DIGITAL COMPUTER SIMULATION OF COMPLEX HYDRAULIC SYSTEMS USING MULTIPORT COMPONENT MODELS

By

CHRISTOPHER K. SMITH II

Bachelor of Science in Mechanical Engineering
Tennessee Technological University
Cookeville, Tennessee
1967

Master of Science
Tennessee Technological University
Cookeville, Tennessee
1969

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
July, 1975

DIGITAL COMPUTER SIMULATION OF COMPLEX HYDRAULIC SYSTEMS USING MULTIPORT COMPONENT MODELS

Thesis Approved:

Henry R. Zeberta
Thesis Adviser

Karl M. Reid

James E. Bose

Ronald P. Pholen

N N Durham
Dean of the Graduate College

ACKNOWLEDGMENTS

The author is very grateful to all who have encouraged him throughout his educational career. Without their guidance and assistance this endeavor would not have been possible.

Deep appreciation and gratitude is expressed to Dr. Henry R. Sebesta, the author's major adviser. His technical guidance, patience and friendship have been major contributing factors towards the completion of this work. Special thanks are also expressed to Dr. Karl N. Reid, whose abilities to convey optimism and inspire the author are outstanding. The time, the constructive criticisms, and the willingness to be helpful given by the members of the thesis committee, Dr. James E. Bose and Dr. Ronald P. Rhoten, are also gratefully appreciated.

Special gratitude is expressed to the author's wife, Doris, and daughter, Lisa, for their understanding and confidence in him and for their many sacrifices.

The author also wishes to thank the Center for Systems Science at Oklahoma State University for supporting the research reported here and to thank Mr. Lael B. Taplin, Dr. A. B. Van Rennes, and Dr. D. Bitondo at Bendix Research Laboratories for allowing the author the time to finish this manuscript. Thanks are also given to Mrs. Virginia Hanadel for her assistance in typing the early drafts of this manuscript.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

a - Constant defined in Equation (7)

b - Constant defined in Equation (12)

F - Functional representation of an algebraic equation

G - Functional representation of a first-order differential equation

t - Independent variable

u - Independent port variable

v - Dependent port variable

x - State or dependent variable

y - Algebraic variable

CHAPTER I

INTRODUCTION

In recent years, interest in the simulation of the dynamics of continuous systems has increased manifold. This increase has been augmented by the increasing availability and popularity of the digital computer and is evidenced by the number of generalized numerical simulation programs being presented in the literature.

The work required for the successful simulation of a dynamic system on a digital computer encompasses three fields of technology: (1) engineering, (2) numerical analysis, and (3) computer programming. Engineering is required to generate the mathematical equations which describe the performance of the physical system, numerical techniques are required for solving the equations, and programming is required for implementing these in a usable package.

Objective and Scope of Study

The objective of this study was the development of modeling equation forms and numerical algorithms which could be used in a digital computer program for the simulation of complex systems. During this study it was assumed that a complex system could be represented by a network of multiport models and that each model could be represented by a set of first-order, ordinary differential equations and a group of algebraic equations. Equation forms were sought which were not restrictive and algorithms were

selected on the basis of accurate and efficient solution. Chapter II contains an explanation of the forms and algorithms which were developed.

This study was the first part of a two-part project. While this study dealt with the conceptual framework of a simulation program, the second part dealt with the implementation of these concepts in the simulation program HYDSIM. The main features of the HYDSIM program include:

1. The implementation of special solution algorithms.
2. A multiport-model representation of components.
3. A library of standard component models.
4. A block-oriented and free-format input data form with a pre-pocessor for error checking and user convenience.
5. The sorting of algebraic equations into sets for increased efficiency.
6. A capability of accepting user programmed component models.

The overall structure of HYDSIM as an implementation of the conceptual framework is discussed in Chapter IV, and the methods of using the HYDSIM program are described in "HYDSIM User's Manual" (1).

Results of Study

This study deals with the dynamic simulation of complex systems. Included in Chapter III of this study are a presentation of functional forms of equations which may be used to model physical systems, numerical algorithms which may be used to solve the modeling equations, and in Chapter IV an introduction to a generalized simulation program for complex systems which utilizes these modeling forms and algorithms. These

forms and algorithms are applicable to the simulation of any complex system which may be represented by a set of first-order, nonlinear differential equations coupled with sets of nonlinear algebraic equations. However, due to error checking and output considerations, HYDSIM was initially developed for the simulation of hydraulic, mechanical and electrical systems.

Two forms of modeling equations are presented in Chapter III. Both are similar and assume that the components can be described by coupled sets of nonlinear algebraic and differential equations using multiport modeling techniques. However, the second form may be used for block-oriented component models.

Two algorithms are presented in Chapter III for the solution of each modeling form. The first algorithm for each form uses "conventional" numerical methods for the solution of the modeling equations. The second algorithm uses "conventional" numerical integration for the solution of the differential equations in conjunction with an implicit method for the solution of the algebraic equations. This method, called the Implicit method, is numerically similar to the Newton-Raphson method, but it is not iterative at each time step which results in a reduction of computer execution time. Finally, a third algorithm is presented for the solution of the block-oriented modeling form. This algorithm combines the methods of the first and second algorithms with a method for separating the set of algebraic equations into a number of smaller sets.

CHAPTER II

PRELIMINARY CONSIDERATIONS

Before the development of a computer algorithm is undertaken, the desired organization and capabilities of the completed program should be considered. This organization will be affected strongly by two areas of interest: (1) analytical techniques presently in use for describing or simulating complex systems, and (2) the organization of previously successful simulation programs.

Related Literature

In 1955, Selfridge (2) introduced a method of coding a digital computer to "operate" as a differential analyzer, more widely known as an analog computer. Since then numerous general-purpose simulation programs have been introduced (3-13). Brennan and Linebarger (14) and Clancy and Fineberg (15) have provided excellent surveys of the first ten years of simulation program development. Several programs containing major contributions to the field of simulation are discussed below.

In 1958, Stein and Rose (3) generated ASTRAL (acronym for Analog Schematic TRanslator to Algebra Language. ASTRAL contained three firsts: (1) the program was a compiler which produced a FORTRAN deck for execution; (2) the simulation was executed using floating-point arithmetic; and (3) a sorting algorithm was used which relieved the user from the task of ordering the program inputs to obtain a correct solution (16).

Gaskill et al. (6) introduced DAS (acronym for Digital Analog Simulator) in 1963. This used a simplified input language and was an excellent simulation program; however, it had two drawbacks. DAS contained an elementary integration routine (Euler's method) and did not contain a sorting algorithm. MIDAS (Modified Integration DAS) introduced in 1964 by Harnett, Sansom and Warshawsky (8) contained a sorting algorithm and a sophisticated fifth-order predictor-corrector integration method with variable step size. MIDAS also included a simple input format and a method of handling algebraic loops (loops in the model equations which are not broken by an integrator, a delay, or any function with memory). Because of these advantages plus the facts that the program was easy to use and was written for the popular 7090-7094 IBM computers, MIDAS gained wide acceptance.

PACTOLUS (the river in which King Midas washed off the golden touch) was introduced in 1964 by R. D. Brennan (9) and it also enjoyed wide acceptance. PACTOLUS retained most of the features of MIDAS, but was written for the small IBM 1620 computer. This program allowed the user to make on-line changes in a simulation through computer switch settings and console entries. Consequently, PACTOLUS came closer to the true hands-on control of an analog computer than any of its predecessors.

The input language of all the programs discussed in detail above may be classified as block-oriented. In block-oriented programs the functional capabilities of each type of block included in the program are defined in terms of the block inputs and outputs by the programmer. User input to the program consists of a description of block interconnections, block type identifications, and various parameters; and it may be prepared from a block diagram of the system to be simulated. This type of input

configuration was a direct result of the program originator's efforts to duplicate the operation of an analog computer.

The block-oriented programs caused difficulty for the program user in programming algebraic expressions which could easily be expressed in languages such as FORTRAN. In addition, some users who modeled systems directly in equation form desired to input these equations directly to the simulation program without creating a block diagram. Consequently, language-oriented simulation programs were introduced.

Three of the most significant language-oriented programs are MIMIC, DSL/90, and 360 CSMP (10, 11, 13). These programs incorporated all of the major features of the block-oriented programs plus increased flexibility due to their input languages. However, these input languages are very FORTRAN oriented and difficult to learn.

In addition to the general-purpose simulation programs discussed above, many special-purpose programs have been introduced. Unlike the general-purpose programs which were intended for the simulation of any system that may be represented by a set of ordinary differential equations, the special-purpose programs were intended for the simulation of a special class of systems (for example, mechanical, electrical, or fluid systems). Consequently, the input language of a special-purpose program may be structured in the most natural format for the class of systems to be simulated and the numerical methods used in the program may be adapted for best solving the types of modeling equations most often used.

ENPORT is unique among the special-purpose simulation programs (17). This program may be used to simulate the same classes of systems as the general-purpose programs. However, the input format of ENPORT has been

structured to be used with the bond graph modeling techniques first introduced by Paynter (18).

One of the most widely used simulation programs for electrical systems is ECAP (19). In addition to providing transient response solutions, this special-purpose program can also provide AC and DC analyses of electrical circuits. ECAP has also been used to simulate systems which were not electrical. However, since an analogy must be established between electrical components and the components of the system to be studied, ECAP is not the most desirable tool from the user's viewpoint for simulating systems which are not electrical. The numerical methods used in ECAP are heavily dependent on linear techniques and nonlinearities may only be simulated through piecewise-linear approximations.

Other widely accepted simulation programs for electrical systems are NET I and SCEPTRE. These programs and ECAP are compared from a user's viewpoint by Lindgren (20).

A summary of the special-purpose programs available for the simulation of hydraulic (fluid power) systems was given by Waterman et al. (21) in a report for the U.S. Air Force. Most notable among the programs described are the HYDSIM and HYTRAN programs. These programs contain the models of various hydraulic components programmed in subroutines. The programs then use these subroutines as required to simulate hydraulic systems. Zielke (22) provided more information about HYTRAN.

The HYDSIM program discussed in Chapter IV of this study is the second version of the program (HYDSIM II). The first version, HYDSIM I, was developed at Oklahoma State University in conjunction with General Dynamic Corporation during 1970 (23, 24). This earlier version lacked most of the features and capabilities of the current program. For

example, it did not have the HYDSIM II capability of simultaneously solving coupled, algebraic equations when the equations were contained in more than one component model. However, HYDSIM I did contain a library of standard components.

Organization of a Generalized Computer Program for Hydraulic System Simulation

Because a hydraulic system is usually composed of many discrete hydraulic components coupled together, a block-oriented input language would seem natural. However, instead of each block representing a basic function as in the "analog-like" languages described above, each block should represent a major hydraulic component such as a valve or a cylinder. The incorporation of a library of models for such blocks would relieve the program user of much repetitive programming. In addition, the organization of the program in a manner where each component model would be contained in a separate subroutine would facilitate using the models in many different interconnection configurations and would ease the task of adding additional models to the program at a later date. Also, since fluid power (not just pressure nor just flow) is the major consideration in a hydraulic system, multiport modeling techniques should be used.

Multiport models of dynamic system components have been in use in various forms for many years. For example, bond graph techniques are based upon the use of multiport models (18). Multiport techniques are in evidence whenever the energy transfer at energy exchange ports of system components (or subsystems) may be expressed as products of pairs of systems variables.

Figure 1 shows a typical method of graphically displaying the pairs of energy exchange variables for multiport models. Typically, the power transferred at the ports connecting the pump and pipe is represented by the product of the pressure variable, P_1, and the flow variable, Q_1. It should be noted that the directions of the arrows shown to represent the variable pair at this port connection do not represent the direction of flow, etc.; the directions represent the causality of the port variables. (This is true of all ports shown in this study.) For example, P_1 is independent to the pump model and dependent to the pipe model, and Q_1 is dependent to the pump model and independent to the pipe model.

The application of multiport modeling techniques to a hydraulic system usually produces a set of nonlinear differential equations which are coupled with sets of nonlinear algebraic equations. The algebraic equations may be manually reduced to some extent within each block, but since these algebraic sets usually span several blocks, they may not be eliminated entirely. Elimination of these sets is also hindered by the fact that for a generalized program the pattern of block interconnections is not known at model programming time and will vary as different systems are simulated. Typical forms of these equations are discussed in Chapter III.

A numerical method is required to solve the coupled sets of nonlinear algebraic and differential equations. Two methods are discussed in Chapter III of this study, and a method has been proposed by Gear (25) and by Brayton, Gustavson, and Hachtel (26). The latter method uses a predictor type of equation to approximate the current value of each state variable in terms of the current state variable derivative value and k previous state variable values. This equation, written for each state

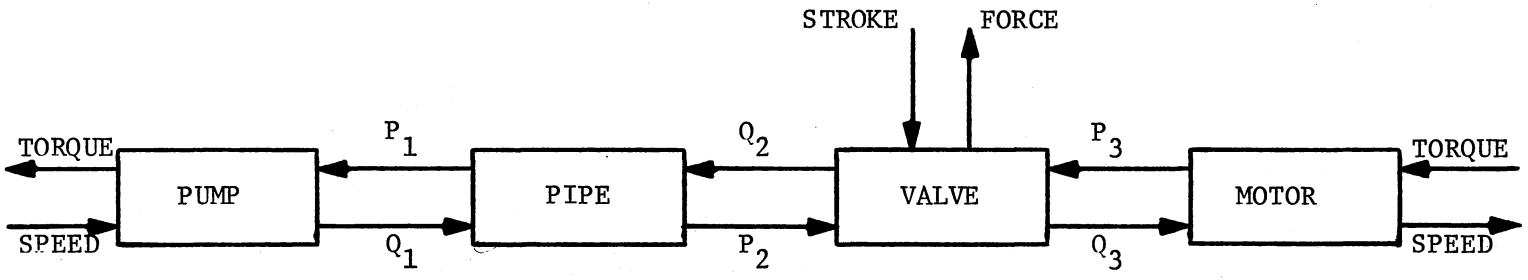


Figure 1. Block Diagram of Multiport Models

variable, may be substituted into the system differential equations to eliminate the state variable derivatives. The resulting algebraic equations and the algebraic equations from the original modeling process form one large set of coupled, nonlinear algebraic equations which may be solved iteratively for the current values of the state and algebraic variables by the Newton-Raphson method.

The first method discussed in Chapter III uses the Newton-Raphson method and any "conventional" numerical integration technique to solve the equation sets. However, the second method uses an implicit method (hereafter called the Implicit method) to determine derivative values for the algebraic variables and then uses a "conventional" numerical integration technique to simultaneously solve for the state and the algebraic variables in a similar manner. The Implicit method is not iterative at each time step, but uses a linear equation similar to that used in one iteration of the Newton-Raphson method. In addition, the Implicit technique does not cause the coupling of all the algebraic equations into one large set, but allows the algebraic equations to be separated into sets based upon coupling of the algebraic variables only.

CHAPTER III

ALGORITHMS FOR SIMULATION OF HYDRAULIC SYSTEMS

A Generalized Form for Hydraulic System Modeling Equations

The system of equations developed during the mathematical modeling of most hydraulic systems is composed of algebraic and ordinary differential equations (27). This section discusses these equations and their functional forms.

In general, the algebraic equations included in a mathematical model of a hydraulic system are nonlinear and may not be reduced analytically. These equations relate a set of algebraic variables, y's, to a set of state variables, x's, and the independent variable time, t. A functional representation for a set of m of these equations is

$$0 = F_i\ (x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_i, \ldots, y_m, t),$$

$$i = 1, 2, \ldots, m. \qquad (1)$$

Examination of the y-variables in the algebraic equations for a specific hydraulic system model often reveals that the equations may be divided into a number of sets where no y-variable coupling exists between sets of algebraic equations. Since this separation must be performed for a specific set of equations and since it would not greatly affect the algorithms discussed below except with regard to execution speed,

separation of algebraic equations will not be discussed until near the end of this chapter.

The ordinary differential equations resulting from the mathematical modeling of a hydraulic system are, in general, nonlinear also. These equations, when written as a set of first-order differential equations, relate the state variables, x's; their first derivatives with respect to time, $\dot{x}$'s; the algebraic variables, y's; and the independent variable time, t.[1]

$$\dot{x}_j = G_j (x_1, x_2, \ldots, x_j, \ldots, x_n, y_1, y_2, \ldots, y_m, t),$$

$$j = 1, 2, \ldots, n. \qquad (2)$$

Equations (1) and (2) form a coupled system of m + n equations which may be used to model hydraulic systems. These equations were assumed to be nonlinear; however, it would be instructive to briefly discuss their solution assuming complete linearity. In addition, the generation of Equations (1) and (2) in a linear form is often suggested in the literature where a manual solution is anticipated and physical system conditions permit this simplification (28).

With an assumption of linearity, Equation (1) may be solved using Cramer's rule (29) to yield

$$y_i = F_i (x_1, x_2, \ldots, x_n, t), \; i = 1, 2, \ldots, m. \qquad (3)$$

Substitution of Equation (3) into Equation (2) yields

$$\dot{x}_j = G_j (x_1, x_2, \ldots, x_j, \ldots, x_n, t),$$

$$j = 1, 2, \ldots, n. \qquad (4)$$

---

[1] It may not be possible to write all differential equations as sets of first-order differential equations; however, this difficulty is more than offset by the increased flexibility allowed during the development of the algorithms in this chapter through the use of Equation (2).

Since Equations (1) and (2) were assumed to be linear, Equation (4) will be linear and may be solved using the state transition matrix technique (30).

In the above solution method, algebraic equations are analytically solved and the algebraic variables are eliminated from the differential equations. This solution method is usually the most satisfactory. However, when the algebraic equations are nonlinear (as for most hydraulic systems) and linearization is not acceptable, numerical methods and a digital computer are often employed for the solution of Equations (1) and (2).

An Iterative Algorithm for the Simultaneous Solution of the Generalized Modeling Equations

Many methods have been given in the literature for numerically solving sets of first-order differential equations and for numerically solving sets of algebraic equations. This section discusses some of these methods and an algorithm for applying these methods to the numerical solution of coupled sets of differential and algebraic equations (the generalized modeling equations).

Some of the techniques given in the literature for numerically integrating sets of differential equations are the Adams and the Runge-Kutta integration methods (31). Both of these explicit integration methods can be used in a similar manner to integrate sets of differential equations in the form of

$$\dot{x}_j = G_j\,(x_1, x_2, \ldots, x_n, t), \quad j = 1, 2, \ldots, n, \qquad (5)$$

where $\dot{x}$ is the first time derivative of the state variable x.

After initialization the Adams or the Runge-Kutta integration method can be used in a generalized manner as shown in Figure 2.[2] The integration method will yield a value of the independent variable, t^* (usually considered to be time), and a set of values for the state variables, x_1^* through x_n^* (not to be considered solution values). The integration method requires values of the first derivatives of the state variables, $\dot{x}_1^*$ through $\dot{x}_n^*$, which may be evaluated from Equation (5) as shown in Figure 3. After several iterations of producing values for the variables and receiving derivative values, the integration method will yield a set of solution values for the state variables, x_1 through x_n. This solution will be for a value of the independent variable, t, which will be advanced by a small increment from the value associated with the previous set of solution values (or initial conditions). This process may be repeated until the integration method has incremented the independent variable through the desired range.

The Newton-Raphson method is an iterative method which can be used to find the solution to a set of nonlinear algebraic equations of the form

$$0 = F_i\ (y_1, y_2, \ldots, y_m), \quad i = 1, 2, \ldots, m. \tag{6}$$

This method requires the evaluation of the partial derivatives of each algebraic equation with respect to each algebraic variable and has been explained in more detail in Appendix A.

Solution of the generalized modeling equations by direct use of the Newton-Raphson method and the generalized integration method discussed

---

[2]Most explicit, numerical integration methods may be used in a similar manner.

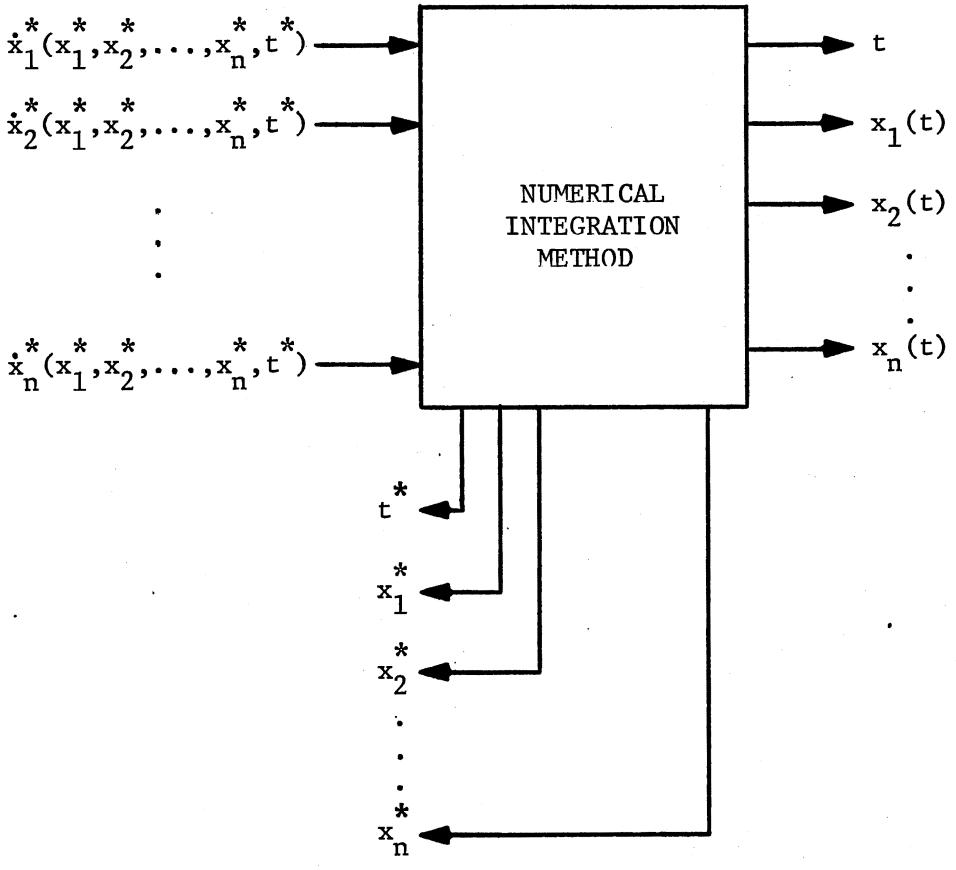


Figure 2. Concept of a Generalized Numerical Integration Method After Initialization

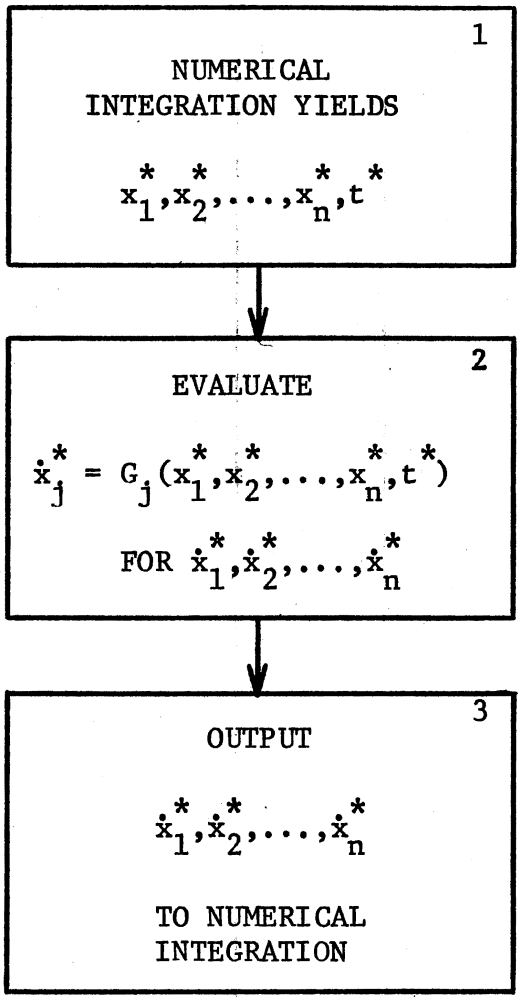


Figure 3. Steps Required for Use of Runge-Kutta or Adams-Moulton Integration After Initialization

above is not possible because Equations (1) and (2) do not match the functional forms required by the solution methods, Equations (5) and (6). However, the steps required to use the generalized integration, as shown in Figure 3, may be modified to allow the incorporation of the Newton-Raphson technique such that the generalized modeling equations may be solved.

Figure 4 shows the modified steps required when the differential and algebraic equations to be solved are coupled as in the generalized modeling equations. Step 2 of this figure shows that the functional form of the algebraic equations may be reduced to the form required for Newton-Raphson solution because the generalized integration method has supplied numerical values for the state variables and the independent variable in Step 1. Step 2 also represents the solution of the reduced algebraic equations by the Newton-Raphson method. Steps 3 and 4 are equivalent to Steps 2 and 3 in Figure 3.

A Modified Algorithm for the Simultaneous Solution of Modeling Equations With Block-Oriented Organization

Block-oriented organization of the mathematical equations which model a hydraulic system implies that the equations which describe each hydraulic component or hydraulic function (block) are defined only in terms of:

1. The independent variable time.
2. The state variables contained within the block.
3. The algebraic variables contained within the block.
4. The independent port variables for the block.

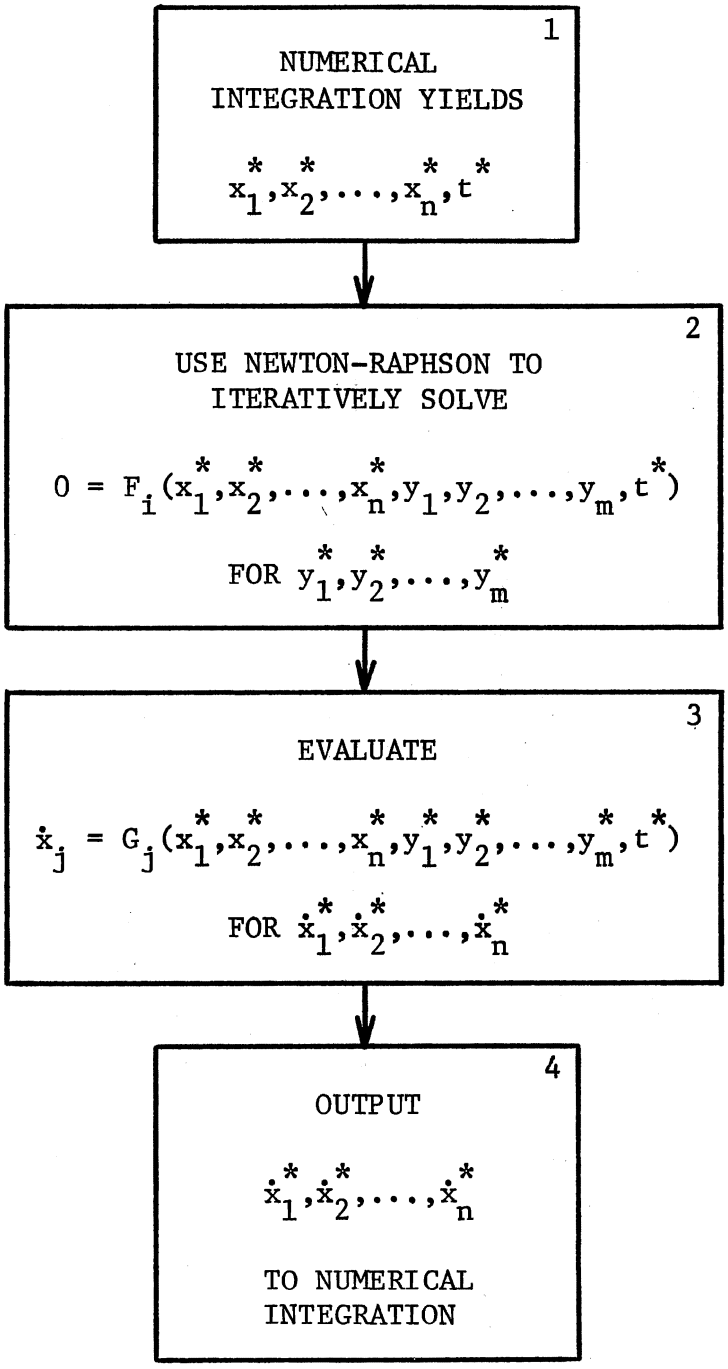


Figure 4. Algorithm to Solve Coupled Algebraic and Differential Equations

Therefore, an algebraic or state variable contained in Block A may not be explicitly included in the equations that model Block B. However, the same effect may be obtained by passing the required information through any port variables which might connect Blocks A and B (directly or indirectly).

A port connection between two blocks (also referred to as components) is illustrated in Figure 5. This figure shows the connection of Port q of Block p to Port s of Block r. The port variables of the blocks are related by

$$u_{pq} = a_{pq} v_{rs} \tag{7}$$

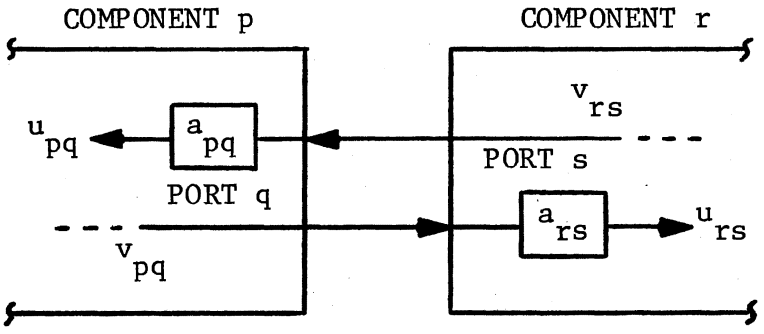
and

$$u_{rs} = a_{rs} v_{pq}, \tag{8}$$

where each "a" is a constant with the value ± 1 (depending on a port variable sign convention). When, due to the generality, it is uncertain what port and block are connected to Port q of Block p, the term v_{rs} in Equation (7) would be replaced by a general term v^c_{pq}. This term, v^c_{pq}, may be defined as the dependent port variable at the port connected to Port q of Block p. Consequently, for the port connection shown in Figure 5, v^c_{pq} and v_{rs} refer to the same variable.

The generalized modeling equations will be modified to conform to the above restrictions. In addition, another subscript will be added to the variables to indicate with which block the variables are associated. For example, y_{hi} will be the i^{th} algebraic variable in Block h.

With the assumption that Block h contains m algebraic variables, n state variables, and p ports (therefore p independent and p dependent port variables), Equations (1) and (2) may be rewritten for Block h as



$$u_{pq} = a_{pq} v_{rs} \text{ AND } u_{rs} = a_{rs} v_{pq}$$

$$\text{WHERE } a_{pq} = \pm 1 \text{ AND } a_{rs} = \pm 1$$

Figure 5. A Port Connection Between Two Components

$$0 = F_{hi}\ (x_{h1},\ x_{h2},\ \ldots,\ x_{hn},\ y_{h1},\ y_{h2},\ \ldots,$$

$$y_{hm},\ u_{h1},\ u_{h2},\ \ldots,\ u_{hp}, t),\quad i = 1, 2, \ldots, m \qquad (9)$$

$$\dot{x}_{hj} = G_{hj}\ (x_{h1},\ x_{h2},\ \ldots,\ x_{hn},\ y_{h1},\ y_{h2},\ \ldots,$$

$$y_{hm},\ u_{h1},\ u_{h2},\ \ldots,\ u_{hp}, t),\quad j = 1, 2, \ldots, n. \qquad (10)$$

If Equations (9) and (10) had been written for Block h in the form of Equations (1) and (2), x and y variables from outside of Block h may have appeared in the equations. These x and y values from outside of Block h are implicitly included in Equations (9) and (10) through the independent port variables, u's.

It will aid in developing an algorithm for the solution of Equations (9) and (10) if functional forms are defined for the equations which will be used to evaluate the dependent port variables within Block h. For the k^{th} port in Block h the dependent port variable may be defined by

$$v_{hk} = v_{hk}\ (x_{h1},\ x_{h2},\ \ldots,\ x_{hn}, t) \qquad (11)$$

or

$$v_{hk} = b_{hk} \cdot y_{h\ell}, \qquad (12)$$

where b_{hk} is a constant and $y_{h\ell}$ is any algebraic variable within Block h. It should be noted that Equations (11) and (12) do not allow a dependent port variable of Block h, v_{hk}, to be explicitly a function of an independent port variable of Block h nor a combined function of an algebraic and a state variable of Block h. These restrictions may be circumvented where necessary by temporarily defining the dependent port variable, v_{hk}, in terms of time and any state, algebraic, and independent port variables within h. A new algebraic variable may then be introduced

into Block h and used to replace v_{hk} in its defining equation. The defining equation can then be considered an algebraic equation of Block h and be rearranged into the form of Equation (9), and v_{hk} can be defined in terms of the new algebraic variable following the form of Equation (12). Thus a dependent port variable of Block h can be an implicit function of time and any state, algebraic, and independent port variable within Block h.

The steps for using the generalized integration shown in Figure 4 may be modified to solve Equations (9) and (10) as shown in Figure 6. Each step shown in Figure 6 must be completed for every block (component) in the hydraulic system before proceeding to the next step. Step 1 indicates that the generalized integration technique will supply specific values for the independent variable time and all of the state variables. Step 2 uses these variables and Equation (11) to evaluate those dependent port variables which are only a function of state variables and time. Equation (7) is used in Step 3 to transfer the values of any dependent port variables evaluated in Step 2 to the independent port variables of connected blocks. Step 4 illustrates that at this point the variables remaining unknown in the algebraic equations (Equation (9)) are the algebraic variables and some of the independent port variables. It should be noted that all of the independent port variables remaining unknown in Equation (9) can be related through Equation (7) to dependent port variables defined only by Equation (12) in other blocks. Therefore, each of the independent port variables remaining unknown in Equation (9) represents a constant multiplied by an algebraic variable from another block. Step 4 also represents the solution of the algebraic equations for the algebraic variables using the Newton-Raphson method. The

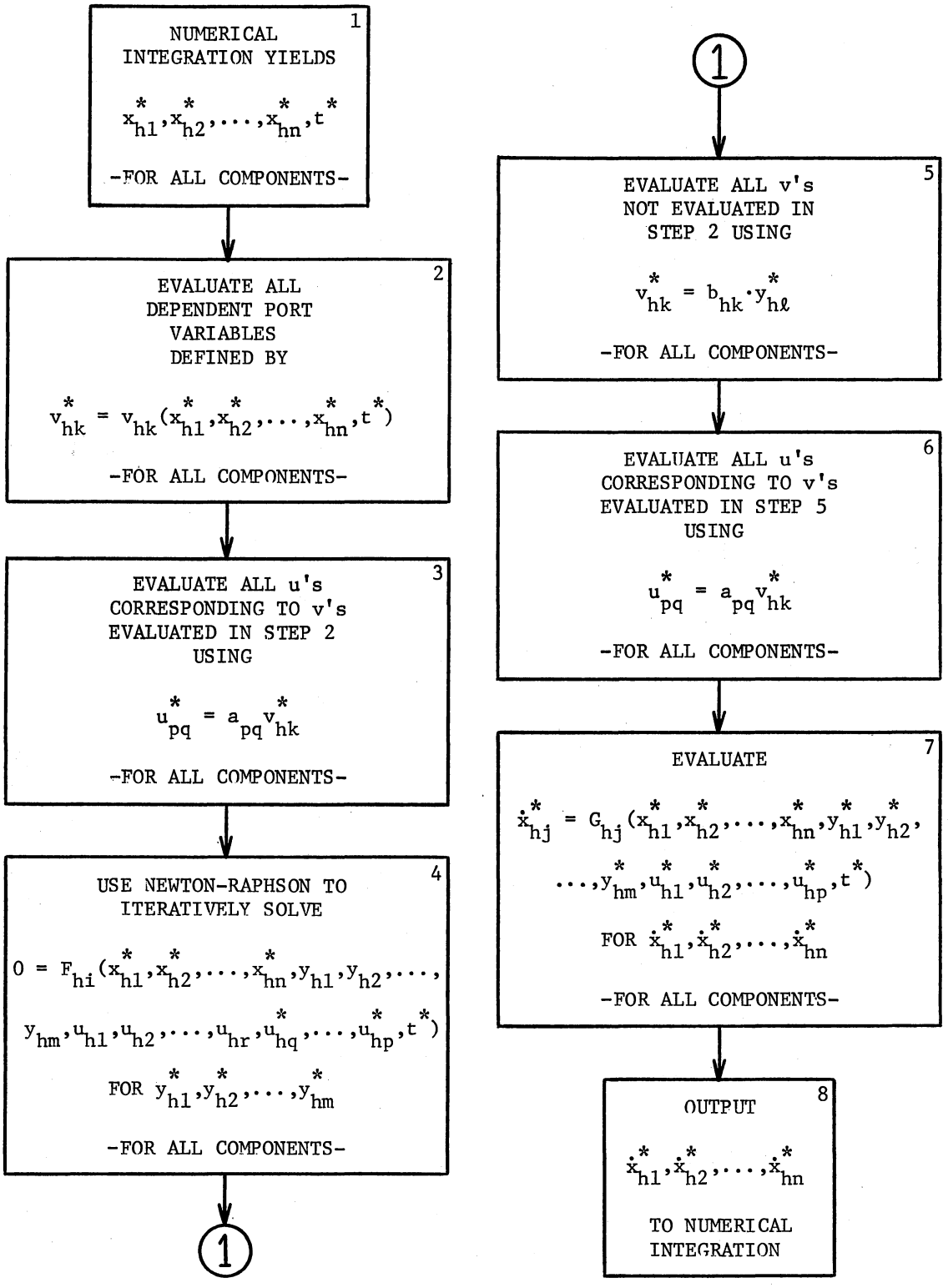


Figure 6. Algorithm to Solve Coupled and Block-Oriented Algebraic and Differential Equations

technique of applying the Newton-Raphson method for solving these equations is explained in Appendix B. Step 5 uses Equation (12) to evaluate the dependent port variables not evaluated in Step 2, and Step 6 uses Equation (7) to transfer these dependent port variable values to independent port variables in connected blocks. Step 7 uses Equation (10) to evaluate the derivatives of all the state variables, and Step 8 illustrates the submission of these derivative values to the generalized integration method.

An Implicit Method for Solving Nonlinear Algebraic Equations Without Iteration

The numerical solution methods explained above for solving coupled sets of nonlinear differential and algebraic equations require the Newton-Raphson solution of the algebraic equations prior to each evaluation of the state variable derivatives. The Newton-Raphson method is iterative and may require several iterations for convergence to a solution. This section develops an implicit method for evaluating the algebraic equations. This method, hereafter called the Implicit method, is not iterative and requires a similar amount of computation as one iteration of the Newton-Raphson method.

The Implicit method is based upon the use of a numerical integration algorithm to solve the algebraic equations simultaneously with the differential equations. Use of the algorithm in this manner will require the evaluation of the time derivatives of the algebraic variables each time the time derivatives of the state variables are evaluated.

Explicit methods for determining dy_i/dt from the generalized modeling equations (rewritten here for convenience),

$$0 = F_i\,(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m, t),$$

$$i = 1, 2, \ldots, m \tag{13}$$

$$\dot{x}_j = G_j\,(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m, t),$$

$$j = 1, 2, \ldots, n, \tag{14}$$

may not be used since these methods would first require the analytical solution of Equation (13). However, the time derivatives of the algebraic variables can be evaluated in an implicit manner.

The chain rule allows the derivatives of Equation (13) to be written as

$$\frac{dF_i}{dt} = \sum_{j=1}^{n} \frac{\partial F_i}{\partial x_j}\frac{dx_j}{dt} + \sum_{\ell=1}^{m} \frac{\partial F_i}{\partial y_\ell}\frac{dy_\ell}{dt} + \frac{\partial F_i}{\partial t}\,, \quad i = 1, 2, \ldots, m. \tag{15}$$

Since the right side of Equation (15) is equal to zero, it may be rearranged in vector-matrix form to yield

$$\begin{bmatrix} \frac{\partial F_1}{\partial y_1}\Big|_* & \frac{\partial F_1}{\partial y_2}\Big|_* & \cdots & \frac{\partial F_1}{\partial y_m}\Big|_* \\ \frac{\partial F_2}{\partial y_1}\Big|_* & \frac{\partial F_2}{\partial y_2}\Big|_* & \cdots & \frac{\partial F_2}{\partial y_m}\Big|_* \\ \vdots & \vdots & & \vdots \\ \frac{\partial F_m}{\partial y_1}\Big|_* & \frac{\partial F_m}{\partial y_2}\Big|_* & \cdots & \frac{\partial F_m}{\partial y_m}\Big|_* \end{bmatrix} \begin{bmatrix} \frac{dy_1}{dt} \\ \frac{dy_2}{dt} \\ \vdots \\ \frac{dy_m}{dt} \end{bmatrix} = \begin{bmatrix} -\sum_{j=1}^{n} \frac{\partial F_1}{\partial x_j}\Big|_* \frac{dx_j}{dt}\Big|_* - \frac{\partial F_1}{\partial t}\Big|_* \\ -\sum_{j=1}^{n} \frac{\partial F_2}{\partial x_j}\Big|_* \frac{dx_j}{dt}\Big|_* - \frac{\partial F_1}{\partial t}\Big|_* \\ \vdots \\ -\sum_{j=1}^{n} \frac{\partial F_m}{\partial x_j}\Big|_* \frac{dx_j}{dt}\Big|_* - \frac{\partial F_m}{\partial t}\Big|_* \end{bmatrix}, \tag{16}$$

where the bar and asterisk to the right of each derivative implies the evaluation of the derivatives at the values $x_1^*, x_2^*, \ldots, x_n^*, y_1^*, y_2^*, \ldots, y_m^*, t^*$. Equation (16) may be solved for dy_1/dt through dy_m/dt at $x_1^*, x_2^*, \ldots, x_n^*, y_1^*, y_2^*, \ldots, y_m^*, t^*$ with the solutions being used by the numerical integration algorithm to solve for $y_1, y_2, \ldots, y_m$.

The form of Equation (16) should be compared with the form of an equation used in the Newton-Raphson method, Equation (34) in Appendix A. Both of these equations are linear, of the same order, and contain the same matrix of partial derivatives. Equation (34) must be solved repeatedly until convergence is attained for each Newton-Raphson solution of the algebraic equations; however, in the Implicit method described above, Equation (16) is only solved once per algebraic solution.

Figure 7 illustrates the steps required to solve Equations (13) and (14) using Equation (16). After initialization, these steps use a generalized integration method (as illustrated in Figure 2) in exactly the same manner as if m + n differential equations were being solved. Step 1 shows that the integration algorithm is used to supply specific values for $y_1, y_2, \ldots, y_m$ in addition to values for $x_1, x_2, \ldots, x_n$ and t. In Step 2 the differential equations of Equation (14) are used to evaluate the time derivatives of the state variables. Steps 3 and 4 are used to evaluate array elements for Equation (16) which is solved for the time derivatives of the algebraic variables in Step 5. Step 6 completes the process by showing that the derivatives of the state and algebraic variables are made available to the integration algorithm.

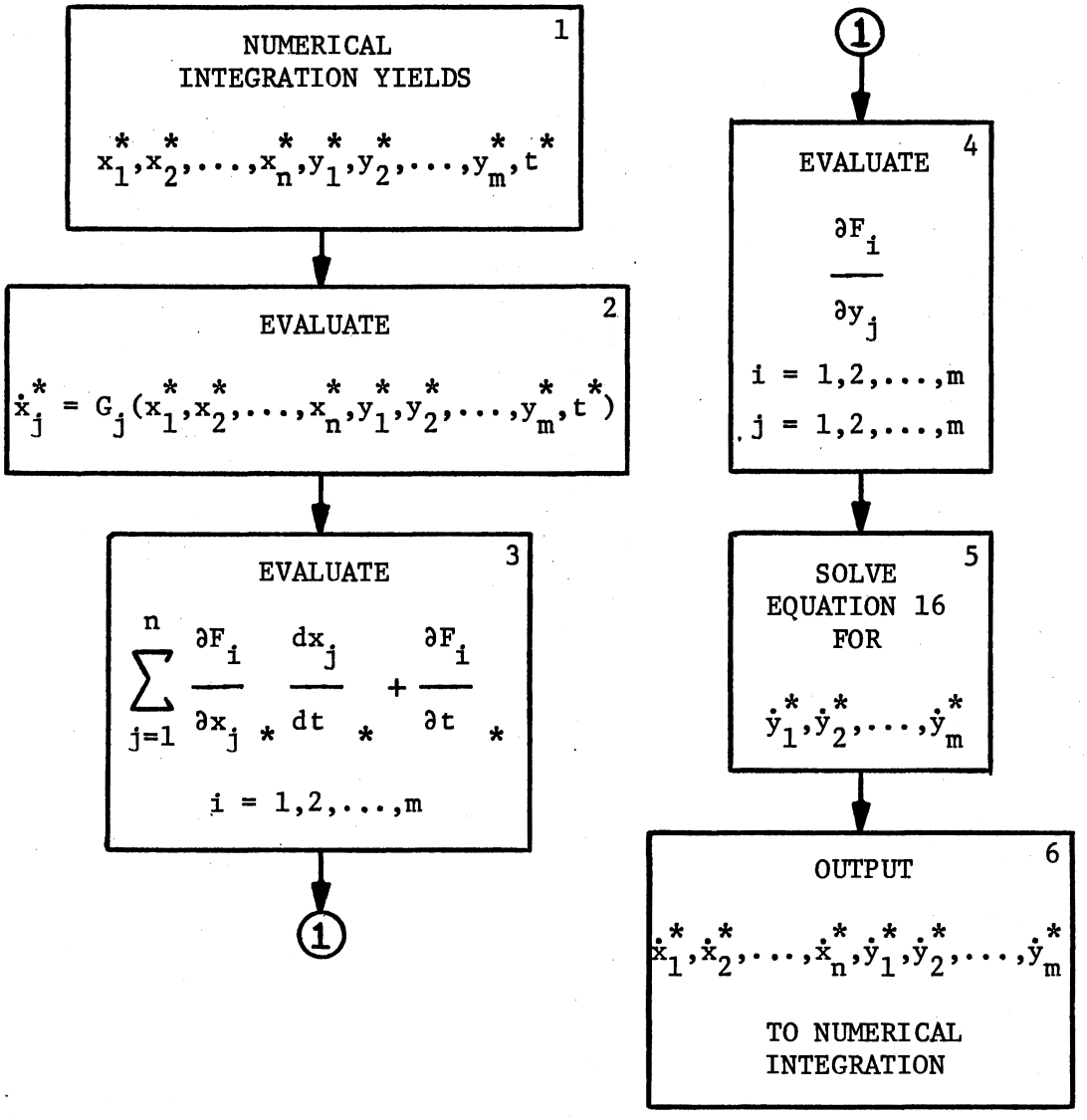


Figure 7. Algorithm to Solve Generalized Modeling Equations by the Implicit Method

At the beginning of the solution of Equations (13) and (14) using the Implicit method, the numerical integration method will require initial conditions for the state variables and the algebraic variables. Initial conditions for the state variables and some of the algebraic variables may be provided by examining the initial state of the physical system which Equations (13) and (14) describe. However, it is possible that some or all of the algebraic initial conditions may not be determined in this manner without solving a set of algebraic equations. In this case another solution method for algebraic equations such as the Newton-Raphson method may be used once at the start of the solution to initialize the algebraic variables.

Some important differences should be pointed out between the Implicit method explained above and the Newton-Raphson method. During each iteration, the Newton-Raphson method improves its solution values by applying solution correction equations developed from Taylor series expansions of the algebraic equations (see Appendix A). Since the Taylor series expansions are truncated and usually only include the first-order terms, the solution correction equations used are approximations to the true solution correction equations (those including all terms in the Taylor series). Therefore, the correction equations must be applied repetitively until the solutions are sufficiently corrected or the method diverges.

A Newton's method has been developed which uses solution correction equations that include through the second-order Taylor series terms (32). This method usually converges more rapidly than the first-order Newton-Raphson method. However, the second-order method is more difficult to apply since it requires second-order partial derivatives of the algebraic

equations to be derived and evaluated in addition to the first-order partial derivatives required by the first-order method.

The Implicit method makes no approximations in the solution of the algebraic equations except those included in the numerical integration method. The method requires approximately the same number of computations to calculate exact values for the time derivatives of the algebraic equations as required for one iteration of the first-order Newton-Raphson method.

A high-order numerical integration method may be used in conjunction with the Implicit method in order to minimize the error in solving for the algebraic variables. For example, a method such as a fourth-order Runge-Kutta method may be used (31). This integration method is approximately as accurate as numerical integration using a Taylor series expansion of the solution variables which is truncated after the fourth-order terms. However, the Runge-Kutta method does not require derivative values above the first-order as required by a high-order Taylor series integration method.

A Modified Implicit Method for the Simultaneous Solution of Modeling Equations With Block-Oriented Organization

The Implicit method may be adapted for use with block-oriented modeling equations. For the h^{th} block (component) of a system containing ℓ blocks, the block-oriented modeling equations may be written as:

$$0 = F_{hi}\,(x_{h1}, x_{h2}, \ldots, x_{hn_h}, y_{h1}, y_{h2}, \ldots, y_{hm_h}, u_{h1}, u_{h2}, \ldots, u_{hr_h}, u_{hq_h}, \ldots, u_{hp_h}, t),$$
$$i = 1, 2, \ldots, m_h, \qquad (17)$$

$$\dot{x}_{hj} = G_{hj}\,(x_{h1}, x_{h2}, \ldots, x_{hn_h}, y_{h1}, y_{h2}, \ldots, y_{hm_h}, u_{h1}, u_{h2}, \ldots, u_{hp_h}, t),\ j = 1, 2, \ldots, n_h, \tag{18}$$

$$u_{hk} = a_{hk}\, v^{c}_{hk},\quad k = 1, 2, \ldots, p_h, \tag{19}$$

and

$$v_{hk} = v_{hk}\,(x_{h1}, x_{h2}, \ldots, x_{hn_h}, t),\quad k = 1, 2, \ldots, p_h, \tag{20}$$

or

$$v_{hk} = b_{hk} \cdot y_{hi},\quad k = 1, 2, \ldots, p_h. \tag{21}$$

Equations (20) and (21) illustrate the two allowable equations forms for defining dependent port variables. Each of the p_h dependent port variables in Block h must be defined as a function of time and the state variables within Block h (Equation (20)) or as the product of a constant and one algebraic variable within Block h (Equation (21)). Equation (19) illustrates the relationship of the k^{th} independent port variable in Block h, u_{hk}, to the dependent port variable at the port of the block connected to Port k, v^{c}_{hk}. As discussed concerning Equation (7) in an earlier section of this chapter, the term v^{v}_{hk} is used in Equation (19) since, due to generality, the port and block which are connected to Port k of Block h are undetermined. The variable v^{c}_{pq} in Equation (22) represents the variable shown as v_{rs} in Figure 5, and a_{pq} is ± 1 depending on a port variable sign convention.

Equation (18) defines the first-order time derivatives of the n_h state variables within Block h, and Equation (17) illustrates the m_h algebraic equations containing the m_h algebraic variables within Block h. Equation (17) and Figure 8 show that the independent port variables

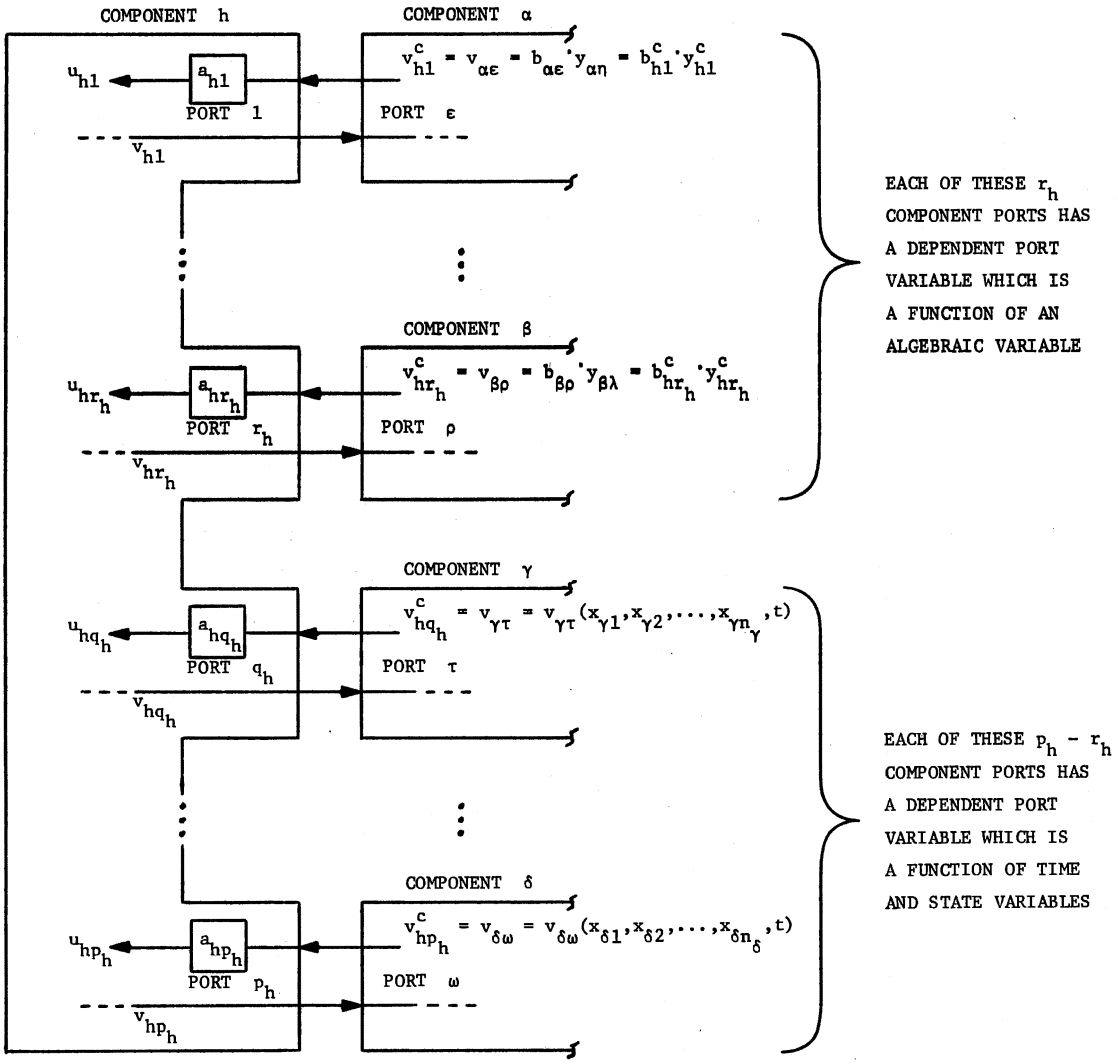


Figure 8. Connections Between Independent Port Variables of Component h and Dependent Port Variables of Other Components

within Block h may be divided into two groups. As shown in Figure 8, each of the first r_h independent port variables of Block h (u_{h1} through u_{hr_h}) is connected with a dependent port variable (v^c_{h1} through $v^c_{hr_h}$) of another block (α through β) and the dependent port variable is a function of an algebraic variable ($y_{\alpha\eta}$ through $y_{\beta\lambda}$) within the connected block. The remainder of the independent port variables, u_{hq_h} through u_{hp_h} (where $q_h = r_h + 1$), are connected with dependent port variables ($v^c_{hq_h}$ through $v^c_{hp_h}$) of other blocks (γ through δ) and are functions of time and state variables within the connected block.

The Implicit method may be derived for the block-oriented modeling equations by evaluating the derivatives of the algebraic functions in Equation (17) as

$$\frac{dF_{hi}}{dt} = \sum_{j=1}^{n_h} \frac{\partial F_{hi}}{\partial x_{hj}} \frac{dx_{hj}}{dt} + \sum_{j=1}^{m_h} \frac{\partial F_{hi}}{\partial y_{hj}} \frac{dy_{hj}}{dt} + \sum_{j=1}^{r_h} \frac{\partial F_{hi}}{\partial u_{hj}} \frac{du_{hj}}{dt}$$

$$+ \sum_{\ell=q_h}^{p_h} \frac{\partial F_{hi}}{\partial u_{h\ell}} \frac{du_{h\ell}}{dt} + \frac{\partial F_{hi}}{\partial t}, \quad i = 1, 2, \ldots, m_h. \tag{22}$$

The chain rule is used to expand the terms in the third summation group of Equation (22) and yields

$$\frac{\partial F_{hi}}{\partial u_{hj}} \frac{du_{hj}}{dt} = \frac{\partial F_{hi}}{\partial u_{hj}} \frac{\partial u_{hj}}{\partial v^c_{hj}} \frac{\partial v^c_{hj}}{\partial y^c_{hj}} \frac{dy^c_{hj}}{dt}, \tag{23}$$

where the dependent port variable of the port which is connected to Port j of Block h is a function of the algebraic variable y^c_{hj}. (In other words, the dependent port variable of Block α in Figure 8 is shown as $y_{\alpha\eta}$, but the term y^c_{h1} will be used for $y_{\alpha\eta}$ when it is unimportant or impossible to determine which port is connected to Port 1 of Block h.)

Equation (23) may also be written as

$$\frac{\partial F_{hi}}{\partial u_{hj}} \frac{du_{hj}}{dt} = \frac{\partial F_{hi}}{\partial y^c_{hj}} \frac{dy^c_{hj}}{dt} , \tag{24}$$

where

$$\frac{\partial F_{hi}}{\partial y^c_{hj}} = \frac{\partial F_{hi}}{\partial u_{hj}} \frac{\partial u_{hj}}{\partial v^c_{hj}} \frac{\partial v^c_{hj}}{\partial y^c_{hj}} . \tag{25}$$

Equations (23) and (24) demonstrate that the algebraic equations of Block h, Equation (17), are implicitly (but not explicitly) a function of algebraic variables within blocks other than h. This is accomplished by the independent port variables $u_{h1}, u_{h2}, \ldots, u_{hr_h}$ in Equation (17) and Equations (19) and (21).

The chain rule may also be used to expand the terms in the fourth summation group of Equation (22) as

$$\frac{\partial F_{hi}}{\partial u_{h\ell}} \frac{du_{h\ell}}{dt} = \frac{\partial F_{hi}}{\partial u_{h\ell}} \frac{\partial u_{h\ell}}{\partial v^c_{h\ell}} \frac{dv^c_{h\ell}}{dt} , \tag{26}$$

where, if Port e of Block f were connected to Port ℓ of Block h, $dv^c_{h\ell}/dt$ in Equation (26) may be evaluated by

$$\frac{dv^c_{h\ell}}{dt} = \frac{dv_{fe}}{dt} = \sum_{j=1}^{n_f} \frac{\partial v_{fe}}{\partial x_{fj}} \frac{dx_{fj}}{dt} + \frac{\partial v_{fe}}{\partial t} . \tag{27}$$

Since the right side of Equation (22) is equal to zero, Equations (22), (24) and (26) may be combined to form Equation (28) below. After evaluation of its terms, Equation (28) may be solved for the time derivatives of the algebraic variables which are then available to the numerical integration routine.

Figure 9 shows the steps required for applying the Implicit method to solve the block-oriented modeling equations. Step 1 shows that the

$$
\begin{bmatrix}
\frac{\partial F_{11}}{\partial y_{11}}\Big|_* & \frac{\partial F_{11}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{11}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{11}}{\partial y_{21}}\Big|_* & \frac{\partial F_{11}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{11}}{\partial y_{\ell m_\ell}}\Big|_* \\
\frac{\partial F_{12}}{\partial y_{11}}\Big|_* & \frac{\partial F_{12}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{12}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{12}}{\partial y_{21}}\Big|_* & \frac{\partial F_{12}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{12}}{\partial y_{\ell m_\ell}}\Big|_* \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
\frac{\partial F_{1m_1}}{\partial y_{11}}\Big|_* & \frac{\partial F_{1m_1}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{1m_1}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{1m_1}}{\partial y_{21}}\Big|_* & \frac{\partial F_{1m_1}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{1m_1}}{\partial y_{\ell m_\ell}}\Big|_* \\
\frac{\partial F_{21}}{\partial y_{11}}\Big|_* & \frac{\partial F_{21}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{21}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{21}}{\partial y_{21}}\Big|_* & \frac{\partial F_{21}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{21}}{\partial y_{\ell m_\ell}}\Big|_* \\
\frac{\partial F_{22}}{\partial y_{11}}\Big|_* & \frac{\partial F_{22}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{22}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{22}}{\partial y_{21}}\Big|_* & \frac{\partial F_{22}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{22}}{\partial y_{\ell m_\ell}}\Big|_* \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
\frac{\partial F_{\ell m_\ell}}{\partial y_{11}}\Big|_* & \frac{\partial F_{\ell m_\ell}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{\ell m_\ell}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{\ell m_\ell}}{\partial y_{21}}\Big|_* & \frac{\partial F_{\ell m_\ell}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{\ell m_\ell}}{\partial y_{\ell m_\ell}}\Big|_*
\end{bmatrix}
\begin{bmatrix}
\frac{d_{y11}}{dt} \\
\frac{dy_{12}}{dt} \\
\vdots \\
\frac{dy_{1m_1}}{dt} \\
\frac{dy_{21}}{dt} \\
\frac{dy_{22}}{dt} \\
\vdots \\
\frac{dy_{\ell m_\ell}}{dt}
\end{bmatrix}
=
\begin{bmatrix}
\left(-\sum_{j=1}^{n_1}\frac{\partial F_{11}}{\partial x_{1j}}\frac{dx_{1j}}{dt}-\sum_{k=q_1}^{p_1}\frac{\partial F_{11}}{\partial u_{1k}}\frac{\partial u_{1k}}{\partial v^c_{1k}}\frac{dv^c_{1k}}{dt}-\frac{\partial F_{11}}{\partial t}\right)\Big|_* \\
\left(-\sum_{j=1}^{n_1}\frac{\partial F_{12}}{\partial x_{1j}}\frac{dx_{1j}}{dt}-\sum_{k=q_1}^{p_1}\frac{\partial F_{12}}{\partial u_{1k}}\frac{\partial u_{1k}}{\partial v^c_{1k}}\frac{dv^c_{1k}}{dt}-\frac{\partial F_{12}}{\partial t}\right)\Big|_* \\
\vdots \\
\left(-\sum_{j=1}^{n_1}\frac{\partial F_{1m_1}}{\partial x_{1j}}\frac{dx_{1j}}{dt}-\sum_{k=q_1}^{p_1}\frac{\partial F_{1m_1}}{\partial u_{1k}}\frac{\partial u_{1k}}{\partial v^c_{1k}}\frac{dv^c_{1k}}{dt}-\frac{\partial F_{1m_1}}{\partial t}\right)\Big|_* \\
\left(-\sum_{j=1}^{n_2}\frac{\partial F_{21}}{\partial x_{2j}}\frac{dx_{2j}}{dt}-\sum_{k=q_2}^{p_2}\frac{\partial F_{21}}{\partial u_{2k}}\frac{\partial u_{2k}}{\partial v^c_{2k}}\frac{dv^c_{2k}}{dt}-\frac{\partial F_{21}}{\partial t}\right)\Big|_* \\
\left(-\sum_{j=1}^{n_2}\frac{\partial F_{22}}{\partial x_{2j}}\frac{dx_{2j}}{dt}-\sum_{k=q_2}^{p_2}\frac{\partial F_{22}}{\partial u_{2k}}\frac{\partial u_{2k}}{\partial v^c_{2k}}\frac{dv^c_{2k}}{dt}-\frac{\partial F_{22}}{\partial t}\right)\Big|_* \\
\vdots \\
\left(-\sum_{j=1}^{n_\ell}\frac{\partial F_{\ell m_\ell}}{\partial x_{\ell j}}\frac{dx_{\ell j}}{dt}-\sum_{k=q_\ell}^{p_\ell}\frac{\partial F_{\ell m_\ell}}{\partial u_{\ell k}}\frac{\partial u_{\ell k}}{\partial v^c_{\ell k}}\frac{dv^c_{\ell k}}{dt}-\frac{\partial F_{\ell m_\ell}}{\partial t}\right)\Big|_*
\end{bmatrix}
\tag{28}
$$

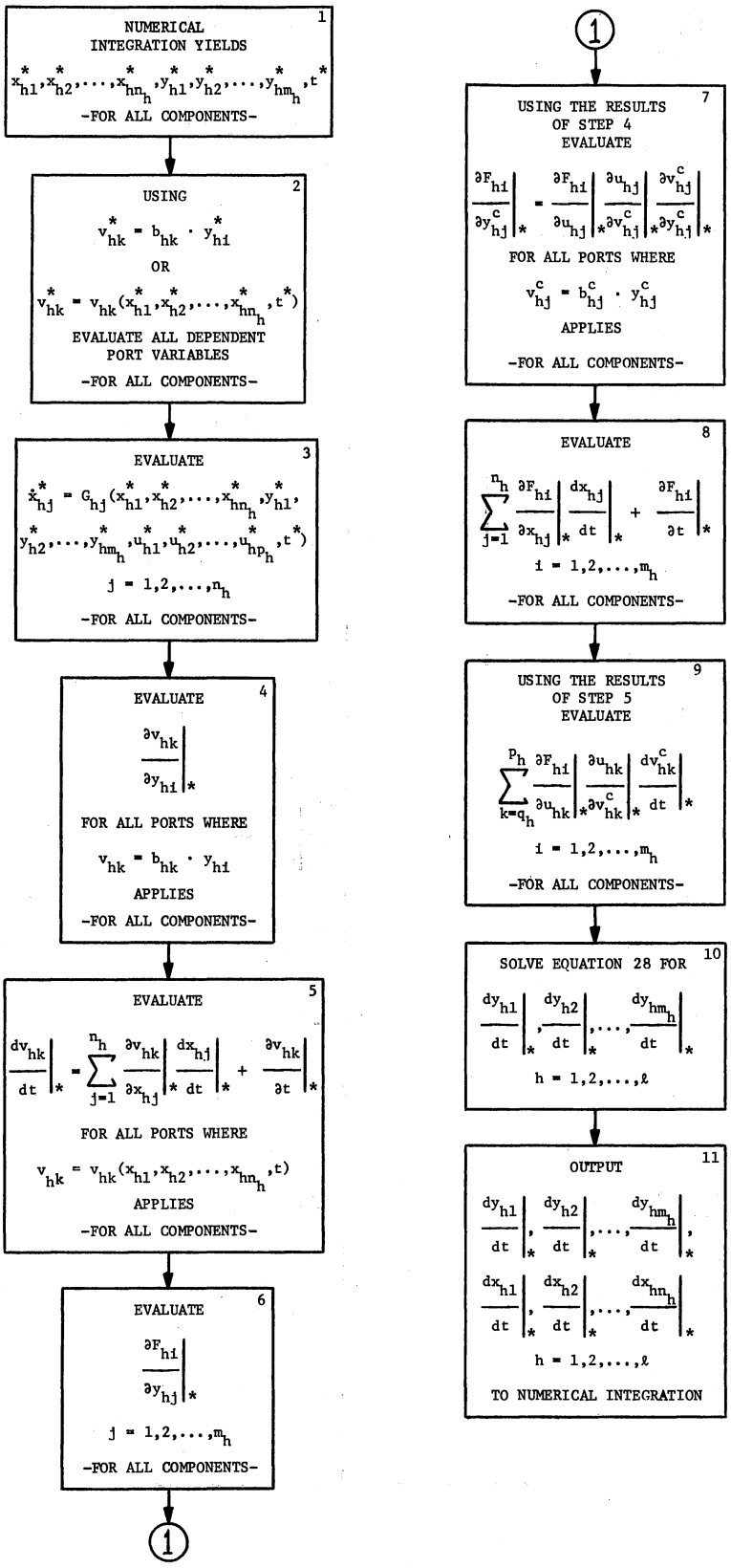


Figure 9. Algorithm to Solve Block-Oriented Modeling Equations by the Implicit Method

generalized integration method furnishes values for the state variables, the algebraic variables, and time. In Step 2 the dependent port variables are evaluated using Equation (20) or Equation (21) for each. Equation (18) is used to evaluate the time derivatives of the state variables in Step 3.

Step 4 illustrates the evaluation of $\partial v_{hk}/\partial y_{hi}$ for all dependent port variables which are only a function of an algebraic variable (as shown in Equation (21)). From Equation (21)

$$\left.\frac{v_{hk}}{y_{hi}}\right|_{*} = b_{hk}. \tag{29a}$$

Step 5 uses Equations (20) and (27) and the results of Step 3 to evaluate the time derivatives of all dependent port variables which are functions of time and state variables (as shown in Equation (20)).

Steps 6 and 7 are used to evaluate the terms of the first array in Equation (28). Step 6 evaluates partial derivatives of the algebraic functions (Equation (17)) with respect to all algebraic variables explicitly contained in the functions. Step 7 uses the results of Step 4 and Equations (17), (19) and (25) to evaluate the partial derivatives of the algebraic functions with respect to all algebraic variables not explicitly included in the functions.

Steps 8 and 9 illustrate the evaluation of the terms in the rightmost vector of Equation (28). Step 8 uses the results of Step 3 and Equation (17). Step 9 uses the results of Step 5 and Equations (17) and (19).

After the completion of Step 9, all terms are evaluated in Equation (28) except the time derivatives of the algebraic variables. Step 10 represents the solution of Equation (28) for these variables.

$$
\begin{bmatrix}
\frac{\partial F_{11}}{\partial y_{11}}\Big|_* & \frac{\partial F_{11}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{11}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{11}}{\partial y_{21}}\Big|_* & \frac{\partial F_{11}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{11}}{\partial y_{\ell m_\ell}}\Big|_* \\
\frac{\partial F_{12}}{\partial y_{11}}\Big|_* & \frac{\partial F_{12}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{12}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{12}}{\partial y_{21}}\Big|_* & \frac{\partial F_{12}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{12}}{\partial y_{\ell m_\ell}}\Big|_* \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
\frac{\partial F_{1m_1}}{\partial y_{11}}\Big|_* & \frac{\partial F_{1m_1}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{1m_1}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{1m_1}}{\partial y_{21}}\Big|_* & \frac{\partial F_{1m_1}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{1m_1}}{\partial y_{\ell m_\ell}}\Big|_* \\
\frac{\partial F_{21}}{\partial y_{11}}\Big|_* & \frac{\partial F_{21}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{21}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{21}}{\partial y_{21}}\Big|_* & \frac{\partial F_{21}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{21}}{\partial y_{\ell m_\ell}}\Big|_* \\
\frac{\partial F_{22}}{\partial y_{11}}\Big|_* & \frac{\partial F_{22}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{22}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{22}}{\partial y_{21}}\Big|_* & \frac{\partial F_{22}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{22}}{\partial y_{\ell m_\ell}}\Big|_* \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
\frac{\partial F_{\ell m_\ell}}{\partial y_{11}}\Big|_* & \frac{\partial F_{\ell m_\ell}}{\partial y_{12}}\Big|_* & \cdots & \frac{\partial F_{\ell m_\ell}}{\partial y_{1m_1}}\Big|_* & \frac{\partial F_{\ell m_\ell}}{\partial y_{21}}\Big|_* & \frac{\partial F_{\ell m_\ell}}{\partial y_{22}}\Big|_* & \cdots & \frac{\partial F_{\ell m_\ell}}{\partial y_{\ell m_\ell}}\Big|_*
\end{bmatrix}
\begin{bmatrix}
\frac{dy_{11}}{dt} \\
\frac{dy_{12}}{dt} \\
\vdots \\
\frac{dy_{1m_1}}{dt} \\
\frac{dy_{21}}{dt} \\
\frac{dy_{22}}{dt} \\
\vdots \\
\frac{dy_{\ell m_\ell}}{dt}
\end{bmatrix}
=
\begin{bmatrix}
\left(-\sum_{j=1}^{n_1}\frac{\partial F_{11}}{\partial x_{1j}}\frac{dx_{1j}}{dt}-\sum_{k=q_1}^{p_1}\frac{\partial F_{11}}{\partial u_{1k}}\frac{\partial u_{1k}}{\partial v^c_{1k}}\frac{dv^c_{1k}}{dt}-\frac{\partial F_{11}}{\partial t}\right)\Big|_* \\
\left(-\sum_{j=1}^{n_1}\frac{\partial F_{12}}{\partial x_{1j}}\frac{dx_{1j}}{dt}-\sum_{k=q_1}^{p_1}\frac{\partial F_{12}}{\partial u_{1k}}\frac{\partial u_{1k}}{\partial v^c_{1k}}\frac{dv^c_{1k}}{dt}-\frac{\partial F_{12}}{\partial t}\right)\Big|_* \\
\vdots \\
\left(-\sum_{j=1}^{n_1}\frac{\partial F_{1m_1}}{\partial x_{1j}}\frac{dx_{1j}}{dt}-\sum_{k=q_1}^{p_1}\frac{\partial F_{1m_1}}{\partial u_{1k}}\frac{\partial u_{1k}}{\partial v^c_{1k}}\frac{dv^c_{1k}}{dt}-\frac{\partial F_{1m_1}}{\partial t}\right)\Big|_* \\
\left(-\sum_{j=1}^{n_2}\frac{\partial F_{21}}{\partial x_{2j}}\frac{dx_{2j}}{dt}-\sum_{k=q_2}^{p_2}\frac{\partial F_{21}}{\partial u_{2k}}\frac{\partial u_{2k}}{\partial v^c_{2k}}\frac{dv^c_{2k}}{dt}-\frac{\partial F_{21}}{\partial t}\right)\Big|_* \\
\left(-\sum_{j=1}^{n_2}\frac{\partial F_{22}}{\partial x_{2j}}\frac{dx_{2j}}{dt}-\sum_{k=q_2}^{p_2}\frac{\partial F_{22}}{\partial u_{2k}}\frac{\partial u_{2k}}{\partial v^c_{2k}}\frac{dv^c_{2k}}{dt}-\frac{\partial F_{22}}{\partial t}\right)\Big|_* \\
\vdots \\
\left(-\sum_{j=1}^{n_\ell}\frac{\partial F_{\ell m_\ell}}{\partial x_{\ell j}}\frac{dx_{\ell j}}{dt}-\sum_{k=q_\ell}^{p_\ell}\frac{\partial F_{\ell m_\ell}}{\partial u_{\ell k}}\frac{\partial u_{\ell k}}{\partial v^c_{\ell k}}\frac{dv^c_{\ell k}}{dt}-\frac{\partial F_{\ell m_\ell}}{\partial t}\right)\Big|_*
\end{bmatrix}
\tag{29b}
$$

Step 11 ends the algorithm by illustrating that the time derivatives of the algebraic and state variables are made available to the numerical integration algorithm.

An Algorithm for Separating Algebraic Equations With Block-Oriented Organizations Into Separate Sets

In the previous sections of this chapter, all of the algebraic equations from a system have been considered to form one equation set. This assumption usually causes a large, sparse matrix to be formed during the solution of these equations which is similar to that formed in other solution methods (25, 26). However, unlike these other methods, the Newton-Raphson and Implicit methods explained in this study allow the set of algebraic equations to be separated into several smaller sets. This separation is usually possible for complex hydraulic systems and is highly desirable since the number of operations required to solve a set of equations increases approximately as the third power of the number of equations in the set assuming a method such as Gaussian elimination is used in the solution (31). For example, if a large set of equations is separated into two smaller sets of equal size, the number of operations required for a solution will be reduced by approximately 75 percent.

The method for separating a set of algebraic equations into several smaller sets is based only upon the coupling of algebraic variables between the equations. However, for modeling equations with block-oriented organization, it will be seen that the method is complicated by the fact

that independent port variables in the algebraic equations of one block may implicitly introduce algebraic variables from another block.

For a specific system, the separation process may be initiated by examining the blocks contained in the system and forming a list of all the algebraic equations contained in the system. (Algebraic equations may be identified by block number and algebraic equation number within the block.) For each algebraic equation entered into the equation list, a secondary list must be formed which identifies the algebraic variables explicitly contained in the algebraic equation. (Algebraic variables may be identified by block number and algebraic variable number within the block.) Algebraic variables which are implicitly contained in the algebraic equation must also be added to the secondary list. These may be located by checking the port of each independent port variable contained in the algebraic equation. If the port is connected to another port with a dependent port variable which is a function of an algebraic variable, this algebraic variable is implicitly contained in the algebraic equation under examination and must be added to the equation's secondary list. The remainder of the steps in the separation process are shown in Figure 10 and listed below.

1. Start a new set by transferring the first equation from the equation list into the new set (removing its entry in the equation list) and by examining its associated secondary list.

2. Search the secondary lists of the remaining equation list entries to identify any equations containing one or more of the algebraic variables in the secondary list placed under examination in Step 1 (or Step 3). Transfer any equations found with matching variables into the new set and remove their equation list entries.

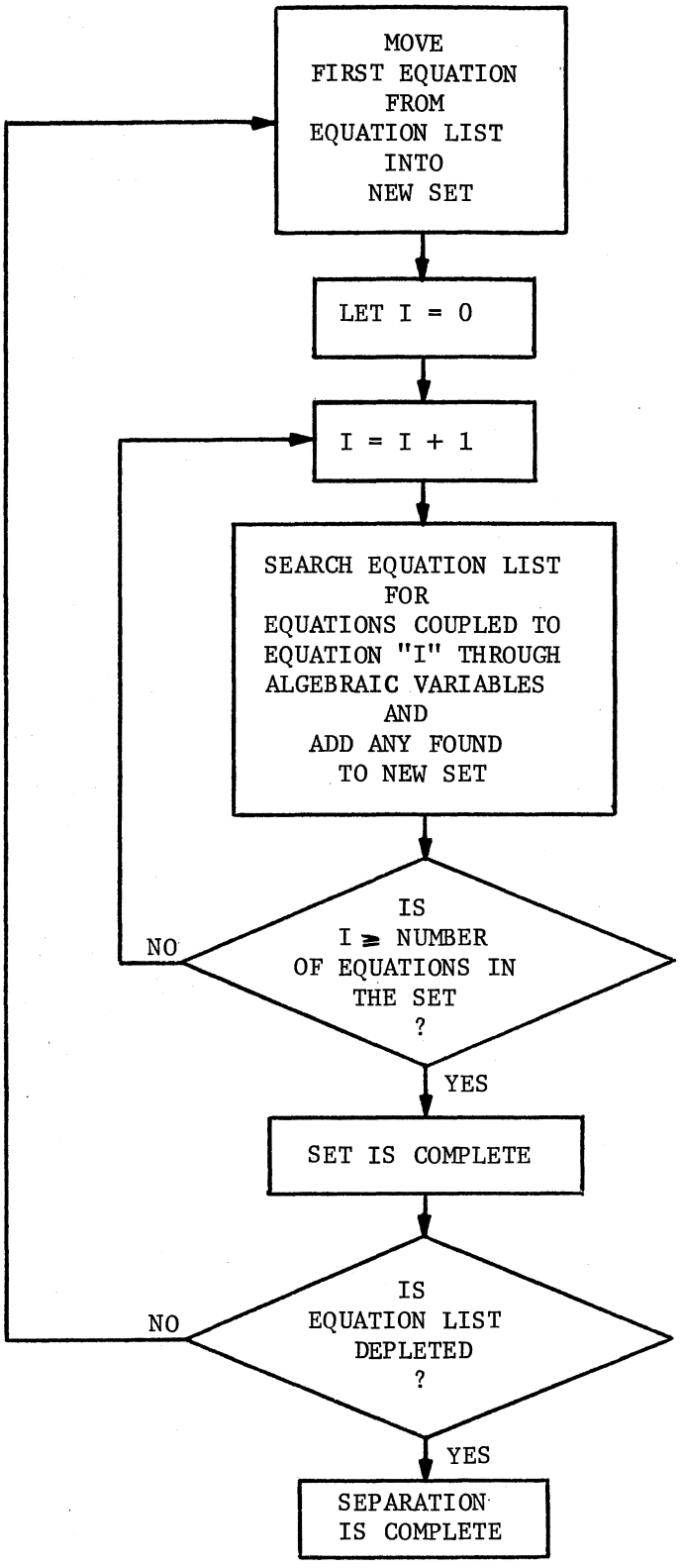


Figure 10. Algorithm for Separating Algebraic Equations Into Sets

3. Repeat Step 2 for the secondary list of each equation placed into the new set. When all secondary lists of the equations placed in the new set have been examined, the set is complete.

4. If any equation entries remain in the equation list, an additional set may be formed by starting with Step 1 and repeating the above. When no more entries remain in the equation list, the separation is complete.

A Combined Algorithm for the Solution of Modeling Equations With Block-Oriented Organization

An algorithm may be formed for the solution of block-oriented modeling equations by combining the solution methods shown in Figures 6 and 9. Some advantages of the resulting algorithm are:

1. The Newton-Raphson technique may be used to calculate the initial values of all the algebraic variables.

2. The algorithm may be structured to solve modeling equations containing multiple sets of algebraic equations resulting from the application of the separation method shown in Figure 10.

3. After initialization any algebraic set of equations may be switched from the Newton-Raphson solution method to the Implicit method or back to the Newton-Raphson method at any point in time without affecting the solution of any other algebraic set.

As shown in Figure 11, the combined algorithm receives specific values of time, the state variables, and any algebraic variables included in algebraic equation sets to be solved by the Implicit method (hereafter called Implicit variables). After receiving these specific variable

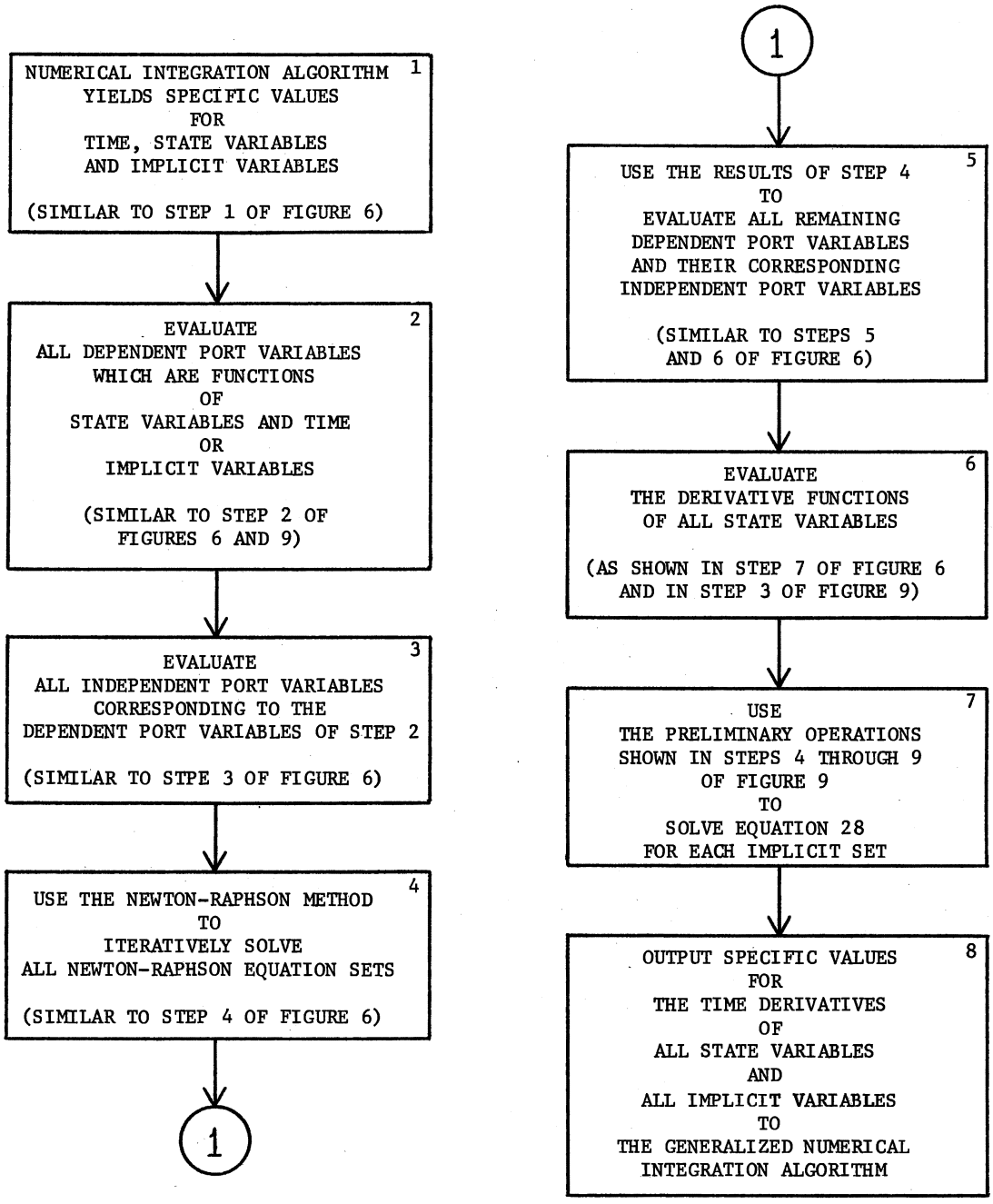


Figure 11. Algorithm for Combining Implicit and Newton-Raphson Solution Methods

values, the combined algorithm uses the methods of Figure 6 to solve for specific values of any algebraic variables included in algebraic sets not to be solved by the Implicit method (hereafter called Newton-Raphson variables). At this point, specific values are known for all algebraic variables and the methods of Figure 9 may be used to determine the time derivatives of all Implicit variables. The passing of these derivative values and those of the state variables to the generalized integration method completes the combined algorithm.

The above algorithm must be modified slightly when used in conjunction with a generalized integration method which requires past values of derivatives of the variables being integrated. The explicit Adams integration methods are methods of this type (31). As shown in Figure 11, the combined algorithm will not furnish to the numerical integration algorithm derivative values of algebraic variables included in equation sets being solved by the Newton-Raphson technique. Consequently, the integration algorithm will not contain past derivative values for any Newton-Raphson variables if it should be desired to convert a set of algebraic equations from Newton-Raphson to Implicit solution. However, this problem may be remedied by performing Steps 7 and 8 of the combined algorithm for any set being solved by the Newton-Raphson method which may later be solved by the Implicit method.

CHAPTER IV

APPLICATION OF SIMULATION ALGORITHMS

Some of the numerical algorithms and the computer program organization discussed above have been implemented in the HYDSIM (HYDraulic System SIMulation) program. HYDSIM is a transient analysis program which was primarily designed for the simulation of complex hydraulic systems. However, it may be used to simulate any type of continuous system which may be modeled by sets of nonlinear algebraic and differential equations. This chapter introduces the program and presents an example simulation problem.

HYDSIM - A Block-Oriented Simulation Program for Hydraulic Systems

HYDSIM is an interpreter type of computer program which uses a stored library of component models in conjunction with optional user-supplied models to simulate a system. The program is written in FORTRAN IV. The program controls all branching between component models, and the HYDSIM user is not required to write any FORTRAN programs unless he is furnishing a special component model. In addition, no special ordering of the input data entires is required to obtain a correct solution.

No emphasis has been placed on the solution of any one type of component or component model in the HYDSIM program. Instead, an emphasis has been placed on treating a group of components as a system. All

components in a hydraulic circuit are treated similarly and all modeling equations are solved in a simultaneous manner without special constraints on allowable component configurations.

The HYDSIM program contains a library of standard component models such as valves, accumulators, cylinders, loads, etc. (1). The models have been developed for these components using multiport modeling techniques and all significant nonlinearities have been preserved. If the library components are not adequate for a simulation, the program user may code special component models (each in a FORTRAN IV subroutine) and add them to the program. These component subroutines may be added to HYDSIM temporarily by including them with the program input data or added permanently by program modification.

Two types of block-oriented input data formats are accepted by HYDSIM. The fixed-format type of input is processed very rapidly by HYDSIM, but the user must take care to place entries in the proper card columns. The free-format type of input allows the user to place circuit description entries in any card columns and in any order. Both types of input data are subjected to an error analysis to check for parameter and circuit description errors. HYDSIM also has a rerun capability and will accept simulation data in a batch mode.

Program output consists of printer plots and tabulated values of user specified circuit variables.

Multiple solution methods are available in HYDSIM. The methods use a fourth-order Runge-Kutta or Adams-Moulton method to solve the differential equations and the Newton-Raphson or Implicit method to solve the algebraic equations. One of three combinations of these methods may be specified by an input entry as:

1. Runge-Kutta and Newton-Raphson.
2. Adams-Moulton and Newton-Raphson.
3. Adams-Moulton and Implicit.

All of the solution methods solve the modeling equations in a simultaneous manner without making any unusual approximations and without using variable values from a previous point in time when a current value should be used.

The Adams-Moulton and Implicit solution method used by the HYDSIM program incorporates the combined solution algorithm discussed in the previous chapter. The algorithm is programmed to determine the initial conditions and perform the first three time steps using the Newton-Raphson technique and fourth-order Runge-Kutta integration. After variable values have been computed at these first four points in time, all integration is performed by a fourth-order Adams-Moulton integration method and the algebraic equation sets are switched to the Implicit solution method. The HYDSIM program also monitors the accuracy of the Implicit solution of each algebraic set and returns any set being solved inaccurately to the Newton-Raphson solution method until accurate Implicit solution may be obtained.

The changing of solution algorithms for a set of algebraic equations requires methods for determining when to change. Since the Implicit algorithm uses a fourth-order Adams-Moulton predictor-corrector numerical integration algorithm to solve for the algebraic variables, the amount of truncation error occurring in a solution may be indicated by comparing the predicted and the corrected values for each variable. If these differ by more than a fixed amount (see the "HYDSIM User's Manual" (1) for a more complete discussion of the error criteria), the algebraic

set containing the erring variable is changed to the Newton-Raphson algorithm for solution using the predicted values as starting values for the iteration. An algebraic set which has been switched to the Newton-Raphson solution algorithm will continue to use this algorithm for solution until each of several consecutive Newton-Raphson solutions (usually two) has been completed in less than a fixed number of iterations. At this point it is assumed that the algebraic set may again be solved accurately by the Implicit algorithm and the set is returned to Implicit solution.

HYDSIM Simulation of a Fuel-Injection System

As an example of the use of the algorithms discussed in the previous chapter, the HYDSIM program has been used to simulate the mechanical and hydraulic parts of the fuel-injection system shown in Figure 12. This system would be used on a four-cylinder automotive type of engine and contains four injectors which are actuated in pairs. Injectors 1 and 3 are actuated first followed by 2 and 4. This simulation could be used to study the possibility of fuel starvation at the injectors. Starvation might occur because of the differing methods of pairing the injectors for fuel supply and actuation.

The block diagrams which were used to develop the HYDSIM simulation of the fuel-injection system are shown in Figures 13, 14 and 15. Figure 13 shows the blocks used for each injector, Figure 14 shows the blocks used for the regulator, and Figure 15 shows the remainder of the system. All blocks were represented by standard HYDSIM library component models

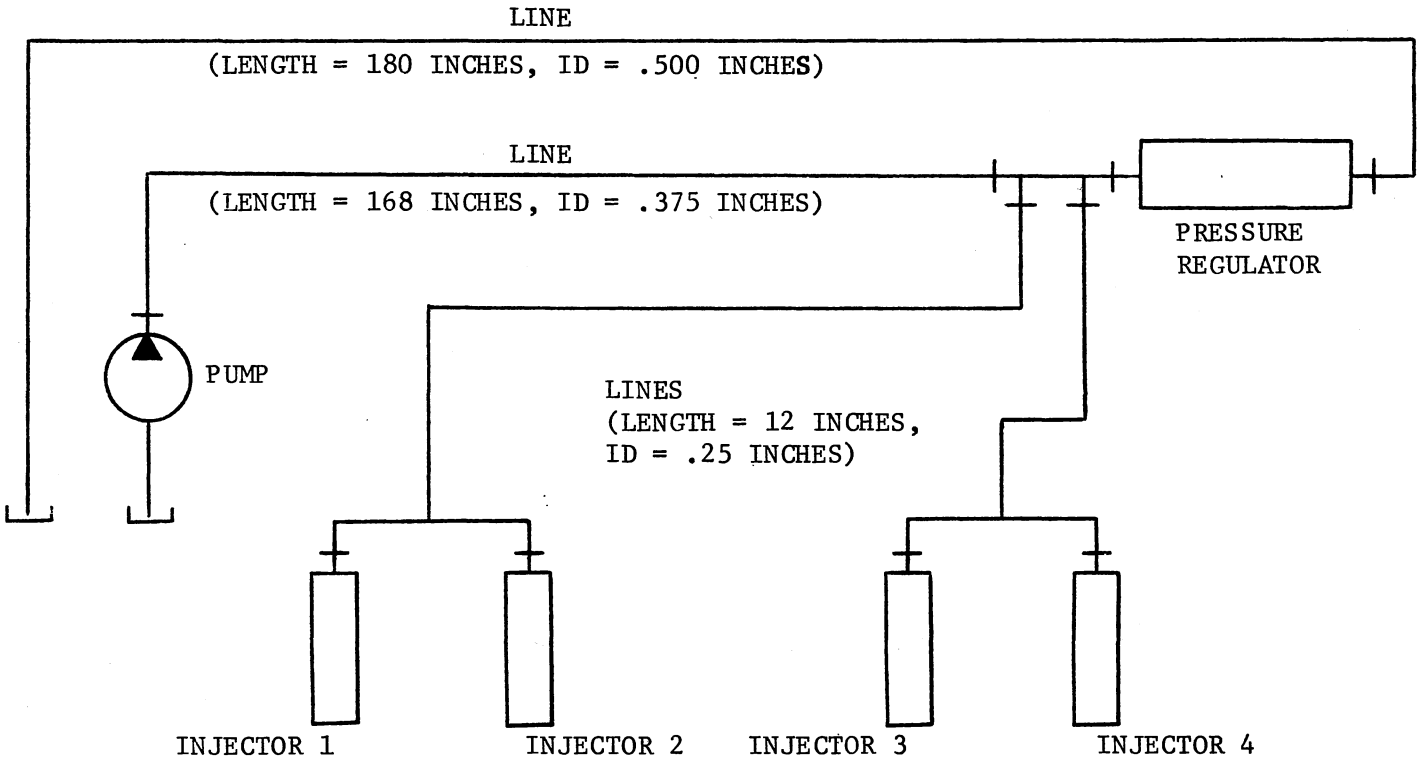


Figure 12. Diagram of Fuel Injection System

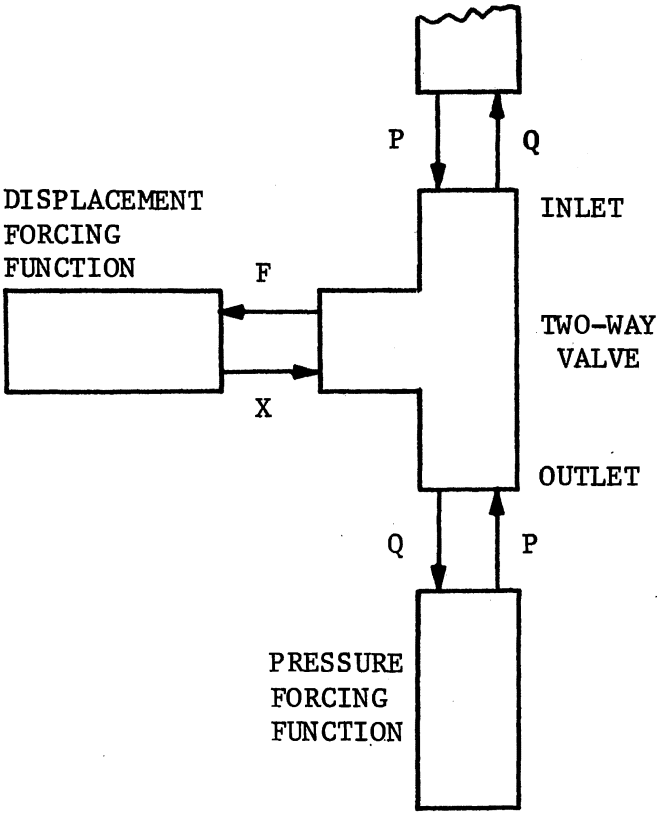


Figure 13. HYDSIM Representation of Fuel Injector

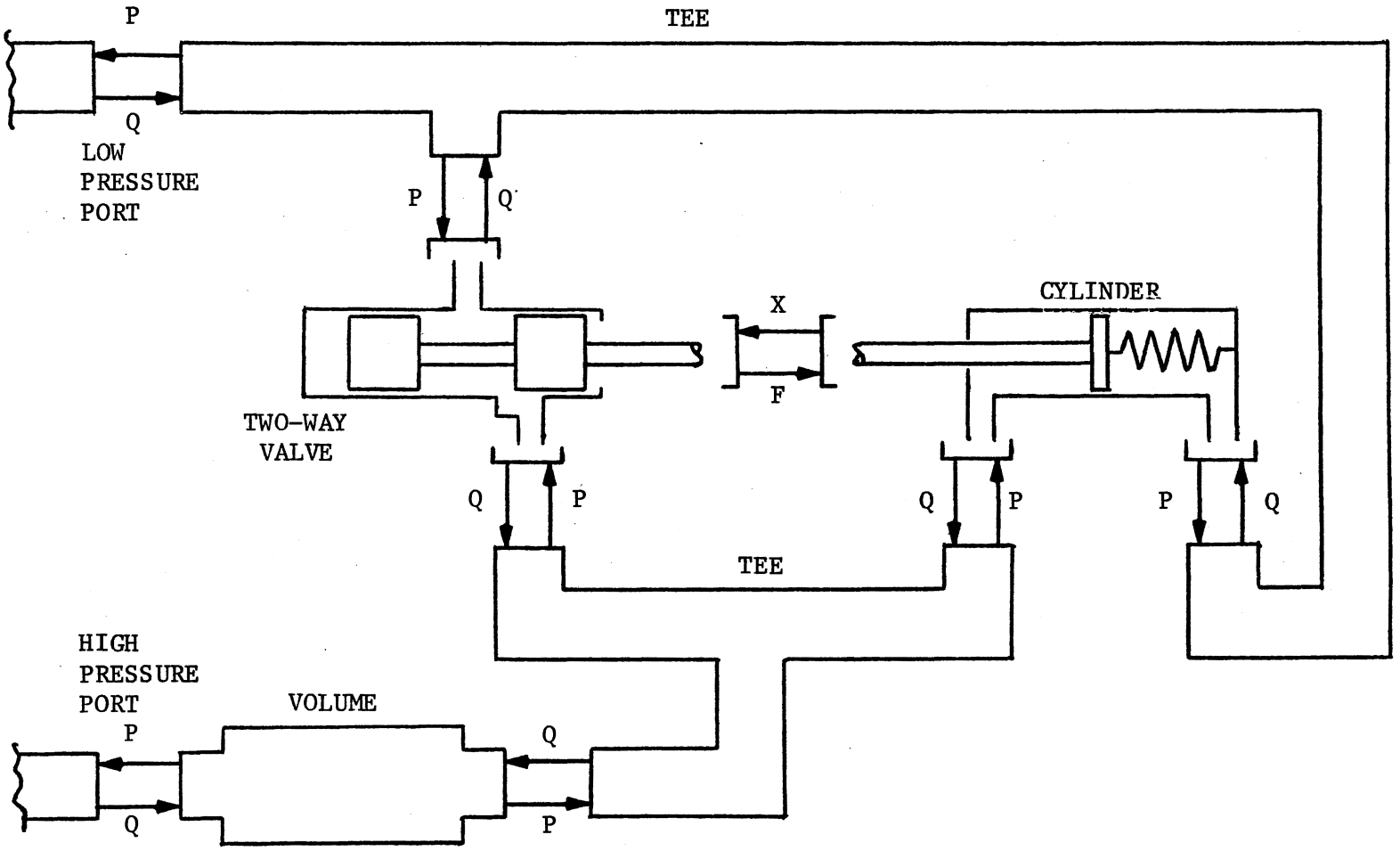


Figure 14. HYDSIM Representation of a Simplified Fuel Pressure Regulator

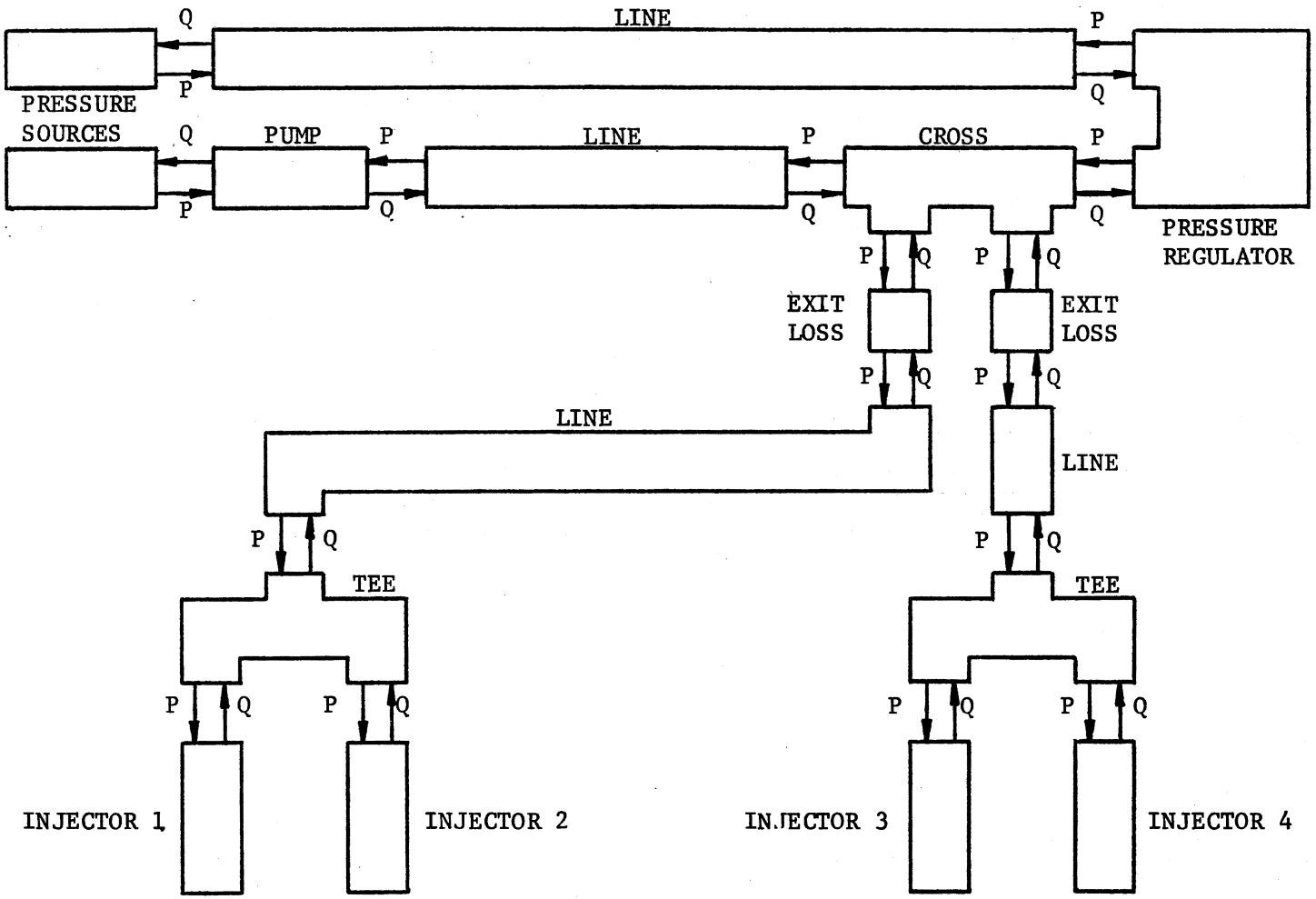


Figure 15. HYDSIM Representation of a Fuel-Injection System

except the displacement forcing function shown in Figure 13 which was programmed for this simulation.

The HYDSIM simulation of the fuel injection system included 26 first-order differential equations and 27 algebraic equations. The algebraic equations were separated into 5 sets of 1, 6, 6, 6, and 8 equations.

Figure 16 shows a graph of the pressure at the inlet of Injector 2 during the first eight milliseconds of time after system start-up from a condition of steady regulator flow with no injector actuation. Appendix C shows a copy of the HYDSIM input for this simulation and additional graphs of system variables.

During the simulation, integration was performed with a step size of 0.00001 second and a total of 800 steps. A group of 15 pairs of port variable values were printed out at every tenth integration step with a total of 81 groups printed.

Three simulation runs were performed using three different solution methods and the computer execution time was recorded for each. A fourth run was performed for one integration step to evaluate initialization time. All runs were performed separately on an IBM 370 Model 155 computer with no other jobs executing in the computer system. Execution times were: 0.06 seconds for initialization, 2.26 seconds for the Runge-Kutta and Newton-Raphson solution methods, 1.22 seconds for the Adams-Moulton and Newton-Raphson solution method, and 0.95 seconds for the Adams-Moulton and Implicit solution method. For the latter method, error monitoring of the Implicit method resulted in algebraic equation sets being reset nineteen times to the Newton-Raphson solution method. The Implicit method was used for 98 percent of the algebraic equation set solutions. Solution values from the three solution methods were

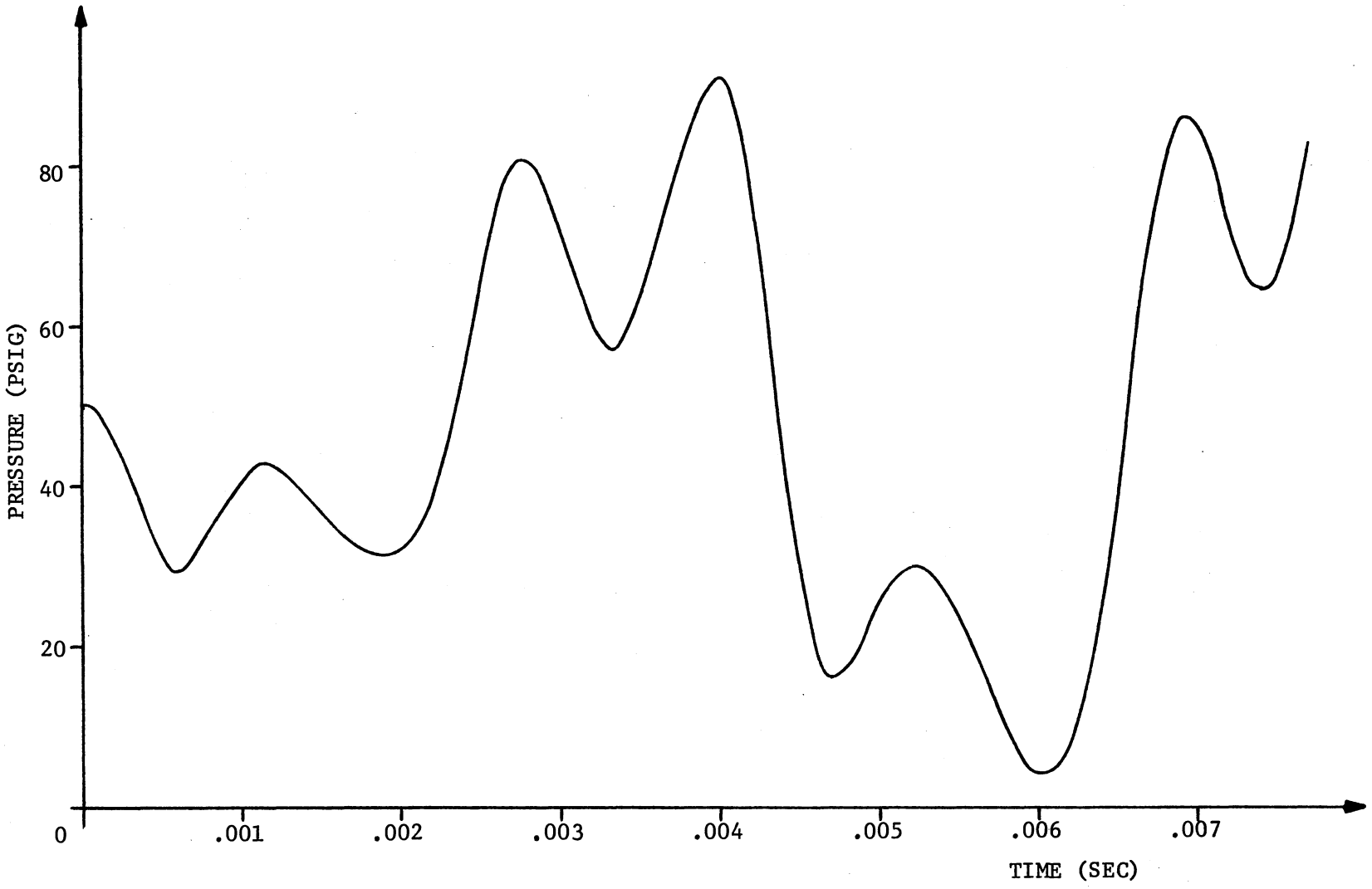


Figure 16. Pressure at Injector Inlets

comparable. The solutions were usually identical to four digits of accuracy (all that were printed out) with all values agreeing within ± 0.5 percent except when magnitudes of the compared values were much smaller than one.

Discussion of Results

The results of the application of the algorithms contained in this study for the simulation of complex systems have been illustrated by the simulation of an automotive fuel-injection system. Three different solution methods were used to produce equivalent simulation runs for this example, and a comparison of the results of these runs will illustrate the capabilities of the solution algorithms used.

The fuel-injection system simulation used ten component model blocks from the HYDSIM library and resulted in the solution of 26 differential equations and 27 algebraic equations. The algebraic equation separation algorithm separated these algebraic equations into five sets with each containing from one to eight equations. The set of eight contained equations from four different components and each of the three sets of six contained equations from three different components. Each set was solved simultaneously by the solution algorithms.

The fuel-injection example illustrates one of the major advantages of the block-oriented algorithms given in this study. These algorithms allow various component models to be programmed individually and then connected for a simulation in any manner which matches port variables. Therefore, many simple components may be programmed for the HYDSIM component library and later connected in a simulation to form a more complex

component as illustrated by the fuel-pressure regulator shown in Figure 14.

A comparison of the computer time required for each of the simulation runs indicates the relative speeds of the algorithms. However, some problems exist in determining how much computer time was spent during each run on solving the algorithms and how much time was spent doing other things such as printing.

The computer used for the simulation runs was an IBM 370-155 operated in a multiprogrammed environment (IBM MFT), but the simulation runs were performed one at a time with no other jobs executing in the computer system. Consequently, execution time spent on computer interrupts was minimized. In an attempt to evaluate the amount of computer execution time required during each run by the IBM Operating System and required by HYDSIM for initialization, a fourth simulation run was made which performed initialization and one integration step (compared to 800 steps for the three comparison runs). Subtracting this execution time from the time of the three comparison runs and comparing the results yields:

1. The Adams-Moulton-Newton-Raphson solution method required 30 percent more execution time than the Adams-Moulton-Implicit method.

2. The Runge-Kutta-Newton-Raphson solution method required 147 percent more execution time than the Adams-Moulton-Implicit method.

The above comparisons would show an increased spread in performance if printing time (the same for each of the three runs) could also be evaluated and subtracted from the execution times.

As would be expected due to the fewer number of derivative evaluations required per time step, the solution methods employing the Adams-Moulton integration method were faster than the method employing

Runge-Kutta when equal integration step sizes were used. (No comparison was attempted for different step sizes.) Of the two runs using the Adams-Moulton integration method, the run using the Newton-Raphson method for solution of the algebraic equations required 30 percent more execution time than the method using the Implicit solution method. It should also be noted that similar execution time comparisons have been noted during HYDSIM simulations of other systems.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

The Implicit method has been presented as a noniterative method of solving nonlinear algebraic equations which are coupled with differential equations, and the method has been incorporated into the computer program HYDSIM, a transient analysis program for hydraulic systems. The Implicit method utilizes a "conventional" numerical integration algorithm to solve the nonlinear algebraic equations in a manner similar to the differential equations. The method uses the single solution of a linear set of algebraic equations to evaluate each set of algebraic variable derivative values required by the integration algorithm. Since this linear equation is very similar to the linear equation which must be used iteratively when the Newton-Raphson method is employed, the Newton-Raphson method may easily be added to the Implicit method for calculating the initial values of the algebraic variables.

Comparative simulation runs of the HYDSIM program for an example have shown the Implicit method to be 30 percent faster than the Newton-Raphson method and to have an equal accuracy in most cases. This agreement in accuracy might have been expected since the Implicit method uses an exact method to evaluate algebraic derivative values and produces no truncation error except that incorporated in the integration algorithm.

Further investigation may show that the Implicit error monitoring scheme employed in HYDSIM is unnecessary.

The HYDSIM program has been developed as an engineering tool to reduce the effort which was previously required to simulate the response of a multicomponent hydraulic system. The program contains a library of previously programmed hydraulic component models which the program user may use in any connection configuration (port variables must match at connections). In addition, the HYDSIM program user may add component models to the program which have been represented by coupled algebraic and ordinary differential equations. The program has a block-oriented input format and will accept fixed or free format data.

Although the HYDSIM program and the Implicit method have been discussed for use in the simulation of hydraulic systems, their use need not be limited to this. The Implicit method may be used to solve any set of algebraic equations as the independent variable is varied if any or all of the equations in the set are an explicit function of the independent variable. In addition, the Implicit method may be used to solve any sets of coupled algebraic and differential equations as the independent variable is varied. However, it should be noted that any application of the Implicit method assumes that all required partial derivatives exist and may be evaluated for all required values of the independent variable.

Recommendations for Further Study

One could spend a lifetime making improvements in a large computer program such as HYDSIM and in its associated algorithms. However, some of the areas in which improvements would be most beneficial are:

1. Develop a simulation algorithm which does not require the matching of port-variable dependencies at the component connections.

2. Perform any modifications which would decrease the time required to execute the simulation algorithms without reducing their flexibility nor accuracy.

Some methods for reducing algorithm execution time might be: (1) removing the Implicit method error monitor; (2) implementing sparse matrix methods for the solution of linear algebraic equations; and (3) eliminating the double subscripting method shown in the block-oriented modeling equations and used in HYDSIM. The first two methods given above may not be appropriate and further investigations will have to be performed to justify these changes.

BIBLIOGRAPHY

(1) Smith, C. K. "HYDSIM User's Manual." Stillwater: School of Mechanical and Aerospace Engineering, Oklahoma State University, August, 1973. (Unpub. notes.)

(2) Selfridge, R. G. "Coding a General Purpose Digital Computer to Operate as a Differential Analyzer." Proceedings 1955 Western Joint Computer Conference (IRE), Vol. 8 (1955), 82-84.

(3) Stein, M. L., J. Rose, and D. B. Parker. "A Compiler With an Analog-Oriented Input Language." Proceedings 1959 Western Joint Computer Conference, Vol. 16 (1959), 92-102.

(4) Hurley, J. R. and J. J. Skiles. "DYSAC: A Digitally Simulated Analog Computer." Proceedings 1963 Spring Joint Computer Conference, Vol. 23 (1963), 69-82.

(5) Stover, R. F. and H. A. Knudston. "H-800 PARTNER-Proof of Analog Results Through a Numerically Equivalent Routine." Minneapolis: Honeywell Scientific Computing Aeronautical Division, Document No. U-ED 15002, April, 1962.

(6) Gaskill, R. A., J. W. Harris, and A. L. McKnight. "DAS--A Digital Analog Simulator." Proceedings Spring Joint Computer Conference, Vol. 23 (1963), 83-90.

(7) Rideout, V. C. and L. Tavernini. "MADBLOC, A Program for Digital Simulation of a Hybrid Computer." Simulation, Vol. 4 (January, 1965), 21-25.

(8) Harnett, R. T., F. J. Sansom, and L. M. Warshawsky. "MIDAS--An Analog Approach to Digital Computation." Simulation, Vol. 3 (September, 1964), 17-43.

(9) Brennan, R. D. and H. Sano. "PACTOLUS--A Digital Analog Simulator Program for the IBM 1620." Proceedings 1964 Fall Joint Computer Conference, Vol. 26 (1964), 299-312.

(10) Peterson, H. E. and F. J. Sansom. "MIMIC--A Digital Simulator Program." Wright-Patterson AFB, Ohio: SESCA Internal Memo 65-12, May, 1965.

(11) Syn, W. M. and R. N. Linebarger. "DSL/90--A Digital Simulation Program for Continuous System Modeling." Proceedings 1966 Spring Joint Computer Conference, Vol. 28 (1966), 165-187.

(12) 1130 Continuous System Modeling Program (1130-CX-13X). IBM Report No. H20-0209-1. White Plains, N.Y.: IBM Corp., 1966.

(13) System/360 Continuous System Modeling Program (360A-CX-16X). IBM Report No. H20-0367-0. White Plains, N.Y.: IBM Corp., 1967.

(14) Brennan, R. D. and R. N. Linebarger. "A Survey of Digital Simulation Languages." Simulation, Vol. 3 (December, 1964), 22-36.

(15) Clancy, J. J. and M. S. Fineberg. "Digital Simulation Languages: A Critique and a Guide." Proceedings 1965 Fall Joint Computer Conference, Vol. 27 (1965), 23-36.

(16) Stein, M. L. and J. Rose. "Changing from Analog to Digital Programming by Digital Techniques." ACM Journal, Vol. 7 (January, 1960), 10-23.

(17) Rosenberg, R. C. "A User's Guide to ENPORT-4." East Lansing, Mich.: Michigan State University, Division of Engineering Research, June, 1972.

(18) Paynter, H. M. Analysis and Design of Engineering Systems. Cambridge: The MIT Press, 1961.

(19) The 1620 Electronic Circuit Analysis Program (ECAP) (1620-EE-02X). IBM Report No. H20-0170. White Plains, N.Y.: IBM Corp., 1965.

(20) Lindgren, A. G. "Computer-Aided Circuit Design: A User's Viewpoint." Simulation, Vol. 16 (May, 1971), 195-206.

(21) Waterman, A. W., A. K. Trikha and K. D. Groom. Aircraft Hydraulic System Dynamics. USAF Technical Report No. AFAPL-TR-73-2. Wright-Patterson AFB, Ohio: Air Force Aero Propulsion Laboratory, February, 1973. (Report submitted to Air Force Aero Propulsion Laboratory, Wright-Patterson AFB under Contract No. F33615-72-C-1699 by Boeing Commercial Airplane Company.)

(22) Zielke, W. "Digital Simulation of Airplane Hydraulic Systems." ASME Paper No. 71-WA/FE-21, November, 1971. (Paper was presented at the 1971 ASME Winter Annual Meeting.)

(23) Smith, C. and R. P. Braden. "Hydraulic System Simulation." Stillwater: School of Mechanical and Aerospace Engineering, Oklahoma State University, December, 1970. (Report Submitted to General Dynamics Corp., Convair Aerospace Division, Fort Worth, Texas.)

(24) Smith, C. K. "HYDSIM--A Generalized Hydraulic System Simulation Program." Proceedings Second Southwestern Graduate Research Conference in Applied Mechanics, Vol. 2 (March, 1971), 73-75.

(25) Gear, C. W. "The Automatic Integration of Stiff Ordinary Differential Equations." Information Processing 68. Edited by A. J. H. Morrell. The Netherlands: North Holland Publishing Co., 1969, 187-193.

(26) Brayton, R. K., F. G. Gustavson and G. D. Hacktel. "The Use of Variable-Order Variable-Step Backward Differentiation Methods for Nonlinear Electrical Networks." Mem. Mexico 1971 International IEEE Conference on Systems, Networks, and Computers, Oaxtepec, Mexico, 1971, 102-106.

(27) Merritt, H. E. Hydraulic Control Systems. New York: John Wiley & Sons, Inc., 1967, 52.

(28) Merritt, H. E. Hydraulic Control Systems. New York: John Wiley & Sons, Inc., 1967, 83.

(29) Soloknikoff, I. S. and R. M. Redheffer. Mathematics of Physics and Modern Engineering. New York: McGraw-Hill Book Co., 1958, 708.

(30) Derusso, P. M., R. J. Roy and C. M. Close. State Variables for Engineers. New York: John Wiley & Sons, Inc., 1965.

(31) Ralston, A. A First Course in Numerical Analysis. New York: McGraw-Hill Book Co., 1965.

(32) James, M. L., G. M. Smith and J. C. Wolford. Analog and Digital Methods in Engineering Analysis. Scranton: International Textbook Co., 1964, 259-262.

(33) Ostrowski, A. M. Solutions of Equations and Systems of Equations. New York: Academic Press, Inc., 1960, 43-50.

APPENDIX A

THE NEWTON-RAPHSON METHOD

The Newton-Raphson method can be used to solve a set of algebraic equations. When applied to a set of nonlinear algebraic equations, the method effectively substitutes the iterative solution of a set of linear equations for the more difficult task of solving the nonlinear equations. The method requires an initial set of approximations to the solution set.

The Newton-Raphson method can be derived from a truncated Taylor series expansion of the algebraic equations about an approximate solution point of the equations. Given a set of m algebraic equations of the form

$$0 = F_i\ (y_1, y_2, \ldots, y_m),\ i = 1, 2, \ldots, m \tag{30}$$

and the approximate solution $y_1^o, y_2^o, \ldots, y_m^o$, Equation (30) may be expanded in the Taylor series

$$\begin{aligned} F_i\ (y_1, y_2, \ldots, y_m) &= F_i\ (y_1^o, y_2^o, \ldots, y_m^o) \\ &+ \sum_{j=1}^{m} \left.\frac{\partial F_i}{\partial y_j}\right|_{y^o} (y_j - y_j^o) \\ &+ \sum_{j=1}^{m} \sum_{k=1}^{m} \left.\frac{\partial^2 F_i}{\partial y_j\, \partial y_k}\right|_{y^o} (y_j - y_j^o)(y_k - y_k^o) + \ldots, \\ &i = 1, 2, \ldots, m, \end{aligned} \tag{31}$$

where the bar and y^o to the right of partial derivative implies that the derivatives are to be evaluated at the approximate solution $y_1^o, y_2^o, \ldots, y_m^o$. The truncation of the series in Equation (31) to retain

only the linear terms and the combination of this with Equation (30) yields

$$0 = F_i\,(y_1^o, y_2^o, \ldots, y_m^o) + \sum_{j=1}^{m} \left.\frac{\partial F_i}{\partial y_j}\right|_{y^o} (y_j - y_j^o),$$

$$i = 1, 2, \ldots, m. \tag{32}$$

Equation (32) may be rearranged and written in vector-matrix form as

$$\begin{bmatrix} \left.\frac{\partial F_1}{\partial y_1}\right|_{y^o} & \left.\frac{\partial F_1}{\partial y_2}\right|_{y^o} & \cdots & \left.\frac{\partial F_1}{\partial y_m}\right|_{y^o} \\ \left.\frac{\partial F_2}{\partial y_1}\right|_{y^o} & \left.\frac{\partial F_2}{\partial y_2}\right|_{y^o} & \cdots & \left.\frac{\partial F_2}{\partial y_m}\right|_{y^o} \\ \vdots & \vdots & & \vdots \\ \left.\frac{\partial F_m}{\partial y_1}\right|_{y^o} & \left.\frac{\partial F_m}{\partial y_2}\right|_{y^o} & \cdots & \left.\frac{\partial F_m}{\partial y_m}\right|_{y^o} \end{bmatrix} \begin{bmatrix} (y_1 - y_1^o) \\ (y_2 - y_2^o) \\ \vdots \\ (y_m - y_m^o) \end{bmatrix} = \begin{bmatrix} -F_1\,(y_1^o, y_2^o, \ldots, y_m^o) \\ -F_2\,(y_1^o, y_2^o, \ldots, y_m^o) \\ \vdots \\ -F_m\,(y_1^o, y_2^o, \ldots, y_m^o) \end{bmatrix} \tag{33}$$

The solution of Equation (33) for $(y_i - y_i^o)$ where $i = 1, 2, \ldots, m$ gives values which may be added to the corresponding approximate solution values, y_i^o. Ideally, these additions would yield the solution values y_i; however, in most cases the exact solution values will not be obtained because of the truncation of the Taylor series. Instead, a value of y_i will be obtained which is not the exact solution, but closer to the exact solution than y_i^o if the Newton-Raphson method is convergent (33). This closer approximate solution may then be used to repeat the Newton-Raphson method to obtain a still closer solution. In this manner, the method can be repeated until a sufficiently accurate solution is obtained. It is convenient to rewrite Equation (33) for the k^{th}

iteration as

$$\begin{bmatrix} \left.\frac{\partial F_1}{\partial y_1}\right|_{y^k} & \left.\frac{\partial F_1}{\partial y_2}\right|_{y^k} & \cdots & \left.\frac{\partial F_1}{\partial y_m}\right|_{y^k} \\ \left.\frac{\partial F_2}{\partial y_1}\right|_{y^k} & \left.\frac{\partial F_2}{\partial y_m}\right|_{y^k} & \cdots & \left.\frac{\partial F_2}{\partial y_m}\right|_{y^k} \\ \vdots & \vdots & & \vdots \\ \left.\frac{\partial F_m}{\partial y_1}\right|_{y^k} & \left.\frac{\partial F_m}{\partial y_2}\right|_{y^k} & \cdots & \left.\frac{\partial F_m}{\partial y_m}\right|_{y^k} \end{bmatrix} \begin{bmatrix} (y_1^{k+1} - y_1^k) \\ (y_2^{k+1} - y_2^k) \\ \vdots \\ (y_m^{k+1} - y_m^k) \end{bmatrix} = \begin{bmatrix} -F_1(y_1^k, y_2^k, \cdots, y_m^k) \\ -F_2(y_1^k, y_2^k, \cdots, y_m^k) \\ \vdots \\ -F_m(y_1^k, y_2^k, \cdots, y_m^k) \end{bmatrix} \tag{34}$$

Figure 17 shows a flow chart for a method of using Equation (34) to solve Equation (30).

It should be noted that Equation (30) could also have been written as

$$0 = F_i\,(x_1^*, x_2^*, \ldots, x_n^*, y_1, y_2, \ldots, y_m, t^*),$$

$$i = 1, 2, \ldots, m, \tag{35}$$

where the asterisks indicate that specific values of the variables $x_1, x_2, \ldots, x_n$, t have been substituted into the equations.

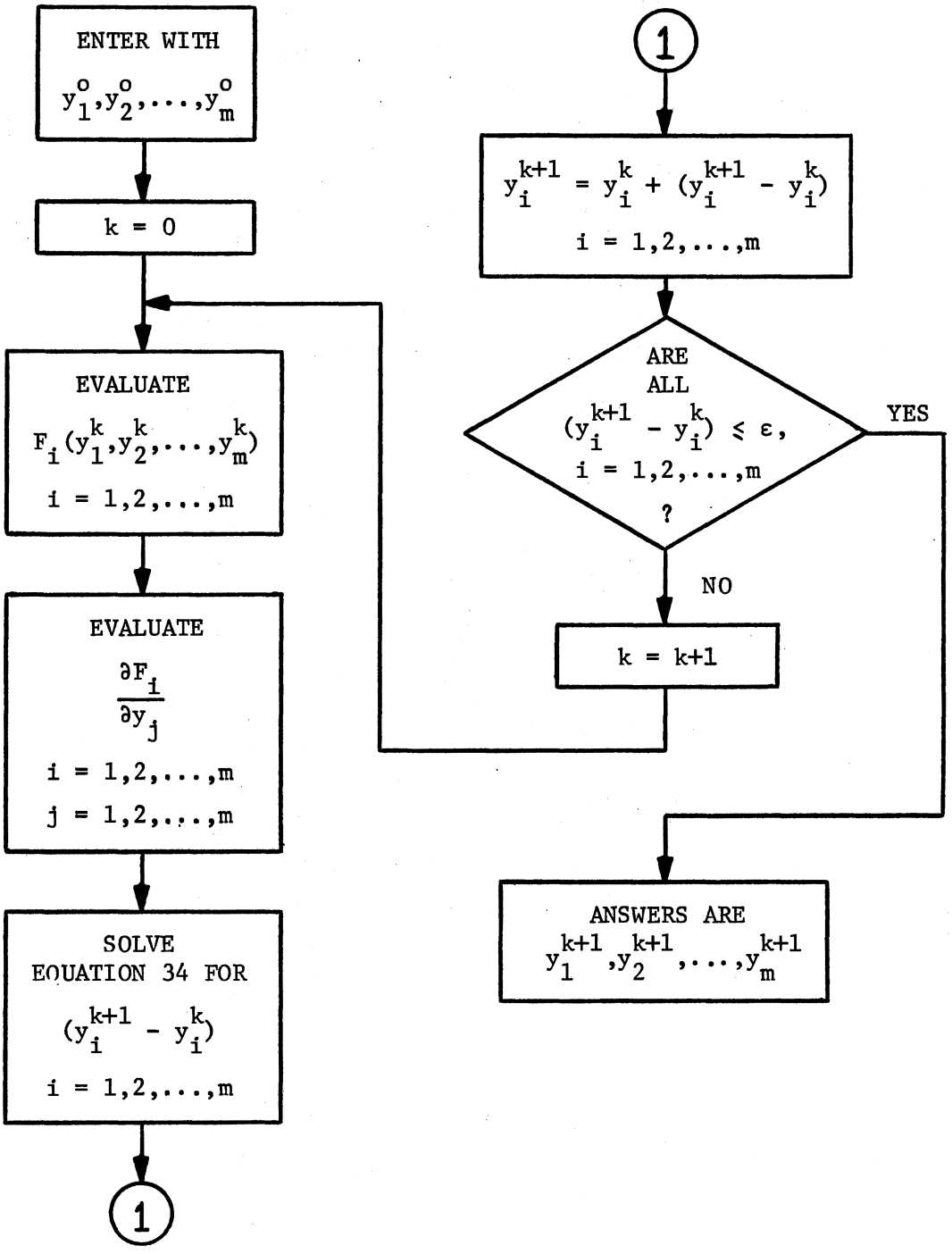


Figure 17. Algorithm to Solve Equation (30) by the Newton-Raphson Method

APPENDIX B

A MODIFIED NEWTON-RAPHSON METHOD

Appendix A derives the equations to be used in the Newton-Raphson method for the solution of algebraic equations of the form

$$0 = F_i' (y_1, y_2, \ldots, y_m), \; i = 1, 2, \ldots, m. \tag{36}$$

This appendix modifies the equations derived in Appendix A for the solution of n algebraic equations of the form

$$0 = F_i (y_1, y_2, \ldots, y_n, u_1, u_2, \ldots, u_q),$$

$$i = 1, 2, \ldots, n, \tag{37}$$

where

$$u_j = u_j (y_1, y_2, \ldots, y_n), \; j = 1, 2, \ldots, q. \tag{38}$$

Because the u-variables have been introduced into the algebraic equations and because the u-variables are defined in terms of the algebraic variables as shown in Equation (38), partial derivatives used in Equation (34), $\partial F_i / \partial y_r$, must be replaced by the terms

$$\frac{\partial F_i}{\partial y_r} + \sum_{j=1}^{q} \frac{\partial F_i}{\partial u_j} \frac{\partial u_j}{\partial y_r} . \tag{39}$$

Therefore, Equation (34) may be rewritten in the following form (Equation (40)). Figure 18 shows a flow chart for a method of using Equation (40) to solve Equations (37) and (38).

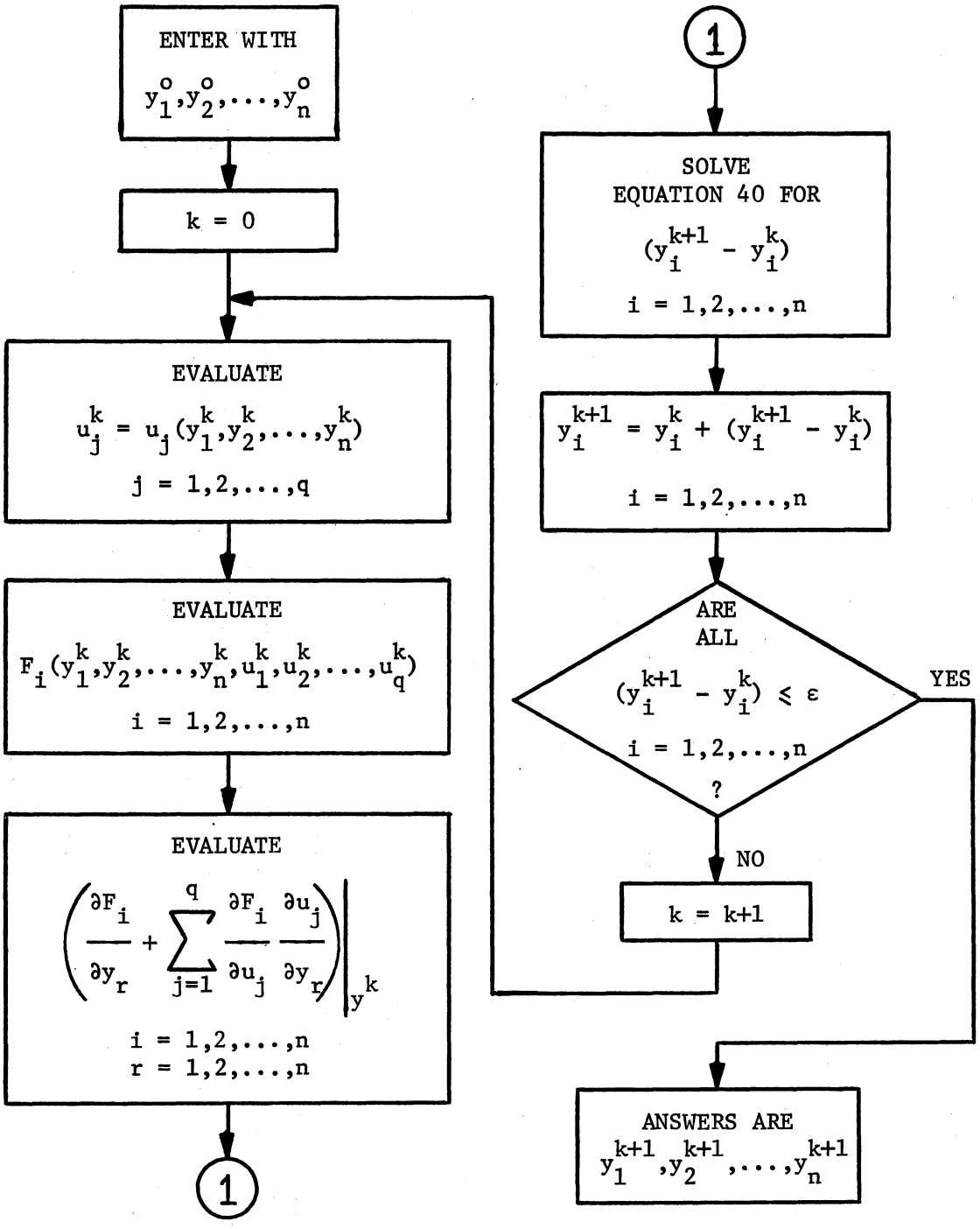


Figure 18. Algorithm to Solve Equation (37) by the Newton-Raphson Method

$$\begin{bmatrix} \left(\frac{\partial F_1}{\partial y_1} + \sum_{j=1}^{q} \frac{\partial F_1}{\partial u_j}\frac{\partial u_j}{\partial y_1}\right)\Big|_{y^k} & \left(\frac{\partial F_1}{\partial y_2} + \sum_{j=1}^{q} \frac{\partial F_1}{\partial u_j}\frac{\partial u_j}{\partial y_2}\right)\Big|_{y^k} & \cdots & \left(\frac{\partial F_1}{\partial y_n} + \sum_{j=1}^{q} \frac{\partial F_1}{\partial u_j}\frac{\partial u_j}{\partial y_n}\right)\Big|_{y^k} \\ \left(\frac{\partial F_2}{\partial y_1} + \sum_{j=1}^{q} \frac{\partial F_2}{\partial u_j}\frac{\partial u_j}{\partial y_1}\right)\Big|_{y^k} & \left(\frac{\partial F_2}{\partial y_2} + \sum_{j=1}^{q} \frac{\partial F_2}{\partial u_j}\frac{\partial u_j}{\partial y_2}\right)\Big|_{y^k} & \cdots & \left(\frac{\partial F_2}{\partial y_n} + \sum_{j=1}^{q} \frac{\partial F_2}{\partial u_j}\frac{\partial u_j}{\partial y_n}\right)\Big|_{y^k} \\ \vdots & \vdots & & \vdots \\ \left(\frac{\partial F_n}{\partial y_1} + \sum_{j=1}^{q} \frac{\partial F_n}{\partial u_j}\frac{\partial u_j}{\partial y_1}\right)\Big|_{y^k} & \left(\frac{\partial F_n}{\partial y_2} + \sum_{j=1}^{q} \frac{\partial F_n}{\partial u_j}\frac{\partial u_j}{\partial y_2}\right)\Big|_{y^k} & \cdots & \left(\frac{\partial F_n}{\partial y_n} + \sum_{j=1}^{q} \frac{\partial F_n}{\partial u_j}\frac{\partial u_j}{\partial y_n}\right)\Big|_{y^k} \end{bmatrix} \begin{bmatrix} (y_1^{k+1} - y_1^k) \\ (y_2^{k+1} - y_2^k) \\ \vdots \\ (y_n^{k+1} - y_n^k) \end{bmatrix} =$$

$$\begin{bmatrix} -F_1\,(y_1^k, y_2^k, \ldots, y_n^k, u_1^k, u_2^k, \ldots, u_q^k) \\ -F_2\,(y_1^k, y_2^k, \ldots, y_n^k, u_1^k, u_2^k, \ldots, u_q^k) \\ \vdots \\ -F_n\,(y_1^k, y_2^k, \ldots, y_n^k, u_1^k, u_2^k, \ldots, u_q^k) \end{bmatrix} \tag{40}$$

APPENDIX C

A HYDSIM EXAMPLE

The response of the fuel injection system shown in Figures 13, 14 and 15 was simulated using the HYDSIM computer program. The simulation started from a condition of steady regulator flow with no injector actuation and was carried out for eight milliseconds of real time.

Figure 19 shows a copy of the HYDSIM input data cards and Table I describes the components used in the simulation. Additional information on HYDSIM components and usage can be found in "HYDSIM User's Manual" (1).

All of the components used in this simulation are contained in the HYDSIM component library except the displacement forcing function used to actuate the injectors. This component was programmed for this simulation. It determines displacement as a function of time by performing linear interpolation on time-displacement points given in the input data. Figures 20 and 21 show the resulting injector actuations. The first three data cards shown in Figure 19 initialize internal HYDSIM tables to include this displacement component. The component model was programmed in a FORTRAN IV subroutine and submitted with the data cards at execution.

Figure 16 shows the pressure at the inlets of the injectors. This pressure is identical at the inlets of all the injectors since both

```
&ADD COMPONENT
  SPEC1,3,10,1,0,0,0,DISP
&END
&FREE
&RKIV
&* ********************** RUNGE-KUTTA TEST **********************
  TF=0.008,  DT=0.00001,  PM=10,  NEWTON=200
  1(1)=8(1)P,  8(2)P=2(3)G,  2(2)=5(2),  2(1)=3(2)
  3(1)=5(3)P,  3(3)=6(3)P,  5(1)=6(2)P
  6(1)=7(2)P,  7(1)P=9(1),  9(2)=10(1)PG,  10(2)P=11(1)
  11(2)=12(1),  9(3)G=29(2),  9(4)=30(2)
  PRESSR/1,12/,  TEE/2,6/,  CYLIDX/3/,  VALV2B/5/
  VOLUME/7/,  LINE2/10/,  PUMP/11/,  LINE1/8/,  CROSS/9/
  1/10*0/
  3/4,4,-0.00245,200,0.1/
  5/75E3,6.4E-5,0.6,0.25,0,0,1/
  7/50,75E3,1.5/
  8/0,180,0.500,6.4E-5,75E3,2.9E-8,0,10,-.46/
  10/0,168,0.375,6.4E-5,75E3,2.9E-8,50,55,-.46/
  11/2.9E-8,0.02,0,0,0,0.61E-8,157/
  12/10*0/
  14(2)=16(1),  16(2)P=23(1),  16(3)P=24(1),  30(1)=14(1)
  23(2)=27(1),  23(3)=19(1),  24(2)=28(1),  24(3)=20(1)
  LINE2/14/,  TEE/16/,  DISP/19,20/,  VALV2B/23,24/
  PRESSR/27,28/,  LOSS21/30/
  14/0,12,0.25,6.4E-5,75E3,2.9E-8,50,50,0/
  19/0,.0005,-.01,.002,-.01,.0025,0,.008,0,0/
  20/0,.004,0,.0045,-.01,.006,-.01,.0065,0,.008/
  23/75E3,6.4E-5,0.6,0.0182,0,0,2/
  24/75E3,6.4E-5,0.6,0.0182,0,0,2/
  27/10*0/
  28/10*0/
  30/1,.5,.25,6.4E-5,-10,50/
  13(2)=15(1),  15(2)PG=21(1),  15(3)PG=22(1),  29(1)=13(1)
  21(2)=25(1),  21(3)P=17(1),  22(2)=26(1),  22(3)P=18(1)
  LINE2/13/,  TEE/15/,  DISP/17,18/,  VALV2B/21,22/
  PRESSR/25,26/,  LOSS21/29/
  13/0,12,0.25,6.4E-5,75E3,2.9E-8,50,50,0/
  17/0,.0005,-.01,.002,-.01,.0025,0,.008,0,0/
  18/0,.004,0,.0045,-.01,.006,-.01,.0065,0,.008/
  21/75E3,6.4E-5,0.6,0.0182,0,0,2/
  22/75E3,6.4E-5,0.6,0.0182,0,0,2/
  25/10*0/
  26/10*0/
  29/1,.5,.25,6.4E-5,-10,50/
&END
```

Figure 19. Input Data for HYDSIM Simulation of Fuel-Injection System

TABLE I

HYDSIM COMPONENTS USED IN SIMULATION
OF FUEL-INJECTION SYSTEM

COMPONENT NUMBER	LIBRARY NAME	COMPONENT LOCATION OR DESCRIPTION
1	PRESSR	Exhaust Pressure For Regulator Return Line
2	TEE	In Low-Pressure Side of Regulator
3	CYLIDX	Cylinder In Regulator
5	VALV2B	Two-Way Valve In Regulator
6	TEE	In High-Pressure Side of Regulator
7	VOLUME	In Regulator
8	LINE1	Line At Low-Pressure Side of Regulator
9	CROSS	Four-Port Connector
10	LINE2	Line At Pump Outlet
11	PUMP	Constant Speed Pump
12	PRESSR	Pressure Source At Pump Inlet
13	LINE2	Fuel Line Leading To Injectors 1 And 2
14	LINE2	Fuel Line Leading To Injectors 3 And 4
15	TEE	At Inlet To Injectors 1 And 2
16	TEE	At Inlet To Injectors 3 And 4
17	DISP	Displacement Function For Injector 1
18	DISP	Displacement Function For Injector 2
19	DISP	Displacement Function For Injector 3
20	DISP	Displacement Function For Injector 4
21	VALV2B	Two-Way Valve In Injector 1
22	VALV2B	Two-Way Valve In Injector 2
23	VALV2B	Two-Way Valve In Injector 3
24	VALV2B	Two-Way Valve In Injector 4
25	PRESSR	Pressure At Injector 1 Outlet
26	PRESSR	Pressure At Injector 2 Outlet
27	PRESSR	Pressure At Injector 3 Outlet
28	PRESSR	Pressure At Injector 4 Outlet
29	LOSS21	Entrance/Exit Loss
30	LOSS21	Entrance/Exit Loss

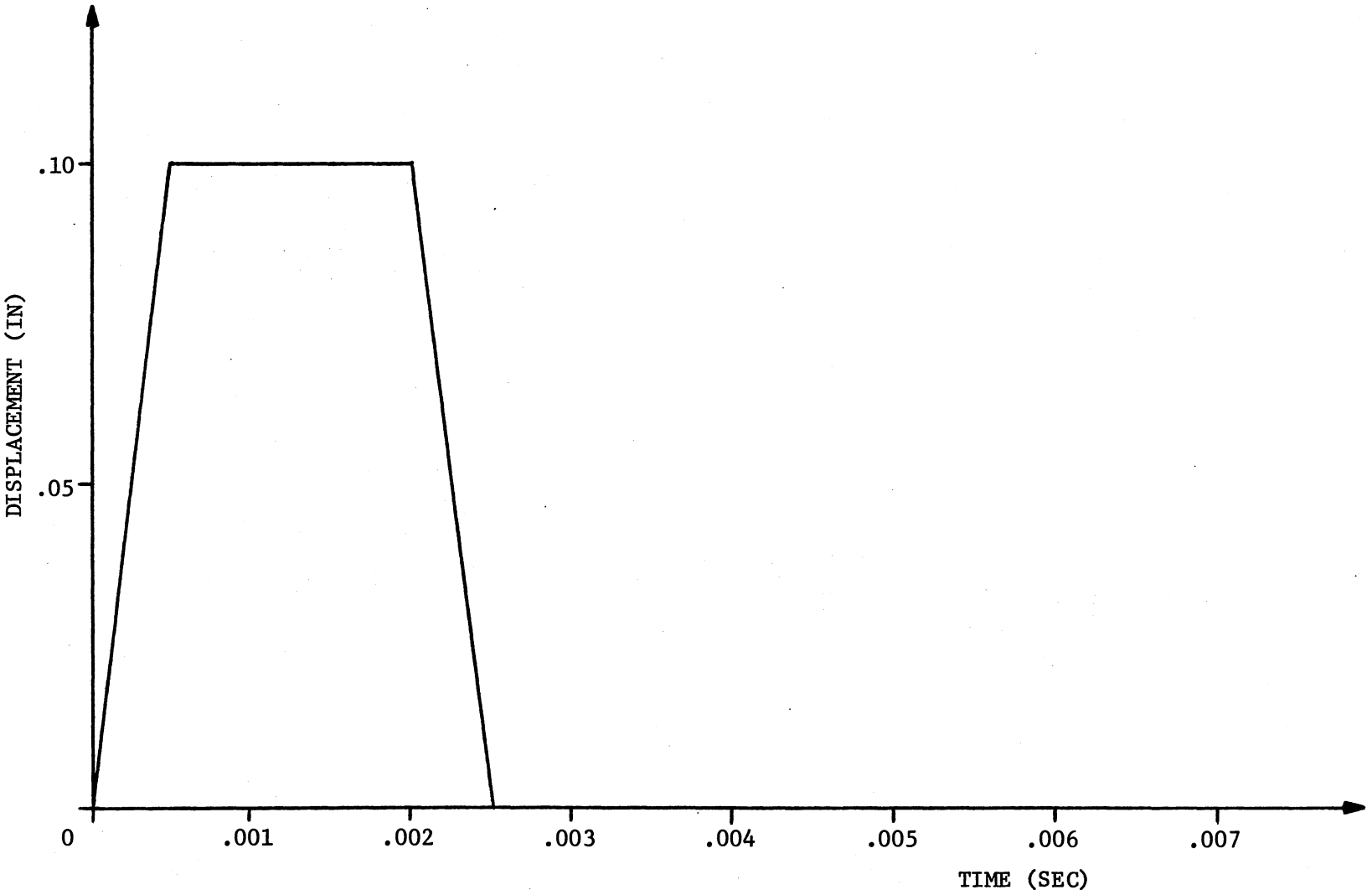


Figure 20. Valve Opening for Injectors 1 and 3

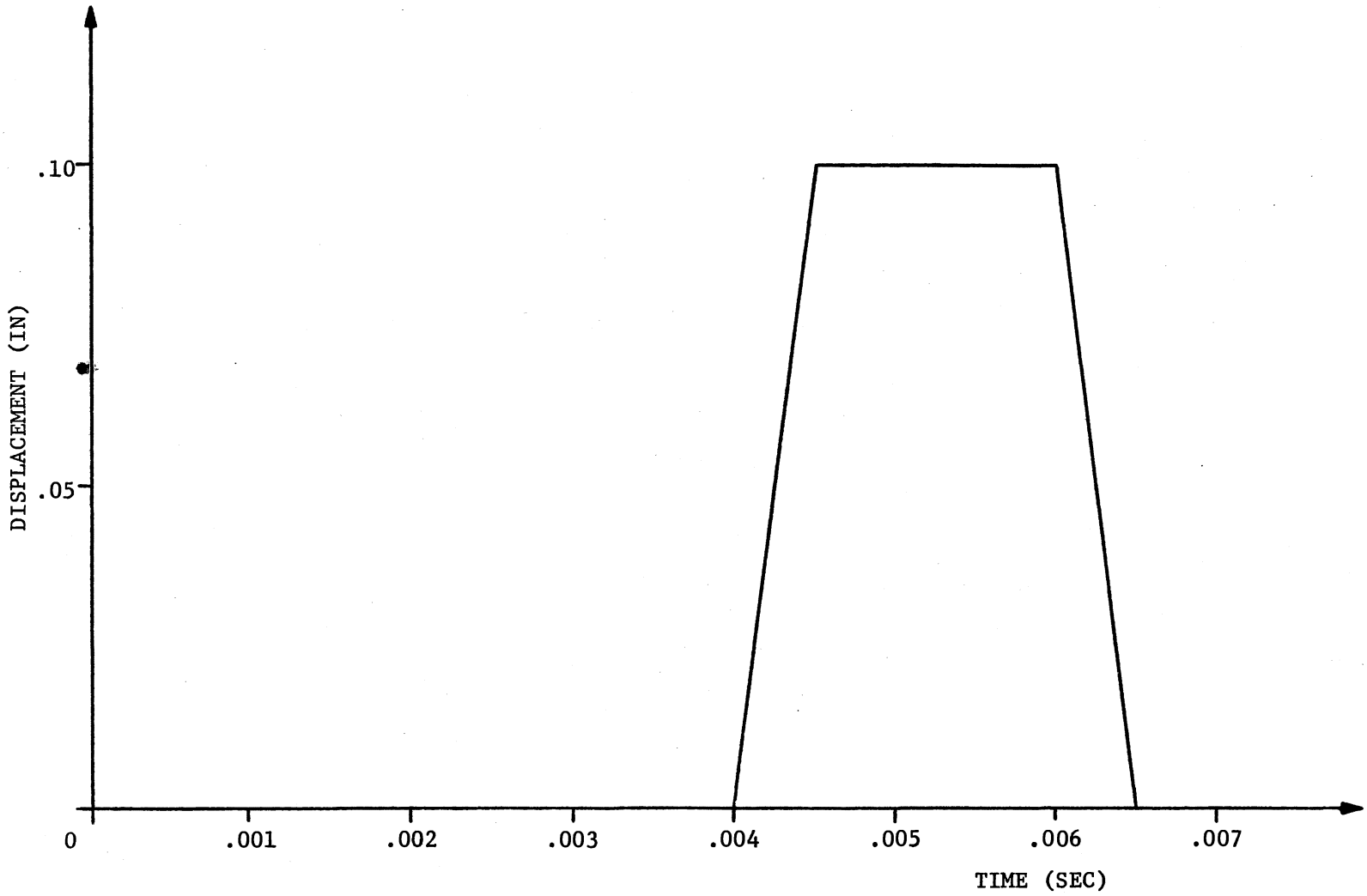


Figure 21. Valve Opening for Injectors 2 and 4

pairs are connected into the system through identical lines and loss components. Figures 22 and 23 show the fuel flow rates through the injectors.

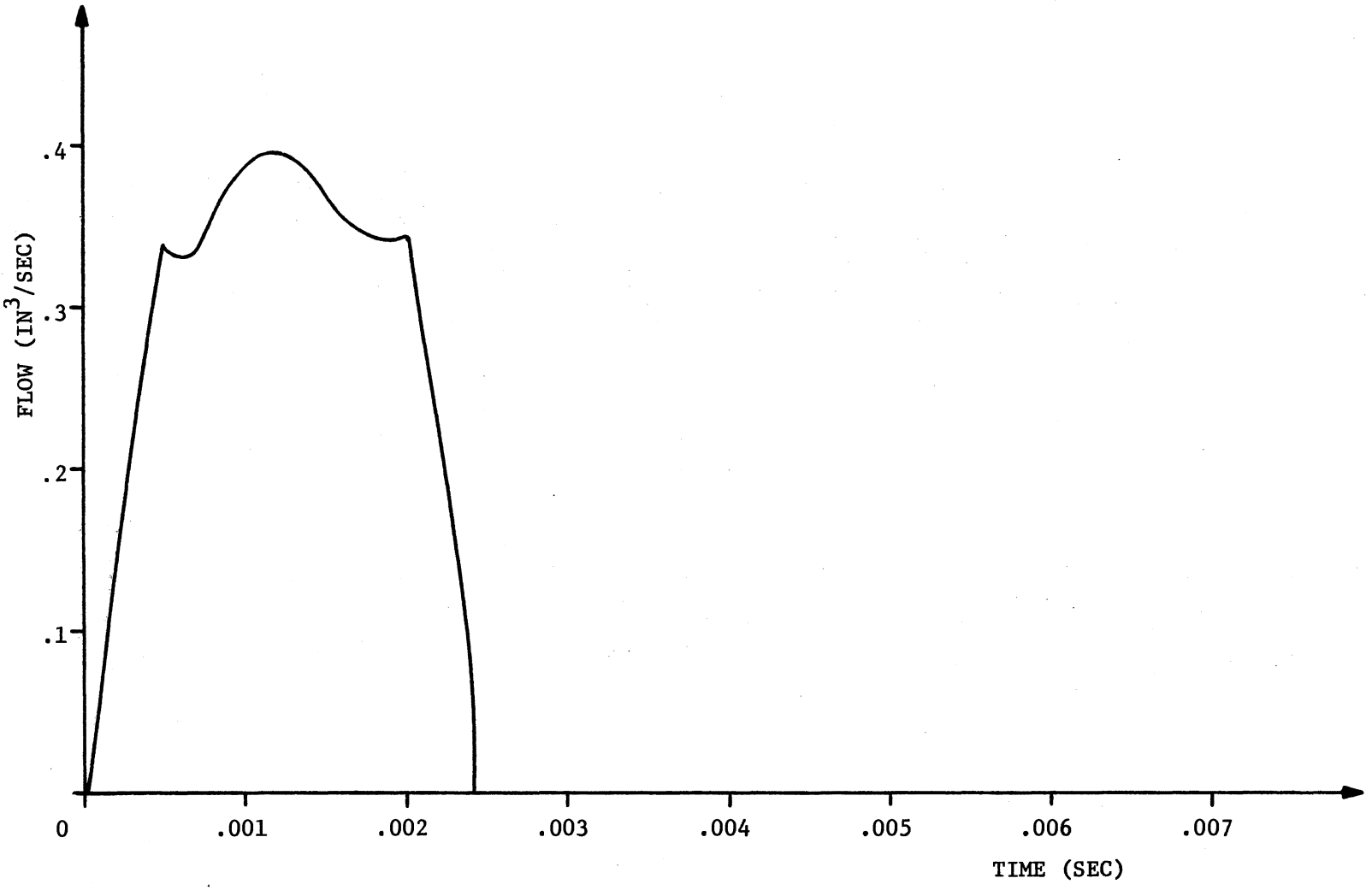


Figure 22. Flow Through Injector 1 or 3

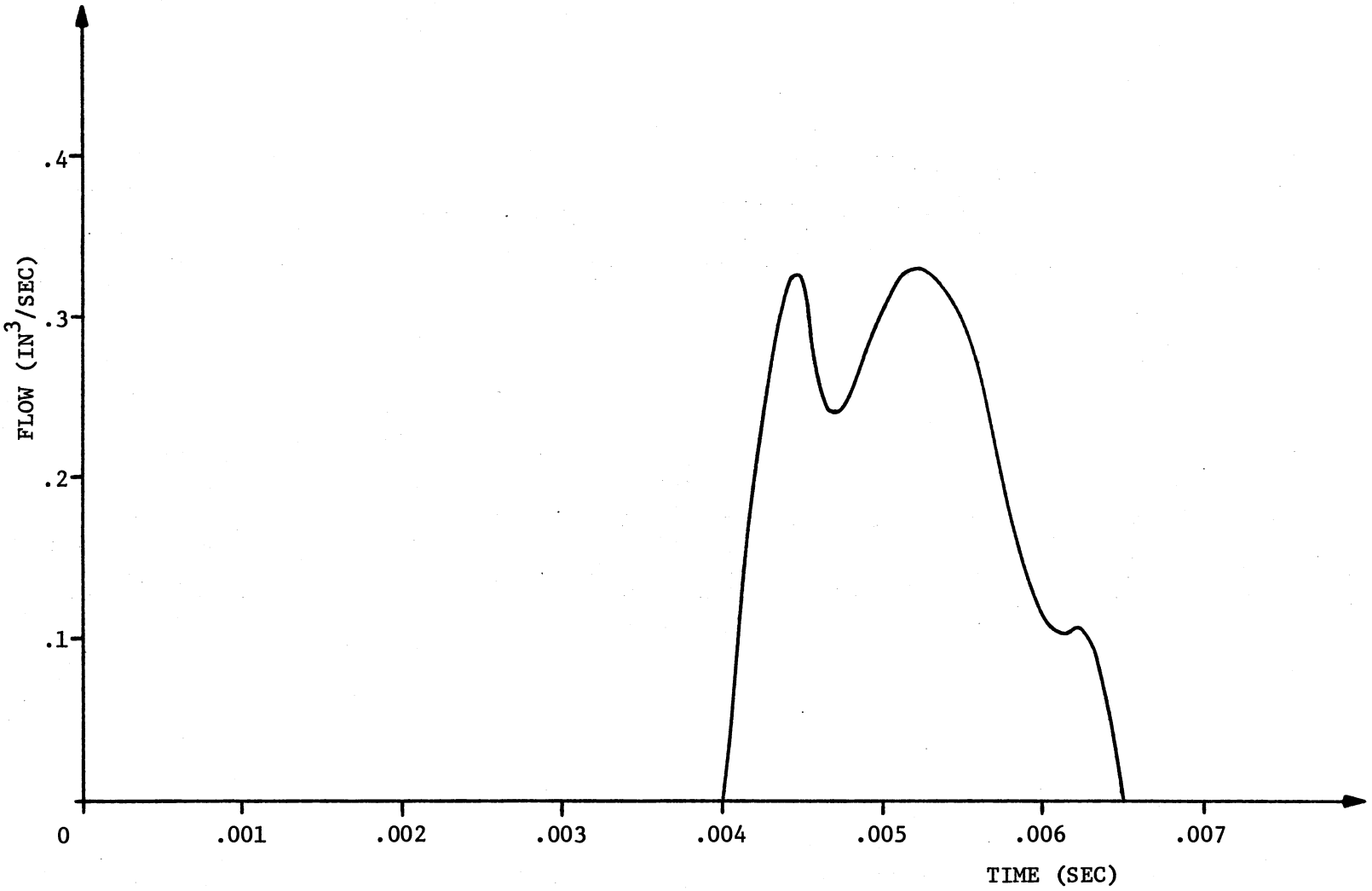


Figure 23. Flow Through Injector 2 or 4

VITA

Christopher K. Smith

Candidate for the Degree of

Doctor of Philosophy

Thesis: DIGITAL COMPUTER SIMULATION OF COMPLEX HYDRAULIC SYSTEMS USING MULTIPORT COMPONENT MODELS

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born at Scofield Barracks, Oahu, Hawaii, on June 30, 1938.

Education: Graduated from Hillsboro High School, Nashville, Tennessee, in June, 1956; received Bachelor of Science degree in Mechanical Engineering from Tennessee Technological University in 1967; received the Master of Science degree from Tennessee Technological University in 1969; completed requirements for the Doctor of Philosophy degree at Oklahoma State University in July, 1975.

Professional Experience: Graduate teaching assistant, Tennessee Technological University, 1967-69; Summer school instructor, Tennessee Technological University, 1968; graduate teaching assistant, Oklahoma State University, 1969; graduate research assistant, Oklahoma State University, 1970-72; member of Engineering Staff, Bendix Research Laboratories, Bendix Corporation, 1972.

Professional Organizations: Member Pi Tau Sigma; member Tau Beta Pi.