\max}$  are their corresponding extreme values. In this case, the machine  $m$  switches its operating constraint when one of the  $\phi_m^j$ 's becomes negative.

At the end of each successful step, the constraint functions for the machines are computed. Changes in machine operating constraints are readily detected if one of the  $\phi$ 's changes sign over the interval  $h$ . Given that  $\phi_m^j(t, \underline{y}_n) \cdot \phi_m^j(t+h, \underline{y}_{n+1}) < 0$ , the interval code based on the bisection method and rational interpolation (BUS(1975)) can be used to compute a new stepsize  $h^*$  which renders  $\phi_m^j(t+h^*, \underline{y}^*) = 0$ . If more than one such  $h^*$  exists for different machines, the smallest one is chosen as this corresponds to the next state event to occur.

To avoid extra function evaluations, only the predicted solution is used in the location process. This means that the disturbance is only located approximately. This is consistent with the idea of suspending the error control during the restart phase.



#### 4.4.2.2 The Detailed Strategies for the Restart Phase

Once a disturbance has been located, the integration must be restarted with the new machine operating conditions and demand flows. During the restart phase, the error control in the solution is suspended; this enables a larger time step to be used in this phase than if the normal phase were enforced.

A suitable stepsize must first be computed for the restart phase. The stepsize chosen must be such that the error in the numerical solution will decrease at each step. It is dependent on the severity of the disturbance and the stiffness of the system at that instance. Following DEW(1978), an estimate of the stepsize is given by,

$$h_R = \text{RESC} * \sqrt{\text{EPS} / \|\underline{f}\|} \quad (4.4.3)$$

where EPS is the user requested error tolerance,  $\underline{f}$  is the updated right-hand function of (1.3.1) and RESC is a parameter depending on the particular numerical method used. Since this formula was originally derived for the first order backward Euler method, it is likely that it underestimates the stepsize that can be used for the higher order methods. To overcome this problem, a larger RESC value is used for the higher order methods. The suitable values of RESC for the numerical methods chosen are given in Appendix D. These values can be reduced quite easily if a more detailed solution is required during the restart phase.

In practice, it is necessary to limit the stepsize that can be used, i.e.

$$h_R \leq h_{\max} \quad (4.4.4)$$

where  $h_{\max}$  is introduced to improve the efficiency of the restart strategy. It has been found from numerical experiment that a suitable value of  $h_{\max}$  is 0.2 hours for the typical 24-hour simulation. Again, this value can be reduced if a more detailed solution is required during the restart phase.

A constant step method is used throughout the restart phase. The same set of strategies (apart from the stepsize strategies) given in section 4.4.1 is used to implement the method. The only exception is that a slight modification to the modified-Newton strategy is made. The usual modified-Newton strategy is to update the iteration matrix whenever the rate of convergence is slow (see section 4.4.1.1); this, however, is unsatisfactory for the restart phase where the solution is changing rapidly. The reason is that the initial prediction for the modified-Newton iteration is likely to be unreliable and hence many iterations will be needed. Thus in order to avoid updating the iteration matrix too soon or too often during the restart phase, it has been found to be necessary to perform at least 5 modified-Newton iterations at the beginning of each step, before the updating of iteration matrix is considered.

To make the restart phase robust, a check is made to see if the local error in successive steps have decreased in magnitude and that the modified-Newton iteration is converging satisfactorily (i.e. within 3 updatings of the iteration matrix in a step). If either of the above is not satisfied, a retry is carried out from the beginning of the restart phase by reducing the stepsize by a factor of 5.

The normal local error control is re-commenced when the global error estimate is less than the required error tolerance providing a minimum of 2 steps have been taken. The last condition was found necessary to ensure that the method is stable and that the transfer from the restart to normal phase is smooth. Numerical experience on both large and small networks has shown that the restart phase is normally terminated in less than 5 steps.

#### 4.5 Further Enhancements

As a result of the experience gained in solving the DAE system (1.3.1) using a variable-step integrator, two possible enhancements in this area can be recommended.

##### A) The Iteration Matrix G for the DAE System

As stated in section 4.4.1, the implementation of a numerical method for solving the DAE system of the form (1.3.1) requires the solution of linear systems of equations of the form

$$G \underline{w} = \underline{r} \quad (4.5.1)$$

$$\text{where} \quad G = E - h\beta A, \quad (4.5.2)$$

$\underline{w}$  contains the required iteration correction vector or the intermediate solution, and  $\underline{r}$  is the known right-hand side vector; all other variables have their usual meanings. The use of matrix G in the above form, however, will lead to the problem of ill-conditioning as h tends to zero because the matrix E is singular for a DAE system.

To overcome this difficulty, a simple modification as suggested in BERZINS(1981) can be used. By separating the equations that correspond to the algebraic equations in the DAE system from the rest of the equations, the system (4.5.1) can be partitioned into the form,

$$\left( \begin{array}{c|c} E_{dd} - h\beta A_{dd} & -h\beta A_{da} \\ \hline -h\beta A_{ad} & -h\beta A_{aa} \end{array} \right) \left( \begin{array}{c} \underline{w}_d \\ \hline \underline{w}_a \end{array} \right) = \left( \begin{array}{c} \underline{r}_d \\ \hline \underline{r}_a \end{array} \right) \quad (4.5.3)$$

where subscripts d and a denote the components relating to the differential and algebraic equations in the DAE system respectively, and the matrices E and A are correspondingly partitioned. As the vector  $\underline{r}_a$  usually contains a factor h, we can therefore divide its corresponding equations (i.e. those equations that correspond to the algebraic equations in the DAE system) by  $-h\beta$ , and the resulting matrix G becomes,

$$G = \left[ \begin{array}{c|c} E_{dd} - h\beta A_{dd} & -h\beta A_{da} \\ \hline A_{ad} & A_{aa} \end{array} \right] \quad (4.5.4)$$

Thus, the new iteration matrix will not give rise to the ill-conditioning problem as  $h$  tends to zero, providing that the submatrices  $E_{dd}$  and  $A_{aa}$  are well-conditioned. This modification will be particularly advantageous to the Rosenbrock-type method as the method is very sensitive to the conditioning of the matrix  $G$ .

#### B) The Consumer Demands from the Network

The demands from the network are normally approximated by using a step function as shown in fig.(5.4). The use of a step function means that the problem is discontinuous and, in general, a restart strategy is needed to handle these discontinuities efficiently. The use of the restart strategy is necessary for handling large, severe changes in demand. However, for a demand profile containing a series of small step changes, as shown in part of fig.(4.5.1), the need to restart the integration at every demand changes will affect the performance of the variable-step integrator considerably.

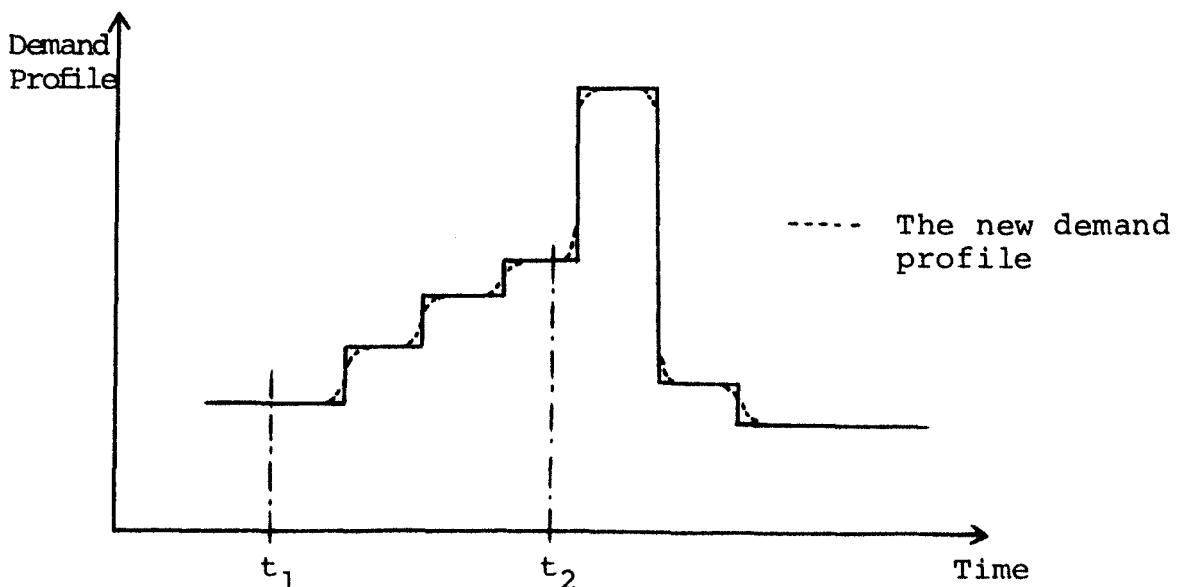


Fig.(4.5.1) - The Demand Profile

By using the idea employed in some parabolic codes in smoothing out the initial conditions, we can instead approximate the step profile by a continuous function by using, for example, the spline technique. An example of the new demand profile is shown in the dotted lines in fig.(4.5.1).

By smoothing out the sharp edges of the step profile, it is hoped that for a series of small demand changes (for example, between time  $t_1$  and  $t_2$  in fig.(4.5.1)), the new profile will be sufficiently smooth to enable the variable-step integration to be used throughout without having to initiate a restart phase. Furthermore, the new profile is a more realistic representation of the actual demand changes because in practice, a short time lag is needed before the demand value can be changed.

## CHAPTER 5

### 5. THE NUMERICAL TESTINGS AND RESULTS

5.1 Introduction

5.2 The Numerical Testings

5.3 The Large Scale British Gas Transmission  
Network

## CHAPTER 5

5.1 Introduction

Four integrators have been developed based on the strategies described in the previous section. The numerical methods used in these integrators are:

- i) the theta method (D.1.1);
- ii) the Rosenbrock-type method (4.3.3);
- iii) the DIRK method (4.3.5) with the above Rosenbrock-type method as a predictor and
- iv) the second order, strongly S-stable embedded DIRK method (D.2.1).

The corresponding integrators developed are referred to as THETA, ROSEN, RDIRK and EDIRK respectively. The details on the implementation of these integrators together with the code of integrator ROSEN can be found in appendix E.

A number of test networks supplied by British Gas are used to illustrate the reliability and accuracy of the integrators. The numerical results obtained from these integrators are compared with those obtained from the Leeds/PAN program. Leeds/PAN is based on a constant-step theta-type method as described in section 1.5; it has been tested against the original PAN program and has been found to be reliable. A 'multi-running technique' is used to establish the stepsize to be used for the Leeds/PAN program in order to obtain a reliable solution. In this technique, a series of solutions is computed by successively reducing the stepsize until the solutions obtained from two successive stepsize are equal to within a specified accuracy tolerance. This technique is a rather tedious but is normally a safe and reliable mean of obtaining the accurate solution. The solution obtained from PAN in this manner can therefore be used as a basis for testing the variable-step integrators.

A set of simple test networks is first used to test the ability of each integrator to handle the types of severe

disturbances that would normally arise from the simulation of a gas transmission network, and to ensure that the numerical methods employed conserve mass. The integrators are then tested and compared using a number of large and realistic transmission networks; the results of the comparison are given at the end of the chapter.

## 5.2 The Numerical Testings

In the initial testing, four simple test networks are chosen. They are used to test the integrators on different aspects of the simulation. The first network is a simple network as shown in fig. (A.1); it contains a sudden, severe step change in the demand profile and is used to test the ability of each integrator to handle large demand changes. The second network (see fig. (A.2)) contains a compressor which shuts off at the beginning of the simulation; this tests the capability of the restart strategy to cope with the disturbance generated by the operation of the machine. The ability of the integrator to handle frequent and linear changes in demand are tested using the network given in fig. (A.3). This network has a varying demand profile as shown in fig. (5.3) attached to the demand node. For this test, it can also be seen that the total inflow of gas at source over the 12-hour period of simulation is equal to the total outflow at the demand node. Hence by the law of conservation, the total amount of gas stored in the network (referred to as linepack) must be conserved and in particular, the linepack should be the same at the end of the simulation as it was at the beginning. Thus this test network can also be used to check whether a numerical method conserves mass. Lastly, to complete the initial testing, a more complicated network as shown in fig. (A.4) is also used. This network consists of five machines which have both linear and nonlinear constraints, and a varying demand profile at the demand nodes; the machines change operating constraints many times during the 24-hour period of simulation. It is used to test the robustness of the integrators in



handling nonlinear operating constraints (i.e. nonlinear algebraic equations) and other important types of disturbances normally encountered in a large scale gas transmission network.

To facilitate comparison, the results obtained from the integrators and the Leeds/PAN program are plotted as shown in figs. (5.1)-(5.4) for networks (A1)-(A4) respectively. The graphs show the variation of pressure against time over the whole period of simulation for the node where the largest changes in pressure takes place; the actual pressure, instead of the difference in pressure between the programs, is plotted in order to provide an overall view on how the solution is changing with the disturbances. Three sets of results at different stepsizes for the Leeds/PAN program are included for each test. This demonstrates how the PAN solutions are converging and provides an indication on the accuracy of the results obtained from the variable-step integrators. Because the results obtained using each of the variable-step integrators differed only by a maximum of 0.5 psi, in order to avoid complicating the graphs, only the results of one integrator are included. The accuracy tolerance of  $0.5E-2$  is used for the variable-step integration.

From the graphs, it can be seen that the results obtained from the Leeds/PAN program converges to those of the variable-step integrators as the stepsize is reduced. This clearly demonstrates that the integrators are both accurate and reliable. The results of network A4 also indicate that the integrators are able to handle DAE systems with highly nonlinear algebraic equations and containing a large number of disturbances.

Because the test networks considered in this section, except network (A4), are very small and simple, it would be unrealistic to compare the relative efficiency of the variable-step integrators on these networks as the result of the comparison does not necessarily apply to the large scale transmission networks that we are interested in. Instead, the efficiency of the integrators are compared in the next section using three large, realistic transmission networks.

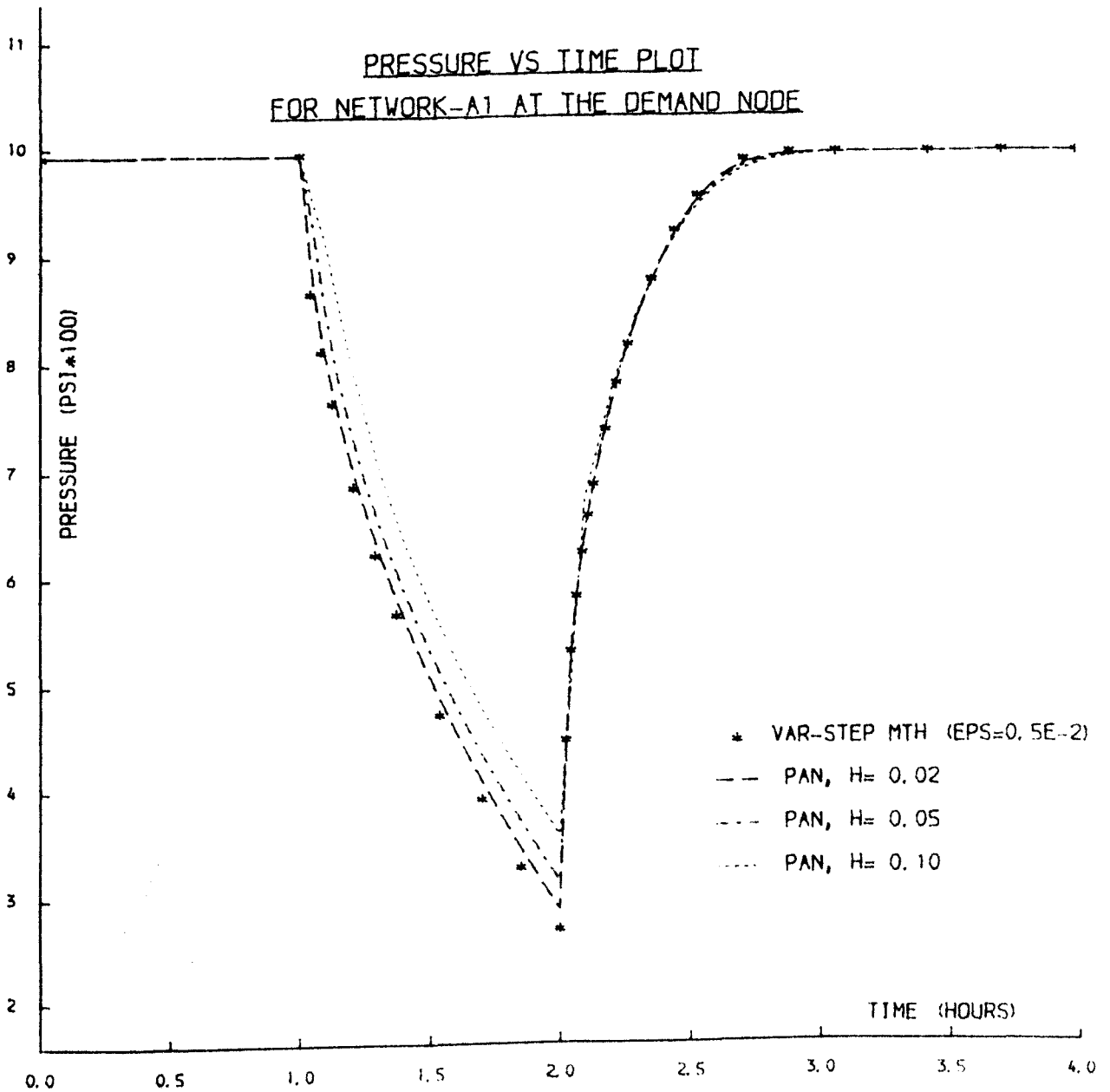
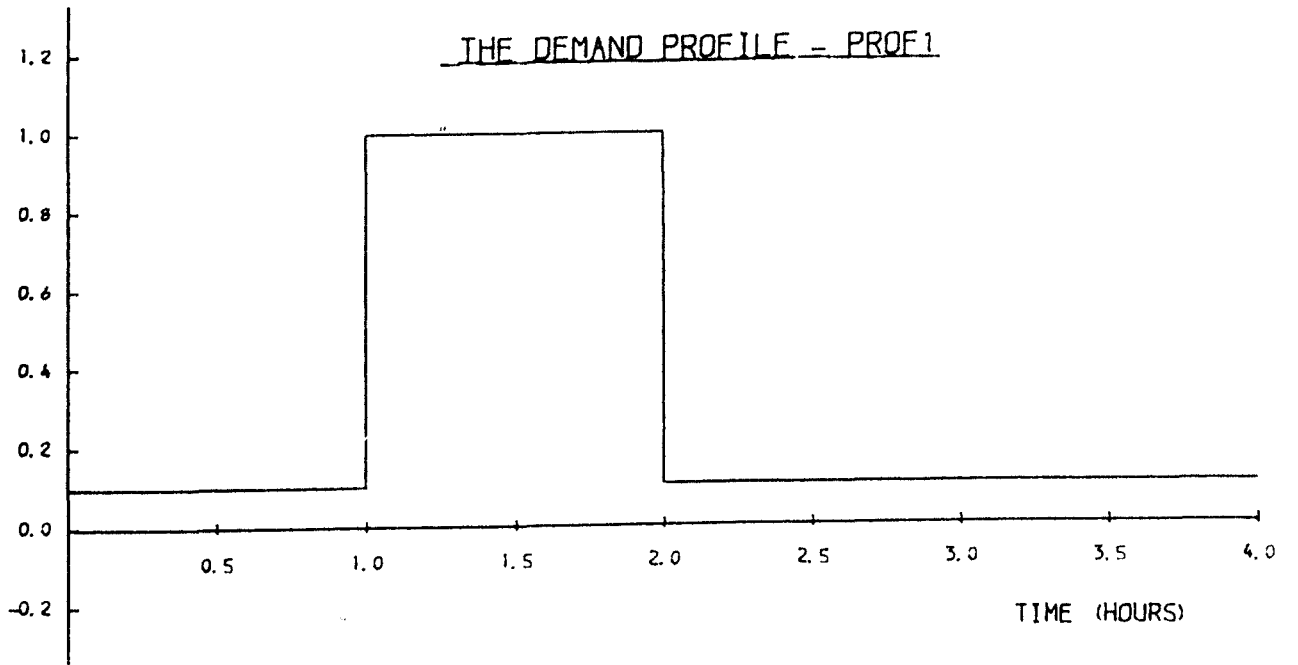


Fig. (5.1)

PRESSURE VS TIME PLOT  
FOR NETWORK-A2 AT THE DEMAND NODE

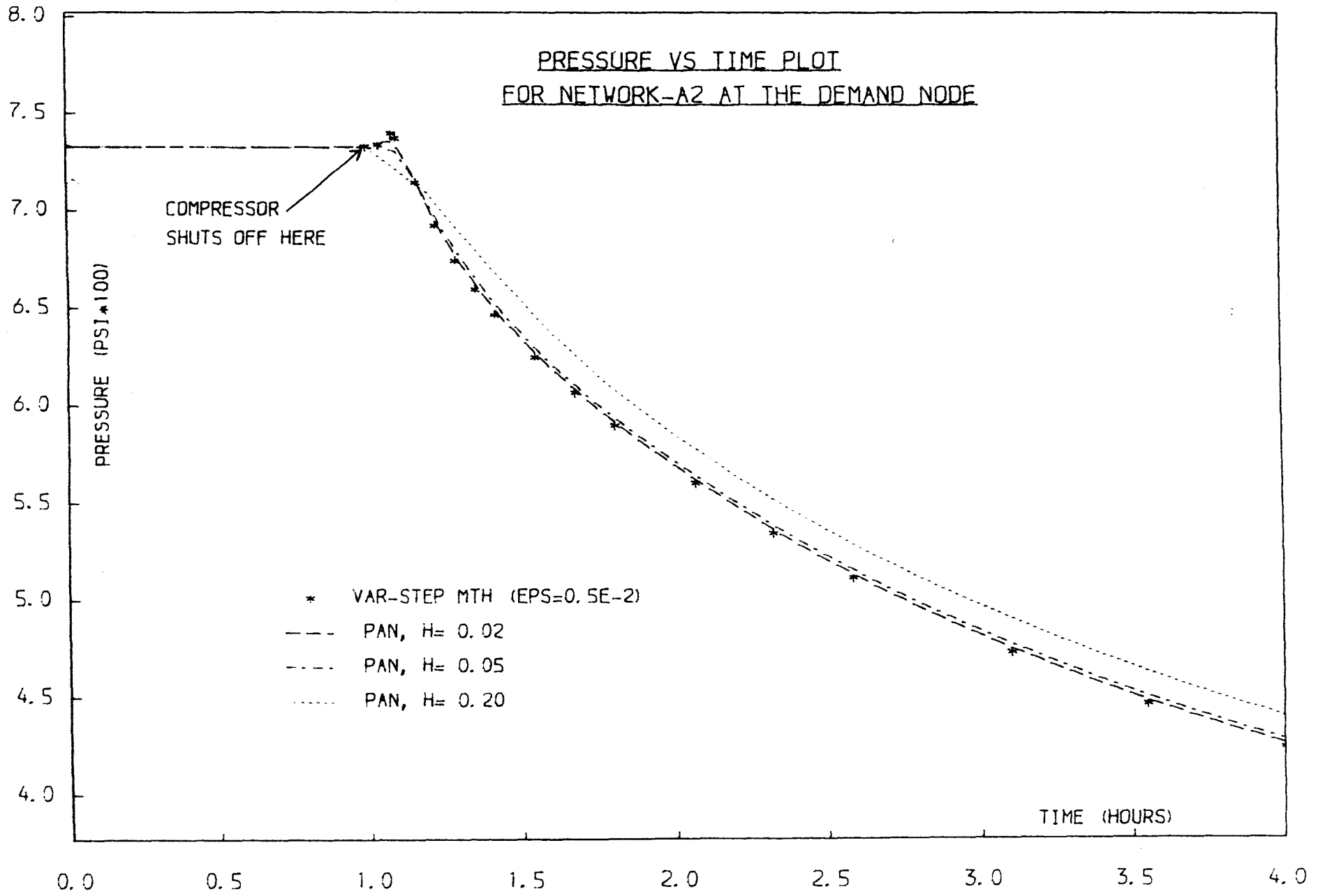


Fig. (5.2)

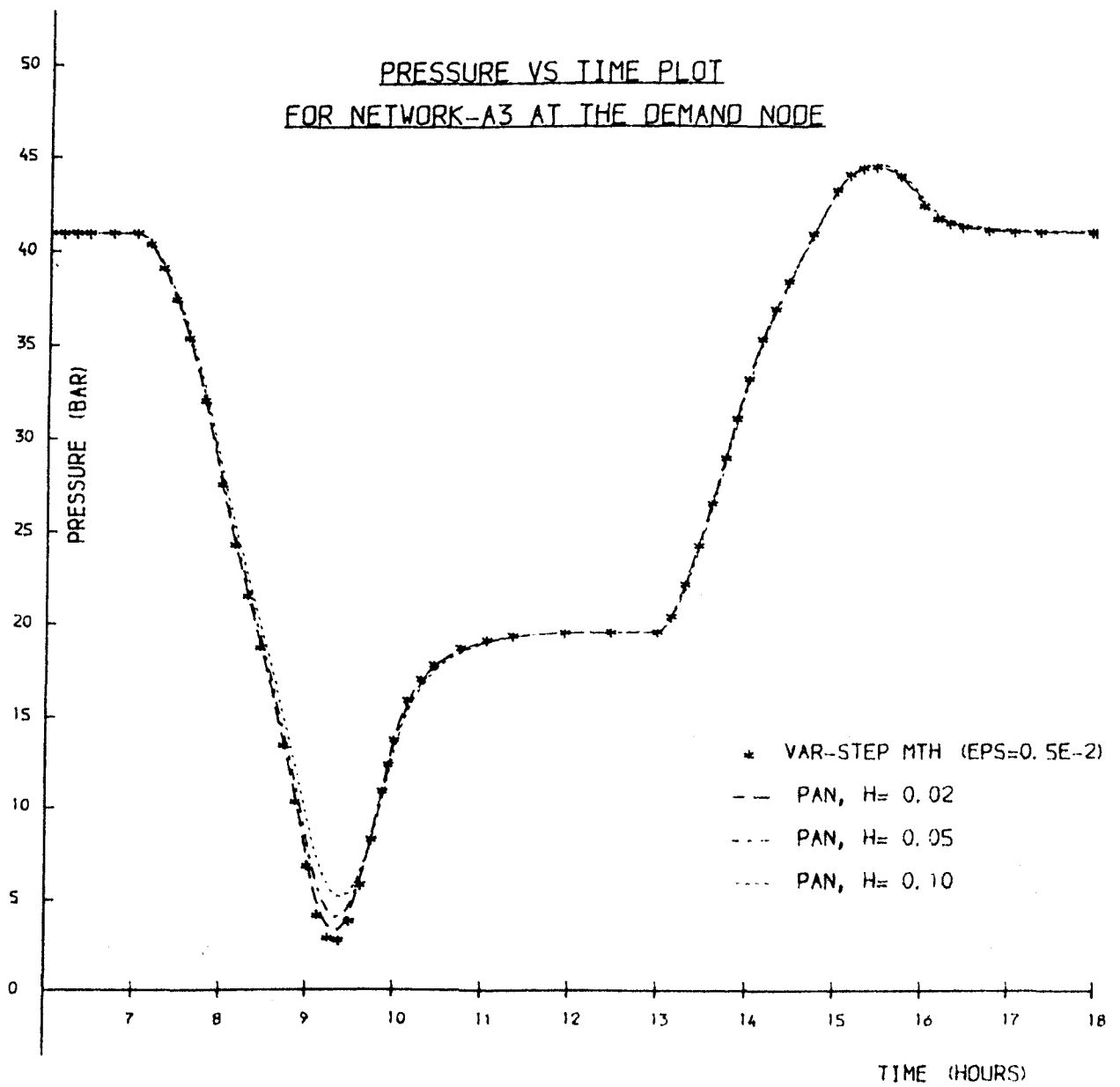
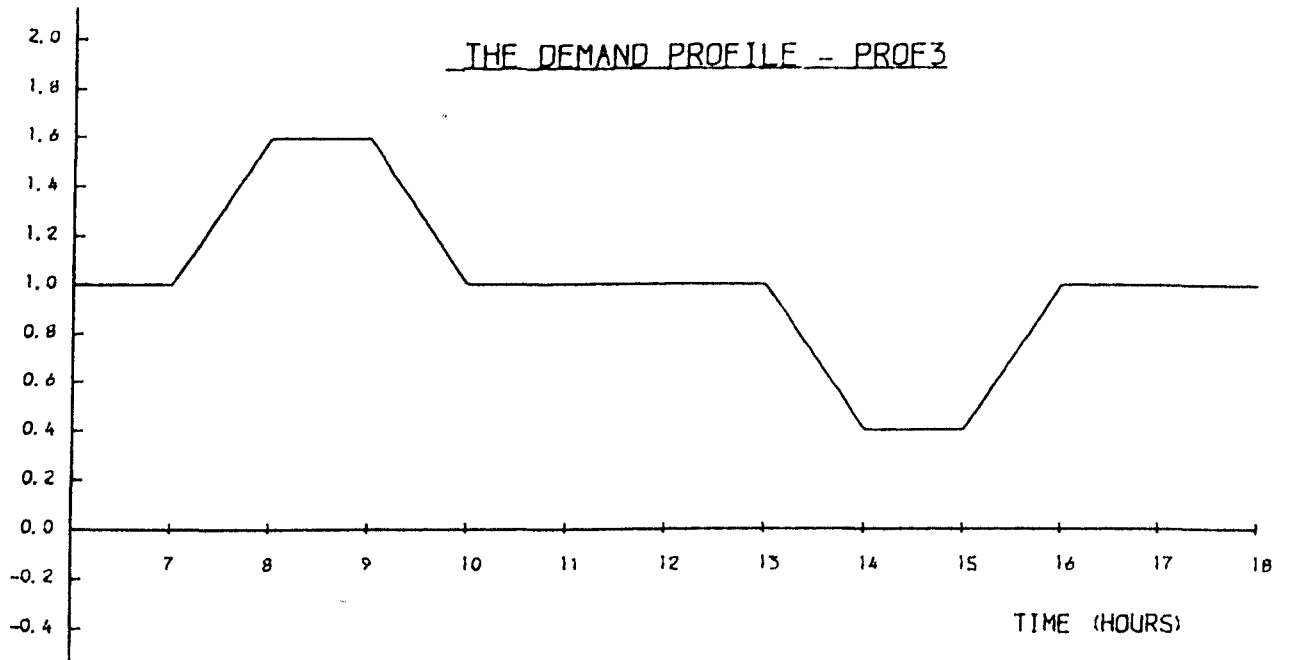


Fig. (5.3)

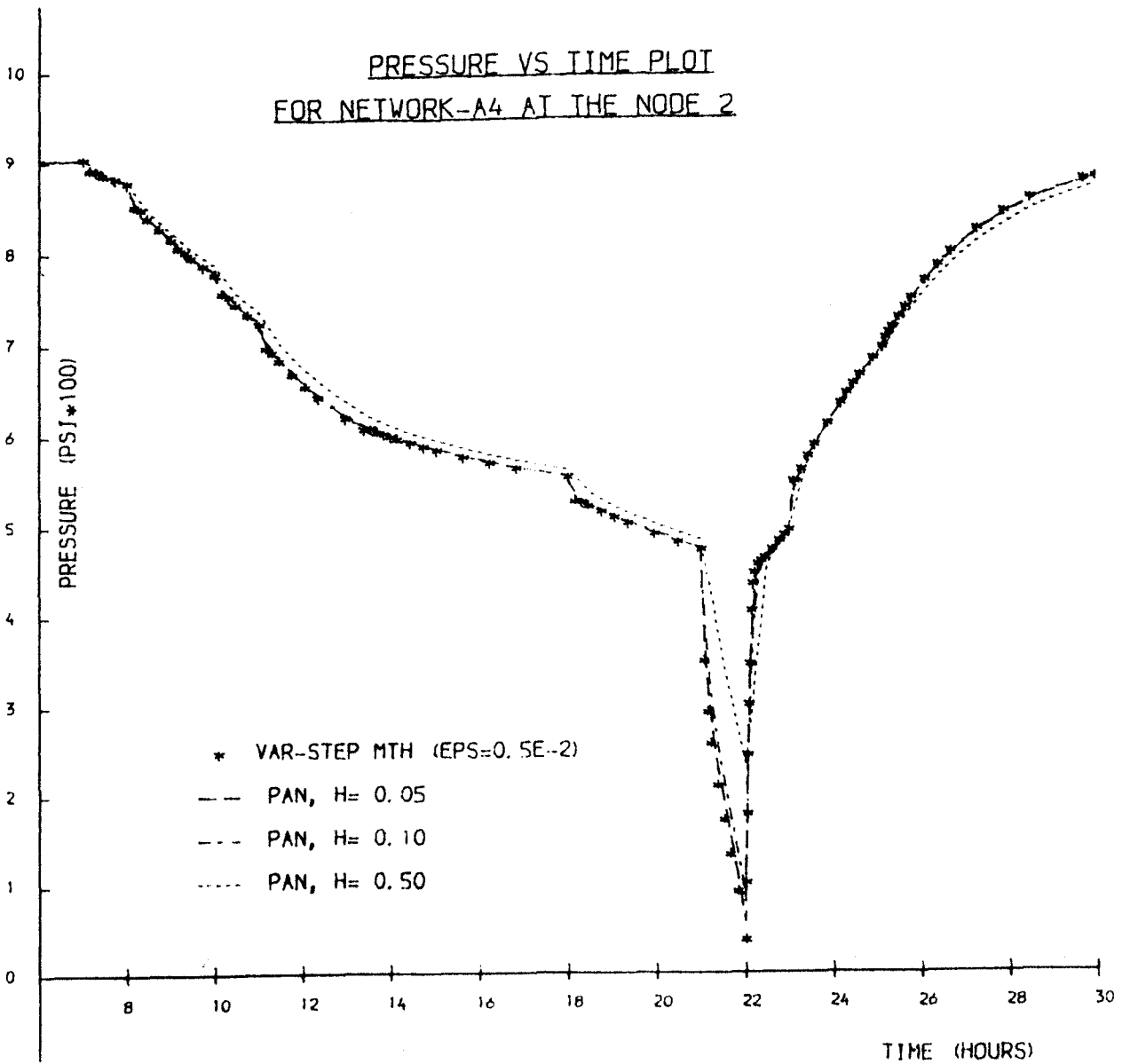
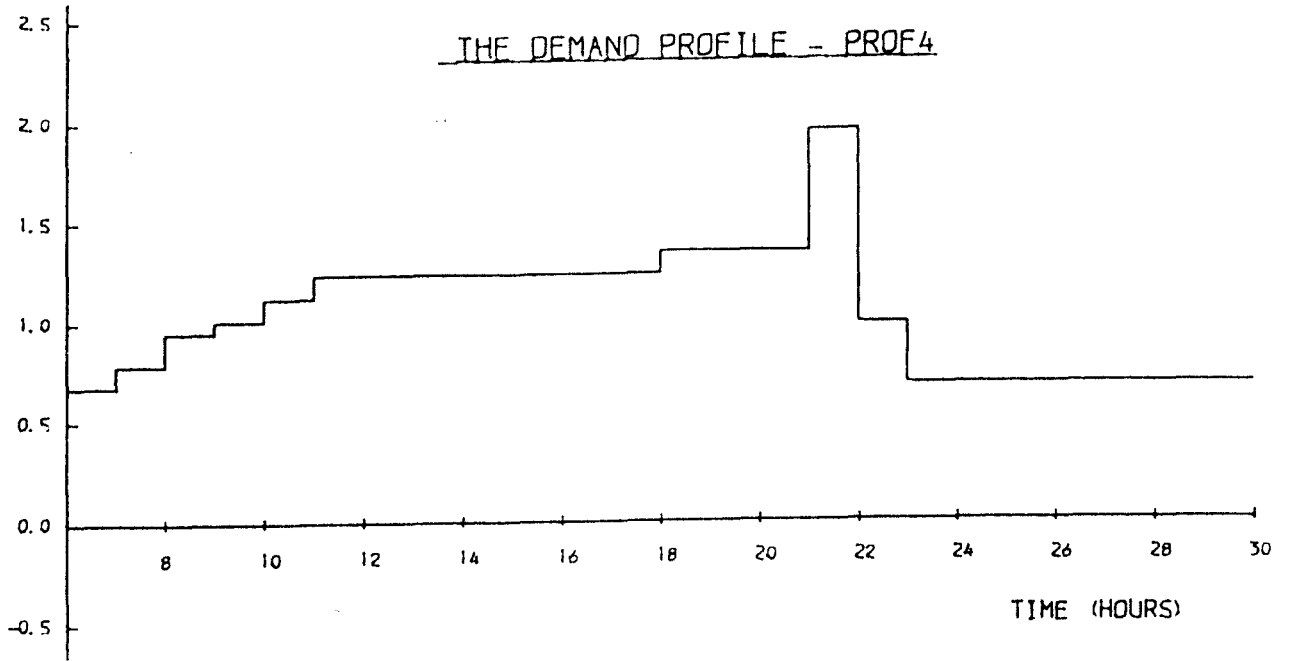


Fig. (5.4)

Finally to test the conservation property of the numerical methods used in the integration, the changes in linepack (where linepack is the amount of gas stored in the network) for network A.3 are also considered. They are listed in Table 5.1 which gives the relative change in linepack at the end of the simulation compared with those at the beginning, for the integrators and the Leeds/PAN program at different stepsizes. The table again shows that the difference in linepack computed by the Leeds/PAN program reduces as the stepsize is reduced, and that the integrators produced a very small difference in linepack which is well within the user requested error tolerance of  $0.5 \text{ E-}2$ . Of all the four integrators considered, the integrators RDIRK and EDIRK are found to perform extremely well in linepack conservation and can be chosen to carry out linepack calculation.

Method	Final Linepack $LP_f$ (mscf)	Relative change in Linepack = $\left  \frac{LP_f - LP_i}{LP_i} \right $
Leeds/PAN h = 0.1 hour	58953	$0.117 \times 10^{-2}$
Leeds/PAN h = 0.05 hour	58919	$0.594 \times 10^{-3}$
Leeds/PAN h = 0.02 hour	58898	$0.238 \times 10^{-3}$
THETA	58869	$0.255 \times 10^{-3}$
ROSEN	58865	$0.323 \times 10^{-3}$
RDIRK	58882	$0.340 \times 10^{-4}$
EDIRK	58886	$0.340 \times 10^{-4}$

where the initial linepack  $LP_i = 58884$  mscf and the error tolerance used in the variable-step integration is  $0.5 \text{ E-}2$ .

Table 5.1 - The Change in Linepack for Network A3.

Other tests on linepack conservation have also been carried out and similar conclusions were obtained. We can therefore conclude that all numerical methods considered conserve mass.

### 5.3 The Large Scale British Gas Transmission Networks

Having established that the integrators work on the set of simple test networks, it is now necessary to test the integrators using some larger and more realistic networks. Three such networks of varied sizes and complexities provided by British Gas are considered. The first two networks are the regional networks and the third one is a version of the National Transmission Network used in British Gas. The sketches of these networks are given in figs. (A.5)-(A.7) with their details being summarised in table 5.2. These networks are all very complicated with varying demands (for example, those given in figs. (5.3)-(5.4)) and machines that change operating constraints frequently throughout the simulation.

The same test procedures as those carried out in section 5.2 are used to check the reliability of the integrators on these networks. For illustration, the graphs of the pressure against time for the demand node where the solution changes rapidly are given in figs. (5.5)-(5.7) for networks (A4)-(A7) respectively; the accuracy tolerance of  $0.5E-2$  is used in the integrators to produce the results. The graphs again show that the results obtained from the variable-step integrators are both accurate and reliable.

Further tests also carried out on the variable-step integrators using a higher accuracy tolerance (EPS) of  $0.5E-4$ . The tests showed that more than three times the amount of CPU-time was required to compute the numerical solution compared with the case when  $EPS=0.5E-2$ . However, only a slight improvement in the accuracy of the solution of up to 0.5 psi for the pressure variables was obtained. This is insignificant for most engineering purposes because the initial data and the machine characteristics are only known approximately. It

PRESSURE VS TIME PLOT  
FOR NETWORK-A5 AT A DEMAND NODE

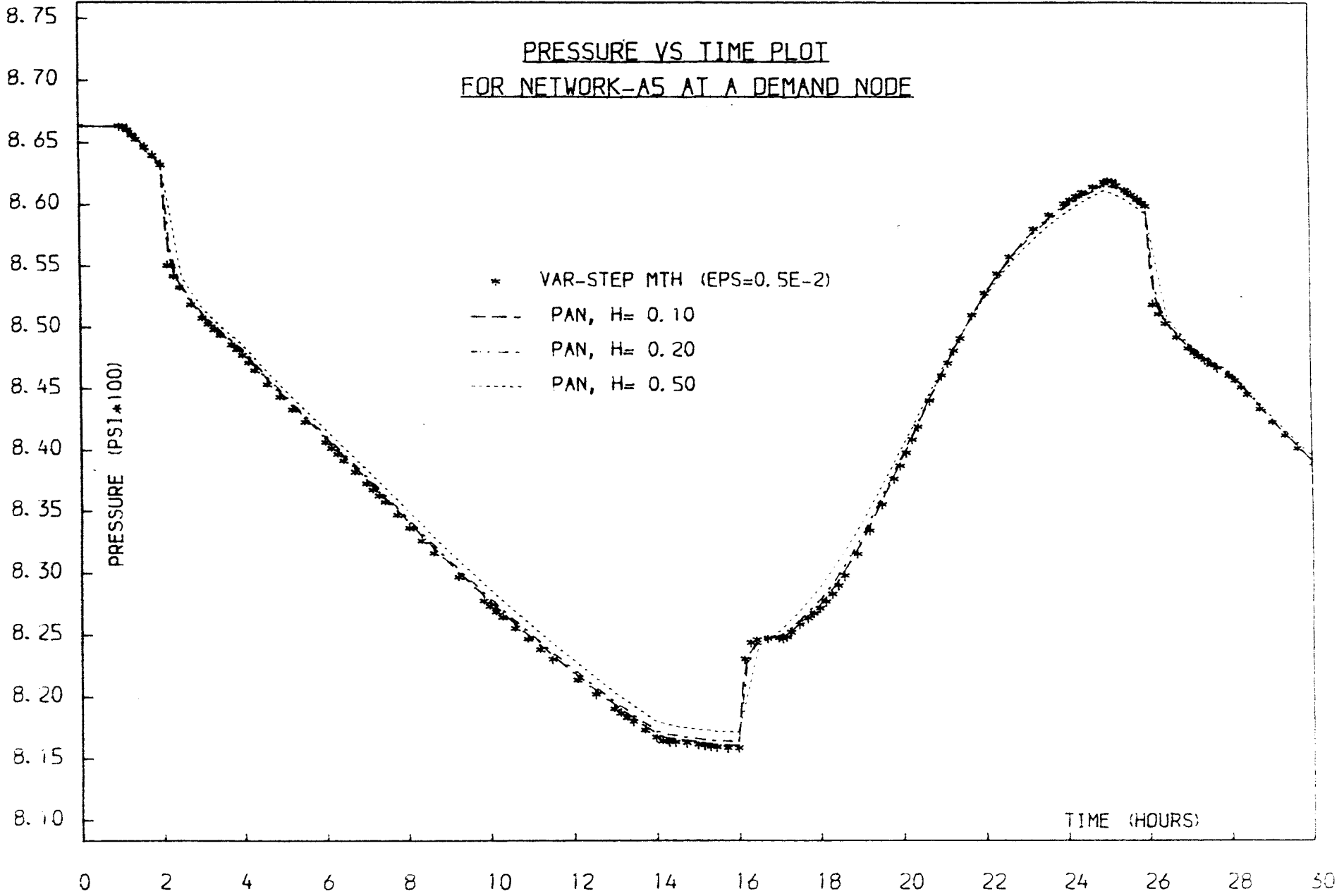


Fig. (5.5)



Fig. (5.6)

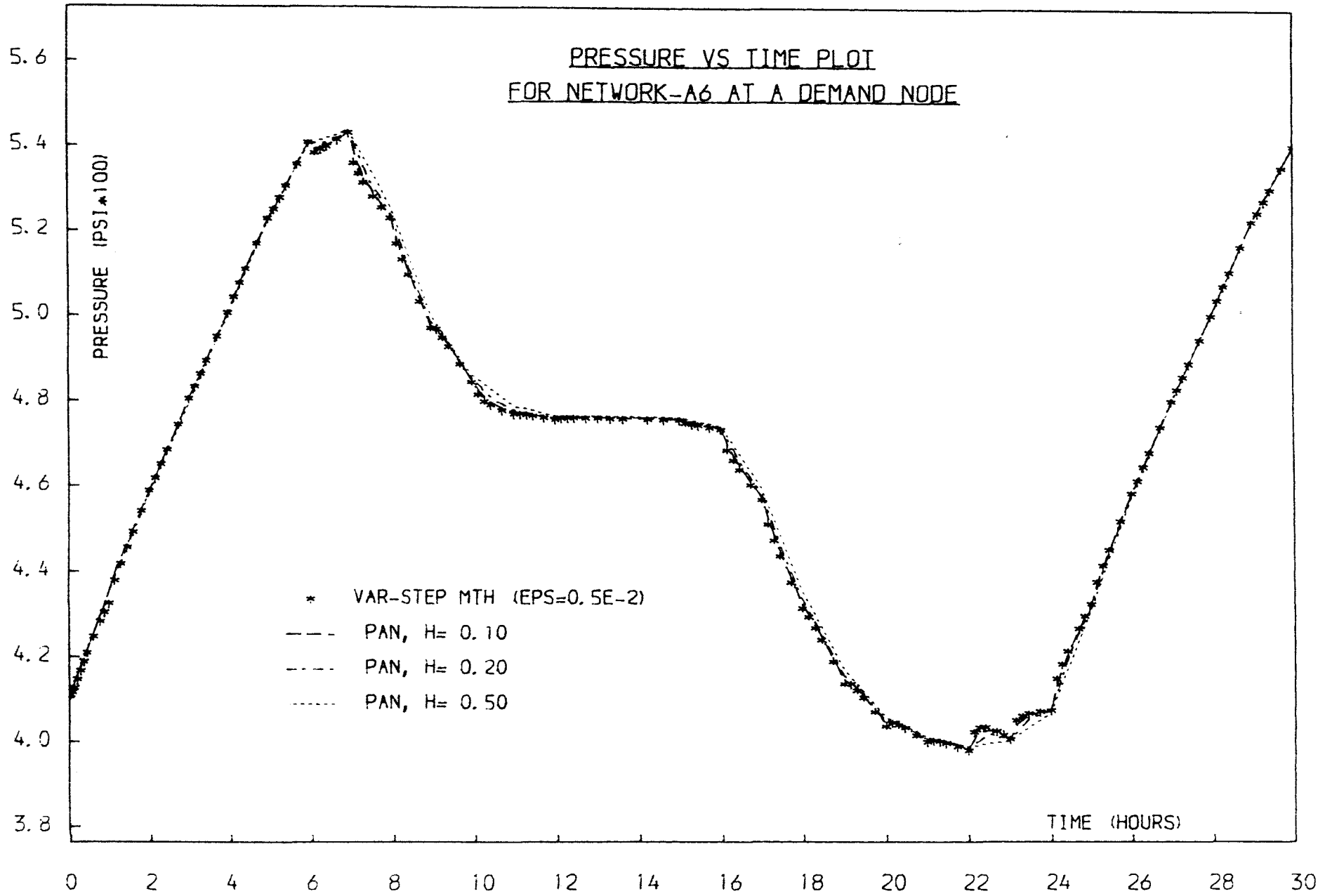
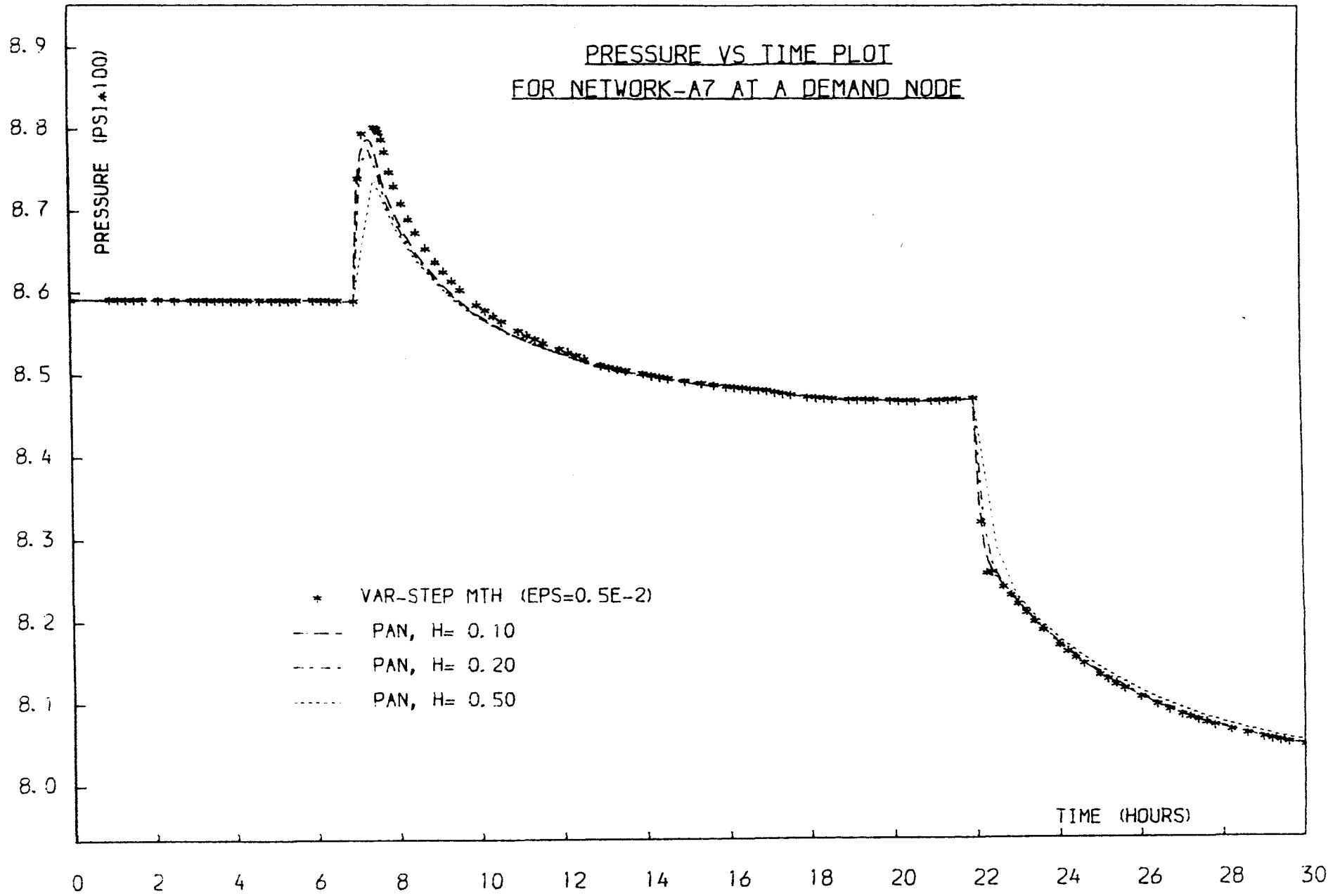


Fig. (5.7)



therefore shows that more stringent accuracy tolerance is not necessary for the gas transmission network problem.

To measure the efficiency of the variable-step integrators, a summary of the statistics required to compute the solution using the accuracy tolerance of  $0.5 \text{ E-}2$  is given in Table 5.3. The statistics include four components for measuring the cost of solving the problem; they are used in ENRIGHT(1975) for comparing the numerical methods for stiff ODE's. These components are the number of steps taken to compute the solution, NSTP; the number of function evaluations, NFUN; the number of iteration matrix updatings and factorizations, NMAT and the amount of CPU-time used, CPUT. The test criteria discussed in ENRIGHT(1975) are used as the basis for the comparison.

From table 5.3, it can be seen that substantial saving in computing time can be achieved by using the variable-step integrators as compared with the Leeds/PAN program. In assessing the efficiency of the integrators, one must also remember that they return reliable solution to the required accuracy.

The table also shows that the integrators RDIRK and EDIRK are less efficient than integrators THETA and ROSEN on all three networks considered. As integrators RDIRK and EDIRK require to solve 2 and 3 systems of nonlinear algebraic equations respectively at each time step, considerably more computational effort are needed for these integrators compared with THETA and ROSEN at each stage of the solution; however, because of the severe restriction on the stepsize, a larger time step cannot be used for these integrators and hence a longer solution time is needed. This is reflected in table 5.3 by the large number of function evaluations required by these integrators compared with THETA and ROSEN. The above observation suggests that higher order methods (for example order 3 and above) are not suitable for this application.

The integrator ROSEN generally requires fewer number of iteration matrix updatings than integrator THETA on the networks considered, however, more function evaluations are required. Their performance are very similar in terms of

Network	No. of nodes	No. of machines	No. of demands	No. of unknown variables	Period of simulation (hours)
Network A5: the regional network	57	15	40	72	0 - 30
Network A6: the north-west super grid	103	13	100	116	0 - 30
Network A7: the national grid (smaller version)	158	20	106	178	0 - 30

Table 5.2 - Details of the test networks

METHOD	Network A5				Network A6				Network A7			
	NSTP/NFUN/NMAT/CPUT				NSTP/NFUN/NMAT/CPUT				NSTP/NFUN/NMAT/CPUT			
THETA	119 / 236 / 48 / 7.5				136 / 272 / 57 / 11.4				124 / 276 / 63 / 27.0			
ROSEN	104 / 345 / 42 / 7.4				147 / 432 / 37 / 12.5				122 / 409 / 44 / 24.9			
RDIRK	102 / 565 / 42 / 9.1				140 / 673 / 36 / 15.5				120 / 660 / 48 / 30.9			
EDIRK	104 / 650 / 39 / 10.4				128 / 635 / 40 / 16.9				114 / 725 / 54 / 36.3			
Leeds/ PAN*	291 / 291 / 291 / 24.6				291 / 291 / 291 / 23.4				291 / 291 / 291 / 74.1			

where NSTP - number of steps attempted;

NFUN - number of function evaluations;

NMAT - number of iteration matrix updatings (and factorizations) and

CPUT - CPU-time used in seconds on the Amdahl V7 computer at the University of Leeds.

\* The performance profile of Leeds/PAN is obtained by using the stepsize which gives a solution of comparable accuracy to those computed by the variable-step integrators (see figs. (5.5)-(5.7)).

Table 5.3 - The Performance Profile of the Integrators

CPU-time with integrator ROSEN slightly more efficient than integrator THETA on the National Transmission Network (A7). We can therefore conclude that the theta method and the Rosenbrock-type method appear to be equally efficient and reliable in solving the gas transmission network problem.

Since the basic assumption in this work is that the gas transmission network model is parabolic in nature, it seems sensible to select a numerical method which reflects this. Because the stability property of the Rosenbrock-type method is more appropriate than those of the theta method for a parabolic problem, the Rosenbrock-type method can be recommended for general use.

For specific application on the calculation of line-pack in the network, however, the results in section 5.2 indicate that integrator RDIRK is more suitable.

## CHAPTER 6

- 6 THE CONFLICTING CONSTRAINTS PROBLEM
- 6.1 Introduction
- 6.2 The Gas Flow Model
- 6.3 The Dual Extremum Principles
- 6.4 A Variational Inequality Model
- 6.5 The Implementation
  - 6.5.1 The Scaling of the Constraint Matrix
- 6.6 Testing and Results
- 6.7 Further Enhancements

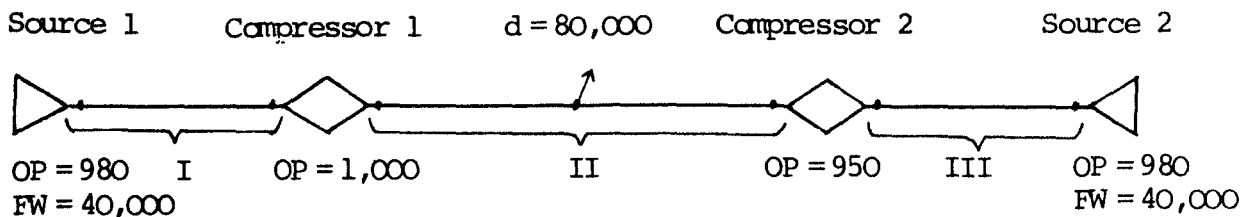
## 6.1 Introduction

The simulation of a gas transmission network involves the solution of a large system of nonlinear differential equations subject to a set of inequality constraints. The constraints model the operating limits of the machines and examples of these constraints can be found in appendix B.

As it is very difficult to deal with inequality constraints, a simple algorithm based on constraint modelling is used in PAN to model the machines in the network. Essentially, the algorithm replaces the set of inequality constraints with a smaller set of algebraic equations by assuming that for each machine, one of the constraints (referred to as the operating constraint) is actually equal to its extreme value. This gives rise to a set of differential/algebraic equations which can be solved efficiently using a variable-step integrator. It then checks for the rest of the machine constraints; if any is violated, then the corresponding operating constraint of the machine is changed to the violated one. A new solution is found and the process is repeated until there are no violations.

This algorithm is very easy to implement and is employed in many network analysis programs, for example, PAN, GANESI and WYLIE(1974). However, there are two fundamental drawbacks to this algorithm. First of all, it only tackles the problem locally by replacing the operating constraint of the machine by the violated one, without taking into account the effect of this on the overall solution of the problem. Secondly, it requires the machines to operate at one of its constraint values at all times and thus severely limits the feasible region of the solution. Because of these difficulties, the algorithm is not very robust and for certain networks, it fails to obtain a feasible solution. The problem is referred to as the "conflicting constraints problem". This problem occurs more often during the steady state analysis than the time-dependent solution, because the former has a further restriction on the solution that the continuity of the flow at the nodes must be satisfied exactly.

To illustrate the conflicting constraints problem, consider a simple test network as shown in fig. (6.1.1). This network consists of two sources, each feeding through a compressor, supplying to a common demand point. The network is symmetrical about the demand node.



Note - see appendix A for the meanings of the symbols used.

Fig.(6.1.1) - A Test Network with Conflicting Constraints Problem

By deleting the machines, the network can be separated into three components, I, II and III as shown in fig.(6.1.1). For steady state analysis, the pressure must be defined at at least one node in each network component so that the problem is well defined. This means that the two sources and at least one compressor must operate on their maximum outlet pressure constraints. Because of the difference in the compressor outlet pressure constraints, it can be seen that the gas flow from the left of the demand node will be greater than the flow from the right. Thus, in order to meet the demand, source 1 would have to violate its flow constraint. As the source cannot be switched to operate on flow constraint for this problem, the simple algorithm would break down and fail to find a feasible solution.

For this simple network, it can be easily deduced by symmetry that a feasible solution can be obtained by lowering the outlet pressure constraint of compressor 1 to 950 psi. However, for more complicated networks with a large number of machines, it is not normally possible to adjust the constraints in this way. In general, a mathematical programming approach is needed to decide which machine is to operate within its constraints, rather than on one of them.



British Gas have recently developed an optimization program based on linear programming (PRATT(1982)) which can be used to overcome the above problem. This chapter considers an alternative formulation of the gas flow problem into a variational inequality model for resolving the conflicting constraints problem; a rigorous mathematical theory based on the dual extremum principles (NOBLE(1972)) is used in the formulation.

## 6.2 The Gas Flow Model

In order to facilitate later discussions and to define the notations used throughout this chapter, the basic gas flow equations are summarised here. The derivation of these equations can be found in appendix B.

The gas flow and pressure along a pipe is modelled by the momentum and continuity equations in the standard manner and the resulting partial differential equations are discretised in the spatial variable using finite differences or the finite element method. This gives rise to a set of stiff ordinary differential equations (ODE's) of the form given in eqn.(B.4.4) which can be solved using a stiff integration method. The nonlinear equations that result from numerically solving the ODE's are solved by the modified-Newton method which in turn gives rise to two sets of algebraic equations of the form (see section B.5).

$$Z_{11}\underline{\Omega} + Z_{12}\underline{\Phi} + K_I \underline{Q} = \underline{b}_1 \quad (6.2.1)$$

$$Z_{21}\underline{\Omega} + Z_{22}\underline{\Phi} - \underline{Q} = \underline{b}_2 \quad (6.2.2)$$

where  $\underline{\Phi}$  is a vector containing the unknown pressures at the machine nodes,  $\underline{\Omega}$  denotes the set of pressures at the other network nodes and  $\underline{Q}$  is a vector containing the unknown machine flows. The vectors  $\underline{b}_1$  and  $\underline{b}_2$  are the known right-hand side quantities which include the demands from the network. The matrices  $Z_{ij}$ 's denote the connections within the network; the matrix formed by the  $Z_{ij}$ 's

$$Z_M = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}$$

is symmetric and positive-definite. The matrix  $K_I$  is the flow incidence matrix of the machine inlet nodes, and is defined as

$$K_{I_{\ell k}} = 1 \text{ if } \ell \text{ is the inlet node of machine } k \quad (6.2.3)$$

and  $\quad = 0$  otherwise.

Similar sets of equations can also be obtained for the steady state analysis.

Eqn.(6.2.2) relates to the machine outlet nodes and eqn.(6.2.1) corresponds to the other network nodes. Equations (6.2.1) and (6.2.2) are essentially the same as those given in eqn.(3.2.1) except that a different partitioning of the unknown pressures at the nodes is used. This is to facilitate the derivation and implementation of the optimization model described below, where the set of machine outlet pressures,  $\underline{\phi}$ , will be used as the controllable variables (the set of variables that can be adjusted during the optimization in order to obtain an optimal solution) in the optimization process.

The solution of eqns.(6.2.1) and (6.2.2) must also satisfy the operating constraints for the machines in the network. These machine constraints are modelled using a set of inequalities which has the general form (see fig. (6.2.1)),

$$\Psi_i(\Omega_k, \phi_k, Q_k) \leq \sigma_i \quad (6.2.4)$$

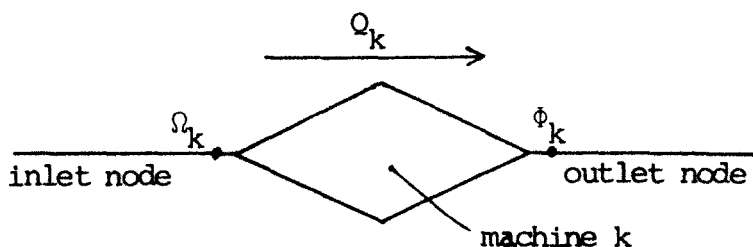


Fig.(6.2.1) - A Two-noded Machine

where  $\Psi_i$  is usually a nonlinear function of the machine inlet pressure  $\Omega_k$ , the machine outlet pressure  $\phi_k$  and the machine flow  $Q_k$ ; and  $\sigma_i$  is a known right-hand side quantity. The exact form of  $\Psi_i$  and  $\sigma_i$  depends on the type of machine constraint involved. Examples of the inequality constraints can be found in appendix B.

A complete solution is therefore needed which satisfies eqns.(6.2.1) and (6.2.2) together with the constraints of the form given by eqn.(6.2.4). We see that the simulation of the gas transmission network is a complex nonlinear mathematical programming problem.

### 6.3 The Dual Extremum Principles

This section outlines the important theoretical results of the dual extremum principles as given in SEWELL(1973). Further details can be found in that paper and in NOBLE(1972).

Consider two real inner product spaces  $E, F$  with elements denoted by  $x, u$  and inner products  $(,)$ ,  $\langle, \rangle$  respectively. Consider a functional  $H[x,u]$  defined in the product space  $EXF$ , generating the problem

$$\frac{\partial H}{\partial x} = 0 \quad (\alpha) \tag{6.3.1}$$

$$\frac{\partial H}{\partial u} = 0 \quad (\beta)$$

If the functional  $H$  is a saddle function concave in  $x$  and convex in  $u$ , then the dual extremum principles state that:

- i) any solution of the whole problem (6.3.1) will minimise the quantity

$$J(\alpha) = H - (x, \frac{\partial H}{\partial x}) \tag{6.3.2}$$

among all the solutions of the underdetermined subproblem (6.3.1 $\alpha$ );

- ii) any solution of the whole problem (6.3.1) will maximise the quantity

$$K(\beta) = H - \left\langle u, \frac{\partial H}{\partial u} \right\rangle \quad (6.3.3)$$

among all the solutions of the underdetermined subproblem (6.3.1 $\beta$ );

- iii) the minimum value of  $J$  and the maximum value of  $K$  which are provided by any solution of the whole problem (6.3.1) are the same and equal to the solution value of  $H$ ;
- iv) the solution value of  $u$  is unique when the convexity of  $H$  with respect to  $u$  is strict; the same is true for  $x$  if the concavity of  $H$  with respect to  $x$  is strict.

Similar properties i)-iv) hold when eqn.(6.3.1 $\alpha$ ) is replaced by three conditions

$$\left\{ \begin{array}{l} \frac{\partial H}{\partial x} \leq 0 \quad (\alpha) \\ x \geq 0 \quad (\beta) \\ (x, \frac{\partial H}{\partial x}) = 0 \end{array} \right. \quad (6.3.4)$$

with eqn.(6.3.1 $\beta$ ) retained; and also when eqn.(6.3.1 $\alpha$ ) is retained but eqn.(6.3.1 $\beta$ ) is replaced by

$$\left\{ \begin{array}{l} \frac{\partial H}{\partial u} \geq 0 \quad (\beta) \\ u \geq 0 \quad (\alpha) \\ \langle u, \frac{\partial H}{\partial u} \rangle = 0 \end{array} \right. \quad (6.3.5)$$

and again when both of eqns.(6.3.1) are replaced by eqns. (6.3.4) and (6.3.5). A complete proof of the above can be found in NOBLE(1972).

In the above example, we have considered two variables  $x$  and  $u$ . The idea generalises to  $n$  variables providing that

they can be divided into two groups of variables for which the given functional  $H$  is jointly convex with respect to one group and jointly concave with respect to the other. To illustrate this, consider the addition of a third real inner product space  $G$  with an element denoted by  $\lambda$  and inner product  $\{, \}$ . Consider a given functional  $H[x, u, \lambda]$  defined in  $ExFxG$  generating the problem given in eqns.(6.3.1) and an additional equation

$$\frac{\partial H}{\partial \lambda} = 0 \quad (6.3.6)$$

If  $H$  is concave jointly in  $x$  and  $\lambda$ , and convex in  $u$ , then the appropriate grouping of spaces should be  $ExG$  and  $F$ . It follows that the new equation (6.3.6) should be labelled  $\alpha$ ; and the associated extremum principles maximise  $K(\beta)$  given by eqn.(6.3.3) as it stands, and minimise the  $J(\alpha)$  obtained by adding the term  $-\{\lambda, \frac{\partial H}{\partial \lambda}\}$  to eqn.(6.3.2).

The same dual extremum principles also apply in the case when some or all of the eqns.(6.3.1) and (6.3.6) are replaced by inequalities of the form suggested by eqns.(6.3.4) and (6.3.5). Lastly, there is no reason why some of the conditions labelled as  $(\alpha)$  cannot be imposed in applying the  $K(\beta)$  principle (see eqn.(6.3.3)), and vice versa, if it is advantageous (p.147, SEWELL(1973)).

#### 6.4 The Variational Inequality Model

This section describes how the gas flow model can be cast into a variational inequality framework using the dual extremum principles described above. The two algebraic equations (6.2.1) and (6.2.2), and the inequality machine constraints are used in the formulation. For simplicity, only the linear constraints are considered; additional constraints can be incorporated later into the model. The sets of linear constraints considered in this section consist of

a) the maximum flow constraints

$$\underline{Q} \leq \underline{Q}_{\max} \quad (6.4.1)$$

where  $\underline{Q}_{\max}$  denotes the set of maximum flows through the machines;

b) the maximum machine outlet pressure constraints

$$\underline{\phi} \leq \underline{\phi}_{\max} \quad (6.4.2)$$

where  $\underline{\phi}_{\max}$  gives the set of maximum machine outlet pressures, and

c) the minimum machine inlet pressure constraints

$$K_{\underline{I}}^T \underline{\Omega} \geq K_{\underline{I}}^T \underline{\Omega}_{\min} \quad (6.4.3)$$

where  $\underline{\Omega}_{\min}$  denotes the set of minimum allowable pressures at the other network nodes;  $K_{\underline{I}}^T \underline{\Omega}$  gives the set of machine inlet pressures with  $K_{\underline{I}}$  being defined in eqn.(6.2.3).

We can define a new variable

$$\hat{\underline{Q}} = \underline{Q}_{\max} - \underline{Q} \quad (6.4.4)$$

and consider a functional  $H[\underline{\Omega}, \underline{\phi}, \hat{\underline{Q}}]$  given by

$$\begin{aligned} H = & \frac{1}{2} \underline{\Omega}^T Z_{11} \underline{\Omega} + \underline{\Omega}^T Z_{12} \underline{\phi} + \frac{1}{2} \underline{\phi}^T Z_{22} \underline{\phi} - (\underline{b}_1 - K_{\underline{I}} \underline{Q}_{\max})^T \underline{\Omega} \\ & - (\underline{b}_2 + \underline{Q}_{\max})^T \underline{\phi} - (\underline{\Omega} - \underline{\Omega}_{\min})^T K_{\underline{I}} \hat{\underline{Q}} + (\underline{\phi} - \underline{\phi}_{\max})^T \hat{\underline{Q}} \end{aligned} \quad (6.4.5)$$

It can be easily verified that the functional  $H$  is strictly convex jointly in  $\underline{\Omega}$  and  $\underline{\phi}$  because  $Z_{11}$  and  $Z_{22}$  are both positive-definite, and weakly concave in  $\hat{\underline{Q}}$  since it is only linear in  $\hat{\underline{Q}}$ . From the definition of  $H$ , it follows that

$$\frac{\partial H}{\partial \underline{\Omega}} = 0 \quad (\beta) \quad (6.4.6)$$

$$\frac{\partial H}{\partial \underline{\phi}} = 0 \quad (\beta) \quad (6.4.7)$$

$$\frac{\partial H}{\partial \hat{\underline{Q}}} \leq 0 \quad (\alpha) \quad (6.4.8)$$

$$\hat{\underline{Q}} \geq 0 \quad (\beta) \quad (6.4.9)$$

$$\left. \begin{array}{l} \\ \\ \end{array} \right\} \text{and } (\hat{\underline{Q}}, \frac{\partial H}{\partial \hat{\underline{Q}}}) = 0 \quad (6.4.10)$$

where the  $(\alpha)$ ,  $(\beta)$  labellings are given as a result of the saddle property of the functional  $H$ . Eqns.(6.4.6), (6.4.7) and (6.4.9) are simply the algebraic equations (6.2.1), (6.2.2) and the maximum flow constraint (6.4.1) respectively. Eqn.(6.4.8) is of the form

$$\frac{\partial H}{\partial \underline{Q}} = (\underline{\phi} - \underline{\phi}_{\max}) - K_I^T (\underline{\Omega} - \underline{\Omega}_{\min}) \leq 0$$

which is equivalent to imposing both the maximum outlet pressure constraint (6.4.2) and the minimum inlet pressure constraint (6.4.3). Lastly, eqn.(6.4.10) is an orthogonality condition which will automatically be satisfied at the optimal solution; it takes the form

$$\hat{\underline{Q}}^T (\underline{\phi} - \underline{\phi}_{\max} - K_I^T (\underline{\Omega} - \underline{\Omega}_{\min})) = 0 \quad (6.4.11)$$

For sources with only an outlet node, this condition simply means that either the outlet pressure or the machine flow is equal to the maximum. For two-noded machines such as the compressors, however, it is a lot more complicated and difficult to satisfy. Numerical experience has revealed that it is not necessary to satisfy this condition exactly in order to obtain a solution, and that the condition only affects the efficiency of the optimization model. As  $Q_{\max}$  is usually not imposed for the machines other than the sources, its values can be adjusted during each stage of the optimization to make  $\hat{\underline{Q}} = 0$ . This will improve the efficiency of the optimization model as the orthogonality condition will be more closely satisfied at all times. Further details on how this can be done will be given in the next section.

From the dual extremum principles, any solution of eqns.(6.4.6) to (6.4.10) will minimise the quantity.

$$J(\alpha) = H - (\hat{\underline{Q}}, \frac{\partial H}{\partial \underline{Q}}) \quad (6.4.12)$$

subject to the  $\alpha$ -condition (6.4.8); and will maximise the quantity,

$$K(\beta) = H - \langle \underline{\Omega}, \frac{\partial H}{\partial \underline{\Omega}} \rangle - \{ \underline{\Phi}, \frac{\partial H}{\partial \underline{\Phi}} \} \quad (6.4.13)$$

subject to the  $\beta$ -conditions (6.4.6), (6.4.7) and (6.4.9).

Unfortunately, it is very difficult to implement the model described by eqn.(6.4.13), so in this thesis only the model given by eqn.(6.4.12) is used. By substituting for  $H$  and simplifying the resulting expression using eqns.(6.2.1) and (6.2.2), the model (6.4.12) becomes

MINIMIZE

$$J(\alpha) = -\frac{1}{2} \underline{b}_1^T \underline{\Omega} - \frac{1}{2} \underline{b}_2^T \underline{\Phi} + \frac{1}{2} (\underline{Q} - 2\underline{Q}_{\max})^T (\underline{\Phi} - K_I^T \underline{\Omega}) \quad (6.4.14)$$

subject to the  $\alpha$ -conditions

$$\underline{\Phi} \leq \underline{\Phi}_{\max} \quad (6.4.15a)$$

$$\text{and } K_I^T (\underline{\Omega} - \underline{\Omega}_{\min}) \geq 0 \quad (6.4.15b)$$

Other machine constraints such as the maximum compression ratio or maximum horsepower constraints can also be imposed together with eqn.(6.4.15) without affecting the nature of the problem.

## 6.5 The Implementation of the Model

The NAG library routine, EO4UAF, is used to solve the variational inequality model (6.4.14). The routine is based on a sequential augmented Lagrangian method and uses the quasi-Newton method to solve the minimization subproblems involved. Details on the use of this routine is given in chapter E04 of the NAG library manual (NAGFLIB(1982)) and the theoretical background can be found, for example, in GILL(1974).

Since the solution of the whole network can be determined uniquely from equations (6.2.1) and (6.2.2) by fixing only the outlet pressures of the machines,  $\underline{\Phi}$ , only this set is treated as the controllable variables (see the definition in section 6.2) in the optimization process. This simplifies the implementation and reduces the amount of computation at each



stage of the optimization. Initially, the  $\phi$ 's are set to their maximum outlet pressures. In addition, a set of values for the maximum machine flows,  $Q_{\max}$ , must also be supplied for evaluating the objective function (6.4.14). As the flows of the machines other than the sources are usually not constrained, it is necessary to select carefully a suitable set of  $Q_{\max}$  values as this will improve the performance of the optimization model. As discussed in section 6.4, this can be done by setting  $Q_{\max}$  to the current machine flow value for those machines where a realistic  $Q_{\max}$  value is not available. This ensures that the orthogonality condition (6.4.11) is more closely satisfied which would speed up the convergence of the optimization process.

An algorithm that can be used to compute a feasible solution is given by:

#### Stage 1 - Initialization

Set all machine outlet pressures to their maximum limits, i.e. set  $\underline{\phi} = \underline{\phi}_{\max}$

#### Stage 2 - Solution of Nonlinear Equations

Compute  $\underline{\Omega}$  and  $\underline{Q}$  using eqns.(6.2.1)-(6.2.2) based on the set of  $\underline{\phi}$  values determined. The modified-Newton method is used to solve the system of nonlinear equations.

#### Stage 3 - Feasibility Test

Check the feasibility of the solution obtained. If the solution is feasible, then stop; otherwise goto next stage.

#### Stage 4 - Setting Up

Set up and factorize the coefficient matrices  $Z_{11}$ ,  $Z_{12}$  and  $Z_{22}$ , of eqns.(6.2.1)-(6.2.2), for use in the NAG library routine EO4UAF. Set  $Q_{\max}$  to the current value of  $Q$  for those machines where the flow is not constrained; otherwise set  $Q_{\max}$  to the corresponding maximum flow constraint.

#### Stage 5 - Optimization

Carry out the optimization using the NAG library routine, EO4UAF, based on the model (6.4.14)-(6.4.15). Other nonlinear

constraints such as the compressor horsepower constraints, which are not used explicitly in the development of the model, are also included. The variables  $\underline{\Omega}$  and  $\underline{Q}$  are expressed in terms of  $\underline{\phi}$  via eqns.(6.2.1)-(6.2.2) using the coefficient matrices set up in the previous stage. At the end of the optimization, goto stage 2.

As it is well known that the performance of the optimization codes are very sensitive to the scaling of the variables, some initial scaling of the problem is essential. We shall follow the recommendation given in NAGFLIB(1982) and scale the controllable variables,  $\underline{\phi}$ , so that they are as close to 1 as possible, and divide the inequality constraints by their corresponding maximum limits to give a value of less than unity. However, even with this initial scaling, the program implemented is still very unreliable in solving networks with a large number of machines (for example, more than 5 machines). Further scaling is required; this was confirmed by the fact that during the solution of these networks, a large conditional number was recorded by the NAG library routine which shows that the problem is very ill-conditioned.

As very little is known about the proper scaling of the nonlinear constraints of the problem, it was decided to linearise all nonlinear constraints (as is done in PAN) into the general form

$$m_1 \Omega_k + m_2 \phi_k + m_3 Q_k \leq \sigma_k \quad (6.5.1)$$

where  $m_1$ ,  $m_2$ ,  $m_3$  and  $\sigma_k$  are linearised coefficients depending on the type of constraints involved. By expressing  $\underline{\Omega}$  and  $\underline{Q}$  in terms of  $\underline{\phi}$ , the set of linearised constraints can be written as

$$B \underline{\phi} \leq \underline{\sigma} \quad (6.5.2)$$

where B is a linear, rectangular matrix which is referred to as the constraint matrix of the problem. A wide range of techniques available for scaling linear programming problems can be employed to scale the constraint matrix B.

### 6.5.1 The Scaling of the Constraint Matrix

A survey of various methods that can be used for scaling linear problems is given in TOMLIN(1975). Briefly, there are two classes of methods available; the first is the simple empirical method which is very easy to implement and has been found to work well in practice; the second is the 'optimal' method which involves solving a linear programming problem of at least the same dimension in order to determine the best scaling factors that can be used. An example of the second approach can be found in CURTIS(1972).

As the second approach is very difficult to implement and is also beyond the scope of this study, only the simple empirical methods are investigated. They are given by:

- i) Equilibration - each row of matrix B is scaled to make the largest element of order unity, followed by a similar scaling on the columns.
- ii) Geometric Mean - for each row of matrix B, we compute  $(\max_j |b_{ij}| \cdot \min_j |b_{ij}|)^{\frac{1}{2}}$  and divide the row by this number. This is followed by a similar column scaling.
- iii) Arithmetic Mean - each row is divided by the arithmetic mean of the elements in that row. Again this is followed by a similar treatment on the columns.

Either of the later techniques may be followed by equilibration.

These techniques will be considered in the next section for scaling the linearised constraint equations.

## 6.6 Testing and Results

The algorithm outlined in the previous section is used to solve the variational inequality model (6.4.14) with linearised constraints. Four scaling methods described in the previous section are considered; they are:

- i) the Geometric Mean (GM);
- ii) the Arithmetic Mean (AM);
- iii) the Geometric Mean followed by equilibration (GM & E) and
- iv) the Arithmetic Mean followed by equilibration (AM & E).

The method of Equilibration alone is not considered as it has not been found to perform well in practice; a similar conclusion was also found in TOMLIN(1975).

As a preliminary investigation, the model with and without scaling is applied to the simple conflicting constraints network as described in section 6.1 (see fig.(6.1.1)). In all cases, the model was able to return a feasible solution which confirms that the model formulated is reliable.

To further test the efficiency and reliability of the variational inequality model, two larger conflicting constraints networks of greater complexity are considered. The first network, given in fig.(A.8), consists of 12 pipes, 2 compressors, 2 sources and a valve. The second network is part of the British Gas National Transmission Network as shown in fig.(A.9); it consists of 29 pipes, 4 compressors and 4 sources. Both networks contain a large number of nonlinear inequality constraints. Only the steady state analysis is carried out for these networks.

The results of the investigation are summarised in table (6.6.1) for the model using different scaling methods. The table gives the amount of CPU-time and the number of evaluations of the objective function (6.4.14) required to compute a feasible solution (the first feasible solution encountered) for the networks considered. To illustrate the effectiveness of the scaling methods employed, the result obtained from the model without any overall scaling (except those recommended by NAGFLIB(1982)) is also included.

Scaling \ Network	Network A8		Network A9	
	NFNO	CPUT	NFNO	CPUT
Unscaled	1735	3.81	<fail>	
GM	907	1.62	1602	4.54
AM	1570	3.70	689	2.35
GM & E	1294	2.43	411	1.56
AM & E	1148	2.23	452	1.72

where CPUT - the amount of CPU-time used in seconds on the Amdahl-V7 computer and

NFNO - the number of evaluations of the objective function.

Table (6.6.1) - The Performance Profile of the Variational Inequality Model

From table (6.6.1), it can be seen that the scaling of the constraint matrix, using one of the empirical methods described above, generally improves the efficiency and reliability of the model implemented. Furthermore, the variational inequality model with scaled linearised constraints is reliable in solving the gas transmission network with the conflicting constraints problem.

### 6.7 Further Enhancements

As the aim of this chapter is simply to investigate the reliability of the model in resolving the conflicting constraints problem, no attempt has been made to devise an efficient implementation of the model. Further work is therefore needed in the following areas:

- a) to develop an efficient algorithm for carrying out the optimization;
- b) to investigate the effects of the 'optimal' scaling methods on the overall performance of the model with linearised constraints and
- c) to treat the nonlinear constraints in their usual form instead of linearising them, and explore the suitable methods that can be used for scaling the nonlinear problems. An example of such method is described in LASDON (1981).

Further tests on the model are also needed using the large scale transmission network.

## 7. Conclusions

The project described in this thesis was concerned with the development and evaluation of computational techniques to the simulation of large scale British Gas transmission network problems. One of the difficulties in solving the problem of this kind is to devise a suitable set of test problems which can be used to model the behaviour of the actual problem involved; and in this respect, I am grateful to British Gas, and J.R. Mallinson in particular, for supplying the test networks.

The general simulation of a gas transmission network involves the solution of a large system of stiff differential/algebraic equations (DAE) containing frequent severe disturbances. The research topics considered in this thesis can be divided into three main areas - the solution of the DAE system, the solution of linear equations, and the formulation and solution of a variational inequality model. Because of the diverse nature of this thesis, the different areas are discussed separately below.

### A. The Solution of DAE System using a Variable-step Integrator

The solution of a DAE system using a variable-step integrator is still a relatively new area of research in which very little theoretical results is available. The most recent and comprehensive work in this area was by PETZOLD(1981). From the analysis given in PETZOLD and the extensive numerical experimentation carried out during the project, it was found to be necessary to extensively modify the strategies and local error estimate employed in the usual variable-step ODE codes. In particular, the new error estimate of the form (4.3.2) was employed and new sets of strategies (see section 4.4.2) were developed to implement the numerical methods. Furthermore, because the system contains frequent severe disturbances, a restart strategy as described in section 4.4.2 was also developed; the strategy is independent of the numerical methods that is

used in the integration and relies only on the assumption that the DAE system arises from the discretization of a system of parabolic PDE's. The integrators developed based on the above strategies, using the numerical methods outlined in appendix D, were found to be both robust and efficient in solving the gas transmission network problem.

As the techniques employed for handling the DAE system is mainly empirical, further research is still needed to develop a theoretical framework to support these techniques. Further enhancements as suggested in section 4.5 for solving the DAE system can also be attempted.

#### B. The Solution of Linear Equations

The dynamic simulation of gas transmission network requires the solution of large systems of linear equations at each time step. The system is very sparse and a large part of it is symmetric and positive-definite. In order to take advantage of its underlying structure, three block matrix partitioning schemes based on the idea given in GEORGE(1974) were developed. The details of these schemes together with the comparison results are given in chapter 3. The results indicated that the best solution scheme described in chapter 3 is very effective, as it reduces considerably both the storage requirement and the number of arithmetic operations required.

#### C. The Variational Inequality Model

A variational inequality model was developed in chapter 6 for resolving the conflicting constraints problem. The model was formulated using a rigorous mathematical theory based on the dual extremum principles outlined in NOBLE(1972). As an initial investigation, the model with scaled linearised constraints was implemented using a NAG library routine, EO4UAF. The program implementing the model had been found to be reliable on three test networks considered. Further work on the model in the directions as suggested in section 6.7 can be carried out.

The above research was carried out in collaboration with a research team in British Gas Corporation. British Gas now have plans to incorporate the variable-step control into the new version of their network analysis program; the strategies described in chapter 4 will be used in the implementation.



REFERENCES

- ALEXANDER 1977  
Alexander, R. - "Diagonally Implicit Runge-Kutta Methods for Stiff ODE's", SIAM J. Numer. Anal., 14, 1006-1021.
- AZAR 1975  
Azar, A.R., Nichols, K.G. - "Sparse Matrix Algorithm for Transient Analysis of Nonlinear Electrical Networks", Proc. IEEE, 122 (8), 791-794.
- BARRY 1978  
Barry, D.E., Pottle, C., Wirgau, K.A. - "Technology Assessment Study of near term Computer Capabilities and their Impact on Power Flow and Stability Simulation Programs", EPRI EL-946.
- BERESFORD 1980  
Beresford, P.J. - "Sparse Matrix Techniques in Engineering Practice", in DUFF(1980a), 175-190.
- BERZINS 1981  
Berzins, M. - "Chebyshev Polynomial Methods for Parabolic Equations", Ph.D. Thesis, University of Leeds, England.
- BICKART 1977  
Bickart, T.A. - "An Efficient Solution Process for Implicit Runge-Kutta Methods", SIAM J. Numer. Anal., 14, 1022-1027.
- BROWN 1973  
Brown, R.L., Gear, C.W. - "DOCUMENTATION FOR DFASUB--A Program for the Solution of Simultaneous Implicit Differential and Nonlinear Equations", UIUCDCS-R-73-575, Uni. of Illinois at Urbana-Champaign.
- BUI 1979  
Bui, T.D. - "Some A-stable and L-stable Methods for the Numerical Integration of Stiff ODE's", J. ACM, 26, 483-493.
- BUS 1975  
Bus, J.C.P., Dekker, T.J. - "Two Efficient Algorithms with Guaranteed Convergence for Finding a Zero of a Function", ACM TOMS, 1 (4), 330-345.
- BUTCHER 1964  
Butcher, J.C. - "Implicit Runge-Kutta Processes", Maths. Comp., 18, 50-64.
- BUTCHER 1975  
Butcher, J.C. - "A Stability Property of Implicit Runge-Kutta Methods", BIT, 15, 358-361.

BUTCHER 1976

Butcher, J.C. - "On the Implementation of Implicit Runge-Kutta Methods", BIT, 16, 237-240.

BUTCHER 1979

Butcher, J.C., Burrage, K., Chipman, F., - "An Implementation of Singly-Implicit Runge-Kutta Methods", Research Report 149, Uni. of Auckland.

BUTCHER 1981

Butcher, J.C. - "Stability Properties for a General Class of Methods for ODE's", SIAM J. Numer. Anal., 18 (1), 37-44.

CALAHAN 1968

Calahan, D.A. - "A Stable, Accurate Method of Numerical Integration for Nonlinear Circuits", Proc. IEEE, 56, 744.

CARVER 1977

Carver, M.B. - "Efficient Integration over Discontinuities in ODE's", in 'Numerical Methods for Differential Equations and Simulations', by Bennett, A.W., Vichnevetsky, R. (ed.), North-Holland Publishing Co., Amsterdam.

CARVER 1978

Carver, M.B., MacEwen, S.R. - "Numerical Analysis of a System described by Implicitly-defined ODE's containing numerous Discontinuities", Appl. Math. Modelling, 2, 280-286.

CASH 1976

Cash, J.R. - "Semi-implicit Runge-Kutta Procedures with Error Estimates for the Numerical Integration of Stiff Systems of ODE's", J. ACM, 25, 455-460.

CASH 1979

Cash, J.R. - "Diagonally Implicit Runge-Kutta Formulae with Error Estimates", J. IMA, 24, 293-301.

CASH 1982

Cash, J.R. - "A Survey of Runge-Kutta Methods for the Numerical Integration of Stiff Differential Systems", Paper presented at the International Conference on Stiff Computation' held at Park City, Utah.

CHUA 1982

Chua, T.S. - "Collection of Programs and Numerical Results", in the Departmental Library, Dept. of Computer Studies, Uni. of Leeds.

CROUZEIX 1975

Crouzeix, M. - "Sur l'approximation des équations différentielles opérationnelles Linéaires par des méthodes de Runge-Kutta", Thèse présentée à l'Université Paris VI, Paris.

COVINGTON 1979

Covington, N.T. - "Transient Models Permits Quick Leak Identification", Pipeline Ind., 51 (2), 71-73.

CURTIS 1972

Curtis, A.R., Reid, J.K. - "On the Automatic Scaling of Matrices for Gaussian Elimination", JIMA, 10, 118-124.

DAHLQUIST 1963

Dahlquist, G. - "A Special Stability Problem for Linear Multistep Methods", BIT, 3, 27-43.

DAHLQUIST 1975

Dahlquist, G. - "Error Analysis for a Class of Methods for Stiff Nonlinear Initial Value Problems", Proceedings of the Dundee Conf. on Numerical Analysis, Springer-Verlag, Berlin 506, 60-74.

DAHLQUIST 1982

Dahlquist, G. - "Some Comments on Stability and Error Analysis for Stiff Nonlinear Problems", Paper presented at the 'International Conference on Stiff Computation' held at Park City, Utah.

DEW 1978

Dew, P.M., West, M.R. - "A Package for Integrating Stiff Systems of ODE's based on Gear's Method", Uni. of Leeds, Dept. of Computer Studies, Report 111.

DEW 1981

Dew, P.M., Walsh, J.E. - "A set of Library Routines for Solving Parabolic Equations in One Space Variable", ACM TOMS, 7, 295-314.

DISTEFANO 1970

Distefano, G.P. - "PIPETRAN, version IV, A Digital Computer Program for the Simulation of Gas Pipeline Network Dynamics", Cat. no. L20000, American Gas Association Inc., New York.

DUFF 1976

Duff, I.S., Erisman, A.M., Reid, J.K. - "On George's Nested Dissection Method", SIAM J. Numer. Anal., 13, 686-695.

DUFF 1980a

Duff, I.S. - (ed.) "Sparse Matrices and Their Uses", Academic Press, London.

DUFF 1980b

Duff, I.S. - "A Sparse Future", in DUFF(1980a), 1-30.

EHLE 1969

Ehle, B.L. - "On Pade Approximations to the Exponential Function and A-stable Methods for the Numerical Solution of Initial Value Problems", Uni. of Waterloo, Dept. of Applied Analysis and Computer Science, Report CSRR 2010.

ELLISON 1981

Ellison, D. - "Efficient Automatic Integration of ODE's with Discontinuities", Math. Comp. Simul., 23 (12)

ENRIGHT 1975

Enright, W.H., Hull, T.E., Linberg, B. - "Comparing Numerical Methods for Stiff ODE's", BIT, 15, 10-48.

ERISMAN 1980

Erisman, A.M. - "Sparse Matrix Problems in Electric Power System Analysis", in DUFF(1980a), 31-56.

FINCHAM 1979

Fincham, A.E. and Goldwater, M.H. - "Simulation Models for Gas Transmission Networks", Trans. Inst. Measurement and Control, 1, 3-12.

FINCHAM 1980

Fincham, A.E. and Goldwater, M.H. - "Modelling of Gas Supply Systems", in 'Modelling of Dynamic Systems', Ed. by Nicholson, Peter Peregrinus Ltd., London.

FINCHAM 1982

Fincham, A.E. - "Private Communication", London Research Station, British Gas.

GASEMN 1981

"Gas Engineering and Management", 21, Nov/Dec 1981, 448-450.

GEAR 1969

Gear, C.W. - "The Automatic Integration of Stiff ODE's", in 'Information Processing', 68, ed. by Mornel, A.J.H., North-Holland Publishing Co., Amsterdam, 187-193.

GEAR 1971a

Gear, C.W. - "DIFSUB for the Solution of ODE's - Alg 407", Com. ACM, 14 (3), 185-190.

GEAR 1971b

Gear, C.W. - "Numerical Initial Value Problems in ODE's". Prentice-Hall, New Jersey.

GEAR 1971c

Gear, C.W. - "Simultaneous Numerical Solution of Differential-Algebraic Equations", IEEE Trans. on Circuit Theory, CT-18 (1), 89-95.

GEAR 1980

Gear, C.W. - "Runge-Kutta Starter for Multistep Methods", ACM TOMS, 6 (3), 585-603.

GEAR 1981

Gear, C.W., Hsu, H.H., Petzold, L. - "Differential/Algebraic Equations Revisited", Proc. Workshop Numerical Method for Solving Stiff Initial Value Problems, Germany.

GEORGE 1973

George, A. - "Nested Dissection of a Regular Finite Element Mesh", SIAM J. Numer. Anal., 10, 345-363.

GEORGE 1974

George, A. "On Block Elimination for Sparse Linear Systems", SIAM J. Numer. Anal., 11, 585-603.

## GEORGE 1980

George, A. - "Direct Solution of Sparse Positive Definite Systems: some Basic Ideas and Open Problems", in DUFF(1980a), 283-306.

## GEORGE 1981

George, A. and Liu, J.W. - "Computer Solution of Large Sparse Positive Definite System", Prentice-Hall, New Jersey.

## GIBBS 1976

Gibbs, N.E., Poole, W.G., Stockmeyer, P.K. - "An Algorithm for reducing Bandwidth and Profile of a Sparse Matrix", SIAM J. Numer. Anal., 13, 236-250.

## GILL 1974

Gill, P.E., Murray, W. - (ed.) "Numerical Methods for Constrained Minimization", Academic Press.

## GOACHER 1970

Goacher, P.S. - "Steady and Transient Analysis of Gas Flows in Networks", J. Inst. Gas Eng., 10, 242-264.

## GOLDWATER 1976

Goldwater, M.H., Rogers, K. and Turnbull, D.K. - "The PAN Network Analysis Program - Its Development and Use", Inst. Gas Eng., Communication 1009.

## GOODWIN 1982

Goodwin, N. - "Private Communication", London Research Station, British Gas.

## GOURLAY 1970

Gourlay, A.R. - "Hopscotch: a Fast Second-order Partial Differential Equation Solver", JIMA, 6, 357-390.

## HALIN 1976

Halin, H.J. - "Integration of ODE's containing Discontinuities", Proc. of 'Computer Simulation conf.', Washington D.C.

## HAY 1974

Hay, J.L., Crosbie, R.E., Chaplin, R.I. - "Integration Routine for Systems with Discontinuities", Comp. J., 17, 275.

## HEATH 1969

Heath, M.J. and Blunt, J.C. - "Dynamics Simulation Applied to the Design and Control of a Pipeline Network", J. Inst. Gas Eng., 9, 261-279.

## HINDMARSH 1973

Hindmarsh, A.C. - "GEARB: Numerical Solution of ODE's having Banded Jacobian", Report UCID-30059, Lawrence Livermore Lab., Livermore, California.

## HOPKINS 1976

Hopkins, T.R., Wait, R. - "A Comparison of Numerical Methods for the Solution of Quasilinear Partial Differential Equations", Comp. Methods in App. Mech. and Eng., 9, 181-190.

JENNINGS 1966

Jennings, A. - "A Compact Storage Scheme for the Solution of Symmetric Linear Simultaneous Equations", *Comp. J.*, 9, 281-285.

JACOBS 1980

Jacobs, D.A.H. - "The Exploitation of Sparsity by Iterative Methods", in DUFF(1980a), 191-222.

LAMBERT 1973

Lambert, J.D. - "Computational Methods in ODE's", Wiley, London.

LASDON 1981

Lasdon, L.S., Beck, P.O. - "Scaling Nonlinear Programs", *O.R. Letters*, 1(1), 6-9.

LEWIS 1980

Lewis, J.G., Poole, W.G. - "Ordering Algorithms applied to Sparse Matrices in Electric Power Problems", in 'Electric Power Problems: The Mathematical Challenge', ed. by Erisman, A.M., Neves, K.W., Dwarakamath, M.H., SIAM press.

LIPTON 1979

Lipton, R.J., Rose, D.J., Tarjan, R.E. - "Generalised Nested Dissection", *SIAM J. Numer. Anal.*, 16, 346-358.

LISTER 1960

Lister, M. - "The Numerical Solution of Hyperbolic Partial Differential Equations by the Method of Characteristics", in 'Mathematical Methods for Digital Computers', Ed. by Wilf, A. and Ralston, H.S., Wiley, New York.

NAGFLIB 1982

"NAG FORTRAN Library Manual", Mark 9, Vol. 3, Chapter E04.

NOBLE 1972

Noble, B., Sewel, M.J. - "On Dual Extremum Principles in Applied Mathematics", *J. Inst. Maths Applics.*, 9, 123-193.

NØRSETT 1974

Nørsett, S.P. - "Semi-explicit Runge-Kutta Methods", *Maths. and Comp. Report 6/74*, Uni. of Trondheim.

NØRSETT 1977

Nørsett, S.P., Wolfbrandt, A. - "Attainable Order of Rational Approximations to the Exponential Function with only Real Poles", *BIT*, 17, 200-208.

PAINE 1982

Paine, J. - "Private Communication", School of Maths., Uni. of Bristol, England.

PARTER 1961

Parter, S.V. - "The Use of Linear Graphs in Gaussian Elimination", *SIAM Rev.*, 3, 119-130.

PETZOLD 1981

Petzold, L. - "Differential/algebraic Equations are not ODE's", Sandia National Lab. - Livermore, Report San81-8668.

PRATT 1982

Pratt, K.F., Wilson, J.G. - "Optimization of the Operation of Gas Transmission Systems", to appear in the Trans. Inst. of Measurement and Control.

PROTHERO 1974

Prothero, A., Robinson, A. - "On the Stability and Accuracy of One-step Methods for Solving Stiff Systems of ODE's", Math. Comp., 28 (125), 145-162.

PROTHERO 1977

Prothero, A., Robinson, A. - "Global Error Estimates for Solution of Stiff Systems of ODE's", Paper presented at the Numerical Analysis Conf. at Dundee Uni.

RACHFORD 1974

Rachford, H.H. Jr and Dupont, T. - "A Fast Highly Accurate Means of Modelling Transient Flow in Gas Pipeline Systems by Variational methods". Soc. Pet. Eng. J., 14, 165-178.

ROSE 1972

Rose, D.J. - "A Graph-theoretic Study of the Numerical Solution of Sparse Positive Definite Systems of Linear Equations", in 'Graph Theory and Computing', Ed. by Read, R.C., Academic Press, New York.

ROSENBROCK 1963

Rosenbrock, H.H. - "Some General Implicit Processes for the Numerical Solution of Differential Equations", Comp. J., 5, 329-330.

SACK-DAVIS 1977

Sack-Davis, R. - "Error Estimates for Stiff Differential Equation Procedure", Math. Comp., 31 (140), 939-953.

SATO 1963

Sato, N., Tinney, W.F. - "Techniques for Exploiting the Sparsity of the Network Admittance Matrix", IEEE Trans., PAS-82, 944-949.

SCHEEL 1972

Scheel, L.F. - "Gas Machinery", Gulf Publishing Co., Houston, Texas.

SCHMIDT 1977

Schmidt, G. and Weimann, A. - "Instationäre Gasnetzberechnung mit dem Programming GANESI", GWF-Gas/Erdgas, 118, 53-57.

SCRATON 1981

Scraton, R.E. - "Some L-stable Methods for Stiff Differential Equations", Intern. J. Computer Maths., Section B, 9, 81-87.

## SENS 1970

Sens, M., Jouve, Ph. and Pelletier, R. - "Détection d'une Rupture Accidentelle de Conduite", Paper IGU/C 37-70 presented at the 11th International Gas Conference, Moscow. (English Translation: British Gas Internal Report LRS T448).

## SEWELL 1973

Sewell, M.J. - "The Governing Equations and Extremum Principles of Elasticity and Plasticity Generated from a Single Functional - Part I and II", J. Struct. Mech., Part I, 2(1), 1-32. Part II, 2(2), 135-158.

## SHERMAN 1975

Sherman, A.H. - "On the Efficient Solution of Sparse Systems of Linear and Nonlinear Equations", Report 46, Dept. of Computer Science, Yale Uni.

## SINCOVEC 1979

Sincovec, R.F., Dembart, B., Epton, M.A., Erisman, A.M., Manke, S.W., Yip, E.L. - "Solvability of Large Scale Descriptor Systems", Boeing Computer Service Company.

## STEIHAUG 1979

Steihaug, T., Wolfbrandt, A. - "An Attempt to avoid Exact Jacobian and Nonlinear Equations in the Solution of Stiff Differential Equations", Maths. Comp., 33, 521-534.

## STONER 1968

Stoner, M.A. - "Analysis and Control of Unsteady Flows in Natural Gas Piping Systems", Ph.D. Dissertation, The Uni. of Michigan, Am Arbor.

## STONER 1969

Stoner, M.A. - "Steady State Analysis of Gas Production, Transmission and Distribution Systems", Paper SPE 2554 presented at SPE-AIME 44th Annual Fall Meeting, Denver, Colorado.

## STREETER 1970

Streeter, V.L. and Wylie, E.B. - "Natural Gas Pipeline Transients", Soc. Pet. Eng. J., 10, 357-364.

## TAYLOR 1978

Taylor, B.A. - "The Flow in Pipelines Following Catastrophic Failure", British Gas Internal Report, LRS 338.

## TINNEY 1967

Tinney, W.F., Walker, J.W. - "Direct Solution of Sparse Network Equations by Optimally Ordered Triangular Factorization", Proc. IEEE, 55, 1801-1809.

## TOMLIN 1975

Tomlin, J.A. - "On Scaling Linear Programming Problems", Mathematical Programming Study, 4, 146-166.

## VARAH 1979

Varah, J.M. - "On the Efficient Implementation of Implicit Runge-Kutta Methods", Maths Comp, 33, 557-561.



WARD-SMITH 1971

Ward-Smith, A.J. - "Pressure Losses in Ducted Flows", Butterworths, London.

WEIMANN 1978

Weimann, A. - "Modellierung and Simulation der Dynamik von Gasverteilnetzen im Hinblick auf Gasnetzführung and Gasnetzüberwachung", Dr. Ing. Thesis, Munich Technical University. (English Translation: British Gas internal report LRS T435).

WILKINSON 1965

Wilkinson, J.H. - "The Algebraic Eigenvalue Problem", Clarendon Press, Oxford.

WYLIE 1967

Wylie, E.B., Streeter, V.L. - "Hydraulic Transients", McGraw-Hill, New York, Chapter 15.

WYLIE 1971

Wylie, E.B., Streeter, V.L. and Stoner, M.A. - "Network System Transient Calculations by Implicit Method", Soc. Pet. Eng. J., 11, 356-362.

WYLIE 1974

Wylie, E.B., Streeter, V.L. and Stoner, M.A. - "Unsteady-state Natural Gas Calculations in Complex Pipe Systems", Soc. Pet. Eng. J., 14, 35-43.

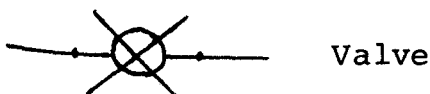
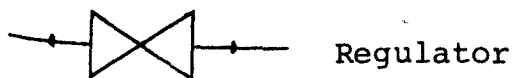
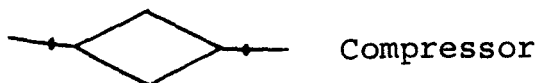
YOW 1972

Yow, W. - "Numerical Error in Natural Gas Transient Calculations", Trans. ASME, Series D, J. Basic Eng., 94, 422-428.

## APPENDIX A

THE TEST NETWORKSSymbols used in the network

$\overline{\hspace{2cm}}$  10/36 Pipe length(mile)/diameter(inch)



OP = maximum outlet pressure (psi - pound per square inch)

FW = maximum flow (mscfh - thousands of standard cubic feet per hour)

HP = maximum horsepower (hp - horsepower)

CR = maximum compression ratio

IP = minimum inlet pressure (psi)

d = demand (mscfh)

PROF = demand profile

- Note - 1) the demand profiles for the networks are shown in their respective diagrams containing the results and  
2) the units for the variables are as quoted above unless otherwise stated

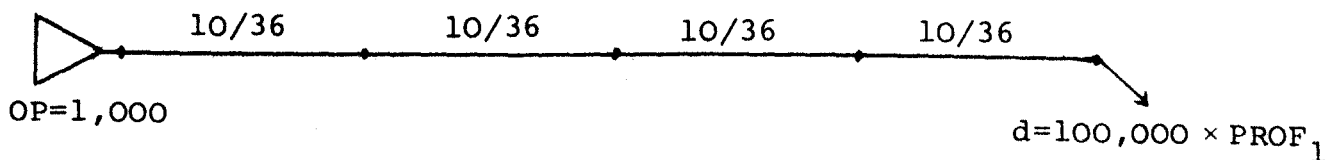
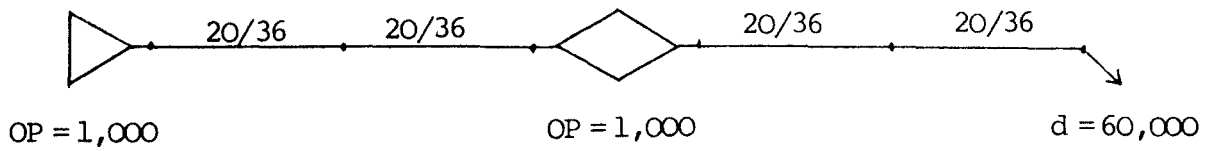
Network A1

Fig. (A.1)

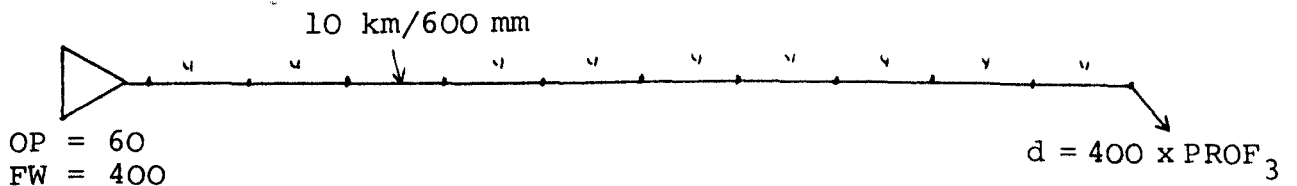
Network A2



The compressor is shut off after the first hour for the 4-hour period of simulation.

Fig. A.2

Network A3



where the unit of pressure (in OP) is in bar  
and the unit of flow (in FW and d) is in thousands of cubic metre per hour

Fig. A.3

Network A4

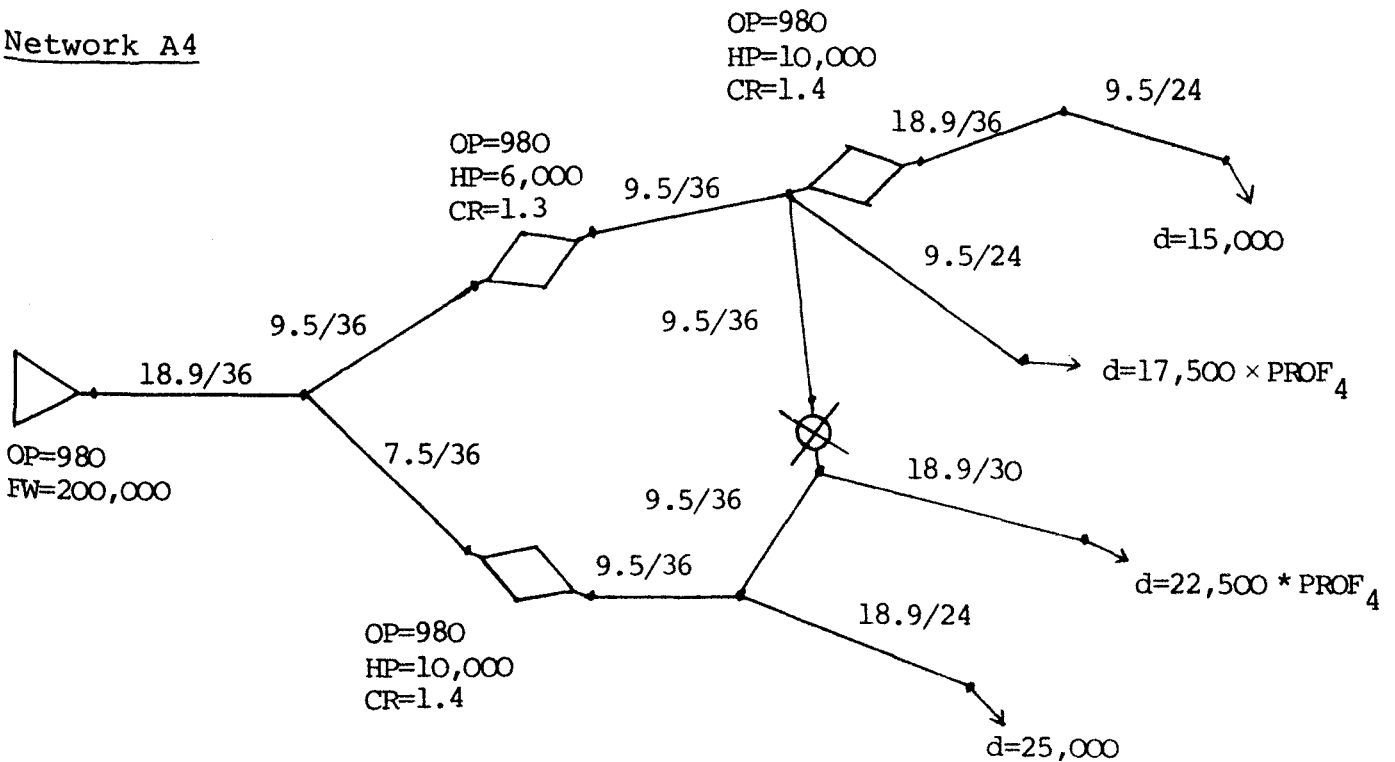


Fig. A.4

Network A5

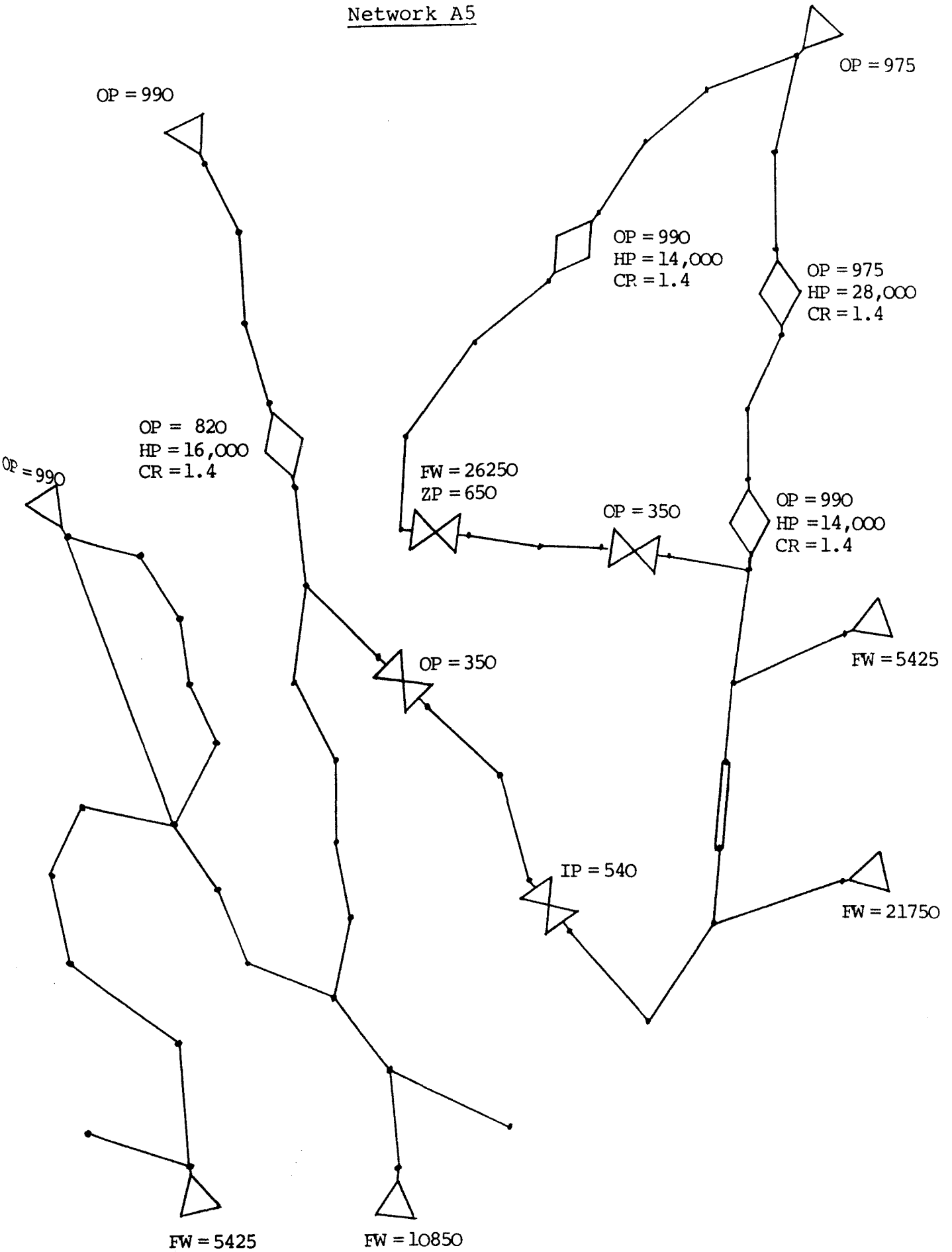





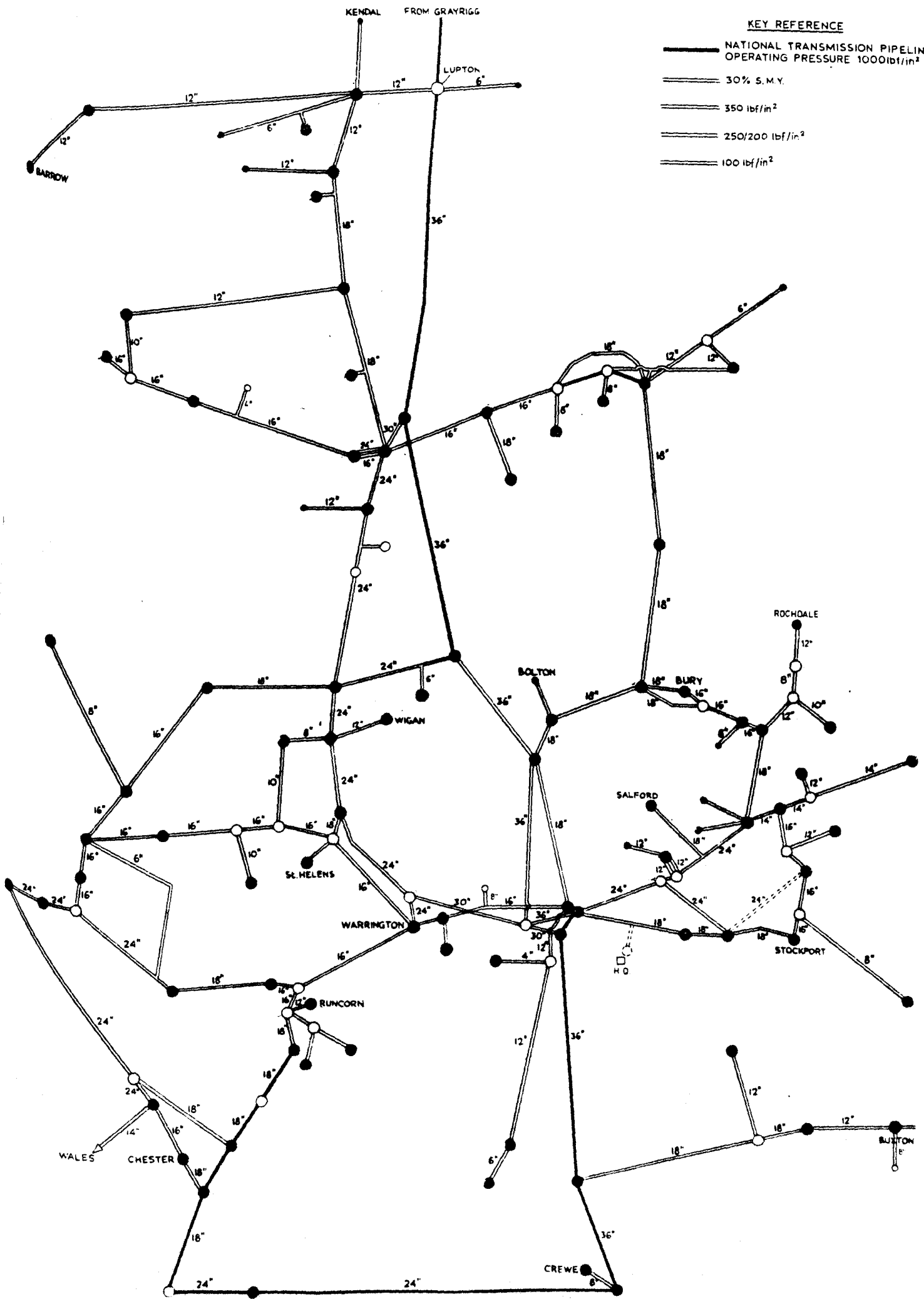


Fig. (A.5)

**Network A6 BRITISH GAS CORPORATION**  
**NORTH WEST REGION**  
 TRANSMISSION SYSTEM

**KEY REFERENCE**

-  NATIONAL TRANSMISSION PIPELINES  
OPERATING PRESSURE 1000lb/in<sup>2</sup>
-  30% S.M.Y.
-  350 lb/in<sup>2</sup>
-  250/200 lb/in<sup>2</sup>
-  100 lb/in<sup>2</sup>



**Fig. (A.6)**

**BRITISH GAS CORPORATION  
NATIONAL GAS  
TRANSMISSION SYSTEM**

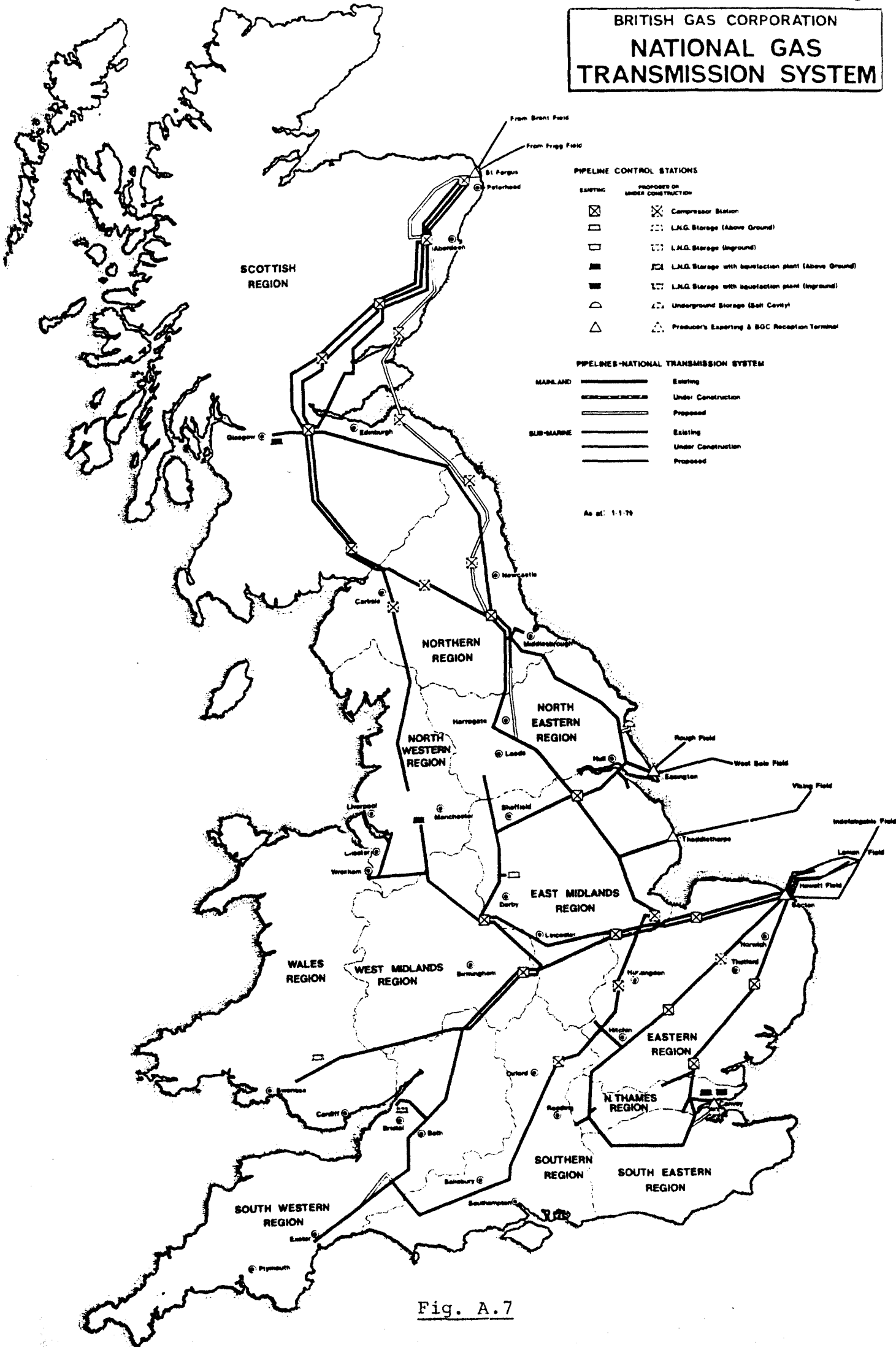


Fig. A.7

Network A8

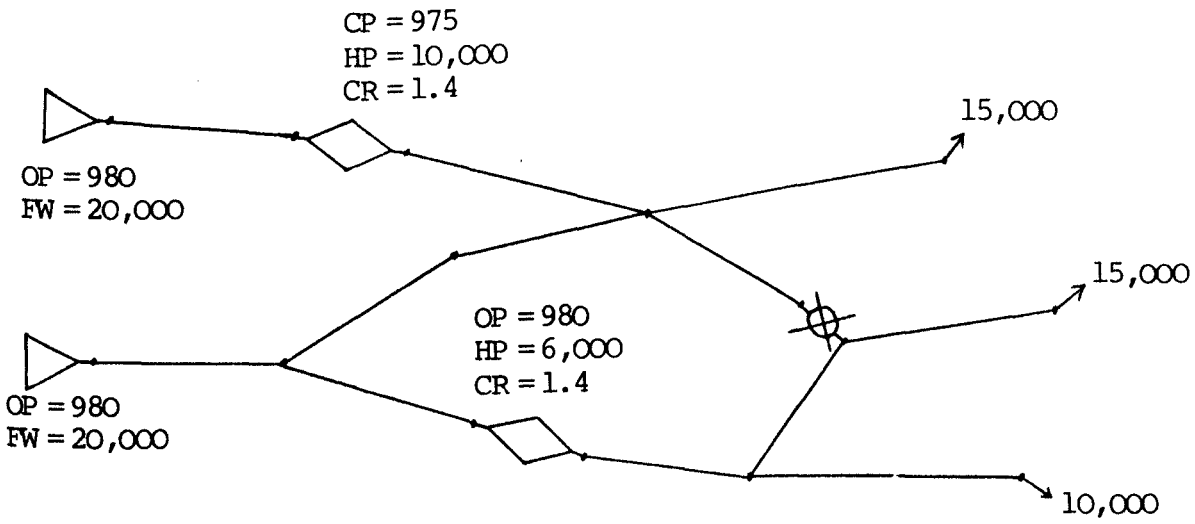


Fig. (A.8)

Network A9

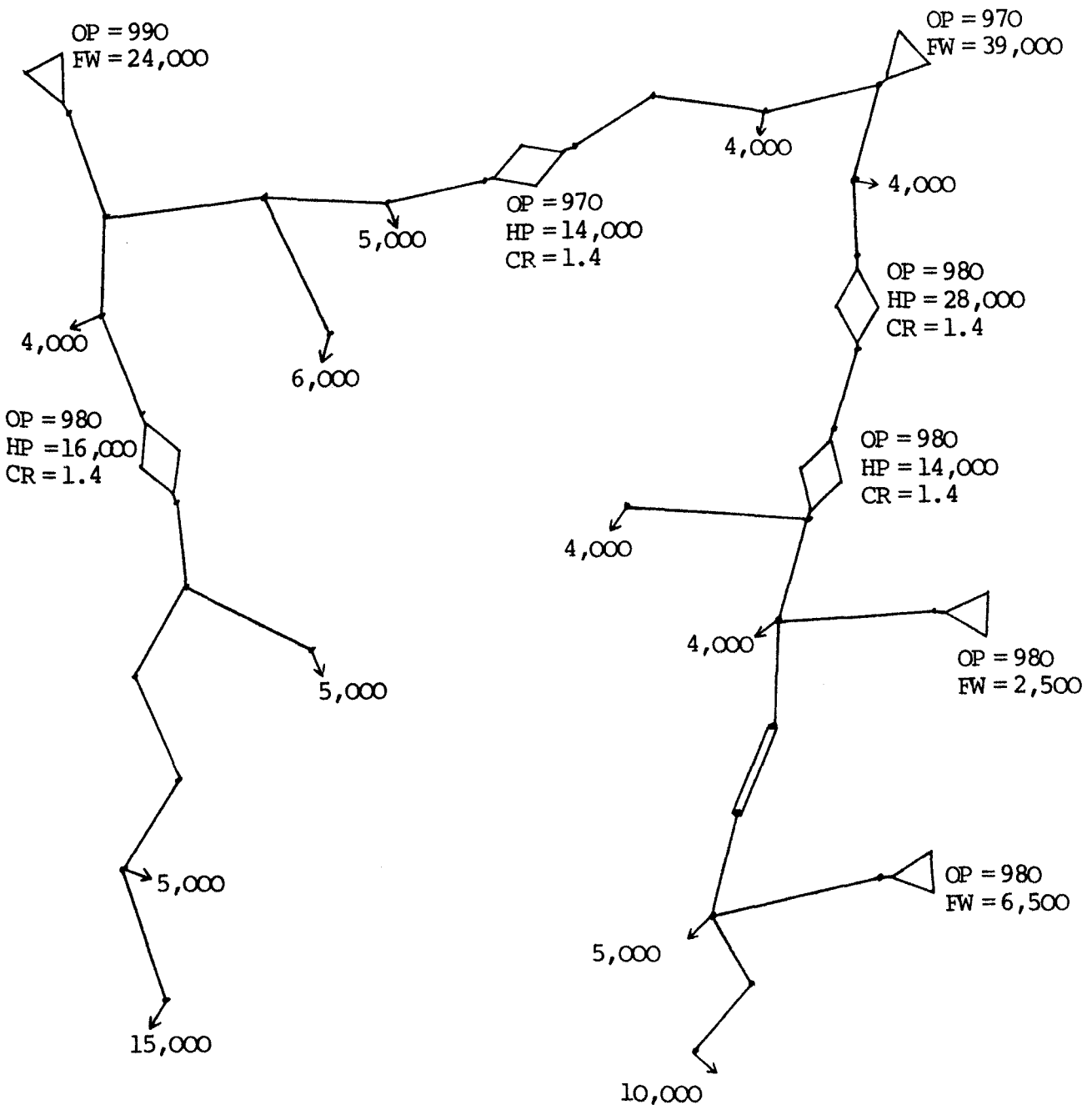


Fig. (A.9)

## APPENDIX B

The Mathematical Background to the PAN program

The purpose of this appendix is to give the mathematical background to the British Gas network analysis program PAN and in particular, show how the system of differential/algebraic equations (DAE's) considered in the main body of the thesis can be derived. An example network is provided to illustrate the structure of the DAE system.

B.1 The Basic Gas Flow Model

The gas flow in a single pipe can be modelled by two partial differential equations (PDE's) which describe the conservation of mass and momentum. The derivation of these equations has been discussed by many authors (for example, FINCHAM (1979)) and need not be repeated here. The equations are:

i) the conservation of mass

$$A \frac{\partial \rho}{\partial t} + M \frac{\partial q}{\partial x} = 0 \quad (\text{B.1.1})$$

and ii) the conservation of momentum

$$\frac{\partial P}{\partial x} + \frac{M}{A} \frac{\partial q}{\partial t} + \frac{M^2}{A^2} \frac{\partial}{\partial x} \left( \frac{q^2}{\rho} \right) + \frac{\pi d \tau}{A} + \rho g \sin \theta = 0 \quad (\text{B.1.2})$$

where the independent variables are the distance,  $x$ , and time,  $t$ . The dependent variables are the pressure,  $P$ ; the density,  $\rho$ ; the mass flow rate,  $q$  and the shear stress on the pipewall,  $\tau$ . The constants include the pipe diameter,  $d$ ; the pipe cross-sectional area,  $A$ ; the molecular weight of gas,  $M$ ; the acceleration due to gravity,  $g$  and the slope of the pipe inclined upward in the direction of flow at an angle,  $\theta$ .

These equations are derived under the assumptions that the pipe is straight with constant, circular cross-section



and that the pipe friction can be modelled using the friction equations derived under steady flow condition. A one-dimensional flow model is used which has been shown in WARD-SMITH(1971) to give an adequate representation of the dynamic gas flow in the network. Furthermore it is also assumed that the gas flow temperature is constant which means that the flow process is isothermal; this is true for long pipelines with slow dynamic changes where the temperature equalization with the ground outside takes place (WEIMANN (1978)).

A third equation is needed before eqns.(B.1.1)-(B.1.2) can be solved. This is the equation of state for an isothermal system:

$$a^2 = \frac{zRT}{\left[1 - \frac{P}{z} \frac{dz}{dP}\right]} = \left[\frac{dP}{d\rho}\right] \quad (\text{B.1.3})$$

where  $a$  denotes the isothermal speed of sound,  $T$  denotes the gas temperature and  $R$  is the real gas constant;  $z$  denotes a variable compressibility so the gas is treated as non-ideal.

In fully developed steady flow, the shear stress in eqn.(B.1.2) can be represented by the Fanning friction factor  $f$ ,

$$f = \frac{|\tau|}{\frac{1}{2}\rho v^2} \quad (\text{B.1.4})$$

where  $f$  is a dimensionless quantity and in general is a function of the Reynolds number of the flow and the relative roughness of the pipewall.

In the isothermal flow model described above, the unknown variables to be solved are usually the pressure,  $P$  and the mass flow rate,  $q$  or a variant of these variables. Other formulations based on the isentropic or adiabatic flow model have also been attempted (STONER(1968), SENS(1970), TAYLOR(1978) and COVINGTON(1979)). In these formulations, a further set of unknown variables such as the temperature is usually required to be solved. These formulations require

considerably more computational effort than the isothermal one and are normally employed to carry out specific simulation tasks such as the modelling of pipebreaks. From a perturbation analysis carried out in WEIMANN(1978) using the isothermal and isentropic models, it was shown that the former one only exhibits slight error compared with the later for very short pipes under extreme disturbances. Furthermore for normal simulation, the isothermal model has been found by many authors to predict the overall behaviour of the gas flow accurately. The isothermal assumption is therefore a reasonable one under most operating conditions.

Even with the assumption that the flow is isothermal, we must still solve a pair of hyperbolic PDE's which are difficult to solve on a computer. Simplifications to the model are normally made; the usual simplifications are either to neglect both inertia terms,  $\frac{M^2}{A^2} \frac{\partial}{\partial x} \left( \frac{q^2}{\rho} \right)$  and  $\frac{M \partial q}{A \partial t}$  in eqn.(B.1.2), or to neglect the first but include the second. As both inertia terms are very small (less than 1%) compared with the friction term during normal operation of the transmission network, it is reasonable to neglect them. However, when there is a large disturbance in the network such as a valve opening, a compressor shut down or a pipe-break, then the inertia terms can be significant in the neighbourhood of the disturbance. The disturbance, however, will be localised and will die away very rapidly. If the main interest is to investigate the overall behaviour of the network, it is still an acceptable approximation to neglect them. Only when one is interested in the local effect of a large disturbance should the inertia terms be included.

Neglecting both of the inertia terms means that the model is parabolic in nature which can be solved more readily on a computer.

## B.2 The Derivation of the Network Equations

This section considers the derivation of network equations without any machines in the network. As PAN is designed mainly for simulating slow dynamics, the two inertia terms,  $\frac{M}{A} \frac{\partial q}{\partial t}$  and  $\frac{M^2}{A^2} \frac{\partial}{\partial t} \left( \frac{q^2}{\rho} \right)$ , in the momentum equation are neglected as unimportant. The model is now isothermal and parabolic in nature. By substituting the equation of state (B.1.3) and the Fanning friction equation (B.1.4) into eqns. (B.1.1) and (B.1.2), we have

$$\frac{A}{a} \frac{\partial p}{\partial t} + M \frac{\partial q}{\partial x} = 0 \quad (\text{B.2.1})$$

and

$$\frac{\partial P}{\partial x} + \frac{M^2}{A^2} \frac{2f|q|}{\rho d} = 0 \quad (\text{B.2.2})$$

where  $q = \rho AV/M$ ; the term  $\rho g \sin \theta$  is also omitted from the momentum equation for the sake of simplicity. The treatment of this term is straight-forward and no loss of generality results from its omission.

Eqn. (B.2.2) can be written as

$$q = - \frac{1}{\lambda} \frac{\partial P}{\partial x} \quad (\text{B.2.3})$$

where

$$\lambda = \lambda(P, q) = \frac{M^2}{A^2} \frac{2f|q|}{\rho d} \quad (\text{B.2.4})$$

and is known as the resistivity of the pipe. Substituting eqn. (B.2.3) into eqn. (B.2.1), the non-linear parabolic equation in P can be obtained,

$$\frac{A}{Ma^2} \frac{\partial P}{\partial t} = \frac{\partial}{\partial x} \left\{ \frac{1}{\lambda} \frac{\partial P}{\partial x} \right\} \quad (\text{B.2.5})$$

The network can be regarded as consisting of individual pipes joining at nodes. It is convenient to take the origin of the pipes to be at the node being considered. Let the

length, cross-sectional area and resistivity of the  $i^{\text{th}}$  pipe be  $l_i$ ,  $A_i$  and  $\lambda_i$  respectively, and denote the pressure along the pipe by  $P_i(x_i)$  (see Fig.(B.2.1)). By assuming that the resistivity of the pipe,  $\lambda_i$ , varies sufficiently slowly along the pipe so that it can be approximated by a constant throughout the pipe, eqn.(B.2.5) can be written as

$$\frac{\lambda_i A_i}{Ma^2} \frac{\partial P_i(x_i)}{\partial t} = \frac{\partial^2 P_i(x_i)}{\partial x^2} \quad (\text{B.2.6})$$

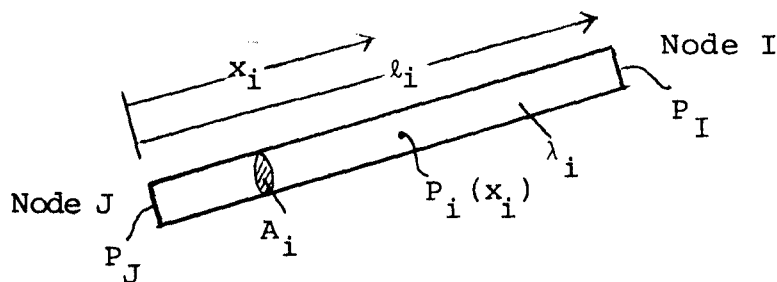


Fig.(B.2.1) The  $i^{\text{th}}$  pipe with nodes J, I

It is assumed in PAN that the gas pressure at the node is the same for all pipes intersecting at that node, that is

$$P_i(0) = P_J \quad \text{for all } i \in S_J \quad (\text{B.2.7})$$

where  $S_J$  is the set of pipes joining at node J and  $P_J$  is the common pressure at that point.

Furthermore, at each node there is also a continuity equation to be satisfied

$$\sum_{i \in S_J} q_i(0) = -d_J \quad (\text{B.2.8})$$

where  $d_J$  is the offtake at node J; and the flows are taken to be positive away from the node.

We can now derive the network equation for the  $i^{\text{th}}$  pipe joining at node J. By considering the Taylor series

expansion of  $P_i(l_i)$  about the point  $x_i=0$ ; and simplifying the expression using eqns.(B.2.3) and (B.2.6) and rearranging, we get

$$\frac{P_i(l_i) - P_i(0)}{\ell_i \lambda_i} = -q_i(0) + \frac{A_i \ell_i}{2Ma^2} \left[ \frac{2}{3} \frac{dP_i(0)}{dt} + \frac{1}{3} \frac{dP_i(l_i)}{dt} \right] + O(\ell_i^3) \quad (\text{B.2.9})$$

Summing eqn.(B.2.9) over all pipes joining at node J and applying the continuity equation gives

$$\sum_{I \in S_J} C_{JI} (P_I - P_J) = d_J + \sum_{I \in S_J} \frac{V_{JI}}{2Ma^2} \left[ \frac{2}{3} \frac{dP_J}{dt} + \frac{1}{3} \frac{dP_I}{dt} \right] \quad (\text{B.2.10})$$

where  $P_I$  is the pressure at the other end of the  $i^{\text{th}}$  pipe joining at node J;  $V_{JI}$  and  $C_{JI}$  are the volume and conductivity of the pipe connecting nodes J, I and are defined as

$$V_{JI} = A_i \ell_i \quad \text{and} \quad C_{JI} = \frac{1}{\lambda_i \ell_i} \quad (\text{B.2.11})$$

A system of ordinary differential equations (ODE) can be obtained by combining eqn.(B.2.10) over all the nodes in the network

$$E^* \frac{d\underline{P}}{dt} = A^*(\underline{P}) \underline{P} - \underline{d}(t) \quad (\text{B.2.12})$$

where the elements of matrix  $E^*$  are defined as

$$E_{JJ}^* = \frac{2}{3} \sum_{I \in S_J} \left( \frac{V_{JI}}{2Ma^2} \right)$$

$$E_{JI}^* = \begin{cases} 0 & \text{if } I \notin S_J \\ \frac{1}{3} \left( \frac{V_{JI}}{2Ma^2} \right) & \text{if } I \in S_J \end{cases}$$

and the elements of matrix  $A^*$  are defined as

$$A_{JJ}^* = - \sum_{I \in S_J} C_{JI}$$

$$A_{JI}^* = \begin{cases} 0 & \text{if } I \notin S_J \\ C_{JI} & \text{if } I \in S_J \end{cases}$$

The elements of matrix  $A^*$  are functions of  $\underline{p}$  because the conductivities of the pipes,  $C_{JI}$ 's, depend on the pipe flows  $q$ , (see the definition of  $C_{JI}$ 's in eqns. (B.2.4) and (B.2.11)), which in turn are expressible in terms of  $\underline{p}$  via eqn. (B.2.2). Both matrices  $A^*$  and  $E^*$  are symmetric and diagonally dominant.

In PAN, the Crank-Nicolson method is used to solve the linearised differential equations (B.2.10); further details of this will be given in section B.6, after the derivation of the DAE system.

### B.3 The Machine Models

Machines are active components in the network which are capable of some form of control. They can be used to increase or reduce the flow or pressure of gas in the network. Four types of machines are used in PAN. They are compressors, regulators, sources and valves. Machines are assumed to lie between two nodes referred to as the inlet and the outlet nodes. A source simply has an outlet node.

The internal working of these machines are very complex. Fortunately, because we are only interested in modelling the effects of the machines on the rest of the network, we do not need to simulate their behaviour in detail. An adequate representation of the machine can usually be achieved by using less than 7 inequalities involving not more than 3 variables. The variables required are the machine inlet pressure  $P_I$ , the machine outlet pressure  $P_O$  and the flow through the machine  $Q$  (see Fig. (B.3.2)). The models are described separately below.

#### a) The Compressor

The compressor is the most complicated type of machine and is used to boost the gas pressure in the network. The

simplest model assumes that the outlet pressure or the compressor flow is controlled to a fixed value. Unfortunately this model is not realistic enough for most applications and further restrictions on the operation of compressor using a compressor envelope is required. The compressor envelope defines the safe operating limits of the compressor; a typical envelope for the centrifugal compressor is given in Fig.(B.3.1). In addition, the compressor must satisfy several physical limits on the maximum horsepower, maximum compressor speed, maximum compression ratio and maximum flow. Because it is very difficult to deal with the compressor envelope, only the physical limits are considered (PAN, WEIMANN(1978), STONER(1969)). These limits are modelled using a set of inequalities. The compressor envelope is sometimes used merely (for example, PAN) for outputting warning messages if it is violated.

The basic compressor model is given in fig.(B.3.2). The compression ratio, the horsepower and the compressor speed are non-linear functions of  $P_I$ ,  $P_O$  and  $Q_{comp}$ . As the expression for the compressor speed is rather complicated, only the expressions for the compression ratio and horsepower are given here,

$$\text{Compression ratio (C.R.)} = (P_O + E_O) / (P_I - E_I) \quad (\text{B.3.1})$$

$$\text{Horsepower} = (c Q_{comp} ((\text{C.R.})^\beta - 1)) \quad (\text{B.3.2})$$

where  $E_O$ ,  $E_I$  are pressure losses on outlet, inlet side of compressor; and  $c$ ,  $\beta$  are constants depending on the particular compressor used. Further details on the working of the compressor can be found in SCHEEL(1972).

#### b) The Regulator

The regulator is used to lower the gas pressure. It must therefore satisfy the operating restrictions that the outlet pressure is less than the inlet pressure and that the

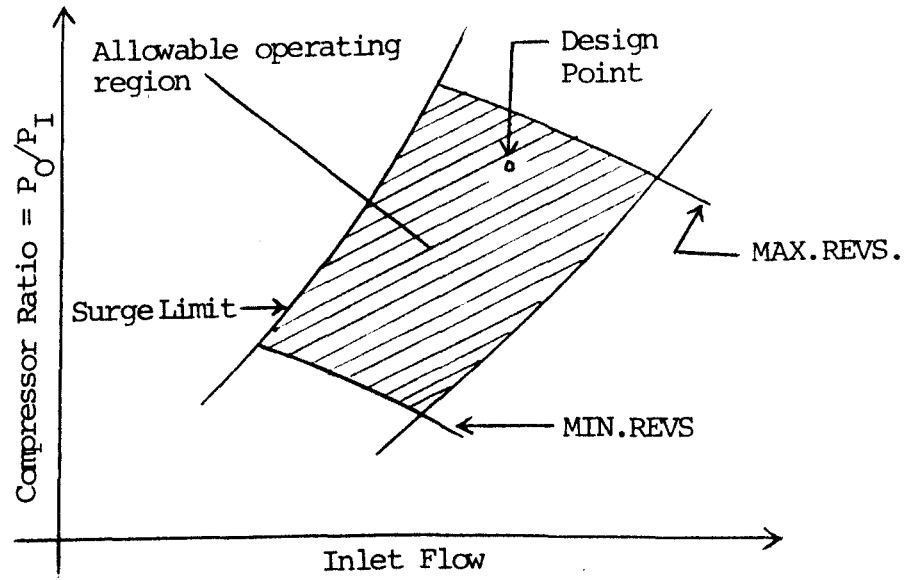
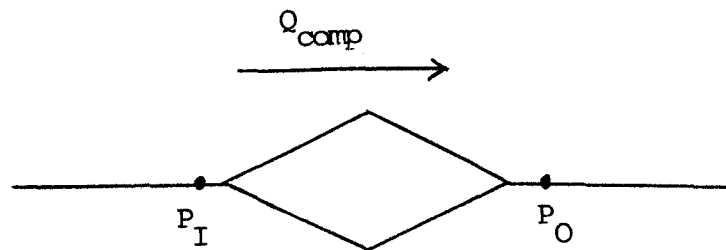


Fig. (B.3.1) - Operating envelope for centrifugal compressor



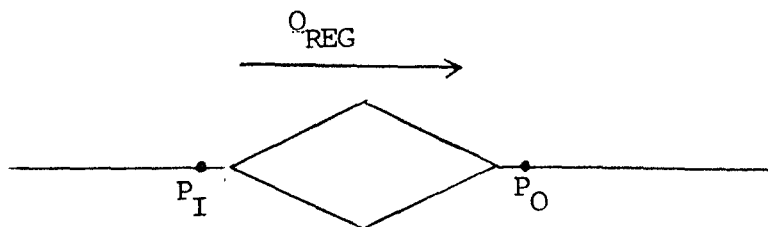
Constraints

- $P_O$
  - $Q_{comp}$
  - Compressor Ratio
  - Horse Power
  - Compressor Speed
- }  $\leq$  maximum

- $P_I$
  - $P_O - P_I$
- }  $\geq$  minimum

Fig. (B.3.2) - The Compressor



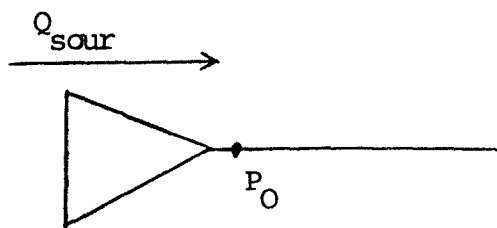


Constraints

$$\left. \begin{array}{l} P_O \\ Q_{REG} \end{array} \right\} \leq \text{maximum}$$

$$\left. \begin{array}{l} P_I \\ P_I - P_O \end{array} \right\} \geq \text{minimum}$$

Fig. (B.3.3) - The Regulator

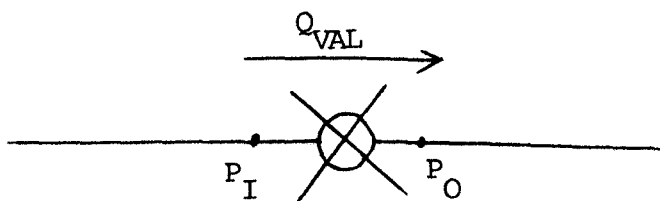


Constraints

$$\left. \begin{array}{l} P_O \\ Q_{sour} \end{array} \right\} \leq \text{maximum}$$

$$Q_{sour} \geq \text{minimum}$$

Fig. (B.3.4) - The Source



Closed

$$Q_{VAL} = 0$$

Open

$$P_I = P_O$$

Fig. (B.3.5) - The Valve

gas should not flow backwards through the regulator. A regulator would usually have its outlet pressure or flow controlled to a set value.

c) The Source

The source is the gas supply point in the network and hence its model consists of only an outlet node. It could be a gas storage field or an off-shore gas supply terminal or even a point where the gas is transferred from another network. To model the different types of sources accurately, different models are normally required. For example, STONER (1969) used a non-linear equation involving the flow and pressure to describe the operation of a storage field. Fortunately for most applications, a simpler model which restricts the pressure to be less than a maximum value and the flow to vary within a fixed range is usually adequate.

d) The Valve

Valves are important only in dynamic simulations. It can be considered as a particularly simple type of machine with only two operating states, open or closed. When open the flow is unaffected by the valve and when closed, no flow is allowed to pass.

A summary of the machine models used in this thesis is outlined in figs.(B.3.2) to (B.3.5). Similar machine models are also used in WEIMANN(1978).

#### B.4 Solution of Network with Machines

The presence of machines introduces extra unknown variables (machine flows) and hence additional equations are needed for the solution. Even by using the simplified machine model described in the previous section, a set of inequalities is still needed to be solved for each machine. Because of difficulties in dealing with inequalities, PAN assumes that for each machine, one of the constraints actually

reaches its extreme value and is used as the controlling constraint for the machine. This gives rise to an algebraic equation for each machine. Any nonlinear constraints are linearised using Newton's method to give a general machine equation of the form,

$$m_1 P_I + m_2 P_O + m_3 Q = \sigma \quad (\text{B.4.1})$$

where  $m_1$ ,  $m_2$ ,  $m_3$  and  $\sigma$  are coefficients whose values depend on the controlling constraint of the machine. For example, consider the simple case when the machine is operating on outlet pressure constraint, we require a machine equation of the form,

$$P_O = P_{\max} \quad (\text{B.4.2})$$

and hence the coefficients are simply given by

$$m_1 = m_3 = 0, \quad m_2 = 1 \quad \text{and} \quad \sigma = P_{\max}$$

where  $P_{\max}$  is the maximum machine outlet pressure.

For the inlet or outlet node of the machine, the network equation (eqn.(B.2.10)) must be modified in order to take into account the flow through the machine. As the machine flow  $Q$  merely acts as an additional supply at the outlet machine node and as a demand for the inlet machine node, eqn.(B.2.10) becomes

$$\sum_{I \in S_m} C_{mI} (P_I - P_m) = d_m + kQ + \sum_{I \in S_m} \frac{V_{mI}}{2Ma^2} \left( \frac{2}{3} \frac{dP_m}{dt} + \frac{1}{3} \frac{dP_I}{dt} \right) \quad (\text{B.4.3})$$

where  $m$  refers to either the inlet or the outlet node of the machine and  $k = \begin{cases} 1 & \text{if } m \text{ is the machine inlet node;} \\ -1 & \text{if } m \text{ is the machine outlet node.} \end{cases}$

By collecting the network equations (eqns.(B.2.10) and (B.4.3)) over all the nodes in the network, we get

$$E^* \frac{dP}{dt} = A^* (P) P - K Q - d(t) \quad (\text{B.4.4})$$

where  $K$  is the machine flow incidence matrix defined as

$$K_{ij} = k \text{ as defined in eqn. (B.4.3) if node } i \text{ is the inlet or outlet node of machine } j \\ \text{and } = 0 \text{ otherwise}$$

Eqn. (B.4.4) together with the algebraic machine equations are combined to give a set of DAE of the form,

$$E \frac{dy}{dt} = A(y) y - \underline{d}(t) \quad (\text{B.4.5})$$

where  $y = [P, Q]$ . The matrix  $A(y)$  now contains the asymmetric machine equations and the machine flow incidence matrix  $K$ ; and the matrix  $E$  is singular.

#### B.5 The Structure of the DAE system

To understand the structures of matrices  $E$  and  $A$ , it is necessary to use a test network. Consider a simple test network given below.

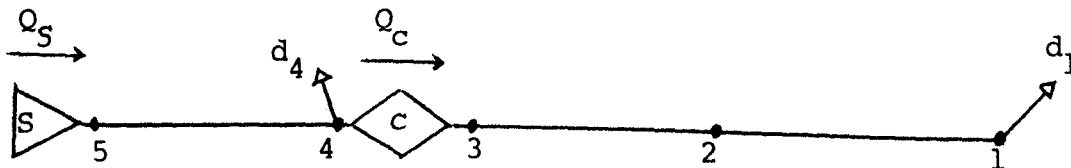


Fig. (B.5.1) - Test Network

The network consists of a compressor  $C$ , a source  $S$  together with two demands,  $d_1$  and  $d_4$ , at nodes 1 and 4 respectively; the nodes are numbered as shown in Fig. (B.5.1).

Let  $E_{ij} = \frac{1}{3} \left( \frac{V_{ij}}{2Ma^2} \right)$ . From eqn. (B.2.10), it is clear that the network equations at node 1 and 2 are respectively given by,

$$2E_{12} \frac{dP_1}{dt} + E_{12} \frac{dP_2}{dt} = -C_{12} P_1 + C_{12} P_2 - d_1 \quad (\text{B.5.1})$$

and

$$2(E_{12} + E_{23}) \frac{dP_2}{dt} + E_{12} \frac{dP_1}{dt} + E_{23} \frac{dP_3}{dt} = -(C_{12} + C_{23}) P_2 + C_{12} P_1 + C_{23} P_3 \quad (\text{B.5.2})$$

and from eqn. (B.4.3), the respective network equations at the machine nodes 3, 4 and 5 take the form,

$$2E_{23} \frac{dP_3}{dt} + E_{23} \frac{dP_2}{dt} = -C_{23} P_3 + C_{23} P_2 + Q_C \quad (\text{B.5.3})$$

$$2E_{45} \frac{dP_4}{dt} + E_{45} \frac{dP_5}{dt} = -C_{45} P_4 + C_{45} P_5 - Q_C - d_4 \quad (\text{B.5.4})$$

and

$$2E_{45} \frac{dP_5}{dt} + E_{45} \frac{dP_4}{dt} = -C_{45} P_5 + C_{45} P_4 + Q_S \quad (\text{B.5.5})$$

Lastly, the machine equations for the compressor C and the source S can also be written,

$$m_1^C P_4 + m_2^C P_3 + m_3^C Q_C = \sigma_C \quad (\text{B.5.6})$$

and

$$m_2^S P_5 + m_3^S Q_5 = \sigma_S \quad (\text{B.5.7})$$

Put  $\underline{y} = [P_1, P_2, P_3, P_4, P_5, Q_C, Q_S]$ , then it can be easily shown that eqns. (B.5.1) to (B.5.7) can be written as a DAE system of the form given in (B.4.5.). The matrices E and A now have the following structures,

$$E = \begin{pmatrix} 2E_{12} & E_{12} & & & & & \\ E_{12} & 2(E_{12}+E_{23}) & E_{23} & & & & \\ & E_{23} & 2E_{23} & & & & \\ & & & 2E_{45} & E_{45} & & \\ & & & E_{45} & 2E_{45} & & \\ & & & & & & \\ & & & & & & \end{pmatrix} \quad (\text{B.5.8a})$$

$$A(\underline{y}) = \begin{pmatrix} -C_{12} & C_{12} & & & & & \\ C_{12} & -(C_{12}+C_{23}) & C_{23} & & & & \\ & C_{23} & -C_{23} & & & +1 & \\ & & & -C_{45} & C_{45} & -1 & \\ & & & C_{45} & -C_{45} & & +1 \\ & & m_2^C & m_1^C & & m_3^C & \\ & & & & m_2^S & & m_3^S \end{pmatrix} \quad (\text{B.5.8b})$$

where the entries in the matrices are assumed to be zero unless otherwise stated. From the above, it is therefore clear that the matrix  $E$  is singular and matrix  $A$  is partially structured. The symmetric and diagonally dominant part of matrices  $E$  and  $A$  correspond to the unknown pressure at the network nodes. The asymmetric part of matrix  $A$  corresponds only to the unknown machine variables (the pressures at the inlet and outlet nodes of the machine, and the machine flow).

## B.6 The Solution Scheme Employed in PAN

This section briefly outlines the solution scheme employed in PAN to solve the DAE system (B.4.5). PAN handles the differential and algebraic equations in the DAE system separately, where the differential equations are first integrated and then solved together with the algebraic equations in the system. The set of linearised differential equations considered in PAN is given by (see eqn.(B.4.4))

$$E^* \frac{d\underline{P}}{dt} = A^* \underline{P} - K\underline{Q} - \underline{d}(t) \quad (\text{B.6.1})$$

where  $E^*$  and  $A^*$  are linearised matrices as defined in eqn. (B.2.12). These equations are numerically integrated using the constant-step Crank-Nicolson method except for  $-K\underline{Q}$ , which is integrated using the backward Euler method; this gives rise to a set of equations of the form

$$(E^* - \frac{1}{2}hA^*) \underline{P}_{n+1} + hK\underline{Q}_{n+1} = \underline{r}_n \quad (\text{B.6.2})$$

where  $\underline{r}_n = (E^* + \frac{1}{2}hA^*) \underline{P}_n - \frac{1}{2}h(\underline{d}(t_{n+1}) + \underline{d}(t_n))$

and  $h = t_{n+1} - t_n$ ,

$\underline{r}_n$  is a vector containing the known quantities at time  $t_n$ , and  $\underline{P}_{n+1}$ ,  $\underline{Q}_{n+1}$  are the solutions required at the new time level,  $t_{n+1}$ .

Equations (B.6.2) can now be combined with the algebraic equations in the DAE system, which are given by (see eqn.(B.4.1) and section B.5),

$$M_1 \underline{P}_{n+1} + M_2 \underline{Q}_{n+1} = \underline{\sigma} \quad (\text{B.6.3})$$

where matrices  $M_1$ ,  $M_2$  contain the coefficients of the linearised constraint equations (B.4.1), and  $\underline{\sigma}$  is a known right-hand vector depending on the operating constraints of the machines.

Equations (B.6.2) and (B.6.3) provide sufficient equations to solve for the unknown variables,  $P_{n+1}$ ,  $Q_{n+1}$ . A block matrix partitioning method described in GOLDWATER(1976) is employed to solve the system of linear equations at each time step.



## APPENDIX C

Arithmetic Operation Counts

This appendix states some preliminary results quoted in GEORGE(1974) and ROSE(1972) that can be used to count arithmetic operations. The term 'arithmetic operation' means multiplicative operation (multiplication or division) only. The results are derived under the assumption that exact numerical cancellation does not occur. They are stated without proof as follows.

LEMMA C-1

Let  $M$  be an  $N \times N$  symmetric and positive-definite matrix. Then provided we avoid operating on zeros, the number of operations required to factorize  $M$  into its Cholesky factors  $LL^T$  is

$$\theta_{LL^T} = \sum_{i=1}^N (n(L_i^C) - 1)(n(L_i^C) + 2) / 2 \quad (C-1)$$

together with  $N$  square roots.

LEMMA C-2

Let  $L$  be a nonsingular  $N \times N$  triangular matrix and  $\underline{x}$  be the solution to  $L\underline{x} = \underline{b}$ , with  $x_i = 0$  unless  $i = q_j$ ,  $j = 1, 2, \dots, \ell$  ( $\ell \leq N$ ). Then the number of operations  $\theta_L$  required to solve  $L\underline{x} = \underline{b}$  is

$$\theta_L = \sum_{i=1}^{\ell} n(L_{q_i}^C) \quad (C-2)$$

From the above Lemma, we can easily deduce the following result

LEMMA C-3

Let  $L$  be an  $N \times N$  nonsingular triangular matrix with  $LM = \bar{M}$ . Then the number of operations required to compute  $M$  via back-substitution, given  $L$  and  $\bar{M}$ , is

$$\sum_{i=1}^N n(L_i^C) n(M_i^R) \quad (C-3)$$

LEMMA C-4

Let  $A$ ,  $B$  and  $C$  be given sparse matrices, with  $A = BC$ . Then the number of operations required to compute  $A$  from  $B$  and  $C$  is

$$\theta_{BC} = \sum_{i=1}^N n(B_i^C) n(C_i^R) \quad (C-4)$$

where  $N$  is the number of columns in  $B$ .

LEMMA C-5

Let  $B$  be a given sparse matrix with  $A = B^T B$ . Then the number of operations required to compute  $A$  from  $B$  is

$$\theta_{B^T B} = \sum_{i=1}^N n(B_i^R) (n(B_i^R) + 1) / 2 \quad (C-5)$$

since only the upper or lower half of  $A$  needs to be computed.

LEMMA C-6

Let  $B$  and  $C$  be given sparse matrices where the product  $A = B^T C$  is known to be symmetric and  $B_{ij} \neq 0 \implies C_{ij} \neq 0$ . Then the number of operations required to compute  $A$  from  $B$  and  $C$  satisfies

$$\theta_{B^T C} \leq \sum_{i=1}^N n(B_i^R) (n(C_i^R) + 1) / 2 \quad (C-6)$$

LEMMA C-7

Let  $L$  be an  $N \times N$  nonsingular triangular matrix, and let  $\underline{x}$  be the solution to the system  $L\underline{x} = \underline{b}$ . Then

$$n(\underline{x}) \geq n(\underline{b}) \quad (C-7)$$

From the above, it follows that

LEMMA C-8

Let  $L$ ,  $M$  and  $\bar{M}$  be as defined in Lemma C-3. Then

$$n(M) \geq n(\bar{M}) \quad (C-8)$$

The proof of the above Lemma's can be found in GEORGE (1974).

## APPENDIX D

The Numerical Methods

Four methods selected are the first order theta method, the second order strongly S-stable embedded diagonally-implicit Runge-Kutta (DIRK) method, the second order Rosenbrock-type method and the second order DIRK method using the Rosenbrock-type method as a predictor. These methods when applied to a DAE system are described separately below.

D.1 The Theta Method

This simple one stage method when applied to (1.3.1) gives

$$F(\underline{y}_{n+1}) = E\underline{y}_{n+1} - E\underline{y}_n - h[(1-\theta)\underline{f}_n + \theta\underline{f}_{n+1}] = 0$$

where

$$\underline{f}_n = A\underline{y}_n - \underline{d}(t_n), \quad \underline{y}_n = \underline{y}(t_n) \quad (D.1.1)$$

and

$$0.0 \leq \theta \leq 1.0, \quad n = 0, 1, 2, \dots$$

The method is A-stable for  $\theta=1/2$ , strongly A-stable for  $\theta > 1/2$  and second order only when  $\theta = 1/2$ . A reasonable compromise is to choose  $\theta = 0.55$  as suggested in PROTHERO(1974).

The modified-Newton method is used to solve the system of nonlinear algebraic equations at each step. The expression for the local truncation error is rather complicated and shall not be given here, interested readers are referred to HOPKINS(1976) for details. For this formula, an estimate of the global error (see PROTHERO(1977)) at time  $t_{n+1}$  is given by

$$\underline{g}_{n+1} = G^{-1}E \underline{g}_{n+1} - [(1-\theta)I - G^{-1}E] \underline{g}_n / \theta \quad (D.1.2)$$

where

$$G = E - h \theta A \quad (D.1.3)$$

$\underline{g}_n$  is the global error estimate at time  $t_n$   
and  $\underline{l}_{n+1}$  is the local error estimate at time  $t_{n+1}$ .

A suitable value of RESC (see eqn.(4.4.3)) for this method is 5.0.

## D.2 The Embedded Diagonally-Implicit Runge-Kutta Method

The second order, strongly S-stable, embedded DIRK method given in CASH(1979) when applied to (1.3.1) is defined as

$$\begin{aligned} E\underline{k}_1 &= h\underline{f}(t_n + \alpha h, \underline{y}_n + \alpha\underline{k}_1) \\ E\underline{k}_2 &= h\underline{f}(t_n + \tau_2 h, \underline{y}_n + (\tau_2 - \alpha)\underline{k}_1 + \alpha\underline{k}_2) \\ E\underline{k}_3 &= h\underline{f}(t_n + h, \underline{y}_n + b_1\underline{k}_1 + b_2\underline{k}_2 + b_3\underline{k}_3) \end{aligned} \quad (D.2.1)$$

the 3<sup>rd</sup> order solution together with a 2<sup>nd</sup> order embedded solution are given by

$$\underline{y}_{n+1}^{(3)} = \underline{y}_n + b_1\underline{k}_1 + b_2\underline{k}_2 + \alpha\underline{k}_3 \quad (D.2.2)$$

and

$$\underline{y}_{n+1}^{(2)} = \underline{y}_n + c_1\underline{k}_1 + c_2\underline{k}_2 \quad (D.2.3)$$

respectively.

The parameters are defined as

$$\begin{aligned} \alpha &= 0.4358,6652, & \tau_2 &= 0.7179,3326, \\ b_1 &= 1.2084,9665, & b_2 &= -0.6443,6317, \\ c_1 &= 0.7726,3013, & c_2 &= 0.2273,6987. \end{aligned}$$

This formula required the solution of 3 systems of nonlinear algebraic equations at each time step using modified-Newton method. The iteration matrix is the same for each system of equations and is given by

$$G = E - h \alpha A \quad (D.2.4)$$

From eqns.(D.2.2), (D.2.3) and (4.3.2), an estimate of the local error for  $\underline{y}_{n+1}^{(2)}$  is,

$$\underline{\ell}_{n+1} = G^{-1} E (\underline{y}_{n+1}^{(3)} - \underline{y}_{n+1}^{(2)}) \quad (D.2.5)$$

and an estimate of the global error (derived using the procedure outlined in PROTHERO(1977)) is given by

$$\underline{g}_{n+1} = G^{-1} E [(1-\alpha)G^{-1}E - (1-2\alpha)I] \underline{g}_n / \alpha + \underline{\ell}_{n+1} \quad (D.2.6)$$

A suitable value of RESC (see eqn.(4.4.3)) for this method is 6.0.

### D.3 The Rosenbrock-type Method

For the autonomous DAE system of the form

$$E \frac{dy}{dt} = A(y) (y) - \underline{s}(y) = \underline{f}(y) \quad (D.3.1)$$

the 2-stage, 2nd order Rosenbrock-type method given in SCRATON(1981) is defined as,

$$\begin{aligned} \underline{y}_{n+1} &= \underline{y}_n + h b_1 \underline{k}_1 + h b_2 \underline{k}_2 \\ G \underline{k}_1 &= \underline{f}_n \end{aligned} \quad (D.3.2)$$

$$G \underline{k}_2 = \underline{f}_n^* + h d_{21} A \underline{k}_1$$

where

$$\underline{f}_n = A \underline{y}_n - \underline{s}(\underline{y}_n)$$

$$\underline{f}_n^* = \underline{f}(\underline{y}_n + h a_{21} \underline{k}_1)$$

$$G = E - h d A$$

and the parameters are defined as

$$b_1 = 1/4 \quad , \quad b_2 = 3/4 \quad , \quad a_{21} = 2/3 \quad (D.3.3)$$

$$d_{21} = -\frac{4}{3}d \quad , \quad d = 1 - 1/\sqrt{2},$$

It can be shown that the method is 2<sup>nd</sup> order for any matrix A and is strongly A-stable when A is updated at every step. An accurate estimate of the local error can be computed using

$$\underline{\ell}_{n+1} = \frac{h}{6} G^{-1} [E(\underline{k}_1 + 3\underline{k}_2) - \underline{f}_n - 3\underline{f}_n^*] \quad (D.3.4)$$

and an estimate of the global error can also be derived using the procedure of PROTHERO,

$$\underline{g}_{n+1} = [\bar{b}(G^{-1}E)^2 + (1-\bar{b})G^{-1}E] \underline{g}_n + \underline{\ell}_{n+1} \quad (D.3.5)$$

where

$$\bar{b} = b_2(a_{21} + d_{21})/d^2$$

A suitable value of RESC for this method is 5.5.

#### D.4 The DIRK Method using the Rosenbrock-type Method as a Predictor

Associated with the Rosenbrock-type method (D.3.2) is a DIRK formula given by

$$\begin{aligned} \underline{y}_{n+1} &= \underline{y}_n + hb_1\underline{k}_1 + hb_2\underline{k}_2 \\ E\underline{k}_1 &= \underline{f}(\underline{y}_n + h\underline{k}_1) \\ E\underline{k}_2 &= \underline{f}(\underline{y}_n + h(a_{21} + d_{21})\underline{k}_1 + h\underline{k}_2) \end{aligned} \quad (D.4.1)$$

where the parameters are as defined in eqn.(D.3.3). This formula is second order and also strongly A-stable. The expressions for the local and global error estimate defined in section D.3 can still be used for this formula. The

formula requires the solution of two systems of nonlinear algebraic equations at each time step and are solved using the modified-Newton method; the Rosenbrock-type method (D.3.2) is used to provide an accurate initial prediction for the modified-Newton iteration. A suitable value of RESC that can be used for this method is 5.5.

The extension of the above pair of methods to the non-autonomous case is discussed in section 4.3.3.



## APPENDIX E

The Implementation of the Variable-step Integrator

This appendix describes the basic design and implementation of the variable-step integrators, based on the numerical methods outlined in appendix D, for solving the DAE system (1.3.1). The strategies discussed in Chapter 4 are used in the implementation. As the strategies employed rely only on the assumption that the DAE system arises from the discretization of a system of parabolic PDE's, the integrators implemented can also be used for solving other similar systems. Thus the discussion carried out in this appendix will be based on a general DAE system of the form given in (1.3.1) and no reference will be made to the gas transmission network problem.

Because the implementation is largely independent of the numerical methods used, only one integrator will be discussed in detail. The integrator is called ROSEN and is based on the Rosenbrock-type method (D.3.2). This integrator also includes the diagonally-implicit Runge-Kutta (DIRK) method (D.4.1) using the above Rosenbrock-type method as a predictor; the DIRK method is used when the nonlinear algebraic equations in the DAE system have not converged satisfactorily (see section 4.4.1.2).

The FORTRAN code of the integrator ROSEN is given at the end of this appendix. The code is extensively documented and is programmed to be both readable and efficient. It is divided into stages to enhance its readability.

E.1 The Basic Design of the Integrator

The integrator is designed to be a low level routine which implements only the normal and restart phases described in section 4.4; the location of disturbance is not included as this operation is problem dependent. It is written in

"reverse communication" form as discussed in DEW(1981) so that the evaluation of matrix G, the DAE system and the solution of the linear equations are all performed in the calling program. This removes the need to pass the problem specification routines into the integrator which has the advantages that it is easier to maintain the integrator and that it is easier to incorporate the integrator into a large program designed to solve PDE's. The disadvantage of writing the code in "reverse communication" form is that it is necessary to carry along a large number of variables, which has to be passed through the parameter list; this makes the integrator inconvenient to use for the inexperienced users.

In order to shorten the parameter list while allowing flexibility in the use of the integrator, the set of variables which will not be needed by the user during the integration are placed in the common blocks. This set of variables includes the strategic parameters, the coefficients of the numerical methods, the constants of the machines and the error estimates (see the code of ROSEN); they are only required to be set up by the user before the integration begins. Only the working vectors and those variables that need to be set by the user during the computation are included in the parameter list. An example on how the integrator is to be used is shown in fig. (E.1).

To make the integration robust, the following error conditions are implemented.

i) When using the standard error test of the form  $\|\underline{x}\| \leq \text{EPS}$  for accepting the step, the limiting precision tests discussed in DEW(1978) require that,

a) the tolerance must satisfy

$$\text{EPS} \geq 20 U \|\underline{y}\| \quad (4.5.1)$$

b) the stepsize must satisfy

$$|h| \geq 4 U |t| + \sigma \quad (4.5.2)$$

where U is the relative precision of the machine (the smallest number such that  $1+U \neq 1$ ) and  $\sigma$  is the smallest positive number that can be stored in the machine.

```

      setting up
      :
RETRY: CALL ROSEN( ... T,H,Y,DY,DEL,DYY,IND,WT,W, ... )
      IF IND(2)=0 THEN integration has finished,
                           check IND(1) for possible error
      ELSE
      BEGIN
        The action depends on the value of IND(2)...
      IF IND(2)
        =1 THEN FORM the iteration matrix  $G = E - h dA$ 
                and COMPUTE the LU factors of G;
        =2 THEN evaluate a new function  $\underline{f}(t, \underline{y})$  in vector DY;
        =3 THEN CARRYOUT a back-substitution for  $DEL = G^{-1} * \underline{r}$ 
                where  $\underline{r} = E * W(7, .) + DYY$ 
        =4 THEN COMPUTE an iteration correction
                 $DEL = G^{-1} * (E * \underline{y}' - \underline{f}(t, \underline{y}))$ ;
        =5 THEN COMPUTE the algebraic equations and
                return the results in vector DEL;
      GOTO RETRY to recall the integrator
      END;
      :

```

Note: The meanings of the variables used are explained in the coding of integrator ROSEN

Fig. E.1 - Method of using the integrator ROSEN in the calling program.

The integration is terminated with an error indication if either of the above is not satisfied. This often occurs when the solution is unstable. It is necessary to relax the accuracy tolerance before the integration is continued.

ii) During the normal phase (see section 4.4.1), the stepsize is halved either when more than 3 evaluations of matrix G have been carried out or when the estimate of the

local error fails the error test. If the solution is still unsuccessful after 3 such stepsize reductions in a step, the integration is terminated.

iii) In Section 4.4.2, it was stated that during the restart phase a retry is attempted when the integration is unsatisfactory. If more than 5 retries are performed during a restart phase, the integration is terminated.

iv) The integration is also terminated if the matrix G appears to be singular.

In all the above cases, it is likely that the problem has been incorrectly specified or a discontinuity in the solution has not been located correctly, it is necessary to check the problem specification carefully.

## E.2 The Detailed Description of ROSEN

The coding of integrator ROSEN together with a full description of the parameters used are given at the end of this appendix. In integrator ROSEN, it is assumed that the algebraic equations are arranged after the differential equations in the DAE system. Hence equations (NODE+1) to NEQN are algebraic.

Most of the parameters used in ROSEN are straightforward and need no further explanation. The key parameters are IND, RSTART and DIRK. They are explained below.

The parameter IND is a vector which serves as indicators for ROSEN. It consists of 4 components. The first two components, the IND(1) and IND(2), are used by ROSEN to communicate with the calling program. The typical use of these two components is given in fig. (E.1). The IND(2) as well as IND(3) are used as internal indicators within ROSEN. Lastly, since ROSEN includes both the Rosenbrock-type method (D.3.2) and the implicit DIRK method (D.4.1), the IND(4) is used to select the required method to be used for the integration. It is set to 1 when method (D.3.2) is to be used and to

2 when the implicit method is required. This provides two distinct methods within the same integrator.

In order to distinguish between the restart and normal phases, a logical parameter RSTART is used. On entry, RSTART is set to TRUE whenever a restart phase is required. This usually happens when a discontinuity has been detected. It is reset to FALSE within the integrator when the restart phase is to be terminated.

DIRK is another logical parameter which is to be used in conjunction with the Rosenbrock-type method, i.e. when  $IND(4)=1$ . It is set to TRUE within the integrator to force the application of the implicit method when the non-linear algebraic equations in the DAE system have not converged satisfactorily (see Section 4.4.1.2). It can also be set by the user when the solution in the next step is known to be changing rapidly. It is reset to FALSE within the integrator at the end of the step.

The integrator is divided into 7 stages. Stage 1 sets up the parameters at the beginning of each step of integration. It first checks for the value of  $IND(2)$  and branches to the appropriate points in the integrator if  $IND(2)>0$ . It then checks whether the error tolerance is sufficiently large for the machine and initiates a restart phase if the parameter RSTART has just been set to TRUE; a new stepsize to be used for the restart phase is computed using the formula given in eqn. (4.4.3).

Stage 2 computes the solution using the Rosenbrock-type method (D.3.2). The solution computed is improved using the implicit method (D.4.1) in stage 3 if the parameter DIRK is set to TRUE; the strategies outlined in section 4.4.1.1 are used to implement the modified-Newton method for solving the nonlinear systems of equations.

The reduction of stepsize, the updating of iteration matrix and the retry for the restart phase are performed in stage 4; the integration is terminated with an error indication if any of the error conditions described in section E.1 is violated.

Stage 5 estimates the local error which is used during the normal phase for the error test; the step is rejected and the stepsize is halved if the error estimate exceeds the required error tolerance. The global error is also estimated in this stage if RSTART is set to TRUE.

Stage 6 checks for the convergence of the algebraic equations in the DAE system when the Rosenbrock-type method is used; the criterion given in eqn.(4.4.1) is used for the convergence test. The step is repeated using the implicit method (D.4.1) if any of the algebraic equations fails the convergence test.

Finally in stage 7, the appropriate parameters are updated after the solution is accepted. During the restart phase, checks are made to ensure that the local error in successive steps of solution is decreasing in magnitude and to determine whether the restart phase is to be terminated; the normal phase is recommenced when the global error estimate is less than the tolerance. During the normal phase, the new stepsize to be used for the next step is also estimated before returning to the calling program.

### E.3 The Scope and Use of Integrator ROSEN

Before any discussion on the use of integrator ROSEN can be carried out, it is necessary to state the applicability of this integrator in solving general non-autonomous systems. From section 4.3.3, it is clear that the Rosenbrock-type method and its associated DIRK method can be extended to the non-autonomous case in the way as suggested in that section. However, the usual local error estimate will not be as accurate as in the autonomous case, and it is only reliable when  $\| \partial \underline{f} / \partial t \| \ll \| \partial \underline{f} / \partial \underline{y} \|$  for the problem to be solved. The use of a less accurate error estimate will affect the performance of the integrator considerably. Thus in general, it is advisable to transform the problem to be solved into autonomous form before using the integrator.

As the integrator is written as a low level routine, a rather sophisticated driver program is needed to use the integrator. A typical layout of the driver program is given in fig. (E.2).

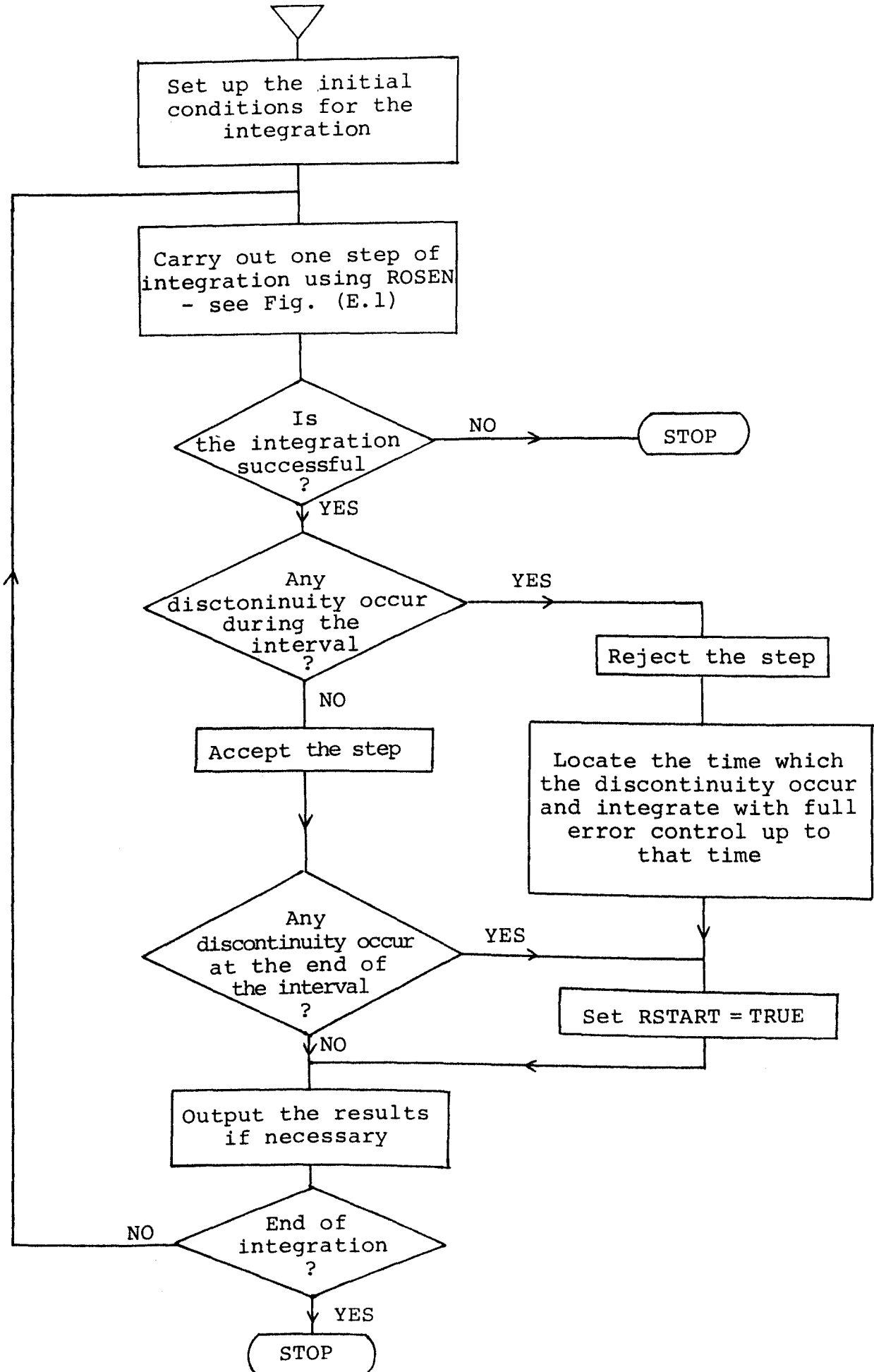
After each successful step, a check is made to see if a discontinuity is to occur during or at the end of the step. This is done by checking through the list of pre-determined times when the discontinuities are known to occur or computing the constraint function  $\phi$ 's which change sign over the discontinuity. In the later case, it is necessary to locate the discontinuity using the procedure outlined in section 4.4.2.1 and compute an accurate solution up to that time before a restart phase is initiated. A restart phase is initiated by setting the indicator RSTART to TRUE.

Because the integrator is written in "reverse communication" form, there is no restriction on the number or type of routines that must be supplied by the user in order to specify the problem. The users are therefore free to write their own routines for this purpose. The following routines, however, are recommended, although the exact number of routines required is dependent on the nature of the problem involved:

- i) A routine to estimate the iteration matrix G;
- ii) routines to factorize and solve the matrix G and
- iii) a routine to compute  $E \frac{dy}{dt}$  and  $\underline{f}(t, \underline{y})$  of eqn. (1.3.1).

Because these quantities are usually required separately by the integrator, either separate routines should be used or a flag should be incorporated in the routine to indicate which of these quantities is required. The routine for computing  $\underline{f}(t, \underline{y})$  must also include an option to compute only the algebraic equations in the system which are required for the convergence test at the end of each step.

Fig. (E.2) - The Layout of the Driver Program for ROSEN





E.4 - The Code of Integrator ROSEN

```

C=====
C
C THE INTEGRATOR ROSEN - WRITTEN IN REVERSE COMMUNICATION
C
C=====
C      SUBROUTINE ROSEN(NEQN,NODE,T,H,Y,DY,DEL,DYY,IND,WT,W,R,EPS,
C      *
C      ZNORM,RSTART,DIRK)
C*****
C
C THIS ROUTINE SOLVES THE SYSTEM OF DIFFERENTIAL/ALGEBRAIC
C EQUATIONS OF THE FORM
C      E Y' = F(Y) (1)
C USING THE ROSENBRACK-TYPE METHOD, WHERE E IS A SINGULAR MATRIX
C WHEN THERE ARE ALGEBRAIC EQUATIONS PRESENT IN THE SYSTEM.
C THE ITERATION MATRIX IS GIVEN BY:
C      G = E - H*ALPHA*DF/DY
C
C ***** NOTE *****
C ALTHOUGH THE INTERATOR CAN IN PRINCIPLE BE USED FOR SOLVING ANY
C NON-AUTONOMOUS SYSTEMS, IT IS ADVISABLE TO TRANSFORM THE SYSTEM
C TO BE SOLVED INTO AUTONOMOUS FORM BEFORE USING THE INTEGRATOR.
C
C
C THE VARIABLES USED HAVE THE FOLLOWING MEANINGS:
C *NEQN--NUMBER OF EQUATIONS TO BE SOLVED.
C *NODE--NUMBER OF ODE'S IN THE SYSTEM; THEY MUST BE THE FIRST NODE
C EQUATIONS IN THE SYSTEM.
C *T --THE INDEPENDENT VARIABLE. ON FIRST CALL IT SHOULD BE SET
C TO THE INITIAL CONDITION. ON RETURN IT CONTAINS THE VALUE
C OF T FOR WHICH Y IS THE SOLUTION.
C *H --PROPOSED STEPSIZE FOR THE STEP. ON FIRST CALL, IT SHOULD
C CONTAIN THE INITIAL ESTIMATE OF THE STEPSIZE .
C *Y --THE DEPENDENT VARIABLE. ON FIRST CALL SHOULD BE SET TO
C THE INITIAL CONDITIONS. ON RETURN IT CONTAINS THE SOLUTION
C AT T. DIMENSIONED AS Y(NEQN).
C *DY --AN ARRAY USED TO HOLD THE RIGHT-HAND FUNCTION F(Y,T).
C ON FIRST ENTRY,IT SHOULD CONTAIN THE INITIAL F(Y,T) VALUES.
C DIMENSIONED AS DY(NEQN).
C *DEL --RETURNS THE NEW ITERATION CORRECTIONS OR THE RESULTS OF A
C BACK-SUBSTITUTION.
C DIMENSIONED AS DEL(NEQN).
C *DYY --WHEN IND(2)=3, IT CONTAINS ON EXIT PART OF RIGHT-HAND VECTOR
C FOR WHICH A BACK-SUBSTITUTION IS TO BE CARRIED OUT
C (SEE W(7,J)). DIMENSIONED AS DYY(NEQN).
C *IND --ARRAY CONTAINING THE INDICATORS FOR THE METHOD.
C DIMENSIONED AS IND(4).
C WHERE:
C IND(1)-INDICATOR FOR THE INTEGRATOR
C ON EXIT >0 FOR SUCCESSFUL CALL
C <0 FOR STEP FAILURE.
C ON ENTRY, IT HAS THE FOLLOWING MEANINGS:
C =0 INITIAL STEP.
C =2 TO RE-EVALUATE THE ITERATION MATRIX.
C ON EXIT, IT HAS THE FOLLOWING MEANINGS:

```

C           = 1 SUCCESSFUL STEP.  
 C           = 2 SUCCESSFUL STEP BUT TO REEVALUATE G ON NEXT STEP.  
 C           = 3 SAME AS 1 EXCEPT F(Y,T) AT THE NEW TIME LEVEL HAS BEEN  
 C           FOUND AND IS CONTAINED IN VECTOR DY.  
 C           = 4 SAME AS 3 BUT A NEW G IS ALSO NEEDED.  
 C           =-1 STEP FAILURE BECAUSE THE TOLERANCE WAS TOO SMALL  
 C           FOR THE MACHINE.  
 C           =-2 STEP FAILED BECAUSE STEPSIZE BECAME TOO SMALL FOR THE  
 C           MACHINE.  
 C           =-3 ERROR TOLERANCE WAS NOT SATISFIED AFTER 3 STEP  
 C           REDUCTIONS.  
 C           =-4 STEP FAILURE AFTER 5 RETRIES DURING THE RESTART PHASE.  
 C IND(2)-POINTER TO INFORM THE CALLING PROGRAM ABOUT THE TYPE OF  
 C           PROBLEM\_DEPENDENT INFORMATION NEEDED BY THE INTEGRATOR.  
 C           ON EXIT.....  
 C           = 0 INTEGRATION HAS FINISHED, CHECK IND(1) FOR POSSIBLE  
 C           ERROR.  
 C           = 1 A NEW ITERATION MATRIX IS REQUIRED.  
 C           THE INFORMATION NEEDED ARE STORED IN VECTOR Y & DY.  
 C           = 2 A NEW RIGHT-HAND FUNCTION F(Y,T) IN EQN.(1) IS NEEDED.  
 C           RETURN THE REQUIRED FUNCTION VALUES IN VECTOR DY.  
 C           = 3 TO COMPUTE THE SOLUTION OF  $G \cdot \Delta = R$ ,  
 C           WHERE  $R = DYY + E * W(7,J)$   
 C           = 4 EVALUATE A NEW ITERATION CORRECTION  
 C            $CORR = G ** (-1) * (E * Y' - F(Y,T))$ ,  
 C           AND RETURN THE RESULTS IN VECTOR DEL.  
 C           = 5 COMPUTE THE ALGEBRAIC EQUATIONS REQUIRED IN THE  
 C           CONVERGENCE TEST AND RETURN THE RESULTS IN VECTOR DEL  
 C           (IN LOCATIONS NODE+1 TO NEQN).  
 C IND(3)-INTERNAL INDICATOR TO INFORM THE INTEGRATOR THE EXACT POINT  
 C           TO RETURN TO.  
 C IND(4)-INDICATE THE METHOD TO BE USED:  
 C           =1 FOR THE ROSENBROCK METHOD AND  
 C           =2 FOR DIRK METHOD WITH THE ROSENBROCK METHOD AS A  
 C           PREDICTOR.  
 C \*WT   --VECTOR OF WEIGHTS FOR ERROR CRITERION.  
 C           DIMENSIONED AS WT(NEQN)  
 C \*W    --ARRAY USED AS WORKSPACE. DIMENSIONED AS W(10,NEQN).  
 C           WHERE:  
 C           W(1,J) HOLDS THE VALUES OF Y AT PREVIOUS TIME LEVEL. IE.Y(N).  
 C           W(2,J) HOLDS THE VALUES OF F(YN).  
 C           W(3,J) HOLDS K1.  
 C           W(4,J) CONTAINS THE VALUES OF K2.  
 C           W(5,J) WORKSPACE FOR HOLDING F(Y\*).  
 C           W(6,J) CONTAINS ON EXIT THE ESTIMATES OF LOCAL ERROR.  
 C           W(7,J) WORKSPACE FOR HOLDING PART OF THE RIGHT-HAND VECTOR  
 C           TO BE PRE-MULTIPLIED BY MATRIX E FOR WHICH A BACK-  
 C           SUBSTITUTION IS REQUIRED. THE OTHER PART IS CONTAINED  
 C           IN VECTOR DYY. SEE DESCRIPTION FOR IND(2)=3.  
 C           W(8,J) HOLDS THE ESTIMATES OF THE GLOBAL ERROR.  
 C           W(9,J) STORE THE Y(J) VALUES AT THE TIME WHEN A RESTART  
 C           PHASE IS INITIATED; THIS IS REQUIRED IN CASE A RETRY  
 C           IS NEEDED.  
 C           W(10,J) STORE THE CORRESPONDING DY VALUES.  
 C \*R    --ARRAY USED AS WORKSPACE, DIMENSIONED AS R(8).  
 C           R(1) -HOLDS THE NORM (ITERATION CORRECTION) FROM THE  
 C           PREVIOUS ITERATION.  
 C           R(2) -HOLDS THE RATE OF CONVERGENCE OF THE ITERATION.

```

C      R(3) -ITERATION TEST CONDITION (=EPS).
C      R(4) -HOLDS THE FACTOR OF THE STEPSIZE CHANGES SINCE THE
C            LAST G UPDATING, IF R(4)<0.5 THEN A NEW J IS REQD.
C      R(5) -HOLDS THE OLD STEPSIZE.
C      R(6) -HOLDS THE OLD VALUE OF T AT THE PREVIOUS TIME LEVEL.
C            THIS IS REQUIRED FOR RE-SETTING T WHEN THERE IS A
C            CHAGNE IN STEPSIZE H.
C      R(7) -MAXIMUM ALLOWABLE STEPSIZE HMAX.
C      R(8) -HOLDS THE TIME AT WHICH A RESTART PHASE IS INITIATED.
C *EPS --THE LOCAL ERROR TOLERANCE SPECIFIED BY THE USER.
C *ZNORM-ROUTINE TO COMPUTE THE ERROR NORM. IT SHOULD BE WRITTEN AS
C      SUBROUTINE ZNORM(NEQN,ERR,WT,E)
C      DOUBLE PRECISION ERR,WT(NEQN),E(NEQN)
C      WHERE E IS THE ERROR VECTOR, WT IS THE WEIGHTS AND ERR
C      RETURNS THE ERROR NORM.
C *RSTART-LOGICAL PARAMETER WHICH WHEN SET TO TRUE MEANS THAT THE
C      RESTART PHASE IS ENFORCED.
C *DIRK -LOGICAL PARAMETER. IT IS SET TO TRUE WHEN DIRK METHOD IS
C      TO BE USED. THE DIRK METHOD IS ALSO USED WHEN IND(4)=1
C      (SEE IND(4) FOR DETAILS).
C
C=====
C
C NOTE--WHEN THE ROUTINE IS FIRST CALLED,
C      IND(1), IND(2) MUST BE SET TO -1 AND 0 RESPECTIVELY.
C
C*****
C      EXTERNAL ZNORM
C      LOGICAL RSTART,DIRK,CONVRG
C      DOUBLE PRECISION T,H,EPS,CORC,RESCN,RESCR,FJAC,ALPHA,COEF,ERRL,
C      *      ERRG,TWOU,FOURU,TENP,ONE,P1EPS,ROUND,TEMP,
C      *      RNORM,FAC,RCON,F1,F2,ERRO,RATIO,HMAX,SUM,
C      *      RATIO,CMIN,ZERO
C      DOUBLE PRECISION Y(NEQN),DY(NEQN),DEL(NEQN),DYY(NEQN),R(8),
C      *      W(10,NEQN),WT(NEQN)
C
C      INTEGER IND(4)
C      COMMON /STATS/ ISTAT(6),IJAB,NHALF,NRETRY,ITER
C      COMMON /STRAT/ JSTEP(2),CMIN,CORC,RESCN,RESCR,FJAC,ALPHA,COEF(6)
C      COMMON /RSTAT/ ERRL,ERRG,ERRO,RATIO,HMAX,NRSTEP,NLESS
C      COMMON /MACON/ TWOU,FOURU,TENP
C      COMMON /TRACE/ IDEVO,ITRACE
C      COMMON /ZZZ1/ NORM
C*****
C
C THE COMMON BLOCKS USED ARE:
C /STATS/-CONTAINS THE STATISTICS REQUIRED AT THE END OF THE SOLUTION
C *ISTAT(6)-VECTOR HOLDING THE OVERALL STATISTICS REQUIRED, WHERE
C .. (1)-NUMBER OF STEPS ATTEMPTED
C .. (2)-NUMBER OF FUNCTIONS CALL
C .. (3)-NUMBER OF ITERATION MATRIX UPDATINGS.
C .. (4)-NUMBER OF BACK-SUBSTITUTIONS PERFORMED
C .. (5)-NUMBER OF STEPS FAILED TO REACH ERROR CRITERION
C .. (6)-NUMBER OF STEPS FAILURE DUE TO POOR CONVERGENCE.
C *IJAB -CONTAINS THE NO. OF ITERATION MATRIX UPDATINDS IN A
C      STEP. A MAX OF 3 EVALUATIONS IS ALLOWED BEFORE THE
C      STEPSIZE IS REDUCED.
C *NHALF -COUNTS THE NO. OF STEP REDUCTIONS IN A STEP. ONLY
C      3 STEP REDUCTIONS IS ALLOWED.

```

```

C      *NRETRY-NUMBER OF RETRY DURING A RESTART PHASE. A MAX OF 5
C      RETRIES IS ALLOWED.
C      *ITER -NO. OF ITERATIONS FOR THE MODIFIED-NEWTON ITERATION.
C /STRAT/-DEFINES THE STRATEGY PARAMETERS USED IN THE INTEGRATOR;
C      THEIR VALUES ARE INITIALISED IN BLOCK DATA.
C      *JSTEP -DIMENSIONED AS JSTEP(2). CONTAINS THE INFORMATION ON
C      THE NUMBER OF STEPS TAKEN SINCE THE LAST ITERATION
C      MATRIX UPDATING AND STEPSIZE CHANGES.
C      '' (1)-HOLDS THE NUMBER OF STEPS TAKEN SINCE THE LAST G
C      MATRIX UPDATING. A MAX OF 15 STEPS IS ALLOWED BEFORE
C      G IS TO BE UP-DATED.
C      '' (2)-HOLDS THE NUMBER OF STEPS TO BE USED BEFORE STEPSIZE
C      CHANGES IS CONSIDERED. A MIN OF 3 STEPS IS NEEDED
C      SINCE THE LAST CHANGE IN STEPSIZE.
C      *CMIN -CONSTANT TO DECIDE WHETHER THE ITERATION CAN
C      CONTINUE.
C      *CORC -CONSTANT TO DECIDE WHETHER THE RATE OF CONVERGENCE
C      IS SATISFACTORY (CORC=0.5).
C      *RESCN -FACTOR USED FOR PREDICTING THE INITIAL STEPSIZE FOR
C      THE NORMAL VARIABLE-STEP INTEGRATION.
C      *RESCR -THE CORRESPONDING FACTOR USED FOR THE RESTART PHASE.
C      *FJAC -FACTOR WHICH THE STEPSIZE CAN BE VARIED BEFORE THE
C      ITERATION MATRIX IS CONSIDERED AS OUT-DATED.
C      *ALPHA -PARAMETER OF THE ROSENBROCK-TYPE METHOD.
C      *COEF(6)-HOLDING THE COEFICIENTS DEFINING THE METHOD.
C /RSTAT/-DEFINES THE PARAMETERS USED IN THE RESTART PHASE.
C      *ERRL -HOLDS THE WEIGHTED LOCAL ERROR NORM.
C      *ERRG -CONTAINS THE WEIGHTED GLOBAL ERROR NORM.
C      *ERRO -HOLDS THE LOCAL ERROR NORM IN THE PREVIOUS STEP.
C      *RATIOP-HOLDS THE RATIO BETWEEN ERRO AND THE LOCAL ERROR
C      NORM AT THE STEP BEFORE.
C      *HMAX -THE MAX. ALLOWABLE STEPSIZE FOR THE RESTART PHASE.
C      *NRSTEP-NUMBER OF STEPS TAKEN DURING THE RESTART PHASE.
C      *NLESS -NUMBER OF STEPS WHERE WHERE ERRG IS LESS THAN EPS.
C /MACON/-HOLDS THE MACHINE DEPENDENT CONSTANTS - THE MACHINE
C      ROUND OFF AND THE MACHINE UNDERFLOW NUMBERS U AND P.
C      *TWO U = 2*U
C      *FOUR U = 4*U
C      *TEN P = 10*P
C /TRACE/-TO TRACE THE INTEGRATOR FOR DEBUGGING PURPOSES:
C      *IDEVO -OUTPUT DEVICE NUMBER.
C      *ITRACE-TRACE LEVEL REQUIRED.
C      =0 FOR NO TRACE
C      =1 TO OUTPUT INTERMEDIATE RESULTS AT APPROPRIATE
C      POINTS.
C /ZZZZ1/-DEFINES THE TYPE OF NORM TO BE COMPUTED, SEE ROUTINE ZNORM.
C
C *****
C      ONE=1.0D+0
C      ZERO=0.0D+0
C
C      SET UP THE STRATEGY PARAMETERS FOR THE INITIAL STEP
C
C      IF (IND(1).GE.0) GO TO 10
C      ALPHA=ONE - ONE/DSQRT(2.0D+0)
C      COEF(1)=-4.0D+0/3.0D+0*ALPHA
C      COEF(2)=2.0D+0/3.0D+0
C      COEF(3)=0.25D+0

```

```

COEF(4)=0.75D+0
COEF(5)=(ALPHA+ONE)/(ALPHA - ONE/3.0D+0)
COEF(6)=(0.5D+0 - ALPHA)/(ALPHA*ALPHA)
R(3)=EPS
J1=NODE+1
DO 1000 J=J1,NEQN
1000   W(6,J)=0.0D+0
      IF (IND(4).EQ.2) DIRK=.TRUE.
10    CONTINUE
C
C*****
C
C  STAGE--1 SETTING UP
C
C*****
C
C  IF IND(2)#0 JUMP TO THE APPROPRIATE POINTS IN THE INTEGRATOR
C  ELSE, IT IS A FRESH CALL FOR THE PRESENT TIME LEVEL
C
      K=IND(3)
      I=IND(2)+1
      IND(2)=0
      GO TO (12,20,17,29,35,63), I
12    CONTINUE
C
C  CHECK IF EPS IS SUFFICIENTLY LARGE
C
      P1EPS=0.1D+0*EPS
      CALL ZNORM(NEQN,ROUND,WT,Y)
      ROUND=ROUND*TWO
      IF (P1EPS .GE. ROUND) GO TO 15
C
C  ERROR - THE TOLERANCE IS TOO SMALL FOR THE MACHINE
C
      EPS=10.0D+0*(ROUND*(ONE+FOURU) + TENP)
      IND(1)=-1
      RETURN
15    CONTINUE
      IJAB=0
      NHALF=0
      IND(3)=0
      K=0
      IF (IND(1).LT.0 .OR. IND(1).GE.3) GO TO 17
      ISTAT(2)=ISTAT(2)+1
      IND(2)=2
      RETURN
17    CONTINUE
      IF (IND(1).GE.3) IND(1)=IND(1)-2
      IF (K.EQ.2) GO TO 27
C
C  INITIALISE A RESTART PHASE IF NECESSARY
C
      IF (.NOT.RSTART .OR. NRSTEP.GT.0) GO TO 19
C
C  STORE THE CURRENT SOLUTION VALUES IN W WORKSPACE,
C  AND THE CURRENT TIME IN R(8)
C
      DO 1005 J=1,NEQN

```

```

      W(9,J)=Y(J)
1005      W(10,J)=DY(J)
      ERRG=ZERO
      R(8)=T
C
C      ESTIMATE THE STEPSIZE TO BE USED FOR THE RESTART PHASE
C
      CALL ZNORM(NEQN,SUM,WT,DY)
      H=RESCR*DSQRT(EPS/SUM)
      H=DMIN1(HMAX,H)
      IF (ITRACE.NE.2) WRITE(IDEVO,8003) H
8003      FORMAT(//' === RESTART PHASE === '/' H = ',D12.4)
      IND(1)=2
      DIRK=.TRUE.
19      CONTINUE
C
C      STORE Y & DY IN W(1,J) & W(2,J) RESPECTIVELY
C
      DO 1010 J=1,NEQN
1010      W(1,J)=Y(J)
          W(2,J)=DY(J)
C
C      COMPUTE THE ITERATION MATRIX IF IT IS AN INITIAL STEP
C
      IF (IND(1).LE.0) IND(1)=2
C*****
C      STAGE--2 COMPUTE THE SOLUTION USING ROSENBROCK-TYPE METHOD
C*****
C
C      PRESERVE THE OLD VALUE OF T IN R(6)
C
      R(6)=T
      IF (IND(1).EQ.2) GO TO 45
20      CONTINUE
C
C      CHECK IF THE STEPSIZE IS LARGE ENOUGH FOR THE MACHINE
C
      IF(DABS(H).GE.FOURU*DABS(T)+TENP) GO TO 22
C
C      ERROR - THE STEPSIZE HAS BECOME TOO SMALL FOR THE MACHINE
C
      IND(1)=-2
      RETURN
22      CONTINUE
      IF (K.GT.0) GO TO 33
      IND(3)=1
C
C      COMPUTE K1
C
      DO 1020 J=1,NEQN
1020      DYY(J)=W(2,J)
          W(7,J)=ZERO
C
C      EXIT TO COMPUTE K1; IE. THE BACK-SUBSTITUTION
C      RETURN TO LABEL 29
C

```

```

      ISTAT(4)=ISTAT(4)+1
      IND(2)=3
      RETURN
25    CONTINUE
C
C    COMPUTE K2 IN 2 STAGES
C    1 - EVALUATE FUNCTION F(YN + H*A21*K1)
C    2 - EVALUATE K2 BY BACK-SUBSTITUTION.
C
      IND(3)=2
      IJAB=0
      T=R(6) + COEF(2)*H
      DO 1030 J=1,NEQN
          Y(J)=W(1,J) + H*COEF(2)*W(3,J)
1030  CONTINUE
      ISTAT(2)=ISTAT(2)+1
      IND(2)=2
      RETURN
C
C    RETURN TO LABEL 27 WITH THE REQUIRED FUNCTION IN VECTOR DY
C
27    CONTINUE
      DO 1040 J=1,NEQN
          W(5,J)=DY(J)
          W(7,J)=COEF(1)/ALPHA*W(3,J)
          DYY(J)=DY(J)
1040  CONTINUE
      ISTAT(4)=ISTAT(4)+1
      IND(2)=3
      RETURN
29    CONTINUE
C
C    RETURN HERE WITH K VALUES IN DEL(J) FOR K=1, 2
C
      GO TO (30,30,55,56,57), K
30    CONTINUE
      DO 1050 J=1,NEQN
1050  W(K+2,J)=DEL(J) - W(7,J)
C
C    IMPROVE THE SOLUTION USING THE CORRESPONDING DIRK METHOD
C    IF THE PARAMETER DIRK IS SET.
C
      IF (DIRK) GO TO 31
      IF (K.EQ.2) GO TO 50
      GO TO 25
C
C*****
C
C    STAGE--3 MODIFIED-NEWTON ITERATION FOR THE DIRK METHOD
C*****
31    CONTINUE
      IF (K.EQ.1) ITER=0
      R(1)=0.0D+0
C
C    SET UP THE INITIAL ESTIMATES FOR Yi & Ki VECTORS
C
      IF (K.EQ.2) GO TO 32

```

```

F1=H*ALPHA
T=R(6) + F1
DO 1060 J=1,NEQN
  DY(J)=W(3,J)
  Y (J)=W(1,J) + F1*DY(J)
1060 GO TO 33
32 CONTINUE
F1=H*(COEF(1) + COEF(2))
F2=H*ALPHA
T=R(6) + F1 + F2
DO 1061 J=1,NEQN
  DY(J)=W(4,J)
  Y (J)=W(1,J) + F1*W(3,J) + F2*DY(J)
1061 CONTINUE
33 C
C EXIT TO COMPUTE THE ITERATION CORRECTION
C
  ISTAT(2)=ISTAT(2) + 1
  ISTAT(4)=ISTAT(4) + 1
  IND(2)=4
  RETURN
35 CONTINUE
C
C RETURN HERE WITH THE ITERATION CORRECTION IN VECTOR DEL
C
  ITER=ITER+1
  CALL ZNORM(NEQN,RNORM,WT,DEL)
  RNORM=H*RNORM
  RCON=0.0D+0
  IF (R(1).NE.0.0D+0) RCON=RNORM/R(1)
  R(1)=RNORM
  R(2)=RCON
  IF (ITRACE.EQ.1) WRITE(IDEVO,8040) ITER,RNORM,RCON,K
8040 FORMAT(/' ITER=' ,I2,' RNORM=' ,D12.4,' RCON=' ,D12.4,' K=' ,I2)
  IF (RCON.GE.ONE) GO TO 36
  F1=H*ALPHA
  DO 1062 J=1,NEQN
    DY(J)=DY(J) - DEL(J)
    Y (J)=Y (J) - F1*DEL(J)
1062 CONTINUE
  IF (RNORM.LE.R(3)) GO TO 39
  IF (RCON.LE.CORC) GO TO 33
C
C DURING RESTART PHASE, A MIN OF 5 ITERATIONS IS PERFORMED BEFORE
C THE ITERATION MATRIX UPDATING IS CONSIDERED
C
  IF (RSTART.AND.ITER.LE.5) GO TO 33
36 CONTINUE
C
C A NEW ITERATION MATRIX IS REQUIRED
C
  GO TO 45
39 CONTINUE
C
C ITERATION HAS CONVERGED
C
  DO 1063 J=1,NEQN
    W(K+2,J)=DY(J)

```



```

1063  CONTINUE
      IF (K.EQ.1) GO TO 25
      GO TO 50
C*****
C
C  STAGE--4 STEPSIZE REDUCTION, ITERATION MATRIX UPDATING
C      AND RETRY FOR THE RESTART PHASE
C*****
C
C      IF THE ITERATION CORRECTION IS UNSATISFACTORY OR IF THE
C      STEPSIZE HAS BEEN CHANGED BY MORE THAN A FACTOR OF FJAC
C      RE-EVALUATE THE ITERATION MATRIX
C
C 40  CONTINUE
C
C      RESET Y & DY VECTOR
C
C      IND(3)=0
C      DO 1065 J=1,NEQN
C          DY(J)=W(2,J)
C          Y(J) =W(1,J)
1065  CONTINUE
C
C      IF THE ITERATION MATRIX HAS NOT BEEN UPDATED IN THE LAST TWO
C      STEPS UPDATE IT BEFORE REDUCING THE STEPSIZE
C
C      IF (JSTEP(1).GE.2) GO TO 45
C
C      REDUCE THE STEPSIZE BY HALF
C
C      JSTEP(2)=3
C      H=H*0.5D+0
C      NHALF=NHALF+1
C      IF (NHALF.LE.3) GO TO 45
C
C      TOO MANY STEP REDUCTIONS
C
C      IND(1)=-3
C      RETURN
45  CONTINUE
C
C      NEW ITERATION MATRIX IS REQUIRED
C
C      IND(1)=1
C      IJAB=IJAB+1
C      IF (IJAB.LE.3) GO TO 47
C      IF (RSTART) GO TO 48
C
C      TOO MANY EVALUATIONS OF MATRIX G, REDUCE THE STEPSIZE
C
C      IJAB=0
C      GO TO 40
47  CONTINUE
      ISTAT(3)=ISTAT(3)+1
      JSTEP(1)=0
      R(4)=ONE
      ITER=0

```

```

R(1)=0.0D+0
R(2)=0.0D+0
IND(2)=1
RETURN
48 CONTINUE
C
C THE ITEGRATION IS UNSATISFACTORY DURING THE RESTART PHASE,
C REDUCE THE STEPSIZE BY A FACTOR OF 5 AND RESTART FROM THE
C INITIAL TIME (RETRY).
C
NRETRY=NRETRY+1
IF (NRETRY.LE.5) GO TO 49
C
C ERROR - MORE THAN 5 RETRIES HAS BEEN DONE
C
IND(1)=-4
RETURN
49 CONTINUE
H=0.2D+0*H
T=R(8)
IF (ITRACE.NE.2) WRITE(IDEVO,8111) T
8111 FORMAT(/' RE-START FROM T =',D15.6)
DO 1067 J=1,NEQN
    Y(J) =W(9,J)
    DY(J)=W(10,J)
1067    WT(J)=DMAX1(ONE, DABS(Y(J)) )
ERRG=ZERO
IND(1)=4
DIRK=.TRUE.
NRSTEP=1
GO TO 15
C*****
C
C STAGE--5 ERROR ESTIMATION
C*****
50 CONTINUE
C
C COMPUTE THE NEW SOLUTION AND ESTIMATE THE LOCAL ERROR
C
DO 1070 J=1,NEQN
    Y(J)=W(1,J) + H*(COEF(3)*W(3,J) + COEF(4)*W(4,J))
    DYY(J)= -W(2,J) - 3.0D+0*W(5,J)
    W(7,J)= W(3,J) + 3.0D+0*W(4,J)
1070 CONTINUE
C
C RETURN TO COMPUTE G**(-1)*DEL
C
IND(3)=3
52 CONTINUE
ISTAT(4)=ISTAT(4)+1
IND(2)=3
RETURN
55 CONTINUE
DO 1075 J=1,NODE
    DEL(J)=H*DEL(J)/6.0D+0
1075 W(6,J)=DEL(J)
CALL ZNORM(NODE,ERRL,WT,DEL)

```

```

ERRL=ERRL/EPS
IF (ITRACE.EQ.1) WRITE(IDEVO,9035) ERRL
9035 FORMAT(' ERRL =',D15.6)
IF (ERRL.LE.ONE .AND. .NOT.RSTART) GO TO 60
IF (.NOT.RSTART) GO TO 59
C
C DURING THE RESTART PHASE -- COMPUTE THE GLOBAL ERROR ESTIMATE
C
C IF (ERRG.NE.ZERO) GO TO 53
C
C FIRST STEP DURING THE RESTART PHASE, ERRG=ERRL
C
DO 1078 J=1,NEQN
1078 W(8,J)=W(6,J)
ERRG=ERRL
GO TO 60
53 CONTINUE
DO 1080 J=1,NEQN
1080 DYY(J)=0.0D+0
W(7,J)=W(8,J)
IND(3)=4
GO TO 52
56 CONTINUE
DO 1082 J=1,NEQN
W(7,J)=COEF(6)*DEL(J) + (ONE-COEF(6))*W(8,J)
DYY(J)=0.0D+0
1082 CONTINUE
IND(3)=5
GO TO 52
57 CONTINUE
DO 1086 J=1,NODE
1086 DEL(J)=DEL(J) + W(6,J)
W(8,J)=DEL(J)
CALL ZNORM(NODE,ERRG,WT,DEL)
ERRG=ERRG/EPS
IF (ITRACE.EQ.1) WRITE(IDEVO,9037) ERRG
9037 FORMAT(' ERRG =',D15.6)
GO TO 60
59 CONTINUE
C
C LOCAL ERROR DOES NOT SATISFY THE ERROR TOLERANCE
C REDUCE THE STEPSIZE BY HALF
C
ISTAT(5)=ISTAT(5)+1
GO TO 40
C*****
C
C STAGE--6 THE CONVERGENCE TEST
C
C*****
60 CONTINUE
IF (NODE.EQ.NEQN .OR. DIRK) GO TO 70
IND(2)=5
RETURN
63 CONTINUE
C
C RETURN HERE WITH THE VALUES OF THE ALGEBRAIC EQUATIONS IN
C VECTOR DEL

```

```

C
CONVRG=.TRUE.
I1=NODE+1
P1EPS=0.2D+0*EPS
DO 1100 I=I1,NEQN
  IF (DABS(DEL(I)/WT(I)).LE.P1EPS) GO TO 1100
  CONVRG=.FALSE.
  GO TO 65
1100 CONTINUE
65 IF (CONVRG) GO TO 70
C
C NONLINEAR ALGEBRAIC EQUATIONS HAVE NOT CONVERGED
C
WRITE(IDEVO,9050)
9050 FORMAT(/' NONLINEAR ALGEBRAIC EQUATIONS HAVE NOT CONVERGED')
C
C RESOLVE THE SOLUTION USING DIRK METHOD
C
DO 1110 I=1,NEQN
  Y(I) =W(1,I)
1110  DY(I)=W(2,I)
  T=R(6)
  IND(1)=3
  IND(2)=0
  DIRK=.TRUE.
  GO TO 15
C*****
C
C STAGE--7 SUCCESSFUL STEP
C
C*****
70 CONTINUE
R(5)=H
T=R(6)+H
ISTAT(1)=ISTAT(1)+1
JSTEP(1)=JSTEP(1)+1
JSTEP(2)=JSTEP(2)-1
C
C SET DIRK TO FALSE IF THE ROSENBRACK-TYPE METHOD IS USED
C
IF (IND(4).EQ.1) DIRK=.FALSE.
FAC=ONE
IF (.NOT.RSTART) GO TO 75
IF (.NOT.DIRK .AND. JSTEP(1).GE.5) IND(1)=2
IF (NRSTEP.EQ.1 .AND. ERRL.LT.0.75D+0 .AND. ERRO.LT.0.75D+0)
* GO TO 72
C
C CHECK IF THE STEPSIZE USED IS SENSIBLE - GO TO LABEL 48 IF THE
C LOCAL ERROR DIVERGES OR TOO MANY STEPS HAVE BE COMPUTED
C
IF (ERRG.GT.100.0) GO TO 48
RATIO=ZERO
IF (NRSTEP.GT.0) RATIO=ERRL/ERRO
IF (RATIO.GT.ONE .AND. RATIO.P.GT.ONE .AND. ERRL.GT.ONE) GOTO 48
RATIOP=RATIO
ERRO=ERRL
NRSTEP=NRSTEP+1
NLESS=NLESS+1

```

```

IF (ERRG.GT.ONE) NLESS=0
IF (NLESS.GE.1 .AND. NRSTEP.GE.2) GO TO 72
IF (NRSTEP.GT.15) GO TO 48
GO TO 75
72 CONTINUE
C
C TERMINATE THE RESTART PHASE
C
RSTART=.FALSE.
ERRG=ZERO
NLESS=0
NRSTEP=0
NRETRY=0
JSTEP(1)=12
IF (ITRACE.NE.2) WRITE(IDEVO,8015)
8015 FORMAT(/' === END OF RESTART PHASE (NEXT STEP) ===')
75 CONTINUE
IF (RSTART) RETURN
IF (H.GE.R(7) .OR. ERRL.GT.0.15D+0 .OR. JSTEP(2).GT.0) GO TO 76
C
C DOUBLE THE STEPSIZE TO BE USED NEXT STEP
C
FAC=2.0D+0
IF (H*FAC.GT.R(7)) FAC=R(7)/H
H=H*FAC
R(4)=R(4)*FAC
JSTEP(2)=3
76 CONTINUE
C
C FORCE THE ITERATION MATRIX TO BE RE-EVALUATED NEXT STEP IF THE
C FOLLOWING CONDITIONS ARE SATISFIED
C
IF (IND(4).EQ.2) GO TO 78
IF (R(4).GE.FJAC .OR. JSTEP(1).GE.15 .OR. ERRL.GE.0.85D+0)
* IND(1)=2
GO TO 79
78 IF (R(4).GE.FJAC .OR. JSTEP(1).GE.25 .OR. R(2).GT.CORC) IND(1)=2
79 CONTINUE
IF (ITRACE.EQ.1) WRITE(IDEVO,9060) T,H,FAC,ERRL
9060 FORMAT(/' ROUTINE THETA : T=',D11.5,' H=',D11.5,' FAC=',D11.5,
* ' ERRL=',D11.5)
RETURN
C
C
C END OF SUBROUTINE STALEC
C
C
END
BLOCK DATA
C*****
C
C THIS PROGRAM INITIALISES THE PROBLEM DEPENDENT DATA HELD IN
C COMMON BLOCKS /STRAT/, /RSTAT/, /MACON/ AND /ZZZZ1/
C*****
* DOUBLE PRECISION CMIN,CORC,RESCN,RESCR,FJAC,TWOU,FOURU,TENP,
COMMON /STRAT/ JSTEP(2),CMIN,CORC,RESCN,RESCR,FJAC

```

```

COMMON /RSTAT/ TEMP(4),HMAX,NRSTEP,NLESS
COMMON /MACON/ TWOU,FOURU,TENP
COMMON /ZZZZ1/ NORM
DATA CMIN/1.0D+0/, CORC/0.5D+0/, RESCN/0.1D+0/, RESCR/5.5D+0/,
*   FJAC/2.0D+0/,JSTEP/2*0/
DATA HMAX/2.5D+0/, NRSTEP/0/, NLESS/0/
DATA TWOU/0.22D-15/, FOURU/0.44D-15/, TENP/0.60D-77/
DATA NORM/1/

```

```

C
C
C   END OF BLOCK DATA SEGMENT
C
C

```

```

END
SUBROUTINE ZNORM(NEQN,XNORM,WT,Y)

```

```

C*****

```

```

C   SUBROUTINE ZNORM.

```

```

C*****

```

```

      DOUBLE PRECISION XNORM,DNEQN
      DOUBLE PRECISION WT(NEQN),Y(NEQN)
      COMMON /ZZZZ1/ NORM

```

```

C*****

```

```

C   THIS SUBROUTINE CALCULATES THE NORM (XNORM) OF Y WEIGHTED
C   BY WT.

```

```

C   NORM -- DETERMINES THE TYPE OF NORM TO BE USED.
C   = 1 -- THE MAXIMUM NORM IS USED.
C   = 2 -- THE L2 NORM IS USED.
C   = 3 -- THE AVERAGED L2 NORM IS USED.

```

```

C   NORM IS SET TO 1 IN THE BLOCK DATA SEGMENT. THIS VALUE
C   MAY BE CHANGED AT RUN TIME BY ACCESSING NORM THROUGH THE
C   COMMON BLOCK ZZZZ1.

```

```

C*****

```

```

      XNORM=0.0D+0
      GO TO (100,200,200),NORM
100   DO 1000 I=1,NEQN
1000  XNORM=DMAX1(DABS(Y(I)/WT(I)),XNORM)
      RETURN
200   DO 1010 I=1,NEQN
1010  XNORM=XNORM+(Y(I)/WT(I))**2
      DNEQN = FLOAT(NEQN)
      IF(NORM.EQ.3)XNORM=XNORM/DNEQN
      XNORM=DSQRT(XNORM)
      RETURN

```

```

C
C
C   END OF ZNORM
C
C

```

```

END

```