A FINITE-STEP TRANSITION MATRIX APPROACH FOR NUMERICAL SIMULATION OF "STIFF" DYNAMIC SYSTEMS

By

WILLIAM DONALD SMITH

Bachelor of Science

University of Texas at Arlington

Arlington, Texas

1971

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1974

A FINITE-STEP TRANSITION MATRIX APPROACH FOR

NUMERICAL SIMULATION OF "STIFF"

DYNAMIC SYSTEMS

Thesis Approved:

Henry R Sebesta
Thesis Adviser

[illegible]

J E Bose

N N Durham
Dean of the Graduate College

ACKNOWLEDGMENTS

My sincere appreciation goes to Dr. H. R. Sebesta for his patient guidance, encouragement, and many kindnesses.

I would like to specially thank my parents for their support and encouragement throughout my academic work.

I would also like to express my deep gratitude to my school teacher, Mrs. Emma Roberson, whose instruction and inspiration had a profound influence on my career.

Many thanks go to faculty members and colleagues in both undergraduate and graduate school for their help and encouragement.

Special thanks go to my friends and colleagues, Lynn Ebbesen, B. N. Murali, Jerry Robinson, Mike Bowles, Ed Gostling and Syed Hamid for their help and many valuable suggestions.

Support and motivation for the work was provided by the U. S. Air Force Office of Scientific Research under Grant AFOSR 73-2458. Support was also received from the Center for Systems Science under NSF Grant GU-3160.

Finally, Mrs. Dixie Jennings deserves special thanks for her expert preparation of this manuscript.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1

II. SURVEY OF RELATED TOPICS 8

Numerical Integration Methods 8
Methods Designed to Cope With Small Time Constants. . . . 10

III. DEVELOPMENT OF TRANSITION MATRIX APPROACH. 13

The Transition Matrix 13
The Convolution Integral. 14
The Augmented System. 18

IV. IMPLEMENTING THE METHOD. 26

First-Order Subsystems. 26
Second-Order Subsystems 28
A Computer Program for Implementation on General Linear Subsystems 31
Interfacing the Transition Matrix Method With a Conventional Technique. 32

V. EVALUATING THE METHOD. 35

Analytical Subsystem Error Analysis 36
Computer Subsystem Error Study. 40
A General Procedure for Applying the Method 65
Example Problem 67
Discussion of Results 71

VI. CONCLUSIONS AND RECOMMENDATIONS. 74

Recommendations for Further Study 74

BIBLIOGRAPHY. 76

APPENDIX A - ELEMENTS OF $\Phi(\Delta t)$ FOR A SECOND-ORDER SYSTEM. 77

APPENDIX B - FACTORS AFFECTING THE RESULTS OF MACHINE COMPUTATIONS. . 81

APPENDIX C - EXACT SOLUTION FOR SECOND-ORDER SYSTEM 84

APPENDIX D - ERROR ANALYSIS PROGRAMS. 86

Chapter Page

APPENDIX E - ERROR MEASUREMENT CONSIDERATIONS 91

APPENDIX F - COMPUTATION TIME ESTIMATION. 95

LIST OF TABLES

Table Page

I. Computation Time. 96

LIST OF FIGURES

Figure Page

1. A Complex Dynamic System 5
2. A Specific Example of a Complex Dynamic System 6
3. Solution Trajectories. 9
4. The "Step" Input Approximation 15
5. The "Ramp" Input Approximation 16
6. Block Diagram of a First-Order Subsystem 27
7. Block Diagram of a Second-Order Subsystem. 29
8. A Linear Closed-Loop System. 32
9. Interfacing the Transition Matrix With a Conventional Method . . 33
10. Flow Chart for Interfacing Transition Matrix Method With RK-4. . 34
11. Analytical Error Notation. 38
12. Variation of Error With Time Step for First-Order Subsystems . . 42
13. Variation of Error With Time Step for Second-Order Subsystems. . 44
14. Variation of Error With Input Frequency for First-Order Subsystems 46
15. Variation of Error With Input Frequency for Second-Order Subsystems 47
16. Error for First-Order Subsystems 49
17. Error for Second-Order Subsystems. 50
18. Constant-Error Plot for First-Order Subsystems 52
19. Constant-Error Plot for Second-Order Subsystems. 54
20. Equal-Error Plot for First-Order Subsystems. 55
21. Equal-Error Plot for Second-Order Subsystems 57

Figure Page

22. Equal-Cost Plot for First-Order Subsystems 59

23. Equal-Cost Plot for Second-Order Subsystems. 60

24. Steps/Cycle to Achieve a Given Accuracy for First-Order Subsystems 61

25. Steps/Cycle to Achieve a Given Accuracy for Second-Order Subsystems 62

26. Block Diagram Example. 66

27. An Equivalent Block Diagram. 66

28. System for Example Problem 68

29. Pole-Zero Map for Linear Subsystem B 70

30. Error Measurement. 92

CHAPTER I

INTRODUCTION

Modeling a dynamic system generally results in certain algebraic and differential equations which describe the physical phenomena. These equations may be linear or nonlinear. As the complexity of the system being modeled increases, the resulting equations become more numerous and of higher order.

Until recently, many systems could not be simulated accurately because the resulting set of equations was too complex. The modern digital computer, with its large storage capacity and high computation speed, has made possible the simulation of such systems. Consequently, there has been increasing interest in both the modeling and simulation of complex systems in recent years. Considerable effort is being spent developing numerical techniques to solve systems of algebraic and differential equations. Older techniques which were previously impractical for hand calculations are being revived for use with the computer. New methods are also being developed. However, despite the capabilities of present machines and the availability of a wide range of numerical methods, some problems are still quite difficult to solve with the computer.

Many dynamic systems are so complex that the large size of the resulting set of algebraic and differential equations begins to tax the storage and speed performance of even the largest computers. Special analysis techniques are being developed for such systems (3).

In contrast, some systems that are relatively simple in nature result in sets of equations that require an inordinate amount of computation time to solve with conventional numerical methods. The inefficient solution of these equations drastically increases the simulation cost, which can become so high the information obtained is not worth the expense.

An important type of system which leads to an inefficient numerical solution is one which contains both large and small time constants. Such a system is called "stiff". As the difference between the time constants increases, the stiffness increases. Alternately, stiffness can be considered a measure of the distance between the system poles in the complex plane. The wider the pole separation, the stiffer the system is.

Stiff systems are described by stiff differential equations. A differential equation can exhibit stiff behavior from two sources: the eigenvalues and the input. The following is a differential equation which is stiff due to its eigenvalues:

$$\frac{d^2y}{dt^2} + 1001\frac{dy}{dt} + 1000y = u(t) \quad .$$

This differential equation has two real eigenvalues, -1 and -1000. The two corresponding time constants are 1.0 and 0.001 seconds, respectively. A conventional numerical integration technique would require a step size from 1/10 to 1/100 of the smallest time constant. In this case a step size from 0.0001 to 0.00001 would be required. Such a system, though relatively simple, leads to a very inefficient numerical solution.

The following differential equation exhibits stiff behavior due to the nature of the input:

$$\dot{x} = -x + 0.02\cos(100t) + \sin(t) \quad .$$

The solution of this equation is

$$x = e^{-t} + 1.9998 \times 10^{-6} \cos(100t) + 1.9998 \times 10^{-4} \sin(100t) + 0.5 \cos(t) + 0.5 \sin(t) \quad .$$

The high-frequency portion of the response is very small in magnitude compared with the fundamental response terms e^{-t}, 0.5 cos(t) and 0.5 sin(t). However, this high-frequency response has a period of $2\pi/100$, requiring a step size from $2\pi/1000$ to $2\pi/10000$, or about 0.006 to 0.0006 seconds. This is a very small step size considering the low-frequency of the fundamental response.

Stiff systems occur frequently in nature. Many physical systems have responses which are basically low frequency with high-frequency responses superimposed. The high-frequency response may not be as significant as the low-frequency response in determining the system behavior. However, since the time step used in a conventional numerical integration must be chosen as a fraction of the period of oscillation of the highest natural mode in the system, these high-frequency components cause the solution to be very time consuming.

Since the high-frequency responses may not be needed or even desired in the simulation, a solution to the problem would be to eliminate the sources of the high frequencies from the model. However, in a complex model, this may be quite difficult. Practial simulations frequently contain nonlinearities, complex loop interactions, and sets of linear differential equations describing the dynamics of particular parts of the system. These sets of linear differential equations can often be separated and considered as linear subsystems. When such linear subsystems are stiff, it should be advantageous to propagate their solutions analytically. These solutions could then be interfaced with the solution for the rest of the system, obtained by conventional numerical techniques.

Figure 1 shows a schematic diagram of the situation frequently encountered in complex simulations. It shows a system which is nonlinear and stiff. However, the stiffness originates in certain stiff linear subsystems, which can be identified and separated. Cross-coupling between various parts of the system, nonlinear feedback, control loop interactions and sampling devices may be present, complicating the computation of a solution. In particular, nonlinearities and sampling devices aggravate the numerical problems caused by small time constants since they intermittently cause sudden changes in the inputs to the stiff linear subsystems.

A specific example of a system of this type is shown in Figure 2. Linear subsystems A, B and C are all stiff. Nonlinearities, non-stiff linear subsystems and loop interactions are present. The technique investigated in this study is intended for such systems.

The purpose of this thesis is to investigate a method for simulating systems of the type shown in Figures 1 and 2, and establish guidelines and procedures for the use of the method. The method propagates the solutions of the stiff linear subsystems analytically, and interfaces their solutions with a conventional numerical technique for simulating the remainder of the system. The analytical solutions of the stiff linear subsystems are obtained stepwise by approximating the input to each subsystem as a series of step inputs. For each subsystem an augmented system is formed which is unforced. The state transition matrix for each augmented system is then used to propagate the solutions of the stiff linear subsystems forward in time.

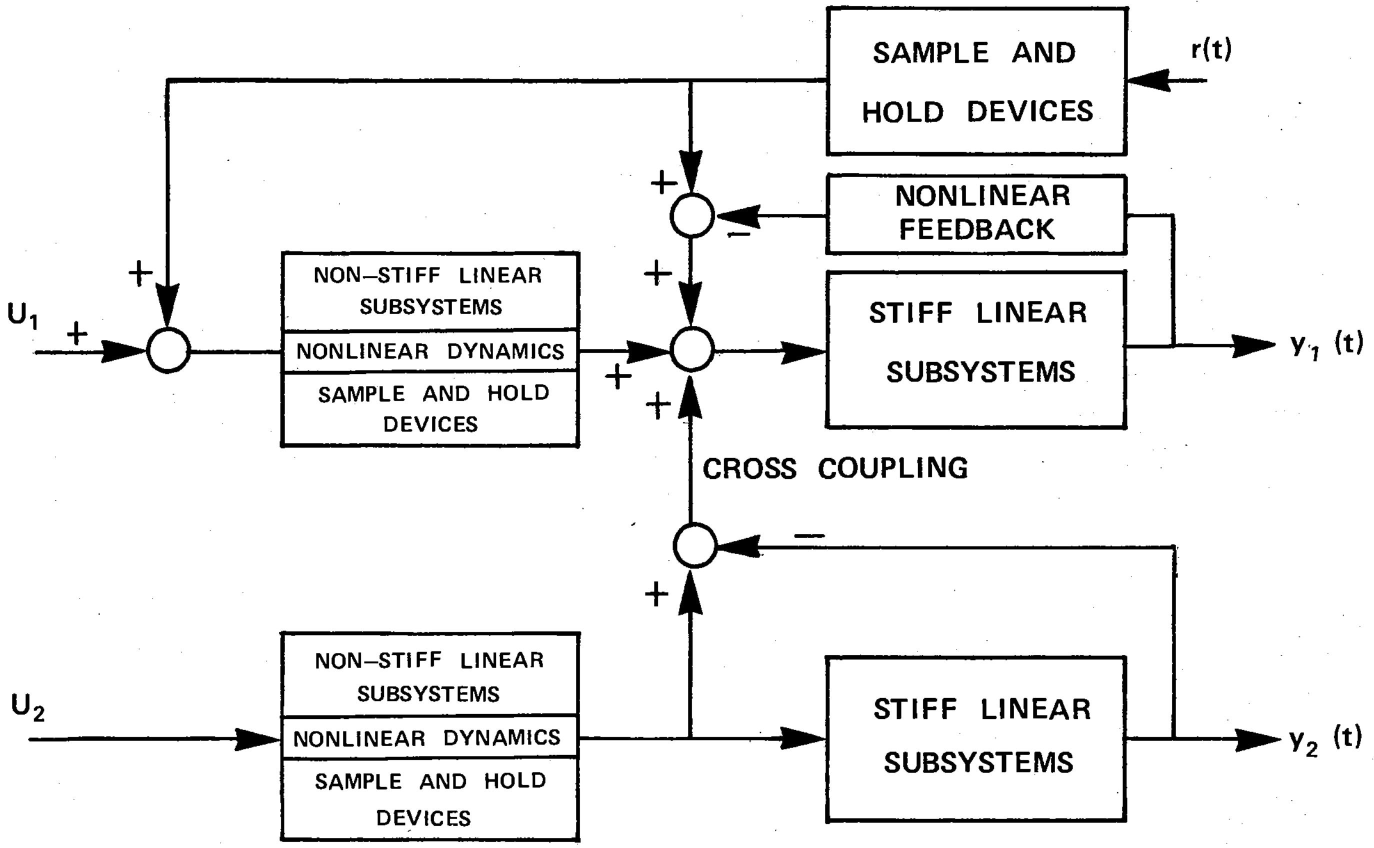

Figure 1. A Complex Dynamic System

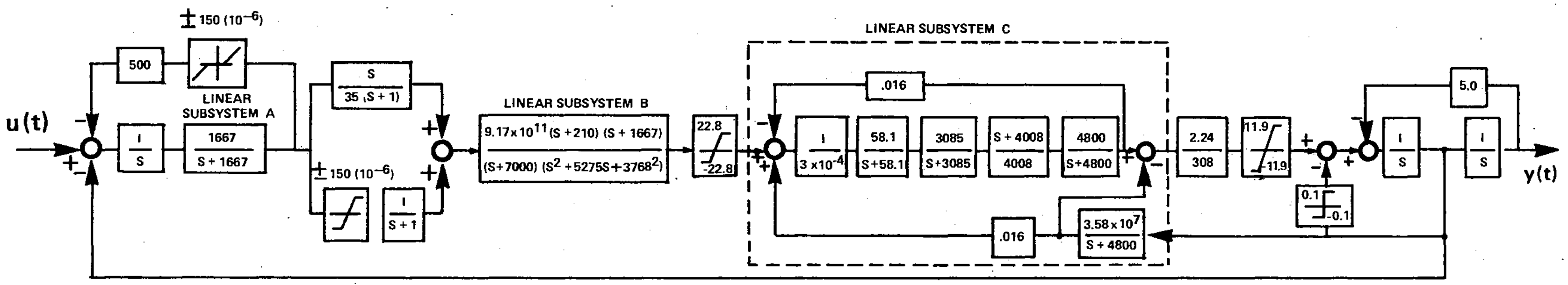

Figure 2. A Specific Example of a Complex Dynamic System

Chapter II presents a short discussion of numerical methods in general and some methods for solving stiff systems found in the literature. Chapter III develops the transition matrix approach for simulating stiff linear subsystems. Chapter IV implements this approach on general first and second-order subsystems, references a computer program useful in implementing the approach on higher-order subsystems, and discusses the interfacing of the analytical solutions for the stiff linear subsystems with the numerical solution of the rest of the system. Chapter V evaluates the method, comparing it with a fourth-order Runge-Kutta algorithm, and presents a general procedure for applying the method to systems of the type shown in Figures 1 and 2. This procedure is demonstrated with an example problem. Chapter VI presents conclusions and recommendations.

The results of the study show that the transition matrix approach achieves more efficient simulation of systems such as Figure 1 when the inputs to the stiff linear subsystems have periods that are long relative to the time constants of the subsystems. Depending on the stiffness of the linear subsystems and the overall loop frequencies of the system, step size increases of two orders of magnitude or more are possible using this approach.

CHAPTER II

SURVEY OF RELATED TOPICS

This chapter will present a short discussion of conventional numerical methods for solving differential equations, and the results of a literature search for methods specifically intended for stiff systems.

Numerical Integration Methods

The basic problem addressed by numerical integration methods is the solution of the first-order vector differential equation

$$\frac{dX}{dt} = f(X,t)$$

subject to some initial condition for vector X. At each point in the X - t hyperplane, the function f(X,t) gives the slope of the solution trajectory. As shown in Figure 3, short line segments with this slope can be drawn at points throughout the plane. If smooth curves are drawn following the indicated slopes, a map of the solution trajectories in the X - t plane is obtained.

Numerical integration methods trace out a particular solution trajectory originating at the initial condition. Two basic types of numerical integration algorithms are in common use. These are one-step methods and multi-step methods.

The simplest one-step method is called the crude Euler method. At the initial condition, the slope of the solution curve is calculated and the solution is advanced along this slope one time step. A new slope is then calculated and the solution proceeds. Higher-order methods utilize past values of the dependent variable and the slope to obtain a better prediction. The Runge-Kutta second and fourth-order methods are examples of higher order one-step methods. One-step integration methods have certain characteristics in common. They do not require iteration to find the next solution point; they are self-starting, meaning they require only an initial point to begin the solution, and they do not provide an estimate of the error incurred at each integration step.

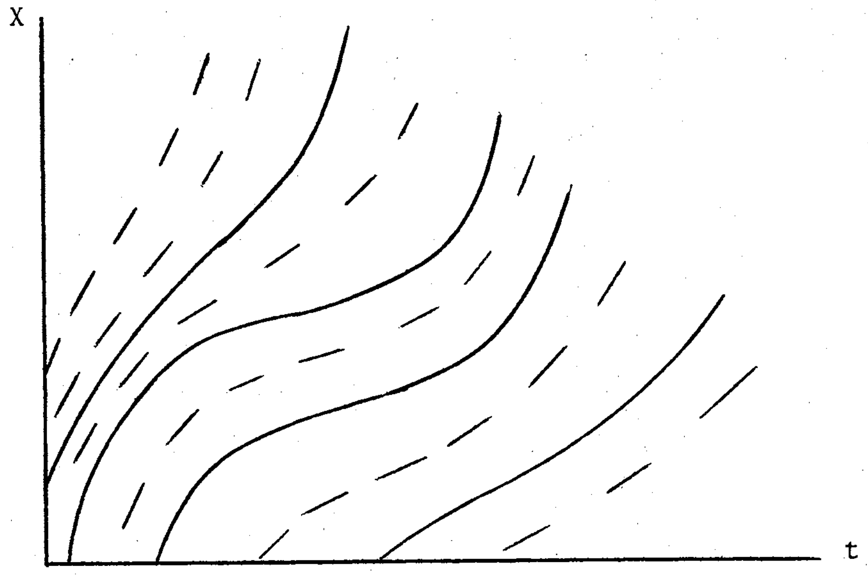

Figure 3. Solution Trajectories

In contrast to the one-step methods, the multi-step, or predictor-corrector methods require iteration at each step; they are not self-starting and they provide error information at each step. These methods use a predictor formula of the one-step type and a corrector formula to recalculate the predicted trajectory point repeatedly until two consecutive calculations agree within a specified error. Then the predictor formula predicts the next point and the corrector formula again iterates and so on. These methods typically adjust the step size according to the number of iterations required by the corrector formula to achieve convergence. The predictor formula, or some other one-step method is used to obtain enough initial points on the trajectory to "start" the corrector formula on its first iteration. Beckett (2) gives a description of several integration techniques of both the single and multi-step type.

Unfortunately, neither single-step nor multi-step methods as described above are well suited for the solution of systems having both large and small time constants.

Methods Designed to Cope With Small Time Constants

Ebbesen (5) develops an algorithm based on the variational principle of mechanics. It is applicable to both linear and nonlinear systems of equations, and allows selection of the step size based on the low-frequency system components. The method is designed for those systems having dominant low-frequency responses, and suppresses high-frequency responses. Significant reductions in computation time along with accurate solutions are reported.

Andrus (1) describes a method applicable to systems of first-order linear differential equations with constant coefficients. A transformation of the original equations into a system called the canonical equations is described. Those canonical equations depending on eigenvalues of large magnitude are discarded when their solutions contribute negligibly to the total system response. This allows a larger integration step size than would ordinarily be possible. When it is not possible to decide which canonical equations to eliminate, the input function is approximated by linear functions over short time intervals. The solution to the canonical equations is expressed analytically in terms of the unknown input. This expression is then substituted into the canonical equations, and they are then integrated numerically. A significant increase in the required step size is reported.

Stineman (9) assumes that the high frequencies in the system decay rapidly. Therefore a time step based on these high frequencies is used during the initial transient. The time step is then increased to approach a fraction of the longest time constant in the system. This method is not effective if the system is non-periodically excited since the high-frequency responses remain in the solution, and the time step must remain small. Conventional predictor-corrector techniques could be used to advantage where an input such as a step is applied to a system, resulting in high-frequency responses that decay off, since the step size would be adjusted upward. Some one-step methods with automatically adjusting step size could also be used to advantage. But these methods are not effective if the system is intermittently excited, for example, by sampling devices or nonlinearities in the model, since any abrupt change in the input, or even any non-periodic input, causes the high-frequency transients to remain in the solution.

Curtiss (4) describes a forward interpolation method which singles out and approximates a particular solution of the differential equation.

Treanor (10) develops a method which is closely related to the Runge-Kutta method. An approximation is made that within an interval the first derivative can be expressed in a special form. In certain cases the algorithm reduces to the fourth-order Runge-Kutta method.

Walters (11) describes a multi-step predictor-corrector approach to solving systems of stiff ordinary differential equations. Stability criteria for multi-step methods are presented.

Benyon (3) provides an excellent survey of existing numerical techniques for digital computer solution of systems of differential equations. It includes a table summarizing the author's experience with various numerical integration techniques applied to several problems, giving the relative computation time for each. An extensive bibliography is included.

The literature search revealed that relatively few techniques have been developed to efficiently solve stiff differential equations. More effort seems to have been spent on methods designed to handle large sets of algebraic and differential equations.

CHAPTER III

DEVELOPMENT OF TRANSITION MATRIX APPROACH

This chapter will develop the transition matrix approach for the solution of stiff linear subsystems.

The Transition Matrix

Consider a set of time-invariant, linear, ordinary differential equations,

$$\dot{X}(t) = AX(t) + BU(t), \qquad X(0) = X_0 \quad . \tag{1}$$

X is an n-vector called the state vector, A is an n x n matrix of constants called the plant matrix, B is an n x m matrix of constants, and U is an m-vector called the control or input. The initial condition for the state is X_0. The general solution can be written

$$X(t) = \Phi(t,t_0)\, X_0 + \int_{t_0}^{t} \Phi(t,\tau)\, BU(\tau)\, d\tau \quad . \tag{2}$$

Φ is the state transition matrix for the system. Note that the solution consists of two parts: a homogeneous and a particular solution. The homogeneous solution is the solution when the input U is zero. It is therefore termed the "zero-input response". Similarly, the particular solution is the solution when the state X_0 is zero, and it is termed the "zero-state response". The integral, which is the particular solution, is called the "convolution integral".

The transition matrix is given by

$$\Phi(t,t_0) = e^{A(t-t_0)}$$

which may easily be evaluated as an n x n matrix of constants for particular values of t and t_0. One convenient means of evaluating the transition matrix is by taking the inverse Laplace transform of the resolvent matrix:

$$e^{At} = \mathcal{L}^{-1}\{R(S)\}$$

where the resolvent matrix, R(S), is given by

$$R(S) = (SI - A)^{-1} \quad .$$

Thus the zero-input response may be readily obtained by evaluating the transition matrix.

The Convolution Integral

Linear, homogeneous systems rarely occur in simulations. Generally, linear subsystems will have some forcing function as input. If this forcing function is known in advance, and an analytical expression for it can be determined, then the exact response of the subsystem can be determined by Equation (2).

Unfortunately, in dynamic simulations, the input to the subsystem is not known in advance, and generally varies in an unpredictable manner. In most cases, then, some numerical method must be used to obtain the subsystem response.

When the subsystem is not stiff, conventional numerical integration techniques such as Runge-Kutta are quite adequate. However, when the subsystem is stiff, it would seem advantageous to make use of the analytical solution, Equation (2), to avoid the numerical problems that would otherwise be encountered.

Some analytical approximation for the input to the stiff linear subsystem can be used, and updated each time step. The analytical solution can then be obtained for each time step, and the approximate solution propagated as a series of short analytical solutions corresponding to the series of input approximations. This basic method of dealing with stiff linear subsystems will be explored in the remainder of this study.

A simple approximation to the subsystem input is a series of step inputs, as shown in Figure 4. Here t, t_0, u(t) and Δt are time, the initial time, the input, and the time step, respectively. Admittedly, this is a crude approximation. It assumes simply that the input is constant during each time step, at the value it had at the beginning of that step. This is a "zero-order" approximation, in that the input is approximated as a zero-order polynomial in time, namely, a constant.

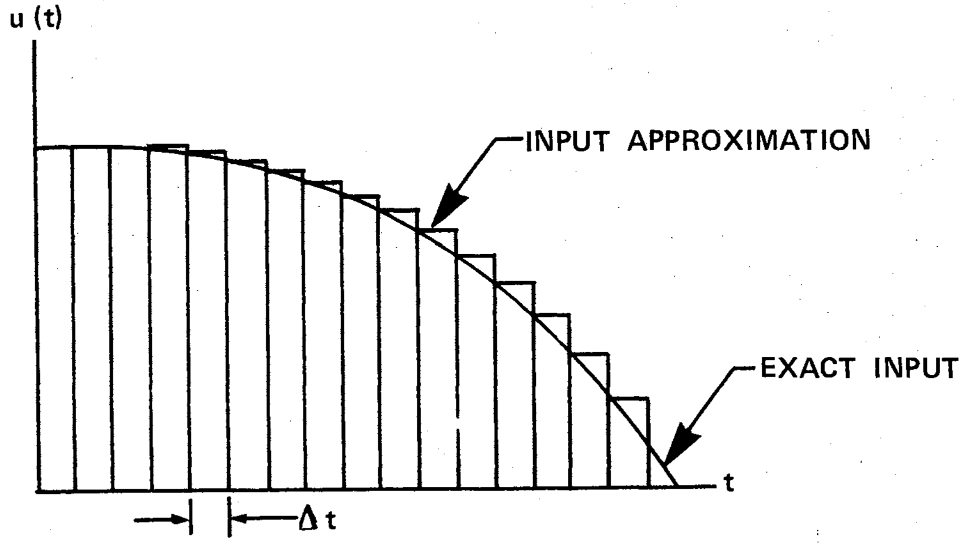

Figure 4. The "Step" Input Approximation

Other subsystem input approximations could be used. A first-order approximation would be as in Figure 5. Here, a series of straight lines connect consecutive points on the input curve, and the input function is approximated by a series of "ramp" inputs. This is obviously a much better approximation to the input curve than the step approximation.

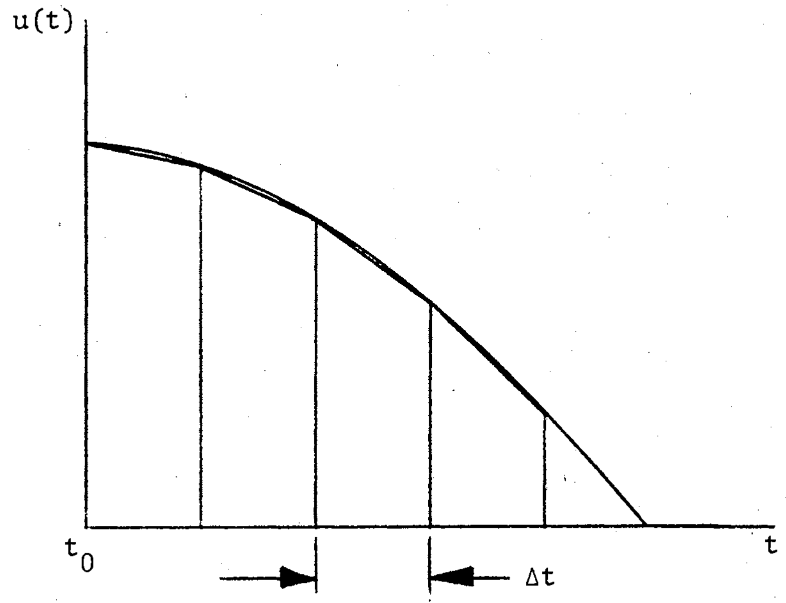

Figure 5. The "Ramp" Input Approximation

Many other analytical approximations to the subsystem input curve could be used, for instance, second, third or higher-order polynomials in time, or exponential functions of time. These would require the storage of several previous values of the input, and would be more time consuming than either the step or ramp approximations. This study focuses on the use of the step approximation for the subsystem input.

The step input approximation offers a unique advantage in terms of the evaluation of the convolution integral. Since the input is assumed constant over each time step, it can be factored outside the integral:

$$\int_{t_0}^{t} \Phi(t,\tau)\ BU(\tau)\ d\tau = u(t_0) \int_{t_0}^{t} \Phi(t,\tau)\ B\ d\tau \quad .$$

Here u is a scalar. It can be shown that the convolution integral is a function only of $t - t_0$, rather than a function of both t and t_0. Denoting

$$\Theta(t - t_0) = \int_{t_0}^{t} \Phi(t,\tau)\ B\ d\tau \quad ,$$

the transformation

$$\lambda = \tau - t_0$$

gives

$$\Theta(t - t_0) = \int_{0}^{t-t_0} \Phi(t,\lambda + t_0)\ B\ d\lambda \quad .$$

Since

$$\Phi(t,\lambda + t_0) = e^{A[(t-t_0)-\lambda]} \quad ,$$

which is denoted $\Phi[(t - t_0) - \lambda]$,

$$\Theta(t - t_0) = \int_{0}^{t-t_0} \Phi[(t - t_0) - \lambda]\ B\ d\lambda \quad ,$$

which is clearly a function of $t - t_0$ rather than t and t_0. This is an important fact, since if t_0 is considered the beginning time of a time step, and t is the time at the end of the step, then

$$\Theta(\Delta t) = \int_0^{\Delta t} \Phi[\Delta t - \lambda]\; B\; d\lambda \qquad ,$$

a function of only the time step Δt. Note that for a particular value of Δt, Θ is an n x 1 constant vector. This means Θ can be evaluated once at the beginning of the solution, and it is constant thereafter, if the time step does not change. So, rewriting Equation (2),

$$X(t_0 + \Delta t) = \Phi(\Delta t)\; X_0 + u(t_0)\; \Theta(\Delta t) \qquad . \qquad (3)$$

This is a formula which can be used to propagate the subsystem solution forward in time. $\Phi(\Delta t)$ and $\Theta(\Delta t)$ are constant throughout the solution. No other input approximation allows such a straight-forward evaluation of the convolution integral. However, as will be seen in the next section, other subsystem input approximations can be handled almost as simply by the formation of an augmented system.

The Augmented System

An alternate way of using the step input approximation to arrive at a propagation formula similar to Equation (3) is to form an augmented system consisting of the original subsystem plus certain new states which contain input information. This method of forming an augmented system can easily be generalized to higher-order input approximations. The method will first be demonstrated on some second-order subsystems. Then a generalization will be presented.

Consider first the subsystem,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} u(t) \quad . \qquad (4)$$

If u(t) is approximated as a series of steps, u can be expressed as some constant

$$u = u_i$$

over each time step, where u_i is the value of the input u(t) at the beginning of the i^{th} step. Define a new state variable

$$x_3 = u$$

with initial condition

$$x_3(t_i) = u(t_i) \quad .$$

Since x_3 is constant over each time step,

$$\dot{x}_3 = 0 \quad .$$

Rewriting Equation (4) in the form of an augmented system,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad .$$

Note that this is a time-invariant, linear, <u>un-forced</u> system. Letting X_a denote the augmented state vector, and A_a denote the augmented plant matrix,

$$\dot{X}_a = A_a X_a \quad . \qquad (5)$$

The stepwise solution for Equation (5) is

$$X_a(t_i + \Delta t) = \Phi_a(\Delta t) X_a(t_i)$$

where Φ_a is the state transition matrix for the augmented plant matrix A_a. Φ_a is a 3 by 3 matrix of constants which may be evaluated for a particular value of Δt at the beginning of the solution. Note that augmenting the second-order subsystem with a zero-order input resulted in a third-order augmented system.

Reconsider subsystem Equation (4) with u(t) approximated by

$$u = m_i t + u_i$$

where u_i is the value of the input at the beginning of the ith step and m_i is the approximate slope of the input during the ith step. This is a ramp input approximation. Define a new state

$$x_3 = u$$

with initial condition

$$x_3(t_i) = u(t_i) \quad .$$

Clearly,

$$\dot{x}_3 = m_i \quad .$$

Now rewriting Equation (4) as an augmented system,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} m_i \quad .$$

Define another new state,

$$x_4 = m_i , \qquad x_4(t_0) = m_1$$

$$\dot{x}_4 = 0 \quad .$$

Augmenting again,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 & 0 \\ a_{21} & a_{22} & b_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad .$$

Again, an unforced system results. A second-order subsystem with a first-order input results in a fourth-order augmented system. In general, augmenting an n^{th}-order subsystem with an m^{th}-order polynomial in t for the input produces an $(n + m + 1)^{th}$-order augmented system. Of course, for each unforced, augmented system obtained by the various input approximations, the corresponding transition matrix may be obtained, evaluated for some time step, and used to propagate the solution.

This method can also be applied when the input is a vector rather than a scalar. Consider

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} . \qquad (6)$$

Let

$$u_1 = u_{1i}$$
$$u_2 = u_{2i} \quad .$$

Define

$$x_3 = u_1, \qquad x_3(t_i) = u_1(t_i)$$
$$x_4 = u_2, \qquad x_4(t_i) = u_2(t_i) \quad .$$

Then

$$\dot{x}_3 = 0$$
$$\dot{x}_4 = 0 \quad .$$

Augmenting

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_{11} & b_{12} \\ a_{21} & a_{22} & b_{21} & b_{22} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} .$$

Augmenting a second-order subsystem with a 2-vector of zero-order inputs yields a fourth-order augmented matrix.

Now consider subsystem Equation (6) with

$$u_1 = m_{1i}t + u_{1i}$$

$$u_2 = m_{2i}t + u_{2i} \quad .$$

Define

$$x_3 = u_1, \qquad x_3(t_i) = u_1(t_i)$$

$$x_4 = u_2, \qquad x_4(t_i) = u_2(t_i) \quad .$$

Thus,

$$\dot{x}_3 = m_{1i}$$

$$\dot{x}_4 = m_{2i}$$

Now let

$$x_5 = m_{1i}, \qquad x_5(t_0) = m_{11}$$

$$x_6 = m_{2i}, \qquad x_6(t_0) = m_{21} \quad .$$

Augmenting,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_{11} & b_{12} & 0 & 0 \\ a_{21} & a_{22} & b_{21} & b_{22} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} .$$

Augmenting a second-order subsystem with a 2-vector of first-order inputs yields a sixth-order augmented system. In general, augmenting an nth-order subsystem with a m-vector input, each element of which is kth-order in t results in an $[n + m(k + 1)]$-order augmented system.

A generalization of the augmenting process for polynomial input approximations follows. The assumption is made that each element of the m-vector input is the same order polynomial. For the nth-order subsystem Equation (1) where each element of U(t) is a kth-order polynomial in t,

$$\dot{X}_A = \left[\begin{array}{c|c|c} A & B & O_1 \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{O_2}} & I \\ \cline{3-3} \multicolumn{2}{c|}{} & O_3 \end{array}\right] X_A \quad .$$

The augmented matrix above is [n + m(k + 1)]-order. 0_1 is an n x mk zero matrix, 0_2 is an m(k + 1) x (n + m) zero matrix, I is an mk x mk identity matrix, and 0_3 is an m x mk zero matrix. To fit this form, the new states must be assigned in the same order as in the examples. That is, first assign state variables to each input-vector element, then to the first derivative of each element, then to the second derivative of each element, and so on.

Broader generalizations of the augmenting procedure are possible. Melsa (7) presents a method for obtaining an augmented-system representation for a general, linear, closed-loop system with an input whose Laplace transform is a rational function of the Laplace variable S. He treats only the scalar input case, however. More information on this method is given in Chapter IV.

CHAPTER IV

IMPLEMENTING THE METHOD

The present chapter will demonstrate the implementation of the step transition matrix approach for first and second-order linear subsystems, reference a computer program for implementing the method on higher-order linear subsystems, and present a method for interfacing the solutions of the stiff linear subsystems with conventional numerical solution of the remainder of the system.

First and second-order subsystems occur more frequently in simulations than any others. This is not to say that most physical systems can be accurately described as first or second-order linear systems. However, even in very complex simulations, certain components in the system can be adequately described by first or second-order models.

First-Order Subsystems

Consider the general first-order subsystem shown in Figure 6. T is the system time constant, U(S) is the input to the system and Y(S) is the output.

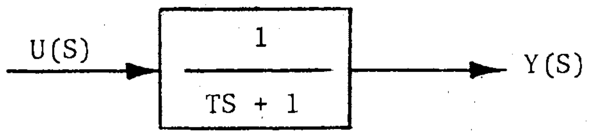

Figure 6. Block Diagram of a First-Order Subsystem

Writing the algebraic expression equivalent to Figure 6 and taking the inverse Laplace transform yields the following differential equation:

$$T \frac{dy(t)}{dt} + y(t) = u(t) \qquad .$$

Assigning the state variable x_1 to the output and rearranging gives

$$\dot{x}_1 = -(\tfrac{1}{T})\, x_1 + (\tfrac{1}{T})\, u(t) \qquad . \qquad (7)$$

At this point, an analytic expression can be selected to approximate input u(t). As stated in Chapter III, the step input is focused upon here for study. Accordingly,

$$x_2 = u_i, \qquad x_2(t_0) = u(t_0)$$

$$\dot{x}_2 = 0 \qquad .$$

Augmenting Equation (7) gives

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{T} & \frac{1}{T} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

for which the stepwise solution is

$$X(t_i) = \Phi(\Delta t)\, X(t_{i-1}) \tag{8}$$

where t_i is the i^{th} point in time, and t_{i-1} is the previous time. Using standard techniques, the transition matrix for the augmented system is found to be

$$\Phi(\Delta t) = \begin{bmatrix} e^{-\frac{\Delta t}{T}} & 1-e^{-\frac{\Delta t}{T}} \\ 0 & 1 \end{bmatrix} .$$

Substituting this result into Equation (8), note that only the first row of the matrix is needed to propagate the solution. The second row merely contains the information

$$x_2(t_i) = x_2(t_{i-1})$$

which is a statement that the input x_2 is constant during the time step. Using Equation (8) with the first row of the transition matrix gives

$$x_1(t_i) = e^{-\frac{\Delta t}{T}}\, x_1(t_{i-1}) + (1 - e^{-\frac{\Delta t}{T}})\, x_2(t_{i-1}) \quad . \tag{9}$$

Equation (9) is the propagation formula for a first-order subsystem using the step input approximation.

Second-Order Subsystems

Consider the general second-order subsystem shown in Figure 7. ζ and ω_η are the damping ratio and natural frequency, respectively.

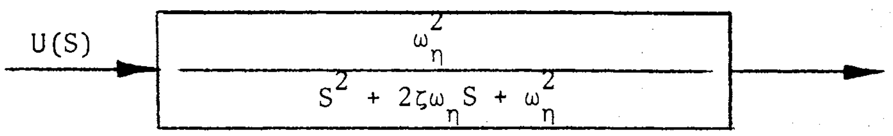
U(S)
$\frac{\omega_\eta^2}{S^2 + 2\zeta\omega_\eta S + \omega_\eta^2}$
Y(S)

Figure 7. Block Diagram of a Second-Order Subsystem

Taking the inverse Laplace transform of the algebraic equivalent to Figure 7 gives the following differential equation:

$$\ddot{y}(t) + 2\zeta\omega_\eta \dot{y}(t) + \omega_\eta^2 y(t) = \omega_\eta^2 u(t) \quad . \tag{10}$$

Changing to state variable form, let

$$x_1 = y(t), \qquad x_1(t_0) = y(t_0)$$

$$x_2 = \dot{y}(t) \quad .$$

Then Equation (10) becomes

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -2\zeta\omega_\eta x_2 - \omega_\eta^2 x_1 + \omega_\eta^2 u(t) \quad .$$

Using the step input approximation, let

$$x_3 = u_i, \qquad x_3(t_0) = u(t_0)$$

$$\dot{x}_3 = 0 \quad .$$

Then the augmented-system representation of Equation (10) is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\omega_\eta^2 & -2\zeta\omega_\eta & \omega_\eta^2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} ,$$

for which the stepwise solution is Equation (8). The resolvent matrix for the augmented system is readily found, and each element expanded in partial fractions. The form of the inverse Laplace transform of the resolvent matrix is different for each of the three cases $0 \leq \zeta < 1$, $\zeta = 1$, and $\zeta > 1$. The resulting formulae for the elements of the $\Phi(\Delta t)$ matrix, being rather lengthy, are given in Appendix A. Thus the propagation formula for the second-order subsystem with step input approximation is

$$\begin{bmatrix} x_1(t_i) \\ x_2(t_i) \\ x_3(t_i) \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t_{i-1}) \\ x_2(t_{i-1}) \\ x_3(t_{i-1}) \end{bmatrix}$$

where the ϕ_{ij} are given in Appendix A by the approximate formulae depending on ζ. Note that only the first two rows of the Φ matrix are used to propagate the states, and the third row says that the input, x_3, is constant over each time step.

The elements of the $\Phi(\Delta t)$ matrix are constant throughout the solution. Even though the formulae for these elements may be lengthy, the time required to calculate them is negligible since they are calculated only once, at the beginning of the solution.

The results above express the step transition matrix propagation formula for any second-order subsystem, since all such subsystems can be expressed in the form of Figure 7.

A Computer Program for Implementation on General Linear Subsystems

Even for a second-order subsystem, the work involved in obtaining the elements of the $\Phi(\Delta t)$ matrix for the simple step input approximation is time consuming. More complex input approximations require even more work. Fortunately, Melsa (7) presents a computer program called "RTRESP" which computes the time response in closed-form of the general closed-loop system,

$$\dot{X}(t) = AX(t) + bu(t)$$

$$u(t) = K[r(t) - k^T X(t)]$$

$$y(t) = C^T X(t)$$

corresponding to Figure 8. In Figure 8, u(t) is a scalar input, r(t) is a scalar reference input, k is a vector feedback coefficient, I is the n^{th}-order identity matrix, C is an n x 1 output vector, and y(t) is the scalar output. A computer program is presented for finding the closed-form time response for this system, given some analytical input function whose Laplace transform is a rational function of the Laplace variable S with a pole-zero excess of at least one. This program can be utilized effectively in conjunction with the transition matrix method, since it forms the augmented system and finds the transition matrix.

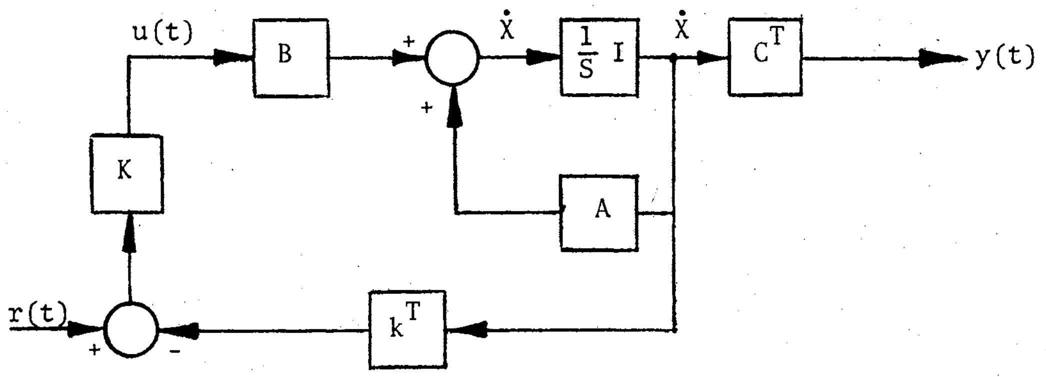

Figure 8. A Linear Closed-Loop System

Sebesta (8) includes a program called "RTRES", a modification of "RTRESP" which will generate the step transition matrix $\Phi(\Delta t)$ for such a system. This transition matrix can then be used to propagate the solution. A restriction on the use of "RTRES" is that the augmented system may not have repeated eigenvalues.

Interfacing the Transition Matrix Method With a Conventional Technique

The transition matrix approach is intended to be used to propagate the solution of stiff linear subsystems within an overall model. The remainder of the system is simulated with a conventional method. Figure 9 shows a flow chart of the interfacing of the two methods. A more specific procedure for interfacing the step transition matrix solution of stiff linear subsystems with a Runge-Kutta fourth-order solution for the remainder of the system is shown in Figure 10. Note that the solutions of the stiff linear subsystems are advanced twice per Runge-Kutta step. This specific procedure is used to solve an example problem in Chapter V.

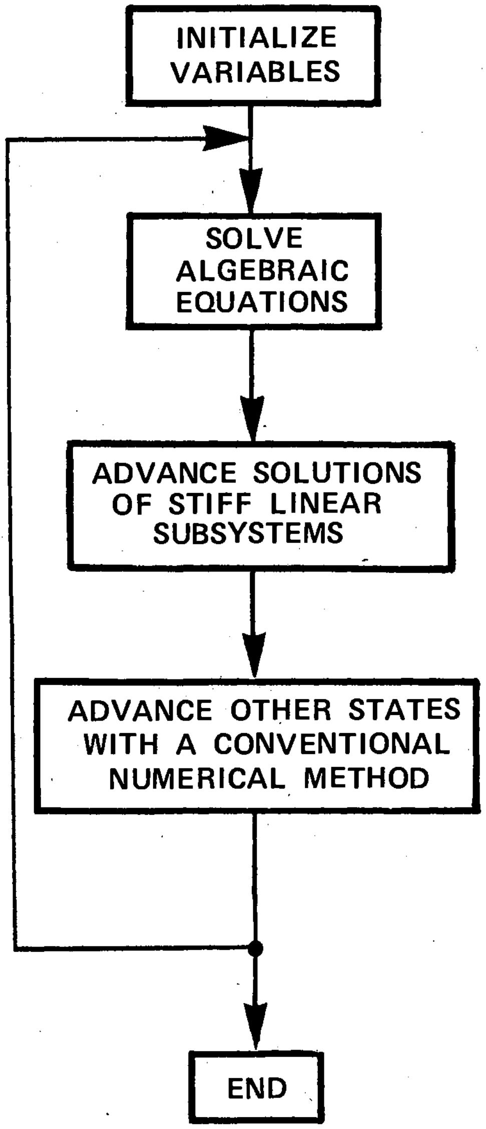
INITIALIZE VARIABLES
SOLVE ALGEBRAIC EQUATIONS
ADVANCE SOLUTIONS OF STIFF LINEAR SUBSYSTEMS
ADVANCE OTHER STATES WITH A CONVENTIONAL NUMERICAL METHOD
END

Figure 9. Interfacing the Transition Matrix With a Conventional Method

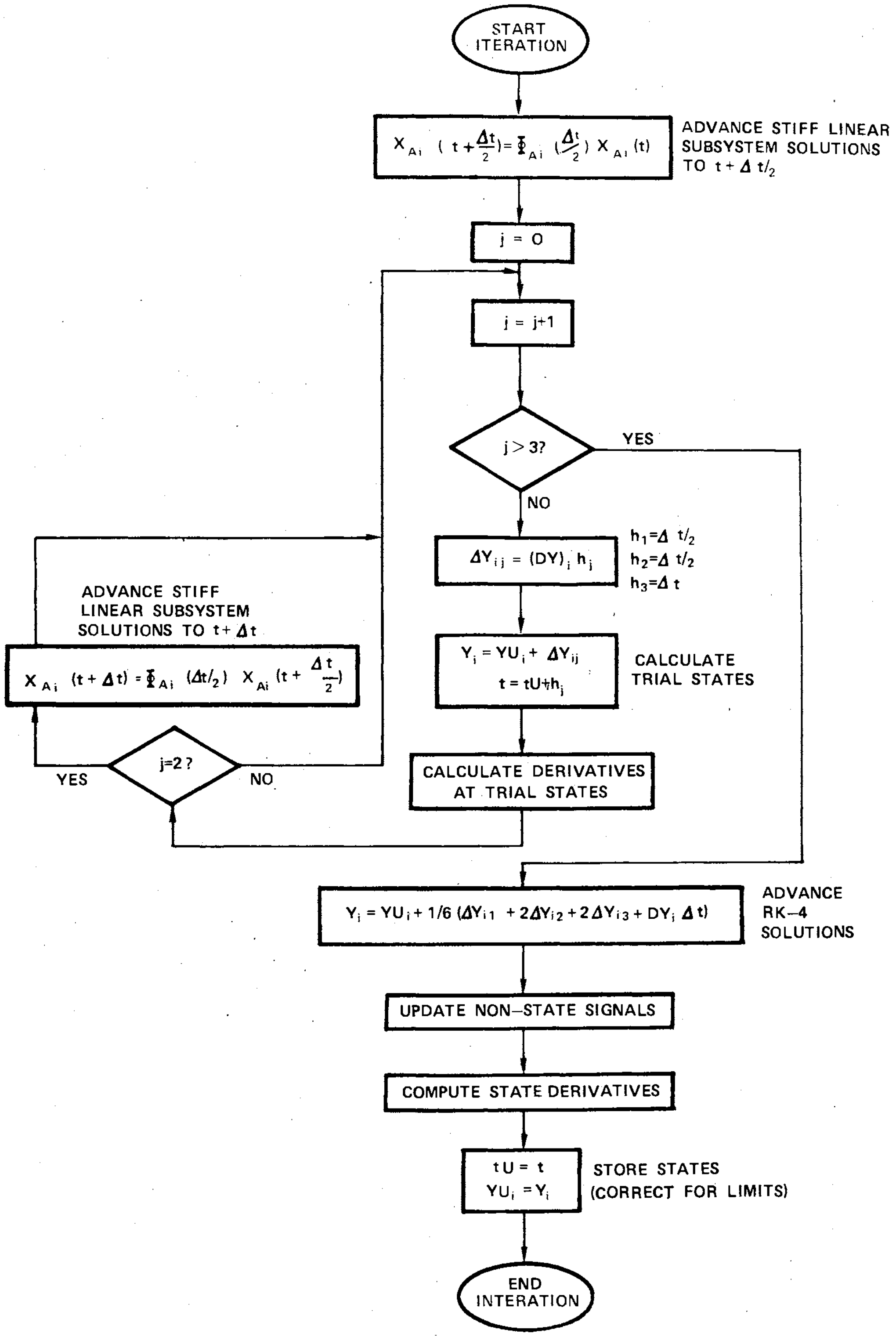
START ITERATION
$X_{Ai}(t+\frac{\Delta t}{2}) = \Phi_{Ai}(\frac{\Delta t}{2}) X_{Ai}(t)$
ADVANCE STIFF LINEAR SUBSYSTEM SOLUTIONS TO $t+\Delta t/2$
$j = 0$
$j = j+1$
$j > 3?$
YES
NO
$\Delta Y_{ij} = (DY)_i h_j$
$h_1=\Delta t/2$
$h_2=\Delta t/2$
$h_3=\Delta t$
$Y_i = YU_i + \Delta Y_{ij}$
$t = tU + h_j$
CALCULATE TRIAL STATES
CALCULATE DERIVATIVES AT TRIAL STATES
ADVANCE STIFF LINEAR SUBSYSTEM SOLUTIONS TO $t+\Delta t$
$X_{Ai}(t+\Delta t) = \Phi_{Ai}(\Delta t/2) X_{Ai}(t+\frac{\Delta t}{2})$
$j=2?$
YES
NO
$Y_i = YU_i + 1/6(\Delta Y_{i1} + 2\Delta Y_{i2} + 2\Delta Y_{i3} + DY_i \Delta t)$
ADVANCE RK-4 SOLUTIONS
UPDATE NON-STATE SIGNALS
COMPUTE STATE DERIVATIVES
$tU = t$
$YU_i = Y_i$
STORE STATES (CORRECT FOR LIMITS)
END INTERATION

Figure 10. Flow Chart for Interfacing Transition Matrix Method With RK-4

CHAPTER V

EVALUATING THE METHOD

This chapter first presents an analytical expression for the error incurred propagating the solutions of the stiff linear subsystems. Then the results of an extensive computer study of the performance of the transition matrix method on first and second-order linear subsystems are reported. Comparison is made with the performance of a fourth-order Runge-Kutta algorithm for accuracy and computation time. The results of this computer study are then used to formulate a general procedure for applying the transition matrix approach to systems of the type shown in Figures 1 and 2 of Chapter I, having stiff linear subsystems of any order. Finally an example problem is considered to demonstrate the application of the procedure, and the results are discussed.

Several factors affect the results of any machine computation. They are considered in Appendix B. Because these factors influence the results of a computer study of any numerical method, analytical predictions for the performance of a numerical method are desirable. They provide a baseline for evaluating the method which is not dependent on these highly variable conditions. To this end, the next section derives an error expression for the transition matrix method applied to linear subsystems.

Analytical Subsystem Analysis

Consider again the linear, time-invariant subsystem

$$\dot{X} = A\ X + B\ U$$

for which the exact solution after the first time step is

$$X(t_1) = X(t_0)\ \Phi(\Delta t) + \int_{t_0}^{t_1} \Phi(t_1 - \tau)\ B\ U(\tau)\ d\tau \quad .$$

U(t) is the exact, continuous input vector. Denote the analytical approximation to that input vector over the first time step as $U_1^*(t)$. The analytical approximation to the input over the ith time step is denoted $U_i^*(t)$. The approximate state after the first time step is

$$X^*(t_1) = X(t_0)\ \Phi(\Delta t) + \int_{t_0}^{t_1} \Phi(t_1 - \tau)\ B\ U_1^*(\tau)\ d\tau$$

where X^* denotes the approximate solution. Denote the additional error due to approximating the input over the ith time step as $E(t_i)$. The error after the first time step is obtained as

$$E(t_1) = X(t_1) - X^*(t_1) = \int_{t_0}^{t_1} \Phi(t_1 - \tau)\ B\ U(\tau)\ d\tau - \int_{t_0}^{t_1} \Phi(t_1 - \tau)\ B\ U_1^*(\tau)\ d\tau .$$

Note that this error is due only to approximating the input as U_1^*. The exact solution after the second time step is

$$X(t_2) = \Phi(\Delta t)\ X(t_1) + \int_{t_1}^{t_2} \Phi(t_2 - \tau)\ B\ U(\tau)\ d\tau$$

and the approximate solution is

$$X^*(t_2) = \Phi(\Delta t)\, X^*(t_1) + \int_{t_1}^{t_2} \Phi(t_2 - \tau)\, B\, U_2^*(\tau)\, d\tau \quad .$$

Denote the total error at time t_i as

$$E_T(t_i) = X(t_i) - X^*(t_i) \quad .$$

After the second time step,

$$\begin{aligned} E_T(t_2) &= \Phi(\Delta t)[X(t_1) - X^*(t_1)] + \int_{t_1}^{t_2} \Phi(t_2 - \tau)\, B\, U(\tau)\, d\tau \\ &\quad - \int_{t_1}^{t_2} \Phi(t_2 - \tau)\, B\, U_2^*(\tau)\, d\tau \\ &= \Phi(\Delta t)\, E(t_1) + E(t_2) \quad . \end{aligned}$$

Note that at the end of the first time step, E and E_T are the same since both the exact and the approximate solutions start from the same initial condition. However, for every point in time after t_1, it is important to distinguish between $E(t_i)$, the additional error incurred by approximating the input over the i^{th} step, and $E_T(t_i)$, the total error at t_i composed of both $E(t_i)$ and the error propagated by the inexact previous state, $X^*(t_{i-1})$. The situation is depicted in Figure 11.

A general expression for the additional error introduced by approximating the input during the i^{th} step is

$$E(t_i) = \int_{t_{i-1}}^{t_i} \Phi(t_i - \tau)\, B\, U(\tau)\, d\tau - \int_{t_{i-1}}^{t_i} \Phi(t_i - \tau)\, B\, U_i^*(\tau)\, d\tau \qquad (11)$$

Carrying the solution one step further in order to generalize, the error at t_3 is

$$E_T(t_3) = \Phi(\Delta t)[X(t_2) - X^*(t_2)] + E(t_3)$$

$$= \Phi(\Delta t)\ E_T(t_2) + E(t_3)$$

$$= \Phi(\Delta t)[\Phi(\Delta t)\ E(t_1) + E(t_2)] + E(t_3)$$

$$= \Phi(2\Delta t)\ E(t_1) + \Phi(\Delta t)\ E(t_2) + E(t_3) \quad .$$

An expression for the total error at the i^{th} time is

$$E_T(t_i) = \sum_{k=1}^{i} \Phi[(i - k)\Delta t]\ E(t_k) \quad . \qquad (12)$$

Equations (11) and (12) together give a method for calculating the error at t_i for the transition matrix approximate solution, for any vector input U(t), and any series of vector analytical input approximations $U_k^*(t)$, k = 1,2,...,i.

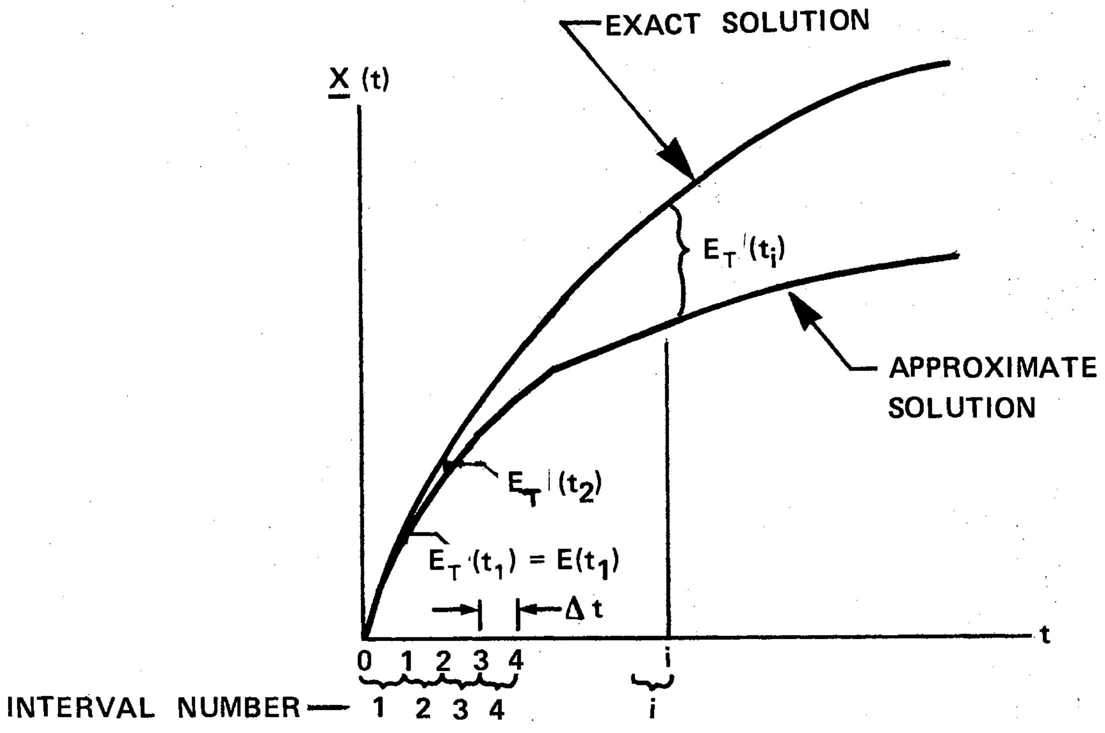

Figure 11. Analytical Error Notation

Equations (11) and (12) describe a recursive procedure for finding the error at t_i, since the error at t_i depends on the error at each previous value of time. More specific error expressions could be derived by restricting the input approximation to a certain type, such as a step approximation. However, a recursive evaluation procedure would still be necessary. Thus, as a practical matter of evaluating the error, it is better to simply propagate the subsystem solution by the transition matrix method and compare with an exact solution, if available. However, Equations (11) and (12) do lend some insight into the sources of error for the transition matrix simulation of a linear subsystem.

Examination of Equations (11) and (12) reveals the following facts.

1. The error at time t_i is a function of the following factors:
   - A. the time step Δt;
   - B. the input $U(t)$ and B;
   - C. the input approximation $U_i^*(t)$; and
   - D. the subsystem itself, since $\Phi(\Delta t)$ is determined from the subsystem A matrix.
2. The total error $E_T(t_i)$ is composed of two parts:
   - A. the error $E(t_i)$ due to the i^{th} approximation to the input;
   - B. the propagation of the previous errors $E(t_k)$, $k=1,2,\ldots,i-1$, through multiplication by the transition matrix.

These observations, although useful from the standpoint of understanding the sources of error for the method, do not give specific information about how accurate the method is, or when it is preferable to a conventional method. To obtain this more useful information, extensive computer studies of the method, as implemented in Chapter IV for first and second-order subsystems, were conducted. These are reported in the next section.

Computer Subsystem Error Study

Chapter IV presented the implementation of the step transition matrix method for first and second-order subsystems. In the computer study of the method, these subystems were subjected to the scalar input function

$$u(t) = \cos(\omega t) \quad . \tag{13}$$

This input function was chosen because the exact, analytical solution for both subsystems is easily obtained for this input, the input frequency ω can be easily varied, and for $\omega = 0$, u(t) is a unit step input.

For the first-order subsystem with this input, the exact solution is

$$y(t) = \frac{1}{1 + \omega^2 T^2} \{-e^{-\frac{t}{T}} + \cos \omega t + T\omega \sin \omega t\} \quad .$$

Initial conditions of zero were used throughout the investigations, except for the input, which has an initial condition of 1.

The exact solution for the second-order subsystem with a cosine input is easily obtained for three cases: an underdamped system ($0 \leq \zeta < 1$), a critically damped system ($\zeta = 1$), and an overdamped system ($\zeta > 1$). These solutions are given in Appendix C.

The step transition matrix method was programmed in double precision on the IBM 360 Model 65 computer to simulate general first and second-order subsystems according to the propagation formulae given in Chapter IV, and using input Equation (13). The approximate solutions obtained were compared with the analytical solutions to determine the error. Comparison was also made with solutions obtained independently using the fourth-order Runge-Kutta numerical integration algorithm.

The coding used to program these solutions is shown in Appendix D. These programs were written strictly as research tools, and the documentation is provided only to show the exact manner in which the error values were obtained. As discussed in Appendix B, the exact coding used to program a problem can often affect the results.

Of several possible error measures, average normalized per-cent error was chosen, because it combines relative-error and absolute-error information. Details on error measurement are given in Appendix E. In the remainder of the study, "error" should be understood as average normalized per-cent error unless stated otherwise.

A series of simulation runs were made to examine the performance of the step transition matrix method on first and second-order subsystems under a variety of conditions. An input amplitude of 1.0 was used throughout the tests, and a time interval of one second was simulated each run, starting at t = 0. The test results will be presented in a series of graphs.

Figure 12 shows error versus time step for the first-order subsystem for an input frequency of 6.28 rad/sec, and a subsystem time constant of 1.0. The error for three methods is shown: the step transition matrix, the ramp transition matrix, and Runge-Kutta fourth order. This graph shows that error increases with time step for all three methods. Note that for the step transition matrix method, halving the step size halves the error. In fact, for this input frequency and time constant, the error can be expressed as

$$\text{Error} = (350)(\Delta t) \quad .$$

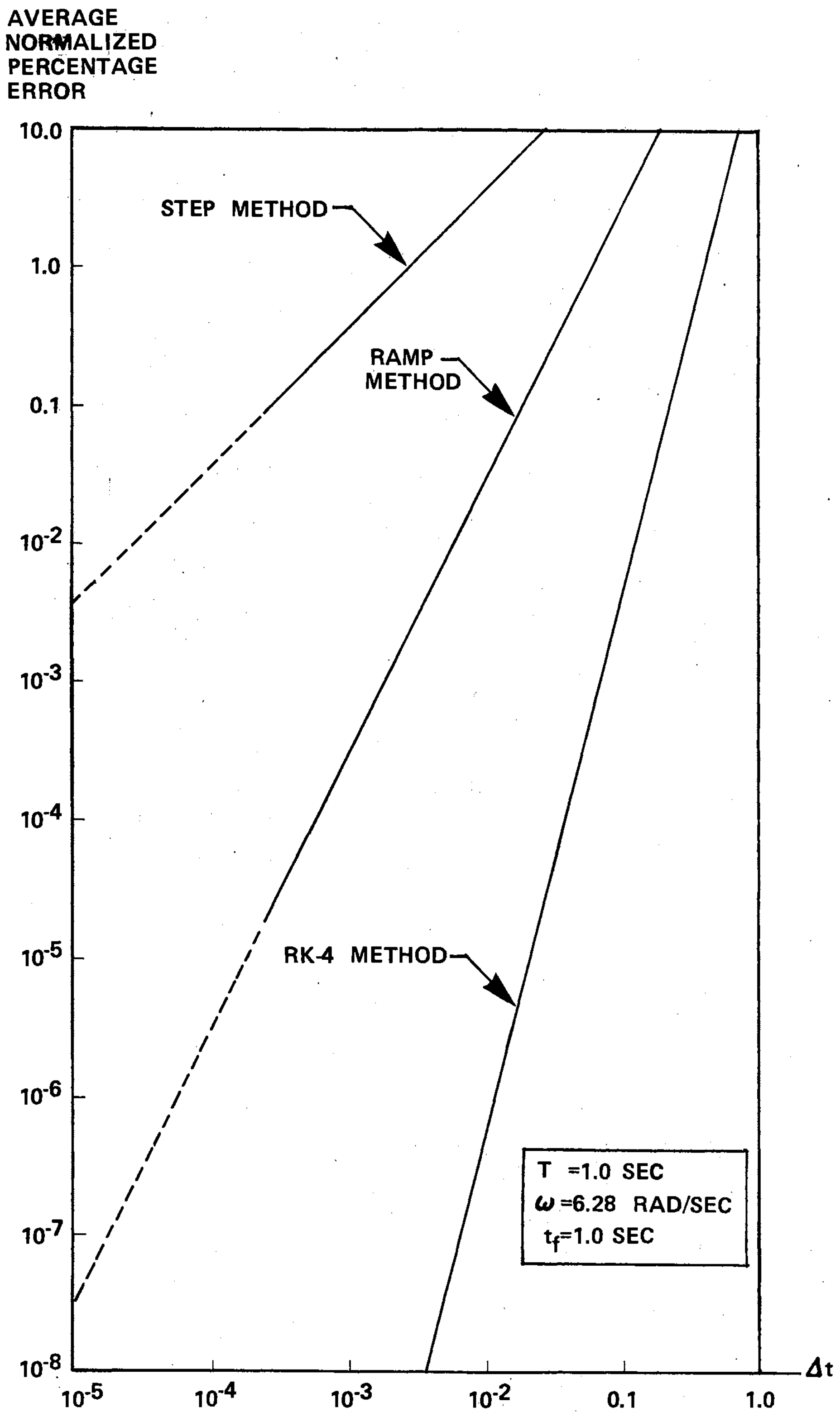

Figure 12. Variation of Error With Time Step for First-Order Subsystems

This indicates that when all other factors are held constant, the error is directly proportional to the time step. For the ramp method, the error is directly proportional to the square of the time step, as

$$\text{Error} = (280)(\Delta t)^2 \quad .$$

These two observations seem to indicate that, other factors being the same, the error for the transition matrix method, with an nth-order input approximation, is given by

$$\text{Error} = k(\Delta t)^{n+1}$$

where k is a constant determined by other factors. As expected, the RK-4 method exhibits error proportional to the fourth power of the time step. For these conditions, that error is approximately given by

$$\text{Error} = (50)(\Delta t)^4 \quad .$$

For this low-frequency subsystem, the RK-4 method has clearly superior accuracy. As expected, the ramp method has considerably better accuracy than the step method since the ramp is a first-order input approximation while the step is a zero-order approximation. Figure 13 shows a corresponding graph for the second-order subsystem. The comments for Figure 12 apply here also. Inspection of the graph reveals that for the step method,

$$\text{Error} = (820)(\Delta t),$$

for the ramp method,

$$\text{Error} = (340)(\Delta t)^2,$$

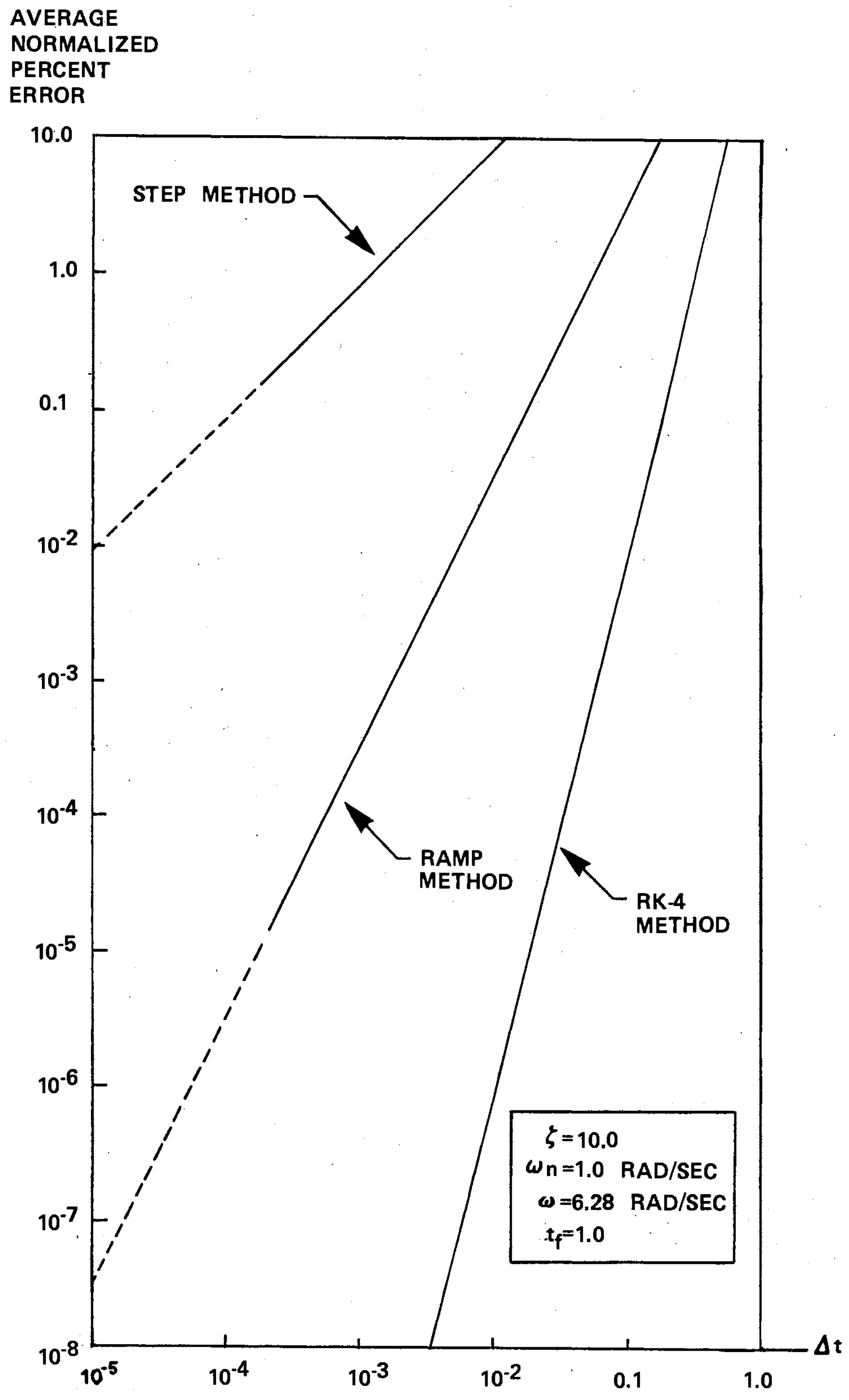

Figure 13. Variation of Error With Time Step for Second-Order Subsystems

and for the RK-4 method,

$$\text{Error} = (85)(\Delta t)^4,$$

for these conditions.

Figure 14 shows error plotted against input frequency for a first-order subsystem with a time step $\Delta t = 0.01$ seconds. The graph shows that error increases with input frequency for all three methods. Close inspection reveals that the error for both step and ramp approximations is proportional to the square of the input frequency, for input frequencies less than 2.0 rad/sec for the step method and 10.0 rad/sec for the ramp method. The approximations are: for the step method,

$$\text{Error} = 0.1\ \omega^2,$$

and for the ramp method,

$$\text{Error} = 4.5 \times 10^{-4}\ \omega^2 .$$

The error for the RK-4 method is proportional to the input frequency to a power between 3 and 4 for these conditions. Figure 15 is a corresponding plot for a second-order subsystem. It also shows that for low frequencies, the error for the transition matrix method is roughly porportional to the square of the input frequency. The RK-4 method exhibits more complex behavior. By the observations made from Figures 12 through 15, it appears that the error for the transition matrix could be approximated as

$$\text{Error} = k(\Delta t)^{n+1}\ \omega^2,$$

for low input frequencies. This approximation will be checked against

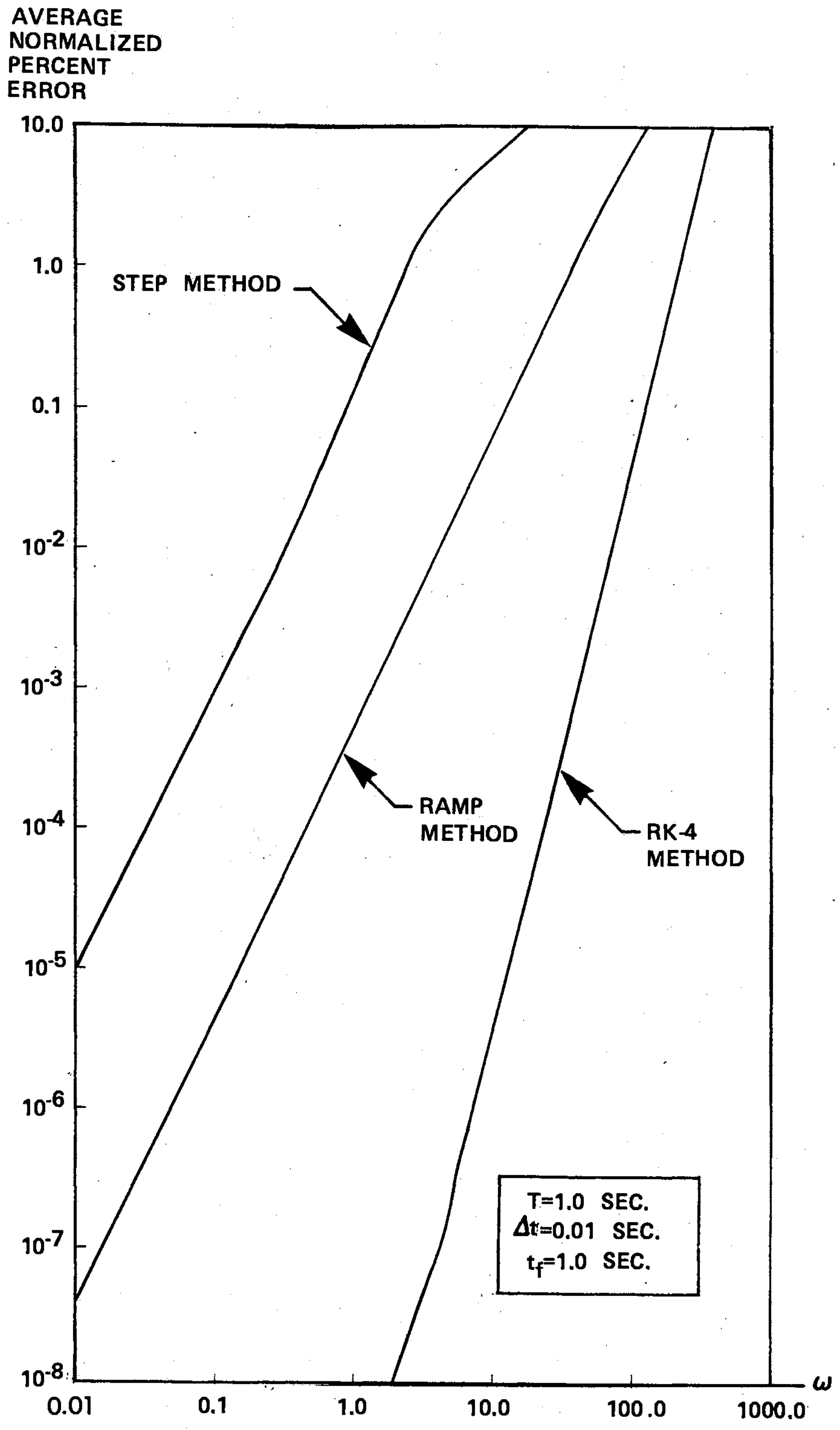

Figure 14. Variation of Error With Input Frequency for First-Order Subsystems

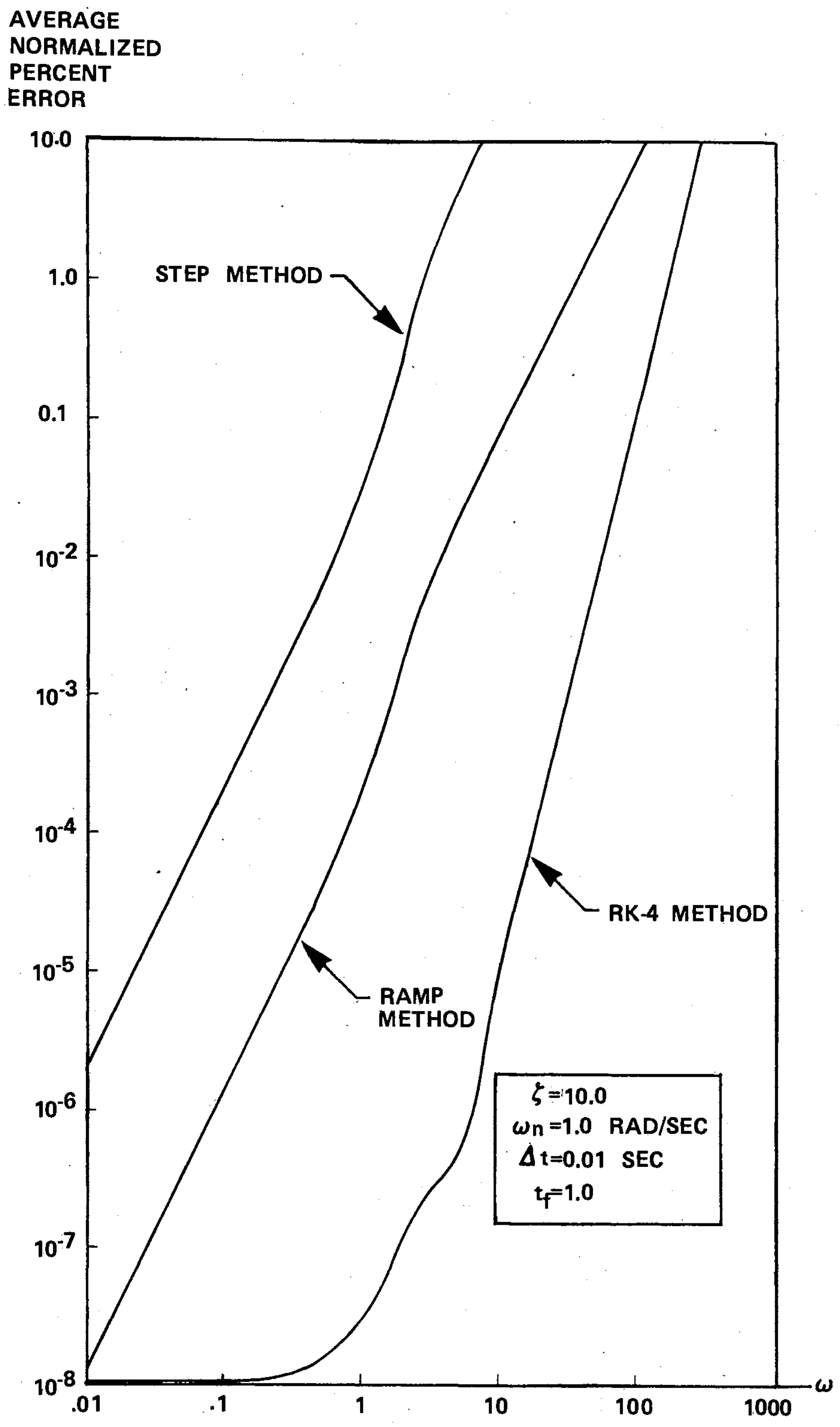

Figure 15. Variation of Error With Input Frequency for Second-Order Subsystems

subsequent results.

Figure 16 shows error plotted against the inverse of the system time constant for first-order subsystems. It is more convenient to use 1/T than T since the value of 1/T is indicative of the subsystem's stiffness. That is, the larger the value of 1/T, the smaller the subsystem time constant T and the stiffer the subsystem. The value of ω was picked as 2π, and error lines are shown for four time steps. First, notice that the error for the transition matrix method is not strongly affected by the stiffness of the subsystem. This seems reasonable since the transition matrix method is analytical, and owes its error to the input approximation rather than the subsystem's response. In contrast, the Runge-Kutta method is shown to be strongly a function of the subsystem time constant, the error increasing sharply as the stiffness increases. This also seems reasonable because for stiff subsystems, the response contains high-frequency transients, which the RK-4 must follow to remain accurate. If the subsystem is stiff enough, the response frequencies become too high for the time step used, and the solution becomes unstable. For the time steps used in Figure 16, the RK-4 method becomes unstable for 1/T in the range 250 to 2000. The transition matrix is clearly more accurate for very stiff subsystems. The ramp transition matrix shows error values about two orders of magnitude lower than the step transition matrix method. The RK-4 method again shows much better accuracy for non-stiff subsystems.

Figure 17 corresponds to Figure 16, except for second-order subsystems. It shows error versus damping ratio for several values of natural frequency. The damping ratio ζ and natural frequency ω_η determine the eigenvalues, and thus, the stiffness of the subsystem. In general,

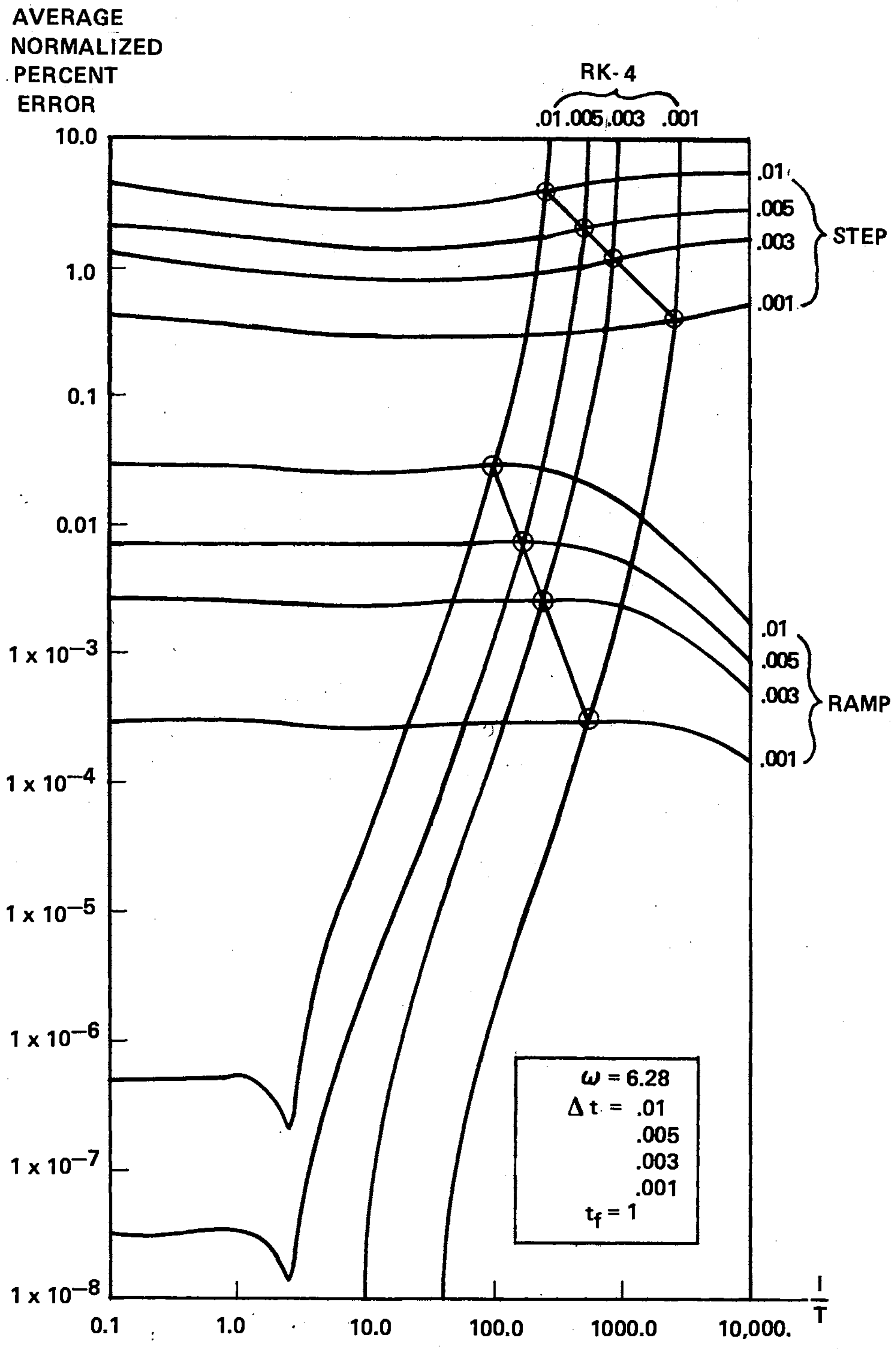

Figure 16. Error for First-Order Subsystems

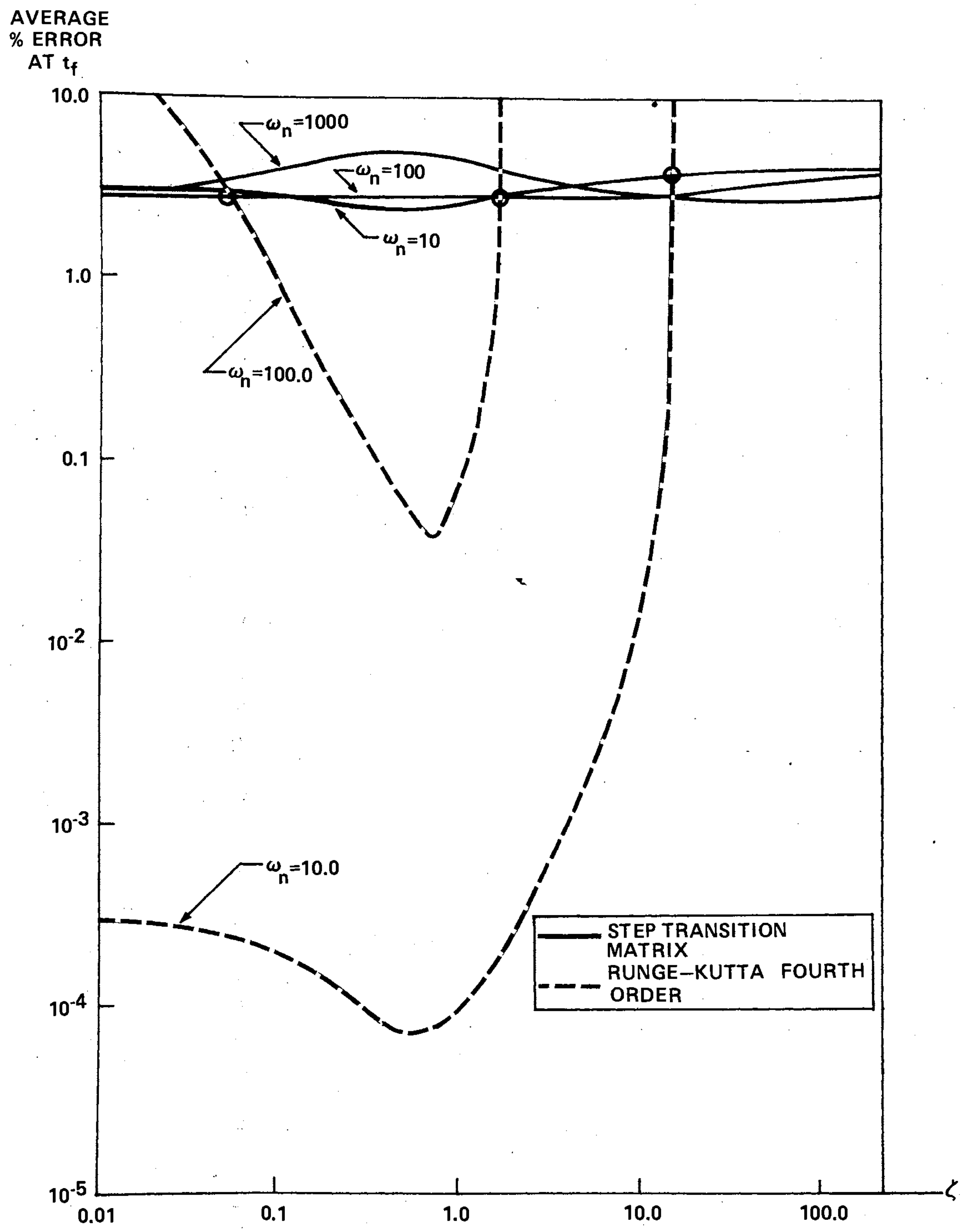

Figure 17. Error for Second-Order Subsystems

stiffness increases with both ζ and ω_η. This graph again clearly shows the accuracy of the transition matrix method relatively unaffected by the subsystem stiffness, since the error is not strongly a function of ζ or ω_η. And again, the RK-4 method's error increases sharply with the subsystem stiffness, being strongly a function of both ζ and ω_η, increasing with each. The exception to this is that for underdamped subsystems, the error increases with decreasing damping ratio, since the response becomes more oscillatory, and harder to follow.

Figure 18 is the first of several plots designed to provide guidelines for the use of the step transition matrix method. It shows lines of constant error plotted on a graph of time step versus input frequency. T = 1.0 was chosen arbitrarily for this graph. As has been seen, very similar results would obtain for other values of T. Error is a parameter on the graph. The lines shown are sets of values for input frequency and time step for which the step transition matrix has constant error. Lines are shown for several error values. To show how these lines can be used, pick any point on the 10% error line. Decreasing either the input frequency or the time step decreases the error, so any point on the graph left of the 10% line has error less than 10%. Any point right of the 10% line has more than 10% error, and any point between the 3% and 10% lines has error between 3% and 10%. This graph can be used, then, as a guide for selecting the time step. If a first-order subsystem is being simulated, and the input frequency is 1.0 rad/sec, Figure 18 dictates that a step size less than 0.075 seconds be used if 1% accuracy is desired. The graph indicates that as the input frequency decreases the allowable step size to achieve a given accuracy increases. It also provides a means of checking the hypothesis, stated earlier, that the error for the step

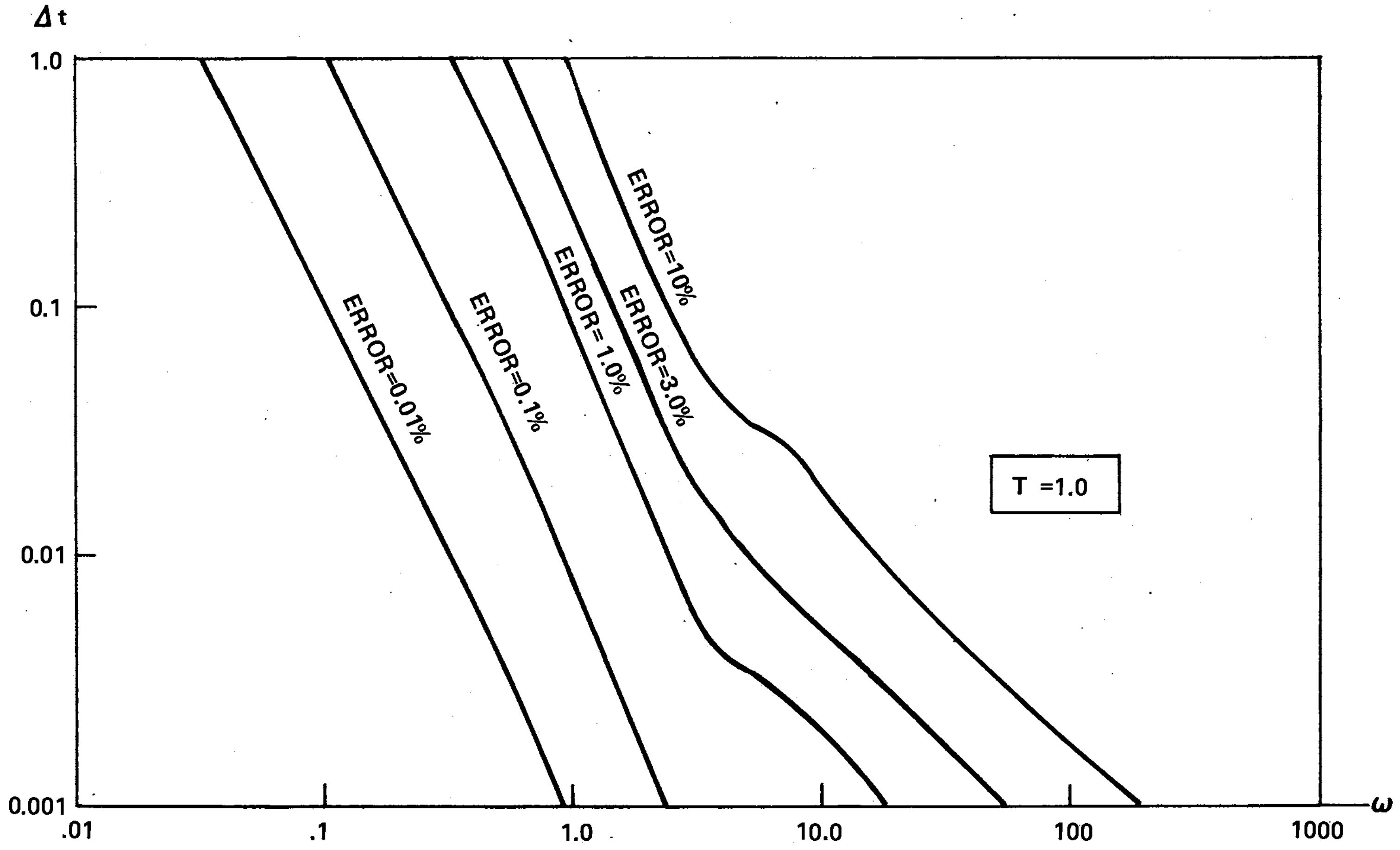

Figure 18. Constant-Error Plot for First-Order Subsystems

transition matrix method can be approximated as

$$\text{Error} = k(\Delta t)\ \omega^2 \quad .$$

Picking an input frequency of 1.0 rad/sec, and observing the time steps required to give specific error values, it can be confirmed that

$$\text{Error} \doteq 13(\Delta t)\ \omega^2$$

is a reasonable approximation for the error for a first-order subsystem for input frequencies less than 3.0 rad/sec. For higher frequencies, the approximation becomes worse, predicting error values that are too high.

Figure 19 is the second-order subsystem equivalent of Figure 18. On this graph, two lines of constant error, 1% and 10%, are shown for the RK-4 method; here, ζ and ω_η were arbitrarily picked as 10 and 100, respectively. This constitutes a stiff subsystem, and notice how small the area is beneath the RK-4 line. For any time step above 0.0014, the RK-4 method has over 10% error for any input frequency. This graph can be used in the same manner described for Figure 18. From Figure 19, the error for the step transition matrix method applied to a second-order subsystem can be approximated as

$$\text{Error} \doteq 23.8(\Delta t)\ \omega^2$$

for input frequencies less than 2.0 rad/sec. For higher input frequencies, this approximation becomes inaccurate, predicting errors that are too high.

Figure 20 deals with the first-order subsystem, and is an accuracy comparison between the step transition matrix method and the RK-4 method. Time step is plotted against 1/T with input frequency a parameter. Each

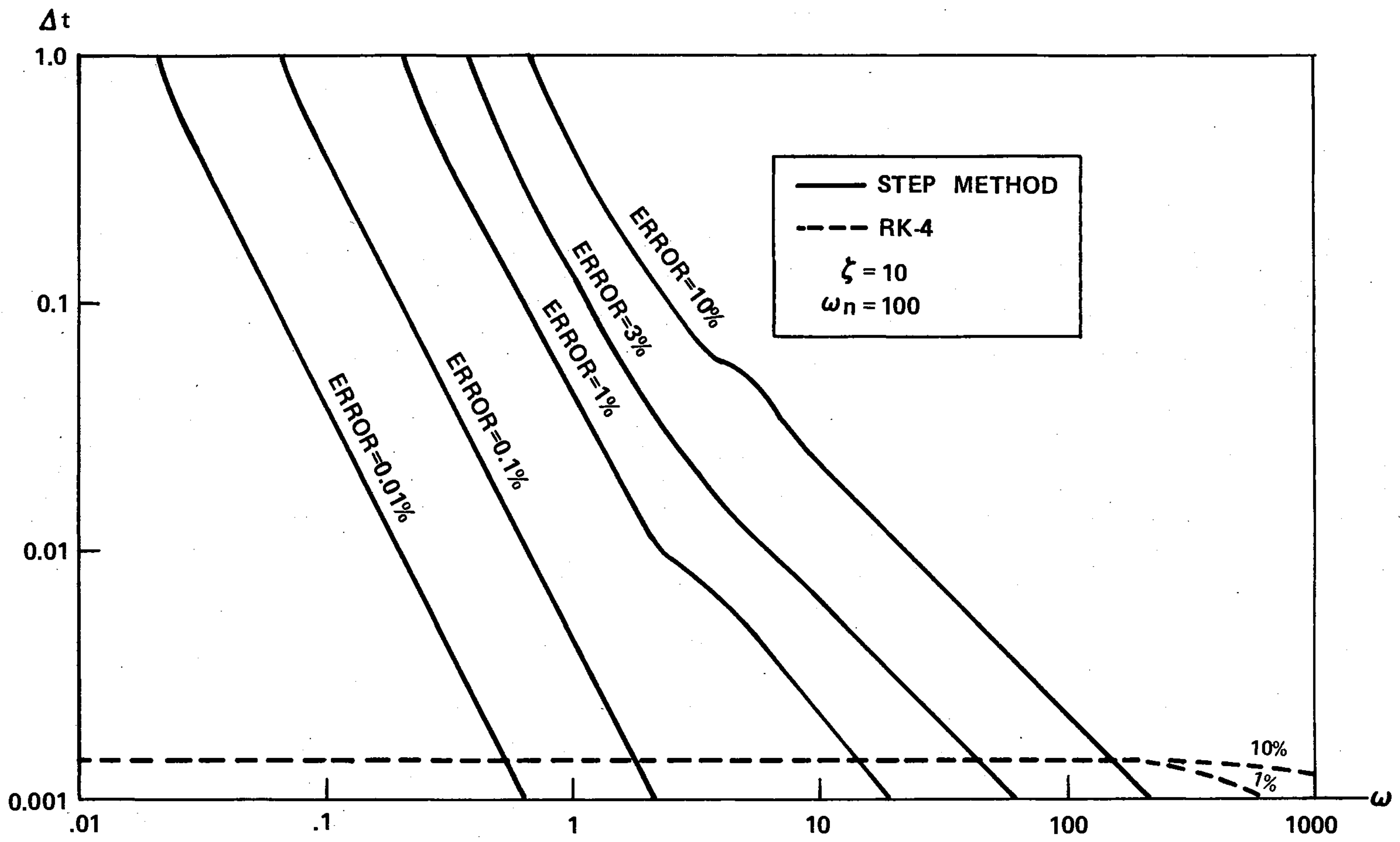

Figure 19. Constant-Error Plot for Second-Order Subsystems

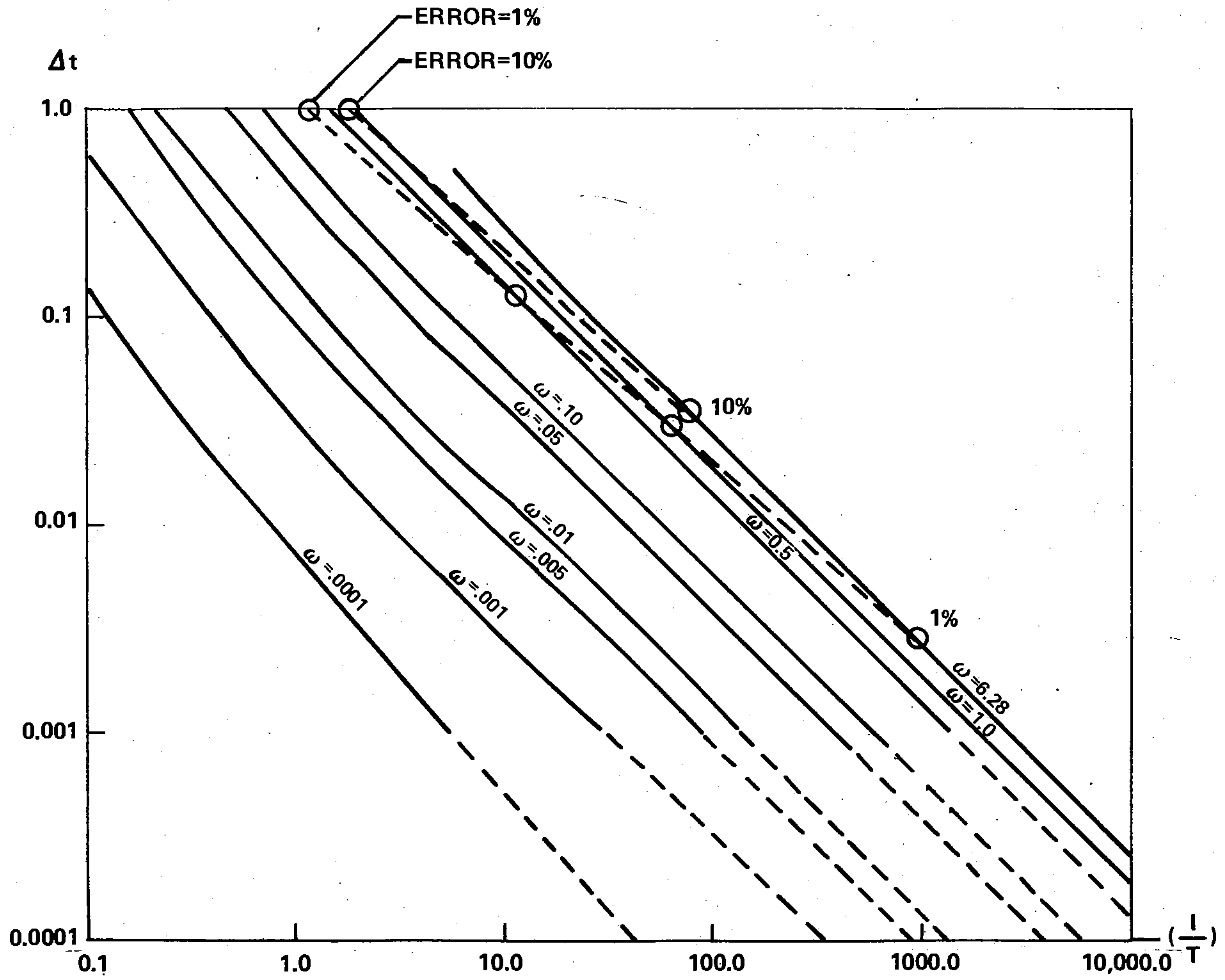

Figure 20. Equal-Error Plot for First-Order Subsystems

point on a particular line is a set of conditions for which the two methods have the same error. If a point is picked on one of the lines, say $\omega = 0.10$, increasing 1/T causes the RK-4 method to have greater error than the transition matrix method. Increasing the time step has the same result. Thus for a given input frequency, points above the line corresponding to that input frequency result in the transition matrix having lower error. For points below the line, the RK-4 method is more accurate. Therefore, Figure 20 provides a quick method of determining which method is more accurate for a given situation. For instance, if the time step is 0.01, and $1/T = 10.0$, and $\omega = 0.10$, the RK-4 method is more accurate because the point corresponding to the given Δt and 1/T falls below the $\omega = 0.1$ line. Two dashed lines of constant error are drawn on the graph to provide an idea of the errors involved. All points beneath the 1% line have error less than 1%. Observe that decreasing the input frequency moves the lines of equal error toward the origin. This means lowering the input frequency decreases the error of the transition matrix method relative to the error of the RK-4 method. Note also that as 1/T increases, the time step required by the RK-4 method to achieve equal accuracy decreases. Therefore it can be concluded that both lowering the input frequency and increasing 1/T increase the relative desirability of the transition matrix method.

Figure 21 is the second-order equivalent to Figure 20. Here, the time step is picked as 0.01, and ω_η is plotted against ζ, with input frequency a parameter. The same general comments made about Figure 20 apply here also. For each input frequency, RK-4 is more accurate for points below the line corresponding to that input frequency, and the transition matrix method is more accurate for points above the line.

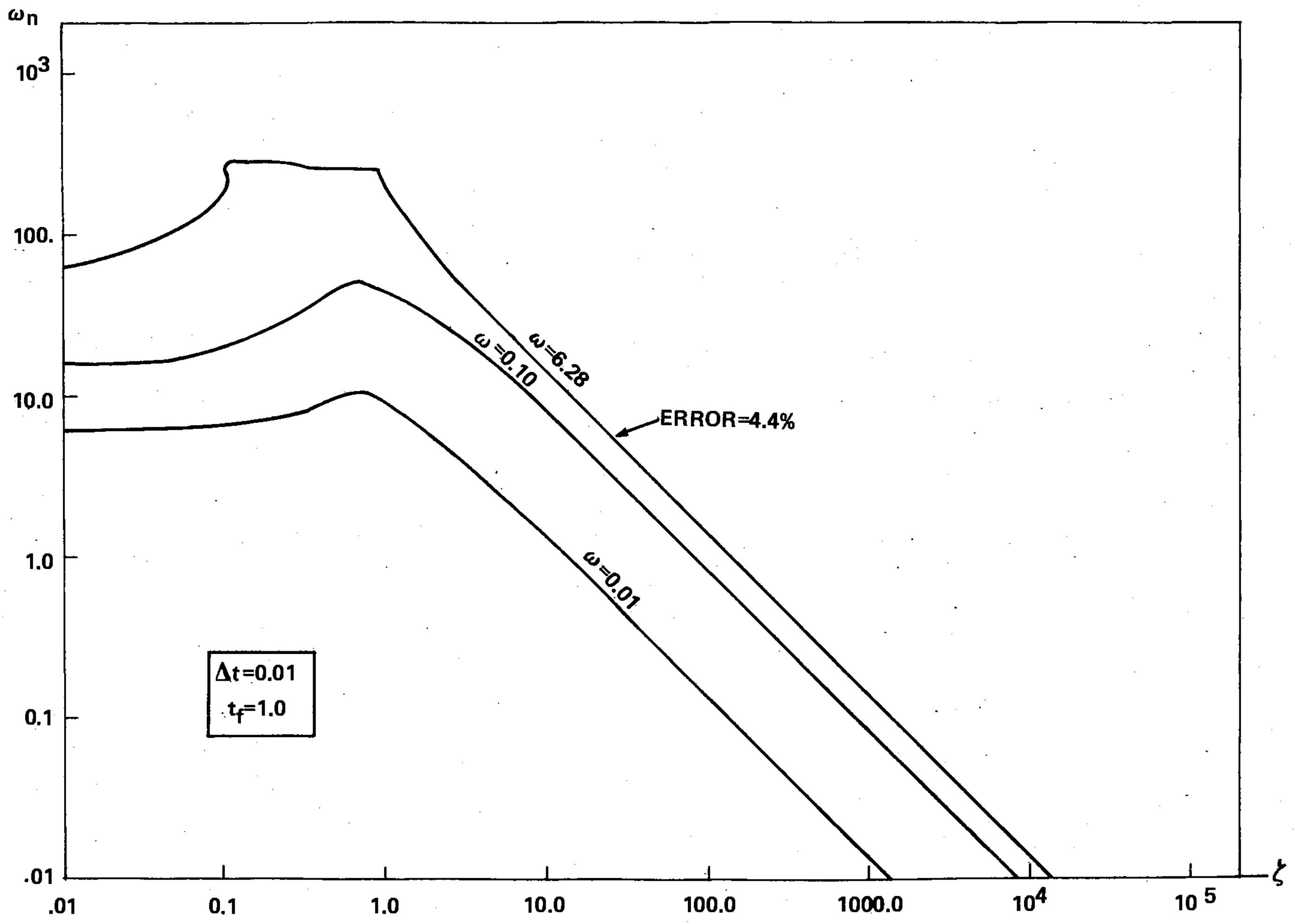

Figure 21. Equal-Error Plot for Second-Order Subsystems

Figures 22 and 23 are perhaps the most significant plots, since they are lines of equal cost. Figure 22 plots input frequency versus 1/T. Each point on the line represents a condition where the cost required to achieve 1% accuracy is the same for both RK-4, and the transition matrix. Starting at a point on the line, increasing 1/T, or decreasing the input frequency causes the RK-4 method to require more computation time than the transition matrix method to achieve a 1% accuracy solution. Thus, for all points below and to the right of the line, the transition matrix is less expensive, and for all points above and to the left of the line, the Runge-Kutta method is less expensive, for 1% error or less. Note that decreasing the input frequency, or increasing 1/T increases the desirability of the transition matrix method.

Figure 23 is the equal-cost plot for the second-order subsystem. Lines are shown for two input frequencies. ω_η is plotted versus ζ with input frequency a parameter. Again, this plot indicates that lower input frequencies and stiffer subsystems favor the transition matrix method. For all points above the lines, the transition matrix is less expensive. For all points below the lines, the Runge-Kutta method is less expensive. Both lines are for 1% error.

The computation costs for determining Figures 22 and 23 were determined by counting the number of calculations done per iteration by each method. More details on computation time estimation are given in Appendix F.

The last two graphs of the computer study, Figures 24 and 25, present the number of time steps per input cycle required to achieve a given accuracy versus input frequency. These graphs are for the transition matrix method only. Figures 24 and 25 show lines for 1% and 10% error

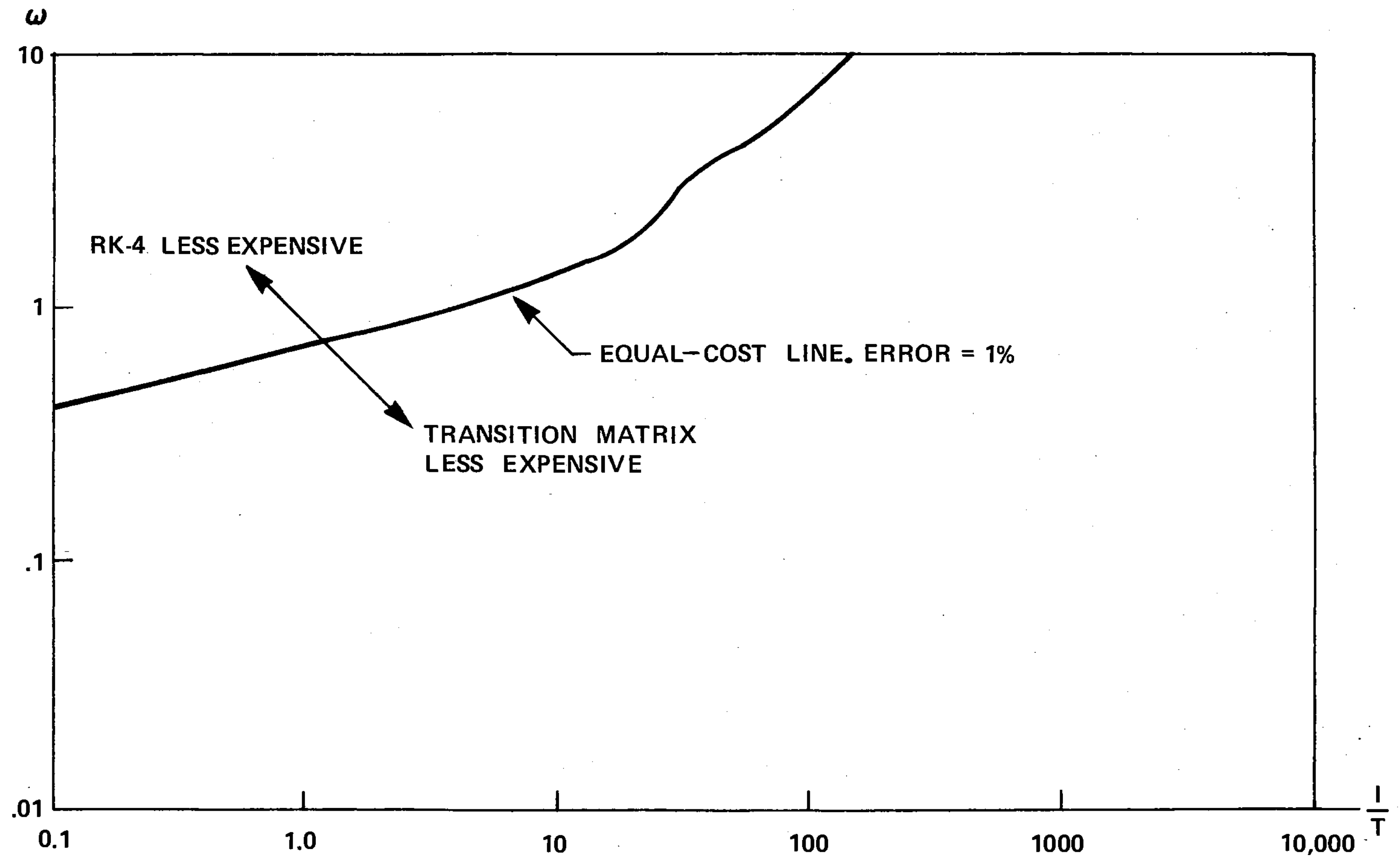

Figure 22. Equal-Cost Plot for First-Order Subsystems

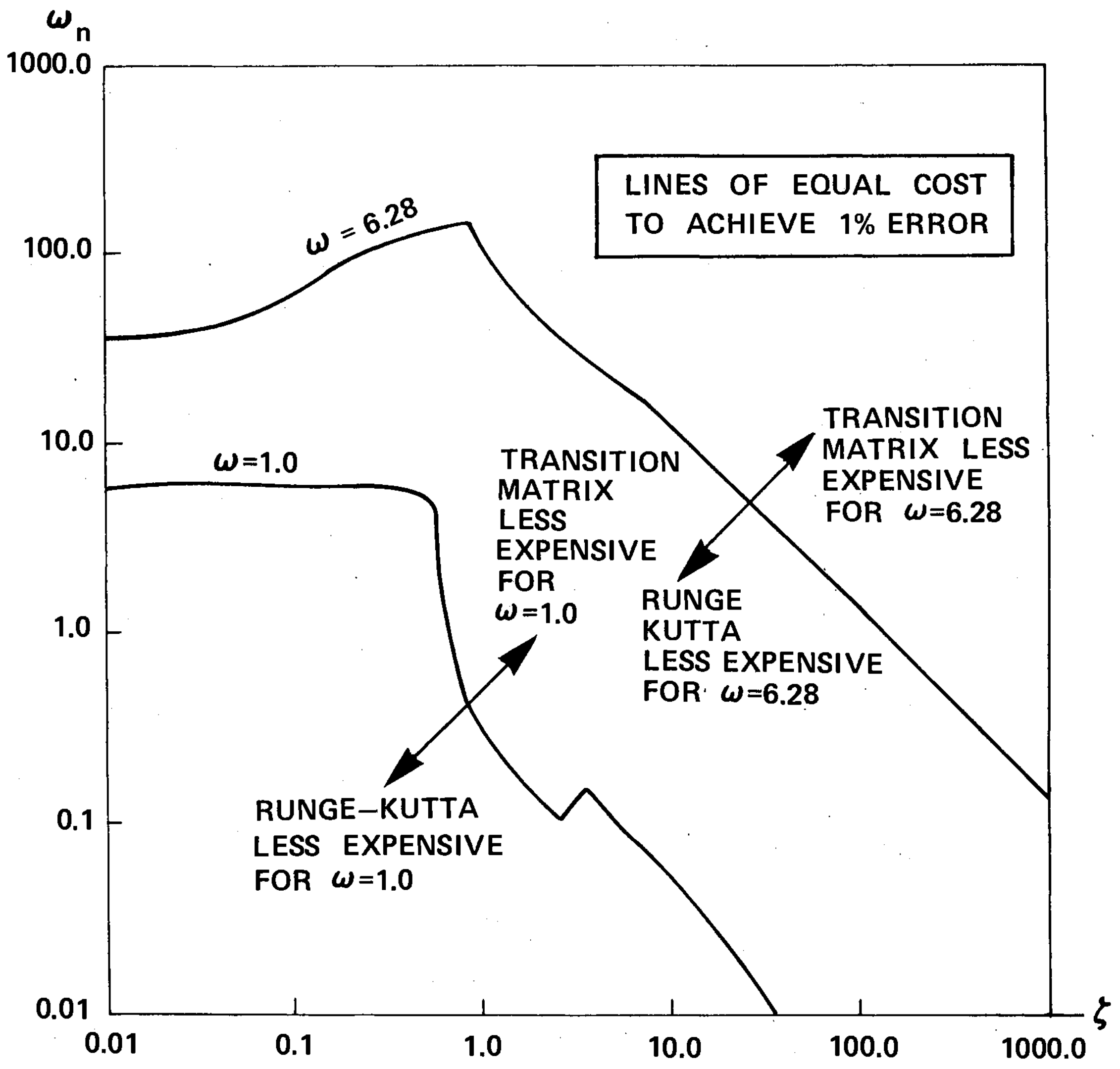

Figure 23. Equal-Cost Plot for Second-Order Subsystems

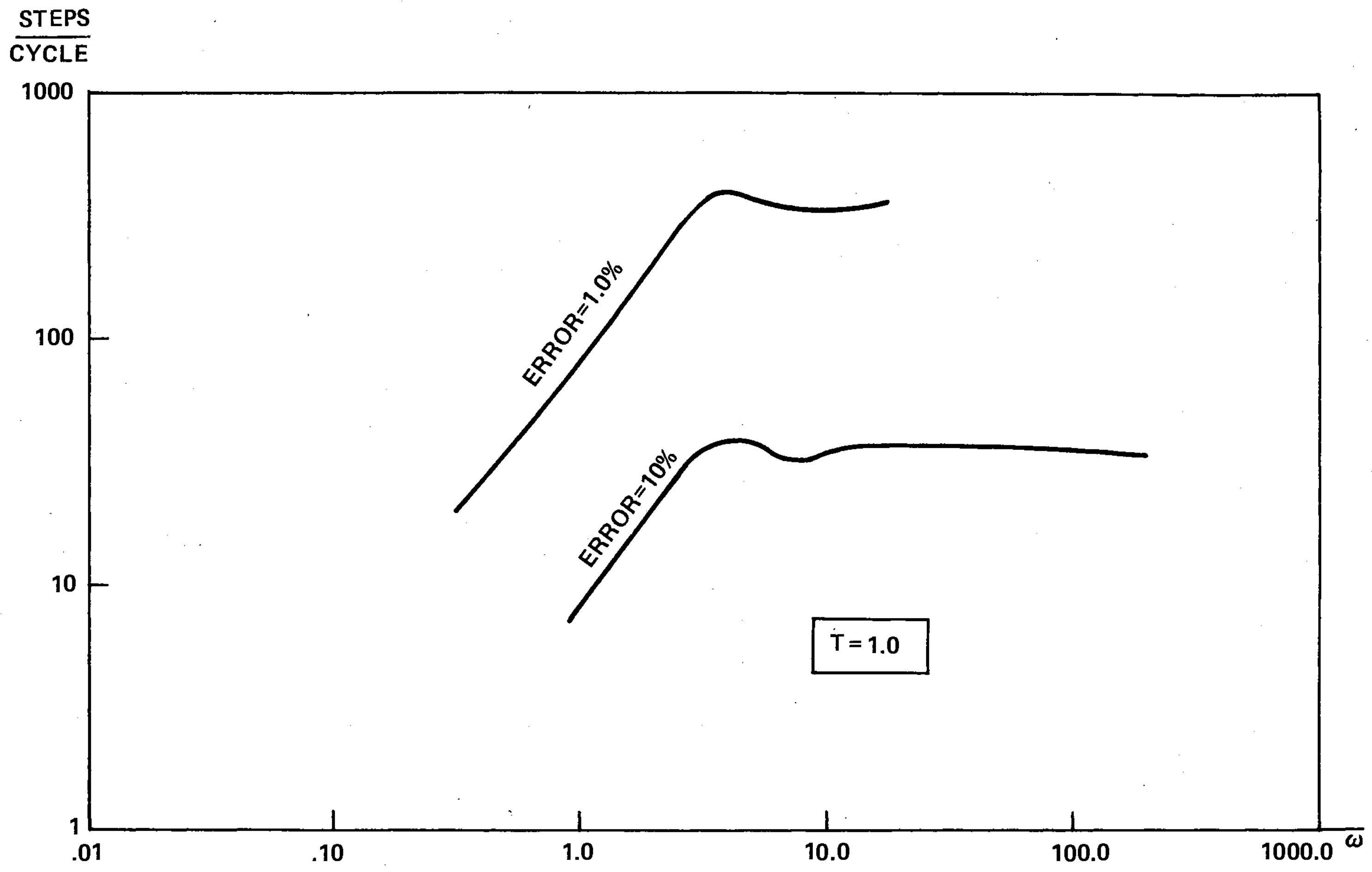

Figure 24. Steps/Cycle to Achieve a Given Accuracy for First-Order Subsystems

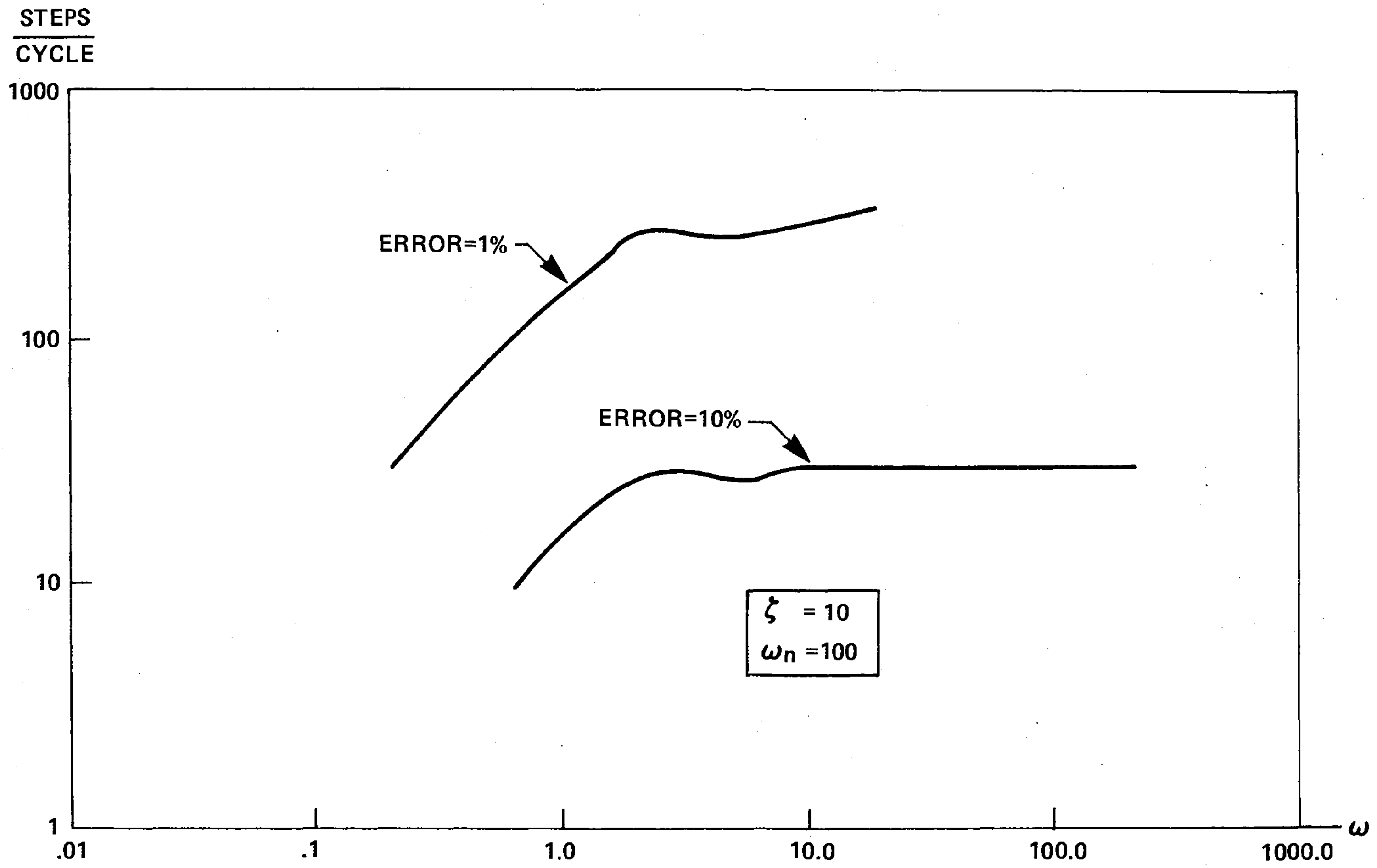

Figure 25. Steps/Cycle to Achieve a Given Accuracy for Second-Order Subsystems

for first and second-order subsystems, respectively. The two figures are remarkably similar. Note that the number of steps per cycle needed increases with input frequency up to a point, then levels off. For the first-order subsystem, a maximum of 390 steps per cycle is needed to achieve 1% accuracy, at $\omega = 4.0$ rad/sec. For the second-order subsystem, the known maximum occurs at 330 steps/cycle at $\omega = 19$ rad/sec. Again, these two graphs emphasize that increasing the input frequency degrades the performance of the step transition matrix method.

An analysis of the computations involved for the transition matrix and Runge-Kutta methods, for which details are presented in Appendix F, results in the following computation time estimates: for the same time step,

A. first-order system--step transition matrix 32% of RK-4;

B. second-order system--step transition matrix 22% of RK-4.

So it can be seen that for the same time step, the step transition matrix offers a considerable reduction in computation time.

The primary factors affecting the accuracy of the transition matrix simulation of linear subsystems are the input frequency and the step size. Results indicate the error can be approximated by

$$\text{Error} = k(\Delta t)^{n+1}\,\omega^2 \tag{14}$$

where n is the order of the input approximation used, and k is a constant. For the step transition matrix method, the constant k is approximately

$$k = 13 \text{ for first-order subsystems,}$$
$$k = 23.8 \text{ for second-order subsystems} \quad .$$

So an approximate rule for selecting the step size may be obtained by solving Equation (14) for Δt. For the step method,

$$\Delta t = \frac{\text{maximum allowable error (\%)}}{13\omega^2} \qquad \omega \leq 3 \tag{15}$$

for first-order subsystems, and

$$\Delta t = \frac{\text{maximum allowable error (\%)}}{23.8\ \omega^2} \qquad \omega \leq 2 \tag{16}$$

for second-order subsystems. These equations are rather conservative for ω larger than the specified bounds. For $\omega > 3$, Figures 24 and 25 may be used for step size selection.

The accuracy of the transition matrix method is relatively unaffected by the subsystem being simulated. For a first-order subsystem, this is equivalent to saying the accuracy is not strongly a function of the system time constant. For a second-order subsystem, the accuracy is not strongly a function of the natural frequency or the damping ratio. For a general linear time-invariant subsystem, the accuracy is not strongly a function of the pole locations of the subsystem.

In general, the transition matrix method is most appropriate when the dominant input frequency to the subsystem is low relative to the subsystem response frequencies. It is very difficult, however, to identify any quick "rules of thumb" to say when the transition matrix should be used, and when not. For instance, consider Figure 23. For $\omega = 1.0$, whenever ω_n is larger than 6.0, the transition matrix is less expensive. So postulate that when the input frequency is less than 1/6 of the natural frequency, the transition matrix method is better. But for $\omega = 6.28$, the ratio is 6.28/150, which is considerably different.

Several "stiffness numbers" were defined to see if correlations could be made with the suitability of the method. For instance, the ratio of the largest to the smallest eigenvalue, or the distance between the poles are possible stiffness numbers. None of these numbers were found to be independent parameters which could be used to predict when the method should be used in preference to a conventional method.

The transition matrix method is very stable, as would be expected since it is an analytical method. Tests indicate that the error for the method stabilizes after about 1 input cycle.

The next section presents a general procedure for applying the transition matrix approach in a complex simulation.

A General Procedure for Applying the Method

The results of the preceding subsystem error study would be of limited usefulness if they could not be applied to third or higher-order subsystems. This section presents a procedure for generalizing the results based on the dominant poles of the subsystem.

In a linear system, the low-frequency poles are called dominant, since they determine the system behavior. The high-frequency poles have only small effect. However, the high-frequency poles often may dictate the choice of solution method because of the adverse effect they have on the efficiency of conventional methods. These facts suggest a general procedure for applying the transition matrix approach. Starting with the basic physical system under study,

1. Model the system.
2. Separate the linear subsystems occurring in the model.
3. For each linear subsystem, estimate the input frequency. Considering the dominant poles of the subsystem, consult the results of the previous sections to determine whether the

transition matrix or a conventional method is more appropriate. If the results indicate the transition matrix is better suited, use it to simulate the subsystem without further consideration. If the results indicate the RK-4 method is more appropriate, consider the high frequency poles as dominant, and again consult the results. If the results here say the RK-4 method is still better, use it; otherwise use the transition matrix.

4. If possible, group together those subsystems for which the transition matrix is better, and those for which RK-4 is better. For instance, a subsystem such as shown in Figure 26 could be divided into two blocks as in Figure 27. In this case, it might be advantageous to use RK-4 to simulate block A, and the transition matrix to simulate block B.

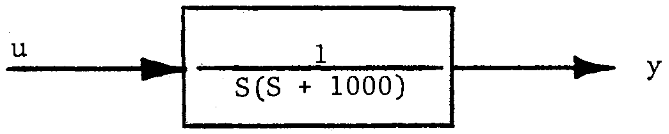

Figure 26. Block Diagram Example

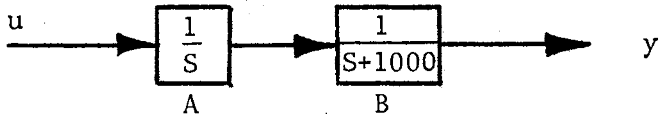

Figure 27. An Equivalent Block Diagram

5. Interface the RK-4 and the transition matrix solutions as shown in Chapter IV, and simulate the system, choosing the step size based on the linear subsystem with the highest dominant input frequency.
6. Run the solution again with a smaller step size and compare with the previous solution. If the two results agree within acceptable error, use the previous step size. If not, continue reducing the step size until the results of two consecutive runs are in acceptable agreement.

The next section demonstrates the application of this procedure on an example problem.

Example Problem

Consider the complex dynamic system shown in Figure 28. Steps 1 and 2 of the procedure just outlined have already been done since the system has been modeled, and a block diagram has been drawn showing the linear subsystems.

Step 3 requires estimating the input frequency of each linear block. By neglecting the nonlinearities and plotting the open-loop frequency response between the input and point z, the overall loop frequency for this system is found to be approximately 280 rad/sec. Note, however, that the linear blocks within the dashed box would experience much higher input frequencies, because the inputs are the responses of high-frequency blocks. It is desirable to form the augmented system for the entire closed loop linear subsystem C in order to avoid simulating the internal blocks with their high-frequency inputs. This is easily accomplished using the program RTRES, mentioned in Chapter IV.

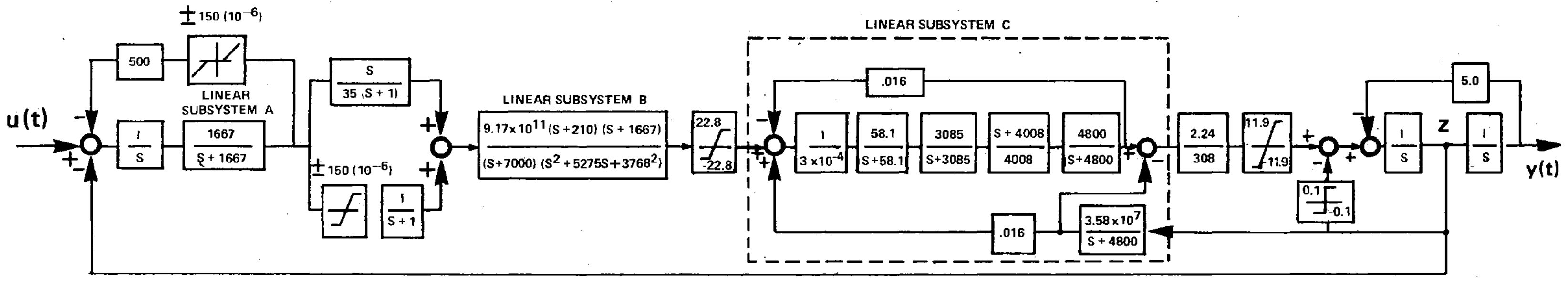

Figure 28. System for Example Problem.

Consider first the linear subsystem A. It is a first-order system with $1/T = 1667$. The time constant is

$$\frac{1}{1667} \doteq 0.0006 \text{ seconds} \quad .$$

The RK-4 method would require a step size from 6×10^{-5} seconds to 6×10^{-6} seconds. Consulting Figure 24, the number of steps per input cycle needed to achieve 1% error with the transition matrix method is estimated at 350. Assuming an input frequency of 280 rad/sec, this corresponds to a time step of 6.4×10^{-5} seconds. Since this is above the upper limit of the possible RK-4 time steps, it is better to use the transition matrix method for this block, especially since it requires less computation than RK-4 for the same time step.

Now consider linear subsystem B. The pole-zero configuration for this subsystem is shown in Figure 29. It is seen that the dominant poles are high-frequency oscillatory poles with $\zeta = 0.70$ and $\omega_\eta = 3768$. The time constant associated with these poles is

$$\frac{1}{2637.5} \doteq 0.00038 \text{ seconds}$$

which would result in a Runge-Kutta step from 4×10^{-5} to 4×10^{-6} seconds. Consulting Figure 25, it is seen that the step size needed to simulate this block with the transition matrix approach is again 6.4×10^{-5} seconds. Thus the transition matrix is more appropriate than RK-4 for this block also.

Linear subsystem C contains time constants as small as 1/4800 seconds, so the transition matrix method should be used here, also. This subsystem was handled in two pieces. The program RTRES was used to generate the transition matrix for the closed loop system, and the first

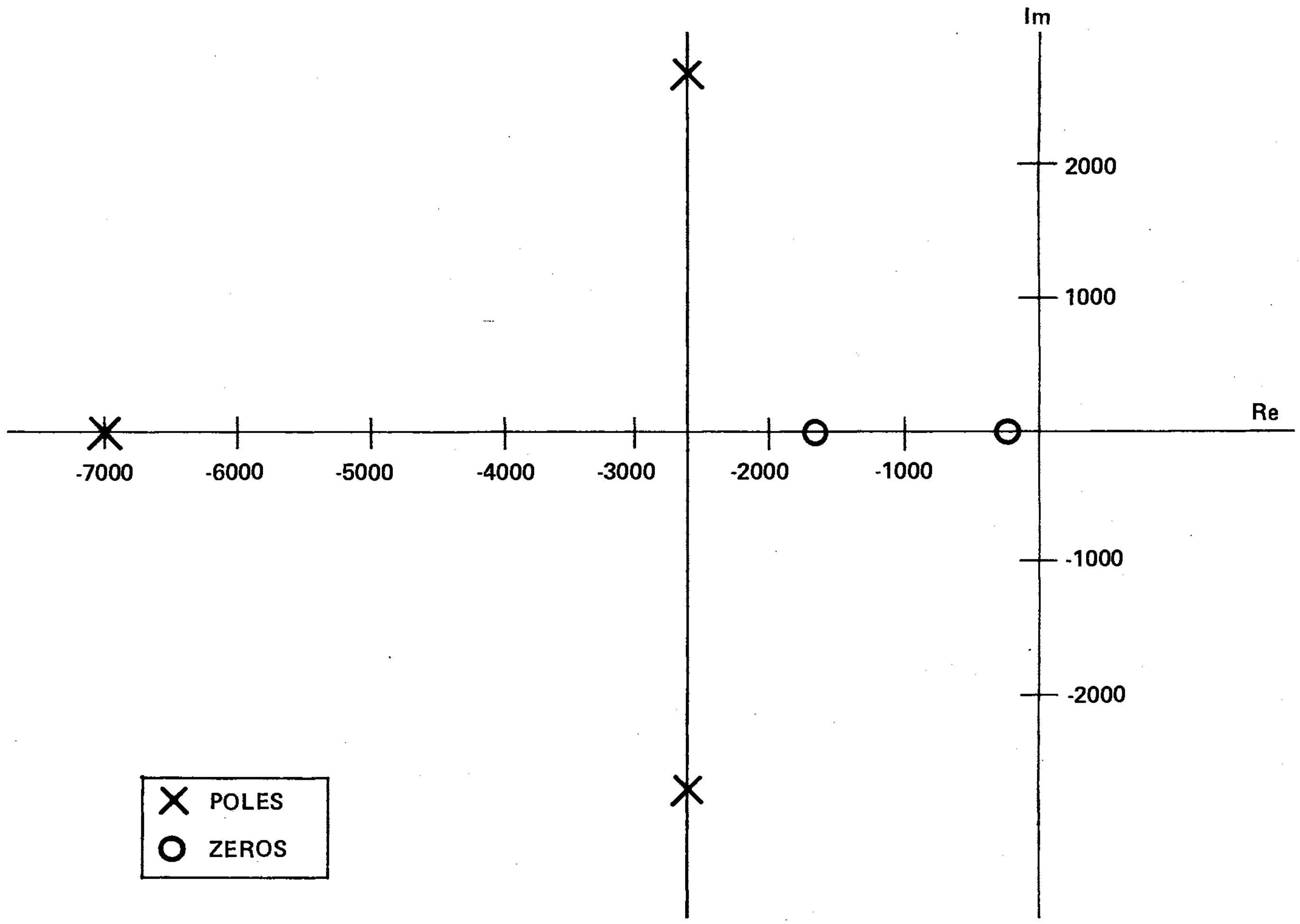

Figure 29. Pole-Zero Map for Linear Subsystem B

order block was solved directly in the coding.

With this high overall loop frequency, the RK-4 method should be used to simulate the remainder of the system, which consists of low-frequency linear blocks and nonlinearities.

To demonstrate the inefficiency of a conventional simulation of this system, the problem was first coded using the RK-4 method to simulate the entire system. Using an input frequency of 1 rad/sec, this approach required a time step of 1×10^{-8} seconds to avoid an unstable solution.

The system was then simulated using the step transition matrix to propagate the solutions of the stiff linear subsystems A, B and C. Their solutions were interfaced with the RK-4 solution of the remainder of the system as shown in Figure 10, Chapter IV. Using this combination, a time step of 0.001 seconds gave less than 2.0% error for an input frequency $\omega = 1.0$ rad/sec. This represents an increase of 5 orders of magnitude in the step size.

Discussion of Results

The example problem just considered is the type of system for which the transition matrix method is extremely well suited. Basically a nonlinear system, it contains many linear subsystems, some of which are stiff. It also contains nonlinearities which can cause abrupt changes in the inputs to the stiff parts of the system, exciting the transient responses. The use of the transition matrix method to simulate these stiff subsystems can affect a considerable savings in computation time, as noted above. Note that the linear subsystems A, B and C have dominant input frequencies that are low relative to the subsystem transient frequencies. It is important to choose the time step for each linear subsystem according to the dominant input frequency. In a system such as Figure 28, the subsystem inputs will contain high frequency transients. If the time step for the subsystem solution is chosen to follow these transients, there would be no advantage to the method. By choosing a step size based on the dominant input frequency, the sampling of the input acts to filter the high frequency transients from the subsystem input.

In this sense, the transition matrix method acts to reduce the adverse effect of high-frequency components on the simulation efficiency, while leaving these components in the model. The effects of the high-frequency components can be included or excluded as the step size is decreased or increased, respectively. This feature of the method suggests its use as a tool for simplifying a complex model, since the effect on the overall system performance of leaving out certain high-frequency components can be easily studied.

It is important that the high-frequency blocks and the low-frequency blocks be grouped together whenever possible. Low-frequency blocks should not be propagated by the transition matrix since higher accuracy can be achieved using conventional methods.

In some cases, the stiffness of the system may be due to nonlinear differential equations. If a very small step size is necessitated, it may be more economical to use the transition matrix for all of the linear blocks, since for the same time step, it requires less computation than RK-4.

In general, more efficient simulation can be achieved when the method and the step size are chosen to suit the particular part of the system being simulated. However, such an approach would be extremely problem dependent, and would require careful implementation.

The general procedure outlined above is intended only as a guide for the application of the method. Since it is impossible to consider all the special cases that could arise, engineering judgment should be applied to each problem to obtain an efficient implementation of the method.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

A method for simulating stiff dynamic systems has been evaluated, and a general procedure for using the method has been presented. The method is well suited for systems having stiff linear subsystems whose transients are high frequency relative to the dominant input frequency for the subsystem. In such cases, step size increases of two orders of magnitude or more are possible over conventional methods.

The computer studies and example problem show the transition matrix approach to be a viable and relatively simple technique. Considering the number of simulations in which stiff linear subsystems arise, this method should find wide application.

Recommendations for Further Study

The following areas merit further investigation.

1. The transition matrix method concentrated on in this study utilized a zero-order input approximation. Preliminary results indicate a ramp input approximation gives much improved accuracy for a small increase in computation time (see Appendix F). The program RTRES, mentioned in Chapter IV, should be modified to handle repeated eigenvalues in the augmented system. This would permit the easy implementation of more sophisticated input approximations for the method. These approximations could then be studied for accuracy and computation time.

2. Various methods for computing the transition matrix for the augmented system should be investigated. The accuracy of this matrix is critical to the accuracy of the method.

3. The accuracy of the method studied here was much lower than RK-4 for non-stiff subsystems. A higher-order transition matrix method might be developed which would have accuracy superior to numerical methods for any linear subsystem, stiff or not. This method could then be used to simulate all linear subsystems in a simulation. It is felt that decreased computation time would also result from using such a method.

4. The system-size limitations for the method should be investigated. The method as presented here is aimed at relatively low-order subsystems. The maximum-order subsystem that this method will handle is probably dependent on how large the augmented matrix can be without causing difficulties in finding the Φ matrix. This may also be a limitation on using higher-order input approximations, since the size of the augmented system increases as the order of the input approximation increases.

5. Higher-order input approximations might make the method tend toward instability for certain inputs. This area could be investigated.

6. The use of the method as a tool for simplifying mathematical models, as discussed in Chapter V, should be studied further.

BIBLIOGRAPHY

(1) Andrus, J. F. "Integration of Control Equations and the Problem of Small Time Constants." NASA Technical Note D-3907, April, 1967.

(2) Beckett, R., and J. Hurt. Numerical Calculations and Algorithms. New York: McGraw-Hill Book Co., 1967.

(3) Benyon, P. R. "A Review of Numerical Methods for Digital Simulation." Simulation, Vol. 11 (1968), 219-238.

(4) Curtiss, C. F., and J. O. Hirschfelder. "Integration of Stiff Equations." Proc. of Nat. Acad. of Sci., Vol. 38 (1952), 235-243.

(5) Ebbesen, L. R. "A Variational Method for the Simulation of Systems With Diverse Frequencies." (Unpub. M. S. thesis, Oklahoma State University, 1970).

(6) Kuo, B. C. Discrete Data Control Systems. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1970.

(7) Melsa, J. L., and S. K. Jones. Computer Programs for Computational Assistance in the Study of Linear Control Theory. New York: McGraw-Hill Book Co., 1970.

(8) Sebesta, H. R. Studies on Advanced Simulation Concepts. Oklahoma State University, School of Mechanical and Aerospace Engineering, Research Report No. ER-73-R-22, p. C1, "User Guide for RTRES".

(9) Stineman, R. W. "Digital Time-Domain Analysis of Systems With Widely Separated Poles." J. Assoc. Comp. Mach., Vol. 12 (1965), 286-293.

(10) Treanor, C. E. "A Method for the Numerical Integration of Coupled First Order Differential Equations With Greatly Different Time Constants." Math. of Comp., Vol. 20 (1966), 39-45.

(11) Walters, Charles M. "A Discussion of Gear's Implementation of Adams and Stiff Methods for Solving Ode's." Technical Report No. AFWL-TR-72-170, Air Force Weapons Laboratory, Air Force System's Command, Kirtland Air Force Base, New Mexico (Nov., 1972).

APPENDIX A

ELEMENTS OF $\Phi(\Delta t)$ FOR A SECOND-ORDER SYSTEM

Case I. $0 \leq \zeta < 1$ (underdamped system)

$$\phi_{11} = \frac{-1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta \Delta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, \Delta t - \phi) + \frac{2\zeta}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta \Delta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, \Delta t)$$

$$\phi_{12} = \frac{1}{\omega_\eta \sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta \Delta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, \Delta t)$$

$$\phi_{13} = 1 - \frac{1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta \Delta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, \Delta t + \phi)$$

$$\phi_{21} = \frac{-\omega_\eta}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta \Delta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, \Delta t)$$

$$\phi_{22} = \frac{-1}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta \Delta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, \Delta t - \phi)$$

$$\phi_{23} = \frac{\omega_\eta}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta \Delta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, \Delta t)$$

$$\phi = \tan^{-1} \left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right)$$

Case II. $\zeta = 1$ (critically-damped system)

$$\phi_{11} = e^{-\omega_\eta \Delta t} (\omega_\eta \Delta t + 1)$$

$$\phi_{12} = \Delta t \; e^{-\omega_\eta \Delta t}$$

$$\phi_{13} = 1 - e^{-\omega_\eta \Delta t} (1 + \omega_\eta \Delta t)$$

$$\phi_{21} = -\omega_\eta^2 \Delta t \; e^{-\omega_\eta \Delta t}$$

$$\phi_{22} = e^{-\omega_\eta \Delta t}(1 - \omega_\eta \Delta t)$$

$$\phi_{23} = \omega_\eta^2 \Delta t \, e^{-\omega_\eta \Delta t}$$

Case III. $\zeta > 1$ (overdamped system)

$$\phi_{11} = \frac{1}{2}\left[1 - \frac{\zeta}{\sqrt{\zeta^2-1}}\right] e^{-(\zeta-\sqrt{\zeta^2-1})\omega_\eta \Delta t} + \frac{1}{2}\left[1 + \frac{\zeta}{\sqrt{\zeta^2-1}}\right] e^{-(\zeta+\sqrt{\zeta^2-1})\omega_\eta \Delta t}$$
$$+ \frac{\zeta}{\sqrt{\zeta^2-1}}\left[e^{-(\zeta-\sqrt{\zeta^2-1})\omega_\eta \Delta t} - e^{-(\zeta+\sqrt{\zeta^2-1})\omega_\eta \Delta t}\right]$$

$$\phi_{12} = \frac{1}{2\omega_\eta\sqrt{\zeta^2-1}}\left[e^{-(\zeta-\sqrt{\zeta^2-1})\omega_\eta \Delta t} - e^{-(\zeta+\sqrt{\zeta^2-1})\omega_\eta \Delta t}\right]$$

$$\phi_{13} = 1 + \frac{1}{2[\zeta^2-1-\zeta\sqrt{\zeta^2-1}]} e^{-\omega_\eta[\zeta-\sqrt{\zeta^2-1}]\Delta t} + \frac{e^{-\omega_\eta[\zeta+\sqrt{\zeta^2-1}]\Delta t}}{2[\zeta^2-1+\zeta\sqrt{\zeta^2-1}]}$$

$$\phi_{21} = -\frac{\omega_\eta}{2\sqrt{\zeta^2-1}}\left[e^{-(\zeta-\sqrt{\zeta^2-1})\omega_\eta \Delta t} - e^{-(\zeta+\sqrt{\zeta^2-1})\omega_\eta \Delta t}\right]$$

$$\phi_{22} = \frac{1}{2}\left[1 - \frac{\zeta}{\sqrt{\zeta^2-1}}\right] e^{-\omega_\eta(\zeta-\sqrt{\zeta^2-1})\Delta t} + \frac{1}{2}\left[1 + \frac{\zeta}{\sqrt{\zeta^2-1}}\right] e^{-\omega_\eta(\zeta+\sqrt{\zeta^2-1})\Delta t}$$

$$\phi_{23} = \frac{\omega_\eta}{2\sqrt{\zeta^2-1}}\left[e^{-(\zeta-\sqrt{\zeta^2-1})\omega_\eta \Delta t} - e^{-(\zeta+\sqrt{\zeta^2-1})\omega_\eta \Delta t}\right]$$

For each case,

$$\phi_{31} = 0$$

$$\phi_{32} = 0$$

$$\phi_{33} = 1 \quad .$$

APPENDIX B

FACTORS AFFECTING THE RESULTS OF MACHINE COMPUTATIONS

In studying any numerical method, two results are of prime importance: the accuracy of the results and the computation time. Both of these performance measures are affected by the following factors:

(a) the particular computer being used;

(b) the word length; and

(c) the particular coding used to program the problem.

Factor (a) often has a strong effect on both accuracy and run-time, since computers vary considerably in computation speed and word length. Factor (b) determines the number of digits the computer can carry through a calculation, and therefore affects the roundoff error. Calculations done on machines with long word lengths, and calculations done in double precision, which doubles the normal word length, have much reduced roundoff error. Factor (c) often has a strong effect on the roundoff error in a calculation. Subtraction of a number from a nearly equal number, and addition of two numbers of widely different magnitude are two examples of calculations which lead to large roundoff errors due to the finite word length of the machine. Such calculations are called "ill-conditioned". These problems can often be made less severe by algebraic rearrangement of the expression being evaluated. For example, the expression

$$1 - \frac{\zeta}{\sqrt{\zeta^2-1}}$$

results in a large roundoff error when ζ is large, since the term on the right approaches 1. An algebraic manipulation converts the expression to

$$\frac{1-\zeta^2}{(\zeta^2-1)^2 + \zeta(\zeta^2-1)^{3/2}}$$

which avoids the subtraction of nearly equal numbers, and is more accurate for large ζ.

The computer investigations in this study were done on an IBM 360 model 65 computer. All calculations were done in double precision, which on this computer gives a 64 bit word length. Ill-conditioned calculations, such as the example above, were avoided by algebraic rearrangement, when possible.

APPENDIX C

EXACT SOLUTION FOR SECOND-ORDER SYSTEM

Case I. $0 \leq \zeta < 1$ (underdamped)

$$y(t) = \frac{-A}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, t - \phi) + \frac{B}{\omega_\eta \sqrt{1-\zeta^2}} e^{-\zeta\omega_\eta t} \sin(\omega_\eta \sqrt{1-\zeta^2}\, t)$$

$$+ C \cos(\omega t) + \frac{D}{\omega} \sin(\omega t)$$

Case II. $\zeta = 1$ (critically damped)

$$y(t) = Ae^{-\omega_\eta t}(1 - \omega_\eta t) + Bte^{-\omega_\eta t} + C \cos(\omega t) + \frac{D}{\omega} \sin(\omega t)$$

Case III. $\zeta > 1$ (overdamped)

$$y(t) = \frac{A}{2}\left[1 - \frac{\zeta}{\sqrt{\zeta^2-1}}\right] e^{-\omega_\eta(\zeta-\sqrt{\zeta^2-1})t} + \frac{A}{2}\left[1 + \frac{\zeta}{\sqrt{\zeta^2-1}}\right] e^{-\omega_\eta(\zeta+\sqrt{\zeta^2-1})t}$$

$$+ \frac{B}{2\omega_\eta\sqrt{\zeta^2-1}}\left[e^{-\omega_\eta(\zeta-\sqrt{\zeta^2-1})t} - e^{-\omega_\eta(\zeta+\sqrt{\zeta^2-1})t}\right]$$

$$+ C \cos(\omega t) + \frac{D}{\omega} \sin(\omega t)$$

where

$$\phi = \tan^{-1}\left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right)$$

$$A = \frac{-\omega_\eta^2(\omega_\eta^2 - \omega^2)}{(\omega_\eta^2 - \omega^2)^2 + 4\zeta^2\omega_\eta^2\omega^2}$$

$$B = \frac{-2\zeta\omega_\eta^5}{(\omega_\eta^2 - \omega^2)^2 + 4\zeta^2\omega_\eta^2\omega^2}$$

$$C = -A$$

$$D = \frac{2\zeta\omega^2\omega_\eta^3}{(\omega_\eta^2 - \omega^2)^2 + 4\zeta^2\omega_\eta^2\omega^2}$$

APPENDIX D

ERROR ANALYSIS PROGRAMS

```
C     ******************************************************************
C     *  ERROR EVALUATION PROGRAM FOR A GENERAL FIRST ORDER SYSTEM.     *
C     ******************************************************************
C     ******************************************************************
C     *  VARIABLE DEFINITIONS AND EXPLANATIONS                         *
C     ******************************************************************
C     *  AMPL IS THE RMS AVERAGE OF THE RESPONSE AMPLITUDE.            *
C     *  APESTP IS THE AVERAGE PERCENTAGE ERROR FOR THE STEP METHOD.   *
C     *  APERMP IS THE AVERAGE PERCENTAGE ERROR FOR THE RAMP METHOD.   *
C     *  APERK4 IS THE AVERAGE PERCENTAGE ERROR FOR THE RK-4 METHOD.   *
C     *  DELT IS THE TIME STEP.                                        *
C     *  ERSTEP IS THE ABSOLUTE ERROR FOR THE STEP METHOD.             *
C     *  ERRAMP IS THE ABSOLUTE ERROR FOR THE RAMP METHOD.             *
C     *  ERRK4 IS THE ABSOLUTE ERROR FOR THE RK-4 METHOD.              *
C     *  EXACT IS THE EXACT SOLUTION.                                  *
C     *  OMEGA IS THE INPUT FREQUENCY.                                 *
C     *  PESTPN IS THE NORMALIZED PERCENTAGE ERROR FOR THE STEP METHOD. *
C     *  PERMPN IS THE NORMALIZED PERCENTAGE ERROR FOR THE RAMP METHOD. *
C     *  PERK4N IS THE NORMALIZED PERCENTAGE ERROR FOR THE RK-4 METHOD. *
C     *  PESSTP IS THE SUM OF THE PERCENTAGE ERROR FOR THE STEP METHOD. *
C     *  PESRMP IS THE SUM OF THE PERCENTAGE ERROR FOR THE RAMP METHOD. *
C     *  PESRK4 IS THE SUM OF THE PERCENTAGE ERROR FOR THE RK-4 METHOD. *
C     *  R IS THE INPUT.                                               *
C     *  RAMP IS THE RAMP TRANSITION MATRIX SOLUTION.                  *
C     *  RK4 IS THE RUNGE-KUTTA FOURTH ORDER SOLUTION.                 *
C     *  STEP IS THE STEP TRANSITION MATRIX SOLUTION.                  *
C     *  T IS THE SYSTEM TIME CONSTANT.                                *
C     *  TIME IS CURRENT TIME.                                         *
C     *  TNITER IS THE TOTAL NUMBER OF ITERATIONS.                     *
C     ******************************************************************
C     *  CALCULATION OF CONSTANTS USED IN ITERATIONS.                  *
C     ******************************************************************
      T1=OMEGA*T
      T9=DELT/2.D0
      T41=T1*T1
      T42=1.D0+T41
      T43=DSQRT(T42)
      T44=1.D0/T42
      T47=-DELT/T
      T48=DEXP(T47)
      T7=T*T48
      T49=1.D0-T48
      T50=DELT-T+T7
      T51=-T47
      T52=T51/2.D0
      T53=1.D0-T51*(1.D0-T52*(1.D0-T52))
      T54=4.D0-T51*(2.D0-T52)
      T55=6.D0-T51*(3.D0-T51*(1.D0-T52/2.D0))
      T56=T51/6.D0
      AMP=.707D-2/T43
      AMPL=.707D0/T43
C     ******************************************************************
C     *  BEGIN ITERATION.                                              *
C     ******************************************************************
    2 TM1=TIME
      TIME=ITIME*DELT
      TNITER=TNITER+1.D0
      RSTOR=R
      R=DCOS(T14)
      T14=OMEGA*TIME
      T45=-TIME/T
      T46=-DEXP(T45)
      T4=DCOS(T14)
      T5=DSIN(T14)
      T6=T1*T5
C     ******************************************************************
C     *  CALCULATE EXACT SOLUTION.                                     *
C     ******************************************************************
      EXACT=T44*(T46+T4+T6)
C     ******************************************************************
C     *  CALCULATE STEP TRANSITION MATRIX SOLUTION.                    *
C     ******************************************************************
      STEP=T48*STEP+T49*RSTOR
C     ******************************************************************
C     *  CALCULATE RAMP TRANSITION MATRIX SOLUTION.                    *
C     ******************************************************************
      SLOPE=(R-RSTOR)/DELT
      RAMP=T48*RAMP+T49*RSTOR+T50*SLOPE
C     ******************************************************************
C     *  CALCULATE RUNGE-KUTTA FOURTH ORDER SOLUTION.                  *
C     ******************************************************************
      TEMPT1=TM1+T9
      R1=DCOS(OMEGA*TEMPT1)
      RK4=RK4+T56*(R+T53*RSTOR+T54*R1-T55*RK4)
C     ******************************************************************
C     *  CALCULATE ERROR VALUES.                                       *
C     ******************************************************************
      ERSTEP=DABS(EXACT-STEP)
      ESSTEP=ESSTEP+ERSTEP
      PESTPN=ERSTEP/AMP
      PESSTP=PESSTP+PESTPN
      APESTP=PESSTP/TNITER
      ERRAMP=DABS(EXACT-RAMP)
      PERMPN=ERRAMP/AMP
      PESRMP=PESRMP+PERMPN
      APERMP=PESRMP/TNITER
      ERRK4=DABS(EXACT-RK4)
      PERK4N=ERRK4/AMP
      PESRK4=PESRK4+PERK4N
      APERK4=PESRK4/TNITER
      IF(TIME.GE.TFINAL) GO TO 8
    8 CONTINUE
      GOTO2
C     ******************************************************************
C     *  END ITERATION.                                                *
C     ******************************************************************
```

```
C     ******************************************************************
C     *  ERROR EVALUATION PROGRAM FOR A GENERAL SECOND ORDER SYSTEM.   *
C     ******************************************************************
C     ******************************************************************
C     *  VARIABLE DEFINITIONS AND EXPLANATIONS.                        *
C     ******************************************************************
C     *  AMPL IS THE RMS AVERAGE OF THE RESPONSE AMPLITUDE.            *
C     *  APESTP IS THE AVERAGE PERCENTAGE ERROR FOR THE STEP METHOD.   *
C     *  APERMP IS THE AVERAGE PERCENTAGE ERROR FOR THE RAMP METHOD.   *
C     *  APERK4 IS THE AVERAGE PERCENTAGE ERROR FOR THE RK-4 METHOD.   *
C     *  DELT IS THE TIME STEP.                                        *
C     *  ERSTEP IS THE ABSOLUTE ERROR FOR THE STEP METHOD.             *
C     *  ERRAMP IS THE ABSOLUTE ERROR FOR THE RAMP METHOD.             *
C     *  ERRK4 IS THE ABSOLUTE ERROR FOR THE RK-4 METHOD.              *
C     *  EXACT IS THE EXACT SOLUTION.                                  *
C     *  OMEGA IS THE INPUT FREQUENCY.                                 *
C     *  OMEGAN IS THE NATURAL FREQUENCY.                              *
C     *  PESTPN IS THE NORMALIZED PERCENTAGE ERROR FOR THE STEP METHOD. *
C     *  PERMPN IS THE NORMALIZED PERCENTAGE ERROR FOR THE RAMP METHOD. *
C     *  PERK4N IS THE NORMALIZED PERCENTAGE ERROR FOR THE RK-4 METHOD. *
C     *  PESSTP IS THE SUM OF THE PERCENTAGE ERROR FOR THE STEP METHOD. *
C     *  PESRMP IS THE SUM OF THE PERCENTAGE ERROR FOR THE RAMP METHOD. *
C     *  PESRK4 IS THE SUM OF THE PERCENTAGE ERROR FOR THE RK-4 METHOD. *
C     *  PHIR11, ETC. ARE THE ELEMENTS OF THE TRANSITION MATRIX FOR THE *
C     *       RAMP INPUT APPROXIMATION METHOD.                         *
C     *  R IS THE INPUT.                                               *
C     *  RAMP1 AND RAMP2 ARE THE TWO STATE VARIABLES USED BY THE RAMP  *
C     *       INPUT APPROXIMATION METHOD.  RAMP1 IS THE SOLUTION.      *
C     *  RK41 AND RK42 ARE THE TWO STATE VARIABLES USED BY THE RK-4    *
C     *       METHOD.  RK41 IS THE SOLUTION.                           *
C     *  STEP1 AND STEP2 ARE THE TWO STATE VARIABLES FOR THE STEP INPUT *
C     *       APPROXIMATION METHOD.  STEP1 IS THE SOLUTION.            *
C     *  TIMF IS CURRENT TIME.                                         *
C     *  TNITER IS THE TOTAL NUMBER OF ITERATIONS.                     *
C     *  ZETA IS THE DAMPING RATIO.                                    *
C     ******************************************************************
C     *  CALCULATION OF CONSTANTS USED IN ITERATIONS.                  *
C     *  PRELIMINARY CALCULATIONS ARE DIVIDED INTO FOUR SETS.          *
C     ******************************************************************
C     *  SET  I.  DEFINED FOR ALL VALUES OF ZETA.                      *
C     ******************************************************************
      T2=OMEGA*OMEGA
      T9=DELT/2.D0
      T60=ZETA*OMEGAN
      T61=T60*DELT
      T62=DEXP(-T61)
      T63=ZETA*ZETA
      T64=(1.D0-ZETA)*(1.D0+ZETA)
      T94=OMEGAN*DELT
      T95=T94+1.D0
      T96=DELT*T62
      T98=OMEGAN*OMEGAN
      T99=T98-T2
      T97=T98*T96
      T102=T99*T99+4.D0*T63*T98*T2
      T101=T98*T99/T102
      T103=2.D0*ZETA*OMEGA*T98*OMEGAN/T102
      T104=-2.D0*ZETA*T98*T98/T102
      T120=-2.D0*T60
      T200=T62*T95
      T201=1.D0-T200
      T202=T62*(1.D0-T94)
      AMPL=T98/DSQRT(T102)*.707D0
      AMP=AMPL/100.D0
      IF(ZETA.GE.1.0D0)GOTO15
C     ******************************************************************
C     *  SET  II.  DEFINED ONLY FOR ZETA LESS THAN ONE.                *
C     ******************************************************************
      T65=DSQRT(T64)
      T66=T94*T65
      T67=T65/ZETA
      PHI=DATAN(T67)
      T68=T66-PHI
      T69=DSIN(T68)
      T70=DSIN(T66)
      T71=ZETA*T70
      T72=T70/OMEGAN
      T73=T66+PHI
      T74=DSIN(T73)
      T76=OMEGAN*T70
      T119=OMEGAN*T65
      T203=T62/T65
      T204=-T69+2.D0*T71
      T205=T203*T204
      T206=T203*T72
      T207=1.D0-T203*T74
      T208=-T203*T76
      T209=-T203*T69
      PHIR11=T62/T65*(-T69+2.D0*T71)
      PHIR12=T62/T119*T70
      PHIR13=1.D0-1.D0/T65*T62*T74
      PHIR21=-T98*PHIR12
      PHIR22=-1.D0/T65*T62*T69
      PHIR23=-PHIR21
      PHIR24=PHIR13
      PHIRT1=-2.D0*ZETA/OMEGAN*(1.D0-PHIR22)+(4.D0*T63-1.D0)*PHIR12
      PHIR14=DELT+PHIRT1
      GOTO17
   15 IF(ZETA.GT.1.0D0)GOTO16
C     ******************************************************************
C     *  SET  III.  DEFINED FOR ZETA EQUAL TO ONE.                     *
C     ******************************************************************
      PHIR11=T62*T95
      PHIR12=T96
      PHIR13=1.D0-PHIR11
      PHIR21=-T98*PHIR12
      PHIR22=T62*(1.D0-T94)
      PHIR14=DELT-2.D0/OMEGAN*(1.D0-PHIR22)+3.D0*PHIR12
      PHIR23=-PHIR21
      PHIR24=PHIR13
      GOTO17
C     ******************************************************************
C     *  SET  IV.  DEFINED ONLY FOR ZETA GREATER THAN ONE.             *
C     ******************************************************************
   16 T82=DSQRT(-T64)
      T88=OMEGAN*T82
      T89=2.D0*T88
      T90=1.D0/T89
      T93=OMEGAN/(2.D0*T82)
      T77=ZETA/T82
      T78=-1.D0/(ZETA*T82-T64)
      T79=0.5D0*T78
      T80=1.D0+T77
      T81=0.5D0*T80
      T91=0.5D0*(T63+ZETA*T82-1.D0)/T64
      T92=0.5D0/(T63-1.D0+ZETA*T82)
      T108=T104/T82
      T83=ZETA-T82
      T84=ZETA+T82
      T162=-OMEGAN*T83
      T163=-OMEGAN*T84
      T85=DEXP(T162*DELT)
      T86=DEXP(T163*DELT)
      T87=T85-T86
      T164=-0.5D0*T101*T78
      T165=-0.5D0*T101*T80
      T166=T108/2.D0
      T210=T81*T85+T79*T86
      T211=T90*(T85-T86)
      T212=T91*T85+T92*T86+1.D0
      T213=T79*T85+T81*T86
      T214=-T93*T87
      PHIR11=T79*T85+T81*T86+T77*T87
      PHIR12=T90*T87
      PHIR13=1.D0+T91*T85+T92*T86
      PHIR21=-T98*PHIR12
      PHIR22=T79*T85+T81*T86
      PHIR23=-PHIR21
      PHIR24=PHIR13
      PHIR14=DELT+(2.D0*ZETA/OMEGAN)*(PHIR22-1.D0)+(2.D0*ZETA-1.D0)*(2.D
     *0*ZETA+1.D0)*PHIR12
C     ******************************************************************
C     *  BEGIN ITERATION.                                              *
C     ******************************************************************
   17 CONTINUE
    2 TM1=TIME
      TNITER=TNITER+1.D0
      TIME=ITIME*DELT
      T14=OMEGA*TIME
      RSTOR=R
      R=DCOS(T14)
      T109=T60*TIME
      T110=DEXP(-T109)
      T111=OMEGA*TIME
      T112=DCOS(T111)
      T113=DSIN(T111)
      T118=OMEGAN*TIME
      IF(ZETA.GE.1.0D0)GOTO80
      T136=T119*TIME
      T137=T136-PHI
      T116=DSIN(T137)
      T117=DSIN(T136)
      GOTO81
   80 IF(ZETA.EQ.1.0D0)GOTO81
      T160=DEXP(T162*TIME)
      T161=DEXP(T163*TIME)
   81 ITIME=ITIME+1
      IF(ZETA-1.D0)27,28,29
C     ******************************************************************
C     *  BEGIN CALCULATION OF EXACT SOLUTION.  THREE CASES.            *
C     ******************************************************************
C     *  CASE I.  ZETA EQUAL TO OR GREATER THAN ZERO AND LESS THAN ONE.
   27 EXACT=T101/T65*T116*T110+T104/T65*T110*T117+T101*T112+T103*T113
      GOTO30
C     *  CASE II.  ZETA EQUAL TO ONE.
   28 EXACT=-T101*T110*(1.D0-T118)+T104*T118*T110+T101*T112+T103*T113
      GOTO30
C     *  CASE III.  ZETA GREATER THAN ONE.
   29 EXACT=T164*T160+T165*T161+T166*(T160-T161)+T101*T112+T103*T113
   30 IF(ZETA-1.D0)31,32,33
C     ******************************************************************
C     *  BEGIN CALCULATION OF STEP TRANSITION MATRIX SOLUTION.         *
C     ******************************************************************
C     *  CASE I.  ZETA EQUAL TO OR GREATER THAN ZERO AND LESS THAN ONE.
   31 TEMP=STEP1
      STEP1=T205*STEP1+T206*STEP2+T207*RSTOR
      STEP2=T208*TEMP+T209*STEP2+T208*RSTOR
      GOTO90
C     *  CASE II.  ZETA EQUAL TO ONE.
   32 TEMP=STEP1
      STEP1=T200*STEP1+T96*STEP2+T201*RSTOR
      STEP2=T97*RSTOR-T97*TEMP+T202*STEP2
      GOTO90
C     *  CASE III.  ZETA GREATER THAN ONE.
   33 TEMP=STEP1
      STEP1=T210*STEP1+T211*STEP2+T212*RSTOR
      STEP2=T214*TEMP-T214*RSTOR+T213*STEP2
C     *  CALCULATE ERROR VALUES FOR THE STEP METHOD.
   90 ERSTEP=DABS(EXACT-STEP1)
      PESTPN=ERSTEP/AMP
      PESSTP=PESSTP+PESTPN
      APESTP=PESSTP/TNITER
C     ******************************************************************
C     *  BEGIN CALCULATION OF RAMP TRANSITION MATRIX SOLUTION.         *
C     ******************************************************************
      SLOPE=(R-RSTOR)/DELT
      Z1TEMP=RAMP1
      RAMP1=PHIR11*RAMP1+PHIR12*RAMP2+PHIR13*RSTOR+PHIR14*SLOPE
      RAMP2=PHIR21*Z1TEMP+PHIR22*RAMP2+PHIR23*RSTOR+PHIR24*SLOPE
C     *  CALCULATE ERROR VALUES FOR THE RAMP METHOD.
      ERRAMP=DABS(EXACT-RAMP1)
      PERMPN=ERRAMP/AMP
      PESRMP=PESRMP+PERMPN
      APERMP=PESRMP/TNITER
C     ******************************************************************
C     *  BEGIN CALCULATION OF RUNGE-KUTTA FOURTH ORDER SOLUTION.       *
C     ******************************************************************
      TEMPT1=TM1+T9
      R1=DCOS(OMEGA*TEMPT1)
      TEMP=T98*R1
      DRK41=RK42
      DRK42=T120*RK42-T98*RK41+T98*RSTOR
      DEL11=DELT*DRK41
      DEL21=DELT*DRK42
      TEMP1=RK41+DEL11/2.D0
      TEMP2=RK42+DEL21/2.D0
      DEL12=DELT*TEMP2
      DEL22=DELT*(T120*TEMP2-T98*TEMP1+TEMP)
      TEMP1=RK41+DEL12/2.D0
      TEMP2=RK42+DEL22/2.D0
      DEL13=DELT*TEMP2
      DEL23=DELT*(T120*TEMP2-T98*TEMP1+TEMP)
      TEMP1=RK41+DEL13
      TEMP2=RK42+DEL23
      DEL14=DELT*TEMP2
      DEL24=DELT*(T12J*TEMP2-T98*TEMP1+T98*R)
      RK41=RK41+(DEL11+2.D0*(DEL12+DEL13)+DEL14)/6.D0
      RK42=RK42+(DEL21+2.D0*(DEL22+DEL23)+DEL24)/6.D0
C  *  CALCULATE ERROR VALUES FOR THE RK-4 METHOD.
      ERRK4=DABS(EXACT-RK41)
      ESRK4=ESRK4+ERRK4
      PERK4N=ERRK4/AMP
      PESRK4=PESRK4+PERK4N
      APERK4=PESRK4/TNITER
      IF(TIME.GE.TFINAL)GOTO55
      GO TO 2
   55 CONTINUE
C   *****************************************************************
C   *  END ITERATION.                                               *
C   *****************************************************************
```

APPENDIX E

ERROR MEASUREMENT CONSIDERATIONS

Consider Figure 30. The best indication of the accuracy of a numerical simulation method is how closely the approximate solution follows the exact solution over some time interval. For this reason a time-averaged error is needed. Furthermore, since the error can be positive or negative, absolute error should be used. So define

$$\text{Absolute Error} = \left| \begin{matrix}\text{exact}\\ \text{solution}\end{matrix} - \begin{matrix}\text{approximate}\\ \text{solution}\end{matrix} \right|$$

$$\text{Error Sum} = \sum_{i=1}^{N} (\text{absolute error})_i$$

$$\text{Average Absolute Error} = \frac{\text{error sum}}{N}$$

where i indicates a particular point in time, and N is the total number of solution steps in the interval.

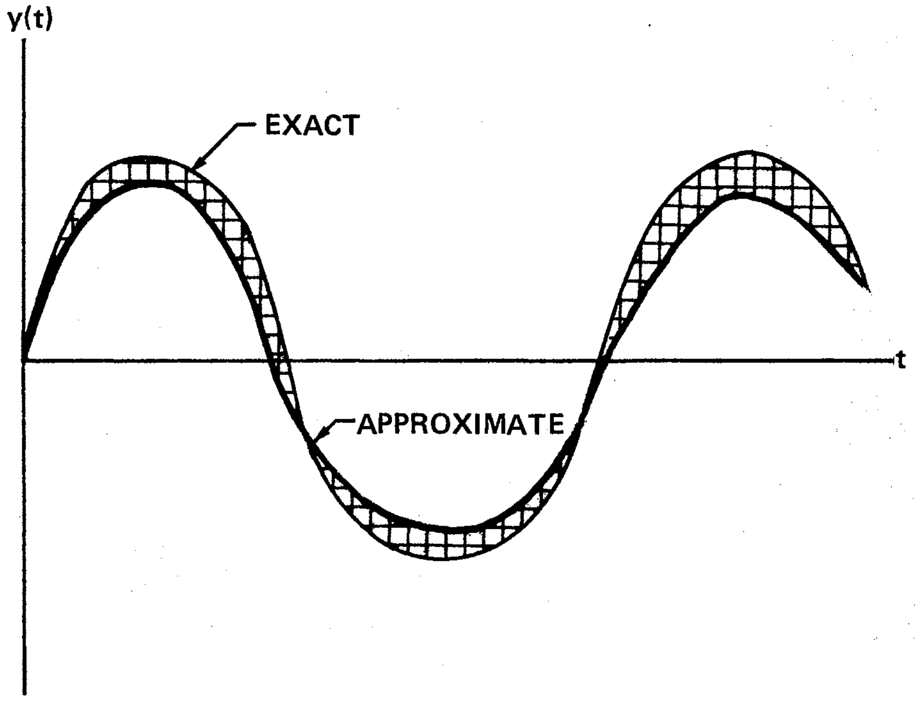

Figure 30. Error Measurement

The average absolute error measures the error throughout the interval, but it does not give relative-error information. If the exact solution is small in magnitude, the absolute error can be small while the percentage error is large. A true per-cent error, on the other hand, gives relative-error information without giving any indication about absolute-error magnitudes. Furthermore, if the solution curve is oscillatory, as in the cases studied here, there is a large peak value of per-cent error whenever the solution passes through zero, which is not at all indicative of how closely the approximation follows the solution. So a normalized per-cent error can be defined by dividing the absolute error by the RMS of the steady-state response amplitude. Since an input amplitude of 1 is used throughout the investigations, the steady-state response amplitude is the same as the amplitude ratio, which is given by

$$\text{Amplitude} = \frac{1}{\sqrt{1 + (\omega T)^2}}$$

for the first-order system and

$$\text{Amplitude} = \frac{\omega_\eta^2}{\sqrt{(-\omega^2 + \omega_\eta^2)^2 + 4\zeta^2\omega^2\omega_\eta^2}}$$

for the second-order system. The RMS average is obtained as

$$\text{AMPL} = \frac{\sqrt{2}}{2}\,(\text{amplitude}) \qquad ,$$

normalized per-cent error is given by

$$\text{NPE} = \left(\frac{\text{absolute error}}{\text{AMPL}}\right) \times 100)\% \qquad ,$$

and finally, averaged normalized per-cent error is

.

$$\text{ANPE} = \frac{\sum_{i=1}^{N} (\text{NPE})_i}{N} \quad .$$

This error measure gives relative-error information since it is a percentage based on a nominal response amplitude. It also gives absolute-error information since it is normalized by a constant, which is known for the particular system being considered. The normalized per-cent error can be multiplied by this known constant to give the absolute error.

APPENDIX F

COMPUTATION TIME ESTIMATION

The time required to do a calculation on a computer is highly variable. It depends on the particular computer being used, the word length, and in some cases, the number of programs being run. The most reliable way to estimate the computation time for a numerical method is to count each basic arithmetic operation done in the calculation. If the time required for the particular computer being used to do each operation is known, an estimate for the speed of the method can be obtained.

The basic operations done in both the transition matrix method and the Runge-Kutta fourth-order method were counted and are tabulated below. The times in Table I were calculated using the following estimates for the IBM 360 model 65:

1 Addition = 0.65 Microseconds

1 Multiplication = 4.9 Microseconds .

The times for the RK-4 method include an estimate of 1 multiplication for an additional input evaluation.

TABLE I

COMPUTATION TIME

		Additions	Multiplications	Total Time per Iteration (Microseconds)	% of RK-4
First Order System	Step	1	2	10.45	32.0
	Ramp	3	4	21.55	66.0
	RK-4	5	5	32.65	100.0
Second Order System	Step	4	6	29.40	21.7
	Ramp	7	9	48.65	33.0
	RK-4	23	26	147.25	100.0

VITA

William Donald Smith

Candidate for the Degree of

Master of Science

Thesis: A FINITE-STEP TRANSITION MATRIX APPROACH FOR NUMERICAL SIMULATION OF "STIFF" DYNAMIC SYSTEMS

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Fort Worth, Texas, September 11, 1949, the son of Mr. and Mrs. Robert O. Smith.

Education: Graduated from Granbury High School, Granbury, Texas in May, 1967; received the Bachelor of Science degree, with a major in Mechanical Engineering, from the University of Texas at Arlington, Arlington, Texas, in December, 1971; completed the requirements for the Degree of Master of Science in May, 1974.

Professional Experience: Data Analyst, Bell Helicopter Company, Summer, 1969; Engineer, Texas Instruments Incorporated, 1972; Research Assistant on Research Projects at Oklahoma State University, 1972 to date.

Professional Organizations: Member of American Society of Mechanical Engineers, American Institute of Aeronautics and Astronautics, Pi Tau Sigma, Tau Beta Pi, Sigma Gamma Tau, Alpha Chi, and Phi Kappa Phi.