POWER SYSTEM DYNAMIC ANALYSIS
PHASE I

EPRI EL-484
(Research Project 670-1)

Final Report

July 1977

Prepared by

Energy Technology Applications Division
BOEING COMPUTER SERVICES, INC.
Seattle, Washington 98124

PRINCIPAL INVESTIGATORS
Benjamin Dembart
Albert M. Erisman
Esko G. Cate
Michael A. Epton
Herman Dommel

Prepared for

Electric Power Research Institute
3412 Hillview Avenue
Palo Alto, California 94304

EPRI Project Managers
Timothy S. Yau
Paul M. Anderson

FOREWORD

The research project entitled "Power System Dynamic Analysis" (RP670-1), reported in this volume, is one of an important group of current research projects in the EPRI System Planning, Security and Control Program. Other related projects are as follows:

Dynamic Simulation Projects:

RP744 Low Frequency Oscillation Analysis
RP745 Mid-Term Dynamic Analysis
RP763 Coherency-Based Dynamic Equivalents
RP764 Long-Term Dynamic Analysis

Dynamic Modeling Projects:

RP849 Load Dynamic Modeling
RP997 Synchronous Machine Modeling

The project discussed in this report concentrates on the analysis and selection of appropriate numerical methods for computer solution of the system dynamic performance.

These projects have been initiated in response to a growing industry need for advanced, accurate, efficient analytical tools for system studies. For some utilities the computer time and resources devoted to stability calculation is a significant fraction of their capability. It is the EPRI goal that these new programs, together with existing stability codes, might form the building blocks on which improved dynamic simulation programs can be constructed.

P.M. Anderson, Program Manager
T.S. Yau, Project Manager
System Planning, Security and
Control Program

June 1977

ABSTRACT

The models required for power system dynamic analysis are continually growing larger and more complex as power system interconnections are growing more significant. However, dynamic studies already require a significant portion of the computer capacity of electric power utilities. Thus the industry is faced with conflicting requirements:

- Reduce the cost of dynamic simulation without sacrificing reliability,
- Allow for more realistic simulation through the use of more complex models.

The purpose of this Boeing Computer Services RP 670 research is to improve both the efficiency and reliability of today's simulation, and to lead to a capability to perform the more complex simulation of tomorrow.

The approach taken was to identify the fundamental characteristics of dynamic power system models, and to relate these to candidate numerical methods. The performance characteristics of the methods were analyzed for efficiency, reliability, and stability. Finally, extensive testing was performed on selected test cases (smaller cases were tested first, with testing on larger cases now proceeding) to identify the best computational procedures for solving power system dynamic problems.

A diagnostic transient stability program has been developed to perform this testing. This program not only performs transient stability computation, but also analyzes the performance of the numerical methods. The diagnostic program, developed entirely on this research project, is presently being documented for use by other researchers.

Conclusions of this first phase include specific recommendations of integration methods, step size requirements, and algebraic solution methods which will provide the basis for a new industry standard transient stability program. Recommendations are also included that will be useful for improving the efficiency and reliability of existing stability codes with minimal changes required.

ACKNOWLEDGMENTS

Boeing Computer Services has completed EPRI research project RP 670-1, "Power System Dynamic Analysis". The RP 670-1 work was managed by Albert M. Erisman of the Energy Technology Applications Division (ETA). The work was performed by Benjamin Dembart, Esko G. Cate, Michael A. Epton and Rose M. Southall, all of ETA. Professor Herman Dommel of the University of British Columbia served as a consultant.

The EPRI project managers were Timothy S. Yau and Paul M. Anderson of the System Planning, Security and Control program for EPRI.

The authors acknowledge the valuable contributions of a Technical Advisory Group of consultants to EPRI who reviewed the project at quarterly meetings: Kenneth Bess, Western Systems Coordinating Council Technical Staff; W. S. Ku, Public Service Electric & Gas Company; Gerhard Steinbrenner, Arizona Public Service Company; William F. Tinney, Bonneville Power Administration; Clifford C. Young and James F. Luini, Pacific Gas and Electric Company.

The authors would also like to acknowledge the assistance provided by Gerhard Steinbrenner and Dennis Brown of the Arizona Public Service Company in making the APS transient stability program operational at our computer facilities. We would like to acknowledge the valuable suggestions made by Demos Gelopulous of Arizona State University, and William Mittelstadt of the Bonneville Power Administration.

CONTENTS

CONTENTS (contd)

FIGURES

TABLES

Section 1

INTRODUCTION AND SUMMARY

The accepted method today of studying the stability of large scale interconnections of generation, transmission and load is by means of power system dynamic analysis. That is, the response of the power system to large disturbances is simulated by computer analysis. It is the purpose of this project to study the methods that can be used by the computer programs that perform this analysis, and to pursue a path that will lead to uncovering the best of these methods, and ultimately to a new generation of computer codes that is superior in many ways to today's codes.

One major step along that path has been taken, and a second is well underway. A diagnostic program has been prepared to assist in the study of the methods. The diagnostic program serves as a test bed where any proposed method can be implemented, and thoroughly analyzed as it performs on a wide variety of common test cases. Diagnostic data such as timing, operational counts, convergence rates, and truncation errors can be computed by the diagnostic program in addition to the usual stability program output. The diagnostic program and test case set will be made available to other researchers so that new methods can be tested and compared to those already studied. The diagnostic program will also be useful for studying the accuracy and efficiency of proposed new models.

Study of the methods that appear most promising for the next generation of stability programs is well underway. The research undertaken thus far on this project has confirmed the conjectures of many other researchers as to the efficiency of various computational methods. At this time we are prepared to make recommendations as to the most suitable methods for further consideration. The recommended methods represent the best compromise between operational efficiency and life cycle cost.

1.1 BACKGROUND

The application of digital computers to power systems dynamic analysis is widespread, and the trend has been towards an evergrowing complexity and magnitude of power system simulations. This trend has progressed so that a significant percentage

of the total computing resources of some power companies is devoted to dynamic simulations. The high cost of the resources devoted to these analyses is justified by the impact they have on system security and on costly power system installations. However, because of the high cost of these analyses and the limited resources of the computing facilities of many power companies, otherwise beneficial dynamic analysis is not always performed. The large quantities of computer resources concentrated in the area of power system dynamic analysis make it a profitable area for research and development aimed at producing dynamic analysis codes that will reduce the cost of these analyses or allow the analysis of a greater number of more complex and realistic models.

1.2 OBJECTIVES

Near Term Objectives

The major short term objective of this research project is to establish an approach for the impartial and scientific testing of algorithms. It is hoped that the methods and tools developed in this research, and used to analyze those algorithms that seem to be the most significant today, will be adopted by other researchers to analyze new algorithms as they become of interest. To accomplish this goal, the analysis methods will be documented, and the analysis tools will be made available to interested researchers.

A second major short term objective is to identify the most promising algorithms for use in the next generation of dynamic stability code and to make that information available through reports such as this one.

Long Term Objectives

The ultimate objective of this research effort is to construct a highly efficient production grade computer code that will make the very latest technology available to any interested power company without having to duplicate the very large expense involved in producing such a code.

Furthermore, the methodology selected for use in the production code will be documented in detail early in the project, and will form a guideline that other code developers can adopt or seek to improve upon.

1.3 APPROACH

A phased approach has been selected for this research effort. In phase 1 a diagnostic program and test case base for studying algorithms has been constructed.

These diagnostic tools have been used to evaluate those solution methods judged to be the most promising for inclusion in the next generation of codes. The strongest of the methods studied are identified in this report.

In an extension to phase 1 the diagnostic program will be fully documented and made available to interested researchers in the power industries. Basic research into algorithms will continue through this extension.

In phase 2 the methods selected in phase 1 will be studied to discover efficient ways to implement them. This includes fine tuning the algorithms, identifying the best approach to handling the sparse matrices that are involved in the calculations, computing estimates of the local and global error produced by the algorithms, and computing the sensitivity of response variables to problem parameters.

Phase 3 will be devoted to building the production grade program. This will involve a team effort, with specialists in power system dynamics, numerical algorithms, and software construction included on the team.

The extension to phase 1 will be completed by the end of 1977. Phase 2 will be completed in 1978, and phase 3 can be completed by 1979 or 1980.

1.4 SUMMARY OF PHASE 1 RESEARCH

Approach

The approach taken in phase 1 to evaluate the algorithms that can be used for power system dynamic analysis was based on testing and analysis. In order to perform the testing, a diagnostic program was written. This program can perform dynamic simulations using any one of a large number of algorithms. New algorithms can be added easily. The diagnostic program produces, in addition to the usual stability program output, performance data that can be used to evaluate the algorithm being tested. This performance data includes timing, operational counts, convergence rates, and truncation errors. A set of test cases was collected and used for testing the algorithms.

The number of algorithms that have been proposed for use in power system dynamic analysis is very large. It was not possible or appropriate to test all of these algorithms. Preliminary analysis was performed to identify those algorithms that merited further testing. The order, accuracy, stability, and complexity of the algorithms were evaluated in this preliminary analysis.

In the preliminary analysis, and subsequent testing of the algorithms, every effort was made to evaluate the life cycle cost of each algorithm considered. This was done by evaluating the generality, applicability, and complexity of each algorithm, as well as its operational costs. This is the first study of this nature to consider these parameters, and they had considerable influence on the recommendations that are made in this report.

Introduction to Algorithms

Initially two major decisions must be made in order to design a computer code for power system dynamic analysis. An interface method must be selected, and an integration algorithm must be chosen. In this research project two interface approaches were considered: simultaneous solution and partitioned solution. The former involves solving all the equations from all components of the power system simultaneously, while the latter consists of solving each generator separately, and interfacing the results through the network. In this research project several integration algorithms were studied. These algorithms can be classified as either implicit or explicit. The implicit methods get their name because an implicit system of equations must be solved at every time step. For explicit methods a time step can be taken without solving such a system. The options are summarized in Table 1-1.

Table 1-1

BASIC APPROACHES TO POWER SYSTEM DYNAMIC ANALYSIS

INTERFACE / INTEGRATION	SIMULTANEOUS	PARTITIONED
IMPLICIT	SI	PI
EXPLICIT	SE	PE

All four of the options listed in Table 1-1 are potential structures for a dynamic stability program. The algorithms being used today in production codes are either of the SI or PE type, and there are good reasons for this which are discussed in the body of this report. This research effort concentrated on the SI and PE approaches.

The computational problems associated with dynamic simulations can be partitioned into several sub-problems. This can be illustrated by outlining the computational processes. When an SI approach is adopted, the process is as shown in Table 1-2. When a PE approach is adopted, the process is as shown in Table 1-3.

Table 1-2

COMPUTATIONAL OUTLINE FOR AN SI APPROACH

I. Initialize variables
II. Compute dynamic response
    A. Take integration step-solve algebraic system of equations
        1. Evaluate model equations
        2. Evaluate difference equations
        3. Solve linear equations
    B. Update variables and time

Table 1-3

COMPUTATIONAL OUTLINE FOR A PE APPROACH

I. Initialize variables
II. Compute dynamic response
    A. Take one step-Jacobi iteration
        1. Extrapolate bus voltages
        2. Integrate all machine equations
        3. Solve network equations - solve algebraic system
    B. Update variables and time

The structure of the stability program that realized these two computational schemes is discussed in Section 2. Other study tools are also covered. In

Section 3 the difference equations used in the SI approach are analyzed. In Section 4 the other three approaches are discussed. Finally, in Section 5 the solution of an algebraic system is covered. In appendices to this report the special problem characteristics of stiffness and linearity are discussed, as are the implications of these characteristics on algorithms and modeling.

Summary of Report

This report has four major sections. In Section 2 the tools used throughout this study are discussed. The key tool was a diagnostic program that was designed, written, and made operational entirely as a result of this research project. The diagnostic program is a transient stability program that functions as a test bed where algorithms can be evaluated in a transient stability environment. The diagnostic program was designed with two major requirements in mind. First, it was necessary that a wide range of solution algorithms be compatible with the program structure. Second, a wide variety of diagnostic data such as timing and operational counts had to be available. It was found that a highly modularized design was required to meet these design objectives.

Other tools used included a production grade transient stability program that was contributed by Arizona Public Service, and test cases.

In Section 3 algorithms for solution of the differential equations, using an SI approach, are discussed. Five difference equations were considered. From these five, three were selected for testing on the diagnostic program. These three were the trapezoidal rule, the APS rule, and the implicit midpoint rule. The testing served to eliminate the last of these rules, but was inconclusive as to the relative merits of the first two.

The trapezoidal rule is recommended because of its simplicity, generality, and stability.

Two methods of estimating the local truncation error of the integrators were considered. The first is based on polynomial extrapolation of past results, the second is based on integrating over an interval with two different step sizes. Each of these approaches has its strengths and weaknesses and it is recommended that both be tested on the diagnostic program. On some test cases, it appears that step size adjustment can be cost effective.

In Section 4 integration algorithms for solution of the differential equations, using PE, SE and PI approaches, are discussed. Two explicit integration algorithms, Runge-Kutta and predictor-corrector, were implemented in the diagnostic program using the PE approach. These methods were tested extensively. Several methods for interfacing the differential and algebraic equations were also tested.

Theoretical investigations and tests revealed that PE integration produces a fundamental inaccuracy associated with the interfaces to the network equations. A consequence of this result is that the accuracy of the integrator must be compatible with the interface accuracy. Low order Runge-Kutta balances well within the interface process and is more efficient than the fourth order classical Runge-Kutta rule. Another consequence is that a trade-off can be made between algebraic step size and iteration count that may well favor taking small steps very rapidly.

Several detailed recommendations on efficient implementation of explicit algorithms have been made.

Testing and theory indicate that the SI approach, using the trapezoidal rule, is significantly more efficient than any PE integration approach.

In Section 5 algorithms for solution of the algebraic equations are discussed. Four iterative algorithms were considered for solving the algebraic equations. Several methods of predicting an estimate of the solution to use as a starting value for the iterations were also considered.

Extensive analysis of timing data was collected using the diagnostic program. This data was used to identify those components of the computation that were the most costly. Algorithms were selected to minimize the use of these critical components.

The four algorithms considered were Newton's method, quasi-Newton's method, very dishonest Newton's method (VDHN), and a fixed point iteration with relaxation (also referred to as triangular factorization). The Newton and VDHN methods were implemented in the diagnostic program based on favorable theoretical considerations. Testing indicated that VDHN is superior to Newton's method and, in fact, gains of up to a factor of three can be achieved.

Testing of the prediction approaches indicated that on the average prediction can save more than one iteration per time step while improving accuracy. Testing also

indicated that when prediction is used in conjunction with VDHN, great efficiencies can be achieved with improved accuracy, but careful control of the process is required to assure efficiency.

In Appendix A to this report stiffness and stability are discussed. It is pointed out that due to recent algorithm developments, models may now be developed without being constrained by the problem of stiffness. The stability properties of the algorithms considered are also discussed.

In Appendix B data is discussed that indicates that the nonlinear terms in the problem formulation make a significant contribution to the problem solution. This verifies that the problems are numerically nonlinear.

Appendix C is a summary of the Western Systems Coordinating Council nine bus test series.

It is the conclusion of this study that algorithms have been identified that can greatly improve the reliability and efficiency of power system dynamic simulation codes. However, to achieve the full benefits of these improvements, careful attention will have to be paid to algorithm control and other implementation considerations.

The results of this research project offer benefits to the users of dynamic simulation codes in the following areas:

- Algorithms have been identified that can be used to improve existing codes.
- A design approach has been used that offers a promise of more flexible, error free, easily maintained codes in the future.
- Results on the interaction of models with the numerical algorithms have been discovered that can affect new model development.

1.5 THE NEXT PHASE

In phase 1 a diagnostic program was developed and used to test algorithms. This program has proved to be a valuable tool in the analysis of algorithms. It also has the potential of being a valuable tool for the testing and evaluation of new models of power system equipment. This is because of the generality of the program in accepting new models, the ease with which new models may be added, and the

performance data that is produced. In the next phase the use of the diagnostic program will be documented, and the program will be prepared for distribution to interested researchers.

One kind of performance data produced by the diagnostic program is estimates of the integration error. One of the reasons for computing this information was to evaluate the possibility of using error estimates to control the step size of the integration process. The results of this analysis indicate that on some test cases error control would not be cost effective, whereas on other cases it would be. For these reasons, the study of the relation between step size and error will continue in the next phase. The effects of step size on the overall reliability of the simulations will be determined, and the search for a method of using step size control to reduce costs will proceed.

In the near future, EPRI will make available to this research effort several test cases representing realistic stability tests from different systems throughout the country. These test cases will be about 150 buses. The algorithms under consideration will be tested against these test cases to assure that the testing environment is as realistic and complete as possible.

It is important to recognize that the efficiencies that can be gained from careful implementation of algorithms and good programming practices are as great as those that follow from selecting good algorithms. To this end, the algorithms that have been selected will be studied carefully, and efficient implementations of these algorithms will be developed.

All of the effort that is planned in the next phase are logical next steps aimed at maturing the research initiated in phase 1.

Section 2

STUDY TOOLS

A number of tools were used in the course of the research that was performed. These included the APS power flow and transient stability programs, the diagnostic program, a set of test cases, and the Boole and Babbage program performance evaluator (PPE). The application of each of these tools to the research program will be described in this section.

2.1 APS PROGRAMS

The Arizona Public Service power flow and transient stability programs were used in two distinct ways. The source listings of the program provided a detailed documentation of a particular implementation of a transient stability program. This documentation was useful in providing a general outline for transient stability as well as for providing detailed information on how specific computational difficulties are handled.

The power flow and transient stability codes together with a set of test cases provided a baseline of transient stability solutions that would be used to verify the models and algorithms of the diagnostic program and to provide timing information for a production grade program.

The APS programs were supplied to us by APS in February 1976. A meeting was held at that time to discuss program details. Personnel familiar with the APS program provided a thorough briefing on the transient stability program. This briefing enabled us to use the APS transient stability program listings effectively as a reference source throughout our research.

The APS power flow and transient stability programs were made operational on the BCS computer facility, and used to provide baseline data and timing data that was used to verify the correctness of diagnostic runs.

2.2 DIAGNOSTIC PROGRAM

The principal research tool used in this study has been the diagnostic program. It has been used for examining algorithms in a transient stability environment. The diagnostic information that has been obtained has been used to identify those aspects of the computation that are critical to solution times and to rate algorithms as to their ability to overcome difficulties. The diagnostic program is capable of performing transient stability analysis using any combination of a large number of different algorithms. As well as the usual transient stability output, the program produces diagnostic output that can be used to rate the performance of the algorithms being used.

Diagnostic Program Design

The diagnostic program was developed in its entirety as a result of this research effort. In order for the program to be useful as a diagnostic tool, it was necessary for it to have characteristics that are different from standard transient stability programs. It was necessary to develop an independence between the mathematical equations that described the component models and those that described the solution algorithms. In particular, it was necessary to be able to change the solution equations to reflect a new method without modifying all the component modeling equations. This made it possible to build a library of solution algorithms that can easily be used in any combination with any component models. Addition of new solution algorithms is also simplified by maintaining independence between method equations and model equations.

A second characteristic of the diagnostic program not normally found in transient stability programs is the ability to compute performance data for the solution algorithms being tested. This performance data includes measures of truncation error for integration algorithms, measures of convergence errors for algebraic solution algorithms, counts for important operations, and timing for important operations.

In order to realize the requirements of a transient stability program with independence between methods and models, and with the ability to compute performance data, a highly modular design was found essential. The modular structure for the diagnostic program permitted building component model modules with well defined interfaces, and solution method modules with well defined interfaces. Any module could be replaced by any other module with a compatible interface, thereby

assuring easy modification of methods, and independence between methods and models. The module interfaces provided convenient points for the computation of diagnostic data.

Transient Stability Formulation

The formulation of the transient stability problem within the diagnostic program will now be described in greater detail.

The breakdown of the system into its constituent modules will be described, as will the interface of the modules. This will clarify the methods used to obtain independence.

The initial partitioning of the transient stability problem is into a network, generators, and loads. The network variables are bus voltages V, load currents I_L, and generator currents I_G. The network equations are current equations in rectangular coordinates:

$$YV = I_G - I_L \tag{2-1}$$

where Y is the bus admittance matrix for the network including all shunt terms. The current form of Kirchoff's law in rectangular coordinates was chosen because it appears to be most nearly linear and thus more suited to Newton's method type solution.

Because a current form of the network equation is being used, a current form of the load equation is used. For each load r

$$I_r = (P_r - jQ_r)/(V_{Br})^* \tag{2-2}$$

where

$$\begin{aligned} P_r &= p_r(|V_{Br}|) \\ Q_r &= q_r(|V_{Br}|) \end{aligned} \tag{2-3}$$

p_r and q_r are quadratics reflecting constant impedance, constant current, and constant power terms. Here Br denotes the bus to which the load is connected.

The current vector I_L is computed by adding the contribution of each load I_r to the corresponding term in the load current vector $(I_L)_{Br}$. The interface between the loads and the network are through the bus voltages V_{Br} and the load current vector I_L.

The generator systems are sufficiently complex to require further partitioning. A generator system is partitioned into an interface system, a swing system, a machine, an exciter, a power system stabilizer, and a governor. The interface system contains the variables needed to interface the generator system with the network. These include the d-axis and q-axis voltage and current V_d, V_q, I_d, I_q. The interface system also includes the variables needed to interface the generator subsystems with each other. These include mechanical power P_M, electrical power P_E, field voltage V_F, supplementary output V_S, and machine angle and speed δ and ω. The interface is as shown in Figure 2-1.

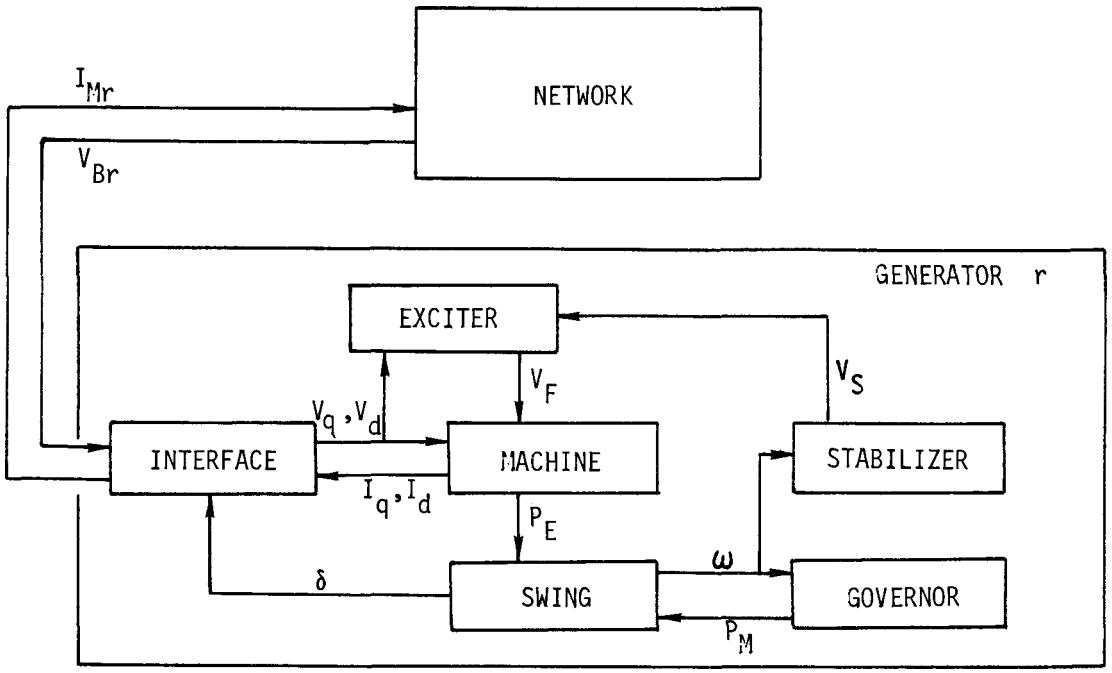


Figure 2-1. Generator System Interfaces

The only equations associated with the interface system are the network interface equations

$$V_{Br} = e^{j\delta}\,(V_q - jV_d)$$
$$I_{Mr} = e^{j\delta}\,(I_q - jI_d) \qquad (2\text{-}4)$$

The swing system is associated with the equations of rotational inertia of the machine

$$(d/dt)\,\delta = \omega$$
$$2H\cdot(d/dt)\,\omega + D\omega = P_M - P_E \qquad (2\text{-}5)$$

The machine equations relate the d-axis and q-axis voltages and currents by means of the electrical equations associated with the rotor and stator windings. The details vary from model to model, but for some models the machine system includes variables and state equations associated with rotor winding flux linkages. In general, the equations take the form

$$(d/dt)\,X_M = f_M(X_M, Y_M, Y_I)$$
$$0 = g_M(X_M, Y_M, Y_I) \qquad (2\text{-}6)$$

where X_M represents machine state variables (usually flux linkages), Y_M represents other machine variables, and Y_I represents interface variables (including V_d, V_q, I_d, I_q, V_F, and P_E).

The exciter system consists of equations that relate the field voltage V_F to the terminal voltage V_T. The details of the equations depend on the type of exciter being modeled. Most are state (differential) equations. However, algebraic equations may be used to simulate limiter blocks or lead-lag blocks. In general, the equations take the form

$$V_T = \sqrt{V_d^{\,2} + V_q^{\,2}}$$
$$(d/dt)\,X_E = f_E(X_E, Y_E, Y_I) \qquad (2\text{-}7)$$
$$0 = g_E(X_E, Y_E, Y_I)$$

where X_E represents the exciter state variables, Y_E represents other exciter variables (if any), Y_I represents interface variables (including V_d, V_q, V_F).

The stabilizer system consists of equations that relate the supplementary output V_S to the machine speed error ω. The general form of the equations is similar to that of the exciter system

$$\begin{aligned} (d/dt)X_S &= F_S(X_S, Y_S, Y_I) \\ 0 &= g_S(X_S, Y_S, Y_I) \end{aligned} \tag{2-8}$$

where X_S represents the stabilizer state variables, Y_S represents other stabilizer variables, and Y_I represents interface variables (including ω and V_S).

The governor system consists of equations that relate mechanical power P_M to the machine speed error ω. The general form of the equations is similar to those of the exciter and stabilizer.

$$\begin{aligned} (d/dt)X_G &= f_G(X_G, Y_G, Y_I) \\ 0 &= g_G(X_G, Y_G, Y_I) \end{aligned} \tag{2-9}$$

where X_G represents the governor state variables, Y_G represents other stabilizer variables, and Y_I represents interface variables (including ω and P_M).

Equation (2-1) represents the network, equations (2-2), (2-3) represent the loads, and equations (2-4)-(2-9) represent the generator systems. Together these equations constitute the transient stability model. It is completely independent of any solution algorithm. How the solution algorithm is adjoined to the stability model will now be discussed.

Algorithm Formulation

Two distinct philosophies for solving the stability model equations have been studied and implemented in the diagnostic program. They are the partitioned explicit approach, and the simultaneous implicit approach. With either approach the first step is to replace the derivative term $(d/dt)X$ with a finite difference approximation

$$(d/dt)X_n \simeq D(X_n, t_n, X_{n-1}, f(X_{n-1}), t_{n-1}) \tag{2-10}$$

Here the subscript refers to time, X_n is the value of X at time t_n. The difference function D depends on the integration method being used. After a method has been selected, equation (2-10) is used to substitute D for (d/dt)X in equations (2-5)-(2-9). Labeling the new equations (2-5')-(2-9'), the system of equations (2-1)-(2-4), (2-5')-(2-9') is an algebraic system which must be solved at each time point to compute the system variables.

The PE approach to solving this system is to guess (based on past data) a value for the bus voltage vector V. Given V, the load currents I_L can be computed from (2-2) and (2-3); and the generator currents I_G can be computed from (2-4), (2-5')-(2-9') (as can be seen from Figure 2-1). Given I_L and I_G a correct value of V can be computed from (2-1). The point is that instead of solving all the generator equations and all the load equations and the network equations all simultaneously, each load and each generator is handled separately.

The Simultaneous Implicit Approach

The SI approach is to solve (2-1)-(2-4), (2-5')-(2-9') all simultaneously using a variation of Newton's method. Each nonlinear equation is approximated by a linear equation. As can be seen from Figure 2-1 (or from equations (2-1)-(2-9)), the resulting system matrix (see Figure 2-2) has a blocked structure.

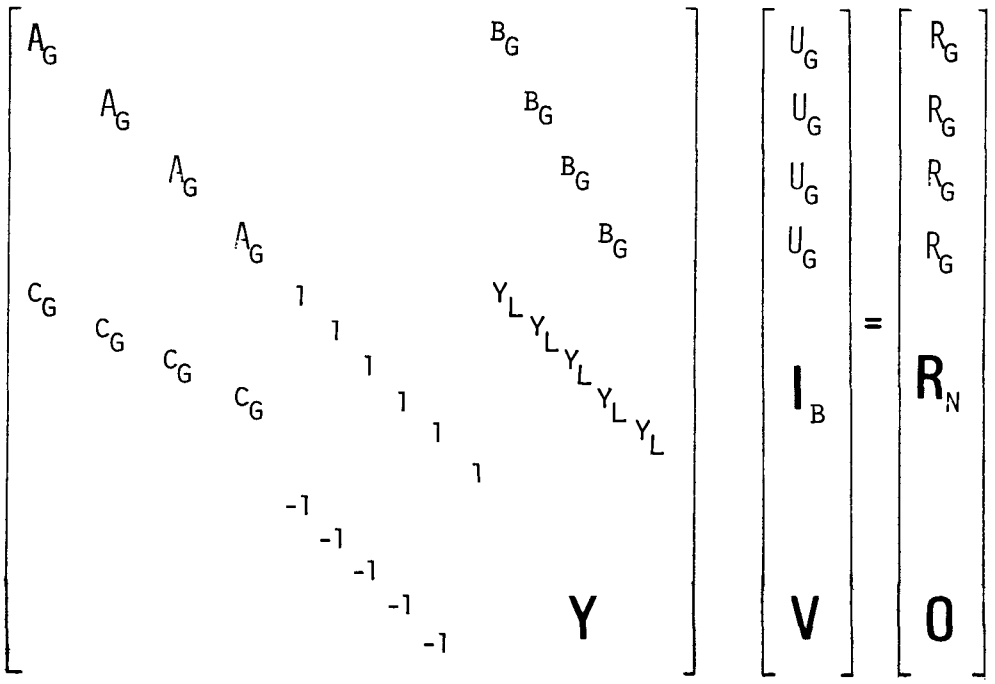


Figure 2-2. System Matrix Structure

Each generator and load is connected to the network through a bus voltage and current. Here U_G represents the composite vector of all generators variables (Y_I,X_M,Y_M,X_E,Y_E,X_S,Y_S,X_G,Y_G), A_G consists of terms in the system matrix depending on U_G (all terms from (2-5')-(2-9') and all but the V_{Br} term from the first part of (2-4)), B_G is the coefficient of V_{Br} from (2-4). The right hand side is the constant term in the linear approximation. The variable I_B is I_G-I_L, the I_G contribution comes from equation (2-4) through the C_G terms, the I_L contribution comes from equations (2-2),(2-3) through the Y_L terms. The last block equation is equation (2-1). This system can be simplified by adding the last two block equations (see Figure 2-3). Here $Y + Y_L$ represents the network admittance matrix with the diagonal term Y_L added at each load bus. This is the system actually solved by the diagnostic program.

$$\begin{bmatrix} A_G & & & & B_G & & & & \\ & A_G & & & & B_G & & & \\ & & A_G & & & & B_G & & \\ & & & A_G & & & & B_G & \\ C_G & & & & & & & & \\ & C_G & & & & & & & \\ & & C_G & & & Y + Y_L & & & \\ & & & C_G & & & & & \end{bmatrix} \begin{bmatrix} U_G \\ U_G \\ U_G \\ U_G \\ \\ \\ V \\ \\ \end{bmatrix} = \begin{bmatrix} R_G \\ R_G \\ R_G \\ R_G \\ \\ \\ R_N \\ \\ \end{bmatrix}$$

Figure 2-3. Simplified System Matrix Structure

The solution method used in the diagnostic program to solve this system is based on the block structure shown in Figure 2-3. For each generator U_G is computed in terms of V_{Br} (which is unknown) and R_G (which is known).

$$\left[U_G\right] = -\left[A_G^{-1}\right]\left[B_G\right] \cdot V_{BR} + \left[A_G^{-1}\right]\left[R_G\right] \tag{2-11}$$

Then

$$\left[C_G\right]\left[U_G\right] = -\left[C_G\right]\left[A_G^{-1}\right]\left[B_G\right]V_{BR} + \left[C_G\right]\left[A_G^{-1}\right]\left[R_G\right] \tag{2-12}$$

This is substituted into the network equation to give

$$\left[Y + Y_L + Y_G\right]\left[V\right] = \left[R_N'\right] \tag{2-13}$$

Here Y_G is a diagonal matrix with

$$-\left[C_G\right]\left[A_G^{-1}\right]\left[B_G\right]$$

on the diagonal corresponding to each generator bus and

$$-\left[C_G\right]\left[A_G^{-1}\right]\left[R_G\right]$$

added to the terms of R_N corresponding to each generator bus. Equation (2-13) is solved for V, and then the values of V_{Br} are substituted into (2-11) to compute U_G.

Computing the System Matrix

The next question that will be addressed is how is the system shown in Figure 2-3 generated.

The diagnostic program has subroutines for evaluating mismatches for each of the algebraic equations (2-2), (2-4)-(2-9), and (d/dt)X is computed for each state variable. In addition, these routines compute partial derivatives that will be required later for Jacobian calculations. For example, a routine for a machine model (equation (2-6)) would compute $(d/dt)X_m$ which is f_M, and the algebraic mismatch which is g_M. It would also calculate $\partial f_M/\partial X_M$, $\partial g_M/\partial X_M$, $\partial f_M/\partial Y_M$, $\partial g_M/\partial Y_M$, etc. It is important to note that the coding of this routine depends only on the machine equations, and is independent of either integration or algebraic method.

The next step is to replace the differential equations with algebraic difference equations. This is performed by a routine that depends on the integration algorithm, but is independent of the system model. Thus, one routine is required for the trapezoidal rule, another for the APS rule, etc. Equation (2-10) is used in this routine. For example, the difference equation for the trapezoidal rule is

$$(d/dt)X_n = -f(X_{n-1}) + 2/(t_n-t_{n-1})\ (X_n-X_{n-1}) \qquad (2\text{-}14)$$

The mismatch for equation (2-14) is the left hand side (which is computed by the modeling routine) minus the right hand side (which is computed by the method routine). The Jacobian for (2-14) is the Jacobian of the left hand side (which is computed by the modeling routine) minus $2/(t_n-t_{n-1})$ on the diagonal.

These linear algebraic difference equations, and the linear approximations to the algebraic equations are collected to form the system matrix shown in Figure 2-3. This system is then solved as described above.

Thus, we can see that in the diagnostic program independence of models and methods has been maintained. The model routines can be used without modification with any method, and the method routines can be used with each of the models.

2.3 TEST CASES

A set of test cases has been collected for use in this research that has a wide variety of characteristics. The important requirements for a test series is that it thoroughly test all model capabilities to verify the models, and that it exhibits typical behavior for stability problems including the most common computational difficulties.

The WSCC nine bus test series was selected as a baseline for the test case series because it contains step-by-step verification of the models and, therefore,

automatically satisfies the first requirement. Other advantages of the nine bus series are that runs are inexpensive to make and to analyze, so many experiments can be performed throughout the research project. Also, the problems in the nine bus series are simple enough that they can be modified to exhibit a specific characteristic quite easily. In fact, this has been done to add unstable and marginally stable problems to the nine bus series.

One problem characteristic that is important in stability analysis and that cannot be added to the nine bus series is problem size. For this reason, a few problems larger than nine busses have been added to the test series. These consist of the 39 bus New England test system (see [1]), and a 274 bus test system from Bonneville Power Administration.

2.4 PROGRAM PERFORMANCE EVALUATOR

PPE is a software tool that can be used to measure the performance of any computer program in very great detail. It runs simultaneously with the program under evaluation, and periodically samples that program to determine its activity. The many thousands of samples are stored on a file for subsequent analysis. During this analysis it is determined what percentage of the total running time is spent in each significant activity.

In this research program PPE has been used on both the diagnostic program and the APS transient stability program. The results obtained from evaluation of the diagnostic program led to small modifications in the sparse matrix algorithms that improved performance by a factor greater than 2. Evaluation of the APS stability program identified a high activity in input/output related tasks. APS has elected to modify their stability program as a result of this analysis. The results of this modification are not available at this writing. Much of the timing data presented in this report has been compiled through PPE analysis.

2.5 REFERENCE

[1] de Mello, R. W., Podmore, R., Stanton, K. N., "Coherency Based Dynamic Equivalents for Transient Stability Studies", Final Report on Electric Power Research Institute Project RP 90-4, Phase II, December 1974.

Section 3

SIMULTANEOUS IMPLICIT INTEGRATION METHODS

3.1 INTRODUCTION

In this section we will discuss the problem of selecting a method for solving the mixed system of differential and algebraic equations described in Section 2. The discussion will be limited to those methods that are appropriate for use with a simultaneous implicit (SI) approach. Methods that are suitable for use with other approaches are described in Section 4. The problem under consideration will be decomposed into two subproblems, namely:

- selecting a difference equation
- controlling step size

This decomposition has been made because each of these two subproblems is mostly independent of the other with the best solution being influenced more by the underlying problem characteristics rather than the solution of the other subproblem.

The mathematics and engineering literatures are rich with integration algorithms (an integration algorithm is a difference equation with a step size strategy). In practice, a great many of these algorithms have merit when applied to problems with the appropriate characteristics. Therefore, it is important at the outset to identify the characteristics of the transient stability problem that are relevant to this study. For the purposes of this research we assumed these characteristics to be as follows:

- The purpose of the programs is to evaluate the stability of systems rather than to predict responses of the system.
- Only enough accuracy is required to accomplish this purpose. Two-three significant figures per step seems to be sufficient.
- Discontinuities in the first and higher order derivatives of the state variables occur frequently during the course of a simulation.
- A moderate degree of stiffness can be expected.

The subject of stiffness is a complex and difficult one and its influence on algorithms is profound. For these reasons, it will be discussed in greater detail later in Appendix A.

From considerations based on the problem characteristics several difference equations were selected for evaluation. These equations and the considerations that led to their selection are described in Section 3.2. Three of these equations were considered sufficiently promising to merit testing. The three are trapezoidal rule, APS rule, and implicit midpoint rule. These three rules were implemented on the diagnostic program and tested on test cases designed to exercise them under all typical conditions. These test cases are described in Section 3.4, and the results of these tests in Section 3.5.

The two major criteria used in evaluating the alternative algorithms were reliability and efficiency. The algorithms were also evaluated with respect to applicability to midterm and long term dynamics. All of these criteria are fully discussed in Section 3.3.

The ultimate numerical reliability and efficiency of any algorithm depends in an important way on the method used to select step size. Step size control will be discussed in Section 3.6.

3.2 DIFFERENCE EQUATIONS CONSIDERED

In this section the difference equations that were studied in this research project will be discussed. The strong and weak points of each method will be listed, and the order and numerical stability properties of each will be given. However, before this is done, order and numerical stability will be discussed.

Order (see [1], § 2, and § 10.1 for greater details)

Any difference equation intended for the solution of differential equations is designed so that the solution to the difference equation is a good approximation to the solution to the differential equation, especially for small step sizes. This is achieved by making the first few terms of the power series expansion of the solution to the differential equation agree with the corresponding terms of the power series expansion of the solution to the difference equation. Consider the differential equation

$$\dot{x} = f(x) \tag{3-1}$$

with initial value

$$x(t_o) = x_o$$

The solution to (3-1) can be expanded in a power series at time $t_1 = t_o+h$

$$x(t_1) = x_o+x_o'h+\frac{1}{2}x_o''h^2+... \qquad (3-2)$$

If we use the difference equation

$$\dot{x}(t_1) = D(x_1,t_1,x_o,t_o,...) \qquad (3-3)$$

(where x_1 denotes the computed value of x at time t_1) to solve (3-1) we substitute for $\dot{x}(t_1)$ in (3-3) to get

$$f(x_1) = D(x_1,t_1,x_o,t_o,...) \qquad (3-4)$$

Equation (3-4) can be solved for x_1. Expanding in a power series we get

$$x_1 = x_o+A_1h+A_2h^2+... \qquad (3-5)$$

Comparing (3-2) with (3-5) we see that x_1 will be a good approximation to $x(t_1)$ if $A_1 = x_o'$, $A_2 = \frac{1}{2}x_o''$, etc. In particular if

$$A_n = x_o^{(n)}/n! \qquad n = 1,...,p$$

then the difference equation (3-3) is said to have order p. It can be seen from equation (3-2) that for very small step size h the solution $x(t_1)$ is determined primarily by the low order terms in the expansion, with the contributions from the high order terms decaying rapidly. In this case, high order methods can be expected to give highly accurate results. For larger step sizes, the higher order terms in (3-2) may contribute significantly to $x(t_1)$. In this case, the behavior of the coefficients A_n as n tends to infinity may have more significance in determining the accuracy than the order p. As a general rule, it is found that if very high accuracy is required, then a high order method with a moderate step size is the most efficient algorithm. However, when lower accuracy is acceptable, a lower order method with a moderate step size is more efficient and reliable. Because of the problem characteristics of the transient stability problem, it was expected that low order integration (p=2) would be best, and results proved this out.

Numerical Stability (see [1] for greater details)

When we study the numerical stability of the difference equation (3-3) we are asking the question, "Does the computed solution to (3-3) decay to a constant, achieve a limit cycle, or blow up to infinity whenever the true solution decays to a constant, achieves a limit cycle, or blows up to infinity?". More specifically, given the equation (3-1) and a fixed step size h, let $t_n = t_o+nh$. Consider the sequence $x(t_o)$, $x(t_1)$,...; where x(t) is the true solution to (3-1). Also consider the sequence $x_o,x_1,\ldots$, where x_n is the computed solution and comes from repeated solution of the difference equation (3-4). It is highly desirable that the two sequences exhibit the same behavior (decay, oscillation, or growth). In practice, the behavior demonstrated by the approximate solution $x_o,x_1,\ldots$ depends in a complicated way on the function f, the difference operator D, and the step size h. There is no known difference operator that generates solutions that will exhibit the proper behavior for every function f and step size h. Since we cannot find a D that works for every f and h, we will look at difference operators that have the desired properties for linear f and all h. For this reason we restrict our attention to the linear differential equation

$$\dot{x} = \lambda x \tag{3-6}$$

where λ is a complex constant. It is well known that the solution to this problem will decay to 0 if $Re(\lambda)<0$, it will oscillate if $Re(\lambda) = 0$, and it will blow up to infinity if $Re(\lambda)>0$.

For any difference operator D and step size h we ask for which values of λ does the computed solution $x_o,x_1,\ldots$ decay to 0, for which values of λ does it oscillate, and for which values of λ does it blow up. We ask these questions independent of the accuracy or order of the method.

Those values λ for which the solution decays comprise the stability region for the difference equation. On the boundary of this region (in the complex plane) the solution will oscillate. Outside of the stability region and its boundary, the solution will blow up. A difference operator D is called symmetrically A-stable if its region of stability is the open left half plane. If a method is symmetrically A-stable, the approximate solution to (3-6) will demonstrate the same stability as the true solution for every complex value of λ and for every step size h. Since they characterize the stability of the test problem (3-6), symmetrically A-stable methods are well suited to stability studies. Any method that is not symmetrically

A-stable has anomalous stability regions. That is, there are values for λ in the left half plane where the computed solution is unstable, or values for λ in the right half plane where the computed solution is stable, or both.

The Trapezoidal Rule

The difference equation for the trapezoidal rule is

$$\dot{x}(t_1) = 2/h(x_1-x_o-(h/2)f(x_o)) \tag{3-7}$$

When this difference equation is substituted into the differential equation (3-1) we get

$$x_1-x_o = (h/2)(f(x_o)+f(x_1)) \tag{3-8}$$

This is the way the trapezoidal rule is usually written. It can be seen from either (3-7) or (3-8) that this rule is implicit. That is, when x_o (and, therefore, $f(x_o)$) is known, the implicit equation (3-8) must be solved to obtain x_1.

The essential properties of the trapezoidal rule are that it has order two and is symmetrically A-stable (see Appendix A).

The trapezoidal rule is used in the Bonneville Power Administration transient stability program [2] and the electromagnetic transients program [3]. It is widely used in many engineering applications and performs reliably. For these reasons, and the relative simplicity of the method, the trapezoidal rule was the initial difference equation implemented in the diagnostic program.

The Implicit Midpoint Rule

The difference equation for the implicit midpoint rule (IMP) is

$$\dot{x}((t_o+t_1)/2) = (x_1-x_o)/h$$

$$x((t_o+t_1)/2) = (x_o+x_1)/2$$

When this difference equation is substituted into the differential equation (3-1) we get

$$x_1-x_o = hf((x_o+x_1)/2) \tag{3-9}$$

This is the way the IMP rule is usually written. It can be seen by comparing (3-9) with (3-8) that if the function f is linear, the IMP rule is equivalent to the trapezoidal rule. It is then not surprising that the two rules are quite similar.

The IMP rule is implicit, it has order two and it is symmetrically A-stable. However, for nonlinear functions f, the IMP rule is not equivalent to the trapezoidal rule and, in fact, it has some superior theoretical properties. In particular, the truncation error coefficients (the difference between A_n and $x_o^{(n)}/n!$ in equation (3-6) for h>2) are smaller for the IMP rule than for the trapezoidal rule.

The IMP rule is not widely known or used, but because of its attractive theoretical properties, it was implemented in the diagnostic program.

The APS Rule

The integration procedure used in the APS transient stability program (see [4]) is not a basic difference equation (as the previous two rules are) that can be applied to any differential equation. Rather, it is a procedure that can be applied to systems of differential equations with a particular kind of structure and which have been partitioned into subsystems. The results obtained depend not only on the system being solved, but also on the way the system is partitioned. The transient stability problem has a natural partitioning (as described in Section 2) that is compatible with this procedure.

The idea of this procedure is to partition the system of equations into dynamic linear subsystems that are coupled by algebraic equations that may be nonlinear. The equations for the r^{th} linear subsystem takes the form

$$\dot{x}_r = A_r x_r + B_r u_r$$
$$y_r = C_r x_r + D_r u_r \qquad (3\text{-}10)$$

where x_r, y_r, u_r are respectively the vectors of state variables, output variables, and input variables to the r^{th} linear subsystem. The independent equations (3-10) are coupled by computing the inputs to each subsystem from the outputs of the other subsystems.

$$u = f(y) \qquad (3\text{-}11)$$

where u and y are the combined vectors of all subsystem inputs and outputs, respectively. The difference equation for the system is obtained by solving (3-10) for x_r at time t_{n+1}

$$x_r(t_{n+1}) = T_r(t_{n+1}-t_n)x_r(t_n)+\int_{t_n}^{t_{n+1}} T_r(t_{n+1}-t)B_r u_r(t)\,dt \qquad (3\text{-}12)$$

where

$$T_r(t) = \exp(A_r t) \qquad (3\text{-}13)$$

If one assumes that over the time interval from t_n to t_{n+1} $u_r(t)$ varies linearly from $u_r(t_n)$ to $u_r(t_{n+1})$ then the integral in (3-12) can be evaluated by integration by parts to give

$$x_r(t_{n+1}) = T_r(h)x_r(t_n)+W_{1r}u_r(t_n)+W_{2r}u_r(t_{n+1}) \qquad (3\text{-}14)$$

where

$$W_{1r} = (A_r^{-1}(T_r(h)-I)-A_r^{-2}h^{-1}(T_r(h)-I-A_r h))B_r$$

$$W_{2r} = A_r^{-2}h^{-1}(T_r(h)-I-A_r h)B_r \qquad (3\text{-}15)$$

$h = t_{n+1}-t_n$ is the step size

The first two terms of (3-14) depend on $x_r(t_n)$ and $u_r(t_n)$ which are both known, so equation (3-14) can be written

$$x_r(t_{n+1}) = K_r(t_n)+W_{2r}u_r(t_{n+1}) \qquad (3\text{-}16)$$

where $K_r(t_n)$ is known and $u_r(t_{n+1})$ must be determined. Equation (3-16) can be used to compute $y_r(t_{n+1})$.

$$y_r(t_{n+1}) = C_r K_r(t_n)+(C_r W_{2r}+D_r)u_r(t_{n+1}) \qquad (3\text{-}17)$$

The linear equations (3-17) are then substituted into equation (3-11) for each subsystem r. The resulting equation is solved for $u_r(t_{n+1})$. Once u is known all the y_r's can be found from (3-17), and the x_r's from (3-14). Equation (3-14) is the difference equation for the method.

If a particular subsystem has a nonlinear term, it can be handled with this procedure simply by breaking the term out of the subsystem and including it in f in (3-11). This will add an output to the nonlinear term and an input from the nonlinear term to the model.

It can be seen from examining (3-12)-(3-17) that the results of applying the APS

procedure to a system depends not only on the system and the step size, but also on how the system is partitioned. In particular, if all the terms in each subsystem are included in f as described above, then $A_r = 0$ for each subsystem and the method will be equivalent to the trapezoidal rule. Because of the dependence on the partition, it is difficult to make general statements about the APS procedure. However, it can be shown that in all cases the method has order two. It can also be shown that if a problem has significant coupling through the f term in (3-11), a stable problem can have an unstable solution and vice versa (see Appendix A). Because the APS transient stability program solves problems with good speed and reliability using this procedure, it was implemented in the diagnostic program.

Backward Differentiation Formulae

There has been a great deal of interest recently in the backward differentiation formulae (BDF), and these methods have been applied widely in chemistry, electronics, and other engineering disciplines. The VISTA project [5] was an attempt to apply this approach to transient stability.

The BDF are not a single difference equation but a series of difference equations [6] with orders varying from one to six. The six equations have been combined into a single algorithm which selects the most appropriate equation to use at any time during the solution by analyzing the past history of the solution (see [6]). Complex algorithms of this type are required to implement the BDF because all the equations with order higher than 1 are not self-starting. This means that because of the requirements for past history, one must start with the order 1 equation, take a few steps to build up some history, then increase order to 2, etc. Once a high order has been built up, the stability problem may encounter one of its frequent discontinuities. When this happens, the past history is no longer valid and the algorithm must start all over at order 1.

The principal reason for the recent interest in the BDF is the stability of the equations. All of the equations have very large regions of stability encompassing most of the left half plane and much of the right half plane as well (see [1]). This makes them ideally suited for stiff-stable problems with rapid time constants and quiescent modes. They are less well suited to stability problems where their overly large regions of stability can make unstable problems have stable solutions, and where not all fast modes are quiescent.

The BDF were not implemented in the diagnostic program because of the complexity

of the algorithms, because of the difficulties associated with discontinuities, because of its extreme stability properties, and because of difficulties reported by the VISTA project with the fast exciter time constants.

A New Multistep Method Suggested by Gross and Bergen

In [7], G. Gross and A. R. Bergen suggested a new nonlinear multistep method that is, in fact, a hybrid of multistep methods that have previously been suggested. The novelty of their approach lies in their noticing that a particular nonlinear implicit multistep method, when applied to the second order swing equation

$$2H\dot{\delta} = D\omega + T_a$$

becomes explicit in the machine angle δ.

Because the nonlinearity associated with the network-machine interface is a function of δ, a method that is explicit in δ will not exhibit this nonlinearity. In fact, when the simpler machine and load models are used the algebraic equations become linear, and can be solved very rapidly. However, when detailed models and nonlinear loads are being used the advantages are not so great.

Gross and Bergen suggested using the BDF to solve the remaining dynamic equations. It was decided not to implement this approach because of its heavy dependence on the BDF. However, this approach does appear to have certain advantages and it is felt that it would merit further testing on the diagnostic program using some other algorithm for the dynamic components other than the swing system.

3.3 CRITERIA FOR EVALUATION

The criteria for evaluation used in this study fall into two major categories: computational reliability and efficiency. These criteria were used to reduce the list of candidate methods down to those that were actually implemented and tested, they were used in evaluating the test results, and they will be used in fine tuning the methods for a production grade program.

The two areas of computational reliability and efficiency are not completely independent. Trade-offs can always be made to achieve more computational reliability

at the expense of higher incurred costs. These trade-offs are most significant in the area of step size control, which will be considered separately in Section 3.6.

Computational Reliability

As stability problems become larger and more complex, it becomes less and less realistic to use intuition to test the results of transient stability analysis. For that reason, it becomes more and more important to use algorithms that can be counted on to produce correct results.

A major consideration in the reliability of an algorithm is its numerical stability properties. Algorithms that are too numerically stable, combined with step sizes that are too large, produce results that are very stable, reasonable, and appealing to the intuition. Therefore, overly stable algorithms without step size control must be rejected. Algorithms with regions of stability that approximate the left half plane must be viewed favorably, with symmetrically A-stable methods very appealing from a theoretical point of view.

Unfortunately, it is difficult to come up with quantitative measures of computational reliability. For this reason, several marginal test cases were developed and run to test the reliability of the algorithms on some difficult cases.

Efficiency

The measure of efficiency is cost, and two kinds of cost are considered. One is the costs associated with developing and maintaining a code, the other is the costs associated with running the code.

The development and maintenance costs will depend on the simplicity of the algorithms used, the number of algorithms used, the generality of the algorithms, and their range of applicability. The high costs that can be involved in developing and maintaining large numbers of complex algorithms are evident. However, if a stability program is going to have a long lifetime, costs associated with major overhauls of the program may exceed the original development costs. Many of these expenses can be avoided by choosing algorithms that will be suitable for the expanded scope of the program. For example, it may prove to be false economy to select an algorithm for its performance in transient stability problems, if the algorithm is unsuitable for mid-term or long-term dynamics. Algorithms that are simple and applicable to a wide range of power system dynamics problems will minimize development and maintenance costs.

The operational costs are primarily determined by the solution times, with memory requirements and I/O requirements playing a significant role only if they are extraordinary. Another factor affecting operational costs is the information content of a particular computer run which will affect the number of runs required to design or analyze a system.

The solution time required to solve a problem is given by

solution time = (time per iteration)(iterations per step)(number of steps)

The choice of integration algorithm can influence all three of these factors. Accuracy of the methods is not used as a criterion for selecting an algorithm. This is because all of the methods can be made as accurate as desired by taking sufficiently small steps. Thus the solution times for each algorithm will be adjusted to reflect comparable accuracies. This is accomplished by estimating the step size required to achieve a prescribed accuracy, and then multiplying the number of steps by the ratio of the actual step size to the required step size.

The criteria for evaluating integration algorithms are summarized in Table 3-1.

Table 3-1
CRITERIA FOR EVALUATION

I. Reliability
   A. Stability
   B. Marginal test case results
II. Efficiency
   A. Development and maintenance costs
      1. Simplicity
      2. Generality
   B. Operational costs
      1. Solution time
         a. time per iteration
         b. iterations per step
         c. number of steps
      2. Memory and I/O requirements
      3. Information content

This table will be referred to later in evaluating the algorithms.

3.4 TEST CASES

The three simultaneous implicit algorithms that were implemented on the diagnostic program were tested on a series of transient stability problems that were selected to test the computational reliability and efficiency of the algorithms. Most of the test problems used in this research come from the nine bus test series which is described in Appendix C.

Test problems 1, 2, 3, 4, 14, 16 and 17 from the WSCC nine bus test series were used to verify that the algorithms were functioning properly. It was found that problem 4 was the most severe test of the algorithm because of the rapid bouncing back and forth between the output limits of the power system stabilizer. For this reason, results on problem 4 were used to judge efficiency of the algorithms. The 39 bus New England test system was also used to judge efficiency.

Modifications of problem 4 were used to test for reliability. The inertia of generator 2 was modified to make the system marginally stable. Marginally stable and marginally unstable runs were made. In a separate test the power system stabilizer gain constants, and limits, and the limits on field voltage were greatly increased to see if an instability or stiffness problems could be induced. It was found that unstable problems, even the marginally unstable problem terminated so rapidly that they were not the best sources of data for measuring efficiency. However, for problem 4 with gains and limits increased, no instabilities were induced and these results were used to judge efficiency as well as reliability.

3.5 TEST RESULTS

Two aspects of the results obtained from running the test cases will be considered in this section: computational reliability and efficiency.

Computational Reliability

Diagnostic runs were made to establish the computational reliability of the algorithms. The results of these runs are shown in Figures 3-1 through 3-14. Table 3-2 relates the test cases to the figures.

Five variations of WSCC nine bus test case #4 were run. These cases consisted of

- stable - WSCC case #4 (E_{MWS} = 640 for generator 2)
- marginally stable - case #4 with E_{MWS} = 450 for generator 2
- marginally unstable - case #4 with E_{MWS} = 430 for generator 2

Table 3-2

FIGURES FOR SECTION 3

FIGURE	TEST CASE	VARIABLE	ALGORITHMS
1	#4 stable	$\delta_2-\delta_1$	TRAP, APS, IMP
2	#4 stable	$\delta_3-\delta_1$	TRAP, APS, IMP
3	#4 marginally stable	$\delta_2-\delta_1$	TRAP, APS, IMP
4	#4 marginally stable	$\delta_3-\delta_1$	TRAP, APS, IMP
5	#4 marginally unstable	$\delta_2-\delta_1$	TRAP, APS, IMP
6	#4 marginally unstable	$\delta_3-\delta_1$	TRAP, APS, IMP
7	#4 unstable	$\delta_2-\delta_1$	TRAP, APS, IMP
8	#4 unstable	$\delta_3-\delta_1$	TRAP, APS, IMP
9	#4 high gain	$\delta_2-\delta_1$	TRAP, APS, IMP
10	#4 high gain	$\delta_3-\delta_1$	TRAP, APS, IMP
11	#4 high gain	V_s	TRAP, APS, IMP
12	39 bus	$\delta_2-\delta_1$	TRAP, APS
13	39 bus	$\delta_3-\delta_1$	TRAP, APS
14	39 bus	$\delta_9-\delta_1$	TRAP, APS
15	#4 stable	h_R/h_A-polynomial	TRAP
16	#4 high gain	h_R/h_A-polynomial	TRAP
17	39 bus	h_R/h_A-polynomial	TRAP
18	#4 stable	h_R/h_A-step doubling	TRAP

Table 3-3

SUMMARY OF TEST RESULTS

	TRAPEZOIDAL RULE			APS RULE		
	STABLE	HIGH GAIN	39 BUS	STABLE	HIGH GAIN	39 BUS
CPU time per iteration (msec.)	34.0	30.0	117.	40.4	34.8	147
Average iterations per time step	4.12	4.15	3.54	4.01	3.93	3.37
Numer of steps (actual)	214	93	142	214	93	142
Accuracy correction factor	1.3	1.7	1.3	1.5	2.0	1.0
Number of steps (adjusted)	278	158	185	321	186	142
CPU seconds for run (adjusted)	38.9	19.7	76.6	52.0	25.4	70.3

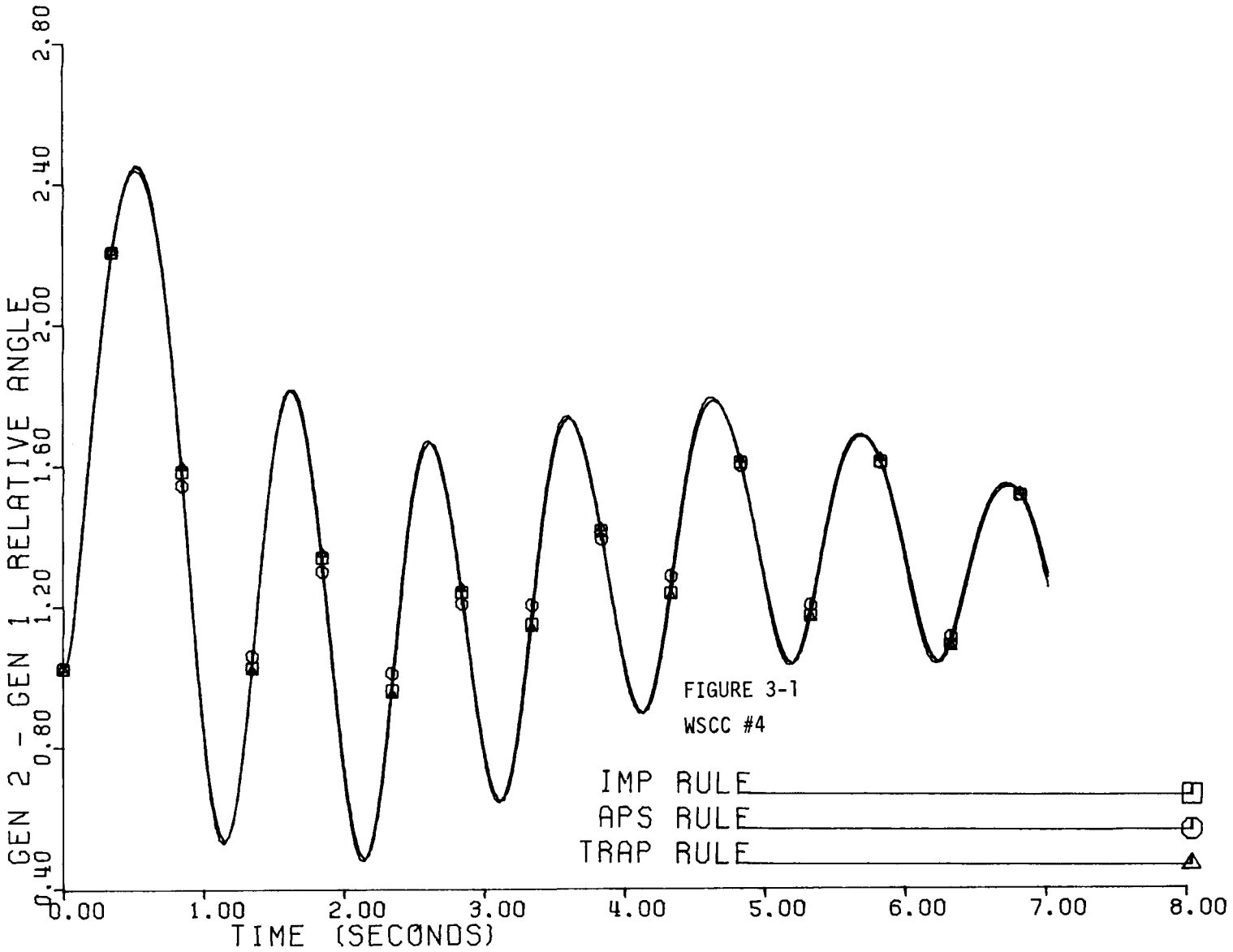


FIGURE 3-1
WSCC #4

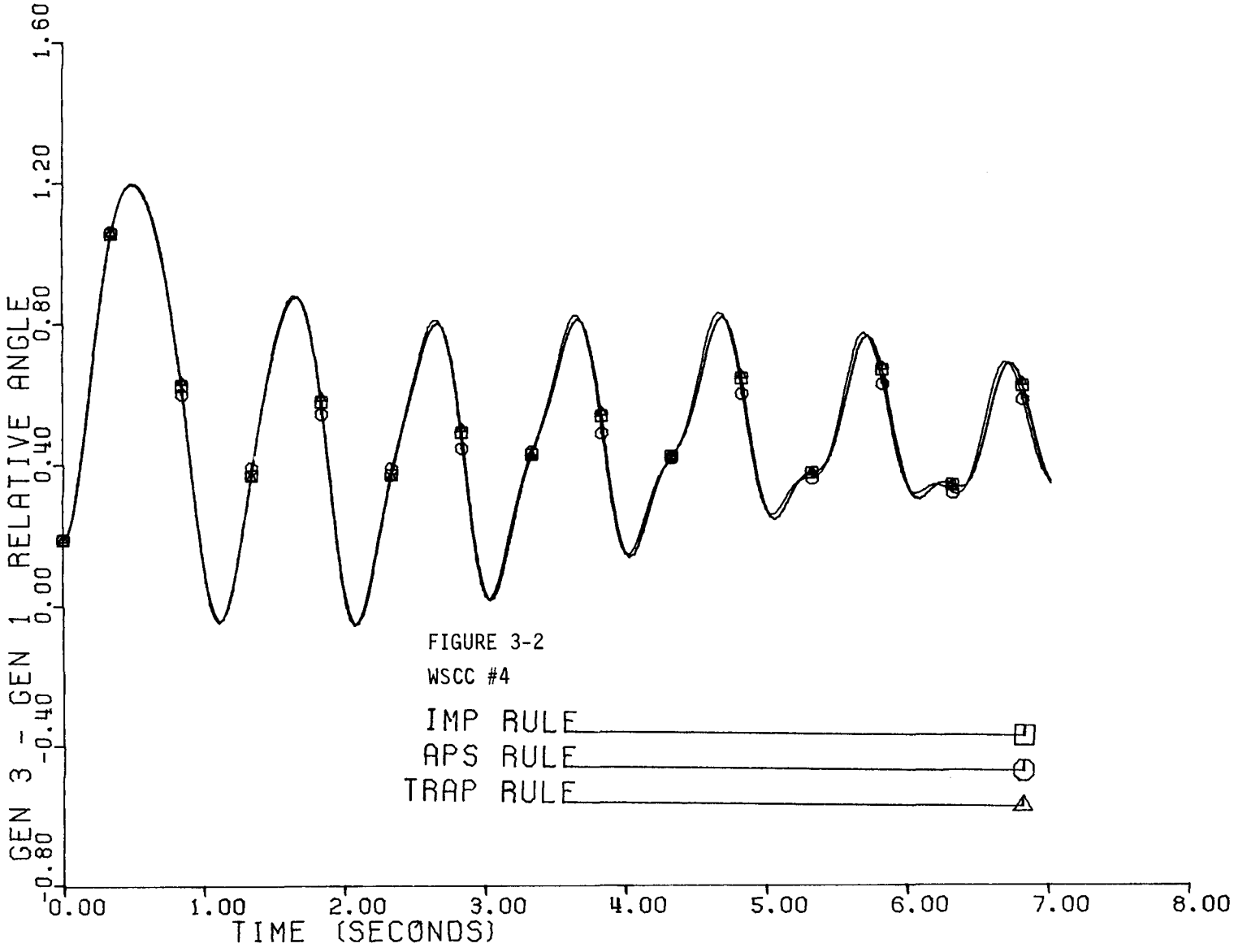


FIGURE 3-2

WSCC #4

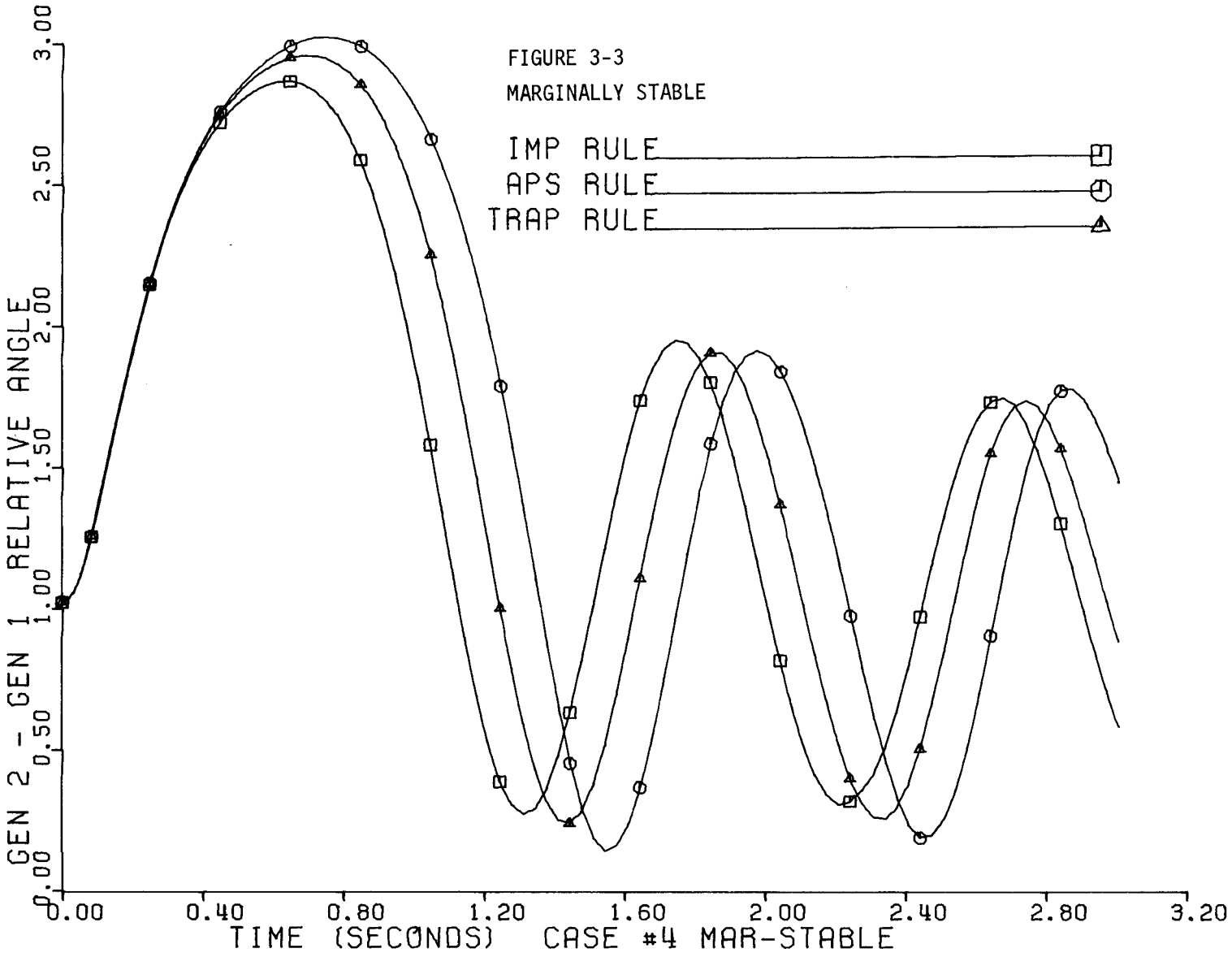


FIGURE 3-3

MARGINALLY STABLE

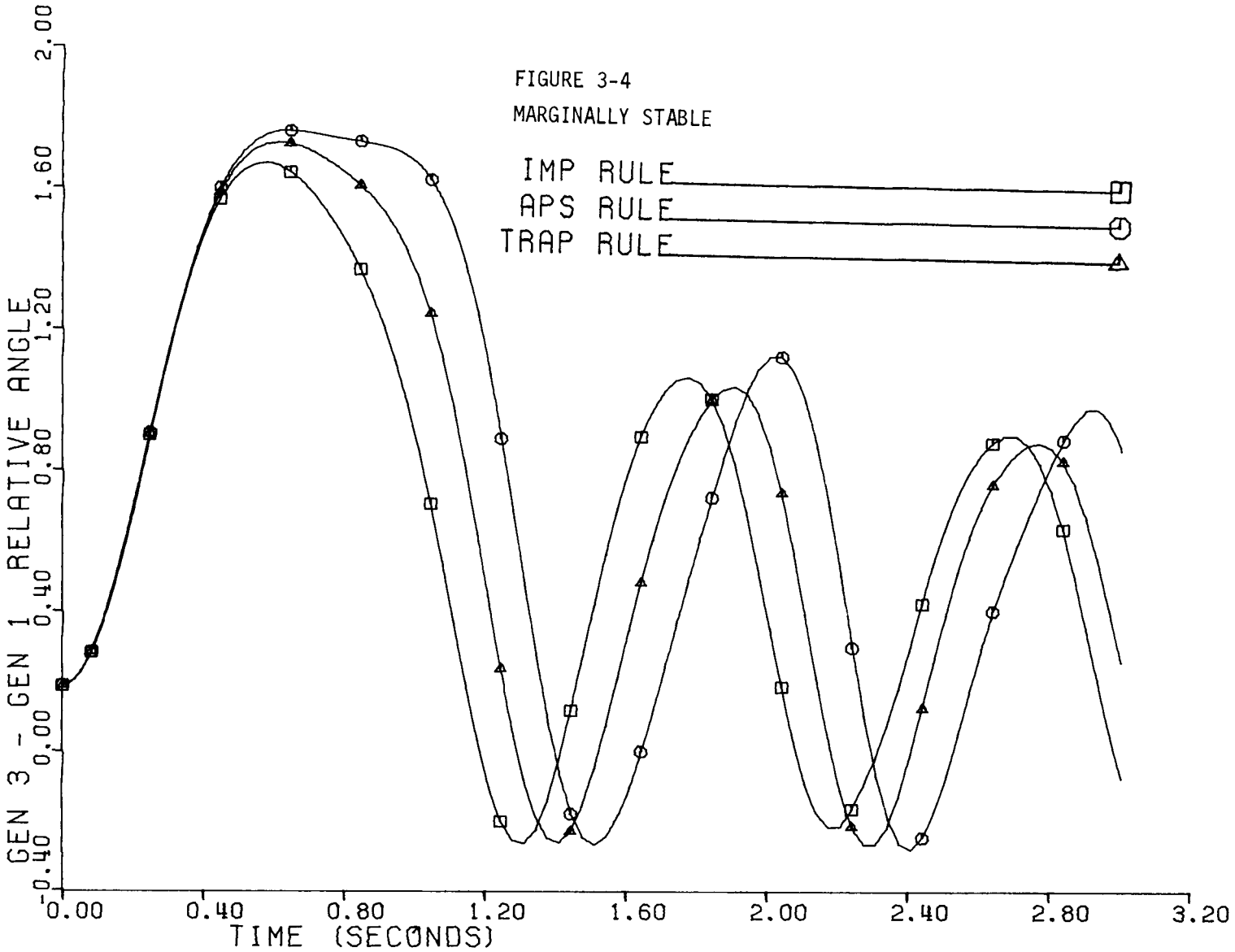


FIGURE 3-4

MARGINALLY STABLE

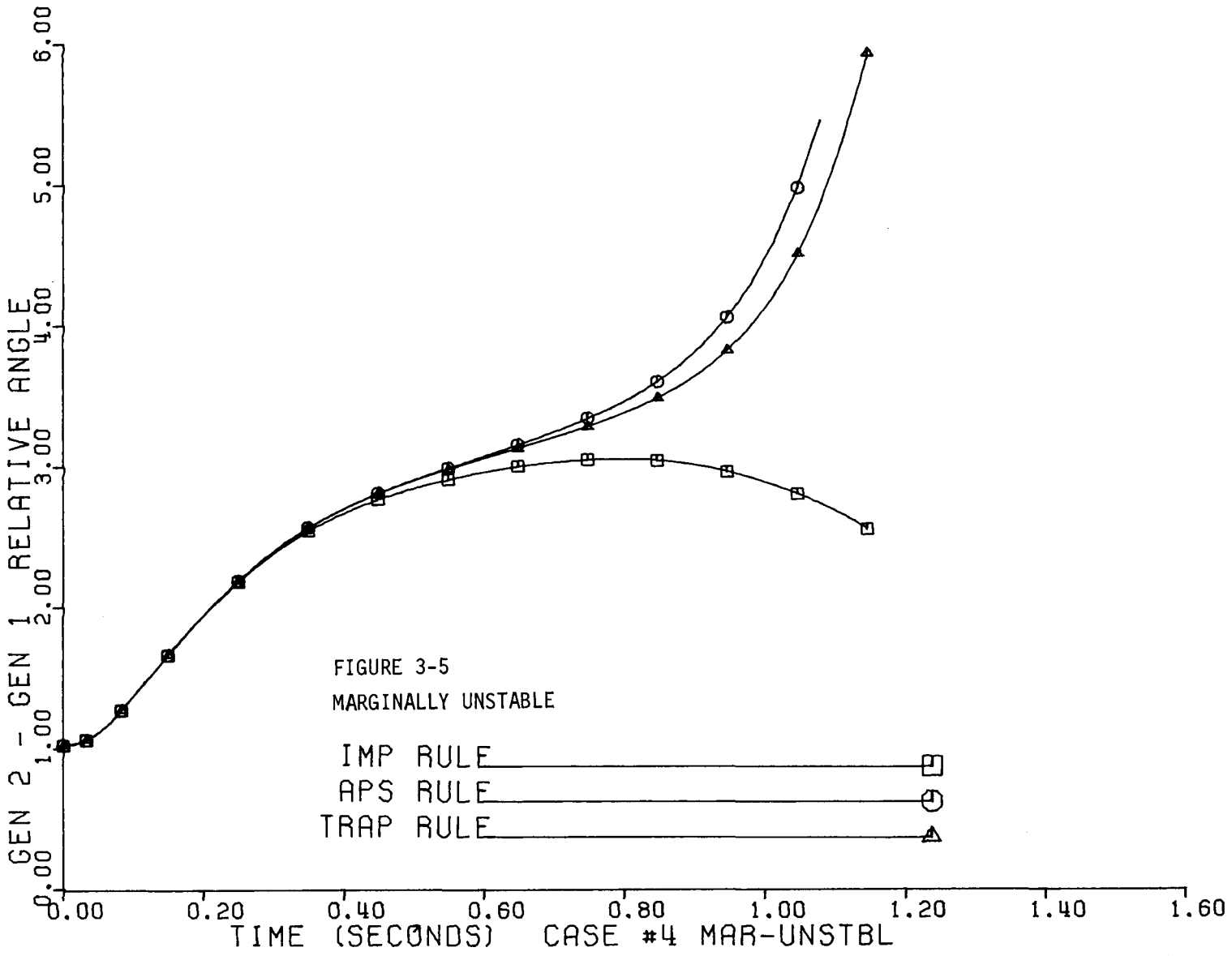


FIGURE 3-5

MARGINALLY UNSTABLE

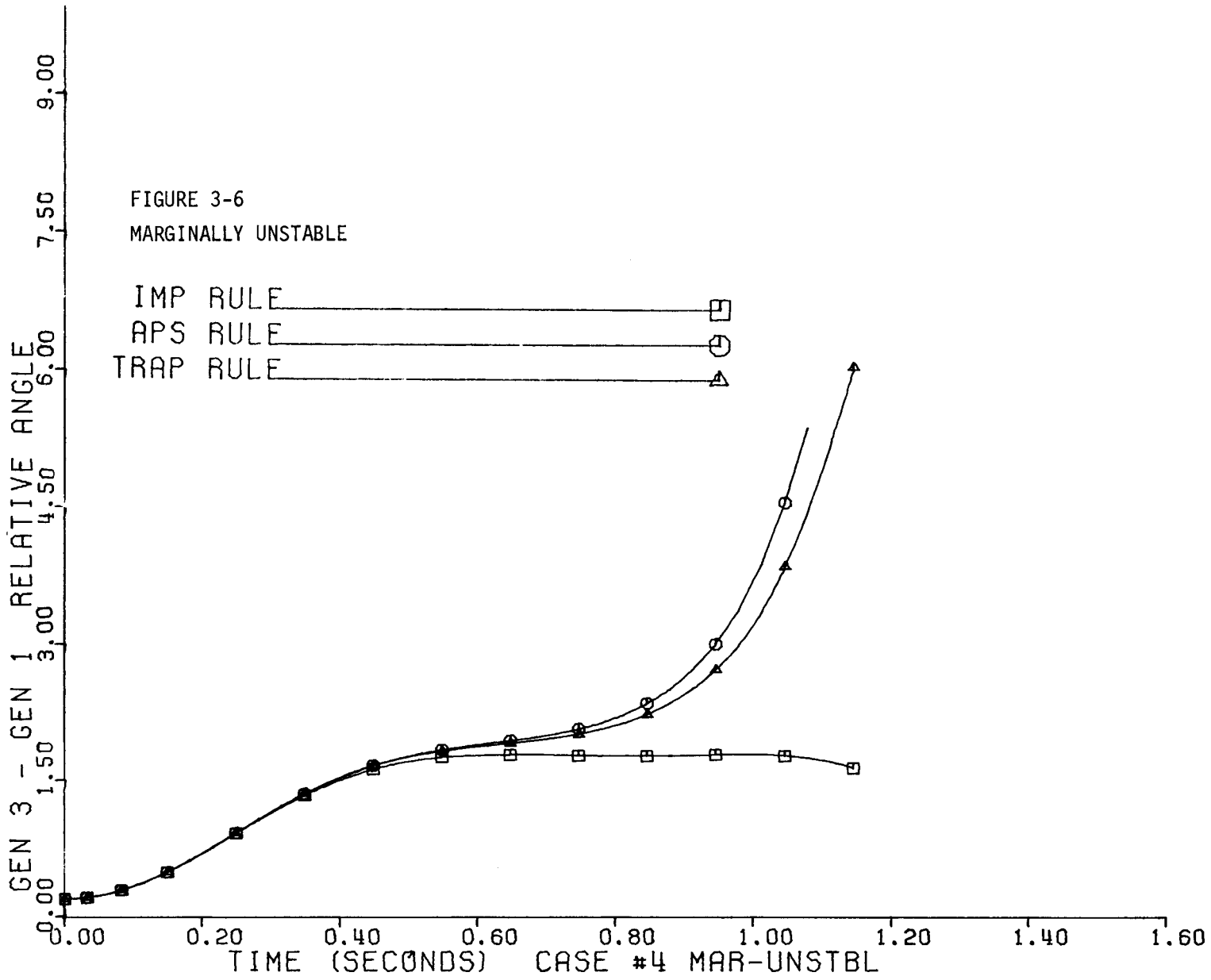


FIGURE 3-6

MARGINALLY UNSTABLE

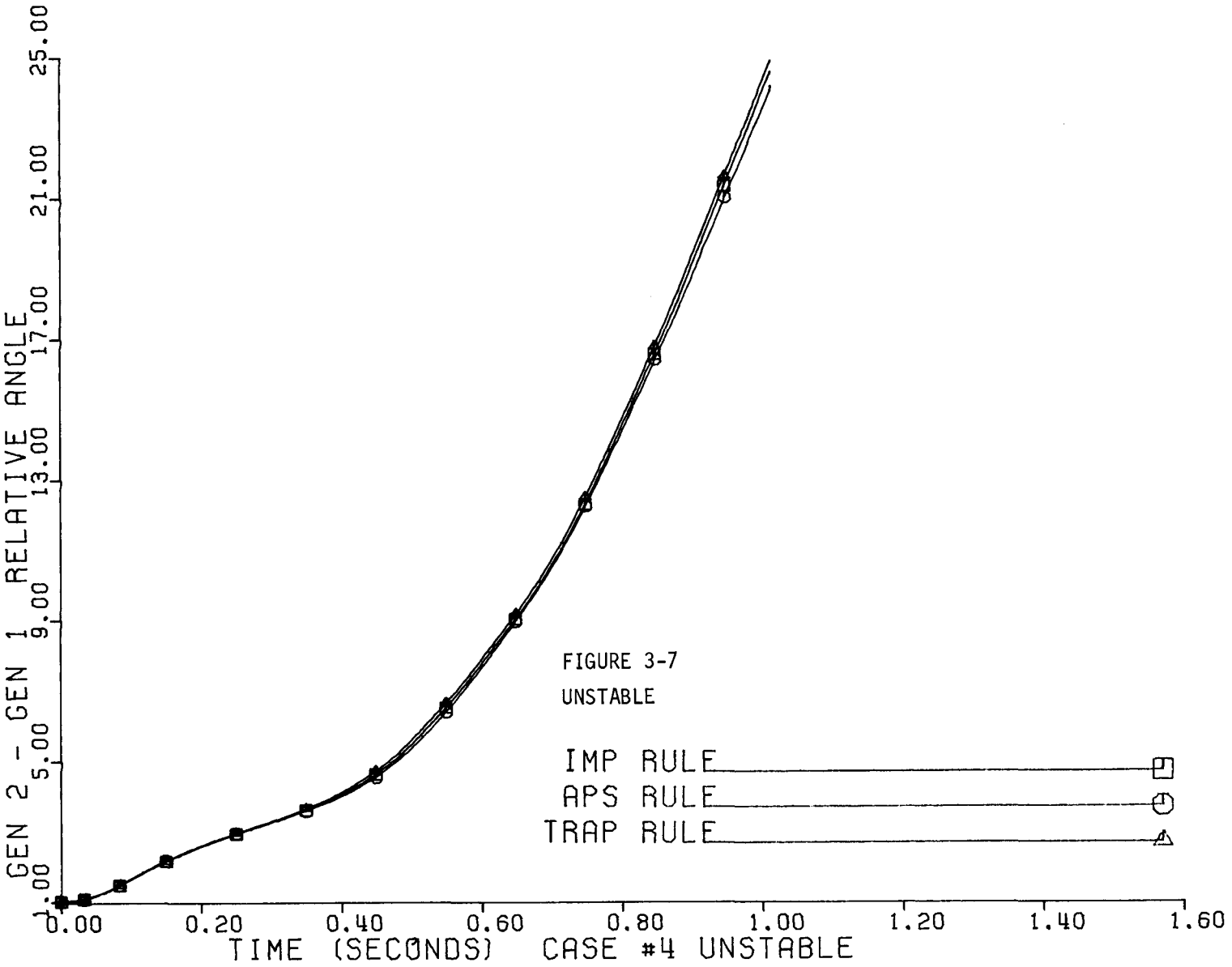


FIGURE 3-7

UNSTABLE

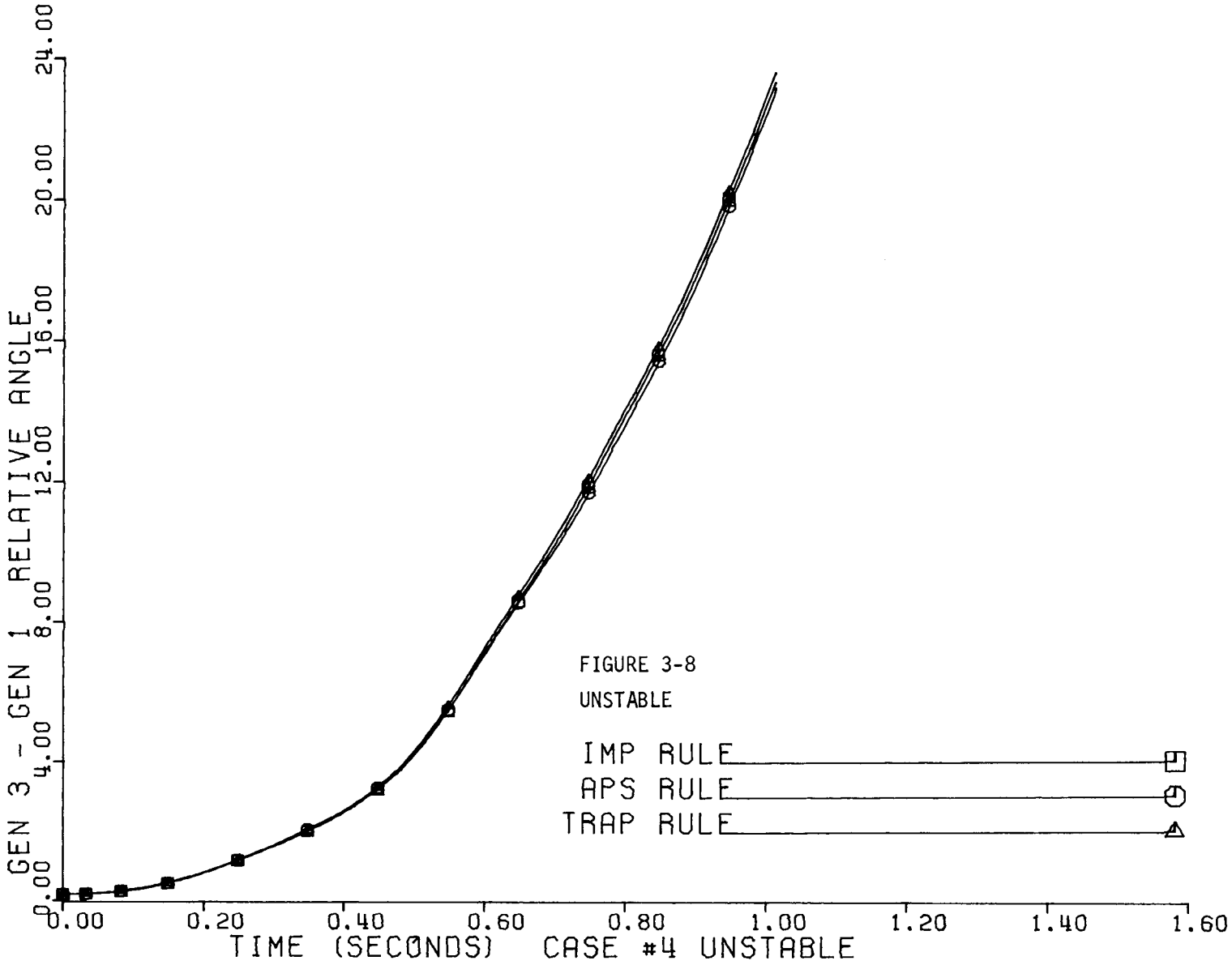


FIGURE 3-8

UNSTABLE

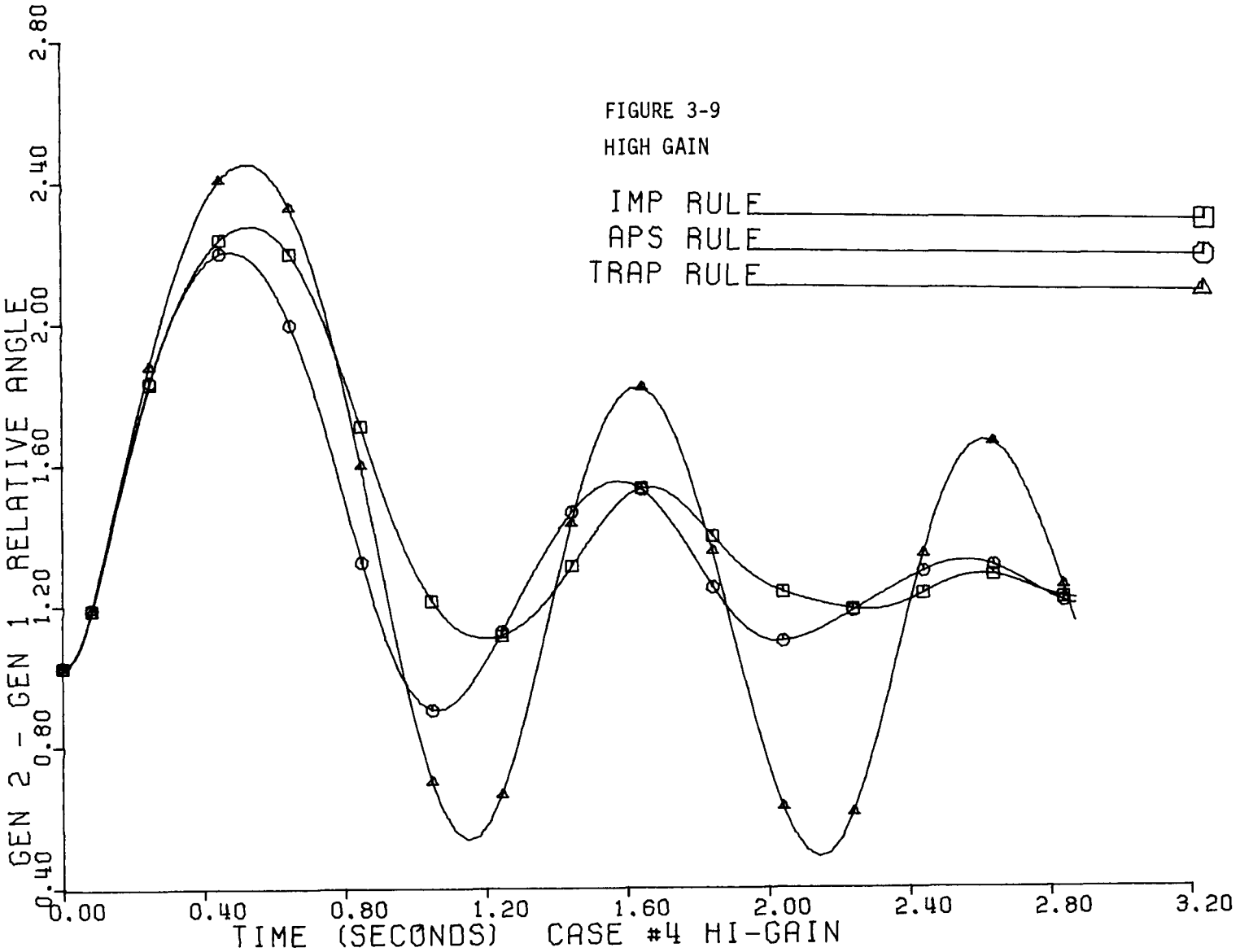
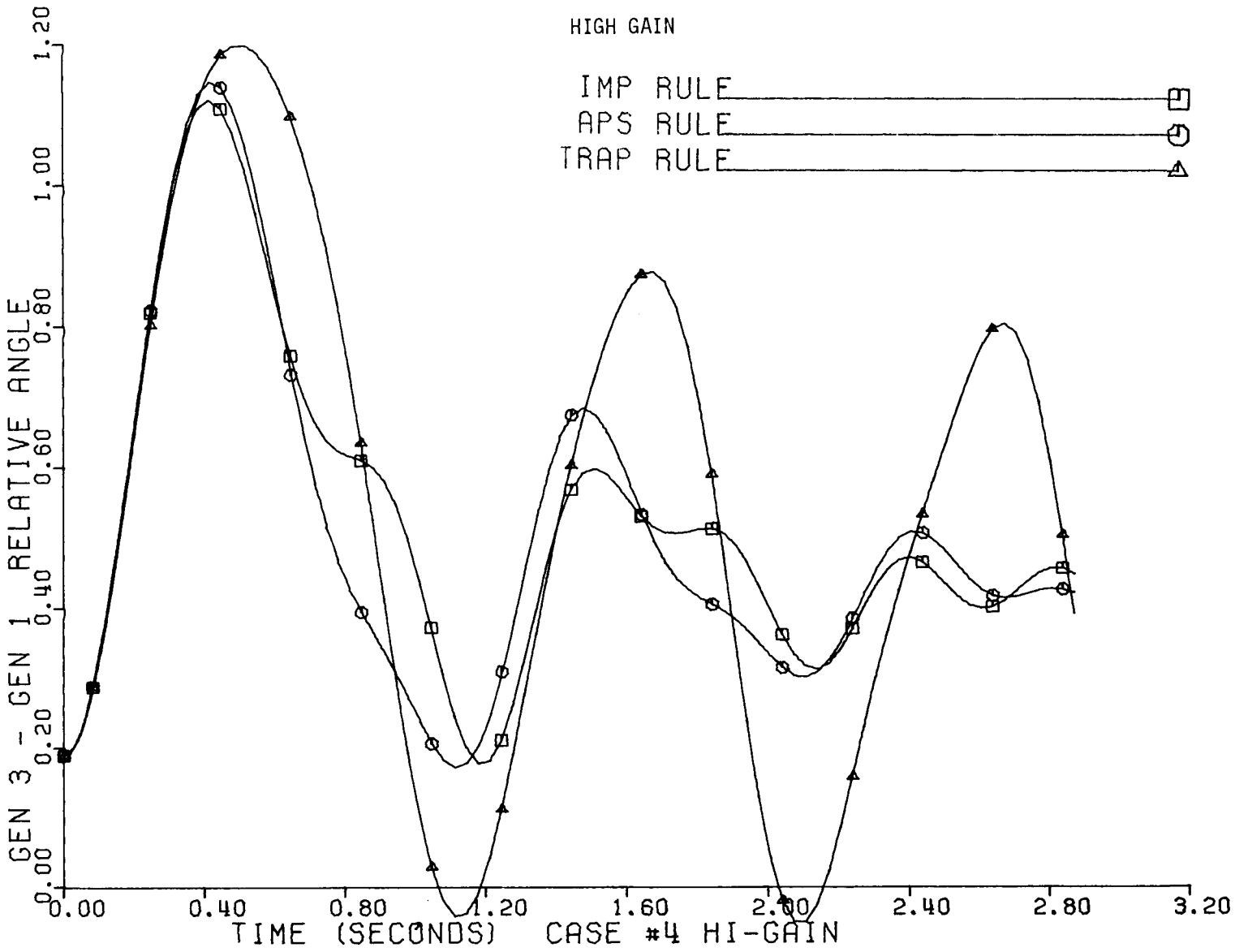


FIGURE 3-9

HIGH GAIN

FIGURE 3-10

HIGH GAIN



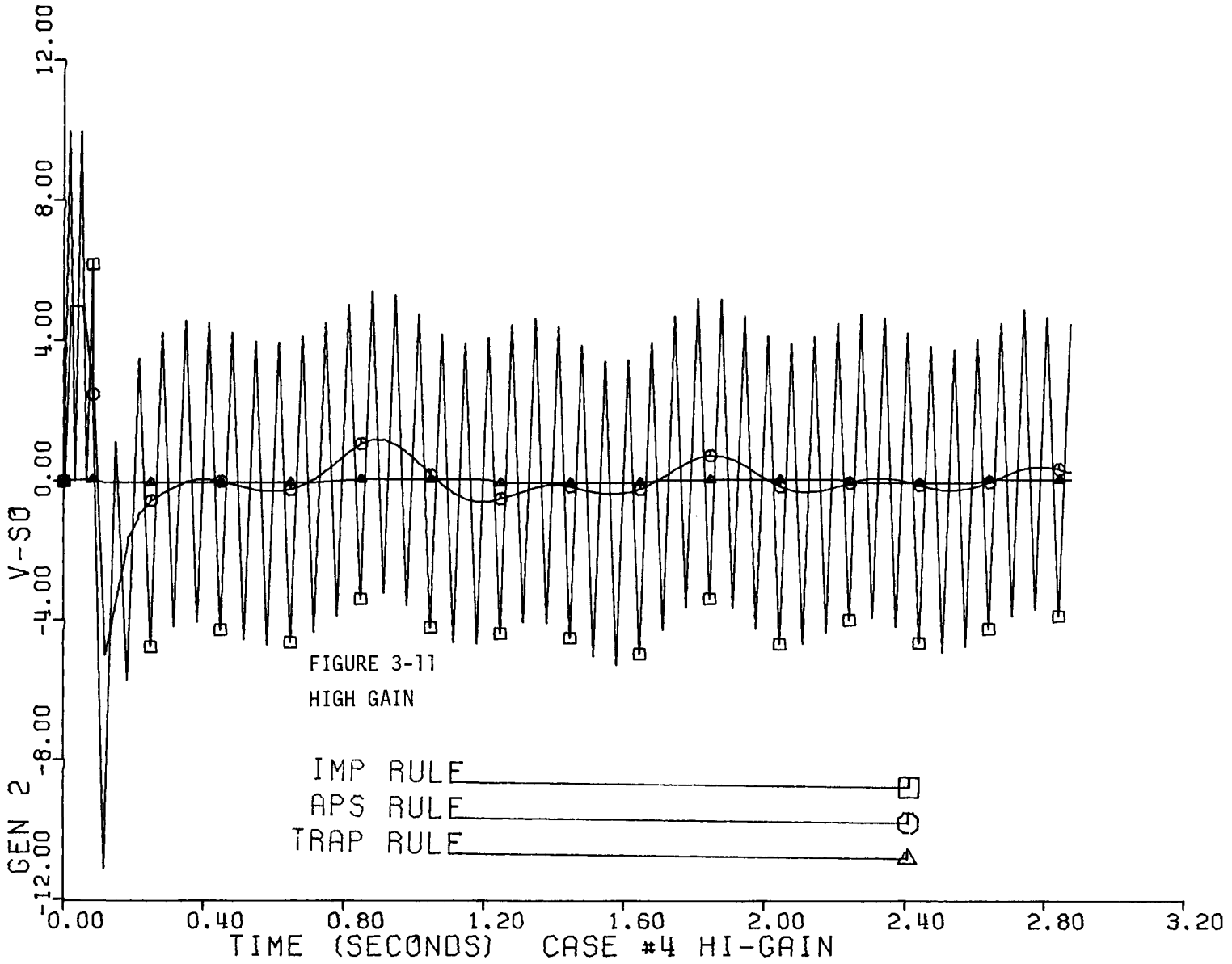


FIGURE 3-11
HIGH GAIN

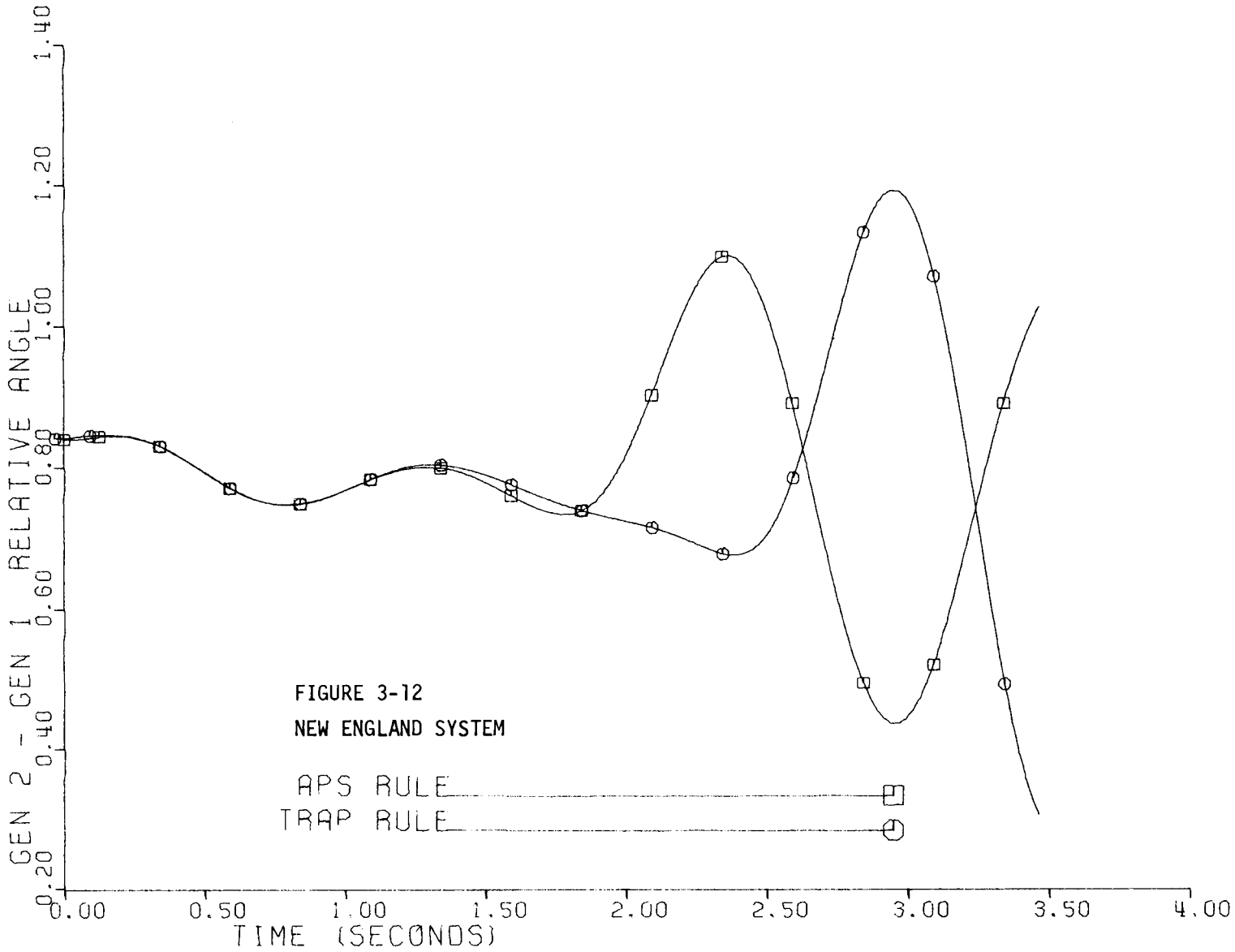


FIGURE 3-12
NEW ENGLAND SYSTEM

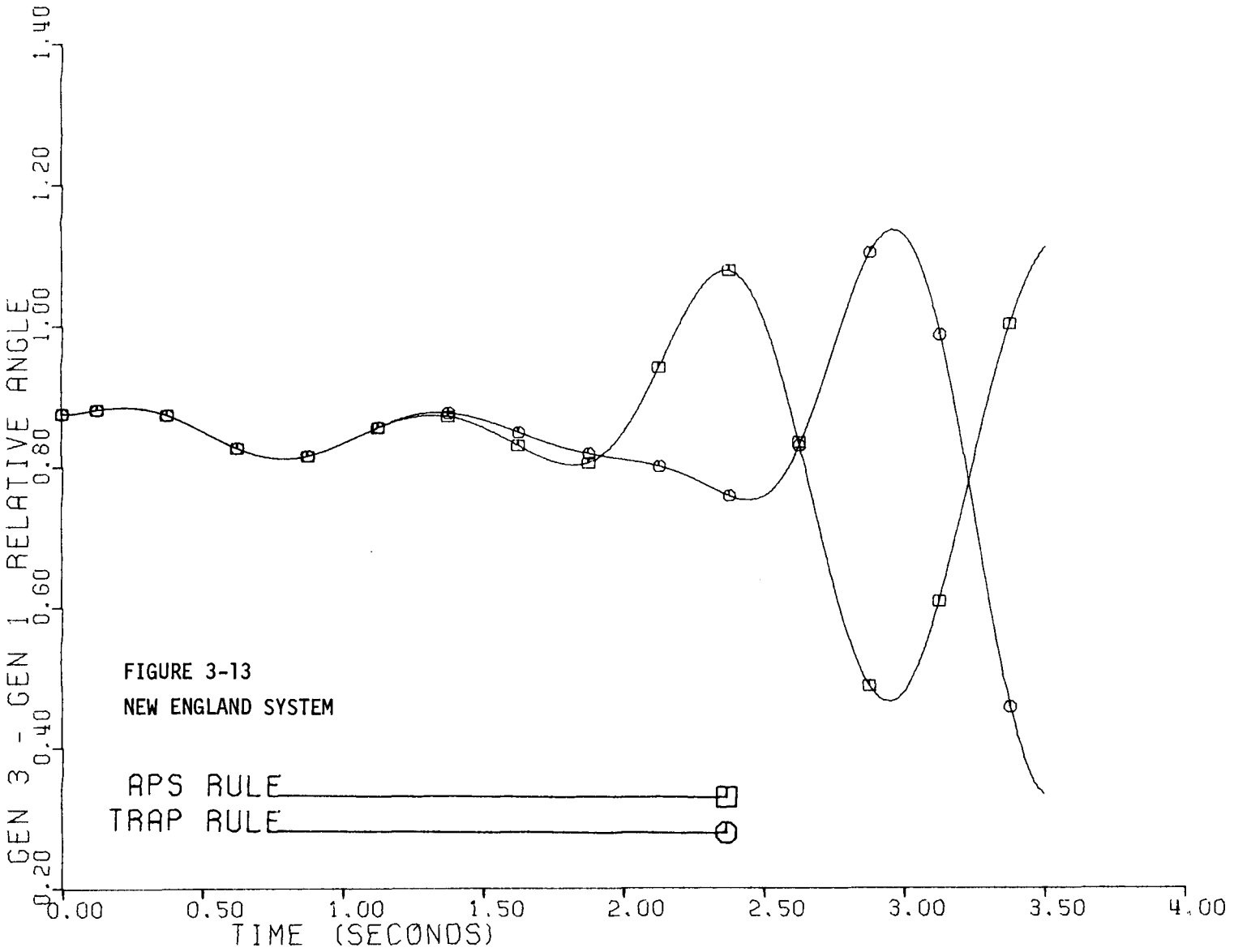
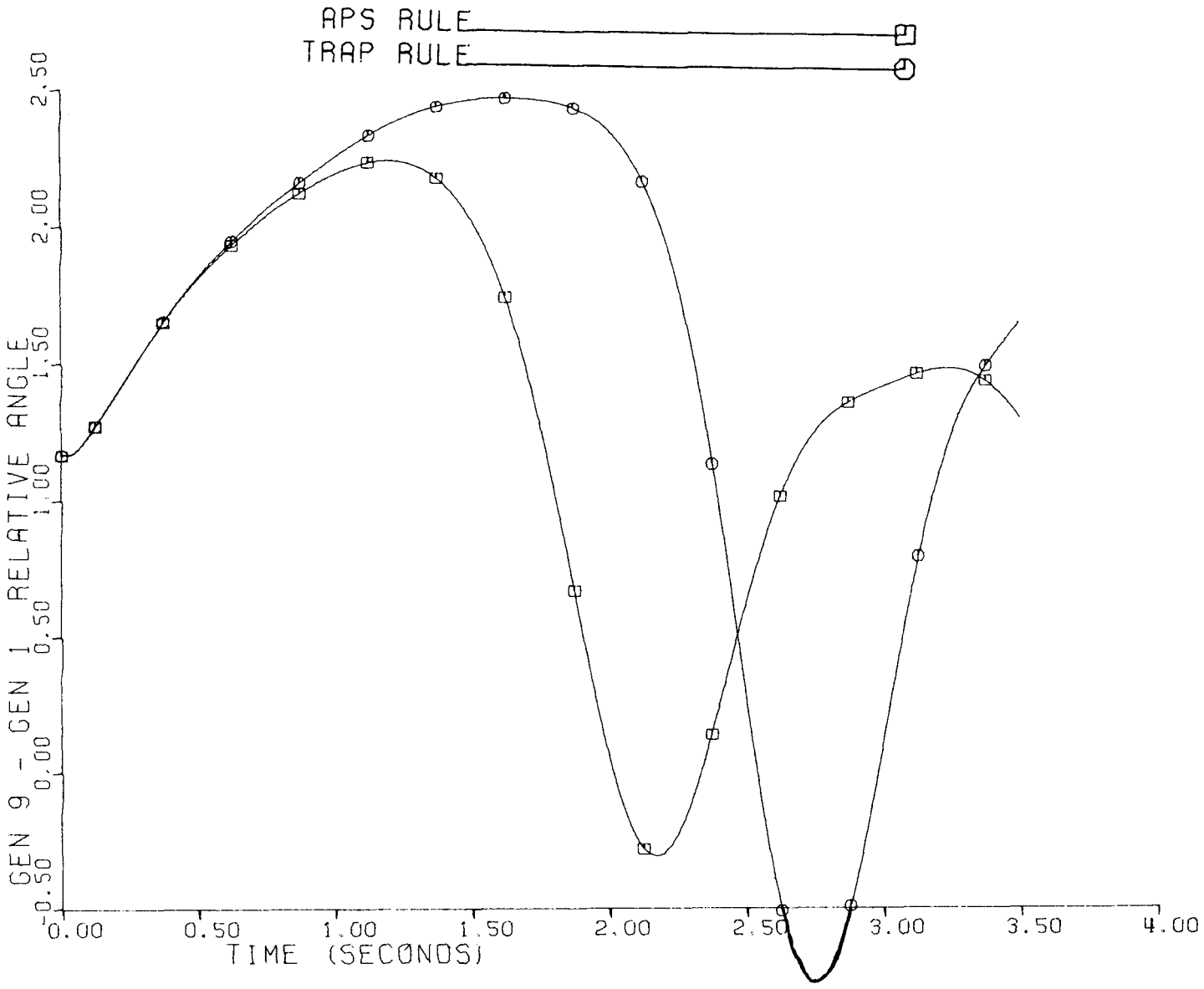


FIGURE 3-13
NEW ENGLAND SYSTEM

FIGURE 3-14

NEW ENGLAND SYSTEM



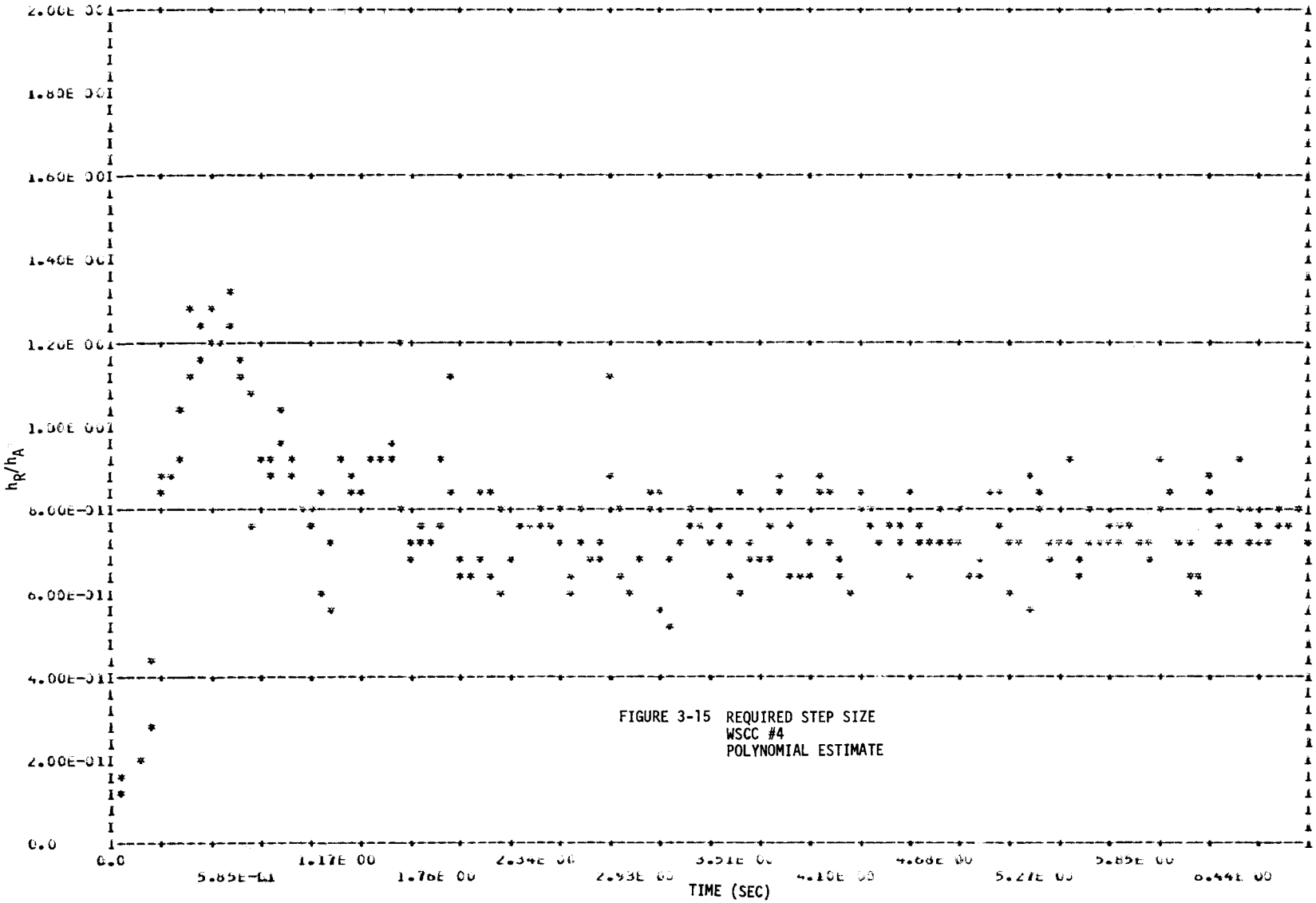


FIGURE 3-15 REQUIRED STEP SIZE
WSCC #4
POLYNOMIAL ESTIMATE

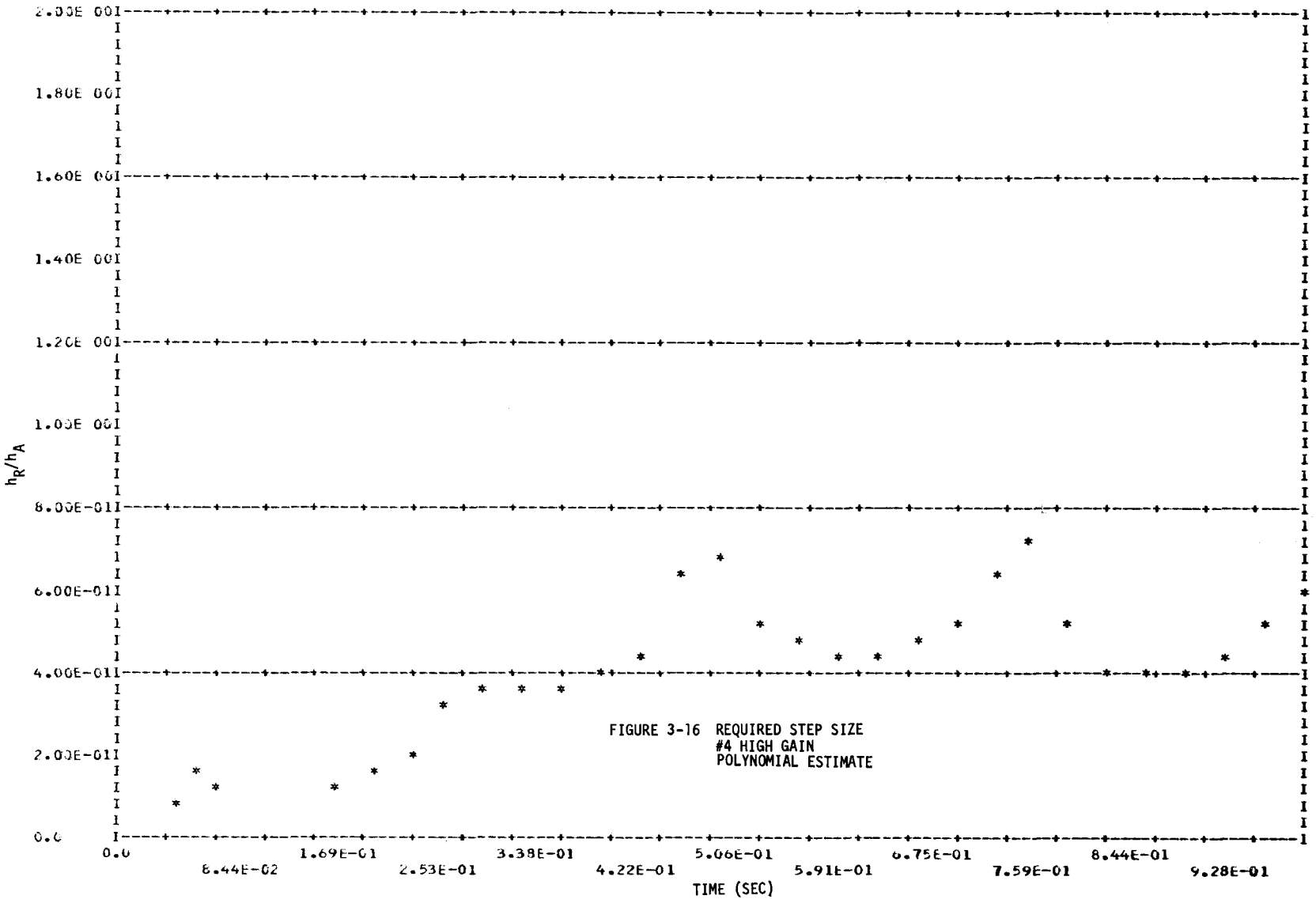


FIGURE 3-16 REQUIRED STEP SIZE
#4 HIGH GAIN
POLYNOMIAL ESTIMATE

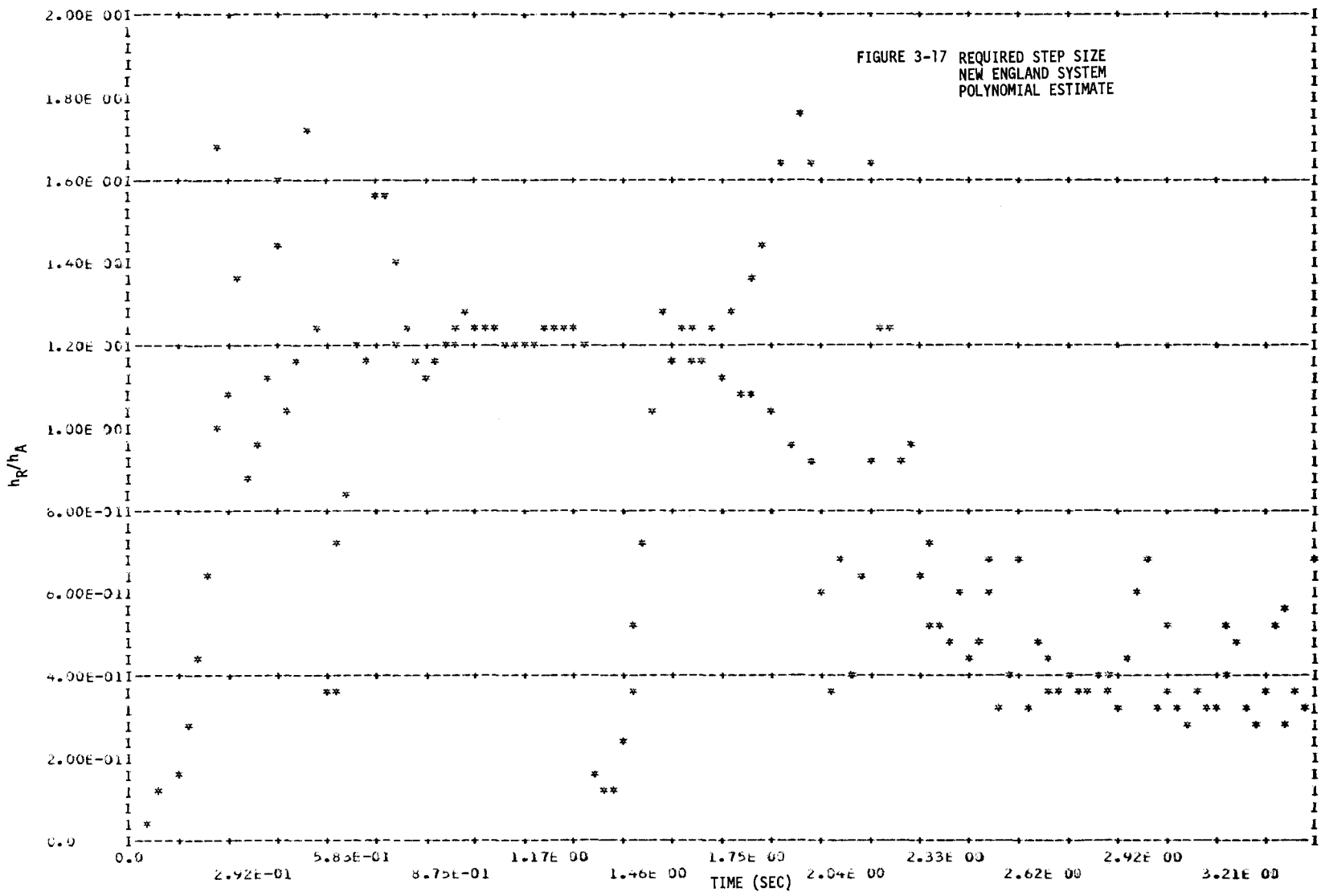


FIGURE 3-17 REQUIRED STEP SIZE
NEW ENGLAND SYSTEM
POLYNOMIAL ESTIMATE

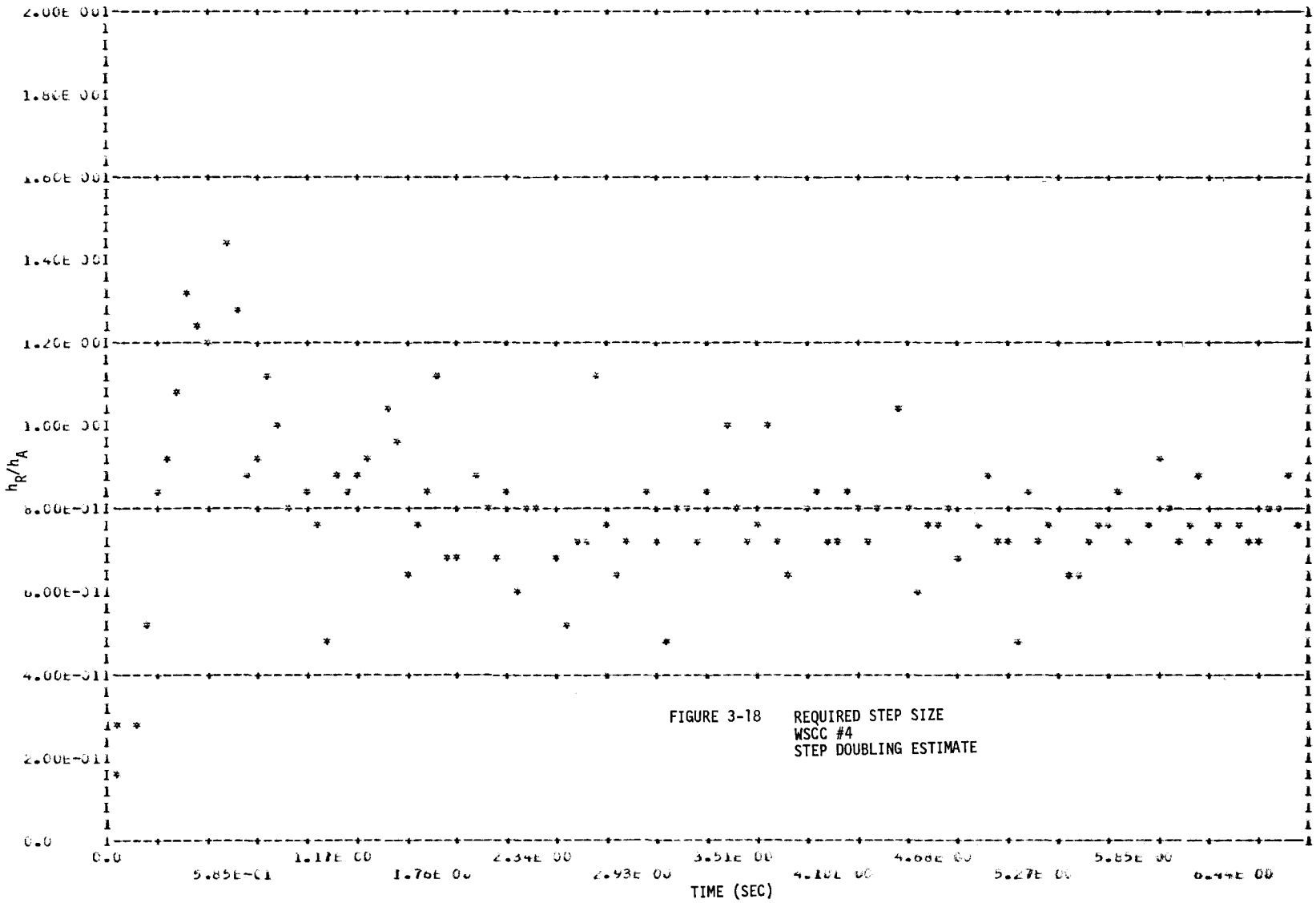


FIGURE 3-18 REQUIRED STEP SIZE
WSCC #4
STEP DOUBLING ESTIMATE

- unstable - case #4 with E_{MWS} = 225 for generator 2
- high gain - case #4 with PSS gain increased by a factor of 5 and limits removed

The swing curves for these five cases, for each of the three algorithms tested are shown in Figures 3-1 through 3-10. It can be seen from Figures 3-5 and 3-6 that the implicit midpoint rule produced stable results for the marginally unstable test problem. Figure 3-11 shows the stabilizer output for the high gain test case. Here the implicit midpoint rule gives very unsatisfactory results, even though the swing curves for this case are adequate. Because of these negative results from the implicit midpoint rule, no further testing was performed on this rule. Figures 3-12 through 3-14 show selected swing curves from the 39 bus New England test system. The behavior beyond two seconds requires further study.

Efficiency

Three test cases were used to compare the operational costs of the trapezoidal rule with the APS rule and implicit midpoint rule. The cases were

- stable - WSCC case #4
- high gain
- 39 bus New England test system

The results of these tests are summarized in Table 3-3. The first three rows represent actual data taken from the runs. The accuracy correction factor is obtained by estimating the step size that would be required to obtain an accuracy of .002 per unit per step on the average. The correction factor is (step size actually used)/(required step size). Thus the number of steps that would be needed if the required step size were used is just the actual number of steps taken multiplied by the correction factor. The CPU time needed to run the problem is the product of the CPU time per iteration, the number of iterations per step, and the number of steps required.

Because of efficiencies that can be achieved by clever implementations that can be as great as a factor of two, the results of these tests must be regarded as inconclusive. One general remark can be made. The single most critical factor seems to be the accuracy correction factor. That is, both methods can take steps in about the same time, so whichever method can take larger steps and still achieve a desired accuracy will be the superior performer.

The efficiency tests comparing the trapezoidal and APS rules were quite inconclusive. This implies that the operational costs of the two approaches are comparable. This agrees with a theoretical analysis of the steps that must be taken to implement the two algorithms. With the operational costs of the two approaches being comparable, development and maintenance costs will be the deciding factor as to which method is cheaper overall. As can be seen from examining the descriptions of the two algorithms given earlier in this section, the trapezoidal rule is far simpler than is the APS rule. Our own experience in programming the two rules in the diagnostic program confirms this. The generality of the trapezoidal rule is confirmed by its widespread use both within the power industry and in other engineering disciplines. The APS rule is not so general because of its requirements that each subsystem be linear.

For these reasons, the trapezoidal rule appears to be far more efficient in the development and maintenance areas than is the APS rule.

3.6 STEP SIZE SELECTION

The fact of life that necessitates that some attention be paid to the selection of step size is that too large a step size will result in poor accuracy, while too small a step size will result in excessive cost. The relationship between step size and accuracy is given by (see [1])

$$LTE \simeq Kx^{(p+1)}h^{p+1} \tag{3-17}$$

where p is the order of the method being used, h is the step size, K is a constant that depends on the problem and on the method being used, x is the solution, and LTE is the local truncation error, that is the error introduced in a single time step assuming the starting point for the step is completely correct. The actual error incurred by a method is called the global truncation error (GTE), and is the difference between the true solution and the computed solution. The GTE is a function of the error introduced in each step (LTE), the rate at which error introduced early in the problem grows or decays (problem stability), and whether the LTE from the various steps add or cancel. In general, GTE is proportional to LTE/h, with the proportionality constant being problem dependent. The GTE at any point in the solution represents an accumulation of errors throughout the simulation, an unacceptably large GTE means that step sizes were too large earlier in the simulation. For this reason, LTE is more appropriate to use as a measure of error for controlling step size.

The basic idea of a step size controlling algorithm is to use equation (3-17) to compute LTE, and then to adjust step size upward if the LTE is smaller than a target value, and to adjust step size downward if LTE is larger than the target. The target LTE must be selected so that GTE will be acceptable, and this may require some experimentation since the relationship between LTE and GTE is not simple.

Two problems arise in trying to use equation (3-17) to compute LTE. The first is that the coefficient K depends on the problem being solved and is not known a priori. The second is that the p+1 derivative of x must be computed.

The situation is simpler for some integration algorithms. For these algorithms the coefficient K is independent of the problem and depends only on the method. Of the methods considered in this report, only the trapezoidal rule and the BDF have this property. For these algorithms all that is required to use equation (3-17) to compute LTE, is to compute the p+1 derivative of x. This can be done by passing a polynomial of degree p+1 through p+2 points, and using the p+1 derivative of the polynomial as an estimate of $x^{(p+1)}$. This approach is called the polynomial method, and is used in many predictor-corrector algorithms. It should be reemphasized that theoretically this approach is not valid for the APS rule or the implicit midpoint rule.

An alternate approach can be used to compute LTE that avoids computation of derivatives of x, and knowledge of the coefficient K. Only the order p must be known. This approach involves taking two steps of length h and then reintegrating over the same interval with a step of length 2h. The results obtained by taking two single steps is given by

$$x_S = x_T + 2 \cdot LTE_S \tag{3-18}$$

where x_S is the result of taking two single steps, x_T is the true solution from the starting point (x_T is not known), and LTE_S is the single step LTE and is given by

$$LTE_S = K x^{(p+1)} h^{p+1}. \tag{3-19}$$

The results obtained from one double step of length 2h is given by

$$x_D = x_T + LTE_D \tag{3-20}$$

where x_D is the result of taking a double step, and LTE_D is the double step LTE and is given by

$$LTE_D = Kx^{(p+1)}(2h)^{p+1} \tag{3-21}$$

Subtracting (3-18) from (3-20) we get

$$x_D - x_S = LTE_D - 2LTE_S = Kx^{(p+1)}h^{p+1}(2^{p+1}-2) = (2^{p+1}-2)LTE_S$$

thus the single step LTE is given by

$$LTE_S = (x_D - x_S)/(2^{p+1}-2) \tag{3-22}$$

This approach to computing LTE is called step doubling.

The step doubling approach has the advantage of being universally applicable, but it has the disadvantage of requiring an extra integration step to compute the LTE. This disadvantage is compounded by the fact that the extra step uses a different step size than does the standard step. Thus the cost increase for computing LTE for every pair of single steps would be about 50%.

The cost/benefit trade-offs will now be examined for both the polynomial and step doubling methods.

The computer time involved in computing LTE by the polynomial method is quite minimal, however, there is a storage penalty associated with storing p past values of x. A more serious cost associated with any step size control method is that when step size changes, the Jacobian must be reevaluated and refactored. In Section 5 of this report the cost advantages of avoiding reevaluating and refactoring the Jacobian are discussed. These considerable advantages can be partly nullified by frequent step size adjustments. One benefit that can be expected from step size control is improved reliability because errors are detected and corrective action is taken. A second benefit is that accuracy and reliability can be achieved without the penalty of very small steps.

To evaluate the potential benefits, LTE was evaluated for several test cases (using the polynomial method). From equation (3-17) an estimate can be made of the step size required to achieve a target LTE. This estimate is given by

$$\frac{LTE_T}{LTE_A} = \left(\frac{h_R}{h_A}\right)^{p+1} \qquad (3\text{-}23)$$

where LTE_T is the target LTE, LTE_A is the actual LTE observed in the test case, h_A is the actual step size used in the test case, and h_R is the step size required to achieve the target LTE.

Figures 3-15 through 3-17 are plots of h_R/h_A for WSCC test case #4, the high gain version of case #4, and the 39 bus New England test system. The first two plots show trends typical of most cases examined. After a short initial period requiring a small step size, the required step size is reasonably constant. This indicates that user controlled step size, or automatic step size control that operates occasionally would be adequate for these cases. The third plot is quite different. It shows large swings, first up, and then down in required step size. In this run, a savings of up to 30% in the number of steps can be expected for an automatic step size control algorithm over a user controlled step size approach, if similar accuracies are to be achieved.

It appears more testing is required to determine if the polynomial approach to automatic step size control can be made cost effective.

Let us now turn our attention to the step doubling method. In addition to the costs described for the polynomial approach, the step doubling method incurs a rather high cost for computing LTE. This will amount to up to 50% of the cost of computing the solution. An additional benefit of step doubling is that the estimates of LTE are more reliable and steady, and will result in less spurious fluctuation in step size. This can be seen from Figure 3-18 which shows a plot of h_R/h_A for WSCC test case #4 (compare to Figure 3-15). For this case the consistency of the step doubling approach is nearly sufficient to overcome the 50% penalty.

The data studied indicates that step size adjustment on every step would not be cost effective, but occasional adjustment would be on some cases. Even though the step doubling approach is more expensive than the polynomial approach, its benefits may outweigh its costs when used on an occasional basis. Due to the potential benefits that step size control may provide, it should be implemented and tested on a wider range of test cases.

3.7 SUMMARY AND CONCLUSIONS

Five difference equations were considered in this research, namely:

1) the trapezoidal rule
2) the APS rule
3) the implicit midpoint rule
4) the backward differentiation formulae
5) a new method proposed by Gross and Bergen

Method 4 was rejected due to its poor stability properties (it is too stable) giving rise to unreliable results. Method 5 was rejected because of its heavy reliance on the backward differentiation formulae to solve most of the differential equations. Methods 1, 2 and 3 were implemented and tested for computational reliability and efficiency. Table 3-4 summarizes how each of these three methods was evaluated according to the criteria in Table 3-1. Method 3 was eliminated because of unreliable results on several marginal cases. The operational efficiency tests on methods 1 and 2 were inconclusive. The trapezoidal rule is considerably more efficient than the APS rule in the areas of development and maintenance.

In view of these findings, the trapezoidal rule is selected for recommendation as the best integration algorithm for dynamic stability analysis on the following basis:

1) It is symmetrically A-stable.
2) It is simple - leading to reduced development and maintenance costs.
3) It is generally applicable to all dynamic stability problems from electromagnetic transients to long term stability - without model alteration.
4) It is compatible with the polynomial method of computing local truncation error.

Furthermore, as long as the problem characteristics of the dynamic stability problem remain unchanged, it is unlikely that any new method will demonstrate enough superiority in operational efficiency to overcome the advantages in reliability and development and maintenance costs listed above.

Two algorithms (a polynomial approach, and step doubling) for estimating local truncation error were evaluated. These evaluations were made to study the feasibility of automatic step size control. Step size control appears to be cost

Table 3-4

EVALUATION OF THE THREE METHODS IMPLEMENTED

EVALUATION CRITERIA	TRAPEZOIDAL	APS	IMP
I Reliability	reliable in theory and tests	reliable in tests	reliable in theory
A. Stability	symmetrically A-stable	not A-stable (see Appendix A)	symmetrically A-stable
B. Marginal tests	good results	good results	bad results
II Efficiency			
A. Development and maintenance	very efficient	not very efficient	very efficient
1. Simplicity	simple	complex	simple
2. Generality	widely applicable and widely used	not widely applicable	?
B. Operational costs	costs roughly comparable for all 3 methods		
1. Solution time	comparable	comparable	comparable
2. Memory and I/O	less memory than APS	more memory than trapezoidal	less memory than trapezoidal
3. Information content	compatible with polynomial error analysis	not compatible with polynomial error analysis	not compatible with polynomial error analysis

effective on some of the test cases studied. In particular, the algorithms should be implemented in the diagnostic program and tested on some real stability problems. Additional testing will be done in the next phase on larger test problems greater than 100 buses.

3.8 REFERENCES

[1] C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Englewood Cliffs, N.J., Prentice-Hall, 1971.

[2] H. W. Dommel and N. Sato, "Fast Transient Stability Solutions", IEEE Trans. PAS, Vol. PAS-91, pp. 1643-1650, July/August 1972.

[3] H. W. Dommel, "Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks", IEEE Trans. PAS, Vol. PAS-88, pp. 388-399, April 1969.

[4] V. Converti, D. P. Gelopulos, M. Housley, G. Steinbrenner, "Long-Term Stability Solution of Interconnected Power Systems", Paper F75444-0 presented at IEEE PES Summer Meeting, San Francisco, July 1975.

[5] K. D. Downing, H. L. Fuller, P. M. Hirsch, J. A. Jordan, Jr., L. B. Lesem, W. G. Tuel, Final Report, "Analytical Techniques for Transient Stability", submitted to Bonneville Power Administration, Contract 14-03-1486N, February 1973.

[6] C. W. Gear, "The Simultaneous Numerical Solution of Differential Algebraic Systems", IEEE Trans. CT, 18, No. 1, pp. 89-95.

[7] G. Gross, A. R. Bergen, "A Class of New Multistep Integration Algorithms for the Computing of Power Systems Dynamical Response", IEEE Trans. PAS, Vol. PAS-96, pp. 293-306, January/February 1977.

Section 4

PARTITIONED EXPLICIT SOLUTION

In Section 2 it was mentioned that a partitioned explicit (PE) approach can be used to solve the mixed system of differential and algebraic equations that represent the transient stability problem. In fact, some of the most widely used transient stability programs have adopted this approach.

Because of their widespread usage throughout the electrical utility industry, PE techniques were implemented in a version of the diagnostic program. The goals of the study of PE integration were the answers to the following questions:

- Which of the available explicit methods is "best" for the power system problems?
- What is the most efficient way of implementing a PE integration algorithm?
- How does the PE compare with the SI approach?

The decision to use explicit integration methods has associated with it two major consequences. The first of these is the requirement that the problem formulation be explicit; that is, the problem must be posed in the standard form

$$\dot{X} = F(X) .$$

The key feature of this form is that $\dot{X}$ can be computed directly as a function of X without iteration. Further discussion of this topic will be found in part 4.1. The second major consequence of explicit integration is that these methods are more susceptible to the difficulties associated with stiff problems and numerical stability than the implicit methods discussed in Section 3. The discussion in part 4.2 centers about this area. With these results as background, the questions raised above are discussed in detail in part 4.3.

4.1 IMPLEMENTATION REQUIREMENTS OF EXPLICIT INTEGRATION

The fundamental requirement of explicit integration is the requirement that the differential equations be posed in either the standard form

$$\dot{X} = F(X) \qquad X(0) = X_o$$

or in the mixed standard form

$$\dot{X} = F(X,Y) \qquad X(0) = X_o$$

$$0 = G(X,Y)$$

where $G(X,Y) = 0$ can be solved for Y whenever X is given.

The second form includes the first as a special case, so it is the second one we will refer to as the mixed explicit form. This form is more applicable to the transient stability problem because it is a mixed differential algebraic system.

Within the framework of the standard formulation described in Section 2 there are at least two ways of achieving the mixed explicit form. For either approach, first we collect all the state variables from generator r into a generator state variable vector X_r; we also collect the non-state variables into the vector Y_r. Then we collect the state equations from generator r into the generator state equation

$$\dot{X}_r = F_r(X_r,Y_r). \tag{4-1}$$

All the non-state equations are collected into the generator non-state equation

$$0 = G_r(X_r,Y_r,V_{Br}). \tag{4-2}$$

V_{Br} is the bus voltage at generator r.

We write the network equations in the form

$$0 = H(X,Y,V) \tag{4-3}$$

where we have used the notations X and Y to describe the composite vectors

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ \\ X_{NGEN} \end{bmatrix}, \qquad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \\ \cdot \\ \\ Y_{NGEN} \end{bmatrix} \tag{4-4}$$

and V is the bus voltage vector.

Collecting terms we have obtained the equations

$$\left.\begin{aligned} \dot{X}_r &= F_r(X_r, Y_r) \\ 0 &= G_r(X_r, Y_r, V_{Br}) \end{aligned}\right\} \quad r = 1,2,\ldots,NGEN \tag{4-5}$$

$$0 = H(X,Y,V).$$

With the above notation for the system equations, it is easy to describe the two techniques for achieving the mixed explicit form. The first method will be called simultaneous explicit (SE) solution. We choose X, Y, F and G in the following fashion:

$$X = \begin{bmatrix} X_1 \\ \cdot \\ \cdot \\ \cdot \\ \\ X_{NGEN} \end{bmatrix}, \quad Y = \begin{bmatrix} Y \\ \\ V \end{bmatrix} = \begin{bmatrix} Y_1 \\ \cdot \\ \cdot \\ \cdot \\ \\ Y_{NGEN} \\ V \end{bmatrix} \tag{4-6}$$

$$F = \begin{bmatrix} F_1 \\ \cdot \\ \cdot \\ \cdot \\ \\ F_{NGEN} \end{bmatrix}, \qquad G = \begin{bmatrix} G_1 \\ \cdot \\ \cdot \\ \cdot \\ G_{NGEN} \\ H \end{bmatrix}.$$

This method was not actually implemented in the diagnostic program during the study primarily because it can be shown *a priori* that this method would be inferior to simultaneous implicit (SI) integration.

The second method for achieving a mixed explicit form, called partitioned explicit (PE) solution, involves the prediction and subsequent correction of the bus voltages on the generator buses. This method, which is widely used in the utility industry [2], has as its basis the realization that if the generator bus voltage V_{Br} is known, the G-equations (4-2) can be solved for Y_r provided X_r is given. Thus when V_{Br} is known, we have

$$\dot{X}_r = F_r(X_r, Y_r)$$
$$0 = G_r(X_r, Y_r, V_{Br}) \qquad (4\text{-}7)$$

is of the desired (mixed explicit) form. This formulation, which was implemented in the diagnostic program and tested, is discussed in detail later.

Implementation Requirements for SE Integration

Here we describe what is required to use the formulation (4-5) to solve the transient stability problem.

Any standard explicit integration package is designed to solve problems in standard form. Such a package may be used on a mixed explicit problem provided that a procedure is available that will compute the non-state variables Y and V, given the state variables X. When this has been accomplished, the first of equations (4-5) can then be used to evaluate the state rates $\dot{X}$. Since the evaluation of the state rates is a very straightforward process once the variables Y_r are known, the principal task is the solution of the implicit equations for Y and V:

$$0 = G_r(X_r, Y_r, V_{Br}) \qquad r=1,\ldots,NGEN$$
$$0 = H(X,Y,V).$$

The procedure used to accomplish this task would be Newton iteration (or some variation of it), preferably implemented with intelligent predictions on Y and V. It is interesting to note that this same task arises in two other situations:

1) Whenever the power transmission system undergoes a configuration change, the state variables are held constant while the remainder of the variables are adjusted to reflect the system change.

2) When the differential equations describing each generation system have been integrated separately (see next section), the predicted values of the generator bus voltages must be corrected to reflect the new estimates of the state variables X.

In fact, for reasons alluded to earlier, SE integration was not implemented in the diagnostic program. To see why it may always be expected to be inferior to use of SI integration, consider the following argument.

Suppose that the (SI) implicit trapezoidal rule is applied to the problem:

$$\dot{X} = F(X,Y) \qquad X(0) = X_o$$

$$0 = G(X,Y).$$

This implementation of the implicit trapezoidal rule requires the solution of the system of equations

$$K = \frac{h}{2} \left[F(X,Y) + F(X+K,Y+L)\right]$$

$$0 = G \left[X+K,Y+L\right],$$

where K and L are, respectively, the increments for X and Y over the integration step h. In order to achieve the same order (second) of accuracy with an explicit integration method, one must perform two evaluations of the function F for either a two-stage second order Runge-Kutta formula or a predictor-corrector (one for the predictor and one for the corrector). The cost of solving for Y and F given X is not sufficiently smaller than the cost of solving for X and Y simultaneous to justify the extra function evaluation (the ratio of costs is about .85, and would have to be .5).

In addition to the above considerations, it is also important to remark that the numerical stability properties of explicit integration procedures are quite inferior to the stability properties of the trapezoidal rule (see [6]). That numerical stability is an important consideration in the transient stability problem has been pointed out by Dommel and Sato [2] and Hirsch, Fuller and Lambie [3]. See also Appendix A for a discussion of this subject.

Implementation Requirements for a PE Solution

We now discuss in detail the implementation of the second method proposed in part 4.1 for achieving a mixed explicit form of the transient stability problem. The implementation we will discuss was realized in the diagnostic program and extensively tested to evaluate its performance. The most significant property of the algorithm to be discussed, in terms of its effect upon computational cost, was the need to perform reintegration of the generating systems because bus voltage predictions (extrapolations) were insufficiently accurate. In the course of the study it was found that systematic reintegration was required to prevent phase drift in the solution.

Perhaps the best way of identifying implementation requirements is to formally present the algorithm that was used and then investigate in detail what was needed to make the algorithm work. First, then, we present this algorithm.

Algorithm for Explicit Integration of Separate Generating Systems

Data. Values for X, Y and V at times t_n and perhaps at time t_{n-1}.

Goal. To obtain values for X, Y and V at time $t_{n+1} = t_n + h$. The process of obtaining these values is called "taking an algebraic step"; h is called the algebraic step size.

Procedure.

1. Given $V^{(n)} = V(t_n)$ and perhaps $V^{(n-1)} = V(t_{n-1})$, extrapolate to obtain an estimate for V at time t_{n+1}; denote the result $\tilde{V}^{(n+1)}$. It will now be possible to estimate V at any time t, $t_n \leq t \leq t_{n+1}$, by some interpolation process. Denote the function of V obtained by this process as $W(t; V^{(n)}, \tilde{V}^{(n+1)})$.

2. Integrate the mixed explicit differential systems

$$\dot{X}_r = F_r(X_r, Y_r) \qquad X_r(t_n) = X_r^{(n)} \tag{4-8}$$

$$0 = G_r(X_r, Y_r), \qquad W_{Br}(t; V^{(n)}, V^{(n+1)}) \tag{4-9}$$

from t_n to t_{n+1}, for each of the NGEN generating systems. If a standard integration program is used to do this, that program will, from time

to time, provide the user with a vector $\bar{X}_r$ and a time $\bar{t}$ and demand that $\dot{\bar{X}}_r$ be provided in return. Thus, the following sub-algorithm is required.

2.1 Given $\bar{t}$, estimate the bus voltage at generator bus B(r) by evaluating $\bar{V}_{Br} = W_{Br}(\bar{t};V^{(n)},\tilde{V}^{(n+1)})$.

2.2 Given $\bar{X}_r$ and $\bar{V}_{Br}$ solve equation (4-9) for $\dot{\bar{Y}}_r$.

2.3 Evaluate $\dot{\bar{X}}_r = F_r(\bar{X}_r,\bar{Y}_r)$ using (4-8).

3. When all of the generating systems have been integrated from t_n to t_{n+1}, estimates of X_r at time t_{n+1} will be available. Thus, it is possible to solve the problem

$$0 = G_r(X_r,Y_r,V_{Br}) \qquad r = 1,\ldots,NGEN \tag{4-10}$$

$$0 = H(X,Y,V)$$

X_r, r = 1,...,NGEN given

to obtain a new estimate for V at time t_{n+1}. Denote the value of V so obtained as $\hat{V}^{(n+1)}$. This step of the algorithm is simply a correction of the original estimate $\tilde{V}^{(n+1)}$.

4. At this point $\hat{V}^{(n+1)}$ may be compared to $\tilde{V}^{(n+1)}$. If these values are sufficiently close, the algebraic step is finished. If excessive error is observed, the assignment

$$\tilde{V}^{(n+1)} \longleftarrow \hat{V}^{(n+1)}$$

is performed and the algorithm is repeated starting with step 2.

With the above algorithm formally stated, we now describe the specific requirements needed for its implementation in the diagnostic program.

1. Prediction of bus voltages. Two methods for the extrapolation of bus voltages were implemented. The first of these, the obvious extrapolation of past data can be written as

$$\tilde{V}^{(n+1)} = V^{(n)} + \frac{t_{n+1}-t_n}{t_n-t_{n-1}}\left(V^{(n)}-V^{(n-1)}\right). \tag{4-11}$$

This method was used to predict V at time t_{n+1} on all steps except those that immediately follow a system change. On these steps, past trends cannot be regarded as reliable. Here, then, the prediction

$$\tilde{V}^{(n+1)} = V^{(n)} \tag{4-12}$$

is used.

The second extrapolation method used involved linear extrapolation of the logarithm of the bus voltage. The motivation for this algorithm stems from two observations:

a) The phase of a bus voltage is much more likely to vary linearly over a short period of the simulation than are its real and imaginary parts.

b) The magnitude of the bus voltage is likely to be very nearly constant over a short period of time.

The second extrapolation method, then, is specified by the equation

$$\ell n(V_k^{(n+1)}) = \ell n(V_k^{(n)}) + \frac{t_{n+1}-t_n}{t_n-t_{n-1}} (\ell n(V_k^{(n)})-\ell n(V_k^{(n-1)})) \tag{4-13}$$

$$k = 1,2,\ldots,NBUS.$$

2. The integration of the system (4-8), (4-9) was accomplished by invoking a standard routine for the solution of ordinary differential equations. (In the diagnostic program, two routines were made available; one, a classical fourth order Runge-Kutta integrator; the second a variable step, variable order Adams-Bashforth-Moulton code.) Because a standard routine assumes that the problem posed to it has the standard form, the sub-algorithm described in step 2 had to be implemented.

   2.1 For generating system r, the bus voltage at its corresponding bus B(r) was computed by interpolation between $V_{Br}^{(n)}$ and $\tilde{V}_{Br}^{(n+1)}$. The interpolation was either linear or logarithmic in accordance with the type of extrapolation used in part 1 of the algorithm.

   2.2 Given $\overline{X}_r$ and $\overline{V}_{Br}$ equation (4-9) was solved for $\overline{Y}_r$. This involved the solution of the system of nonlinear algebraic equations by Newton iteration. The Newton iteration procedure

used here is quite similar to that used elsewhere in the diagnostic program for both the SI and PE approaches. It is pertinent to remark that most transient stability programs do not use Newton iteration to obtain the state rates $\dot{X}_r$ when the states X_r and bus voltage V_{Br} are given. Rather, informed knowledge of the model is used to obtain direct solutions for $\dot{X}_r$ in these circumstances. Although this method is undeniably computationally efficient (in the tests run, the Newton iteration in part 2.2 of our algorithm accounted for about 2/3 of the run time), it was not implemented in the diagnostic program because of the excessive amount of programming effort it would have entailed. Nevertheless, in the data that will later be presented, estimates of the cost of using direct solution will be given.

2.3 The state rates $\dot{\bar{X}}_r$ are computed by evaluating the functions $F_r(\bar{X}_r, \bar{Y}_r)$. The subroutines for the generator models are used to perform this evaluation.

3. The Newton iterations used to solve for $\hat{Y}$ and $\hat{V}$ are quite similar to the procedure used in step 2.2, and much of the coding can be used in both steps. However, it is important to remark that the direct solution code that one should use in step 2.2 would not be useful in this step of the algorithm. This observation stems from the fact that in this stage of the algorithm the bus voltages $\bar{V}$ are unknown; whereas in the iteration in 2.2 the bus voltages $\bar{V}$ are known.

4. No algorithms were developed to assess the accuracy of the prediction of bus voltages. Rather, a single reintegration was routinely performed to assure adequate accuracy in the solutions. The procedure of performing the assignment

$$\tilde{V}^{(n+1)} \longleftarrow \hat{V}^{(n+1)}$$

and repeating the algorithm from step 2 corresponds precisely to use of the Jacobi iteration method (fixed point iteration, functional iteration) to solve a system of equations of the form

$$z = \phi(z) .$$

Because this iteration procedure is quite often inefficient, it was deemed desirable to obtain convergence rate data for it. Some of this data will be presented later in the results section. Generally speaking, the Jacobi iteration procedure was found to be quite effective.

4.2 EXPLICIT INTEGRATION METHODS

In the discipline of numerical analysis known as integration theory*, there occur three broad classes of methods for the solution of the standard initial value problem

$$\dot{X} = F(X) \qquad X(0) = X_o. \tag{4-14}$$

These classes of methods are:

1) Runge-Kutta methods
2) Linear multistep methods
3) Extrapolation methods.

These three classes of methods have been listed in order of increasing continuity requirements. Thus, while the Runge-Kutta methods can quite adequately handle problems that have frequent discontinuities in the derivatives of the solution, linear multistep methods require that the solution trajectory be highly continuous for extended periods of time in order to perform effectively. Finally, extrapolation methods, if implemented in a global fashion, require a high degree of continuity.

Primarily because of their expense, extrapolation methods were not considered in this study. Of the other two classes of methods, Runge-Kutta and multistep, one member was chosen from each class. Before we discuss the criteria by which one chooses a method from one of these classes, we describe the relevant aspects of the environment in which these integrators are to be exercised.

Environmental Considerations

The transient stability problem possesses two important characteristics that must be carefully considered whenever one chooses an integration method. These are:

* An excellent introduction to integration theory can be found in chapters 7 and 8 of Dahlquist, Bjorck and Anderson [4], while Henrici [5] and Gear [6] are excellent sources of more advanced information.

1) Relatively moderate accuracy is required. Typically, answers are considered quite adequate if their global accuracy is in the range 1% to 5%. These requirements are considerably less stringent than those that are typically used by much of the numerical analysis community that performs research on numerical integration methods. Most methods in the numerical analysis literature seem to tacitly assume that the user wants 4 or more significant digits in his numerical simulation.

2) Transient stability problems tend to have a substantial amount of "roughness", time points at which the solution possesses a low order of continuity due to the effect of limiters. The principal consequence of this effect is that the order of an integration step is fundamentally limited by the order of the roughness, no matter what the nominal order of the integration method may be. Thus, the classical Runge-Kutta method which has nominal order of 4 will perform no better than a second order integrator when tracking a solution that is only C^2 (twice continuously differentiable). This limitation is, in fact, quite consistent with the assumption made in deriving the classical Runge-Kutta rule, that the solution be C^4.

In addition to the above characteristics of the transient stability problem, we have the following characteristic of the PE algorithm that was implemented in the diagnostic program.

3) The PE algorithm requires prediction and subsequent correction of generator bus voltages, while some form of interpolation is used to estimate the bus voltages over the course of the algebraic step. The use of this procedure has two major consequences. First, because the interpolation processes used are fundamentally linear, the accuracy of an algebraic step is second order in the algebraic step size. (This fact follows from a careful and rather intricate analysis of the PE algorithm). As a consequence, second order integration of the machine equations (4-8) would appear to be most appropriate.

The only effect that would contradict this advice would be the fact that internal machine variables exhibit an intrinsically greater degree of variation than do bus voltages. Because it is felt that rapid variation of machine variables is a fairly

transitory effect (occurring on only a few algebraic steps), it is generally recommended that integration methods having order no higher than 3 be used. Because it is possible that rapid variations of machine variables may occur, error control is recommended since it is so inexpensive. The costs of several Runge-Kutta (R-K) methods and predictor corrector (P-C) methods in terms of evaluations of F(X) are given below, with and without error control.

Table 4-1

EVALUATIONS OF F(X) REQUIRED FOR VARIOUS INTEGRATION METHODS

	R-K 2ND ORDER	R-K 3RD ORDER	P-C ALL ORDERS
Without error control	2	3	2
Step doubling error control	$2^{.5}$	4	
Embedding formula* error control	3	5	
Predictor-corrector error control			2

*An embedding formula error control for a Runge-Kutta method is a Runge-Kutta method of order p which has the property that in the process of taking an integration step enough data is computed to evaluate a different Runge-Kutta rule of order p-1. The difference between the two rules provides an estimate of the error.

The second major consequence of the bus voltage prediction algorithm affects only multistep methods. Because the variation of bus voltages over the course of the simulation is polygonal, as shown in Figure 4-1,

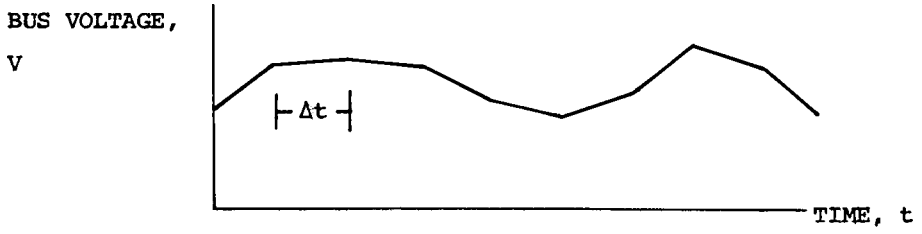


Figure 4-1. Variation of a Bus Voltage

the second derivatives of the computed trajectories will possess discontinuities at the boundaries of the algebraic steps. Thus, any multistep integrator must be restarted at first order at the beginning of each algebraic step. This feature detracts strongly from one great attraction of multistep methods, their ability to achieve high order efficiently. The costs associated with this repeated restarting constitute the fundamental reason for recommending Runge-Kutta methods.

Choice of Methods

Runge-Kutta Methods. In discussing the choice of Runge-Kutta integration method we assume that the algebraic step size has been chosen a priori and adequately reflects the variation in bus voltages that will occur. (If the bus voltages were to exhibit a sinusoidal variation over a period of 1 second, say, an algebraic step of 1/20 of a period or .05 seconds would keep the local truncation error in the bus voltages below .0025 for each algebraic step.) With the algebraic step chosen, we distinguish two separate cases:

1) Local integration error is not controlled.
2) Local integration error is controlled.

1. If the local integration error over the algebraic stop is not controlled, (not a recommended procedure), it must be assumed that no problems of numerical stability will occur. Thus, electrical devices that exhibit stiffness such as exciters and stabilizers should not be part of the system. With this reservation, then, it is recommended that the following three stage third order explicit Runge-Kutta method be used (the Kutta method)

Recommended R-K Rule

$$K_1 = hF(X_n)$$
$$K_2 = hF(X_n + \tfrac{1}{2}K_1)$$
$$K_3 = hF(X_n + K_2)$$
$$X_{n+1} = X_n + \tfrac{1}{6}K_1 + \tfrac{2}{3}K_2 + \tfrac{1}{6}K_3 .$$

This method has an embedded second order rule that should be used to monitor the error, even though error is not controlled. For this purpose it is advised that the quantity

$$\varepsilon = \left|\left| -\tfrac{1}{3}K_1 + \tfrac{2}{3}K_2 - \tfrac{1}{3}K_3 \right|\right|_\infty$$

be computed and reported to the user. If this quantity should exceed .01, some effort should be made to investigate why.

2. If local integration error is to be controlled, the R-K method employed should be that one which requires the fewest function evaluations to integrate through an algebraic step. Named below are four methods, listed in the order of the number of function evaluations per algebraic step, that should be considered. Because they are listed in order of increasing computational cost, they should be tested in the order given. Among the first three rules, the first rule to achieve a single integration step on most algebraic steps should be used.

   A. Third order, three stage Runge-Kutta formula with embedded error control.
   B. Fourth order, five stage Runge-Kutta-Fehlberg formula with embedded error control.
   C. Fifth order, six stage Runge-Kutta-Fehlberg formula with embedded error control.
   D. Fourth order, three stage Kutta formula with step doubling error control (8 function evaluations for two steps).
   E. Fifth order, four stage classical Runge-Kutta formula with step doubling error control (11 function evaluations for two steps).

It is clear from the function counts why method A is preferable to B and B to C if only one integration step is needed. If more than one integration step is required, the machine variables would seem to be experiencing substantial variation, and the higher order integration methods are much more capable of handling this. In addition, method D or E can be more efficient than method C in that they require only 4 or 5.5 function evaluations per step, respectively, as compared with 6 for method C.

The above arguments have been made without any consideration of the problem of roughness. If roughness is substantial (if it occurs on more than 20% of the algebraic steps, say), the higher order methods should be removed so that we are left with A, B and D.

This concludes the discussion of the choice of a Runge-Kutta formula.

Predictor-Corrector Methods. When one considers the P-C formulas, everything is much simpler. Here the best explicit methods are those of the Adams family. Because these methods are typically implemented in a variable order fashion, there is no need to choose any particular Adams method, but rather, the computer code itself chooses the method automatically.

It might be suggested that some of the stiffly stable methods suggested by Gear [6] would be useful. It is the opinion of the author, however, that these methods would be much more appropriate for the simultaneous implicit integration approach of Section 3. In the first place, the effective implementation of Gear's methods requires the use of machine Jacobians. As noted in the discussion of part 2.2 of our explicit integration algorithm, use of Jacobians simply makes the explicit integration techniques under study far too expensive in comparison with implicit integration.

Thus among the multistep methods, the choice of the right method is clear, the only question that is likely to arise concerns the choice of which one, among the many excellent Adams codes, one should choose. For the purposes of this study, we have chosen the Adams code developed at Boeing Computer Services by Dr. Harold Shniad (NWVAR, see [7]). This code implements the Adams-Bashforth-Moulton formulae in a variable step, variable order integration package using divided differences to save the solution history.

4.3 CENTRAL QUESTIONS ANSWERED

Having described the implementation details associated with explicit integration, we are now in a position to state precisely the questions informally posed at the beginning of this section. As we state each of these questions, we will present our conclusions along with some of that data that led to these conclusions.

Which is More Efficient, Explicit or Implicit Integration?

The principal criterion used to answer this question was a comparison of timing data between the SI version of the diagnostic program using the trapezoidal rule and the PE version using one of the explicit integration methods available. In evaluating these data, it is important to realize both the conditions under which they were generated and the rationale for two substantial modifications of the raw data.

The most significant of the test conditions for the explicit version were the following:

1) Fully honest Newton iteration was systematically used. Fully honest Newton iteration was used in two different circumstances: one, whenever the whole network was solved at the end of an algebraic step and two, whenever the G-equations were solved to obtain Y_r. With respect to the solution of the

machine G-equations for Y_r, it is pertinent to point out that, in light of the results of Section 5, a modified ("dishonest") form of Newton's method would very likely achieve substantial cost savings without the need for special code required for direct solution.

2) The generator differential equations were systematically reintegrated. In terms of the algorithm described in Section 4.2, this implies the following standard procedure:

   a) Predict all bus voltage at time $t_{n+1} = t_n+h$.
   b) Integrate all generator differential equations from time t_n to t_{n+1}; obtain an estimate of X at time t_{n+1}.
   c) Solve the network for the bus voltages V and the internal non-states Y_r.
   d) Using the corrected bus voltages from part c), reintegrate all of the generator differential equations; obtain a better estimate for X at time t_{n+1}.
   e) Resolve the network for V and Y_r.

Systematic reintegration was implemented only after it was observed that a single integration produced excessive phase drift in both the bus voltages and the relative machine angles. In particular, on the test case WSCC #1, single integration produced phase drift of about 30 degrees in the bus voltages and about 5 degrees in the relative machine angles at the end of a 2 second simulation. It should be noted that reintegration is not the only way of controlling phase drift. One of the observations made from the local truncation error estimates and verified by direct experiment was that single integration used with a smaller algebraic step size (smaller by a factor of 1/2 was found to be quite adequate) was effective in producing good global accuracy.

The two modifications made to the raw data are associated with what are thought to be the best possible operating conditions for explicit integration. The first of these modifications adjusts the timing data to estimate what the timing would have been had direct solution for Y_r been implemented. This modification was made so that the timing data for explicit solution would be more relevant to the common practice in the electric power industry.

The second of the two modifications estimates the probable CPU time associated with the Runge-Kutta third order three stage embedding formula. (In what follows, we shall refer to this method as R-K 3E.) This method may be optimally expected to require only one integration step, entailing three function evaluations, for

each algebraic step. That it can reasonably be expected to accomplish this can be inferred from two facts:

1) The fourth order Runge-Kutta rule that was tested routinely achieved error estimates two orders of magnitude below the requested error tolerance of 10^{-3} and still managed to integrate across a full algebraic step most of the time.

2) There exists a fundamental consistency between the use of a second order R-K formula and the use of the partitioned solution algorithm, which is itself second order in the algebraic step size.

With the above remarks in mind, we now present our conclusions and the data used to compare SI integration with PE integration.

In Table 4-2 are presented solution times (CPU second on an IBM 370/168) for problems #1 and #4 of the WSCC test series. The solution times given are for 2 seconds (real time) simulations. As can be seen, even in the best of circumstances (when R-K 3E performs optimally), SI integration requires only 63% as much time to simulate WSCC #1 and 59% as much time to simulate WSCC #4.

In fact, in the tests performed, the network solution CPU time alone for the PE integration algorithm exceeded the total CPU time for SI integration. This was true even though the time per Newton iteration was somewhat less (by a factor of .85) and fewer Newton iterations were required in the PE integration case. The root of the problem lies in the fact that PE integration requires either (1) that the machine equations be integrated and the network be solved twice per algebraic step, or (2) that the algebraic step be approximately one half as large as in the SI integration case. These requirements stem from the need to control phase drift sufficiently to keep relative machine angles from becoming too inaccurate. If a network Newton iteration were 2/3 as expensive for the PE integration case, then network solution costs for PE and SI integration could become comparable if:

1) In the case of systematic reintegration, an average of only 3 Newton iterations per algebraic step were required; or

2) the algebraic step size for the method using only a single integration pass could be as large as 2/3 the SI integration step size.

In actuality, somewhat greater improvements would be required to actually make PE integration competitive, for the above "break even" analysis has completely ignored the costs associated with integrating the machines.

TABLE 4-2 COMPARATIVE TIMES FOR IMPLICIT TRAPEZOIDAL RULE INTEGRATION WITH VARIOUS EXPLICIT INTEGRATION METHODS

	Trapezoidal Rule Honest Newton Solution of Implicit equations		Data Gathered From 5th Order R-K Runs (DNRKVS)		
			Actual Times 5th Order R-K HN Solution of G-equations	Projected Times 5th Order R-K Direct Solution of G-equations	Projected Times R-K 3E Direct Solution of G-equations
WSCC #1 (2 sec.)	10.20	Network Solution -	11.80	11.80	11.80
		Fr-Evaluations - -	75.34	11.23	2.63
		Integration OVHD -	1.51	1.51	1.51
			88.65	24.54	15.94
WSCC #4 (2 sec.)	13.88		15.66	15.66	15.66
			173.91	22.08	4.38
			3.48	3.48	3.48
			193.05	41.22	23.52

		Data Gathered From Variable Order/Step Adams Runs (NWDVAR) (These runs used systematic reintegration)		
		Actual Times Adams Method HN Solution of G-equations	Projected Times Adams Method Direct Solution of G-equations	Projected Times R-K 3E Direct Solution of G-equations
WSCC #1 (2 sec.)	Network Solution -	10.57	10.57	10.57
	Fr-Evaluation - -	38.85	6.69	2.97
	Integration OVHD	2.94	2.94	2.94
		52.36	20.20	16.48
WSCC #4 (2 sec.)		16.12	16.12	16.12
		85.14	9.38	3.95
		4.50	4.50	4.50
		105.76	30.00	24.57

		Data Gathered From Variable Order/Step Adams Runs (NWDVAR) (These Runs used ½ nominal step size with a single integration)		
		Actual Times Adams Method HN Solution of G-equations	Projected Times Adams Method Direct Solution of G-equations	Projected Times R-K 3E Direct Solution of G-equations
WSCC #1 (2 sec.)	Network Solution -	11.15	11.15	11.15
	Fr-Evaluation - -	22.67	3.53	2.55
	Integration OVHD -	2.26	2.26	2.26
		36.08	16.94	15.96
WSCC #4 (2 sec.)		15.15	15.15	15.15
		49.70	5.86	3.88
		3.14	3.14	3.14
		67.99	24.15	22.17

Solution times are CPU seconds on an IBM 370/168

For explicit integration methods, network solution times are given on the first line, Fr-evaluation (i.e., G-solution) times are given on the second line, integration overhead on the third line, and totals are given on the fourth.

Abbreviations: HN, fully honest Newton's method; R-K, Runge-Kutta; OVHD, Overhead.

How do the Various PE Methods Compare?

Here again the basis for decision was the timing data gathered by the diagnostic program. As in the comparison of SI with PE integration, the raw data were modified to reflect projected CPU times associated with direct solution of the G-equations. Costs associated with three integration methods are presented.

1) Times for the fifth order Runge-Kutta method with step doubling error control are presented. These data were gathered from runs employing the BCS-developed routine DNRKVS (see [7]).

2) Times for the Adams predictor-corrector method are presented. These data were gathered from runs employing the BCS-developed routine NWDVAR (see [7]).

3) Times for the R-K 3E method are presented. These costs were projected from the results of the other runs, making the optimal assumption that R-K 3E could always integrate across a full algebraic step using a single integration step.

In addition, costs associated with direct solution of the G-equations were projected for the fifth order Runge-Kutta method, the Adams predictor-corrector method and the R-K 3E method.

Tests as described above were run on test problems WSCC #1 and WSCC #4. The results of these tests, presented in Table 4-3, indicate the following ranking of methods:

1) Runge-Kutta third order embedding formula (R-K 3E).
2) Adams predictor-corrector variable step, variable order method.
3) Fifth order classical Runge-Kutta formula with step doubling error control.

The data are quite unambiguous in rating the fifth order Runge-Kutta algorithm last.

The work saving to be expected from the R-K 3E method over the Adams P-C method is associated with two facts:

(i) The R-K 3E method is substantially more likely to integrate across the whole algebraic step in one integration step because the Adams P-C method must restart the integration with, at best, a second order method. In addition, the Adams P-C method requires one G-solution to start the

TABLE 4-3 COMPUTATIONAL COSTS FOR EXPLICIT INTEGRATION ALGORITHMS, THE COSTS ASSOCIATED WITH SOLVING THE G-EQUATIONS AND EVALUATING F_r

		WSCC #1 (2 SEC)		WSCC #4 (2 SEC)	
		Full Step Reintegra.	Half Step Single Integra.	Full Step Reintegra.	Half Step Single Integra.
[Network Solution Times] :		[11.80]	[11.15]	[16.12]	[15.15]
Solution and Integration Methods					
Direct Solution of G-equations	R-K 3E Direct	2.63	2.55	3.94	3.88
	Adams Direct	6.69	3.53	9.38	5.86
	R-K 5th Order Direct	11.23	-	22.08	-
Modified Newton Solution of G-equations	R-K 3E DHN	8.78	8.76	14.42	13.98
	Adams DHN	21.99	12.13	34.27	21.10
	R-K DHN	37.50	-	75.91	-
Full Newton Solution of G-equations	R-K 3E HN	17.64	16.38	35.82	32.93
	Adams HN	38.85	22.67	85.14	49.70
	R-K 5th Order HN	75.34	-	173.91	-

NOTE:
1. Times given are for solution of the G-equations in CPU seconds on an IBM 370/168.
2. Network solution times are given in brackets[], at the top of the column.
3. All times for direct solution and dishonest newton solution (DHN) are projected.
4. Times for the R-K 3E method are projected assuming one integration step per algebraic step.

integration as well as two G-solutions per integration step. Thus, for the Adams P-C method, the first integration step of an algebraic step requires as much work as an integration step using R-K 3E, yet achieves lower order.

(ii) The R-K 3E method, being a one-step method, is much better equipped to handle the roughness in the solution induced by limiting devices. (Such roughness, which occurs in problem WSCC #4, accounts for the fact that R-K 3E may be expected to perform substantially better).

In conclusion, it is recommended that if explicit integration is to be used, a second or third order one-step method such as R-K 3E be employed.

What is the Best Way to Implement a Partitioned Explicit Integration Algorithm?

The subquestions addressed here included the following:

1) In performing extrapolation and subsequent interpolation of bus voltages, is it better to use linear or linear logarithmic extrapolation?

2) Is reintegration required on every algebraic step? Is more than one reintegration ever required?

3) Which of the following three methods is best for obtaining generator state rates $\dot{X}_r$ given the state vector X_r?
   a) Full Newton iteration
   b) Modified Newton iteration
   c) Direct solution

4) Should one use integration error control over the course of an algebraic step?

In addition to the above questions one important issue that was not investigated was quadratic extrapolation and interpolation of bus voltages. We now describe the procedures used to answer the above list of questions.

1) In determining whether linear or linear logarithmic extrapolation is better, the accuracy associated with each of these methods of predicting bus voltages was computed and plotted. The plots produced in this fashion portray the number of significant digits in the bus voltage vector at two stages:
   a) After prediction, before the first integration
   b) After the first correction, following the first integration.

   By comparing corresponding plots for linear and linear logarithmic prediction, it was thus determined that linear logarithmic prediction of bus

voltages produced approximately .4 significant digits greater accuracy in the initial prediction of the bus voltages. In addition, toward the latter part of the simulation, as the system accelerates, the accuracy of logarithmic prediction is maintained while simple linear prediction tends to degrade fairly substantially. Somewhat paradoxically, any improvement in accuracy following the first correction (network solution), is hardly perceptible and often the simulation using linear prediction achieves greater accuracy. However, on those algebraic steps exhibiting the slowest convergence properties, logarithmic prediction produces significantly better results after the first correction (network solution). In addition, those simulations using logarithmic prediction of bus voltages required approximately 10% fewer Newton iterations in the network solution portion of the simulation process. Because of the above considerations and because logarithmic prediction of bus voltages is so inexpensive to implement, it is recommended that it be used.

2) In determining whether reintegration is required or not, simulations were performed at standard simulation step sizes using a) single integration and b) two integrations of the generator differential equations. The results of these tests indicated that using the standard algebraic step size, single integration would experience approximately an order of magnitude greater truncation error on each algebraic step. Thus, if single integration were to be used, an algebraic step approximately 1/2 as large is indicated.

To test this hypothesis, runs using 1/2 the nominal algebraic step size and a single integration were performed on the WSCC test problems #1 and #4 and the local solution accuracy was plotted. (It should be remarked that since the PE method is second order in algebraic step size, step doubling is just as effective a means of measuring local truncation error here as it was found to be in analyzing the results of the trapezoidal rule.)

The results of these tests indicate that with an algebraic step size 1/2 of the nominal value, single integration is approximately as accurate as the reintegration procedure using the nominal algebraic step size. (See Figure 4-2 for the comparison of the accuracy achieved by the two methods.)

In addition to the above results concerning local accuracy, the global accuracy of the different procedures was evaluated by comparing machine angles and slip rates at the end of 3 seconds of simulation time for problem WSCC #1 and after 2 seconds of simulation time for WSCC #4. This data, presented in

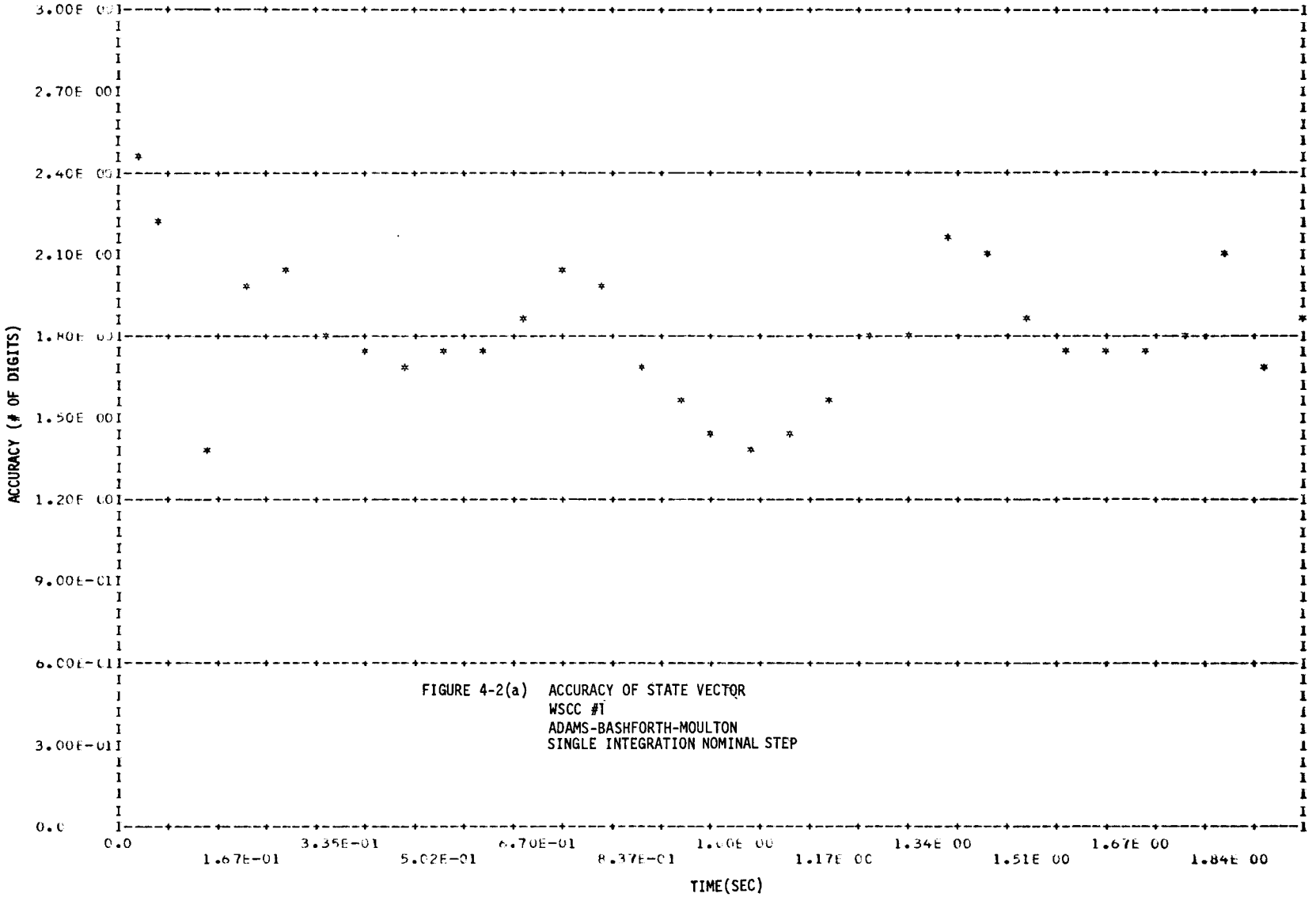


FIGURE 4-2(a) ACCURACY OF STATE VECTOR
WSCC #1
ADAMS-BASHFORTH-MOULTON
SINGLE INTEGRATION NOMINAL STEP

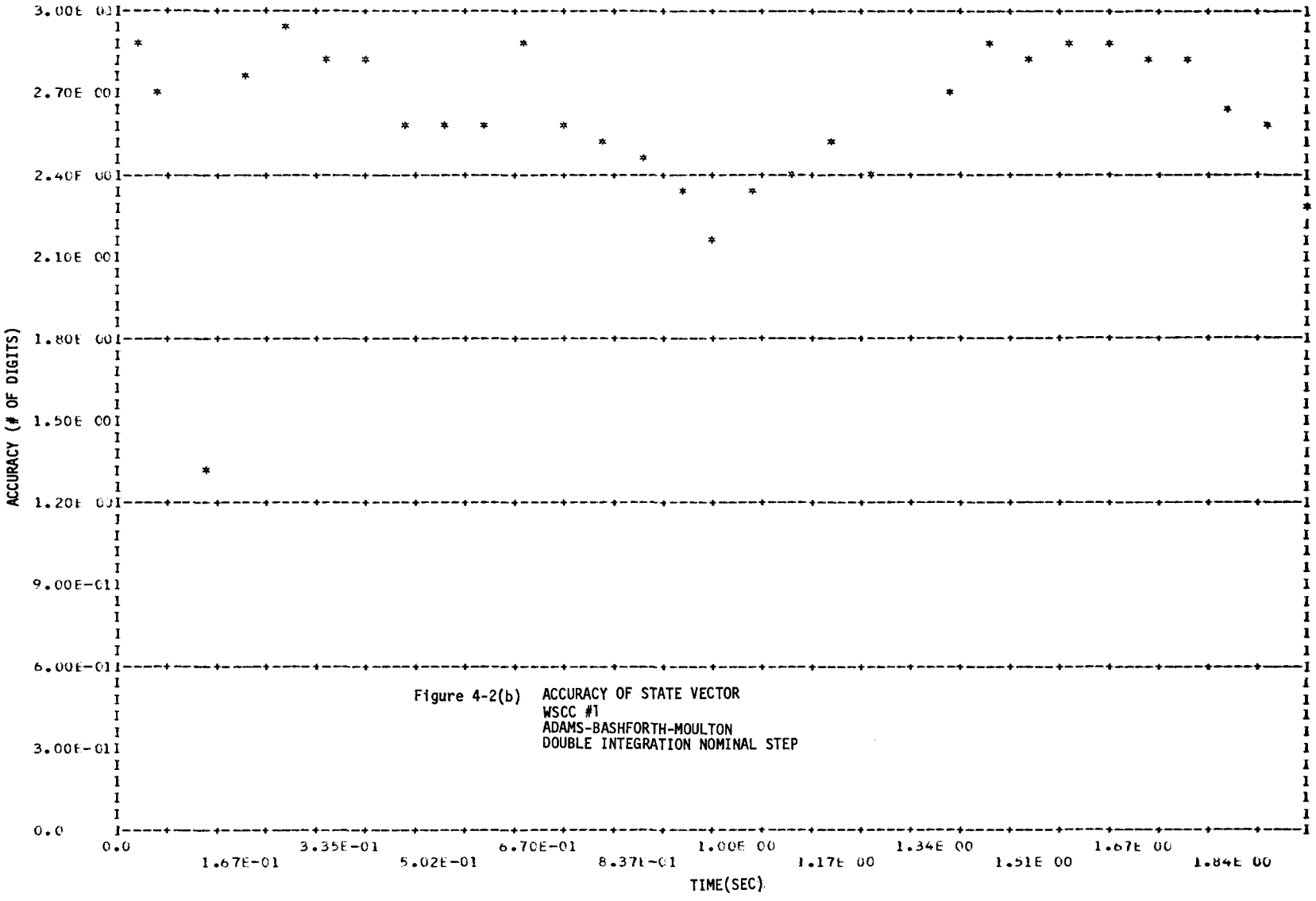


Figure 4-2(b) ACCURACY OF STATE VECTOR
WSCC #1
ADAMS-BASHFORTH-MOULTON
DOUBLE INTEGRATION NOMINAL STEP

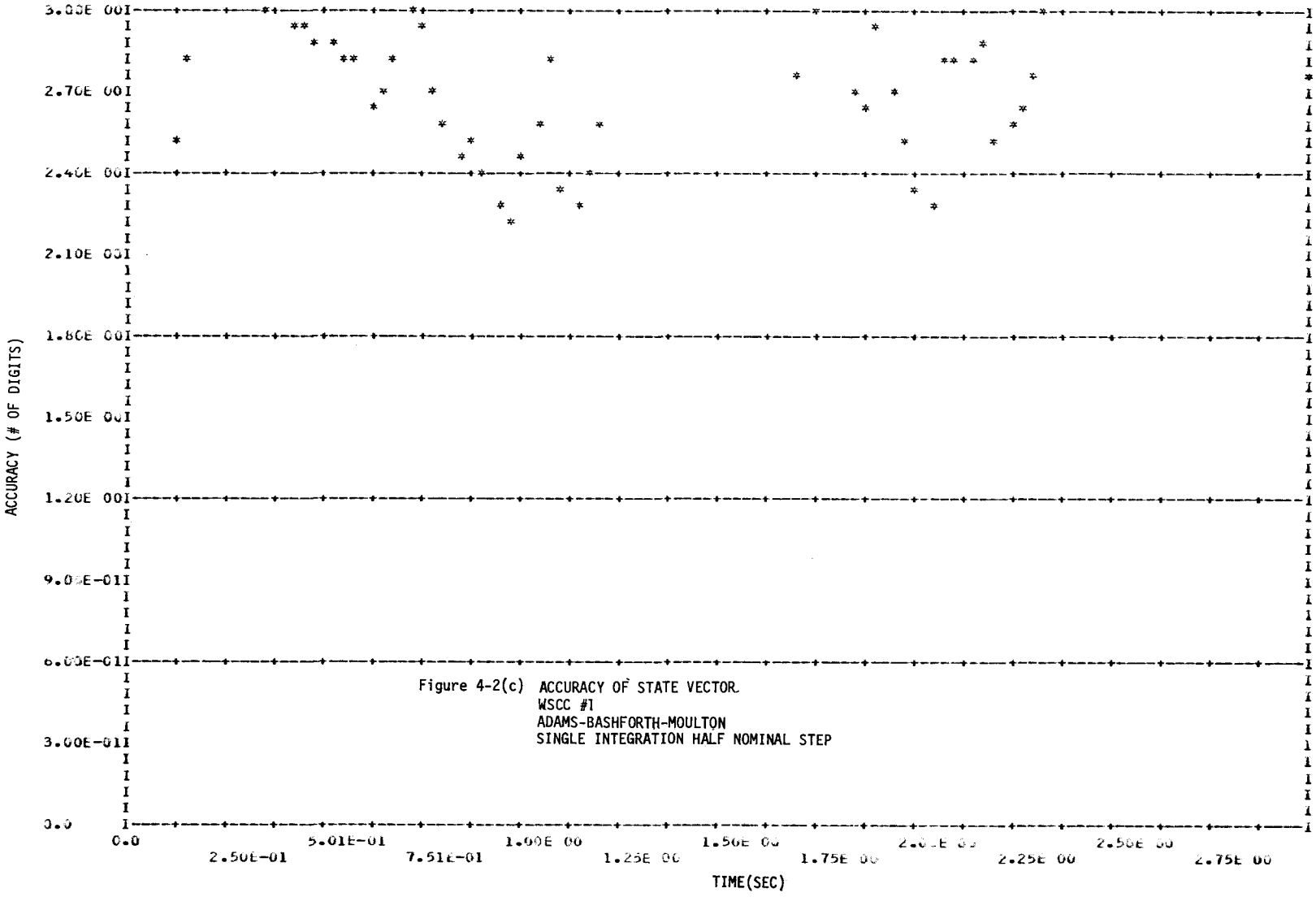


Figure 4-2(c) ACCURACY OF STATE VECTOR
WSCC #1
ADAMS-BASHFORTH-MOULTON
SINGLE INTEGRATION HALF NOMINAL STEP

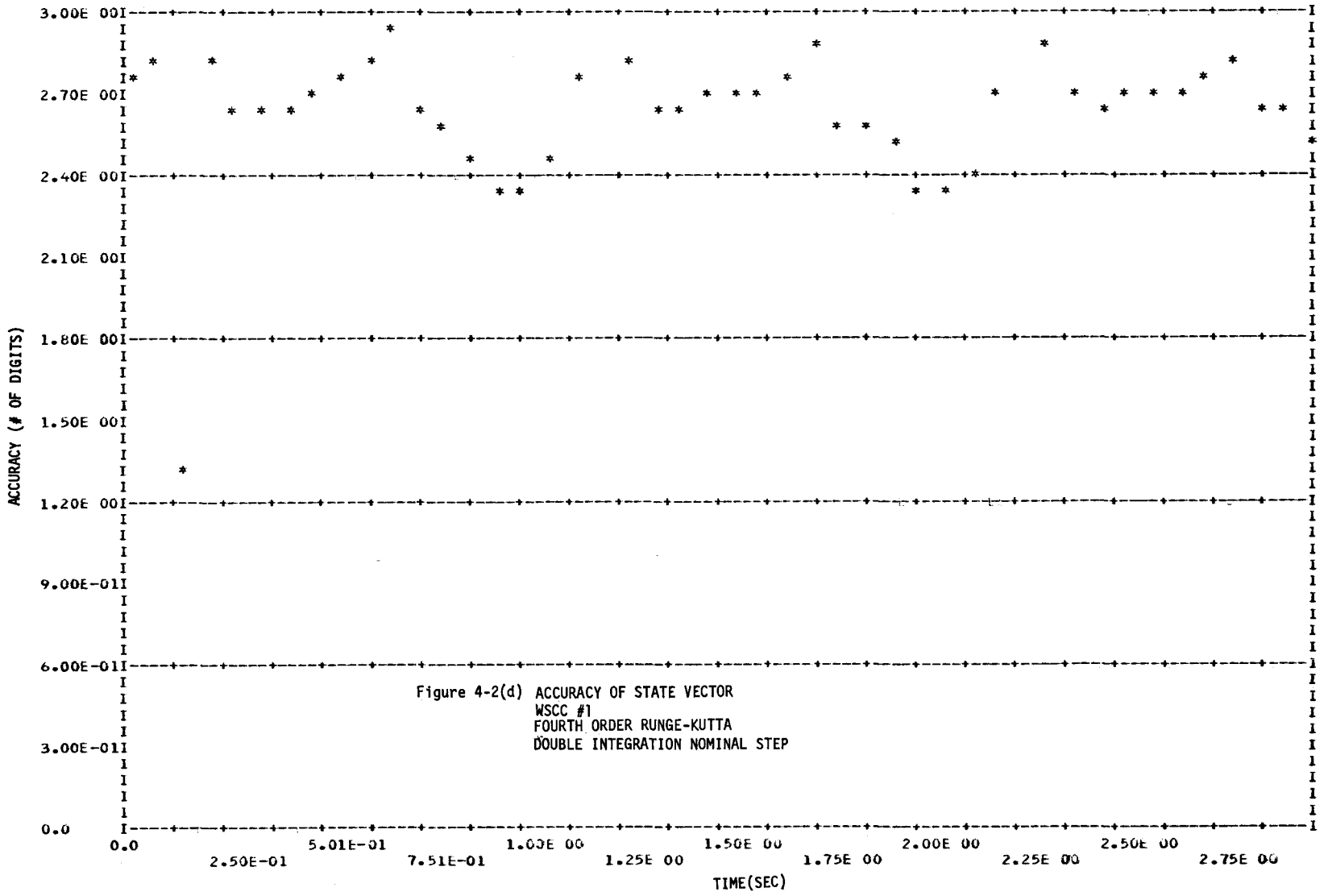


Figure 4-2(d) ACCURACY OF STATE VECTOR
WSCC #1
FOURTH ORDER RUNGE-KUTTA
DOUBLE INTEGRATION NOMINAL STEP

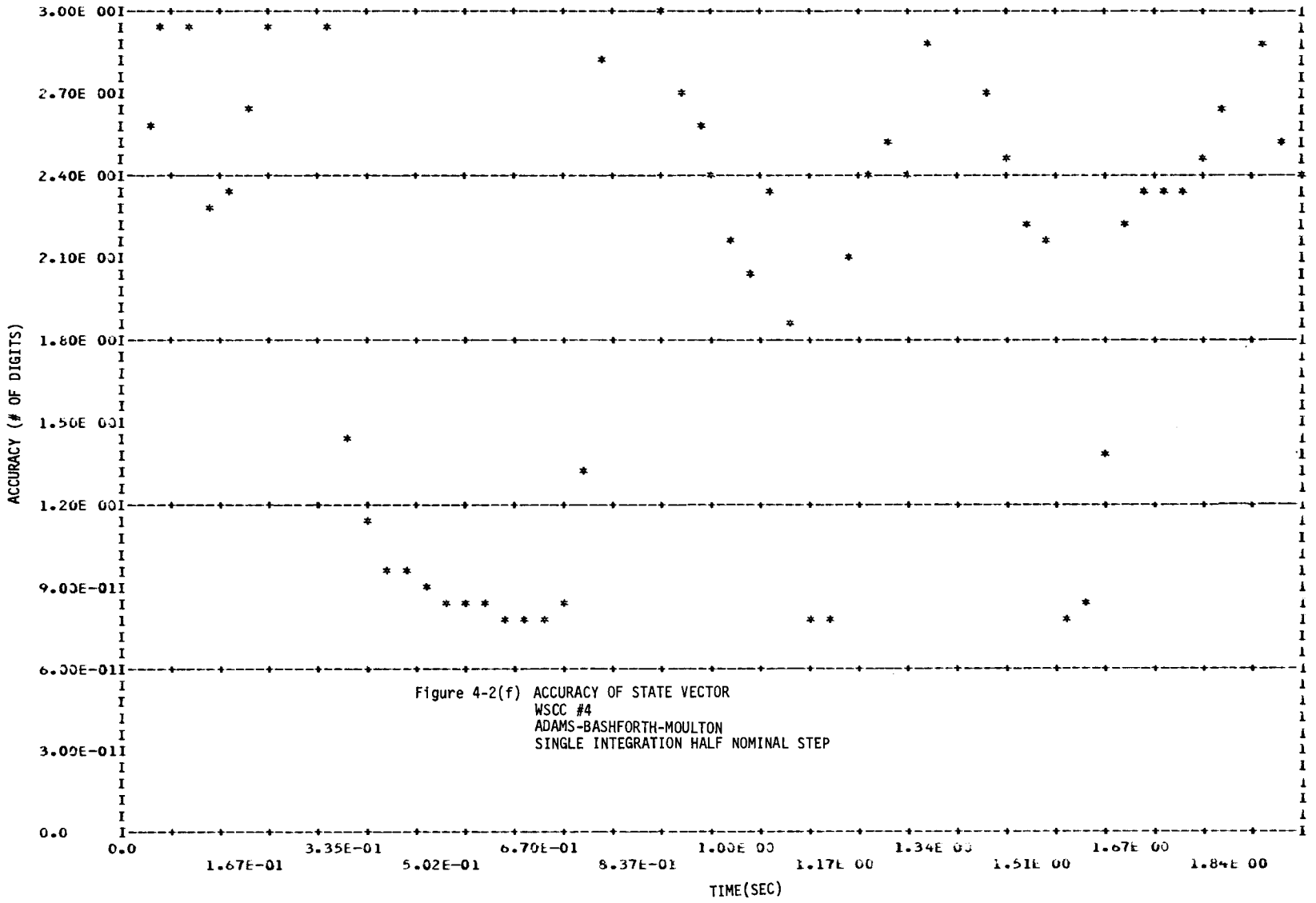


Figure 4-2(f) ACCURACY OF STATE VECTOR
WSCC #4
ADAMS-BASHFORTH-MOULTON
SINGLE INTEGRATION HALF NOMINAL STEP

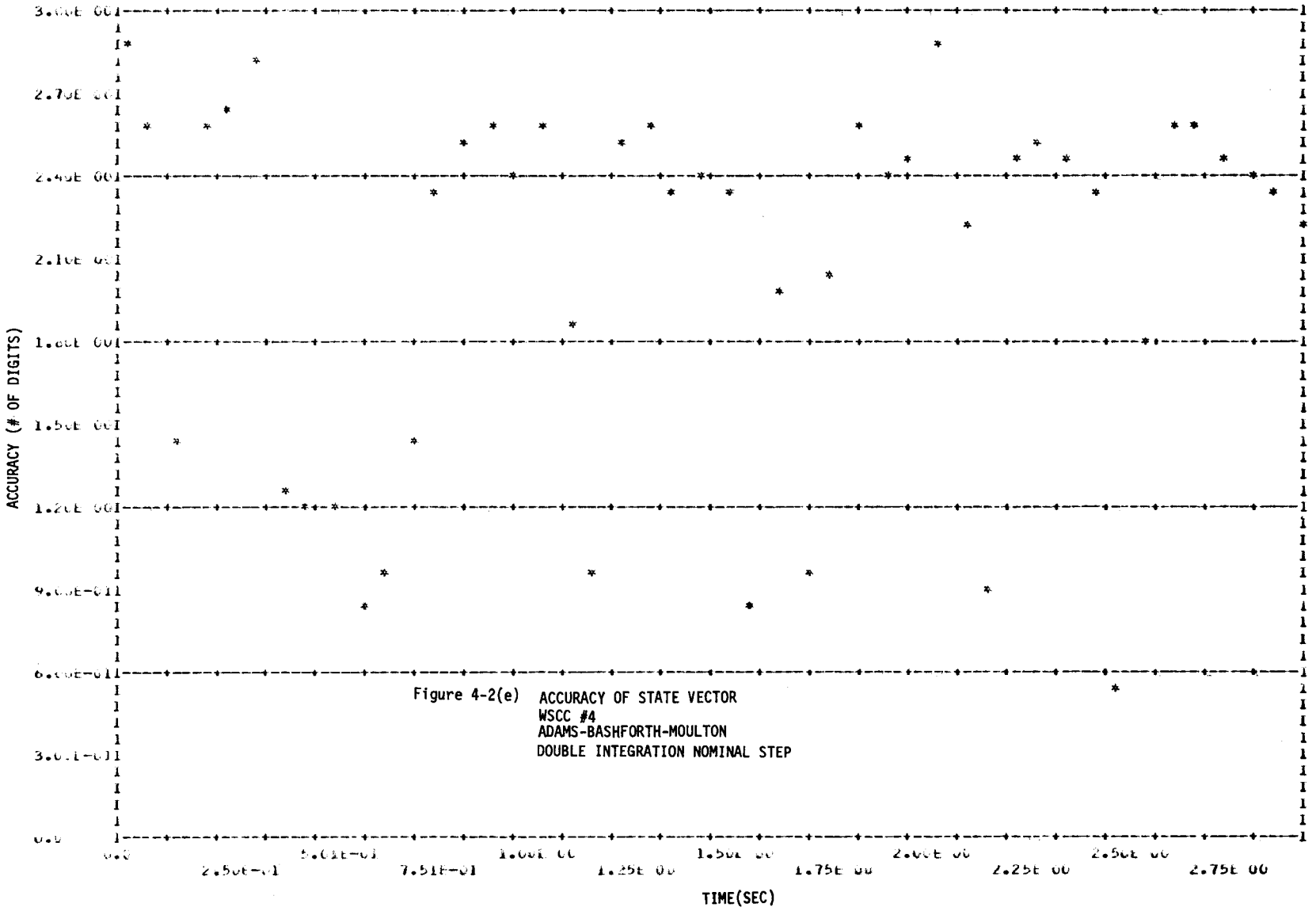


Figure 4-2(e) ACCURACY OF STATE VECTOR
WSCC #4
ADAMS-BASHFORTH-MOULTON
DOUBLE INTEGRATION NOMINAL STEP

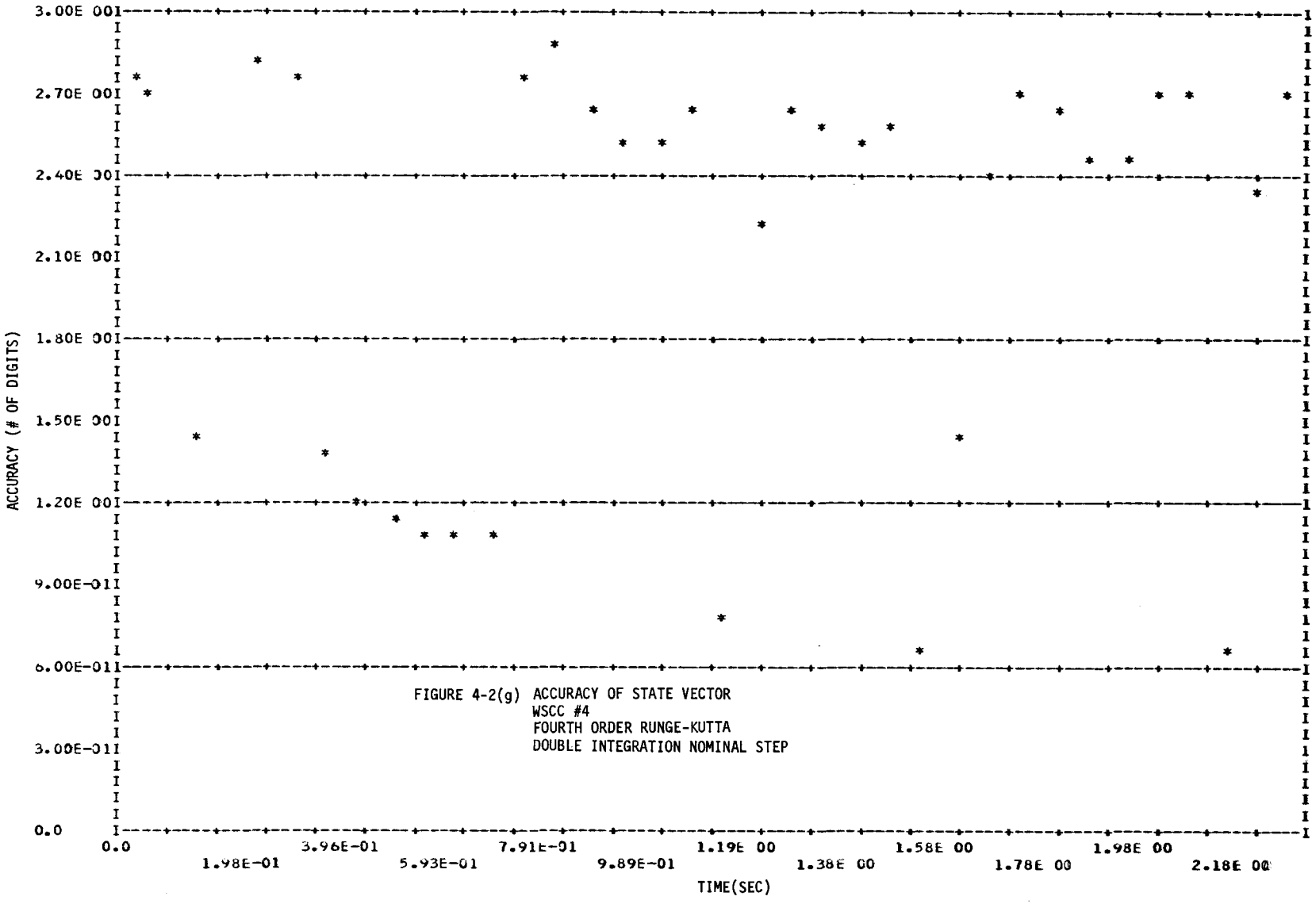


FIGURE 4-2(g) ACCURACY OF STATE VECTOR
WSCC #4
FOURTH ORDER RUNGE-KUTTA
DOUBLE INTEGRATION NOMINAL STEP

Table 4-4, indicates that the 1/2 step method using the variable step integrator NWDVAR is roughly comparable to the full step method with reintegration using the Runge-Kutta integrator DNRKVS, while the full step method using NWDVAR is substantially inferior to the other two methods on problem WSCC #4. (This inadequacy of NWDVAR is believed to be associated with the fact that the error control for one of the exciter variables had to be effectively disabled in order for the problem to be solved at all. When this was not done, solution roughness caused integration failure.)

Because single integration at 1/2 step was found to be roughly comparable to double integration at full step size, the basis for deciding between the two methods must lie in comparative CPU times. Because of the uncertainties in extrapolating the available data, it is not possible to make these comparisons at this time. However, the principal considerations that must go into making such a comparison are the following:

a) The half step integration procedure and the full step integration procedure both require two network solutions for each full algebraic step. Because the full step integration procedure performs its second network solution at the same time point as the first, its second network solution is more efficient than the first. However, with reasonable tuning of the half step integration procedure, the ratio of Newton iterations for the two methods should not exceed 4:3.

b) The half step integration procedure was found to require 60-65% as many G-evaluations as did the full step procedure.

c) It is possible that single integration would perform adequately at a somewhat larger step size. If even 2/3 of the nominal algebraic step size could be achieved, the number of Newton iterations per nominal algebraic step would drop to

$$\frac{2}{(2/3)} = 3$$

precisely what one would expect from the double integration procedure.

3) What is the best method for obtaining generator state rates? This question is the most difficult of the implementation questions because its answer is so heavily complicated by issues of programming and analytical resources and program maintainability.

TABLE 4-4 GLOBAL ACCURACY COMPARISONS OF MACHINE ANGLES AND SLIP RATES

PROBLEM WSCC #1, 2 SEC

	Trapezoidal Rule, Nominal Step Size	Fourth Order R-K Nominal Step Size With Reintegration	A-B-M Nominal Step Size With Reintegration	A-B-M Half Step Size Single Integration
δ_1	12.3225	12.3989	12.4707	12.1902
ω_1	8.3380	8.2205	8.1501	8.2653
δ_2	12.7807	12.7377	12.7069	12.6638
ω_2	4.3654	4.7232	5.1208	4.2993
δ_3	12.5716	12.6050	12.6326	12.4696
ω_3	6.4021	6.5554	6.6809	6.1915

PROBLEM WSCC #4, 2 SEC

δ_1	7.5737	7.6032	6.6199	7.4905
ω_1	6.0798	5.9707	5.9055	6.0224
δ_2	8.2944	8.2796	8.2532	8.2188
ω_2	2.8869	3.0467	3.1225	2.7513
δ_3	7.6143	7.6119	7.6003	7.5300
ω_3	3.8135	4.2572	4.4985	3.8408

Units: δ, radians; ω, radians/sec

On the basis of the results presented in Section 5 (Solution of Difference Equations), it can be directly concluded that a modified Newton iteration procedure is more efficient than a fully honest Newton iteration procedure. Thus the choice can be readily narrowed down to modified Newton iteration (dishonest Newton) or direct solution. It is at this stage that the situation becomes complicated. On the one hand, we have the following considerations that would indicate use of the modified Newton procedure.

a) Code to compute and factor the machine Jacobian must be developed in any case if one wishes to obtain an accurate representation of a machine's contribution to the network Jacobian.

b) Once the basic code for factoring machine Jacobians has been developed, it is a relatively easy matter to implement new models in a transient stability code. Thus, excessive code development costs can be avoided when it is found desirable or necessary to implement new models. This consideration is of special significance when one considers the analytical effort required to implement a device having substantial nonlinearities and multiple feedback loops. In codes using direct solution, these issues have often been addressed by use of decoupling procedures which tend to involve ad hoc prediction procedures combined with some sort of fixed point iteration. While undeniably effective, such decoupling procedures are of necessity model specific so that when new models are introduced into the system, special code must again be developed.

On the other hand, direct solution for generator state rates can very definitely be expected to be more efficient and has been found so in many transient stability programs.

Because of the large programming effort associated with implementing it, direct solution was not implemented in the diagnostic program. However, CPU time projections were made assuming that direct solution would cost as much as the function evaluation portion of a single Newton iteration. These cost projections indicate that direct solution would be substantially more efficient than even a modified Newton procedure. Part of this cost savings derives from the fact that Jacobians are neither computed nor decomposed; but most of it is associated with the fact that two full iterations are required to assure convergence of the modified Newton's method.

4) The question of whether or not one should perform error control arises only when one is using an explicit integrator of the Runge-Kutta type. This is because the Adams-Bashforth-Moulton method contains error control as an integral part of the logic used to control the order of the method. Considering Runge-Kutta methods, then, one observes that when an embedding R-K formula is used (such as the 3E formula discussed above), error control can be essentially free provided one controls the error of the lower order formula (i.e., the second order formula). The embedding formula with order four unfortunately requires 5 stages, so that in this instance one must pay a penalty of 1 extra function evaluation per integration step if one wishes to perform error control. In light of the recommendation to use the R-K 3E method, and because of the presence of both roughness and stiffness in the transient stability problem, it is recommended that error control be performed.

4.4 SUMMARY

In concluding this section, we summarize our overall observations. First, it has been found that the SI approach using the trapezoidal rule may be expected to be more efficient than PE integration whenever the cost of solving the implicit equations of the trapezoidal rule is less than twice the cost of determining a generating system's non-state variables with the state variables known. Second, it has been found that since the PE solution algorithm described in Section 4 is only second order in algebraic step size and because of the solution roughness, low order (second or third) single step integration algorithms are most appropriate for use with the partitioned solution algorithm. Third, the following observations are made concerning the details of implementing the partitioned solution, explicit integration algorithm.

a) Logarithmic extrapolation of bus voltages should be performed.

b) Reintegration of the generator differential equations must be performed if "large" algebraic steps are to be used. Comparable accuracy can be achieved with a single integration pass and an algebraic step size 1/2 as large. Table 4-3 shows that for the integration method R-K 3E these two procedures should be fairly competitive in CPU time.

c) Either direct solution or modified Newton solution of the generator's algebraic equations (the G-equations) should be implemented. Direct solution may be expected to be substantially more efficient while a modified Newton approach offers greater flexibility with respect to the introduction of new device models.

d) Because of the presence of solution roughness together with the possibility of stiffness, local error control should be performed whenever an explicit integration method is used.

4.5 REFERENCES

[1] K. N. Stanton, S. N. Talukdar, "New Integration Algorithms for Transient Stability Studies", IEEE Trans. Power Apparatus and Systems, PAS-89 (June 1970), 485-991.

[2] H. W. Dommel, N. Sato, "Fast Transient Stability Solutions", IEEE Trans. Power Apparatus and Systems, PAS-91 (July/August 1972), 1643-1650.

[3] H. L. Fuller, P. M. Hirsch and M. B. Lambie, "Variable Integration Step Transient Analysis: VISTA", 1973 PICA Conf. Rec., pp. 156-161.

[4] G. Dahlquist, A. Bjorck and N. Anderson (1975), Numerical Methods, Prentice-Hall, Englewood Cliffs, N.J.

[5] P. Henrici (1962), Discrete Variable Methods for Ordinary Differential Equations, John Wiley and Sons, 1962, New York.

[6] C. W. Gear (1971), Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, N.J.

[7] Boeing Computer Services Subprogram Library (BCSLIB), BCS-10023, Boeing Computer Services, Inc. 1974.

Section 5

ALGEBRAIC SOLUTION

This section discusses the iterative solution of the nonlinear equations arising from the implicit formulation and the network. The algebraic solution is a major part of the computation per integration step. A reduction in cost per iteration or number of iterations provides a significant reduction in the total simulation cost.

The characterization of the problem under consideration is as follows. We seek to solve the system of algebraic/differential equations of the form

$$\dot{x} = f(x,y,t)$$

$$0 = g(x,y).$$

The solution is known for some $\hat{t}$ and we seek the solution at $\hat{t}+h$, where h is the step size. Section 3 considered the selection of h and choice of a difference equation, leading to the algebraic problem

$$F(x(\hat{t}+h),y(\hat{t}+h)) = 0, \tag{5-1}$$

where $\hat{x} = x(\hat{t})$, $\hat{y} = y(\hat{t})$ are known, satisfying

$$F(x(\hat{t}),y(\hat{t})) = 0.$$

Newton's method for the solution of (5.1) requires the computation of the Jacobian matrix for some initial approximation to the solution (x_o,y_o).

$$J(x_o,y_o) = \frac{\partial F(x_o,y_o)}{\partial(x,y)}$$

and the solution of the system of linear equations

$$J(x_o,y_o) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = -F(x_o,y_o). \tag{5-2}$$

Then we update the approximate solution

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_o \\ y_o \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix},$$

compute a new $J(x_1,y_1)$ and solve for a new correction term. The process is continued until (x_i,y_i) makes $F(x_i,y_i)$ sufficiently small.

Work in solving these equations is divided into several basic steps. They are

- Choose (x_o,y_o)
- Evaluate $F(x_i,y_i)$, $i = 0,...$
- Compute $J(x_i,y_i)$
- Solve the linear system as in (5.2)
- Update the approximate solution, (x_i,y_i)
- Decide when convergence is satisfactory

The choice of (x_o,y_o) has an impact on the number of iterations required. The solution at the previous time step is a reasonable estimate but can be improved as is shown in subsection 5.3. The solution of the linear system at each iteration is a significant part of the computation. A discussion of the costs of solving these equations is given in subsection 5.1.

The major cost at each iteration is the factorization of the Jacobian matrix when solving the linear system. Thus much of this section is devoted to various ways of reducing the number of factorizations. Study is made of leaving J fixed at $J(x_o,y_o)$ for one or more time steps (Very Dishonest Newton Method). Thus, J is refactored only when recalculated. Study is made of the BPA approach which leaves the network portion of J fixed except when the network changes. This virtually eliminates that portion of the factorization. Also, the Quasi-Newton approach of updating the factorized Jacobian is evaluated.

Thus, all of these methods may be thought of as approximations to Newton's method. The objective is to identify the approximation with the best features of accuracy, reliability, and efficiency. These various methods are analyzed and a number of test cases have been studied using the Diagnostic Program.

This range of iterative techniques largely covers the types of iterations used in transient stability codes. The Arizona Public Service transient stability code

is close to Newton's method. Though an approximation to the Jacobian based on engineering judgment, rather than the exact Jacobian, is used, this approximation is recalculated and refactorized at each iteration. The BPA transient stability code leaves the network portion of the Jacobian approximation fixed except when the network changes. The Newton or APS approach minimizes the number of iterations but maximizes the cost per iteration. The BPA approach minimizes the cost per iteration but takes many more iterations per time step. It will be shown in 5.2 that Very Dishonest Newton's Method costs about the same per iteration as the BPA approach while approaching the iteration count of Newton's method. Prediction will be shown to be an effective way of reducing iteration counts in 5.3.

5.1 SOLUTION OF SPARSE LINEAR EQUATIONS

The solution of the sparse linear system is a major cost of each iteration. The purpose of this discussion is to show the relative costs of the steps in the solution. This provides a basis for evaluating the iterative techniques.

Solution of the sparse system of linear equations involves three major steps. These are (in the terms of the software used, SPARSE, [3].

- Analyze - choose the optimal ordering pivot strategy
- Factor - calculate the LU factorization
- Operate - perform forward and backward substitution for the solution

For a brand new system of equations, all three functions must be performed. If the same matrix is reused, only the operate step need be done. Given a second matrix, with the same sparsity pattern as the first, one can choose to use the same pivot strategy. Thus, only factor and operate functions need be performed. This ignores the numerical differences between the matrices. Subroutine SPARSE estimates error growth to allow the decision to re-analyze.

The data below gives timing information for each of the functions performed by SPARSE.

	9 Bus	39 Bus
Analyze	.13 sec	.47 sec
Factor	.041 sec	.15 sec
Operate	.013 sec	.04 sec

The ratio of about 3 to 1 or 4 to 1 between factor and operate times is typical for transient stability codes. Factorization of a sparse matrix is a k^2N order

process, where k is close to the average number of row/column elements encountered during factorization. Forward and back substitution is a process of order 2kN. An average of about six elements corresponds to the 3 to 1 ratio. The BPA program is quoted at a 5 to 1 ratio [2] which indicates slightly higher fill.

The times required for function and Jacobian evaluation are not drastically different from that required to do a forward and back substitution. Again for the 9 bus case:

Evaluate functions	.009 sec
Evaluate Jacobian matrix	.010 sec

Transient stability programs normally choose a pivot strategy that may be used throughout the simulation. Thus, analyze time is not significant for this study.

To summarize, we have seen that the cost of evaluating the functions, evaluating the Jacobian and operate are about the same. The cost of factorization is 3 to 4 times that of operate. The goal in subsection 5.2 is to reduce the number of times the Jacobian is evaluated and factored.

5.2 ITERATIVE TECHNIQUES

This section discusses in detail the methods used to take an iteration step. Also discussed is the convergence criteria used in the Diagnostic Program and the APS program. The methods are compared based on iteration count and time. Test cases are run on the Diagnostic Program to demonstrate the difference between Newton's method and Very Dishonest Newton.

For simplicity of notation, we will denote the system of nonlinear equations to be solved in vector form as

$$F(X) = 0$$

where F is the vector of residuals and X includes all machine and network variables. Section 2 gives a locally linearized form from which a Jacobian (J) is immediately available.

Given an initial estimate $X^{(o)}$, an iteration solves a system of linear equations of the form

$$J \cdot \Delta^{(o)} = -F(X^{o})$$

to find the correction vector $\Delta^{(o)}$ such that

$$X^{(1)} = X^{(o)} + \Delta^{(o)}$$

is an improvement. The sparsity and structure of the matrix J is exploited to save time and storage. This process is repeated until the convergence criteria is satisfied. The final iterate is accepted as the solution at that time step, t_k. The iteration can then be initiated for the next time step, t_{k+1}.

The solution from one time step to the next provides a good initial estimate of the solution (see subsection 5.4). This eliminates the necessity of a one-dimensional search to assure convergence.

The methods studied in this section are Newton's method, a semi-Newton's method (Dishonest Newton), quasi-Newton, and fixed point iteration with relaxation. The effect on cost per iteration and number of iterations required for convergence of each method is discussed. The cost components of an iteration are identified and compared.

The comparisons are primarily based on timing analysis of the solution of the linear equations. Detailed timing of the Diagnostic Program is done to compare methods and to relate the time in the linear equation solution to other program functions.

Newton's Method. Newton's method calculates the correction vector Δ by solving the linear system

$$J(X^{(i)})\Delta^{i} = -F(X^{(i)})$$

at each iteration. The cost per iteration includes

- Evaluating the residuals (F)
- Evaluating the Jacobian (J)
- Factoring the Jacobian
- Performing forward/back substitution

Many variations of Newton's method have been published to reduce the cost per iteration. The major concept is to use the information from the Jacobian (possibly with updates) on more than one iteration. For notational purposes, let

$$J_k^{(i)} = J(X^{(i)}(t_k))$$

That is, the subscript denotes time and the superscript denotes iteration count.

Dishonest Newton's Method. By the Dishonest Newton's Method (or Very Dishonest Newton, VDHN) we mean an iteration of the form

$$J_k^{(o)} \cdot \Delta^{(i)} = -F(X^{(i)}(t_n))$$

where $n \geq k$. That is, the same Jacobian is used throughout the iteration for one or more time steps.

The VDHN method reduces the cost per iteration by not evaluating the Jacobian or doing the factorization at every step. Using the sample times given for 9 and 39 bus cases above, VDHN will be equivalent in cost to Newton's method if VDHN requires 3 to 4 times as many iterations. (See the section on test case for times and iteration counts.)

To implement VDHN, one must choose an algorithm for deciding when to reevaluate the Jacobian. The simplest approach is to reevaluate every N^{th} time step. Another criteria might measure the speed of convergence and reevaluate when convergence is slow. The program should also provide for iteration failures, to reevaluate and try again. The Diagnostic Program uses all three criteria. The test results listed under test cases required a Jacobian evaluation, at least every fifth time step. If the previous time step required five iterations, it did a reevaluation. Iteration failures are handled.

Fixed Point Iteration with Relaxation (FPIR). Carrying the fixed Jacobian concept one step further, the BPA transient stability program holds a major portion of the system matrix fixed except at network changes [2]. The network related equations (Y matrix of Section 2) is maintained as a complex symmetric matrix with modification to approximate the coupling to the machine equation. The approximation adds a fictitious slack node with an appropriate admittance to cause convergence. This

modified matrix is factorized only at network changes. Otherwise, only forward and backward substitution are performed. For this report, we have termed this method a "fixed point iteration with relaxation (FPIR)". BPA calls it an iterative solution using the triangularized admittance matrix [2].

There are two advantages to the BPA approach. First is the reduced storage for Y. It is reasonable to assume that the sparsity remains the same. The symmetry reduces the storage of the factors of Y by a factor of two. The use of the complex form provides an additional factor of two. Thus, the savings in storage for Y is a factor of four.

The second advantage of the BPA approach is the fact that Y is factorized only at network changes. BPA quotes a ratio of 5/1 between factorization and forward and back substitution (repeat solution) [2]. That is, BPA can use FPIR with up to 5 times as many iterations and be competitive with Newton's method.

With the diagnostic program, we found that when using VDHN, the factorization of Y was about 1.5% of the total time. That is about 10% of the network related use of SPARSE.

To compare FPIR with VDHN, we need to compare (related to the Y matrix)

- Cost per iteration
- Iteration count

To compare the cost per iteration we look at the cost of doing forward and backward substitution. For the complex case the complex matrix, Y_C, is factored into symmetric factors

$$Y_C = LL^T$$

and we denote the number of elements (using the symmetry) $N(LL^T)$. For the real case the real matrix, Y_R, is factored into the unsymmetric factors

$$Y_R = LU.$$

and again we have the notation N(LU). The elements of the complex factorization will number about one eighth that of real case. Forward and backward substitutions cost on the order of the number of elements used. For the complex case that is

$2xN(LL^T)$ due to using the symmetric factor twice. For the real case it is N(LU). To compare these, we note that complex arithmetic costs about 4 times that of real. Thus

$$4 \times (2xN(LL^T)) \approx N(LU)$$

and we conclude that the cost per iteration will be about the same for FPIR and VDHN.

The available information indicates that VDHN takes significantly fewer iterations than FPIR. In our analysis we have used Newton's method as a baseline for comparison. The quoted [2] iteration count for the BPA code is 5 to 7 as compared to 2 iterations for Newton's method in the same code. The BPA code uses a form of prediction (see subsection 5.2). The Diagnostic Program showed an increase of less than 15% in iteration count between VDHN and Newton (without prediction).

Unless the savings in storage is critical, the real formulation with VDHN is the cheaper approach.

Quasi-Newton. Quasi-Newton algorithms again use an "old" Jacobian but improve the approximation by an updating process. Reid [4] has suggested an update of the form

$$J^{(i+1)} = J^{(i)} + \frac{(\gamma^{(i)} - J^{(i)}\Delta)\Delta^T}{||\Delta||^2}$$

where γ is the change in the residual and care is taken to maintain the same sparsity pattern. Unfortunately, this requires a factorization as well as a significant number of matrix operations.

Ideally, the updates should be done on the factored form of J. At this time, no such software is available. However, we can state that the quasi-Newton approach would not be competitive. The analysis for full systems [1] shows that the update requires at least a solution of the linear system of equations, as well as several matrix operations. Thus, quasi-Newton techniques would cost about twice as much per iteration as VDHN and would have to take one-half the iterations to be competitive. The figures given under testing for Newton's method can be used as a lower bound for iteration count for quasi-Newton.

Convergence Criteria. Convergence criteria is an important part of any iterative solution. It would be improper to compare methods without establishing equivalent criteria. Normally an iteration is said to have converged when a measure of both (or either) the correction vector (Δ) or the residual vector are small. The measure is often some norm (least squares or maximum) of the vectors.

The measures used in transient stability programs are enlightened choices of what are important to the simulation. The criteria used in the APS and Diagnostic Programs are described below.

The APS transient stability program convergence criteria tests two data types. At the end of each iteration the mismatch in current for each node is calculated. If any mismatch is greater than .01, another iteration is taken. This is a test on a subset of the residuals. If this test is satisfied, the code estimates the next correction in machine angle based on the present iterate. This is equivalent to taking a Gauss-Siedel iteration on just the machine angles. The Gauss-Siedel correction is used to test the convergence of the present iterate. A discrepancy greater than .005 causes another iteration to be taken.

The Diagnostic Program tests a subset of the correction vector as its first convergence criteria. The subset includes the bus voltages (.01 tolerance) and the machine angles (.005 tolerance). The test on bus voltages is similar in effect to the APS test on current mismatch. The values are of similar magnitude. Once this first test is satisfied, the Diagnostic Program also requires that all machine functions (see Section 2 above) be nearly satisfied. That is, the residuals be less than .01. If, at 5 iterations, the first test is satisfied and the second is not, the Diagnostic Program accepts that iteration.

The criteria used in the Diagnostic Program cause it to take about 1½ more iterations to satisfy the same convergence criteria as the APS program. The comparison of two iterates is used by the Diagnostic Program to test the correction vector. This requires (using zeroth order prediction) at least two iterations. The subsequent calculation of the residuals is similar between the two programs but the Diagnostic Program continues (after the second test) to calculate a new iterate.

The test on the residuals occasionally causes the Diagnostic Program to take 5 iterations when a discontinuity is encountered.

Criteria for Evaluation. The basic criteria for evaluating iterative techniques is time. That is, which method takes the least amount of computer time while maintaining the required accuracy level. The evaluation will be done by detailed comparison of test data in sections that follow. All cases were run on the Diagnostic Program using the same convergence criteria.

It is of value to list the cost factors to be used in the comparison:

- Number of iterations
- Residual evaluations
- Jacobian evaluations
- SPARSE factorization
- SPARSE forward and back substitution

The major accuracy requirement is that the swing curves maintain the same relationship. It is normal to plot relative swing curves to verify that this requirement is met.

Details on the test cases are given in the next section.

Test Cases and Results. The purpose of this section is to illustrate the elements of the evaluation criteria for a specific problem. The test case used is case number 4 of the WSCC 9 bus test series. This case is used to compare Newton's method to VDHN. As a "worst case", case 4 minimizes the improvement observed using VDHN. The comparisons use the trapezoidal integration method as a basis.

Detailed timing information is provided by a special timing program executed with the Diagnostic Program. This program, Program Performance Evaluator (PPE), is a Boole and Babbage, Inc. proprietary product leased by BCS.

Table 5-1 shows total times and PPE breakdown for Newton's method and VDHN on WSCC #4. Timing information is subject to some statistical error. PPE determines timing by sampling the program activity at fixed time intervals. The samples are later analyzed to show the areas of program activity. A detailed breakdown of the portions of SPARSE is given in Table 5-2.

Table 5-1

PPE TIMING, NEWTON AND VDHN (SEC)

WSCC #4, TRAP, NO PRED., 7 SEC

SECTION	NEWTON	VDHN
Control-Misc	3.09	2.31
Loads	.53	.34
Function and Jacobians	5.25	4.15
Generate Linear Differences	4.35	1.82
Network Related	1.40	.69
SPARSE	31.32	7.90
Diagnostic Related	3.88	3.14
I/O	5.29	4.57
Utility	7.12	1.37
Other	2.77	2.71
Total	65.00	29.00

Table 5-2

DETAILS ON SPARSE TIMING (SEC)

WSCC #4, TRAP, NO PRED., 7 SEC

SECTION	NEWTON	VDHN
Control	5.5	2.9
Analyze	.43	.46
Factor	13.9	.2
Operate	18.4	8.80

The effect of VDHN on iteration count is shown in Table 5-3. Table 5-3 lists the number of steps versus the number of iterations required for convergence. It also gives the average iteration count for Newton and VDHN. For case 4, the average

the average iteration count increased by less than 10%. As pointed out by Dommel and Sato [2], the inclusion of nonlinear loads significantly affects the iteration counts. For WSCC case #16 the average iterations went from 3.7 for Newton to 4.3 for VDHN, an increase of 15%.

Table 5-3

NUMBER OF STEPS/ITERATIONS REQUIRED

NEWTON AND VDHN

	Iterations Required					Average Iteration Count
	1	2	3	4	5	
Newton	0	0	22	191	1	3.9
VDHN	0	0	5	189	20	4.1

Figures 5-1 through 5-4 show an RMS measure of the convergence of X for Newton's method during the simulation. This is an RMS measure of the entire X vector, not just those elements used in the convergence criteria. The steady convergence from iteration to iteration is readily apparent. Approximately seven digits of accuracy are achieved by the fourth iteration.

Figures 5-5 through 5-8 give data for VDHN similar to 5-1 through 5-4. Figure 5-5 shows the same initial conditions as did 5-1. Figure 5-6 shows that VDHN gives slower improvement as the Jacobian approximate gets older. This is to be expected. Figures 5-7 and 5-8 show the total improvement achieved. VDHN does not achieve the same total accuracy as Newton's method. Recall that the convergence criteria used tests a subset of the iteration variables and these figures are an RMS norm of the total vector. The VDHN code does not achieve as much accuracy as Newton's method. The poor convergence at about 6 sec shows up in the swing curve plot for generator 3 (Figure 5-12).

Figures 5-9 through 5-12 give swing curve plots for generators 2 and 3 using Newton's method and VDHN. This data indicates VDHN maintains the required accuracy.

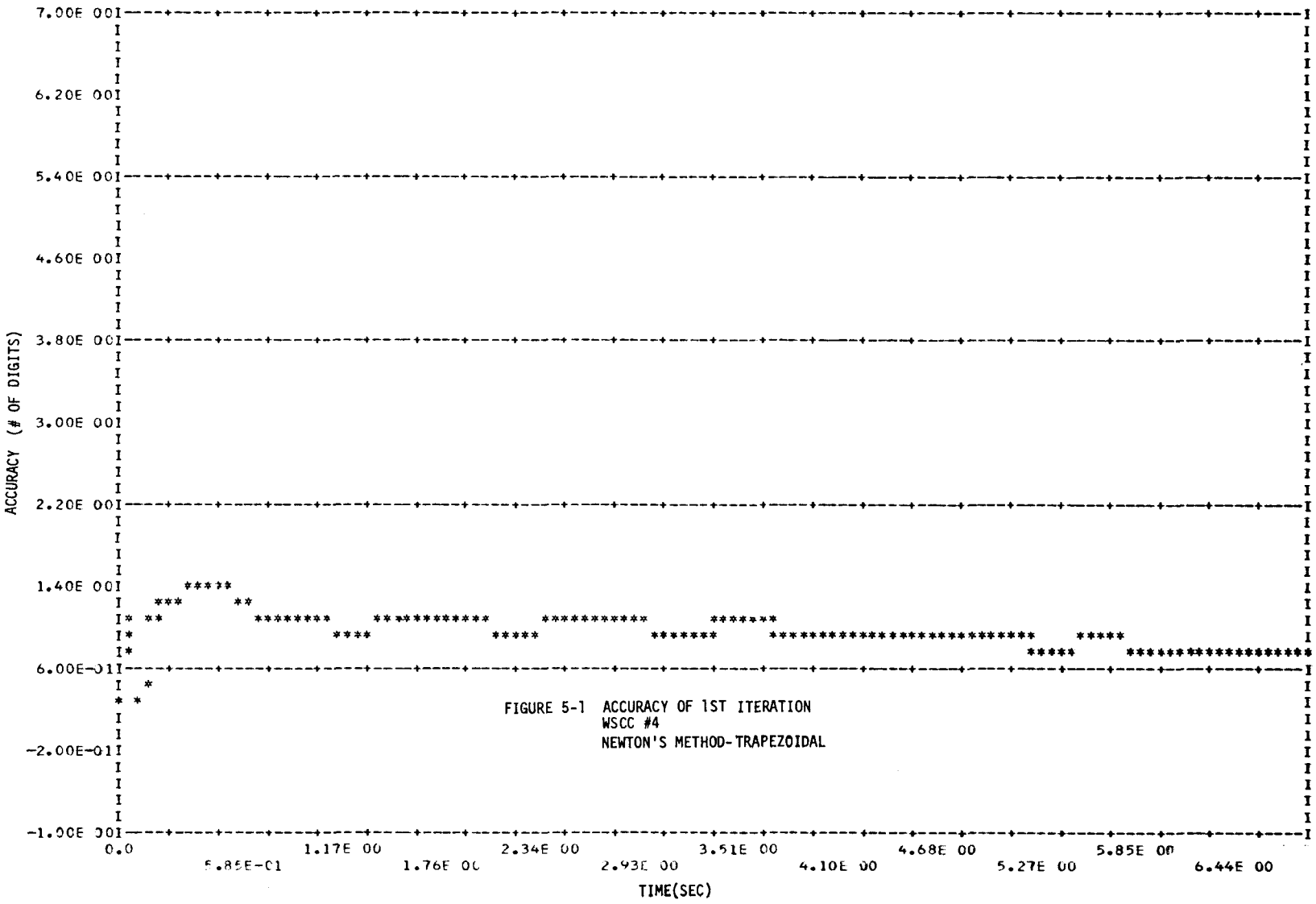


FIGURE 5-1 ACCURACY OF 1ST ITERATION
WSCC #4
NEWTON'S METHOD-TRAPEZOIDAL

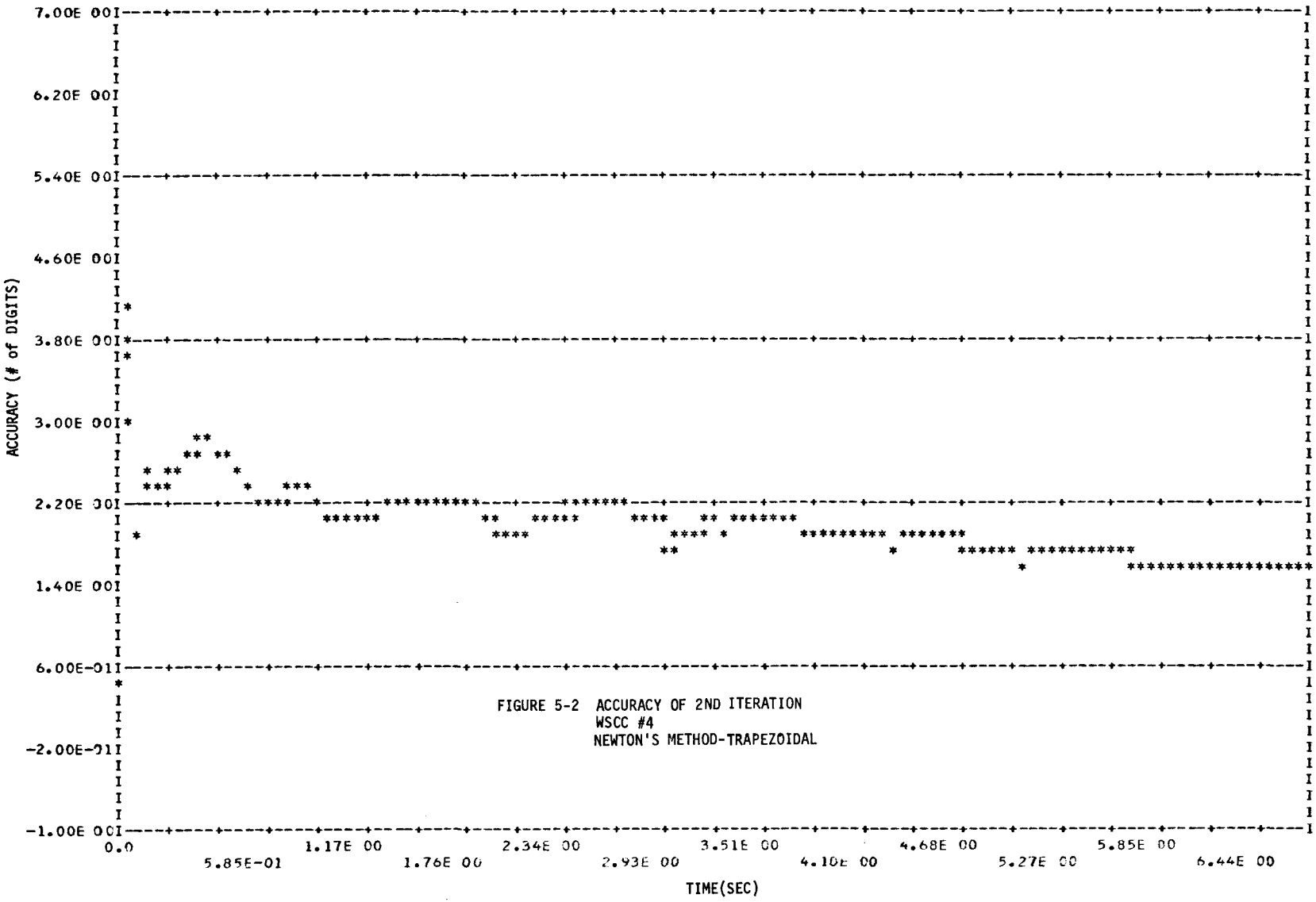


FIGURE 5-2 ACCURACY OF 2ND ITERATION
WSCC #4
NEWTON'S METHOD-TRAPEZOIDAL

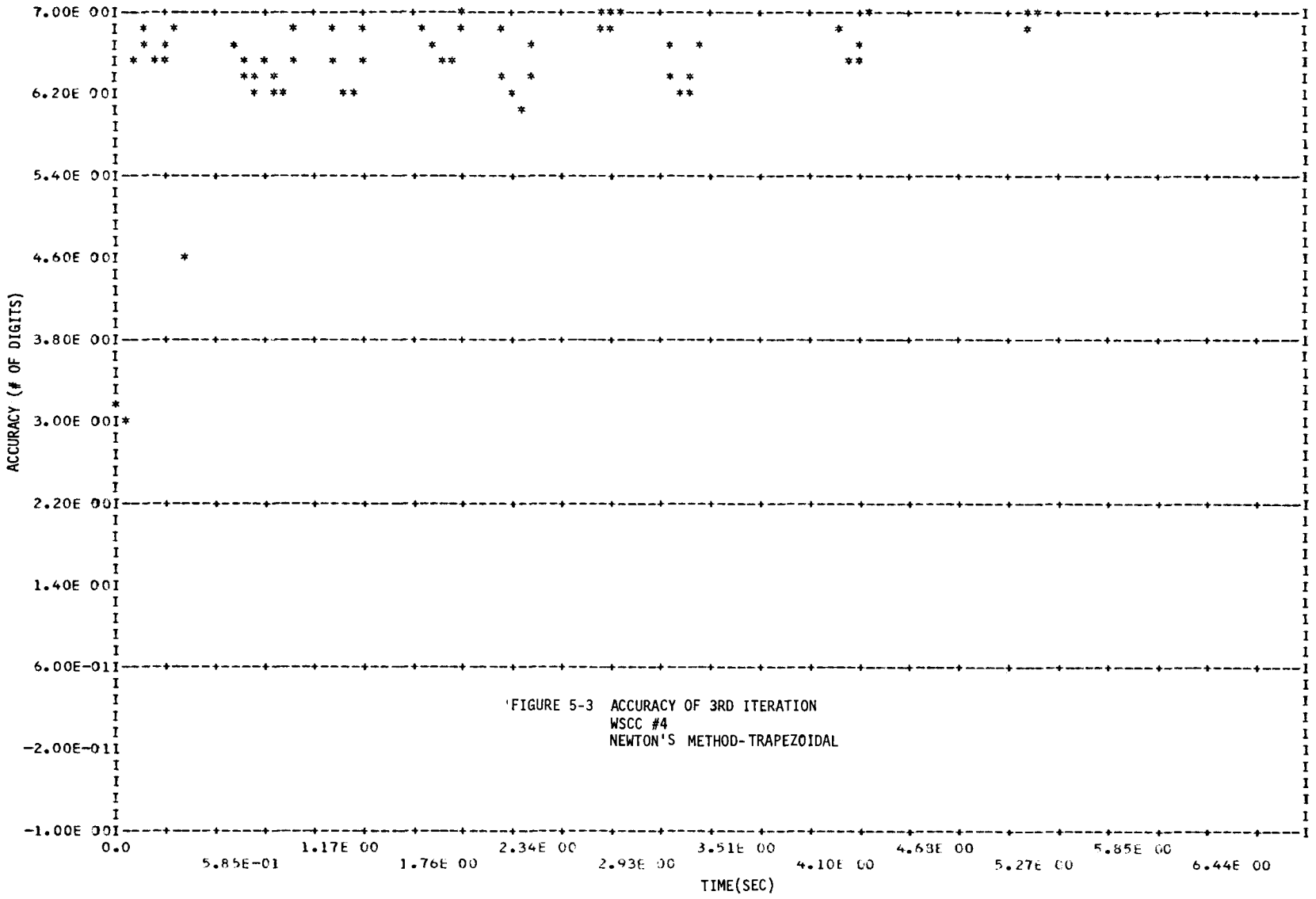


FIGURE 5-3 ACCURACY OF 3RD ITERATION
WSCC #4
NEWTON'S METHOD-TRAPEZOIDAL

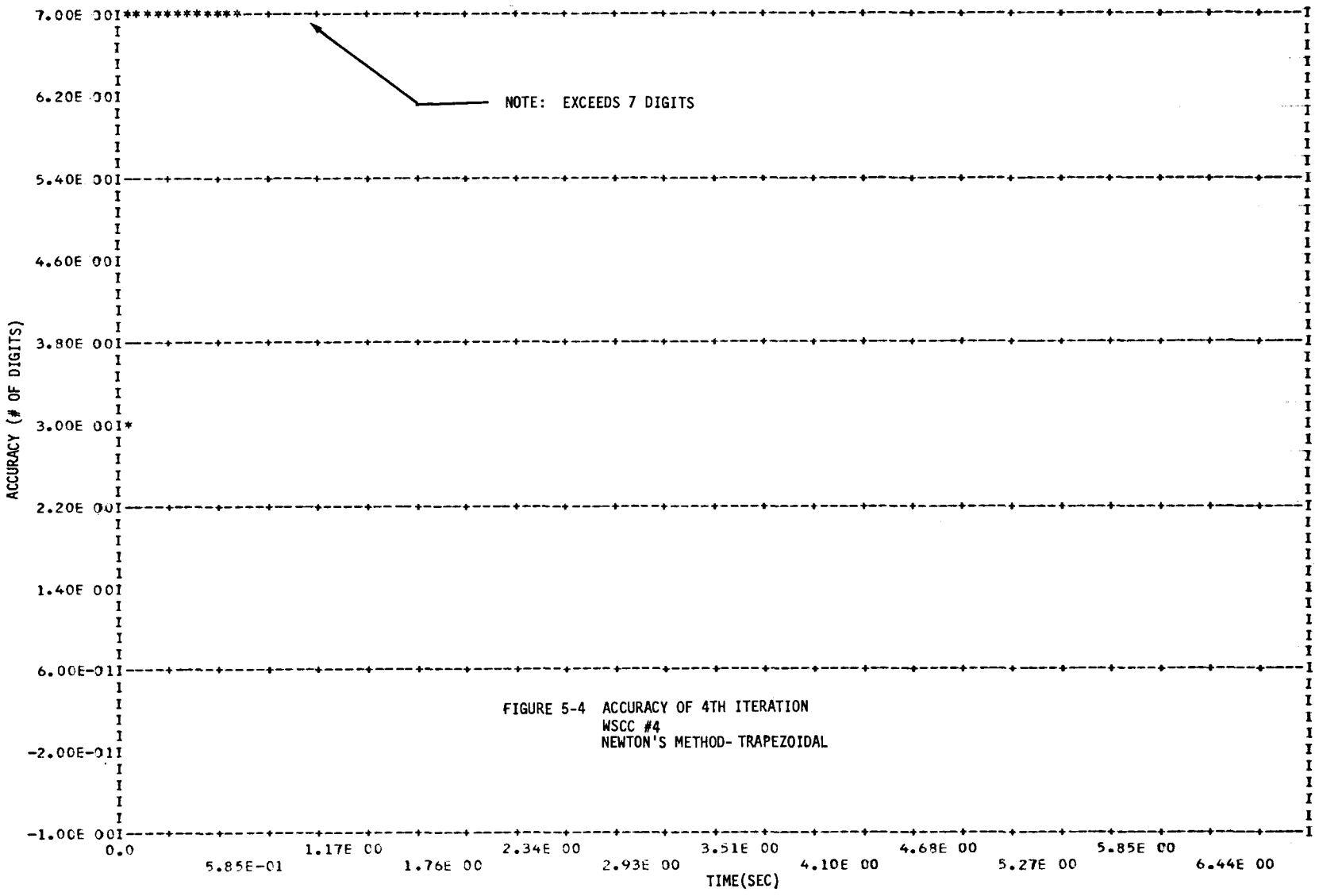


FIGURE 5-4 ACCURACY OF 4TH ITERATION
WSCC #4
NEWTON'S METHOD- TRAPEZOIDAL

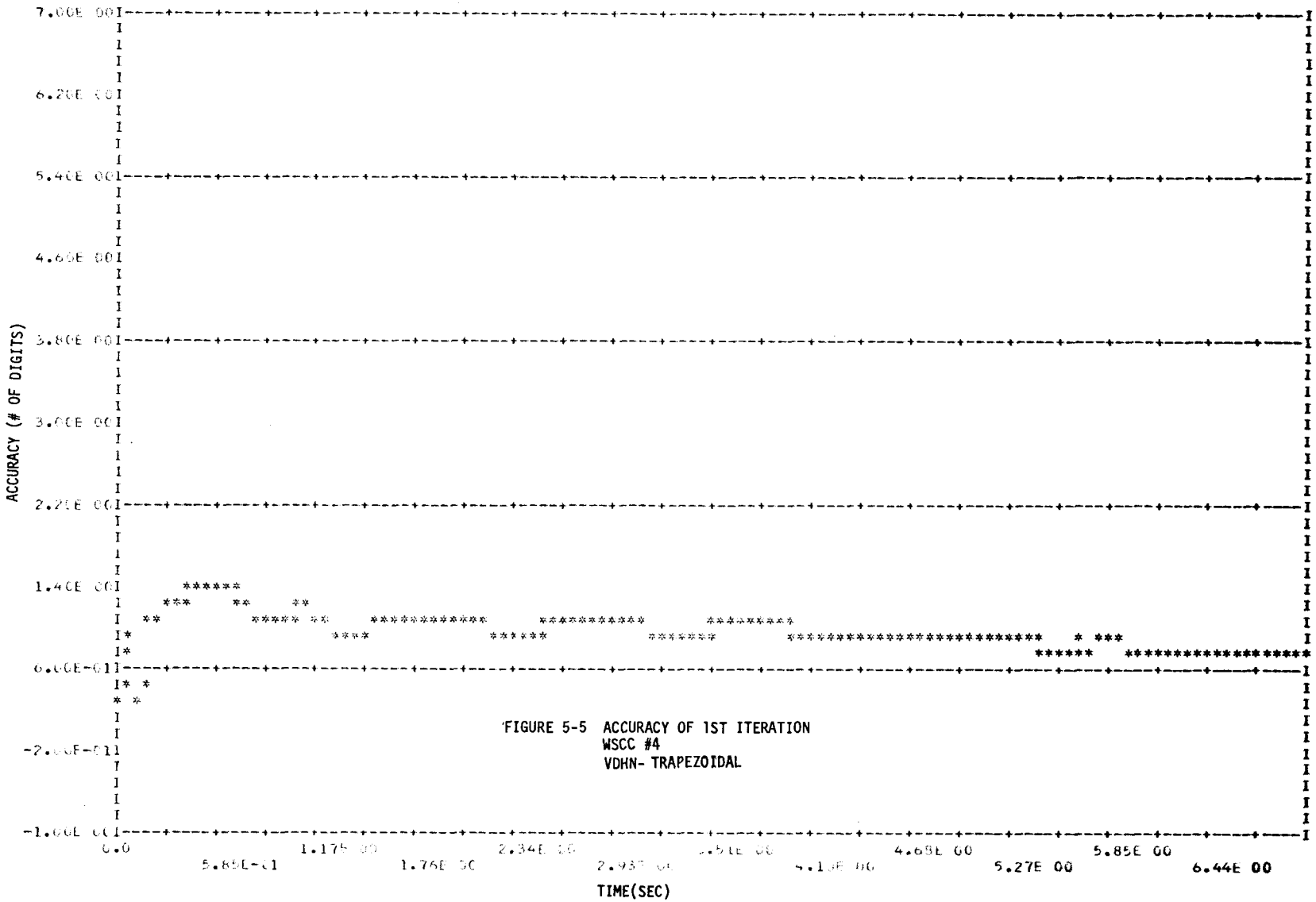


FIGURE 5-5 ACCURACY OF 1ST ITERATION
WSCC #4
VDHN- TRAPEZOIDAL

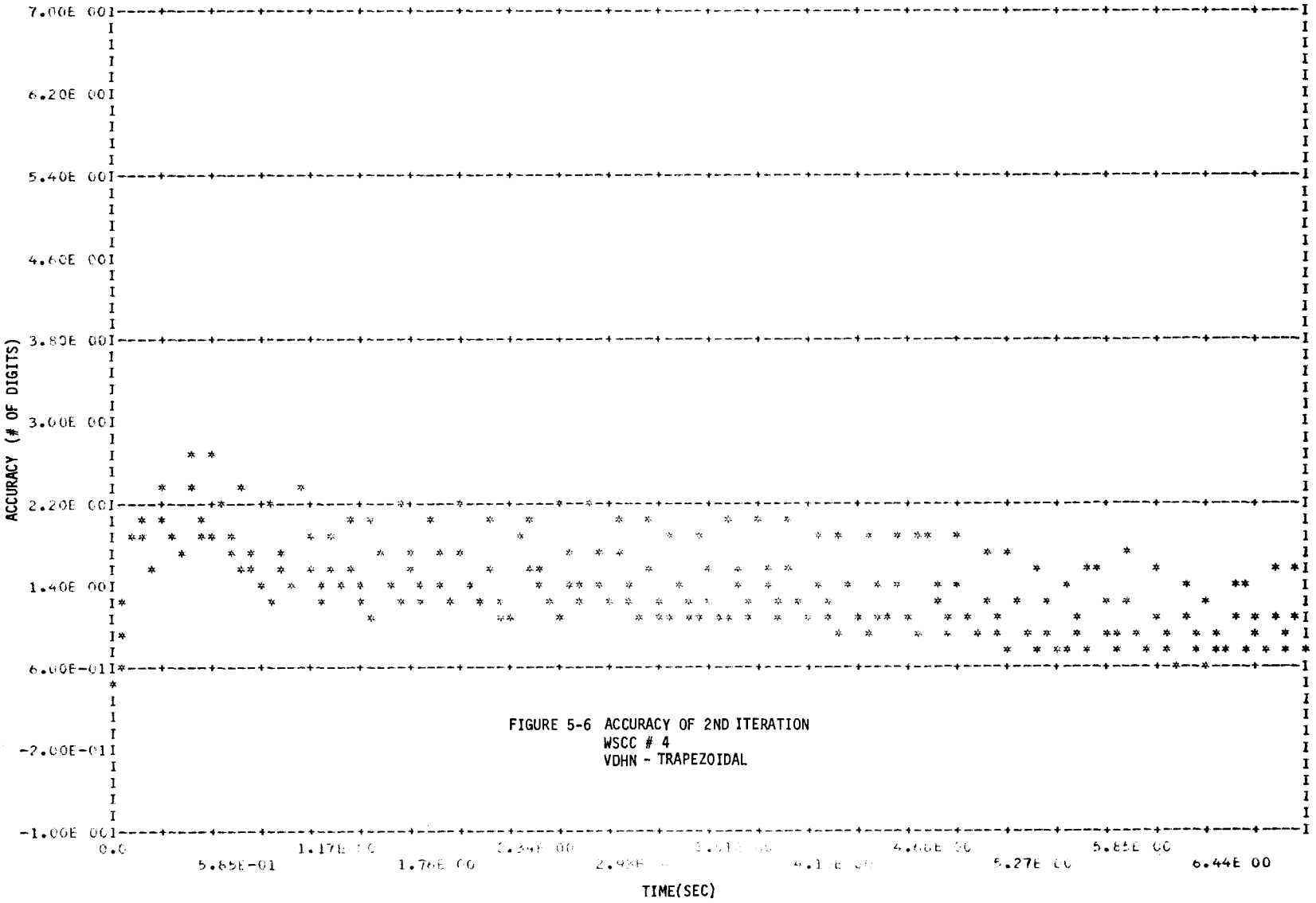


FIGURE 5-6 ACCURACY OF 2ND ITERATION
WSCC # 4
VDHN - TRAPEZOIDAL

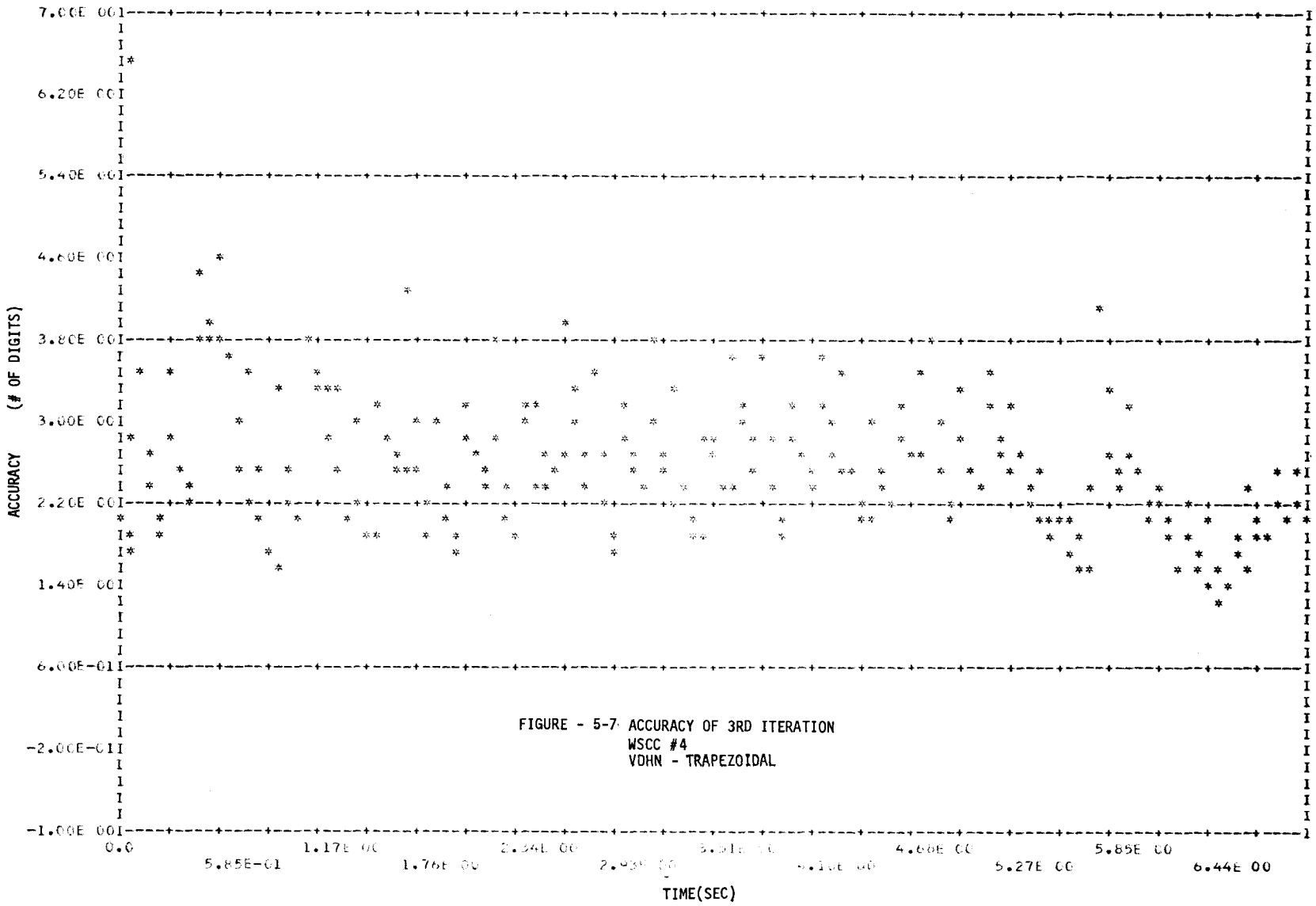


FIGURE - 5-7 ACCURACY OF 3RD ITERATION
WSCC #4
VDHN - TRAPEZOIDAL

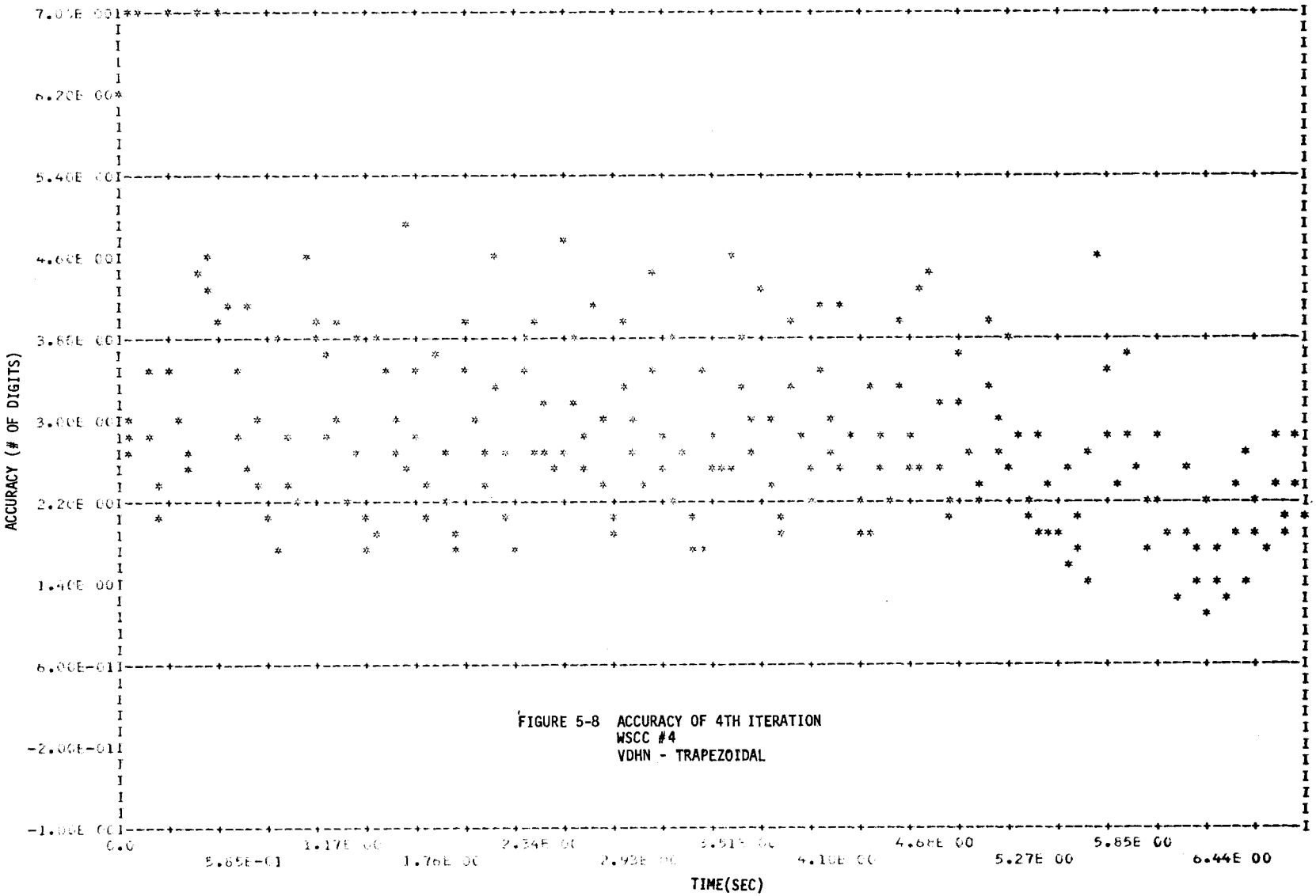


FIGURE 5-8 ACCURACY OF 4TH ITERATION
WSCC #4
VDHN - TRAPEZOIDAL

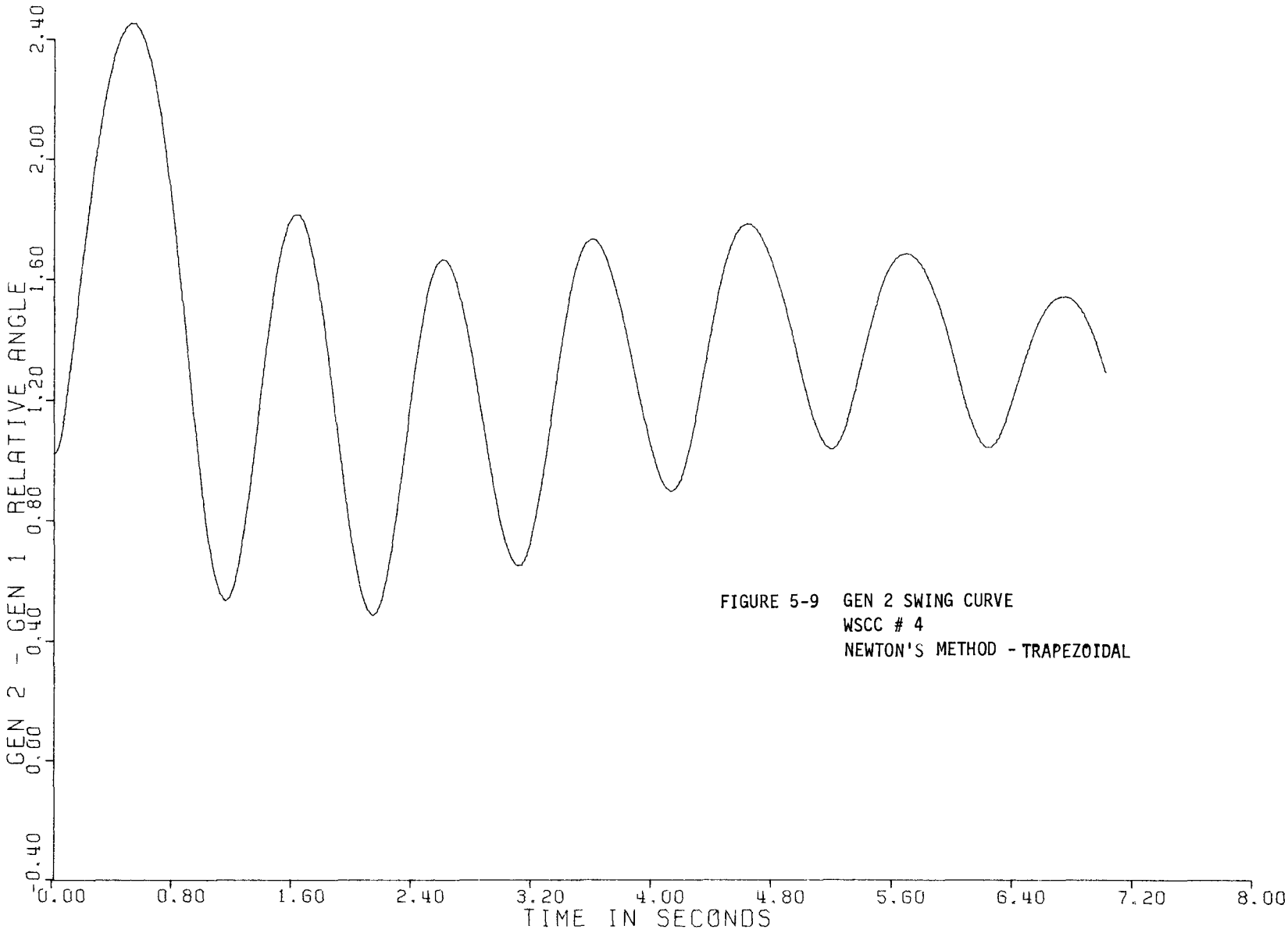


FIGURE 5-9 GEN 2 SWING CURVE
WSCC # 4
NEWTON'S METHOD - TRAPEZOIDAL

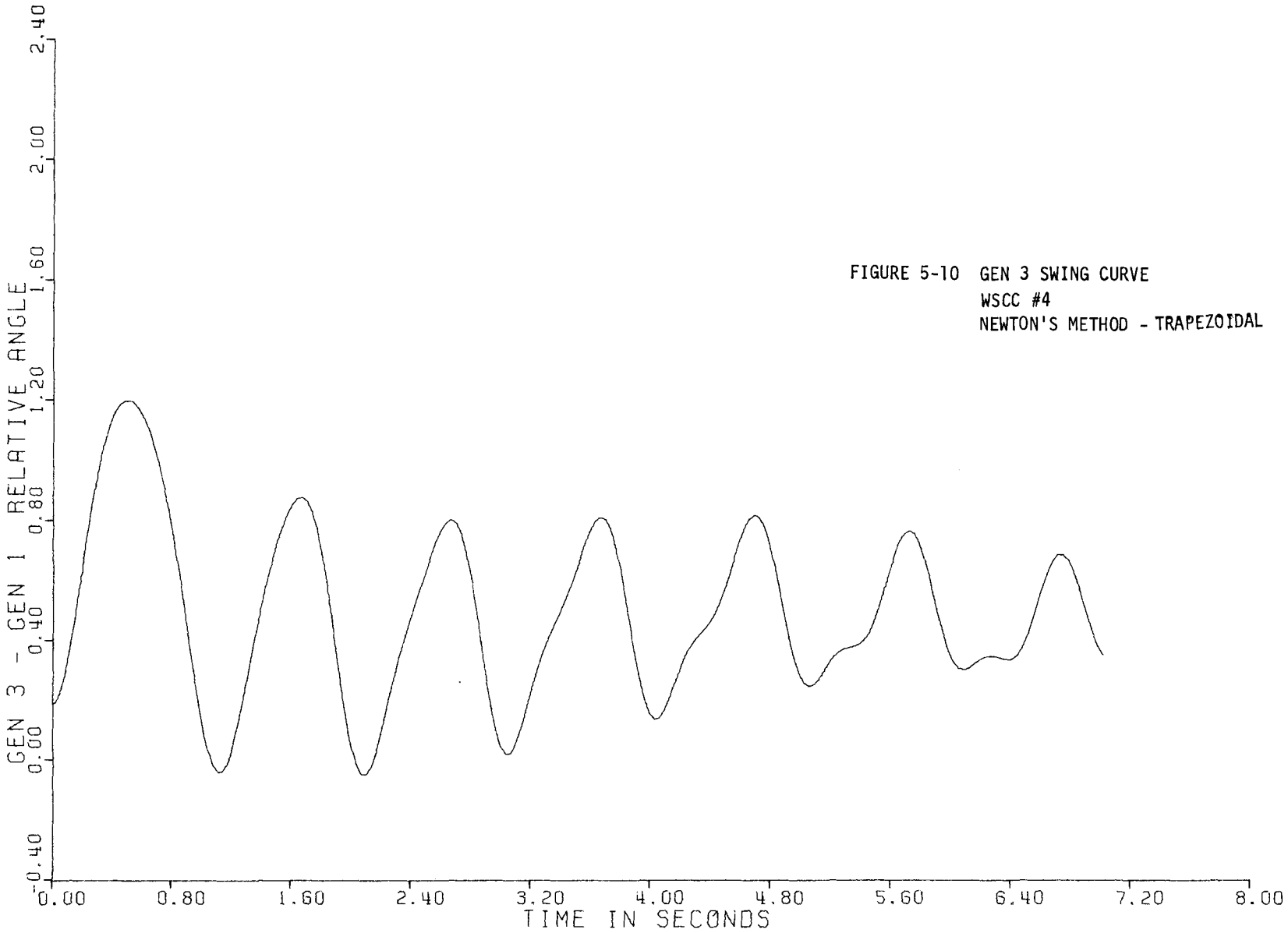


FIGURE 5-10 GEN 3 SWING CURVE
WSCC #4
NEWTON'S METHOD - TRAPEZOIDAL

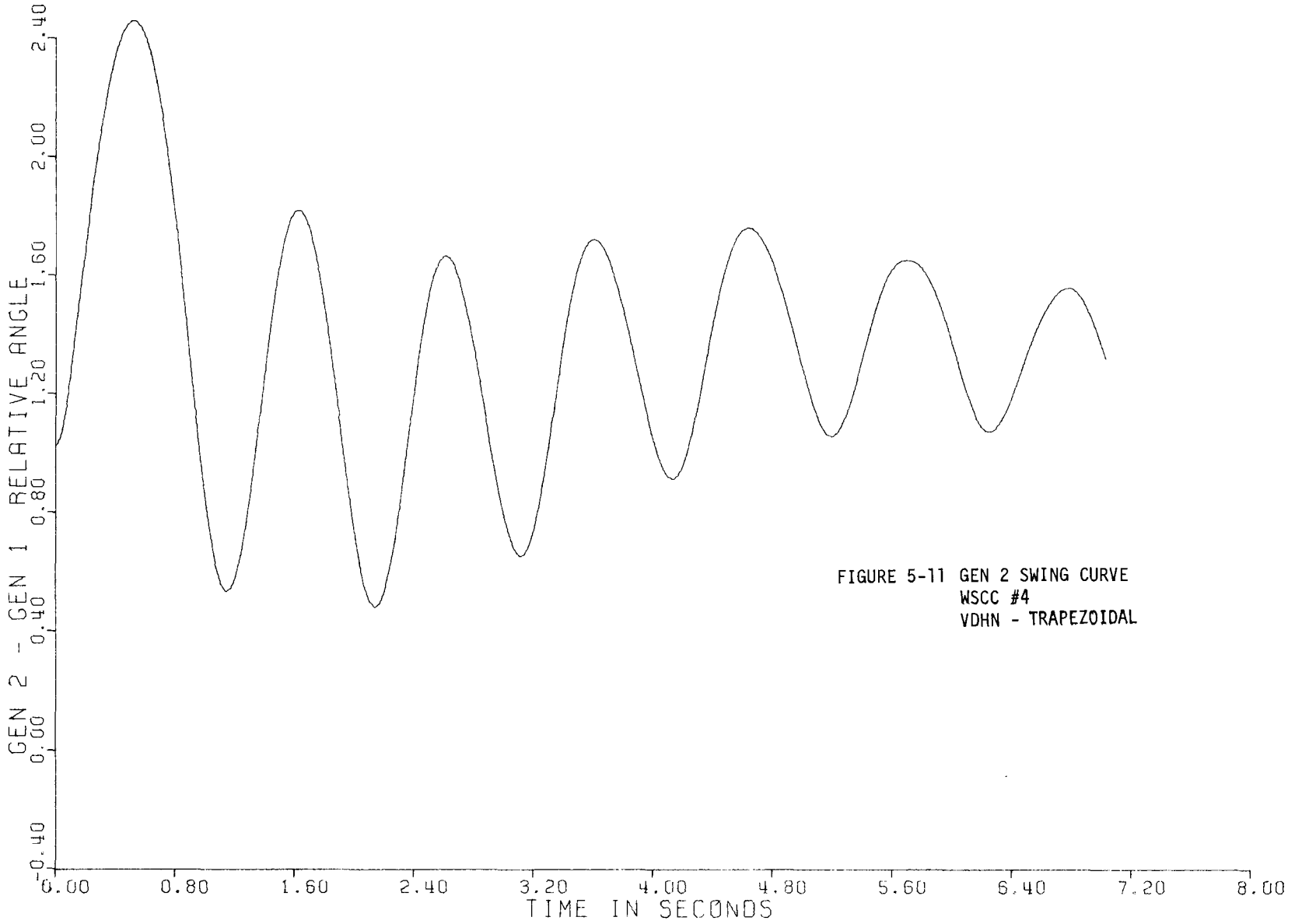


FIGURE 5-11 GEN 2 SWING CURVE
WSCC #4
VDHN - TRAPEZOIDAL

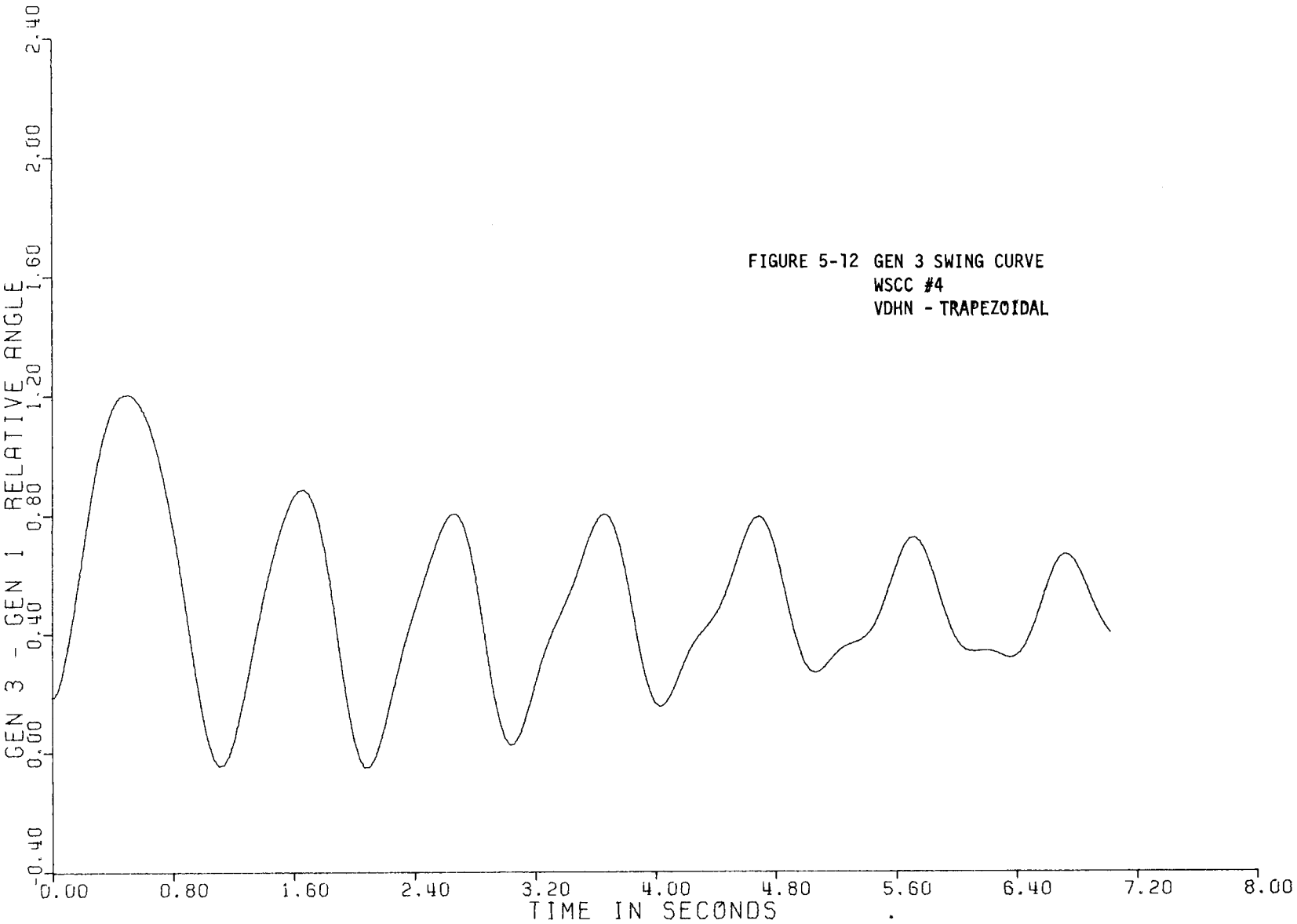


FIGURE 5-12 GEN 3 SWING CURVE
WSCC #4
VDHN - TRAPEZOIDAL

Conclusion on Iteration Techniques. The tests show that VDHN can effectively use the reduced cost per iteration to reduce total simulation cost. The slight degradation in number of iterations is not sufficient to degrade the time savings achieved through reduction of factorizations and of Jacobian evaluations. The initial implementation showed a savings of about a factor of 2. This can approach the estimated ratio between factorization and repeat solution, a factor of 3 or 4.

Neither quasi-Newton nor fixed point iteration with relaxation are considered competitive. Quasi-Newton costs about the same as Newton's method with no other gains. Fixed point iteration with relaxation requires less storage due to using a symmetric Y matrix but requires more computer time due to higher iteration counts.

5.3 PREDICTION

The discussion that follows covers the area of initial conditions for iteration. Mathematical theory for iterative convergence always assumes the initial iterate is "sufficiently close" to the desired solution to assure convergence. Transient stability programs have historically used the conditions at the previous time point as the "sufficiently close" initial guess. We will be discussing some alternatives and evaluate these alternatives based on iteration count. The interaction of prediction with stability and VDHN is discussed and some areas for refinement are pointed out.

Theoretical Background for Prediction. The accuracy of the initial conditions, $X^{(o)}$, for iteration, affect the number of iterations required to satisfactorily converge. The normal approach is to use a zeroth order prediction:

$$X^{(o)}(t_k) = X^{(o)}(t_{k-1})$$

Simple alternatives are to use low order prediction formulas based on previous information. First order (linear) prediction could be

$$X^{(o)}(t_k) = X(t_{k-1}) + \Delta X(t_{k-1})$$

whose Δ is the linear difference operator, i.e.,

$$\Delta X(t_{k-1}) = X(t_{k-1}) - X(t_{k-2})$$

Similarly, second order (quadratic) prediction would be

$$X^{(o)}(t_k) = X(t_{k-1}) + \Delta X(t_{k-1}) + \Delta^2 X(t_{k-1})$$

The second order difference Δ^2 is

$$\Delta^2 X(t_{k-1}) = \Delta X(t_{k-1}) - \Delta X(t_{k-2})$$

Preliminary analysis of second order prediction indicated it gave good estimates except for network variables under high machine acceleration. That led to the conclusion that prediction of network variables should consider the rotational nature of these variables. In particular, the prediction of the angle is more important than the magnitude. This was implemented by what is termed geometric (or logarithmic) prediction of the network variables as complex numbers.

For linear-geometric prediction, the complex network variables, V,

$$\ell n V^{(o)}(t_k) = \ell n V(t_{k-1}) + \Delta \ell n V(t_{k-1})$$

or equivalently

$$V^{(o)}(t_k) = \frac{\left[V(t_{k-1})^2\right]}{V(t_{k-2})}$$

For quadratic-geometric prediction

$$\ell n V^{(o)}(t_k) = \ell n V(t_{k-1}) + \Delta \ell n V(t_{k-1}) + \Delta^2 \ell n V(t_{k-1})$$

or equivalently

$$V^{(o)}(t_k) = \frac{\left[V(t_{k-1})^3 V(t_{k-3})\right]}{V(t_{k-2})^3}$$

In the discussion that follows, it will be assumed that the geometric forms are used for the network variables.

The prediction methods proposed are essentially explicit multistep integration formulas. By themselves, or in conjunction with normal corrector formulas, they have poor stability characteristics. When used as initial conditions for an

iterative process, it is the stability of the underlying implicit integration technique that is important. However, a word of caution is in order. As we attempt to reduce the number of iterations and/or weaken the iteration (VDHN) the stability of the resulting technique should be considered.

A possible problem with predictions is that it will produce a worse estimate than the zeroth order prediction. This could occur at a discontinuity in the system. Analysis of quadratic prediction on test data indicates that prediction improves the initial estimate more than 95% of the time. This was borne out in testing using prediction.

The BPA transient stability program uses prediction on terminal voltages and machine angle [2]. The predicted increment in terminal voltage angle is the same as the change in machine angle in previous time step, i.e., linear. This prediction is quoted as being responsible for reducing the iteration count by a factor of 2. The prediction of machine angle is a second order prediction formula based on integrating a linear predictor for speed.

Criteria for Evaluation. The criteria for evaluating prediction are total time, iteration count and resulting accuracy. Of these, average iteration count is the most graphic element. The additional storage required for quadratic prediction is recognized but not used in an evaluation.

The time element includes the cost of doing the prediction. It is shown that the reduction in iteration count offsets the cost of prediction.

Accuracy can be maintained by requiring sufficiently tight convergence criteria. The criteria used in the diagnostic program was not sufficient to detect an instability interaction between quadratic prediction and VDHN. Suggestions for controlling this are given in Conclusions on Prediction below.

Test Cases and Results. Case 4 of the WSCC nine bus series was chosen to illustrate the results of prediction. The many times that limits are reached in the stabilizer make case 4 a "worst case". The results from other cases were similar. All results refer to the use of the trapezoidal integration rule.

Table 5-4 summarizes the variation in time and iteration count for various combinations of prediction and Newton iteration. The times given are total run times for case 4 to 7 sec. Detail breakdowns of the best and worst VDHN runs are given in Table 5-5.

Table 5-4

TIME AND ITERATION SUMMARY

WSCC #4, TRAP, 7 SEC, 214 STEPS

Method	Time (sec)	Iterations	
		Total	Per Step
NEWTON			
NO PRED	65	835	3.9
LIN-GEOM	54	623	3.0
QUAD-GEOM	40	461	2.2
VDHN			
NO PRED	29	871	4.1
LIN-GEOM	23.6	639	3.0
QUAD-GEOM	23.2	551	2.6

Table 5-5

PPE TIMING, PREDICTION

WSCC #4, TRAP, VDHN, 7 SEC

Section	No PRED	QUAD-GEOM
Control - Misc.	2.31	1.67
Loads	.34	.30
Functions and Jacobians	4.15	1.80
Generate LInear Differences	1.82	1.29
Network	.69	.36
Sparse	7.90	7.48
Diagnostic Related	3.14	2.61
I/O	4.57	4.08
Utility	1.37	1.08
Prediction	0	.005

The cost of doing a prediction is several orders of magnitude cheaper than doing an iteration. It is on the order of .0001 sec per integration step for quadratic prediction. As is to be expected, the use of prediction with VDHN does not produce as high a reduction in time as the use of prediction with Newton. As the number of iterations is reduced, the number of times the factorized Jacobian is used is also reduced. The case shown held the Jacobian fixed for five integration steps or until convergence was considered slow. A review of the detailed iteration count history indicated that a more sophisticated approach should be implemented. The retention of the Jacobian should be adaptive.

Figures 5-13 through 5-18 demonstrate the accuracy that can be expected by prediction. These plots show an RMS norm measure of the accuracy of linear and quadratic prediction. Positive values represent number of digits of accuracy expected. By comparing these to Figures 5-1 and 5-2 we see that prediction is roughtly equivalent to between one and two Newton iterations. This supports the data presented in Table 5-4. The average number of iterations is reduced by about 1.5 by quadratic prediction.

The poor prediction of the machine variables at isolated points (Figures 5-13 and 5-16) corresponds to discontinuities. These discontinuities are caused by limits occurring in the stabilizer output (V_S) for case 4 (see Figure 5-19).

Figures 5-20 and 5-21 give the RMS norm measure of accuracy after one iteration for VDHN using linear and quadratic prediction, respectively. The general improvement due to the more accurate prediction can be observed.

Figures 5-21 through 5-24 trace the accuracy measure through all iterations for VDHN using quadratic prediction. Two items should be noted from these plots. First, at the early times, the slow convergence shown in Figures 5-5 through 5-8 for VDHN is greatly improved. Prediction has significantly improved the accuracy achieved. The second item is the unstable action at about 6 sec. This occurred with VDHN without prediction but is accentuated here. It points out the need for refinement of the controls of VDHN and prediction. This unstable action did not occur with linear prediction.

Figures 5-25 through 5-28 give relative swing curves for VDHN using prediction. These curves correspond to those given in Figures 5-9 and 5-10 for Newton's method.

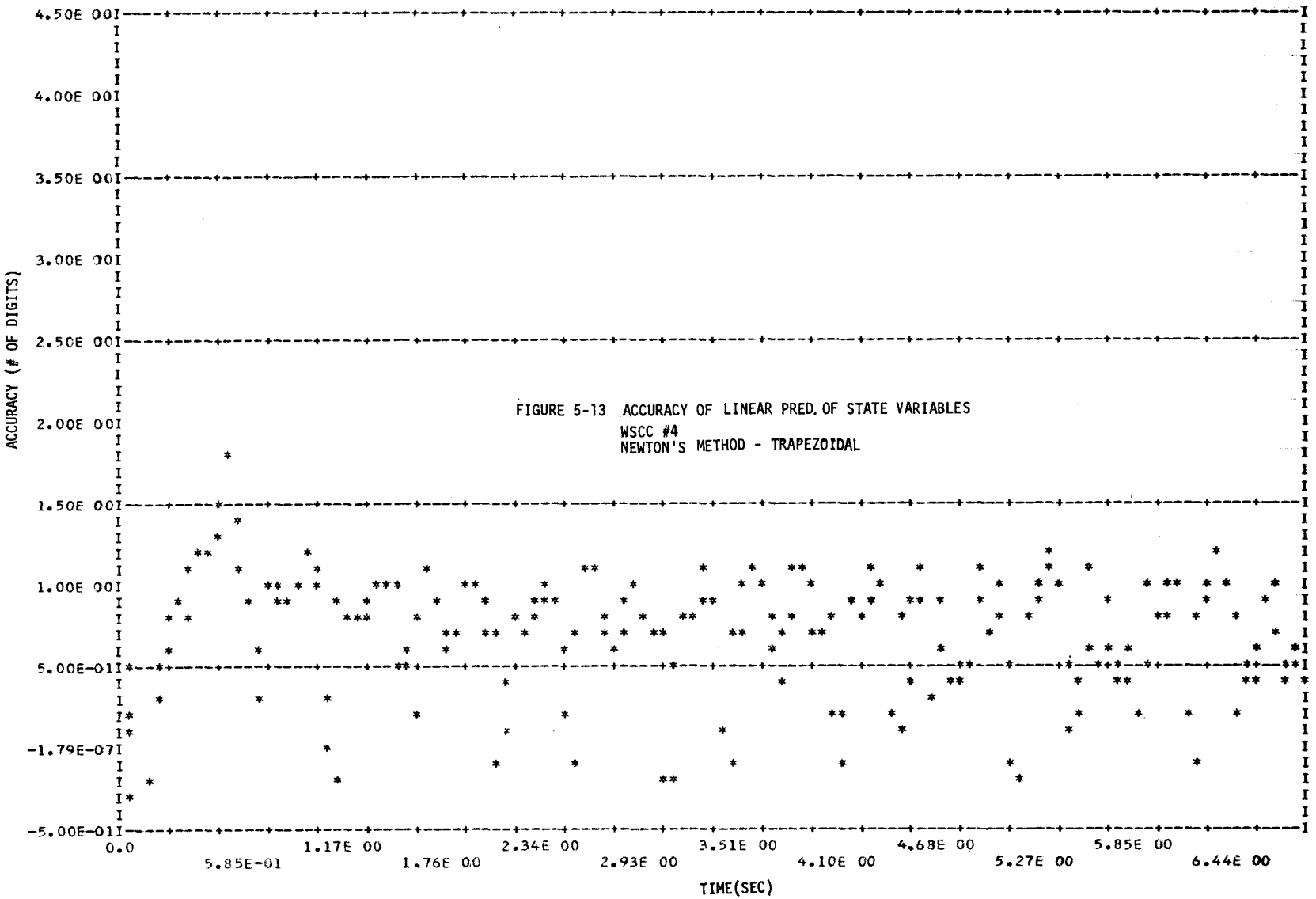


FIGURE 5-13 ACCURACY OF LINEAR PRED. OF STATE VARIABLES
WSCC #4
NEWTON'S METHOD - TRAPEZOIDAL

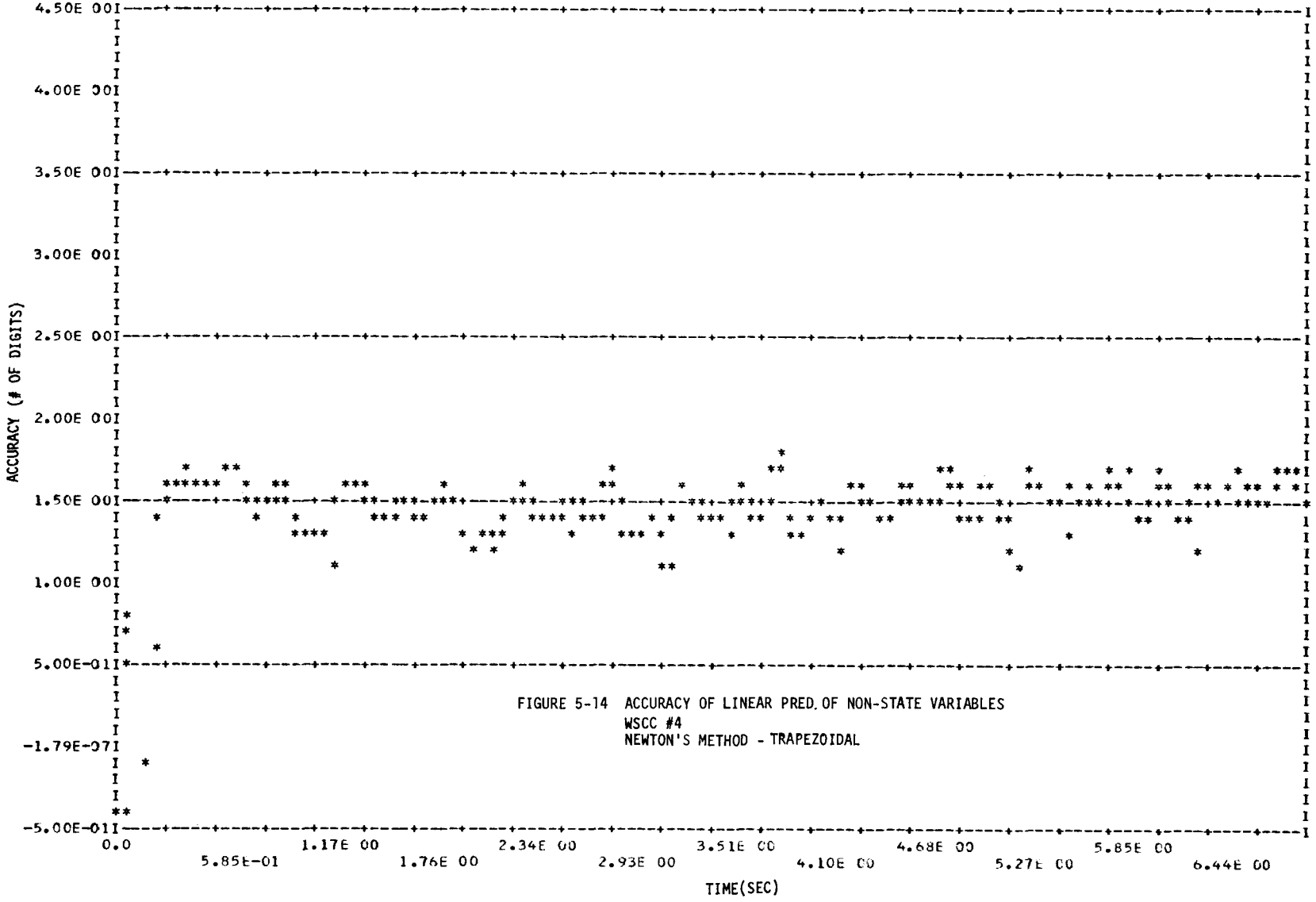


FIGURE 5-14 ACCURACY OF LINEAR PRED. OF NON-STATE VARIABLES
WSCC #4
NEWTON'S METHOD - TRAPEZOIDAL

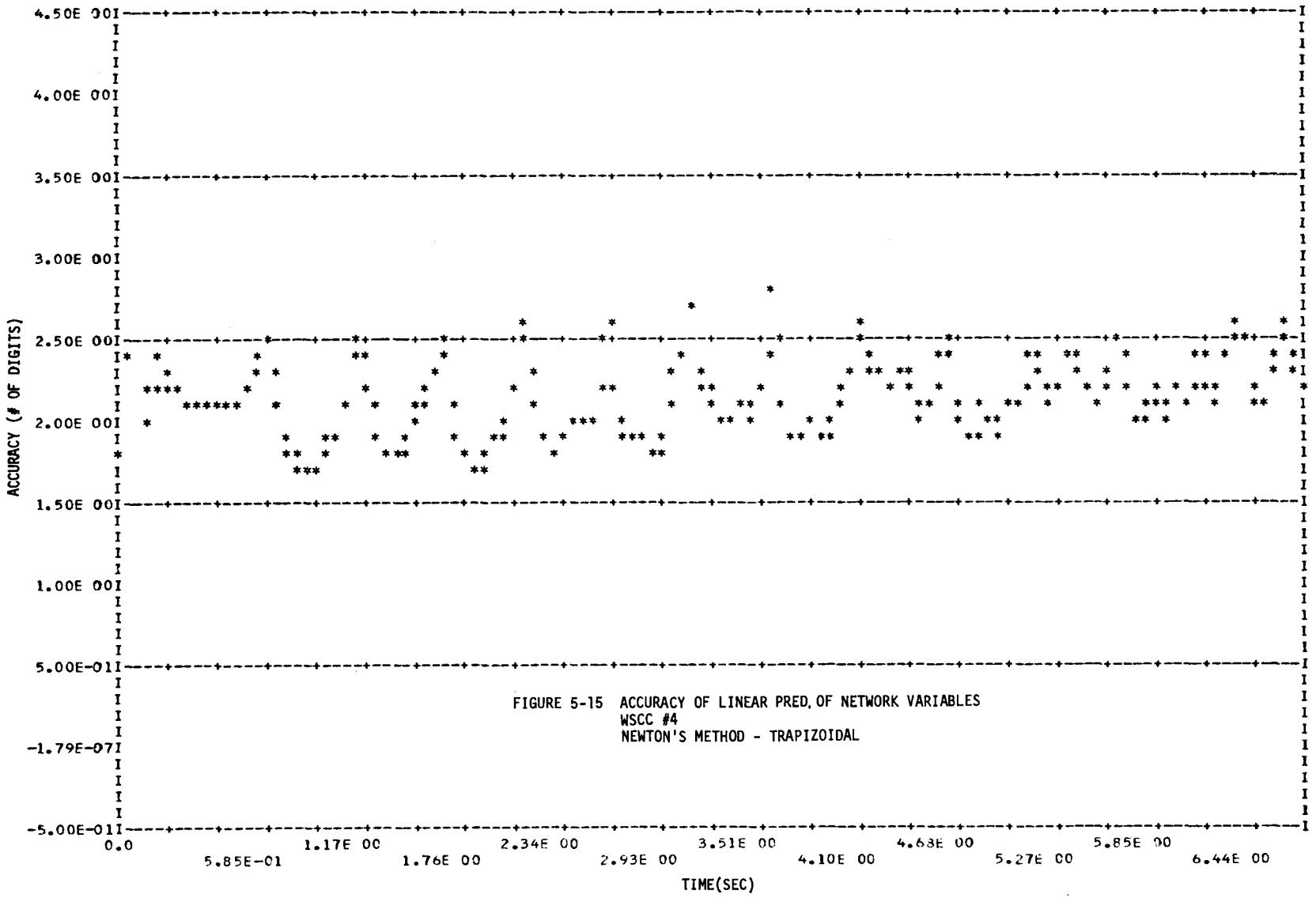


FIGURE 5-15 ACCURACY OF LINEAR PRED. OF NETWORK VARIABLES
WSCC #4
NEWTON'S METHOD - TRAPIZOIDAL

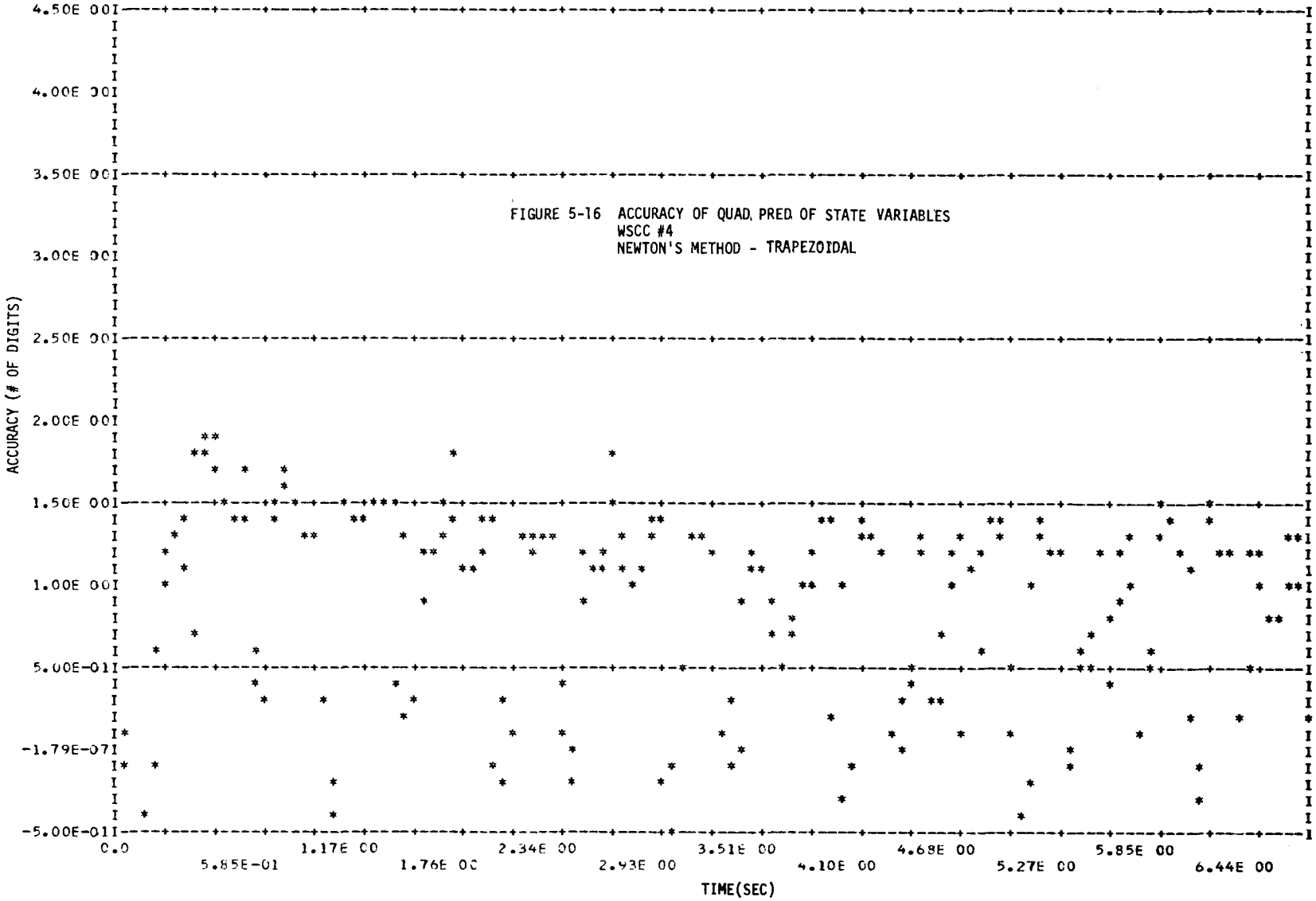


FIGURE 5-16 ACCURACY OF QUAD. PRED. OF STATE VARIABLES
WSCC #4
NEWTON'S METHOD - TRAPEZOIDAL

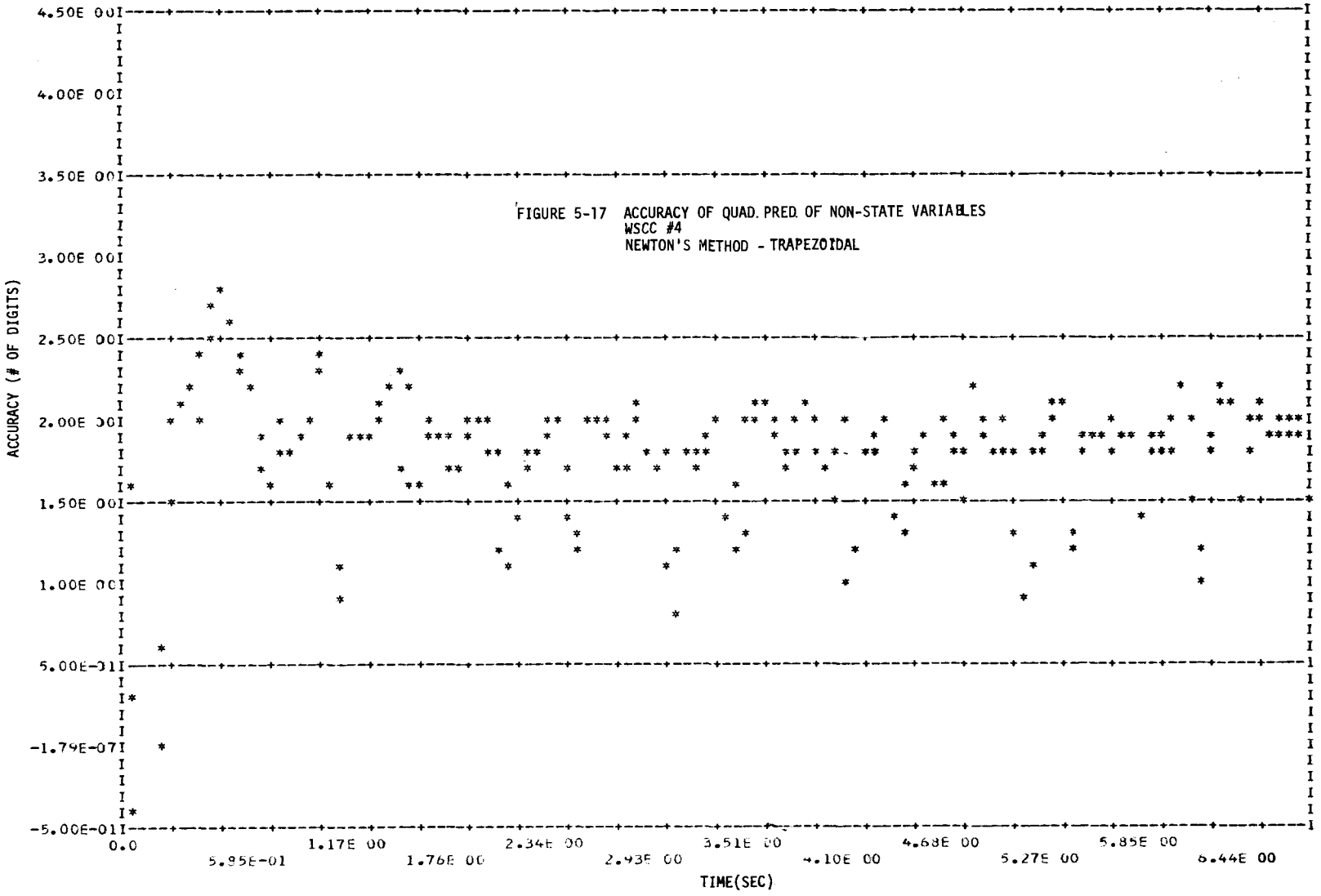


FIGURE 5-17 ACCURACY OF QUAD. PRED. OF NON-STATE VARIABLES
WSCC #4
NEWTON'S METHOD - TRAPEZOIDAL

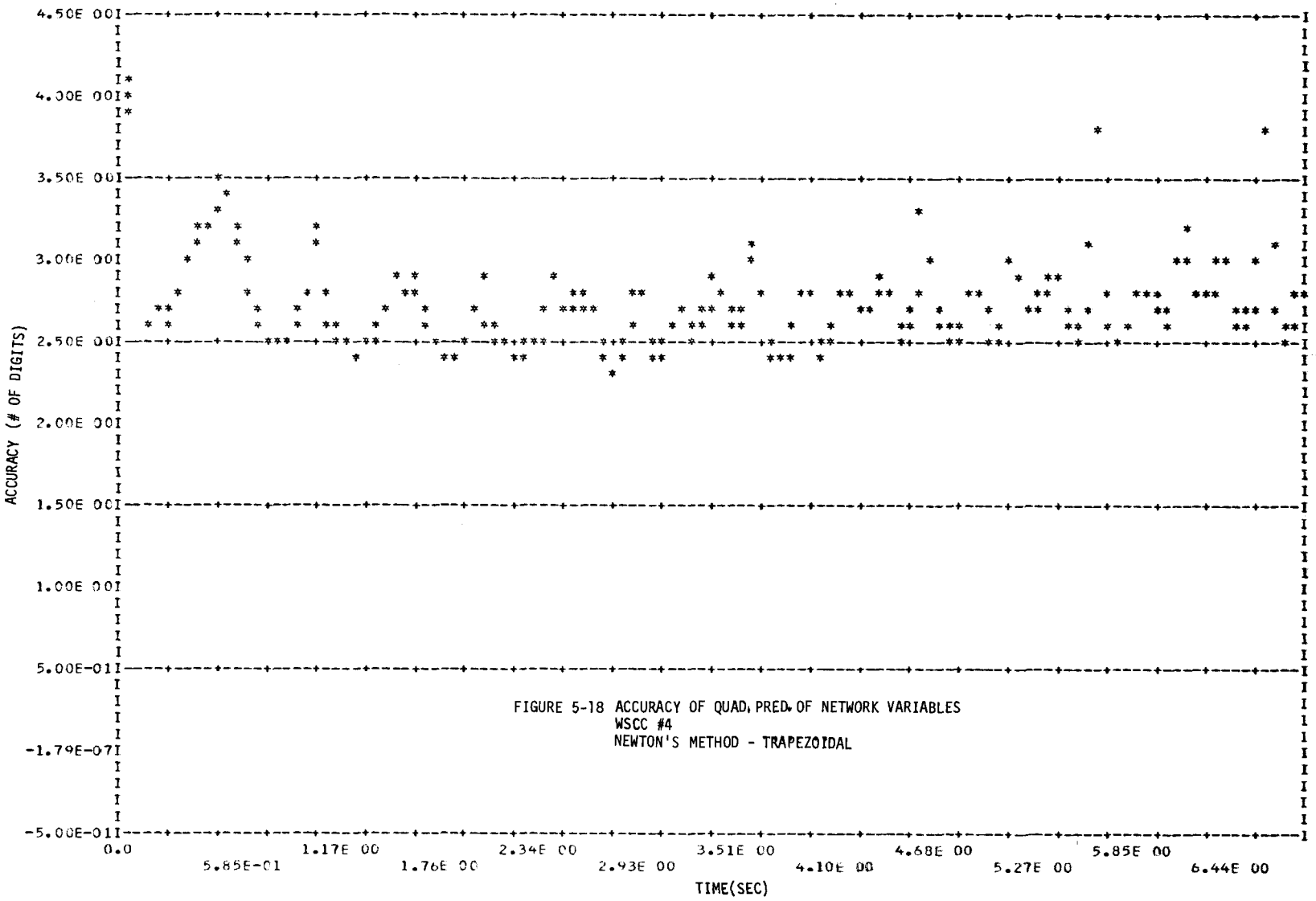


FIGURE 5-18 ACCURACY OF QUAD. PRED. OF NETWORK VARIABLES
WSCC #4
NEWTON'S METHOD - TRAPEZOIDAL

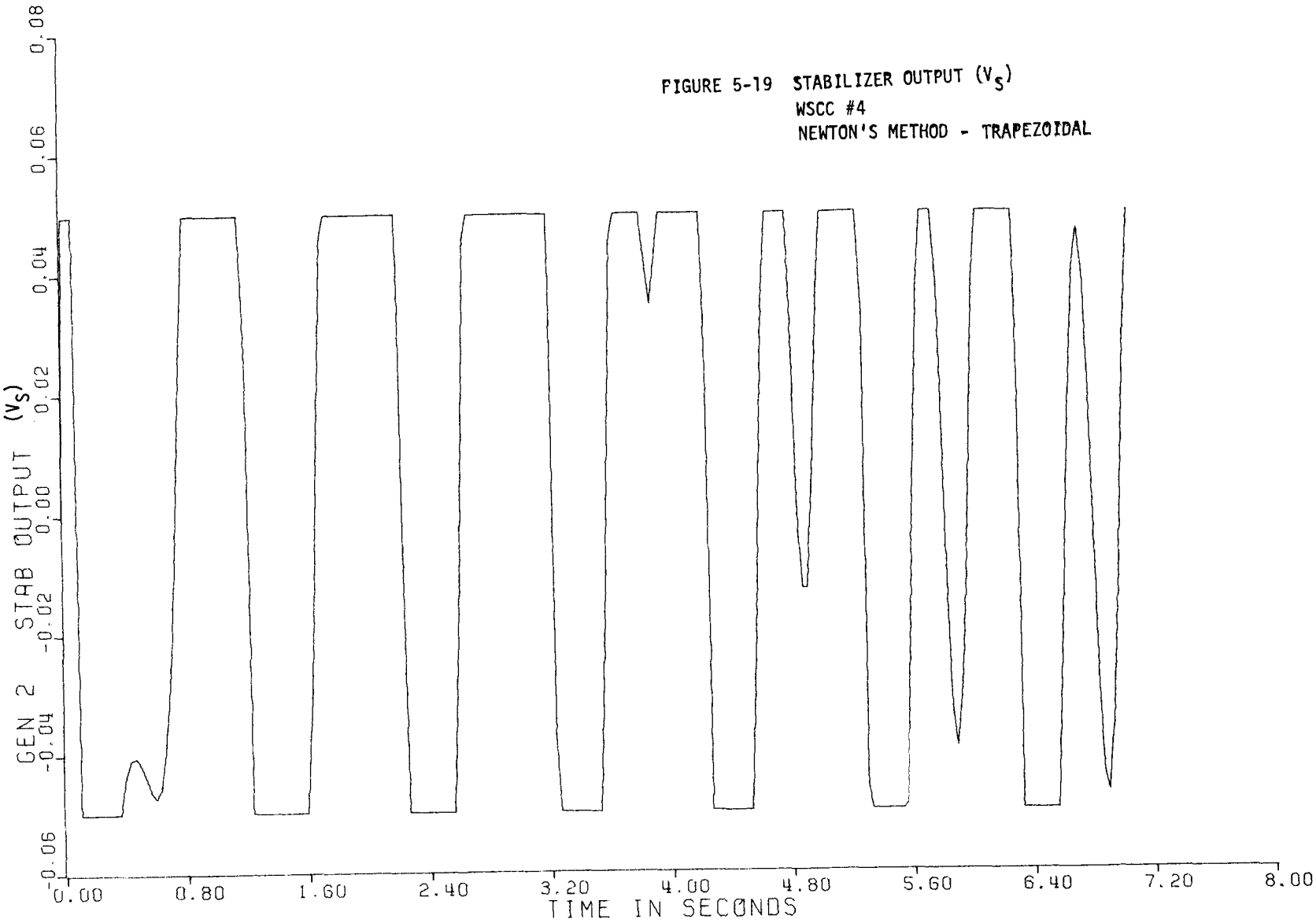


FIGURE 5-19 STABILIZER OUTPUT (V_S)
WSCC #4
NEWTON'S METHOD - TRAPEZOIDAL

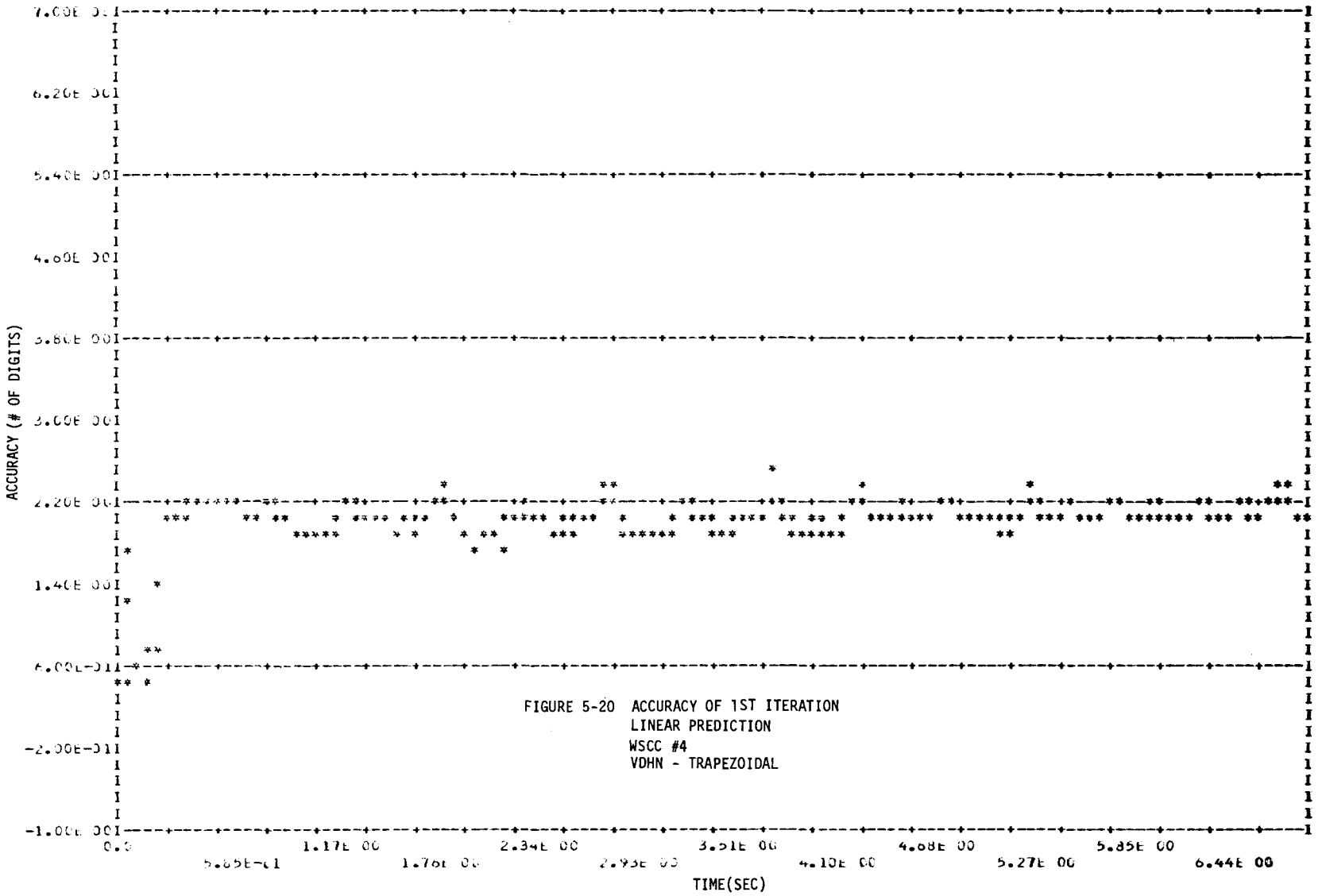


FIGURE 5-20 ACCURACY OF 1ST ITERATION
LINEAR PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

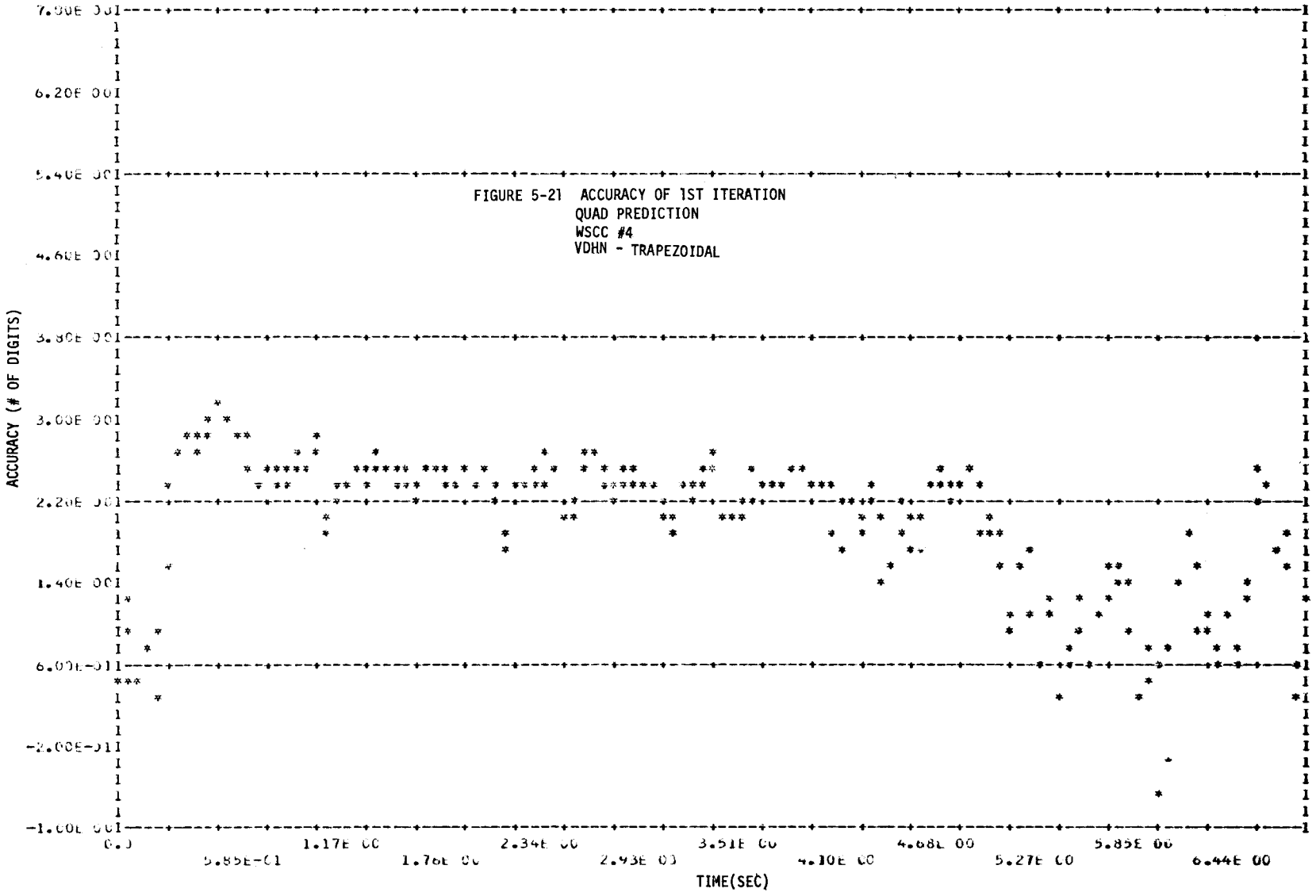


FIGURE 5-21 ACCURACY OF 1ST ITERATION
QUAD PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

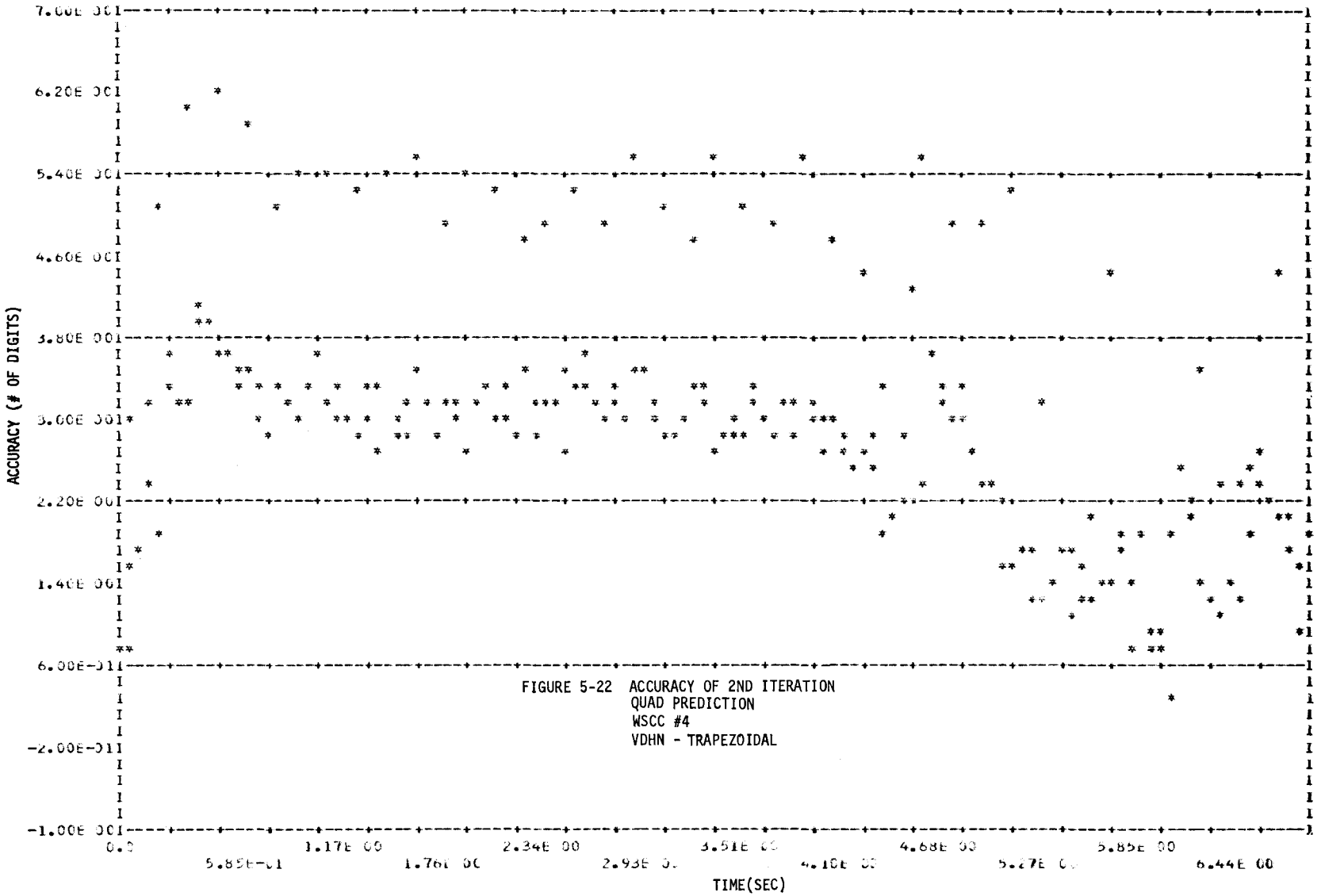


FIGURE 5-22 ACCURACY OF 2ND ITERATION
QUAD PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

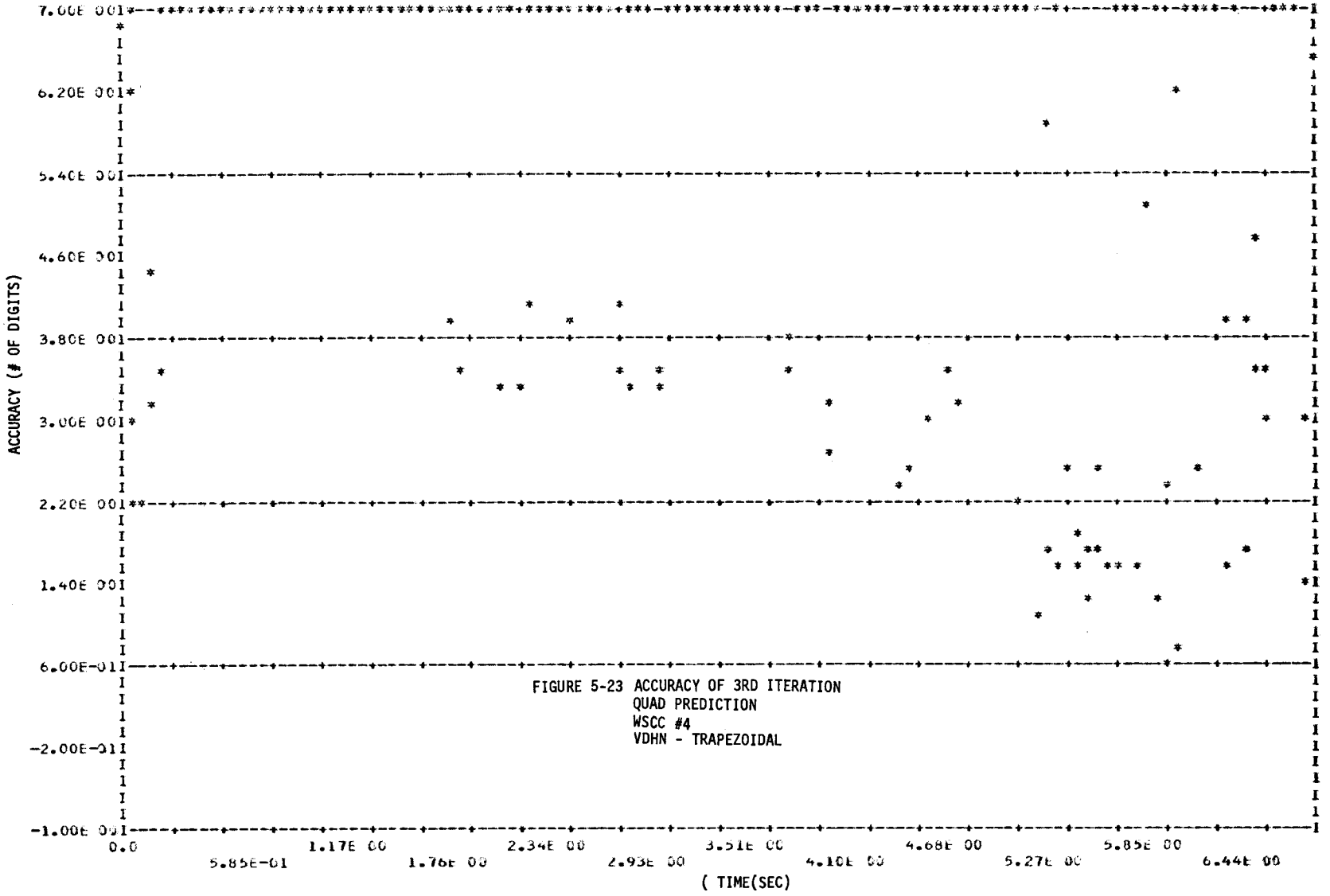


FIGURE 5-23 ACCURACY OF 3RD ITERATION
QUAD PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

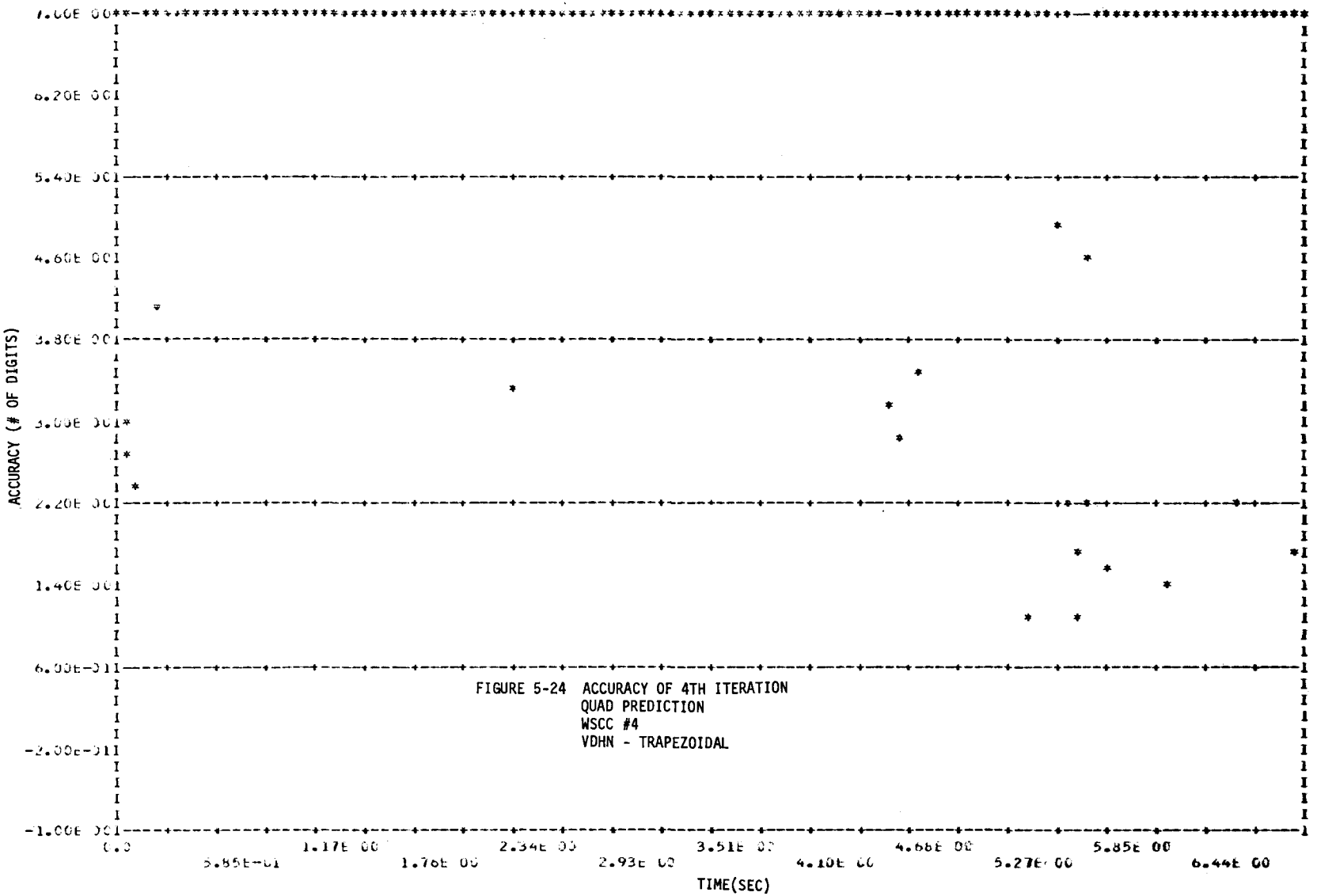


FIGURE 5-24 ACCURACY OF 4TH ITERATION
QUAD PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

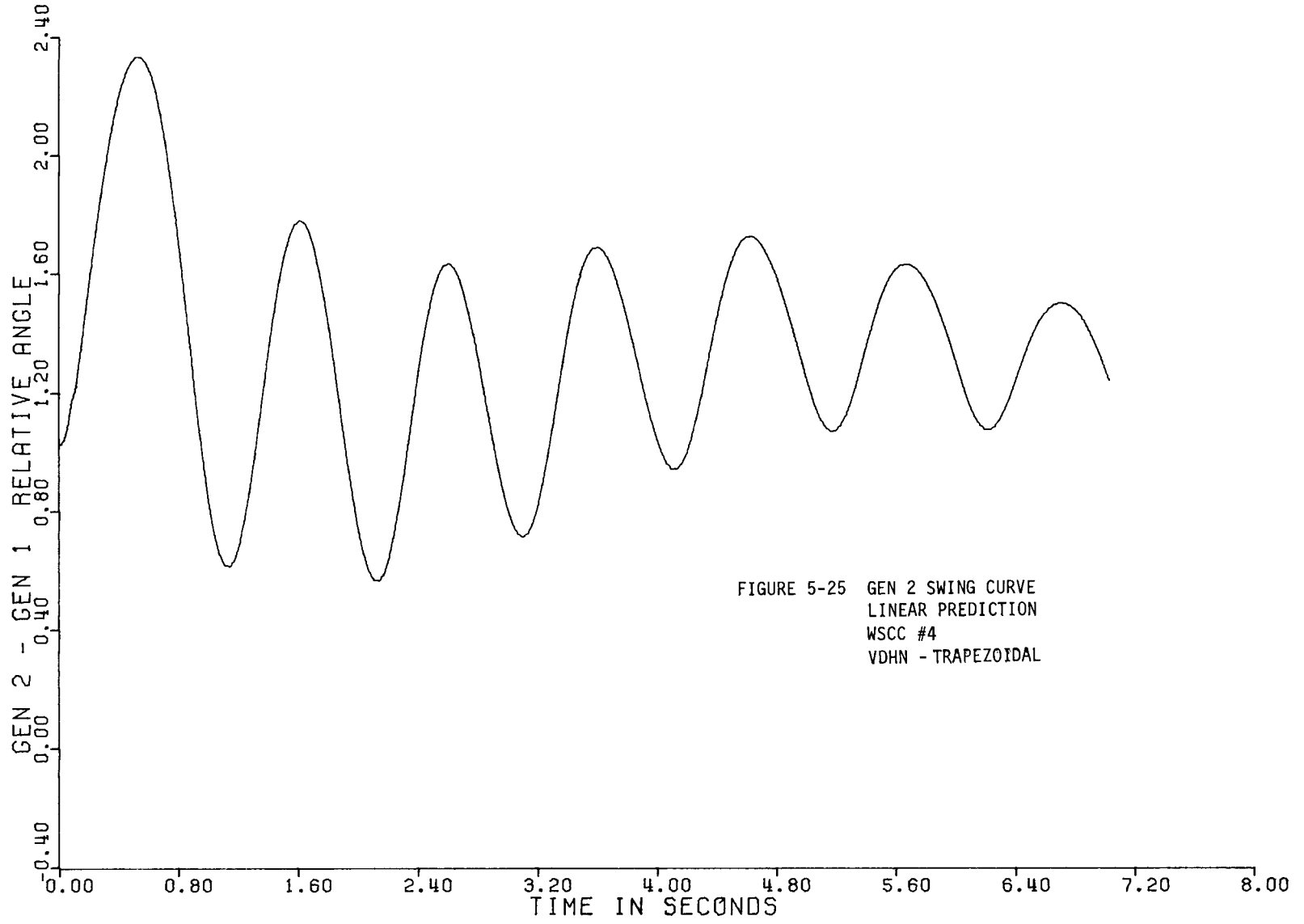


FIGURE 5-25 GEN 2 SWING CURVE
LINEAR PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

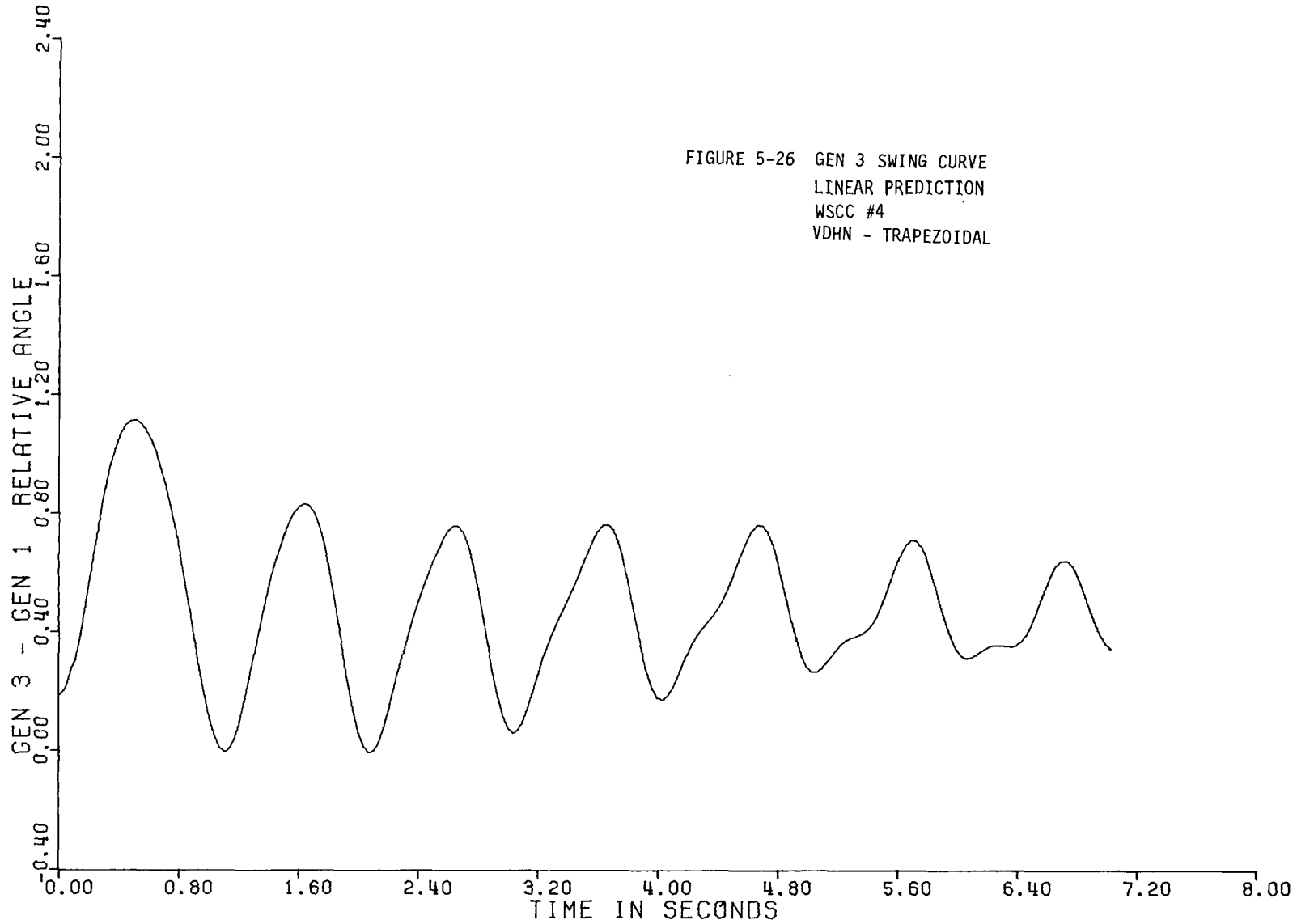


FIGURE 5-26 GEN 3 SWING CURVE
LINEAR PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

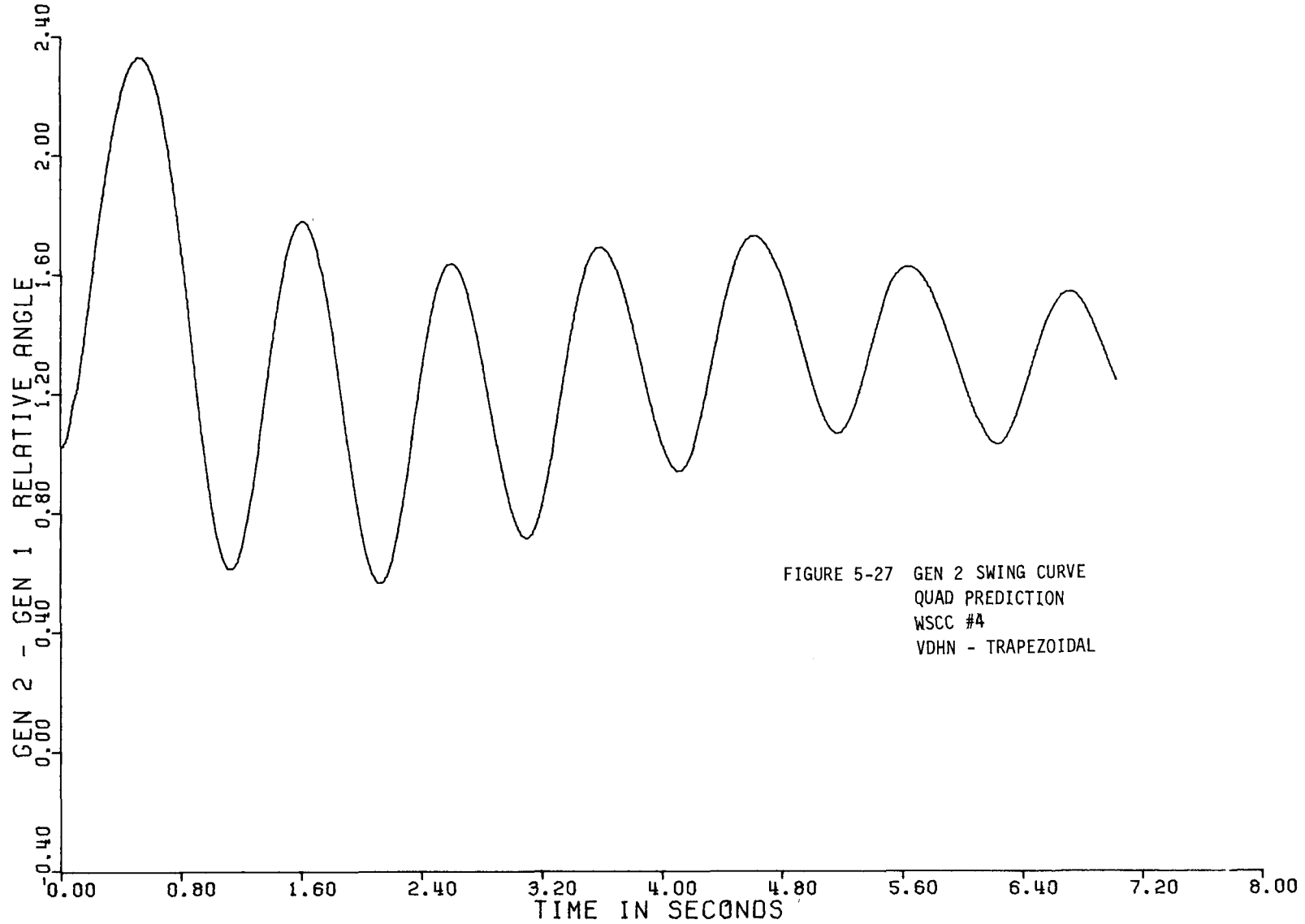


FIGURE 5-27 GEN 2 SWING CURVE
QUAD PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

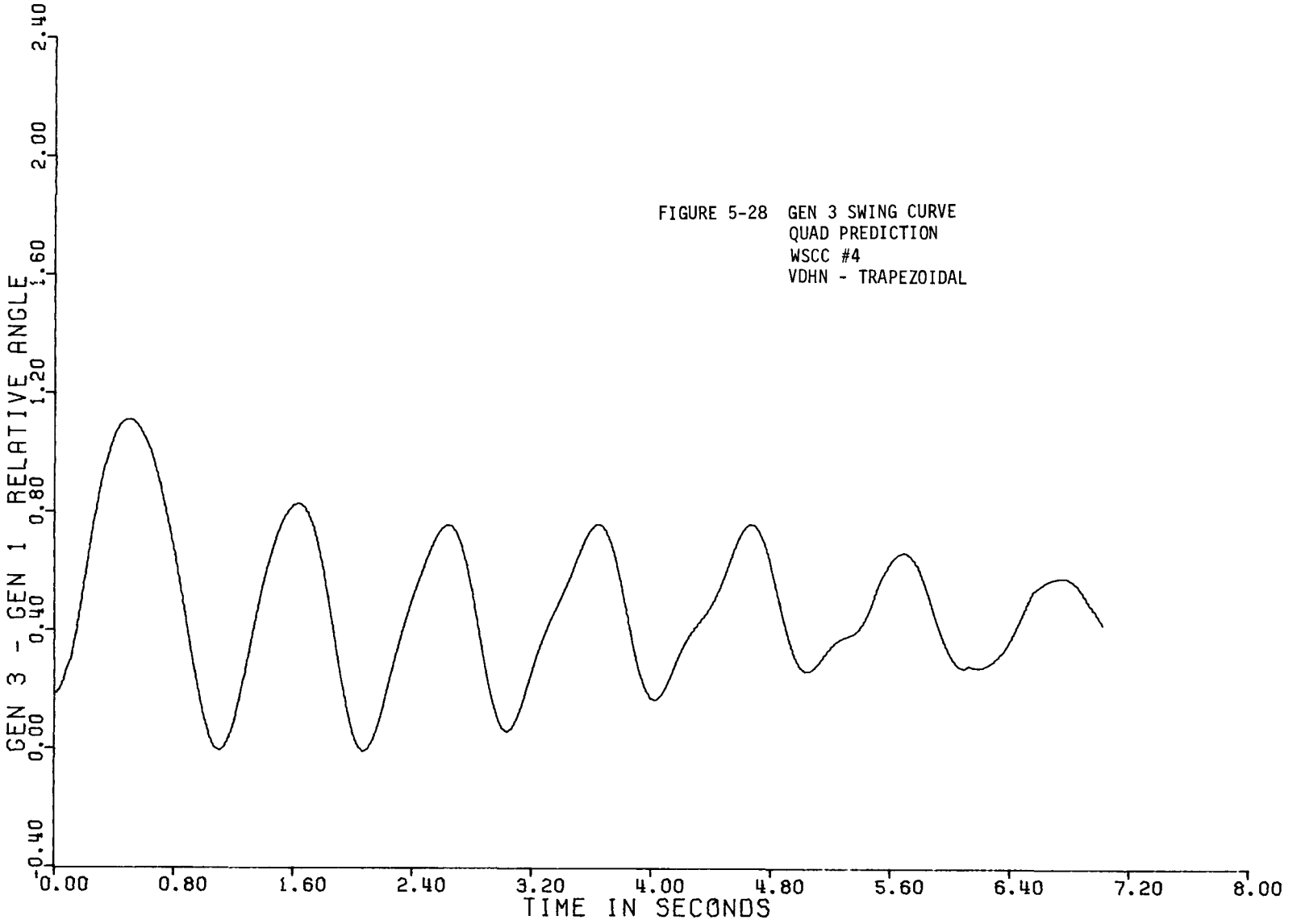


FIGURE 5-28 GEN 3 SWING CURVE
QUAD PREDICTION
WSCC #4
VDHN - TRAPEZOIDAL

Conclusion on Prediction. Prediction has been shown to significantly reduce the iteration count and/or improve accuracy. Quadratic prediction reduces the average iteration count by about 1.5 iterations. Linear prediction produces an average savings of about one iteration per step. The unstable interaction between quadratic prediction and VDHN points out the need for additional control and refinement.

The recommendation is that the use of prediction be refined. Possible refinements include using a combination of linear and quadratic prediction. For example, use linear prediction on all variables except swing angles, which use quadratic predictions. A sophisticated code might note when limiting occurs to eliminate poor predictions. Adequate controls on the interaction between prediction and the iterative technique are essential. Poor prediction does not appear to cause non-convergence. However, a production grade code should be capable of recovering, should it occur.

5.4 SUMMARY AND RECOMMENDATIONS

As mentioned in the conclusions to Sections 5.1 and 5.2 above, both VDHN and prediction reduce iteration costs. A direct net savings of a factor of 3 is shown. The simultaneous use of VDHN and prediction can produce some erratic behavior in the iteration process using simple controls.

It is recommended that use of VDHN and prediction be refined by building controls that adjust to the problem being solved. The choice of when to evaluate the Jacobian should be problem dependent. It is possible that an appropriate combination of linear and quadratic prediction (such as linear in network variables and quadratic in machine variables) will produce optimum results. In all cases, the program should recover from a bad choice

5.5 REFERENCES

[1] Bennett, J. M., Triangular Factors of Modified Matrices, Numerisch Mathemetik, 7, 217-221 (1965).

[2] Dommel, H. W. and Sato, N., Fast Transient Stability Solutions, Trans. on Power Apparatus and Systems, July/Aug., 1972, 1643-1650.

[3] Lu, P., SPARSE - Uers Manual, Document 10208-018, Boeing Computer Services, July 1976.

[4] Reid, J. K., FORTRAN Subroutines for the Solution of Sparse Systems of Non-linear Equations, Atomic Energy Research Establishment, Harwell, UK, 1972.

Appendix A

STIFFNESS AND NUMERICAL STABILITY

In this appendix the concept of stiffness for differential equations will be discussed. Where stiffness comes from and how it can be avoided will be covered. The ability of certain integration algorithms to overcome the problem of stiffness will also be discussed.

A.1 STIFFNESS AND MODELING

The phenomenon of stiffness in ordinary differential equations (ODE) has been known for some time, but only recently has it been studied and understood. Every ODE has a family of solutions, with each solution corresponding to a distinct initial value. A stiff differential equation has the property that solutions corresponding to widely spaced initial values can converge to each other very rapidly and then proceed along nearly parallel paths very close to one another. The characteristic of these equations is that they possess stable solutions, and if an initial value is selected that is not on the path of a stable solution, then the solution through that initial value will rapidly approach one of the stable solutions. The reason for this behavior is that the function f that defines the differential equation (3-1) is small along a stable solution, but grows rapidly as X moves away from the path of a stable solution. A simple example of a stiff differential equation is

$$\dot{X} = f(X,t) = -1000(X-\sin(t)) + \cos(t) \qquad \text{(A-1)}$$

The stable solution is

$$X(t) = \sin(t) \qquad \text{(A-2)}$$

Along the stable solution f(X,t) grows no larger than 1., but if X is as little as .1 off the stable solution f(X,t) grows to 100. It is this wild variation of f as X moves away from the stable solution coupled with the inability of any algorithm to stay exactly on the stable solution that causes the computational difficulties associated with stiff equations. Usually it is the case (as it is in the above

example) that stiffness is associated with a time constant that is much smaller than the time frame of interest.

Two approaches that have been used to overcome the difficulties of stiffness are to identify the differential equations with small time constants and either increase them so that they are comparable to the time frame of interest, or decrease them to zero making the equation algebraic. The first approach has the advantage that mixed differential-algebraic systems are avoided. The second has the advantage of reducing the number of differential equations and is quite accurate where the stiff time constants are very small indeed. Both approaches have the disadvantage that the stiff equations must be identified and modified before the model can be used. Furthermore, the range of usefulness of the models can be greatly limited by the time constant modification, a modification that has been carried out for reasons that are outside the domain of modeling, but based on computational efficiency.

In transient stability the latter approach has been taken with the differential equations associated with stator windings, series and shunt capacitance and inductance, etc., replaced by algebraic equations. As a result of these approximations, the transient stability models are not applicable to studying phenomena that are somewhat faster than typical transient stability responses (such as subsynchronous resonance).

The result of these modeling techniques is to produce models for long-term, mid-term, short-term, subsynchronous resonance, and electromagnetic transients that exhibit very little stiffness (it has all been removed in the modeling process). These models also have very little overlap of applicability, for example, it would be difficult to reproduce long-term results on a mid-term model. Thus separate models for long-term, mid-term, etc., are developed and maintained. As time passes, these models evolve in different directions, further reducing model compatibility.

A.2 NUMERICALLY STABLE INTEGRATION ALGORITHMS

A recent advance in the study of integration algorithms is the discovery that certain algorithms (called numerically stable) have the capability to overcome the problem of stiffness. All of the methods considered in Section 3 are numerically stable and are suitable for the solution of stiff differential equations. However, some have better stability properties than others. All of the algorithms considered in Section 3 (with the exception of the implicit midpoint rule) have recently been seriously proposed for and/or implemented in transient stability programs. It is

curious that in spite of the fact that the stiffness has been removed from the models, the recent trend has been to numerically stable algorithms. There are two reasons for this. Firstly, when small time constants are being set to zero, those slightly smaller than the time frame of interest are left unchanged. Secondly, the apparent time constants that the modeler sees are not exactly the true time constants of the system. System feedback paths can result in some faster time constants than are apparent in any single equation. The result is that the system will have a small amount of stiffness. It is this small amount of stiffness in the model that accounts for the recent trend to numerically stable integrators. It also accounts for the time step increases (a factor of 2-4 increase) that has been reported.

The point of all this is that because of the modeling techniques that have been used prior to the trend to numerically stable algorithms the full power of these algorithms is not being used (increases in step size by a factor of 1000 or more have been reported for some of these algorithms in other engineering disciplines). If an integration algorithm with the very best stability properties is selected, the inclusion of fast differential equations in models need no longer be a concern. With these algorithms, models that are valid over a much wider time range can be run efficiently.

A.3 CLASSIFICATION OF NUMERICAL STABILITY

The traditional approach to studying the numerical stability of an algorithm is to study the test equation

$$\dot{x} = \lambda x \quad \text{with } x_o = 1 \tag{A-3}$$

where λ is any complex scalar. The true solution to this equation will grow if λ lies in the right half plane, and decay if λ lies in the left half plane. The procedure for testing an algorithm is to integrate the test equation for a single step with size h. The results of this step will depend on the value of λh. The complex plane $z = \lambda h$ is divided into two regions. The first region contains those points $z = \lambda h$ where the magnitude of the result of the integration step is smaller than 1. (region of stability). The second region contains those points $z = \lambda h$ where the magnitude of the result of the integration step is larger than 1 (region of instability). The region of stability is then compared to the left half plane (where the true solution is stable).

If the region of stability contains the entire left half plane, the method is called A-stable. A-stable algorithms are very suitable for stable stiff equations, but the region of stability may contain some points z in the right half plane, and the methods will then give stable results for unstable problems.

If the region of stability is the left half plane, then the region of instability is the right half plane. In this case, the method is called symmetrically A-stable. For these methods the computed solutions will grow or decay according to whether the true solution grows or decays, independent of step size.

Certain methods (such as APS, and the Gross-Bergen method) depend on the partitioning of the system as well as the equation being solved. For this reason, two additional test problems were developed. Both of these test problems are partitioned into a differential and algebraic equation. The first equation tests behavior that is typical of control equipment

$$\dot{x} = ax + bu \qquad \text{with } x_o = 1$$
$$u = x \qquad \text{(A-4)}$$

The second equation tests harmonic behavior that is typical of the swing equation

$$\ddot{x} = -K^2 u$$
$$u = x \qquad \text{(A-5)}$$

Equation (A-4) reduces to

$$\dot{x} = (a+b)x$$

thus it is equivalent to (A-3) with $\lambda = a+b$. Equation (A-5) reduces to

$$\ddot{x} = -K^2 x$$

thus it is equivalent to (A-3) with $\lambda = \pm Kj$ $\left(j = \sqrt{-1}\right)$.

It must be reemphasized that although equations (A-4) and (A-5) are equivalent to (A-3), the results obtained from a method may differ because of the partitioning.

A.4 STABILITY OF THE TRAPEZOIDAL RULE

The difference equation for the trapezoidal rule is given in equations (3-7) and (3-8). When applied to test problem (A-3) we get

$$X_1 - X_o = (h/2)\ (\lambda(X_o + X_1))$$

or

$$X_1 = (1+\lambda h/2)/(1-\lambda h/2)$$

It can be seen that the magnitude of X_1 is greater than 1 when $z = \lambda h$ is in the right half plane, is less than 1 when $z = \lambda h$ is in the left half plane, is equal to 1 when $z = \lambda h$ is on the imaginary axis. Thus the trapezoidal rule is symmetrically A-stable.

Applying the trapezoidal rule to (A-4) we get

$$\begin{aligned} X_1 - X_o &= (h/2)\ (aX_o + bu_o + aX_1 + bu_1) \\ u_o &= X_o \\ u_1 &= X_1 \end{aligned}$$

Eliminating u we get

$$X_1 - X_o = (h/2)\ ((a+b)(X_o + X_1))$$

thus the results are equivalent to the results obtained from (A-3) with $\lambda = a+b$.

Before the trapezoidal rule can be applied to (A-5) it must be converted to a pair of first order equations

$$\begin{aligned} \dot{x} &= Ky \\ \dot{y} &= -Ku \\ u &= X \end{aligned} \qquad \text{(A-6)}$$

Applying the trapezoidal rule we get

$$\begin{aligned} X_1 - X_o &= (h/2)\ (K(y_o + y_1)) \\ y_1 - y_o &= (h/2)\ ((-K)(u_o + u_1)) \\ u_o &= X_o \\ u_1 &= X_1 \end{aligned}$$

eliminating u and writing the results in vector form we get

$$\begin{bmatrix} X \\ y \end{bmatrix}_1 - \begin{bmatrix} X \\ y \end{bmatrix}_o = (h/2) \begin{bmatrix} 0 & K \\ -K & 0 \end{bmatrix} \left\{ \begin{bmatrix} X \\ y \end{bmatrix}_o + \begin{bmatrix} X \\ y \end{bmatrix}_1 \right\}$$

thus the results are equivalent to the results obtained from (A-3) with

$$\lambda = \pm Kj$$

The results obtained by the trapezoidal rule on the two partitioned test problems are equivalent to the results obtained on the equivalent unpartitioned problem. Thus the trapezoidal rule is symmetrically A-stable on these test problems.

A.5 STABILITY OF THE APS RULE

The difference equation for the APS rule is given by equations (3-14) and (3-15).

First consider problem (A-3). Comparing this to (3-10) we get $A = \lambda$, $B = 0$. Then

$$X_1 = \exp(\lambda h)$$

Note that the W_1 and W_2 terms are missing - this is because $B = 0$.

This solution is exact. Thus the APS rule is symmetrically A-stable on test problem (A-3). Now consider problem (A-4). This problem is equivalent to (A-3), except it is partitioned. Comparing (A-4) to (3-10) we get $A = a$, $B = b$. Then

$$X_1 = \exp(ah) + W_1 u_o + W_2 u_1$$

where

$$W_1 = \left[(\exp(ah)-1)/a - (\exp(ah)-1-ah)/(a^2 h) \right] b$$

$$W_2 = \left[(\exp(ah)-1-ah)/(a^2 h) \right] b$$

and

$$u_o = X_o = 1$$

$$u_1 = X_1$$

Thus

$$X_1 = \frac{a^2 h \exp(ah) + (ah(\exp(ah)-1) - (\exp(ah)-1-ah))b}{a^2 h - (\exp(ah)-1-ah)b}$$

Now let us compute the region of stability. This region consists of those values of a and b such that the magnitude of X_1 is less than 1. It is bounded by the region where the magnitude of X_1 is 1(i.e. $X_1 = \pm 1$). To compute this region, first consider $X_1 = 1$. Then

$$a^2h \exp(ah) + (ah(\exp(ah)-1) - (\exp(ah)-1-ah))b = a^2h - (\exp(ah)-1-ah)b$$

cancelling terms and dividing by a^2h we get

$$\exp(ah) + (\exp(ah)-1)b/a = 1$$

or

$$(a+b)\ (\exp(ah)-1)/a = 0$$

Since $(\exp(ah)-1)/a \neq 0$ we must have $a+b = 0$. Recall that $\lambda = a+b$ so we get $\lambda = 0$. This is good because the true solution $X_1 = \exp(\lambda h)$ has $\lambda = 0$ on the boundary of the stability region. Next consider $X_1 = -1$. Then

$$a^2h \exp(ah) + (ah(\exp(ah)-1) - (\exp(ah)-1-ah))b = (\exp(ah)-1-ah)b-a^2h$$

Collecting terms we get

$$b(2(\exp(ah)-1-ah) - ah(\exp(ah)-1)) = a^2h \exp(ah) + a^2h$$

adding $a(2(\exp(ah)-1-ah) - ah(\exp(ah)-1))$ to both sides we get

$$(a+b)(2(\exp(ah)-1-ah) - ah(\exp(ah)-1)) = 2a \exp(ah)-2a$$

or

$$\lambda h = (a+b)h = \frac{2ah \exp(ah)-2ah}{2 \exp(ah)-2-ah-ah \exp(ah)} \qquad \text{(A-7)}$$

In Figure A-1 the region of stability for this problem as solved by the APS rule is shown. It is bounded by the curve $\lambda h = 0$ and the curve given in (A-7). Values for ah are plotted on the x axis, and values of λh are plotted on the y axis. The true solution is stable for $\lambda h<0$ (lower half plane) and unstable for $\lambda h>0$ (upper half plane). Anomalous stability regions can be seen in the second and fourth quadrants of Figure A-1. Thus the APS rule is not A-stable for this partitioning of the problem.

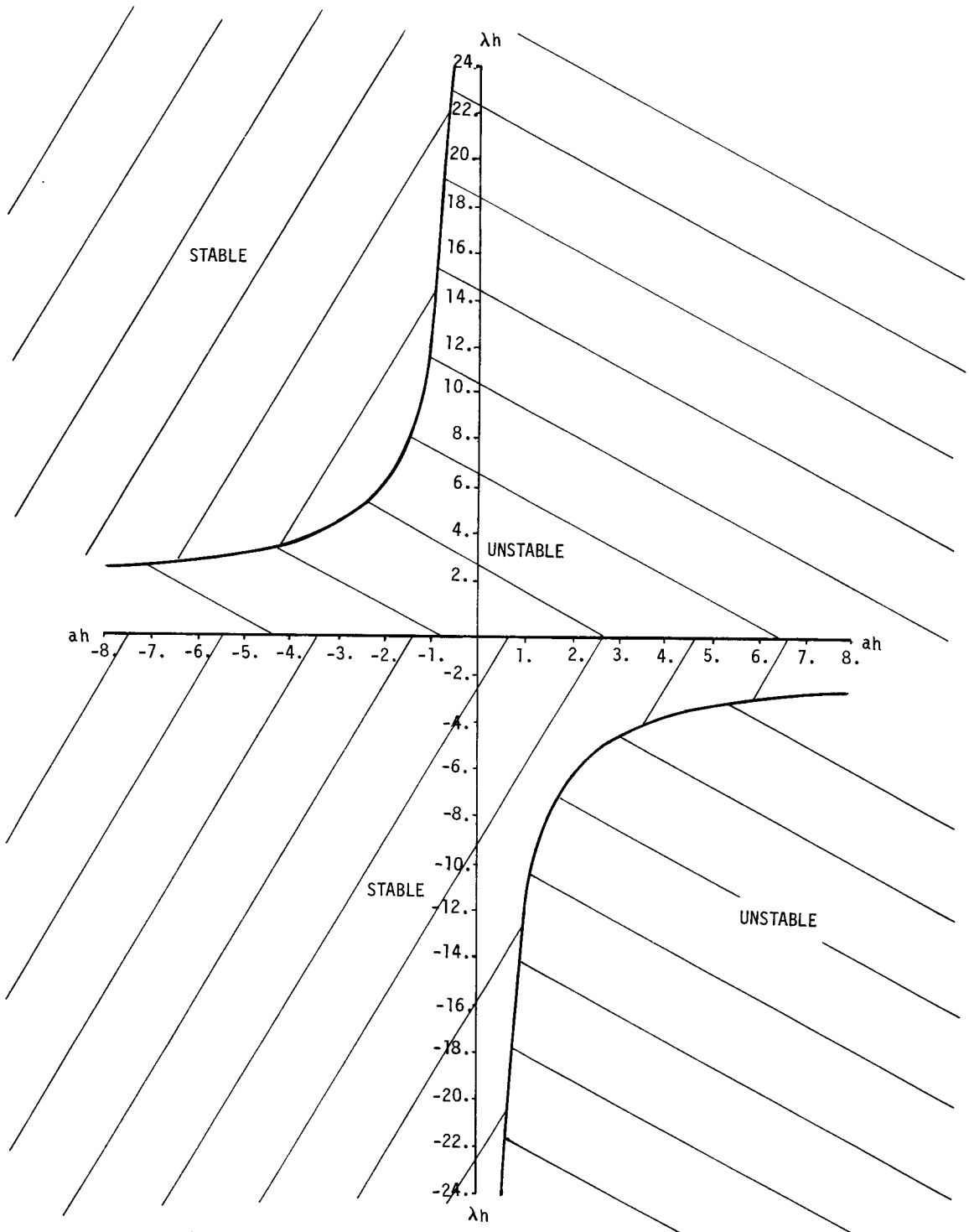


Figure A-1. Region of Stability for APS Rule on Problem (A-4)

Now consider problem (A-5). We will use the first order form of this problem (A-6). Comparing (A-6) to (3-10) we get

$$A = \begin{bmatrix} 0 & k \\ 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ -K \end{bmatrix}$$

Then

$$\begin{bmatrix} x \\ y \end{bmatrix}_1 = \left(\exp \begin{bmatrix} 0 & kh \\ 0 & 0 \end{bmatrix}\right)\begin{bmatrix} x \\ y \end{bmatrix}_o + W_1 u_o + W_2 u_1$$

Now using Taylor expansions for exp we get

$$\exp \begin{bmatrix} 0 & Kh \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & Kh \\ 0 & 1 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} h/2 & Kh^2/3 \\ 0 & h/2 \end{bmatrix} \begin{bmatrix} 0 \\ -K \end{bmatrix} = \begin{bmatrix} -K^2h^2/3 \\ -Kh/2 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} h/2 & Kh^2/6 \\ 0 & h/2 \end{bmatrix} \begin{bmatrix} 0 \\ -K \end{bmatrix} \quad \begin{bmatrix} -K^2h^2/6 \\ -Kh/2 \end{bmatrix}$$

and

$$u_o = X_o$$

$$u_1 = X_1$$

Combining terms we get

$$\begin{bmatrix} 1+K^2h^2/6 & 0 \\ Kh/2 & 1 \end{bmatrix} \begin{bmatrix} X \\ y \end{bmatrix}_1 = \begin{bmatrix} 1-K^2h^2/3 & Kh \\ -Kh/2 & 1 \end{bmatrix} \begin{bmatrix} X \\ y \end{bmatrix}_o$$

and

$$\begin{bmatrix} X \\ y \end{bmatrix}_1 = \begin{bmatrix} 1+K^2h^2/6 & 0 \\ Kh/2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1-K^2h^2/3 & Kh \\ -Kh/2 & 1 \end{bmatrix} \begin{bmatrix} X \\ y \end{bmatrix}_o$$

$$= \frac{1}{1+K^2h^2/6} \begin{bmatrix} 1-K^2h^2/3 & Kh \\ -Kh+k^3h^3/12 & 1-K^2h^2/3 \end{bmatrix} \begin{bmatrix} X \\ y \end{bmatrix}_o \qquad \text{(A-8)}$$

Clearly this algorithm will produce stable results on this problem if and only if the spectral radius (magnitude of largest eigenvalue) of the coefficient matrix in (A-8) is not greater than 1. The eigenvalues of the coefficient matrix are given by

$$\mu = \frac{-(1-K^2h^2/3) \pm \sqrt{(1-K^2h^2/3)^2 - (1+K^2h^2/6)^2}}{1+K^2h^2/6}$$

As long as K^2h^2 is not greater than 12, these eigenvalues will be complex and have magnitude 1. Thus the solution will exhibit harmonic behavior. This is appropriate for the true solution is harmonic for all values of Kh. When K^2h^2 is greater than 12, (A-8) will have an eigenvalue greater than 1 in magnitude. This represents an anomalous instability for this problem.

Thus for both test cases (A-4) and (A-5) the APS rule can start exhibiting anomalous stability properties for values of λh around 3-6. Thus this algorithm does not appear to be suitable for very stiff problems.

A.6 STABILITY OF OTHER METHODS

It was pointed out in the discussion of the implicit midpoint rule that it is equivalent to the trapezoidal rule for linear problems. All three of the test problems for stability are linear, therefore, the stability properties of the implicit midpoint rule are exactly the same on these three test problems. It would require a nonlinear test problem to distinguish the two rules.

The stability properties of the BDF have been well documented (see [1] Figures 11.6, 11.7) elsewhere. The details will not be repeated here. A brief summary of these properties is that:

- The rules of order 1 and 2 are A-stable.
- The rules of order 3 and 4 are nearly A-stable, but there are small regions in the left half plane near the imaginary axis where the rules are unstable.
- The rules of order 5 and 6 are stable along the negative real axis but have extensive regions of instability in the left half plane.
- The results obtained from all rules are independent of problem partitioning.
- The regions of stability for all rules include almost all of the right half plane.

A.7 REFERENCE

[1] C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Englewood Cliffs, N. J., Prentice-Hall, 1971.

Appendix B

MEASURING THE NONLINEARITY OF THE DIFFERENTIAL EQUATIONS

As a study tool, a quantitative measure of the nonlinearity of the differential equations in transient stability was generated. This measure was to determine if the known nonlinearities in these models were numerically significant. The answer to this question impacts the algebraic solution and choice of numerical integration technique. In particular, if the differential equations are effectively linear (numerically), then a special numerical integration technique might use this to an advantage.

The measure derived estimates a ratio of terms involving a second derivative of the state equations. This estimate is valid for the trapezoidal integration rule using Newton's method without prediction. In particular, given the differential equation

$$\dot{x} = f(x)$$

assuming the algebraic equations are satisfied, then

$$\dddot{x} = f_x^2 f + f_{xx} f^2 .$$

In this expression the term $f_x^2 f$ involves only linear terms of f while $f_{xx} f^2$ involves a nonlinear term. A measure of this nonlinear term is a measure of the nonlinearity of the differential equation.

For the trapezoidal method, the local truncation error has the asymptotic form

$$\eta \simeq \frac{-h^3}{12} \dddot{x} .$$

It can also be shown that the second Newton correction satisfies

$$x^{(2)} \simeq \frac{h^3}{4} f_{xx} f^2 .$$

The value defined by

$$NLe = \frac{1}{3}\frac{|x^{(2)}|}{|\eta|} \simeq \frac{|f_{xx}f^2|}{|f_x^2 f + f_{xx}f^2|}$$

is the measure desired. Small values of NLe will indicate that the differential equation is effectively linear. Medium ($\geq$ .5) to large values of NLe will indicate that the nonlinear terms are of significant magnitude.

The value of η was estimated by a third order polynomial method. The value of $x^{(2)}$ was taken from the iteration process.

In all test cases run, NLe ranges from quite small to moderate values. For WSCC Case #1, using only classical machines, NLe $\simeq$ .7 was consistently reached. For cases involving nonlinear elements (WSCC #4), values of NLe $\geq$ 15. were not unusual.

The conclusion of this study is that the equations are fundamentally nonlinear numerically, and no special techniques could be used to take advantage of a near linearity in the differential equations. Potential problem decomposition into fundamentally nonlinear and almost linear subsystems was not investigated.

Appendix C

THE WSCC NINE BUS TEST CASES

The Western Systems Coordinating Council (WSCC) nine bus test case series is a series of test cases designed to test each of the models in a stability program. Each test case in the series introduces one new model, so that by running the series any problem with a model can quickly be detected and isolated.

The test system consists of nine busses and three machines and is shown in Figure C-1. This system is used for all cases in the series, so that only one power flow is required to solve the entire series. Case 1 is a base case for the series. All subsequent cases involve adding or replacing a model in the base case. Table C-1 lists the stability data for the base case. Table C-2 contains a list of the individual cases in the series that were used in this research together with the additional stability data for each new case. In each case the disturbance is a three phase fault applied to the GEN 2 230KV bus at time zero. The fault is cleared and the line from the GEN 2 230KV bus to the STATION A 230KV bus is opened at time .083 seconds. New test cases are continually being added to the series as new models are added to the stability program. The data shown in Table C-1 is as of September 1975.

Table C-1

STABILITY DATA FOR WSCC CASE 1

MODEL	BUS	STABILITY DATA*
Load (Type A)	STATION A 230KV	100% constant impedance, real and reactive
Load (Type A)	STATION B 230KV	100% constant impedance, real and reactive
Load (Type A)	STATION C 230KV	100% constant impedance, real and reactive
Classical Machine	GEN 1 16.5KV	E_{MWS} = 2364., X'_d = .0608
Classical Machine	GEN 2 18.0KV	E_{MWS} = 640., X'_d = .1198
Classical Machine	GEN 3 13.8KV	E_{MWS} = 301., X'_d = .1813

*Impedance data is in per unit on a 100 MVA base. Power is in MW.

Table C-2

STABILITY DATA FOR WSCC TEST SERIES

Case	Model	Model Description	Bus	Stability Data*
1	None	(base case)		
2	WSCC MF	full machine	GEN 2 18.0KV	E_{MWS} = 640., X'_d = .1198, X'_q = .1969, X_d = .8958, X_q = .8645, T'_{do} = 6., T_{qo} = .54
3	WSCC MF	full machine	GEN 2 18.0KV	same as for case 2
	A exciter	continuously acting DC rotating excitation system	"	T_R = .06, K_A = 20., T_A = .2, T_{A1} = 0., K_E = 0., T_E = .314, $S_{.75EMAX}$ = .104, S_{EMAX} = .293, EFD_{min} = -3.984, EFD_{max} = 3.984, K_F = .063, T_F = .35
4	WSCC MF		GEN 2 18.0KV	same as for case 2
	A exciter		"	same as for case 3
	S stabilizer	shaft slip input	"	K_{QS} = .5, T_{QS} = 0., T_Q = 10., T_{Q1} = .02, T'_{Q1} = 1.3, T_{Q2} = .02, T'_{Q2} = 1.3, T_{Q3} = 0., T'_{Q3} = 0., $V_{s\ limit}$ = .05, $V_{c\ cutoff}$ = 2.0
14	WSCC MF		GEN 2 18.0KV	same as for case 2
	WSCC GH	hydro-mechanical governor turbine	"	P_{MAX} = 180., T_G = .04, T_P = 1., T_D = 5., $.5T_W$ = .5, D_D = .31, VEL_{OPEN} = .1, VEL_{CLOSE} = -.1
16	WSCC LA	load (type A)	STA B 230KV	100% constant power, real and reactive
17	WSCC LA	load (type A)	STA B 230KV	100% constant current, real and reactive

*Impedance data is in per unit on a 100 MVA base. Power is in MW. Time is in seconds.

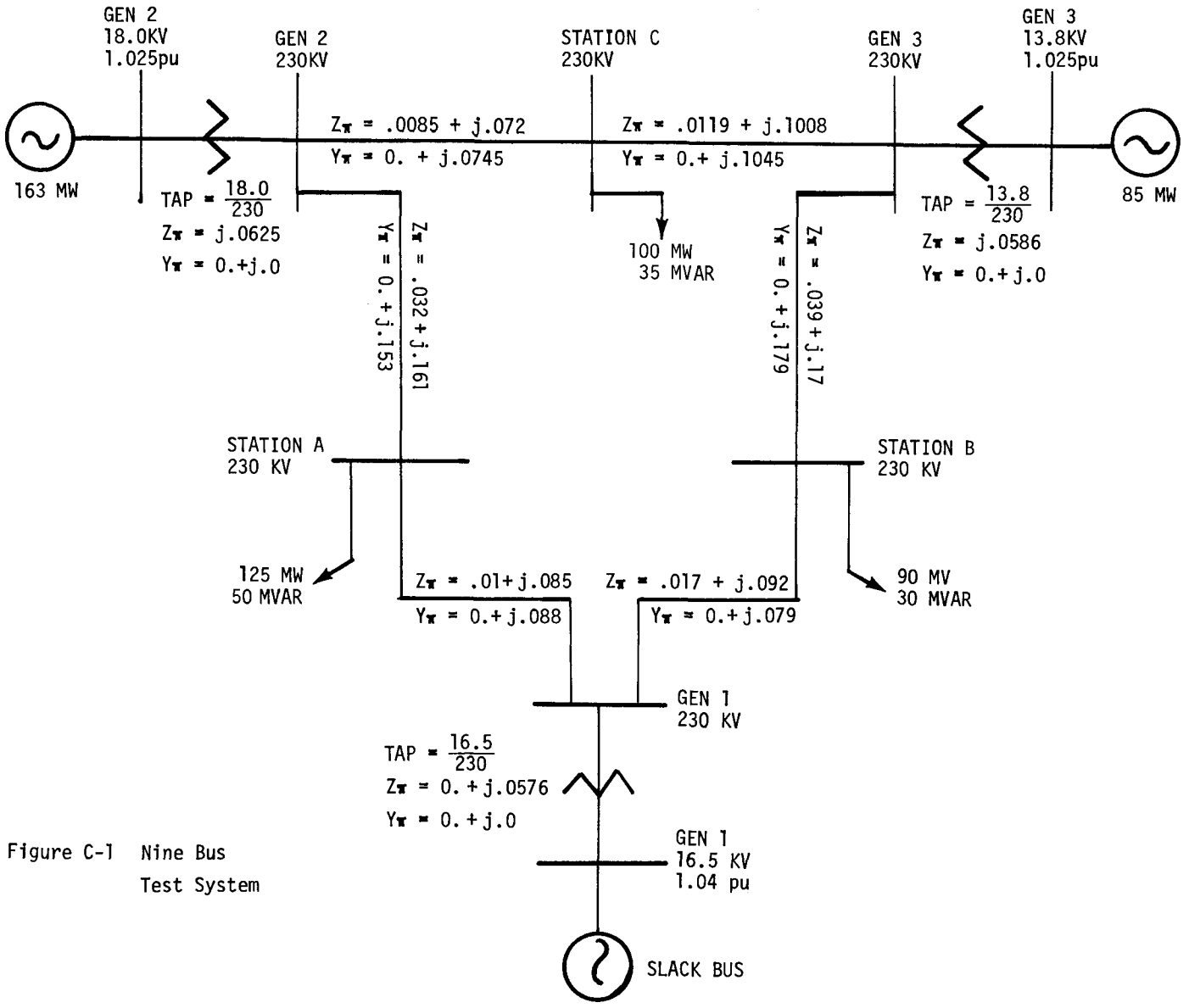


Figure C-1 Nine Bus Test System