ELECTRICAL ENGINEERING

SEVERAL METHODS FOR DIGITAL SIMULATION OF PHYSICAL SYSTEMS

PREPARED BY
ADVANCED STUDIES GROUP
AUBURN UNIVERSITY
J. L. LOWRY, PROJECT LEADER

TECHNICAL REPORT
FEBRUARY 1968

CONTRACT—NAS 8-20004
GEORGE C. MARSHALL SPACE FLIGHT CENTER
NATIONAL AERONAUTICS & SPACE ADMINISTRATION
HUNTSVILLE, ALABAMA 35812

ENGINEERING EXPERIMENT STATION
AUBURN UNIVERSITY
AUBURN, ALABAMA

SEVERAL METHODS FOR DIGITAL SIMULATION OF PHYSICAL SYSTEMS

Prepared by

ADVANCED STUDIES GROUP

AUBURN UNIVERSITY

J. L. Lowry, Project Leader

TECHNICAL REPORT

February 1968

Contract - NAS8-20004

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Huntsville, Alabama 35812

APPROVED BY:

C. H. Holmes, Head Professor
Electrical Engineering
Department

SUBMITTED BY:

J. L. Lowry, Professor
Electrical Engineering
Department

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FOREWORD

This is a technical report prepared by the Advanced Studies Group, Electrical Engineering Department, Auburn University, toward fulfillment of Contract NAS8-20004 granted to Auburn Research Foundation (name changed to Engineering Experiment Station), Auburn Alabama. This contract was originally awarded January 19, 1965, and was extended to January 18, 1966. It was further extended to April 18, 1966, then to December 18, 1966, and thereafter to February 18, 1968.

SUMMARY

A number of methods which are suitable for digitally simulating physical systems are presented in detail. Several of the methods are well-known, while two are presented for the first time. A brief discussion of the error involved in each method is presented along with the method derivation. Each class of methods is illustrated with one or more solved examples, and the computer results are given in tabular and graphic forms. The approximations, restrictions, and limitations for each class of methods are presented along with that class.

PERSONNEL

The following staff members of Auburn University have actively contributed in this study:

J. L. Lowry	-	Professor of Electrical Engineering
D. W. Kelly	-	Instructor of Electrical Engineering
T. L. Richards	-	Graduate Assistant in Electrical Engineering
L. A. Ward	-	Teaching Assistant in Electrical Engineering

INTRODUCTION

In the course of analyzing a physical system, the engineer is often confronted with the problem of determining the system's behavior in response to a class of specified inputs. In many such cases a digital computer can be used if the system can be translated into a form that the computer can utilize. The object of this treatise will be to present several methods for arranging the system equations into iterative forms that can be computer adapted.

Chapter I is a discussion of the most popular classical numerical integration technique, Runge-Kutta numerical integration. Of the infinite variety of orders and modifications of Runge-Kutta, only a few of the more useful ones are given.

Chapter II presents a class of methods that uses a series of known solution points to predict future points of the solution of the set of differential equations describing the system. The formulas in this class are iterative ones in that they can be repeatedly applied to a block of points until the solution at each point is obtained.

In Chapter III an attempt is made to solve the same problem without attempting to directly solve the system of simultaneous differential equations. Instead, approximations are made concerning the Laplace transform transfer function relating the input and output of the system. The resulting expression, a function of the time delay operator z^{-1}, is a discrete filter that weights past inputs and outputs and the present input to form the present output.

In Chapter IV the system of equations is solved directly using state variable theory. The convolution integral that results in the exact solution can be simplified by approximating the input curve with a first-order curve connecting input points. The result is a repetitive method that can be used to simulate the system.

The method discussed in Chapter V approaches the problem from the viewpoint that if the system's response to a certain input is known then the system's response to a combination of this type input can also be found. Two different input approximations are used and summations are developed that describe the desired output in terms of the response due to the known input and a new input. Conditions are developed under which the summations can be truncated to yield considerable simplicity.

The class of methods appearing in each of Chapters I through V is applied to a third-order linear system to demonstrate application. The results appear at the end of each chapter.

I. RUNGE-KUTTA INTEGRATION FORMULAS[1,2,3,4]

The problem of simulating a physical system described by a differential equation, or set of differential equations, can usually be reduced to one of solving a set of first-order differential equations. If the set contains only one or two equations and if the forcing functions are analytic functions, then the system can be solved analytically with little trouble. However, the solution may be desired for both analytic forcing functions and forcing functions for which only discrete values are known. This leads to the need for a numerical iteration technique that will be suitable for solving the first-order differential equation

$$\dot{x} = f(x,t) \ , \tag{1}$$

where $\dot{x}$ denotes the time derivative of x.

Such a technique, commonly referred to as the Runge-Kutta integration technique, can be derived by approximating the Taylor series expansion of the exact solution of x with a truncated series. The truncated series solution will be an approximation of the exact solution.

If x is expanded into a Taylor series about the point t_o , the resulting expansion, with $x(t_o)$ denoted by x_o , is

$$(x_o+\Delta x_o) = x_o + \dot{x}_o(t_o+h-t_o) + \frac{\ddot{x}_o}{2!}(t_o+h-t_o)^2 \tag{2}$$
$$+ \frac{\dddot{x}_o}{3!}(t_o+h-t_o)^3 + \frac{x_o^{(4)}}{4!}(t_o+h-t_o)^4 + \ldots + \frac{x_o^{(K)}}{K!}(t_o+h-t_o)^K + \ldots \ .$$

In this form, $(x_o + \Delta x_o)$ represents the solution at (t_o+h). Written in compact form, (2) becomes

$$(x_o + \Delta x_o) = x_o + \sum_{K=1}^{\infty} \left(\frac{h^K}{K!}\right) x_o^{(K)} \quad . \tag{3}$$

Of course, if the function f could be differentiated a great number of times and evaluated at x_o, then a sufficiently accurate solution could be obtained using (3). However, the use of a small number of terms makes the expansion seem much less formidable. Using a subscript to denote partial differentiation (i.e., $f_x = \frac{\partial f}{\partial x}$) the first few total derivatives of x can be written as

$$\begin{aligned}
\dot{x} &= f(x,t) \\
\ddot{x} &= \frac{d(\dot{x})}{dt} = \frac{d[f(x,t)]}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\frac{dx}{dt} = f_t + f_x f \\
\dddot{x} &= f_{tt} + 2f_{xt}f + f_{xx}f^2 + f_x f_t + f f_x^2 \\
x^{(4)} &= (f_{ttt} + 2f_t f_{xt} + 2f f_{xtt} + f_{xxt}f^2 + 2f_{xx}f f_t + f_{xt}f_t \\
&\quad + f_x f_{tt} + 2f f_{xt} f_x + f_t f_x^2) + (f_{ttx}f + 2f_{xxt}f^2 + 2f_{xt}f_x f \\
&\quad + f_{xxx}f^3 + 2f_{xx}f^2 f_x + f_{xx}f_t f + f_x f_{xt} f + f_x^3 f + 2f^2 f_{xx} f_x) \ .
\end{aligned} \tag{4}$$

These will be used in subsequent operations as substitutions in the series expansion of $(x_o + \Delta x_o)$ in (3).

Instead of computing the derivatives to obtain the solution of the differential equation, it is desirable to express the expansion in terms of the value of f at a number of points in the vicinity of (x_o, t_o), expressed in equation form as

$$(x_o + \Delta x_o) = x_o + \sum_{i=1}^{m} \omega_i k_i \ , \text{ where} \tag{5}$$

$$k_i = hf(t_o + \alpha_i h,\ x_o + \sum_{j=1}^{i-1} \beta_{ij} k_j) \ . \qquad (6)$$

The constants α_i, β_{ij}, and ω_i are to be determined so that (5) agrees with the actual expansion given by (3) through the first m terms. A Runge-Kutta method of order m is then the result given in (5) and (6). The process for developing the method with m=4 (i.e., fourth-order Runge-Kutta) is outlined in the following section.

To reduce (6) to a usable form, it is expanded into a Taylor series in two variables having the general form

$$f(x_o + A, y_o + B) = f + Af_x + Bf_y + \frac{A^2}{2!} f_{xx} + ABf_{xy} + \frac{B^2}{2!} f_{yy}$$
$$+ \frac{A^3}{3!} f_{xxx} + \frac{A^2B}{2!} f_{xxy} + \frac{AB^2}{2!} f_{xyy} + \frac{B^3}{3!} f_{yyy} + \ldots \ .$$

The expansion of (6) for i=1,2,3, and 4 are

$$k_1 = hf + h^2\alpha_1 f_t + \frac{h^3}{2}\alpha_1^2 f_{tt} + \frac{h^4}{6}\alpha_1^3 f_{ttt} + \ldots \qquad (7)$$

$$k_2 = hf + (\alpha_2 h^2 f_t + \beta_{21} h k_1 f_x) + \left(\frac{\alpha_2^2}{2} h^3 f_{tt} + \alpha_2 h^2 f_{xt}\beta_{21} k_1 \right. \qquad (8)$$
$$\left. + \frac{h}{2} \beta_{21}^2 k_1^2 f_{xx}\right) + \left(\frac{\alpha_2^3}{6} h^4 f_{ttt} + \frac{\alpha_2^2}{2} h^3 \beta_{21} k_1 f_x f_{tt} \right.$$
$$\left. + \frac{\alpha_2}{2} h^2 \beta_{21}^2 k_1^2 f_t f_{xx} + h \frac{\beta_{21}^3}{6} k_1^3 f_{xxx}\right) + \ldots$$

$$k_3 = hf + (\alpha_3 h^2 f_t + h\beta_{31} k_1 f_x + h\beta_{32} k_2 f_x) + \left(\frac{\alpha_3^2 h^3}{2} f_{tt} \right. \qquad (9)$$
$$+ \alpha_3 h^2 \beta_{31} k_1 f_{xt} + \alpha_3 h^2 \beta_{32} k_2 f_{xt} + \frac{h}{2} \beta_{31}^2 k_1^2 f_{xx} + \frac{h}{2} \beta_{32}^2 k_2^2 f_{xx}$$
$$\left. + h\beta_{31}\beta_{32} k_1 k_2 f_{xx}\right) + \left(\frac{\alpha_3^3 h^4}{6} f_{ttt} + \frac{h^3 \alpha_3^2}{2} (\beta_{31} k_1 + \beta_{32} k_2) f_{ttx} \right.$$
$$\left. + \frac{\alpha_3 h^2}{2} (\beta_{31}^2 k_1^2 + 2\beta_{31}\beta_{32} k_1 k_2 + \beta_{32}^2 k_2^2) f_{txx} \right.$$

$$+ \frac{h}{6}(\beta_{31}^3 k_1^3 + 3\beta_{31}^2 k_1^2 \beta_{32} k_2 + 3\beta_{31} k_1 \beta_{32}^2 k_2^2 + \beta_{32}^3 k_2^3) f_{xxx}\Big) + \ldots$$

$$k_4 = hf + (\alpha_4 h^2 f_t + h(\beta_{41} k_1 + \beta_{42} k_2 + \beta_{43} k_3) f_x) + \Big(\frac{\alpha_4^2}{2} h^3 f_{tt} \qquad (10)$$

$$+ \alpha_4 h^2 (\beta_{41} k_1 + \beta_{42} k_2 + \beta_{43} k_3) f_{xt} + \frac{h}{2}(\beta_{41} k_1 + \beta_{42} k_2 + \beta_{43} k_3)^2 f_{xx}\Big)$$

$$+ \Big(\frac{\alpha_4^3 h^4}{6} f_{ttt} + \frac{h^3}{2}\alpha_4^2 (\beta_{41} k_1 + \beta_{42} k_2 + \beta_{43} k_3) f_{ttx}$$

$$+ \frac{\alpha_4 h^2}{2} (\beta_{41} k_1 + \beta_{42} k_2 + \beta_{43} k_3)^2 f_{txx} + \frac{h}{6}(\beta_{41} k_1 + \beta_{42} k_2 + \beta_{43} k_3)^3 f_{xxx}\Big) + \ldots .$$

Since the equations for k_2, k_3, and k_4 contain expressions involving previous values of k, they must be solved to yield k_i in terms of known quantities. This procedure will be demonstrated for k_2, but not for k_3 and k_4 due to the astronomical number of terms involved. Substituting the equation for k_1 into the one for k_2 and regrouping terms in ascending powers of h gives

$$k_2 = hf + h^2(\alpha_2 f_t + \beta_{21} f f_x) + h^3\Big(\alpha_1 \beta_{21} f_t f_x + \frac{\alpha_2^2}{2} f_{tt}$$

$$+ \alpha_2 \beta_{21} f_{xt} f + \frac{\beta_{21}^2}{2} f^2 f_{xx}\Big) + h^4\Big(\alpha_2 f_{xt} \beta_{21} \alpha_1 f_t + \alpha_1 f f_t \beta_{21}^2 f_{xx}$$

$$+ \frac{\alpha_2^3}{6} f_{ttt} + \frac{\alpha_2^2}{2} \beta_{21} f_x f_{tt} f + \frac{\alpha_2}{2} \beta_{21}^2 f_t f_{xx} f^2 + \frac{\beta_{21}^3}{6} f^3 f_{xxx}\Big) + \ldots .$$

The expressions for k_1 and k_2, with corresponding ones for k_3 and k_4, can now be substituted into (5) to yield an approximation of $(x_o + \Delta x_o)$ for m=4. To be an approximation of the solution of the differential equation this result must also coincide as closely as possible with the actual Taylor series expansion of the solution given by (3) and (4). In order to solve for the α_i's, ω_i's, and β_i's the coefficients of like power of h of the two series are equated so that the series are forced to

coincide through m terms. From (6) a solution for α_1 is $\alpha_1=o$. Equating coefficients of h, h^2, h^3, and h^4 yields

$$h: \quad \omega_1 f + \omega_2 f + \omega_3 f + \omega_4 f = f \tag{11}$$

$$h^2: \quad \omega_2(\alpha_2 f_t + \beta_{21} f f_x) + \omega_3(\alpha_3 f_t + \beta_{31} f f_x + \beta_{32} f f_x) \tag{12}$$
$$+ \omega_4(\alpha_4 f_t + \beta_{41} f f_x + \beta_{42} f f_x + \beta_{43} f f_x) = \frac{f_t}{2} + \frac{f f_x}{2}$$

$$h^3: \quad \omega_2\left(\frac{\alpha_2^2}{2} f_{tt} + \alpha_2 \beta_{21} f_{xt} f + \frac{\beta_{21}^2}{2} f^2 f_{xx}\right) + \omega_3\left(\frac{\alpha_3^2}{2} f_{tt}\right. \tag{13}$$
$$+ \beta_{32} f_x \alpha_2 f_t + \beta_{32} f_x^2 \beta_{21} f + \alpha_3 f f_{xt}(\beta_{31} + \beta_{32}) + \frac{f_{xx} f^2}{2}(\beta_{31}+\beta_{32})^2\Big)$$
$$+ \omega_4\left(\frac{f_{xx}}{2} f^2(\beta_{41}+\beta_{42}+\beta_{43})^2 + \alpha_4 f f_{xt}(\beta_{41}+\beta_{42}+\beta_{43}) + \frac{\alpha_4^2}{2} f_{tt}\right.$$
$$\left. + \beta_{43} f_x(\alpha_3 f_t + \beta_{31} f f_x + \beta_{32} f f_x) + \beta_{42} f_x(\alpha_2 f_t + \beta_{21} f f_x)\right)$$
$$= \frac{1}{6}(f_{tt} + 2 f_{xt} f + f_{xx} f^2 + f_x f_t + f f_x^2)$$

$$h^4: \quad \omega_2\left[\frac{\alpha_2^3}{6} f_{ttt} + \frac{\alpha_2^2}{2} \beta_{21} f_x f f_{tt} + \frac{\alpha_2}{2} \beta_{21}^2 f_t f^2 f_{xx} + \frac{\beta_{21}^3}{6} f^3 f_{xxx}\right] \tag{14}$$

$$+ \omega_3\left[\frac{1}{6} f_{xxx} f^3 (\beta_{31}+\beta_{32})^3 + \frac{1}{2} \alpha_3 f_{txx} f^2 (\beta_{31}+\beta_{32})^2 + \frac{1}{2}\alpha_3^2 f_{ttx} f(\beta_{31}+\beta_{32})\right.$$
$$\div \frac{\alpha_3^3}{6} f_{ttt} + \beta_{32}(\alpha_2 f_t + \beta_{21} f f_x)(\alpha_3 f_{xt} + \beta_{32} f_{xx} f + \beta_{31} f_{xx} f)$$
$$\left. + \beta_{32} f_x\left(\frac{\alpha_2^2}{2} f_{tt} + \alpha_2 \beta_{21} f f_{xt} + \frac{\beta_{21}^2}{2} f^2 f_{xx}\right)\right]$$
$$+ \omega_4\left[\frac{f_{xxx}}{6} f^3 (\beta_{41}+\beta_{42}+\beta_{43})^3 + \frac{\alpha_4 f_{txx}}{2} f^2 (\beta_{41}+\beta_{42}+\beta_{43})^2\right.$$
$$+ \frac{\alpha_4^2}{2} f_{ttx} f(\beta_{41}+\beta_{42}+\beta_{43}) + \frac{\alpha_4^3}{6} f_{ttt} + f_{xx} f(\beta_{42}^2 + \beta_{41}\beta_{42} + \beta_{42}\beta_{43})$$
$$\cdot (\alpha_2 f_t + \beta_{21} f f_x) + f_{xx} f(\beta_{43}^2 + \beta_{41}\beta_{43} + \beta_{42}\beta_{43}) \cdot (\alpha_3 f_t + \beta_{31} f_x f + \beta_{32} f_x f)$$

$$+ \alpha_4 f_{xt}\beta_{42}(\alpha_2 f_t + \beta_{21} f f_x) + \alpha_4 f_{xt}\beta_{43}(\alpha_3 f_t + \beta_{31} f_x f + \beta_{32} f_x f)$$

$$+ \beta_{43} f_x\left(\beta_{32} f_x \alpha_2 f_t + \beta_{32} f_x^2 \beta_{21} f + \frac{\alpha_3^2}{2} f_{tt} + \alpha_3 f_{xt} f(\beta_{31}+\beta_{32})\right.$$

$$\left. + \frac{1}{2} f_{xx} f^2 (\beta_{31}+\beta_{32})^2\right) + \beta_{42} f_x\left(\frac{\alpha_2^2}{2} f_{tt} + \alpha_2\beta_{21} f_{xt} f + \frac{\beta_{21}^2}{2} f^2 f_{xx}\right)\Big]$$

$$= \frac{1}{24}\Big[f_{ttt} + 3 f_t f_{xt} + 3 f f_{xtt} + 3 f_{xxt} f^2 + 3 f_{xx} f_t f + f_x f_{tt}$$

$$+ 5 f f_x f_{xt} + f_t f_x^2 + f^3 f_{xxx} + 4 f^2 f_x f_{xx} + f f_x^3\Big] .$$

For these expressions to be valid for any arbitrary function f, the coefficients of f and its partials must be zero. This will be true for equations (11) through (14) if

$$\alpha_2 = \beta_{21} \tag{15}$$

$$\alpha_3 = \beta_{31} + \beta_{32} \tag{16}$$

$$\alpha_4 = \beta_{41} + \beta_{42} + \beta_{43} \tag{17}$$

$$\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1 \tag{18}$$

$$\alpha_2\omega_2 + \alpha_3\omega_3 + \alpha_4\omega_4 = \frac{1}{2} \tag{19}$$

$$\alpha_2^2\omega_2 + \alpha_3^2\omega_3 + \alpha_4^2\omega_4 = \frac{1}{3} \tag{20}$$

$$\alpha_2^3\omega_2 + \alpha_3^3\omega_3 + \alpha_4^3\omega_4 = \frac{1}{4} \tag{21}$$

$$\alpha_2\omega_3\beta_{32} + \omega_4(\alpha_2\beta_{42} + \alpha_3\beta_{43}) = \frac{1}{6} \tag{22}$$

$$\alpha_2^2\omega_3\beta_{32} + \omega_4(\alpha_2^2\beta_{42} + \alpha_3^2\beta_{43}) = \frac{1}{12} \tag{23}$$

$$\omega_3\alpha_2\alpha_3\beta_{32} + \omega_4(\alpha_2\beta_{42} + \alpha_3\beta_{43})\alpha_4 = \frac{1}{8} \tag{24}$$

$$\omega_4\alpha_2\beta_{32}\beta_{43} = \frac{1}{24} \quad . \tag{25}$$

The eleven equations given in (15) through (25) contain thirteen unknowns, so that two degrees of freedom exist in their solution. These

same equations are valid for $m \leq 4$ if the appropriate terms are set equal to zero, thus they can be used to obtain the lower-order Runge-Kutta methods, as will be demonstrated.

Second Order

With m equal to two and all terms with subscripts greater than two being zero, the equations in (15) through (25) become

$$\omega_1 + \omega_2 = 1$$
$$\omega_2\alpha_2 = \frac{1}{2} \text{ and}$$
$$\alpha_2 = \beta_{21}.$$

Runge selected $\alpha_2 = \frac{1}{2}$, giving $\omega_2 = 1$, $\omega_1 = 0$, and $\beta_{21} = \frac{1}{2}$. The resulting iteration is

$$\begin{aligned} k_1 &= hf(t_o, x_o) \\ k_2 &= hf(t_o + \tfrac{1}{2}h, x_o + \tfrac{1}{2}k_1) \\ (x_o + \Delta x_o) &= x_o + k_2 \quad . \end{aligned} \tag{26}$$

Heun selected $\alpha_2 = 1$, giving $\omega_2 = \frac{1}{2}$, $\omega_1 = \frac{1}{2}$, and $\beta_{21} = 1$. The resulting iteration is

$$\begin{aligned} k_1 &= hf(t_o, x_o) \\ k_2 &= hf(t_o + h, x_o + k_1) \\ (x_o + \Delta x_o) &= x_o + \tfrac{1}{2}(k_1 + k_2) \quad . \end{aligned} \tag{27}$$

Equation (27) is the familiar Trapezoidal Rule. A wide variety of second-order methods can thus be obtained, all of which agree with the actual Taylor series expansion for $(x_o + \Delta x_o)$ through the factor of $h^2f^{(1)}$. The error of a second-order Runge-Kutta is of order $h^3f^{(2)}$.

Third Order

For the third-order methods, (15) through (25) become

$$\omega_1 + \omega_2 + \omega_3 = 1$$
$$\alpha_2\omega_2 + \alpha_3\omega_3 = \frac{1}{2}$$
$$\alpha_2^2\omega_2 + \alpha_3^2\omega_3 = \frac{1}{3}$$
$$\alpha_2\beta_{32}\omega_3 = \frac{1}{6}$$
$$\alpha_2 = \beta_{21}$$
$$\alpha_3 = \beta_{31} + \beta_{32} \quad .$$

A widely used method results if $\alpha_2 = \frac{1}{2}$ and $\alpha_3 = 1$, such that $\beta_{21} = \frac{1}{2}$, $\omega_2 = \frac{2}{3}$, $\omega_3 = \frac{1}{6}$, $\omega_1 = \frac{1}{6}$, $\beta_{32} = 2$, and $\beta_{31} = -1$. Then

$$\begin{aligned} k_1 &= hf(t_o, x_o) \\ k_2 &= hf(t_o + \tfrac{1}{2}h, x_o + \tfrac{1}{2}k_1) \\ k_3 &= hf(t_o + h, x_o - k_1 + 2k_2) \\ (x_o + \Delta x_o) &= x_o + \tfrac{1}{6}(k_1 + 4k_2 + k_3) \quad . \end{aligned} \tag{28}$$

Fourth Order

Perhaps the most commonly used Runge-Kutta is a fourth-order one. Again referring to (15) through (25), select $\alpha_2 = \frac{1}{2}$ and $\alpha_3 = \frac{1}{2}$, giving $\alpha_4 = 1$, $\beta_{21} = \frac{1}{2}$, $\omega_1 = \frac{1}{6}$, $\omega_2 = \frac{1}{3}$, $\omega_3 = \frac{1}{3}$, $\omega_4 = \frac{1}{6}$, $\beta_{31} = 0$, $\beta_{32} = \frac{1}{2}$, $\beta_{41} = 0$, $\beta_{42} = 0$, and $\beta_{43} = 1$. The classical method is then

$$\begin{aligned} k_1 &= hf(t_o, x_o) \\ k_2 &= hf(t_o + \tfrac{1}{2}h, x_o + \tfrac{1}{2}k_1) \\ k_3 &= hf(t_o + \tfrac{1}{2}h, x_o + \tfrac{1}{2}k_2) \\ k_4 &= hf(t_o + h, x_o + k_3) \text{ , and} \end{aligned} \tag{29}$$

$$(x_o + \Delta x_o) = x_o + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \ .$$

This may be used to approximate the series expansion for $(x_o + \Delta x_o)$ through a term of order $h^4 f^{(3)}$.

As has been previously mentioned, the inherent error in any order Runge-Kutta is due to truncation of the series expansion of the solution. For instance, if a fourth-order method is used, the first term in which the two series are in disagreement is of order $h^5 f^{(4)}$.

As may be observed, the error is intimately associated with the step-length h. For too large h, the series may not converge, while for too small h, round-off error in the calculations may cause the results to be meaningless. Thus some nominal value must be selected to give adequate results. One method of selecting h is to arbitrarily select a trial value and compute several points. Then halve h and repeat the calculations. If the solution curves are not in agreement within the tolerable degree of accuracy, again halve h. Repeat this procedure until the desired accuracy is achieved. If this recourse fails, double the original h and repeat the same procedure. With some a priori knowledge of the system's behavior and a little luck, a suitable h can usually be found in this manner.

Higher order Runge-Kutta methods are available if the series given in (3) and (5) are matched through higher powers of h. However, the complexity of such a procedure increases rapidly, as does the difficulty in using the results to solve a differential equation. If a higher degree of accuracy is required than is obtained with a fourth-order method, decreasing h is preferred to using a higher order method.

Although the preceding formulas were obtained for a single first-order differential equation, a higher order equation can be treated in the same manner by first reducing it to a set of first-order equations, and then applying the formulas to each first-order equation in sequence. In this manner the solution to the entire set of first-order equations is found.

Although the set of Runge-Kutta formulas has been available for many years, they have not been used extensively to solve differential equations. This is due to the large number of function evaluations required per step, which hand calculation cannot efficiently perform but which a digital computer can. Prior to the computer age the Runge-Kutta formulas were used primarily to obtain a few starting values for methods which required several initial points along the solution curve.

The main objection to solving large systems of equations using Runge-Kutta is the many function evaluations required at each step. A fourth-order method, for instance, applied to n simultaneous equations requires 4n evaluations per step, which becomes extremely time consuming for large n.

Example

A second-order Runge-Kutta is used to solve the third-order differential equation given below from t = 0 to t = 5 with update increment h=0.1.

$$\dddot{x} + 5\ddot{x} + 12\dot{x} + 8x = te^{-3t/2}$$

$$e_o = 8x + \dot{x}$$

$$x = \dot{x} = \ddot{x} = 0 \text{ at } t = 0.$$

These equations may be reduced to the following set of first-order equations

$$x_1 = x$$
$$x_2 = \dot{x}_1 = \dot{x}$$
$$x_3 = \dot{x}_2 = \ddot{x} \quad .$$

The set of equations to be solved is then

$$\dot{x}_1 = x_2 = f_1$$
$$\dot{x}_2 = x_3 = f_2$$
$$\dot{x}_3 = te^{-3t/2} - 5x_3 - 12x_2 - 8x_1 = f_3$$
$$e_o = 8x_1 + x_2 \ ,$$

with $x_{10} = x_{20} = x_{30} = 0$ at $t = 0$.

From equation (27), a suitable second-order method is

$$\begin{cases} dx_{11} = (x_{20})(.1) \\ dx_{21} = (x_{30})(.1) \\ dx_{31} = (-5x_{30} - 12x_{20} - 8x_{10} + 0)(.1) \end{cases}$$

$$\begin{cases} dx_{12} = (x_{20} + dx_{21})(.1) \\ dx_{22} = (x_{30} + dx_{31})(.1) \\ dx_{32} = (-5(x_{30} + dx_{31}) - 12(x_{20} + dx_{21}) - 8(x_{10} + dx_{11}) + (.1)e^{-.15})(.1) \end{cases}$$

$$\begin{cases} x_{11} = x_{10} + \frac{1}{2}(dx_{11} + dx_{12}) \\ x_{21} = x_{20} + \frac{1}{2}(dx_{21} + dx_{22}) \\ x_{31} = x_{30} + \frac{1}{2}(dx_{31} + dx_{32}) \quad . \end{cases}$$

The entire procedure is then repeated using x_{11}, x_{21}, and x_{31} as the new initial values to obtain the solution at t = .2, etc. The solution for e_o is simply $e_o(t) = 8x_{1i} + x_{2i}$.

The previous example was programmed on an IBM 7040 digital computer to give an indication of the accuracy involved. The results are listed in Table 1 along with the results of the third-order method (equation (28)) and the fourth-order method (equation (29)) applied to the same set of equations.

To demonstrate the effect of the time increment upon accuracy, results of a second-, third-, and fourth-order method for h = 0.05, 0.1, 0.2, 0.4, 0.8, and 1.6 are shown in Figures 1, 2, and 3. Also included for comparison is the exact solution.

As can be seen from Table 1 the second-order method yields about 3-digit accuracy, the third-order about 4-digit accuracy, and the fourth-order about 5-digit accuracy.

TABLE 1...Third-order System by Runge-Kutta

with h= 0.1 Seconds.

Time	Analytic Solution	Second-Order Runge-Kutta	Third-Order Runge-Kutta	Fourth-Order Runge-Kutta
0.0	.0	.0	.0	.0
0.1	.0001 6449	.0	.0001 5462	.0001 6622
0.2	.0012 6963	.0009 2526	.0012 6846	.0012 7164
0.3	.0040 6374	.0036 0154	.0040 7659	.0040 6492
0.4	.0090 0563	.0085 1081	.0090 3083	.0090 0584
0.5	.0162 5180	.0158 0036	.0162 8396	.0162 5103
0.6	.0256 9028	.0253 3428	.0257 2313	.0256 8872
0.7	.0370 0047	.0367 6412	.0370 2853	.0369 9845
0.8	.0497 2117	.0496 0358	.0497 4051	.0497 1903
0.9	.0633 1614	.0632 9738	.0633 2460	.0633 1416
1.0	.0772 3077	.0772 7913	.0772 2790	.0772 2917
1.1	.0909 3712	.0910 1593	.0909 2378	.0909 3599
1.2	.1039 6613	.1040 3949	.1039 4420	.1039 6553
1.3	.1159 2817	.1159 6501	.1159 0011	.1159 2812
1.4	.1265 2313	.1264 9970	.1264 9154	.1265 2357
1.5	.1355 4205	.1354 4306	.1355 0946	.1355 4289
1.6	.1428 6261	.1426 8117	.1428 3121	.1428 6373
1.7	.1484 4012	.1481 7692	.1484 1169	.1484 4142
1.8	.1522 9605	.1519 5798	.1522 7185	.1522 9741
1.9	.1545 0519	.1541 0377	.1544 8603	.1545 0654
2.0	.1551 8295	.1547 3259	.1551 6915	.1551 8421

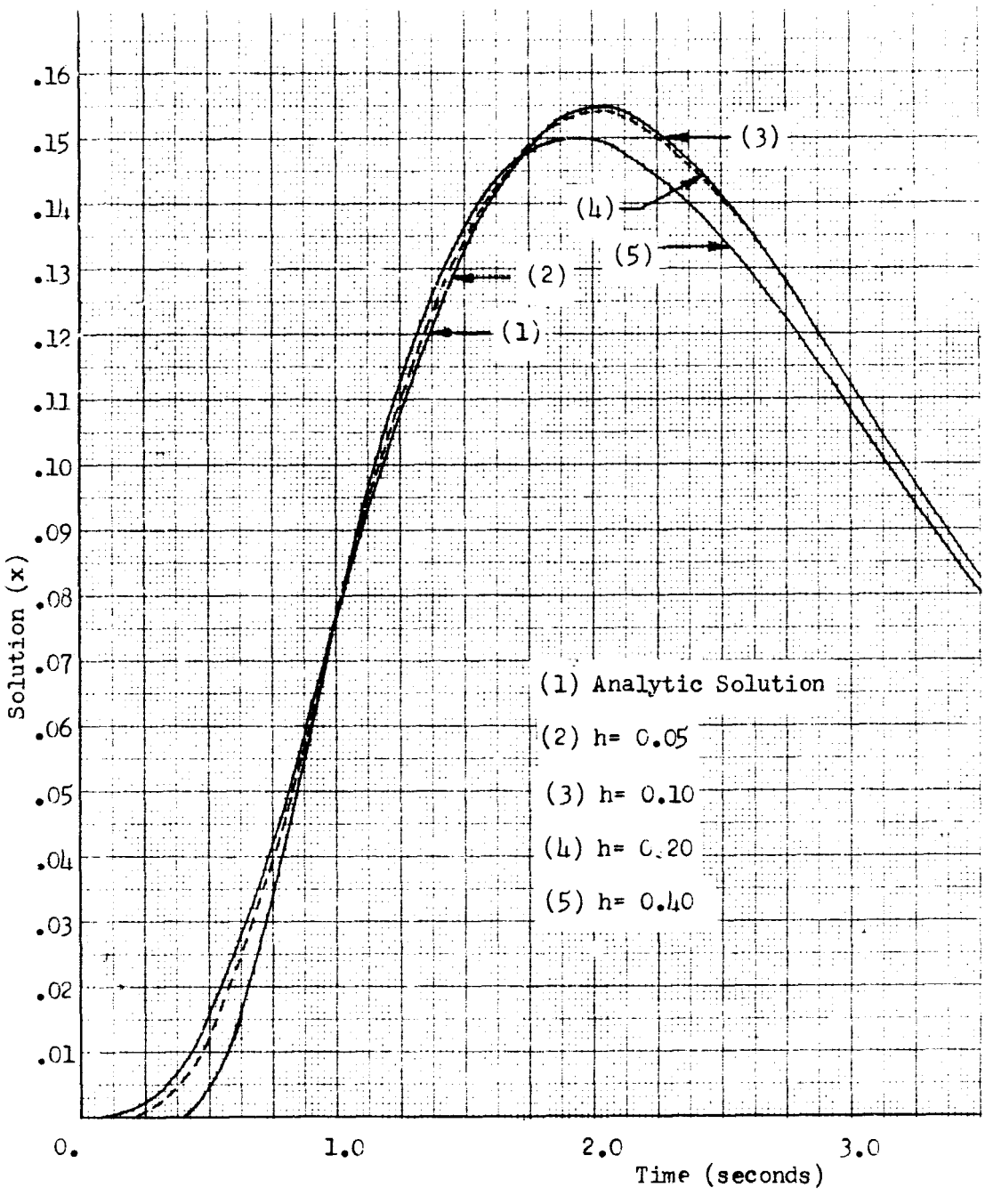

Figure 1...Solution of Third-order System Using a Second-order Runge-Kutta with Various Update Increments.

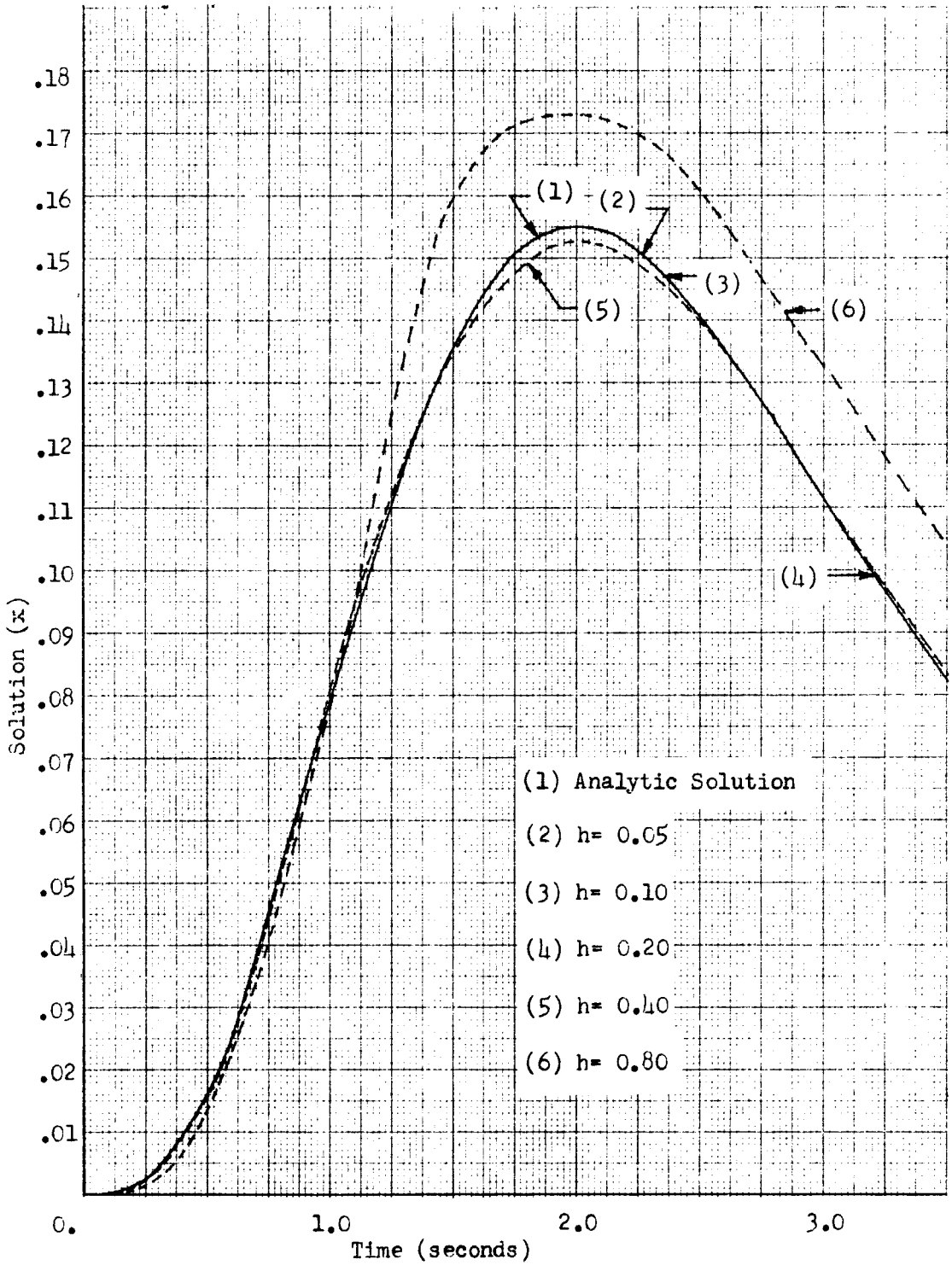

Figure 2...Solution of Third-order System Using a Third-order Runge-Kutta with Various Update Increments.

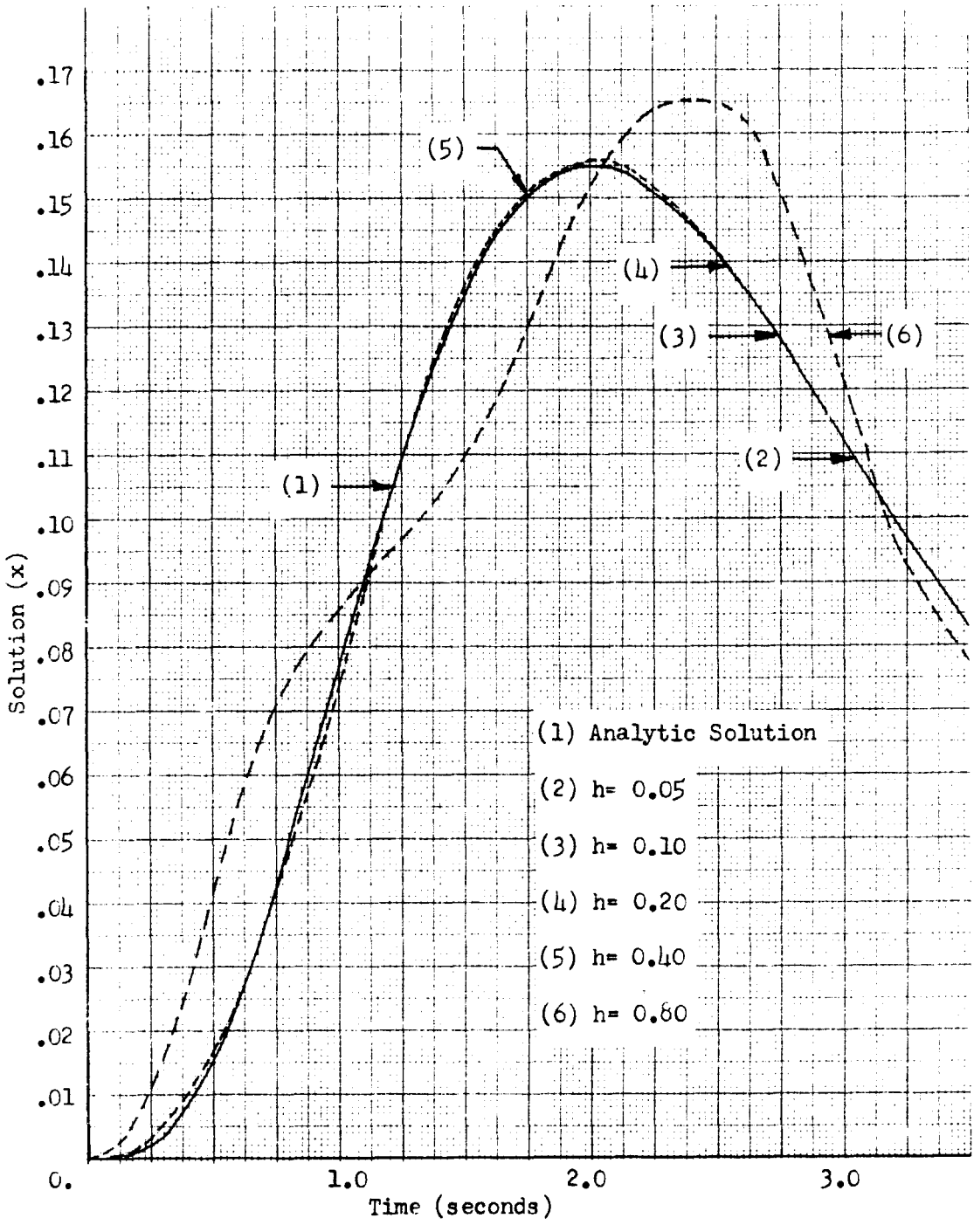

Figure 3...Solution of Third-order System Using a Fourth-order Runge-Kutta with Various Update Increments.

II. MULTI-POINT ITERATION FORMULAS[1,4,5]

There are several well-known formulas which use known values of the solution of $\dot{x} = f(t,x)$ at several points to approximate the solution at future points. Although these methods appear under a variety of names, they all used a polynomial approximation of a function. Perhaps one of the most useful methods is one commonly referred to as the "Milne IV Method," which will be examined in detail in this chapter.

The solution of the first-order differential equation

$$\dot{x} = f(t,x) \tag{1}$$

can be approximated at a series of (n+1) points by an n-th order polynomial in time. Such a polynomial can be written as

$$x \approx P(t) = a_o + a_1 t + a_2 t^2 + \ldots + a_n t^n$$

which is an approximation of the solution of (1). The differential equation becomes $\dot{x} = \frac{d}{dt}(P(t))$. The difference in the solution at two points in time, t_j and t_k, can be found by integrating the differential equation. Then

$$x_k - x_j = \int_{t_j}^{t_k} \frac{d}{dt}(P(t))dt \ . \tag{2}$$

The polynomial $\frac{d}{dt}(P(t))$ can be written, using Lagrange's Interpolation Formula, as

$$\dot{P}(t) = \frac{(t-t_1)(t-t_2) \ \ldots \ (t-t_{n-1})}{(t_o-t_1)(t_o-t_2)\ldots(t_o-t_{n-1})} \dot{x}_o + \frac{(t-t_o)(t-t_2) \ \ldots \ (t-t_{n-1})}{(t_1-t_o)(t_1-t_2)\ldots(t_1-t_{n-1})} \dot{x}_1$$

$$+ \ldots + \frac{(t-t_o) \ldots (t-t_{j-1})(t-t_{j+1}) \ldots (t-t_{n-1})}{(t_j-t_o) \ldots (t_j-t_{j-1})(t_j-t_{j+1}) \ldots (t_j-t_{n-1})} \dot{x}_j + \ldots$$

$$= \sum_{j=0}^{n-1} \left[\prod_{\substack{i=0 \\ i \neq j}}^{n-1} \frac{(t-t_i)}{(t_j-t_i)} \right] \dot{x}_j \quad . \tag{3}$$

If the series of points are chosen so that the time ordinates are equally spaced h units apart with $t_1 - t_o = h$, $t_2 - t_1 = h$, ... , $t_j - t_{j-1} = h$, then (3) reduces to

$$\dot{P}(t) = \sum_{j=0}^{n-1} \left[\prod_{\substack{i=0 \\ i \neq j}}^{n-1} \frac{(t-t_i)}{(j-i)h} \right] \dot{x}_j = \left(\frac{1}{h^{n-1}}\right) \sum_{j=0}^{n-1} \left[\prod_{\substack{i=0 \\ i \neq j}}^{n-1} \frac{(t-t_i)}{(j-i)} \right] \dot{x}_j . \tag{4}$$

Integrating from t_o to t_k, the approximate solution is

$$x_k = x_o + \int_{t_o}^{t_k} \dot{P}(t)dt \quad . \tag{5}$$

The integration is simple to perform; however, due to the number of terms involved, substitution of the limits and simplification are tedious tasks. The integration with n=5 shall be performed to give the set of Five-Point formulas, while other orders will be listed in Appendix B. With n=5 equation (4) becomes

$$\dot{P}(t) = \left(\frac{1}{h^4}\right) \left[\frac{(t-t_1)(t-t_2)(t-t_3)(t-t_4)}{(-1)(-2)(-3)(-4)} \dot{x}_o + \frac{(t-t_o)(t-t_2)(t-t_3)(t-t_4)}{(1)(-1)(-2)(-3)} x_1 \right.$$

$$\left. + \frac{(t-t_o)(t-t_1)(t-t_3)(t-t_4)}{(2)(1)(-1)(-2)} \dot{x}_2 + \frac{(t-t_o)(t-t_1)(t-t_2)(t-t_4)}{(3)(2)(1)(-1)} \dot{x}_3 \right.$$

$$+ \frac{(t-t_o)(t-t_1)(t-t_2)(t-t_3)}{(4)(3)(2)(1)} \dot{x}_4 \Big] \ .$$

$$x_k - x_o = \int_{t_o}^{(t_o+kh)} \dot{P}(t)dt = \left(\frac{1}{5h^4}\right) \left\{ \frac{\dot{x}_o}{24} - \frac{\dot{x}_1}{6} + \frac{\dot{x}_2}{4} - \frac{\dot{x}_3}{6} + \frac{\dot{x}_4}{24} \right\} \left\{ (t_o+kh)^5 - t_o^5 \right\}$$

$$+ \left(\frac{1}{4h^4}\right) \left\{ \left(\frac{-1}{24}\right) (t_1 + t_2 + t_3 + t_4)\dot{x}_o + \left(\frac{1}{6}\right) (t_o + t_2 + t_3 + t_4)\dot{x}_1 \right.$$

$$\left. - \left(\frac{1}{4}\right) (t_o + t_1 + t_3 + t_4)\dot{x}_2 + \left(\frac{1}{6}\right)(t_o + t_1 + t_2 + t_4)\dot{x}_3 - \left(\frac{1}{24}\right)(t_o+t_1+t_2+t_3)\dot{x}_4 \right\}$$

$$\left\{ (t_o+kh)^4 - t_o^4 \right\} + \left(\frac{1}{3h^4}\right) \left\{ \left(\frac{1}{24}\right) (t_1t_2 + t_1t_3 + t_1t_4 + t_2t_3 + t_2t_4 + t_3t_4)\dot{x}_o \right.$$

$$- \left(\frac{1}{6}\right) (t_ot_1 + t_ot_3 + t_ot_4 + t_2t_3 + t_2t_4 + t_3t_4)\dot{x}_1 + \left(\frac{1}{4}\right) (t_ot_1 + t_ot_3 + t_ot_4$$

$$+ t_1t_3 + t_1t_4 + t_3t_4)\dot{x}_2 - \left(\frac{1}{6}\right)(t_ot_1 + t_ot_2 + t_ot_4 + t_1t_2 + t_1t_4 + t_2t_4)\dot{x}_3$$

$$\left. + \left(\frac{1}{24}\right)(t_ot_1 + t_ot_2 + t_ot_3 + t_1t_2 + t_1t_3 + t_2t_3)\dot{x}_4 \right\} \left\{ (t_o + kh)^3 - t_o^3 \right\} +$$

$$\left(\frac{1}{2h^4}\right) \left\{ \left(\frac{-1}{24}\right)(t_1t_2t_3 + t_1t_2t_4 + t_1t_3t_4 + t_2t_3t_4)\dot{x}_o + \left(\frac{1}{6}\right) (t_ot_2t_3 \right.$$

$$+ t_ot_2t_4 + t_ot_3t_4 + t_2t_3t_4)\dot{x}_1 - \left(\frac{1}{4}\right) (t_ot_1t_3 + t_ot_1t_4 + t_ot_3t_4$$

$$+ t_1t_3t_4)\dot{x}_2 + \left(\frac{1}{6}\right) (t_ot_1t_2 + t_ot_1t_4 + t_ot_2t_4 + t_1t_2t_4)\dot{x}_3$$

$$\left. - \left(\frac{1}{24}\right) (t_ot_1t_2 + t_ot_1t_3 + t_ot_2t_3 + t_1t_2t_3)\dot{x}_4 \right\} \left\{ (t_o + kh)^2 - t_o^2 \right\}$$

$$+ \left(\frac{1}{h^4}\right) \left\{ \left(\frac{1}{24}\right) (t_1t_2t_3t_4)\dot{x}_o - \left(\frac{1}{6}\right) (t_ot_2t_3t_4)\dot{x}_1 + \left(\frac{1}{4}\right) (t_ot_1t_3t_4)\dot{x}_2 \right.$$

$$- \left(\frac{1}{6}\right)(t_o t_1 t_2 t_4)\dot{x}_3 + \left(\frac{1}{24}\right)(t_o t_1 t_2 t_3)\dot{x}_4 \Big\} \Big\{ (t_o + \theta h) - t_o \Big\} \quad . \tag{6}$$

Replacing t_1 by $(t_o + h)$, ..., and t_k by $(t_o + kh)$ in (6), and simplifying yields the desired result

$$\begin{aligned} x_k - x_o = \left(\frac{h}{720}\right) \Big[& (6k^5 - 75k^4 + 350k^3 - 750k^2 + 720k)\dot{x}_o \\ & + (-24k^5 + 270k^4 - 1040k^3 + 1440k^2)\dot{x}_1 \\ & + (36k^5 - 360k^4 + 1140k^3 - 1080k^2)\dot{x}_2 \\ & + (-24k^5 + 210k^4 - 560k^3 + 480k^2)\dot{x}_3 \\ & + (6k^5 - 45k^4 + 110k^3 - 90k^2)\dot{x}_4 \Big] \quad . \end{aligned} \tag{7}$$

Equation (7) is a good approximation of x over the interval $x_o \le x \le x_4$ (since n = 5). Evaluation of (7) for k = 1,2,3, and 4 gives the set of five-point iteration formulas

$$\begin{aligned} x_1 - x_o &= \frac{h}{720}(251\,\dot{x}_o + 646\,\dot{x}_1 - 264\,\dot{x}_2 + 106\,\dot{x}_3 - 19\,\dot{x}_4) \\ x_2 - x_o &= \frac{h}{90}(29\,\dot{x}_o + 124\,\dot{x}_1 + 24\,\dot{x}_2 + 4\,\dot{x}_3 - \dot{x}_4) \\ x_3 - x_o &= \frac{h}{80}(27\,\dot{x}_o + 102\,\dot{x}_1 + 72\,\dot{x}_2 + 42\,\dot{x}_3 - 3\,\dot{x}_4) \\ x_4 - x_o &= \frac{4h}{90}(7\,\dot{x}_o + 32\,\dot{x}_1 + 12\,\dot{x}_2 + 32\,\dot{x}_3 + 7\,\dot{x}_4) \quad . \end{aligned} \tag{8}$$

The set of formulas in (8) can be used to compute the solution, x, at a series of points x_1, x_2, x_3, and x_4. Unfortunately, the values of $\dot{x}_i$ are not known exactly, but must in turn be approximated from the

differential equation. This suggests the following iterative procedure; a) obtain a set of n-starting values for x_i by Runge-Kutta integration or other self-starting methods (including guessing), b) substitute these values of x_i into the differential equation in (1) to obtain starting values of $\dot{x}_i$, c) use a set of multi-point formulas of appropriate order similar to (8) to obtain a better approximation of x_i, and d) if the change in each x_i is not within the allowable degree of error, repeat steps b) and c) until the values of x_i do converge.

Initial values for the next block of points can be obtained by assuming that the interpolation polynomial through the n-points in the first block is a good approximation of the solution in the second block. A set of so-called predictor formulas results if (7) is evaluated at k=5, 6, 7 and 8. The five-point predictor formulas are

$$
\begin{aligned}
x_5 - x_o &= \frac{5h}{144}\,(19\,\dot{x}_o - 10\,\dot{x}_1 + 120\,\dot{x}_2 - 70\,\dot{x}_3 + 85\,\dot{x}_4) \\
x_6 - x_o &= \frac{3h}{10}\,(11\,\dot{x}_o - 44\,\dot{x}_1 + 96\,\dot{x}_2 - 84\,\dot{x}_3 + 41\,\dot{x}_4) \\
x_7 - x_o &= \frac{7h}{720}\,(1301\,\dot{x}_o - 5894\,\dot{x}_1 + 11256\,\dot{x}_2 - 9674\,\dot{x}_3 + 3731\,\dot{x}_4) \\
x_8 - x_o &= \frac{8h}{45}\,(206\,\dot{x}_o - 944\,\dot{x}_1 + 1716\,\dot{x}_2 - 1424\,\dot{x}_3 + 491\,\dot{x}_4)\ .
\end{aligned}
\tag{9}
$$

The set of starting values, computed using predictor formulas of appropriate order similar to (9), for the second block can be iterated upon using the step by step procedure listed above. Once this set of points converges to the solution in the second block a set of predictor formulas projects the polynomial into the third block, and so forth. In

this manner the solution of the differential equation can be obtained for the entire desired time interval.

A distinct advantage in this method of iteration is its flexibility in allowing for changing time increments between blocks. If, for instance, a particular block is requiring too many iterations to attain the desired degree of convergence, that block may be repeated using a shorter increment. On the other hand, if a sequence converges in one or two iterations and a high degree of accuracy is not particularly desired, the time increment may be increased. Such a flexible nature makes control of error per block a relatively easy task.

The multi-point formulas can be used to solve a set of simultaneous first-order differential equations in precisely the same manner as can Runge-Kutta. In fact, since the error in Runge-Kutta is of order $h^{n+1}f^{(n)}$, the multi-point formulas of order n yield the same degree of accuracy as a Runge-Kutta method. However, if the system is well-behaved over each block, then the multi-point method can yield the same accuracy as a Runge-Kutta method with a considerable reduction in the number of function evaluations required.

Convergence can be accelerated by using each value of x_i immediately after it is computed to get a better estimate of $\dot{x}_i$. This process makes use of the latest information available and, if the iteration converges, should accelerate convergence.

Example

The third-order differential equation in Chapter I is solved by a three-point iteration method using initial guesses of $x_{1,m}=x_{2,m}=x_{3,m}=0$

for m=0,1,2 and h=0.1. All subscripted state variables follow the form that the first subscript specifies the state variable and the second subscript denotes the corresponding time ordinate t_o, t_1, or t_2. The differential equations are

$$\dot{x}_{1,m} = x_{2,m}$$
$$\dot{x}_{2,m} = x_{3,m} \qquad m = 0,\ 1,\ 2$$
$$\dot{x}_{3,m} = -5x_{3,m} - 12x_{2,m} - 8x_{1,m} + t_m \epsilon^{-1.5t_m}$$

and the update relations are

$$x_{n,1} = x_{n,0} + \frac{h}{12}\,(5\dot{x}_{n,0} + 8\dot{x}_{n,1} - \dot{x}_{n,2})$$

$$x_{n,2} = x_{n,0} + \frac{h}{3}\,(\dot{x}_{n,0} + 4\dot{x}_{n,1} + \dot{x}_{n,2})$$

$$e_{out,m} = 8x_{1,m} + x_{2,m}$$

The above steps are then repeated using the new values of $x_{n,m}$ until $x_{n,1}$ and $x_{n,2}$ cease to change or until the change becomes small. Initial values for the next block are

$$x_{n,3} = x_{n,0} + \frac{3h}{4}\,(x_{n,0} + 3x_{n,2}) \qquad n = 1,\ 2,\ 3$$

$$x_{n,4} = x_{n,0} + \frac{4h}{3}(2x_{n,0} - 4x_{n,1} + 5x_{n,2}) \quad .$$

The results of the above operation are listed in Table 2 along with the results using the two-point, four-point, and five-point methods. Figure 4 shows these results in graphical form.

TABLE 2...Third-order System by Multi-Point Iteration Methods with h= 0.1 Seconds.

Time	Analytic Solution	Two-Point Iteration	Three-Point Iteration	Four-Point Iteration	Five-Point Iteration
0.0	.0	.0	.0	.0	.0
0.1	.0001 6449	.0	-.0005 4781	.0001 6331	.0001 6336
0.2	.0012 6963	.0010 9571	-.0016 9735	.0012 7098	.0012 6966
0.3	.0040 6374	.0038 8040	.0013 6717	.0040 6464	.0040 6326
0.4	.0090 0563	.0087 6209	.0067 4005	.0090 0885	.0090 0571
0.5	.0162 5180	.0159 1147	.0145 2436	.0162 5331	.0162 5178
0.6	.0256 9028	.0252 3771	.0244 7152	.0256 8551	.0256 9013
0.7	.0370 0047	.0364 3930	.0363 0335	.0369 9943	.0370 0017
0.8	.0497 2117	.0490 6972	.0494 5227	.0497 2036	.0497 2018
0.9	.0633 1614	.0626 0208	.0634 2909	.0633 1676	.0633 1553
1.0	.0772 3077	.0764 8610	.0776 4121	.0772 3164	.0772 3023
1.1	.0909 3712	.0901 9384	.0915 6503	.0909 3708	.0909 3670
1.2	.1039 6613	.1032 5313	.1047 4895	.1039 6693	.1039 6516
1.3	.1159 2817	.1152 6911	.1167 9306	.1159 2721	.1159 2775
1.4	.1265 2313	.1259 3534	.1274 3000	.1265 2085	.1265 2301
1.5	.1355 4205	.1350 3638	.1364 4039	.1355 3834	.1355 4227
1.6	.1428 6261	.1424 4378	.1437 3210	.1428 5752	.1428 6323
1.7	.1484 4012	.1481 0762	.1492 5218	.1484 3452	.1484 4085
1.8	.1522 9605	.1520 4515	.1530 4443	.1522 8885	.1522 9679
1.9	.1545 0519	.1543 2815	.1551 7846	.1544 9814	.1545 0569
2.0	.1551 8295	.1550 7005	.1557 8297	.1551 7701	.1551 8267

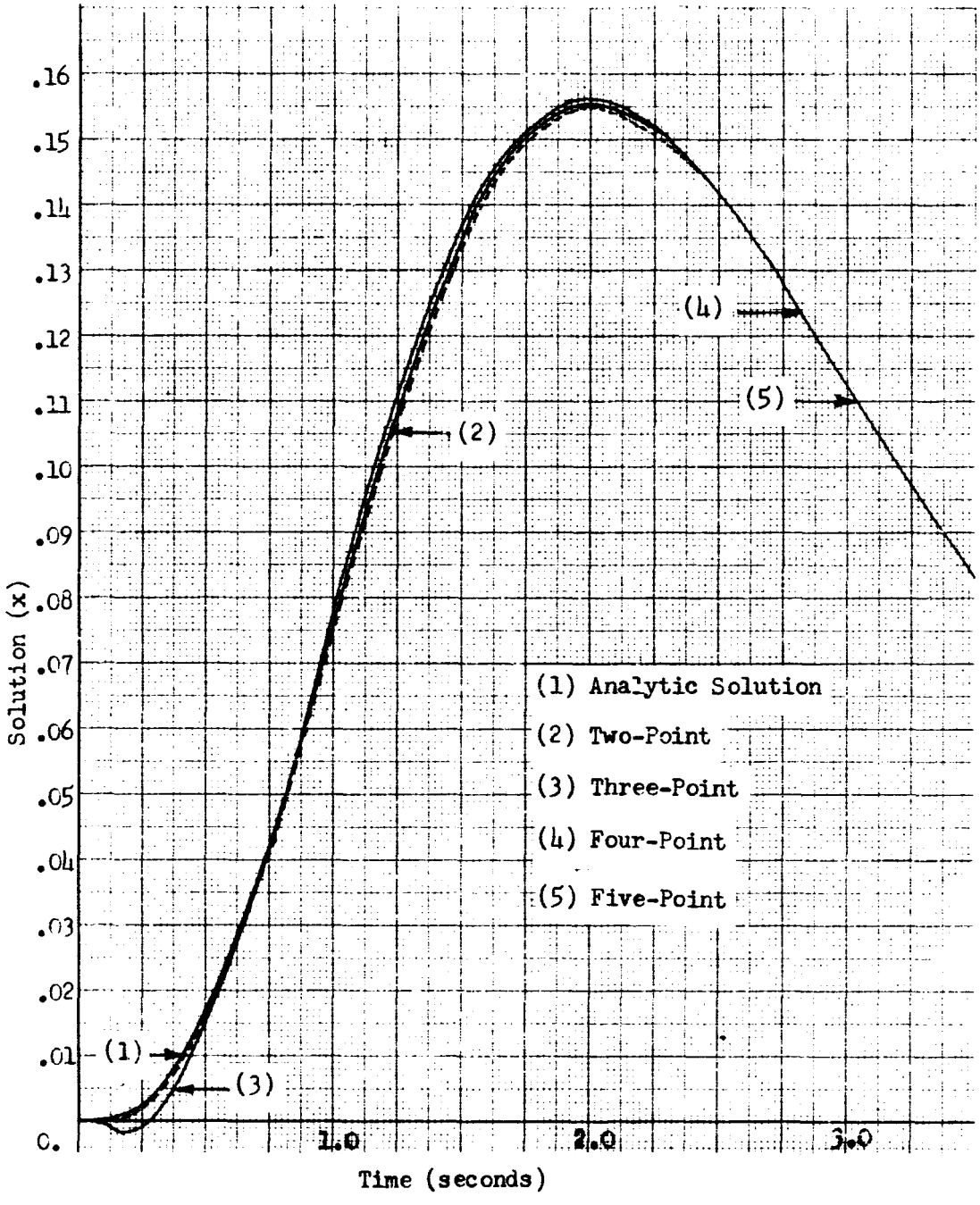

Figure 4...Solution of Third-order System Using Several Orders of Multi-Point Iteration Formulas with h= 0.1 Seconds.

III. DISCRETE FORMS[6,7,8]

In the previous two chapters methods were presented that approached the problem of solving a differential equation by iterating an approximation of the solution. In this chapter a class of methods is developed which uses an approximation of the transfer function between the input (forcing function) and output (solution) of a system of first-order differential equations. Such an approximation results in a formula that relates past inputs and outputs to the present output.

In order to derive the class of discrete methods, first examine a few elementary transformation operations. The Laplace transform of a function of time, f(t), is denoted by $\mathcal{L}(f(t)) = F(s)$. If f(t) is displaced along the time axis an amount of nT seconds, then

$$\mathcal{L}(f(t-nT)\ u(t-nT)) = \epsilon^{-nTs}F(s).$$

To avoid the repeated use of exponential functions, the standard z-transform definition

$$z = \epsilon^{Ts} \tag{1}$$

is used. Thus, z^{-1} represents a time delay of T seconds and z^{-n} represents a delay of nT seconds.

The solution of an n-th order linear differential equation with constant coefficients can be related to the forcing function by a transfer function in the s-domain as

$$\frac{X(s)}{V(s)} = G(s)\ . \tag{2}$$

Here, X(s) denotes the Laplace transform of the output, V(s) the Laplace transform of the input, and G(s) the transfer function. G(s) can also be expressed as the ratio of two polynomials in s as

$$G(s) = \frac{c_1 + c_2 s + \ldots + c_m s^m}{d_1 + d_2 s + \ldots + d_n s^n} \quad . \tag{3}$$

Multiplying (3) by s^{-n}/s^{-n} gives

$$G(s) = \frac{c_1 s^{-n} + c_2 s^{-n+1} + \ldots + c_m{}^{-n+m}}{d_1 s^{-n} + d_2 s^{-n+1} + \ldots + d_n} \quad . \tag{4}$$

If equation (1) is solved for s^{-K}, the result is $s = \frac{1}{T} \ln z$, or

$$s^{-1} = \frac{T}{\ln z} \; , \; s^{-2} = \left(\frac{T}{\ln z}\right)^2 \; , \; \ldots \; , \; s^{-K} = \left(\frac{T}{\ln z}\right)^K \; . \tag{5}$$

To transform (4) into a form that can be digitally simulated, an approximation for s^{-K} must be made such that it is both accurate and simple to substitute into (4). Clearly the problem is to approximate $\ln z$ with some kind of a truncated series. A well known expansion for $\ln z$ is

$$\ln z = 2 \left[\left(\frac{z-1}{z+1}\right) + \frac{1}{3}\left(\frac{z-1}{z+1}\right)^3 + \frac{1}{5}\left(\frac{z-1}{z+1}\right)^5 + \ldots \right] , \tag{6}$$

which converges for all positive z.

A classic substitution, known as the Tustin substitution, results if only the first term of the series is used. The Tustin substitutions are

$$s^{-1} = \frac{T}{\ln z} \simeq \frac{T}{2\left[\frac{z-1}{z+1}\right]} = \frac{T}{2}\left[\frac{z+1}{z-1}\right] \tag{7}$$

$$s^{-2} = \left(\frac{T}{2}\left[\frac{z+1}{z-1}\right]\right)^2$$
$$\vdots$$
$$s^{-K} = \left(\frac{T}{2}\left[\frac{z+1}{z-1}\right]\right)^K .$$

The series in (6) is most accurately approximated with one term if z is in the neighborhood of unity. Likewise, lower powers of lnz are more accurately approximated with one term than are higher powers of lnz. Thus, a requirement for accuracy is that z be very nearly unity.

Another substitution method, called the z-form substitution results if $\frac{1}{\ln z}$ is expanded by synthetic division and only the principal part and constant term are retained. For convenience define

$$u = \frac{z-1}{z+1} \quad , \tag{8}$$

then

$$\ln z = 2\left(u + \frac{u^3}{3} + \frac{u^5}{5} + \ldots\right) .$$

$$\frac{1}{\ln z} = \frac{1}{2}\left(\frac{1}{u} - \frac{1}{3}u - \frac{4}{45}u^3 - \frac{44}{945}u^5 - \cdots\right)$$

$$\frac{1}{\ln z} \approx \frac{1}{2}\left(\frac{1}{u}\right) = \frac{1}{2}\left(\frac{1}{\frac{z-1}{z+1}}\right) = \frac{1}{2}\left(\frac{z+1}{z-1}\right)$$

$$\left(\frac{1}{\ln z}\right)^2 \approx \frac{1}{4}\left(\frac{1}{u^2} - \frac{2}{3}\right) = \frac{1}{4}\left(\left(\frac{1}{\frac{z-1}{z+1}}\right)^2 - \frac{2}{3}\right) = \frac{1}{12}\left(\frac{z^2+10z+1}{(z-1)^2}\right)$$

$$\left(\frac{1}{\ln z}\right)^3 \approx \frac{1}{8}\left(\frac{1}{u^3} - \frac{1}{u}\right) = \frac{1}{8}\left(\left(\frac{1}{\frac{z-1}{z+1}}\right)^3 - \left(\frac{1}{\frac{z-1}{z+1}}\right)\right) = \frac{1}{2}\left(\frac{z(z+1)}{(z-1)^3}\right).$$

Using these approximations in (5), the z-form substitutions are

$$s^{-1} = \frac{T}{2}\left(\frac{z+1}{z-1}\right)$$

$$s^{-2} = \frac{T^2}{4}\left(\frac{z^2+10z+1}{(z-1)^2}\right) \qquad (9)$$

$$s^{-3} = \frac{T^3}{2}\left(\frac{z(z+1)}{(z-1)^3}\right)$$

$$\vdots$$

Since the z-form substitutions given by (9) involve more terms of the series expansion of $\left(\frac{1}{\ln z}\right)$ than do the Tustin substitutions, the z-forms can be expected to be more accurate. Clearly, the error in either method is introduced by truncating the infinite series expansion of lnz after a very few terms.

Returning to the question of simplifying (4), there are now available two substitutions for s^{-K}, the Tustin and z-forms given in (7) and (9). Upon inserting one or the other into (4) the transfer function becomes

$$G(z) = \frac{a_o z^n + a_1 z^{n-1} + \ldots + a_{n-1} z^1 + a_n}{z^n + b_1 z^{n-1} + \ldots + b_{n-1} z^1 + b_n},$$

where the coefficients have been normalized such that $b_o = 1$.

$$\frac{X(z^{-1})}{V(z^{-1})} = G(z^{-1}) = \frac{a_o + a_1 z^{-1} + \ldots + a_{n-1} z^{1-n} + a_n z^{-n}}{1 + b_1 z^{-1} + \ldots + b_{n-1} z^{1-n} + b_n z^{-n}} \qquad (10)$$

$$X(z^{-1})\left[1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}\right]$$

$$= V(z^{-1})\left[a_o + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}\right] \qquad (11)$$

Since multiplication by z^{-1} in the z-domain corresponds to a time delay of T seconds in the time domain, (11) can be written in the time domain

as

$$x(kT) + b_1x(kT-T) + b_2x(kT-2T) + \ldots + b_nx(kT-nT)$$

$$= a_ov(kT) + a_1v(kT-T) + a_2v(kT-2T) + \ldots + a_nv(kT-nT) \quad . \tag{12}$$

Solving (12) for x(kT) yields the recursive relationship

$$x(kT) = a_ov(kT) + a_1v(kT-T) + a_2v(kT-2T) + \ldots + a_nv(kT-nT)$$

$$- b_1x(kT-T) - b_2x(kT-2T) - \ldots - b_nx(kT-nT) \quad . \tag{13}$$

Equation (13) expresses the output at a time kT in terms of a linear combination of the past n outputs and the present and past n values of the input (where n is the order of the system). Once the "a" and "b" coefficients have been computed, the task of repeatedly applying (13) is an easy matter for the computer. Presuming that the substitution used is accurate, then the output x(kT) represents the desired solution of the set of differential equations.

The error involved in the discrete substitution is contributed, in large, by truncation of the infinite series of lnz. However, the truncation error is closely related to the magnitude of z, which in turn is dependent upon the sampling period T and the complex variable s. The first-order approximation is exact if z equals unity. Since $z=\epsilon^{sT}$, then

$$|z| = |\epsilon^{sT}| = |\epsilon^{(\sigma+j\omega)T}| = |\epsilon^{\sigma T}|\ |\epsilon^{j\omega T}| = \epsilon^{\sigma T},$$

where s has been written in complex form. A time constant is related to the real part of s, σ, as $\tau = \frac{1}{|\sigma|}$. Then the magnitude of z is given as

$$|z| = \epsilon^{\sigma T} = \epsilon^{-T/\tau} ,$$

which must be near unity for the Tustin and z-form substitutions to be accurate.

A requirement for selecting an update interval is

$$T \ll \tau . \tag{14}$$

Since higher-order systems contain several natural time constants, the update interval must be chosen so that (14) is true for all of the time constants. Equation (14) provides a logical means of selecting an initial estimate of a workable sampling period. If the error is too great, then this value can be decreased until the desired accuracy is achieved.

Other transform or "pseudo-transform" methods can be obtained which are suitable for use in the recursive formula given by (13). For instance, the standard z-transform of G(s) can be calculated, this being a difficult task for higher-order systems.

Example

The example used is one described by the differential equation

$$\dddot{x} + 5\ddot{x} + 12\dot{x} + 8x = v(t)$$

and

$$\epsilon_o(t) = 8x + \dot{x},$$

where $\ddot{x} = \dot{x} = x = 0$ at $t = 0$.

In Laplace transform notation, this is

$$(s^3 + 5s^2 + 12s + 8)\, X(s) = V(s)$$

$$E_o(s) = (8 + s)\, X(s) \quad , \quad \text{thus}$$

$$\frac{E_o(s)}{V(s)} = \frac{s+8}{s^3+5s^2+12s+8} = \frac{s^{-2}+8s^{-3}}{1+5s^{-1}+12s^{-2}+8s^{-3}} \quad .$$

Using the Tustin substitution, the result is

$$\frac{E_o(z)}{V(z)} = \frac{\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^2 + 8\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^3}{1 + 5\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right] + 12\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^2 + 8\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^3}$$

$$= \frac{\frac{T^2}{4}(z+1)^2(z-1) + T^3(z+1)^3}{(z-1)^3 + \frac{5T}{2}(z+1)(z-1)^2 + 3T^2(z+1)^2(z-1) + T^3(z+1)^3}$$

$$= \frac{\left(\frac{T^2}{4}+T^3\right) + \left(\frac{T^2}{4}+3T^3\right)z^{-1} + \left(-\frac{T^2}{4}+3T^3\right)z^{-2} + \left(-\frac{T^2}{4}+T^3\right)z^{-3}}{\left(1+\frac{5T}{2}+3T^2+T^3\right) + \left(-3-\frac{5T}{2}+3T^2+3T^3\right)z^{-1} + \left(3-\frac{5T}{2}-3T^2+3T^3\right)z^{-2} + \left(-1+\frac{5T}{2}-3T^2+T^3\right)z^{-3}}$$

With appropriate z-form substitutions, the result is

$$\frac{E_o(z)}{V(z)} = \frac{\frac{T^2}{12}\left(\frac{z^2+10z+1}{(z-1)^2}\right) + 8\left(\frac{T^3}{2}\right)\left(\frac{z(z+1)}{(z-1)^3}\right)}{1 + 5\left(\frac{T}{2}\right)\left(\frac{z+1}{z-1}\right) + 12\left(\frac{T^2}{12}\right)\left(\frac{z^2+10z+1}{(z-1)^2}\right) + 8\left(\frac{T^3}{2}\right)\left(\frac{z(z+1)}{(z-1)^3}\right)}$$

$$= \frac{\left(\frac{T^2}{12}\right) + \left(\frac{3T^2}{4}+4T^3\right)z^{-1} + \left(\frac{-3T^2}{4}+4T^3\right)z^{-2} + \left(\frac{-T^2}{12}\right)z^{-3}}{\left(1+\frac{5T}{2}+T^2\right) + \left(-3-\frac{5T}{2}+9T^2+4T^3\right)z^{-1} + \left(3-\frac{5T}{2}-9T^2+4T^3\right)z^{-2} + \left(-1+\frac{5T}{2}-T^2\right)z^{-3}}$$

Consider the criterion presented earlier for selecting T. The roots of the characteristic equation (i.e. denominator of G(s)) are -1 and $-2 \pm j2$. The two natural time constants are 1 and .5, so the update

interval must be chosen to be much less than 0.5 seconds to yield reasonably accurate results. Thus T is selected to be 0.1 and substituted into the recursive formulas. The Tustin equation becomes

$$\frac{E_o(z)}{V(z)} = \frac{.0035 + .0055z^{-1} + .0005z^{-2} - .0015z^{-3}}{1.2810 - 3.2170z^{-1} + 2.7230z^{-2} - .7790z^{-3}}$$

$$= \frac{.00273 + .00429z^{-1} + .00039z^{-2} - .00117z^{-3}}{1 - 2.51132z^{-1} + 2.12568z^{-2} - .60812z^{-3}},$$

while the z-form substitution becomes

$$\frac{E_o(z)}{V(z)} = \frac{.0008 + .0115z^{-1} - .0035z^{-2} - .0008z^{-3}}{1.2600 - 3.1560z^{-1} + 2.6640z^{-2} - .7600z^{-3}}$$

$$= \frac{.00063 + .00913z^{-1} - .00278z^{-2} - .00063z^{-3}}{1 - 2.50476z^{-1} + 2.11429z^{-2} - .60317z^{-3}}.$$

The results of the two methods for an input of $te^{-1.5t}$ are given in Table 3 and plotted in Figure 5. Although for a sampling period of 0.1 seconds both the Tustin and z-form methods give about three-decimal accuracy, the z-form solution is slightly more accurate.

A complete list of both the Tustin and z-form substitutions up through s^{-7} is given in Appendix C for reference.

TABLE 3...Third-order System by Discrete Methods with T= 0.1 Seconds.

Time	Analytic Solution	Tustin Substitution	z-form Substitution
0.0	.0	.0	.0
0.1	.0001 6449	.0002 3517	.0000 5693
0.2	.0012 6963	.0013 6494	.0010 2614
0.3	.0040 6374	.0041 2030	.0036 8960
0.4	.0090 0563	.0089 6711	.0085 2894
0.5	.0162 5180	.0160 7988	.0157 1148
0.6	.0256 9028	.0253 6837	.0251 2705
0.7	.0370 0047	.0365 3219	.0364 5037
0.8	.0497 2117	.0491 2609	.0492 1175
0.9	.0633 1614	.0626 2440	.0628 6496
1.0	.0772 3077	.0764 7788	.0768 4569
1.1	.0909 3712	.0901 5938	.0906 1773
1.2	.1039 6613	.1031 9719	.1037 0581
1.3	.1159 2817	.1151 9663	.1157 1631
1.4	.1265 2313	.1258 5118	.1263 4717
1.5	.1355 4205	.1349 4507	.1353 8927
1.6	.1428 6261	.1423 4937	.1427 2142
1.7	.1484 4012	.1480 1356	.1483 0099
1.8	.1522 9605	.1519 5422	.1521 5193
1.9	.1545 0519	.1542 4249	.1543 5169
2.0	.1551 8295	.1549 9115	.1550 1811

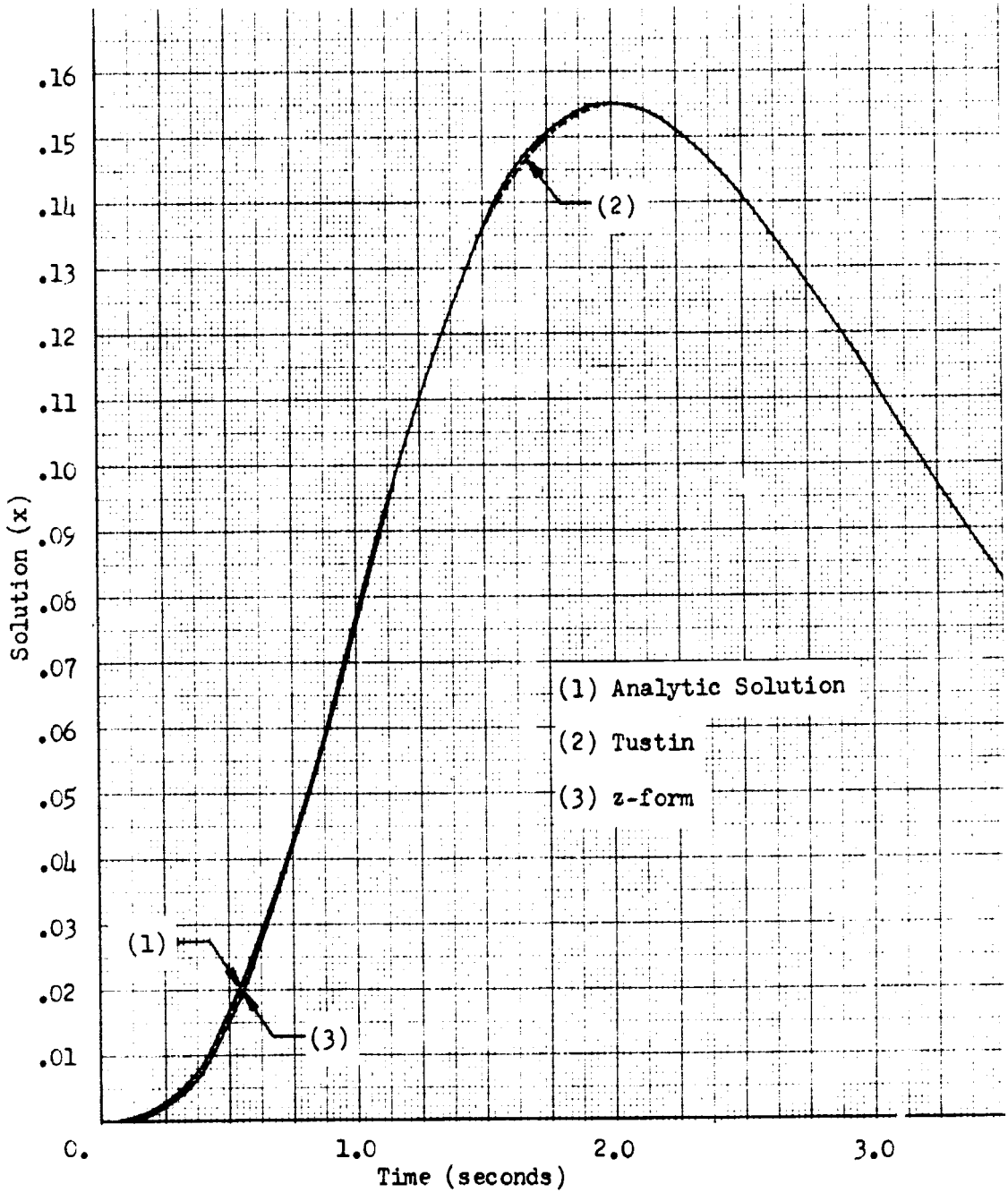

Figure 5...Solution of Third-order System Using Discrete Methods with T= 0.1 Seconds.

IV. STATE VARIABLE SOLUTION[11, 12]

Several methods have been presented in the preceding chapters that utilize either series expansions of the solution, iterative methods of approximating the solution, or approximations of the transfer function. Instead of using any such approximation, this chapter will deal with the actual solution of a set of simultaneous first-order differential equations. To reduce the results to a usable form, approximations will be made concerning the inputs (forcing functions). Although systems with multiple-order roots can be covered by an extension of this method, the following work will be based on systems with simple roots.

The behavior of a linear system can be described by a set of first-order differential equations of the form

$$\dot{\underline{X}}(t) = A\ \underline{X(t)} + B\underline{V(t)} \tag{1}$$

where $\underline{X(t)}$ is an m^{th} order vector, $\underline{V(t)}$ is an n^{th} order vector, A is an m by m matrix, B is an m by n matrix, and $\dot{\underline{X}}(t)$ is the time derivative of $\underline{X(t)}$. The vector $\underline{X(t)}$ is called the state vector, and its components $x_1, x_2, \ldots, x_m$ are the state variables of the system. The vector $\underline{V(t)}$ is the input vector, and its components $v_1, v_2, \ldots, v_n$ are the physical inputs to the system and are real functions of time. The output of the system can be written as a linear combination of the state variables

and the inputs as

$$e_o(t) = C\,\underline{X(t)} + D\,\underline{V(t)} \tag{2}$$

where C is a 1 by m matrix and D is a 1 by n matrix. If the elements of A, B, C, and D are constant, the equations of the system are linear ordinary differential equations with non-time varying coefficients. The equations in (1) and (2) are commonly referred to as the "standard form" of the state equations.

If the coefficients of the A matrix are such that all of the eigenvalues of A are distinct, there exist an m by m transformation matrix M such that

$$M^{-1}AM = \Lambda \tag{3}$$

where M^{-1} is the inverse of M and Λ is a diagonal matrix whose elements are the eigenvalues of A, $\lambda_1, \lambda_2, \ldots, \lambda_m$. The matrix M can be used to transform the state vector $\underline{X(t)}$ into a new state vector $\underline{Y(t)}$ such that

$$\underline{X(t)} = M\,\underline{Y(t)}, \text{ and } \dot{\underline{X}}(t) = M\,\dot{\underline{Y}}(t). \tag{4}$$

Equations (1) and (2) can be written in terms of the state vector $\underline{Y(t)}$ as

$$M\,\dot{\underline{Y}}(T) = AM\,\underline{Y(t)} + B\,\underline{V(t)}$$

or

$$\dot{\underline{Y}}(t) = M^{-1} AM \underline{Y}(t) + M^{-1} B \underline{V}(t) = \Lambda \underline{Y}(t) + M^{-1} B\underline{V}(t) \qquad (5)$$

and

$$e_o(t) = CM \underline{Y}(t) + D \underline{V}(t). \qquad (6)$$

Equation (5) is the matrix representation of m linear differential equations, any one of which can be written in the form

$$\dot{y}_i(t) = \lambda_i y_i(t) + \sum_{j=1}^{n} k_{ij} v_j(t) \qquad 1 \leq i \leq m , \qquad (7)$$

where the k_{ij}'s are the elements of $M^{-1}B$. Note that the equation in (7) is a function of $y_i(t)$ and the inputs, and is not a function of the remaining y(t) state variables. The equations in (5) are therefore m uncoupled equations, and the elements of $\underline{Y}(t)$ are called the canonical state variables of the system.

Equations (5) and (6) can be obtained from (1) and (2) through the evaluation of the transformation matrix M and a good deal of matrix manipulation. However, this time consuming task can be by-passed by using an s-domain analysis which is more familiar to the engineer. Equation (5) can be obtained through an s-domain analysis by grouping the constants in the partial fraction expansions of the transfer functions of the system under investigation to give the matrices $M^{-1}B$, CM, and D.

Λ is a diagonal matrix whose elements are the poles of the transfer functions.

To obtain (5) through an s-domain analysis, each transfer function relating the output $E_o(s)$ to the inputs $V_1(s)$, $V_2(s)$, ... and $V_n(s)$ is partial fractioned. Since the system is linear, superposition holds, and the output is the sum of the responses of the system to each input separately and may be written as

$$E_o(s) = \sum_{j=1}^{n} \left\{ \sum_{i=1}^{m} \frac{K_{ij}}{s-\lambda_i} \right\} V_j(s) + \sum_{j=1}^{n} K_j V_j(s) \qquad (8)$$

where the K_{ij}'s are the residues of the transfer function relating $E_o(s)$ to $V_j(s)$ at the pole λ_i and $K_j = \lim_{s \to \infty} E_o(s)/V_j(s)$. The order of the summation in the first part of (8) can be reversed giving

$$E_o(s) = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{K_{ij}}{s - \lambda_i} V_j(s) + \sum_{j=1}^{n} K_j V_j(s) \qquad (9)$$

or

$$E_o(s) = \sum_{i=1}^{m} Y_i(s) + \sum_{j=1}^{n} K_j V_j(s) \qquad (10)$$

where

$$Y_i(s) = \sum_{j=1}^{n} \frac{K_{ij}}{s - \lambda_i} V_j(s). \qquad (11)$$

The components of $Y_i(s)$ can be written as

$$\underline{Y(s)} = \begin{bmatrix} \frac{1}{s-\lambda_1} & 0 & \cdots \\ 0 & \frac{1}{s-\lambda_2} & \\ \vdots & \vdots & \\ 0 & \cdots & \frac{1}{s-\lambda_m} \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & & \\ \vdots & & & \\ K_{m1} & & & K_{mn} \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \\ \vdots \\ V_n(s) \end{bmatrix} . \tag{12}$$

Equation (12) can be related to (5) by taking the Laplace transform of (5). The Laplace transform of (5) is

$$s\,\underline{Y(s)} - \underline{Y(0)} = \Lambda\,\underline{Y(s)} + M^{-1}B\,\underline{V(s)} \tag{13}$$

or

$$\underline{Y(s)} = [sI - \Lambda]^{-1}\,\underline{Y(0)} + [sI - \Lambda]^{-1} M^{-1}B\,\underline{V(s)}.$$

Since $[sI-\Lambda]$ is a diagonal matrix, its inverse is the diagonal matrix whose elements are the reciprocals of the elements of $[sI-\Lambda]$ and is given by

$$[sI-\Lambda]^{-1} = \begin{bmatrix} \frac{1}{s-\lambda_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{s-\lambda_2} & & \\ \vdots & & & \\ 0 & & & \frac{1}{s-\lambda_m} \end{bmatrix} . \qquad (14)$$

The initial condition terms in the transfer function analysis of (12) are zero due to the way that transfer functions are defined. Therefore, in order to compare (12) and (13), $\underline{Y(0)}$ is (13) must be set equal to zero. Equation (12) and (13) are identical under the assumption of zero initial condition if $M^{-1}B = K$ is the m by n constant matrix in (12). A comparison of (10) with the Laplace transform of (6) shows that, for this particular scaling of the canonical state variable, CM is a 1 by n matrix with all elements one and that $D = [K_1, K_2, \ldots, K_n]$. Therefore, $M^{-1}B$, CM and D can be obtained from the transfer functions of the system.

The solution of (5) can be obtained by taking the inverse Laplace transform of (13), with (14) substituted for $[sI-\Lambda]^{-1}$. The canonical state variables are

$$\underline{Y(t)} = \mathcal{L}^{-1} \left\{ [sI-\Lambda]^{-1} \underline{Y(0)} \right\}$$

$$+\int_0^t \begin{bmatrix} \epsilon^{\lambda_1\tau} & 0 & \cdots & 0 \\ 0 & \epsilon^{\lambda_2\tau} & & \\ \vdots & & & \\ 0 & & & \epsilon^{\lambda_m\tau} \end{bmatrix} M^{-1}B \begin{bmatrix} v_1(t-\tau) \\ v_2(t-\tau) \\ \vdots \\ v_n(t-\tau) \end{bmatrix} d\tau \quad (15)$$

Since in the computer each input $v_i(t)$ is represented by a set of discrete points, the evaluation of the convolution integrals in (15) are out of the question. However, if each element of the input vector can be approximated with a first-order curve connecting two data points, then (15) will reduce to a very useful form. Such an approximation can be written for the first interval as $v_i(t) = v_i(0) + t\, w_i(0)$ where $w_i(0)$ is the slope of the first-order approximation. The convolution integral can then be written as

$$\int_0^t \begin{bmatrix} \epsilon^{\lambda_1\tau} & & \cdots & 0 \\ 0 & \epsilon^{\lambda_2\tau} & & \\ \vdots & & & \\ 0 & & & \epsilon^{\lambda_m\tau} \end{bmatrix} M^{-1}B \begin{bmatrix} v_1(0) + (t-\tau)\, w_1(0) \\ v_2(0) + (t-\tau)\, w_2(0) \\ \vdots \\ v_n(0) + (t-\tau)\, w_n(0) \end{bmatrix} d\tau$$

$$= \begin{bmatrix} \frac{\epsilon^{\lambda_1 t}-1}{\lambda_1} & 0 & \cdots & 0 \\ 0 & \frac{(\epsilon^{\lambda_2 t}-1)}{\lambda_2} & & \\ \vdots & & & \\ 0 & \cdots & & \frac{(\epsilon^{\lambda_n t}-1)}{\lambda_n} \end{bmatrix} M^{-1}B\ \underline{V(0)}$$

$$+ \begin{bmatrix} \frac{(\epsilon^{\lambda_1 t}-1)}{\lambda_1^2} - \frac{t}{\lambda_1} & & \cdots & 0 \\ 0 & \frac{(\epsilon^{\lambda_2 t}-1)}{\lambda_2^2} - \frac{t}{\lambda_2} & & \\ \vdots & & & \\ 0 & \cdots & & \frac{(\epsilon^{\lambda_n t}-1)}{\lambda_n^2} - \frac{t}{\lambda_n} \end{bmatrix} M^{-1}B\ \underline{W(0)} \qquad (16)$$

To develop a repetitive method of evaluating $\underline{Y(t)}$ at a series of points in time, spaced h seconds apart, (16) is substituted into (15) and then (15) is evaluated at t = h to give

$$\underline{Y(h)} = \text{diag}\left[\ \epsilon^{\lambda_1 h}, \epsilon^{\lambda_2 h}, \ldots, \epsilon^{\lambda_m h}\right] \underline{Y(0)}$$

$$+ \text{diag}\left[\ \frac{1}{\lambda_1}\ (\epsilon^{\lambda_1 h}-1), \quad \frac{1}{\lambda_2}\ (\epsilon^{\lambda_2 h}-1) \ \ldots,\right.$$

$$\frac{1}{\lambda_m}\left(\epsilon^{\lambda_m h}-1\right)\Big] M^{-1}B\,\underline{V(0)} \qquad (17)$$

$$+\ \mathrm{diag}\Big[\frac{1}{h\lambda_1^2}\left(\epsilon^{\lambda_1 h}-1\right)-\frac{1}{\lambda_1},\ \frac{1}{h\lambda_2^2}$$

$$\left(\epsilon^{\lambda_2 h}-1\right)-\frac{1}{\lambda_2},\ \ldots,\ \frac{1}{h\lambda_m^2}\left(\epsilon^{\lambda_m h}-1-\frac{1}{\lambda_m}\right)\Big] M^{-1}B\,\underline{W(0)}.$$

Equation (17) can be used to find the value of the canonical state variable at time h if they are known at time zero. In fact (17) can be used to find $\underline{Y(t_o + h)}$ if $\underline{Y(t_o)}$, $\underline{V(t_o)}$ and $\underline{V(t_o + h)}$ are known. For this purpose $\underline{Y(0)}$, $\underline{V(0)}$, and $\underline{W(0)}$ in (17) are replaced by $\underline{Y(t_o)}$, $\underline{V(t_o)}$ and $\left\{\underline{V(t_o + h)} - \underline{V(t_o)}\right\} / h$ respectively to yield

$$\underline{Y(t_o + h)} = \mathrm{diag}\left[\epsilon^{\lambda_1 h},\ \epsilon^{\lambda_2 h},\ \ldots,\ \epsilon^{\lambda_m h}\right]\underline{Y(t_o)}$$

$$+\ \mathrm{diag}\Big[\frac{1}{h\lambda_1^2}\left(\epsilon^{\lambda_1 h}-1\right)+\frac{\epsilon^{\lambda_1 h}}{\lambda_1},\ \ldots,$$

$$\frac{1}{h\lambda_m^2}\left(\epsilon^{\lambda_m h}-1\right)+\frac{\epsilon^{\lambda_m h}}{\lambda_m}\Big] M^{-1}\,B\,\underline{V(t_o)}$$

$$+\ \mathrm{diag}\Big[\frac{1}{h\lambda_1^2}\left(\epsilon^{\lambda_1 h}-1\right)-\frac{1}{\lambda_1},\ \ldots,$$

$$\frac{1}{h\lambda_m^2}\left(\epsilon^{\lambda_m h}-1\right)-\frac{1}{\lambda_m}\Big] M^{-1}B\,\underline{V(t_o + h)}. \qquad (18)$$

Equation (18) is a matrix representation of m algebraic equations that can be used to calculate the canonical state variables at time $t_o + h$ from the values of the canonical state variables at time t and

the values of the inputs at time t_o and $t_o + h$. Equation (6) can then be used to calculate the required output at time $t_o + h$.

The preceding results can be combined to yield the following repetitive method for simulation of a linear system. The step-by-step procedure can be listed as:

1) Obtain a set of n transfer functions relating the n inputs and the desired output.
2) Determine the eigenvalues of the system, i.e., the poles of the transfer functions.
3) Compute the residues of each transfer function in 1) at each pole in 2) to determine the $M^{-1}B$ matrix.
4) Compute the limit as s approaches infinity of each transfer function in 1) to determine the D matrix.
5) Compute, using a selected step-length h, matrices

$$E = \text{diag}\left[\epsilon^{\lambda_1 h}, \epsilon^{\lambda_2 h}, \ldots, \epsilon^{\lambda_m h}\right]$$

$$F = \text{diag}\left[\frac{1}{h\lambda_1^2}(\epsilon^{\lambda_1 h}-1) + \frac{\epsilon^{\lambda_1 h}}{\lambda_1}, \ldots, \frac{1}{h\lambda_m^2}(\epsilon^{\lambda_m h}-1) + \frac{\epsilon^{\lambda_m h}}{\lambda_m}\right] M^{-1}B$$

, and $G = \text{diag}\left[\frac{1}{h\lambda_1^2}(\epsilon^{\lambda_1 h}-1)\right.$

$$- \frac{1}{\lambda_1} , \ldots, \frac{1}{h\lambda_m^2} (\epsilon^{\lambda_m h} - 1) - \frac{1}{\lambda_m} \Big] M^{-1} B$$

6) Using known values of the canonical state variables at t_o, calculate the up-dated state vector

$$\underline{Y(t_o + h)} = E \underline{Y(t_o)} + F \underline{V(t_o)} + G \underline{V(t_o + h)}$$

7) Compute the desired output

$$e_o(t_o + h) = \sum_{i=1}^{m} y_i(t_o + h) + D \underline{Y(t_o + h)}$$

8) Repeat steps 6) and 7) until the solution is found for the complete desired time interval.

The only approximation used in the development of the preceding procedure was the approximation of the input by a straight line over each iteration interval. Therefore, the solution is the exact solution for the approximated input except for possible computer round-off error. The first-order input approximation can usually be made as accurate as desired by decreasing the update interval until the input becomes nearly linear over each interval. In most digital simulations the input data is only represented at specified points, requiring some sort of approxi-

mation between points. The contention is that a first-order approximation is as valid as any if the interval between points is made small enough.

A strong point in favor of the developed method is that any update time increment can be used as long as the input approximation is valid. The method is very good for simulating large scale systems, particularly if speed is a factor.

Example

To illustrate how the step by step procedure is used, a system with the transfer function

$$\frac{E_o(s)}{V(s)} = \frac{(s+8)}{(s+1)(s+2+j2)(s+2-j2)}$$

$$= \frac{1.4}{s+1} \quad \frac{-.7-j.1}{s+2+j2} \quad \frac{-.7+j.1}{s+2-j2}$$

is simulated. The eigenvalues of the system are $\lambda_1 = -1$, $\lambda_2 = -2-j2$ and $\lambda_3 = -2 + j2$. The residues at the poles gives

$$M^{-1}B = [1.4, -.7 - j.1, -.7 + j.1]^T.$$

The $\lim_{s \to \infty} G(s) = 0$, therefore $D = 0$.

The E, F and G matrices, with h = 0.1 are:

$$E = \text{diag}\left[\epsilon^{-.1},\ \epsilon^{-.2-j.2},\ \epsilon^{-.2+j.2}\right]$$

$$= \text{diag}\left[.90484,\ .80241 - j.16266,\ .80241 + j.16266\right]$$

$$F = \begin{bmatrix} 1.19905 \\ .30311 - j.30545 \\ .30311 + j.30545 \end{bmatrix}, \qquad G = \begin{bmatrix} 2.73228 \\ -.03298 - j.00256 \\ -.03298 + j.00256 \end{bmatrix}$$

Since the second and third terms of E, F and G are conjugate pairs, and since the output is a real quantity, $y_2(t)$ and $y_3(t)$ are also conjugate pairs. Therefore, it is only necessary to calculate $y_2(t)$. The iteration equations are

$$y_1(t_o + .1) = 0.90484\ y_1(t_o) + 1.19905\ v(t_o) + 2.73228\ v(t_o + ..1)$$

$$y_2(t_o + .1) = (.80241 - j.16266)\ y_2(t_o) + (.30311 - j.30545)\ v(t_o) + (-.03298 - j.00256)\ v(t_o + .1)$$

and

$$e_o(t_o + .1) = y_1(t_o + .1) + 2\mathrm{Re}\{ y_2(t_o + .1) \}$$

The system was programmed on the IBM 7040 digital computer at the Auburn University Computer Center for the input $t\, e^{-1.5t}$. The results, listed in Table 4 and plotted in Figure 6, show that for this particular input and time increment, a linear input approximation is accurate enough to yield three-digit accuracy.

TABLE 4...Third-order System by Canonical State Variable Method with h= 0.1 Seconds.

Time	Analytic Solution	State Variable Solution
0.0	.0	.0
0.1	.0001 6449	.0001 5238
0.2	.0012 6963	.0012 2196
0.3	.0040 6374	.0039 6168
0.4	.0090 0563	.0088 3762
0.5	.0162 5180	.0160 1304
0.6	.0256 9028	.0253 8224
0.7	.0370 0047	.0366 2964
0.8	.0497 2117	.0492 9763
0.9	.0633 1614	.0628 5207
1.0	.0772 3077	.0767 3926
1.1	.0909 3712	.0904 3110
1.2	.1039 6613	.1034 5768
1.3	.1159 2817	.1154 2799
1.4	.1265 2313	.1260 4014
1.5	.1355 4205	.1350 8334
1.6	.1428 6261	.1424 3342
1.7	.1484 4012	.1480 4398
1.8	.1522 9605	.1519 3493
1.9	.1545 0519	.1541 7979
2.0	.1551 8295	.1548 9284

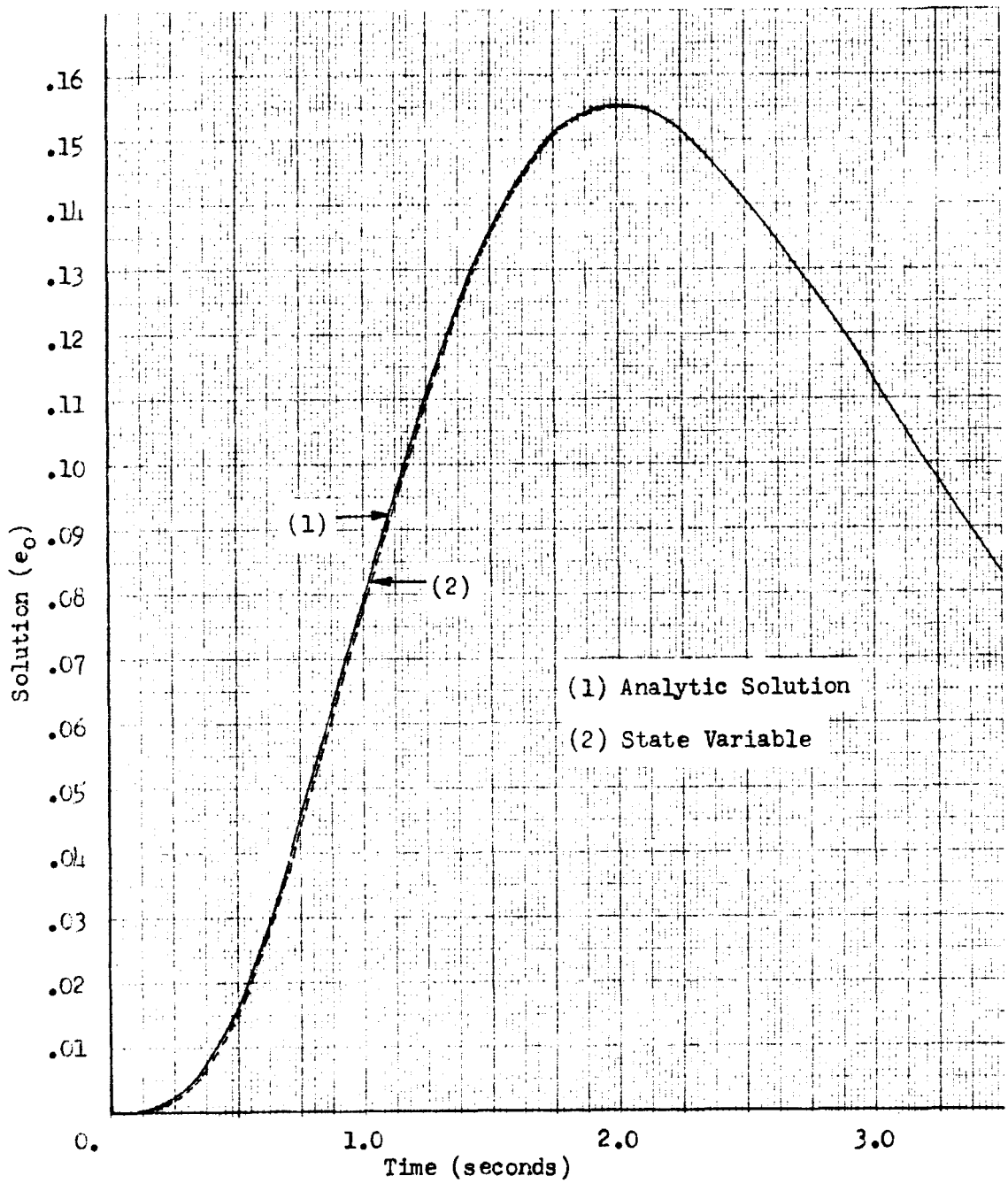

Figure 6...Solution of Third-order System Using the State Variable Method with h= 0.1 Seconds.

V. TRANSIENT RESPONSE METHODS[12,13]

This chapter deals with a class of methods that is not conventionally considered to be a means of solving a differential equation or set of equations. The two primary methods that are investigated, a zero- and a first-order method, both require a knowledge of the system's behavior in response to certain specified inputs. The zero-order method requires that the system's response to a unit step input (i.e., indicial function) be known, and the first-order method requires the system's ramp response be known. A distinct advantage of both of these methods is that a system can be digitally simulated even though a transfer function or differential equation describing the system may not be known.

The transfer function of a linear system, G(s), is related to the Laplace transform of the input, $E_i(s)$, and the Laplace transform of the output, $E_o(s)$, as

$$G(s) = \frac{E_o(s)}{E_i(s)} . \tag{1}$$

To derive the zero-order method let the input be a unit step function

$$E_i(s) = \frac{1}{s} , \tag{2}$$

so that

$$E_o(s) = \frac{G(s)}{s} = A(s) . \tag{3}$$

The inverse Laplace transform of A(s) in (3) is called the indicial function.

If the input to the system consists of a series of step-functions, one beginning every T seconds, then the Laplace transform of the input can be expressed as

$$E_i(s) = \frac{e_i(o)}{s} + [e_i(T) - e_i(o)]\,\frac{\epsilon^{-sT}}{s} + [e_i(2T) - e_i(T)]\,\frac{\epsilon^{-2sT}}{s}$$

$$+ \ldots + [e_i(kT) - e_i(kT-T)]\,\frac{\epsilon^{-kTs}}{s} + \ldots \quad . \tag{4}$$

With (4) as the input the Laplace transform of the output can be expressed as

$$E_o(s) = E_i(s)G(s) = \frac{e_i(o)}{s}\,G(s) + [e_i(T) - e_i(o)]\,\frac{\epsilon^{-sT}}{s}\,G(s)$$

$$+ \ldots + [e_i(kT) - e_i(kT-T)]\,\frac{\epsilon^{-kTs}}{s}\,G(s)$$

$$= e_i(o)A(s) + [e_i(T) - e_i(o)]A(s)\epsilon^{-sT}$$

$$+ \ldots + [e_i(kT) - e_i(kT-T)]A(s)\epsilon^{-kTs} + \ldots \quad . \tag{5}$$

In the time domain the output between KT and KT + T becomes

$$e_o(t) = e_i(o)\,A(t) + [e_i(T) - e_i(o)]A(t-T) + \ldots$$

$$+ [e_i(kT) - e_i(kT-T)]A(t-kT)$$

$$= \sum_{n=o}^{k} [e_i(nT) - e_i(nT-T)] \cdot A(t-nT) \ , \tag{6}$$

where $e_i(-T) \triangleq o$. Equation (6) is true for $kT \leq t < kT + T$, and in particular if $t = KT$

$$e_o(kT) = \sum_{n=o}^{k} [e_i(nT) - e_i(nT-T)]A(kT-nT) \ . \tag{7}$$

Equation (7) provides a logical means for simulating a system, except for the fact that large values of k often require many arithmetical operations. However, (7) can be rewritten as

$$e_o(kT) = \sum_{n=o}^{k} [A(nT) - A(nT-T)] \cdot e_i(kT-nT) \tag{8}$$

where $A(-T) = o$, which often can be simplified to give a more convenient form than (7).

The sequence of coefficients in (8) can be truncated if the coefficients approach zero with increasing values of n. If the sequence approaches a constant then

$$\lim_{nT \to \infty} [A(nT) - A(nT-T)] = C \ . \tag{9}$$

Applying the Final Value Theorem to (9) yields

$$\lim_{t \to \infty} [A(t) - A(t-T)] = \lim_{s \to o} [sA(s) - sA(s)\, \epsilon^{-sT}]$$

$$= \lim_{s \to o} [G(s) - G(s)\epsilon^{-sT}] = \lim_{s \to o} G(s)\, [1-\epsilon^{-sT}] = C \ . \tag{10}$$

Equation (10) is true if all poles of sG(s) have negative real parts;

then either G(s) has only poles with negative real parts and

$$\lim_{s \to o} G(s) = K_1 , \tag{11}$$

or G(s) has a single pole at the origin and

$$\lim_{s \to o} sG(s) = K_1 . \tag{12}$$

Using (11) and (10)

$$\lim_{t \to \infty} [A(t) - A(t-T)] = \lim_{s \to o} G(s)[1-\epsilon^{-sT}] = o, \tag{13}$$

and using (12) and (10)

$$\lim_{t \to \infty} [A(t) - A(t-T)] = \lim_{s \to o} G(s) [1-\epsilon^{-sT}] = K_1 \lim_{s \to o} \left[\frac{1-\epsilon^{-sT}}{s}\right]$$

$$= K_1 \lim_{s \to o} T \epsilon^{-sT} = K_1 T . \tag{14}$$

Equation (13) and (14) are significant in that they give the constant value to which the coefficient sequence in (8) converges. If G(s) has no poles at the origin then the sequence approaches zero and (8) can be used directly. If G(s) has a single pole at the origin then $[A(nT) - A(nT-T) - K_1T]$ approaches zero. An alternate form of (8) is then

$$e_o(kT) = \sum_{n=o}^{k} [A(nT) - A(nT-T) - K_1T] \cdot e_i(kT-nT) + K_1T \sum_{n=o}^{k} e_i(kT-nT) . \tag{15}$$

Equations (8) and (15) give a zero-order method for digitally simulating a system if the input is assumed to be constant over each update time interval T. Under the circumstances that the input is varying rapidly over the chosen interval and increased accuracy is desired over the zero-order method, a higher order method can be developed using the previous reasoning.

Assume that the input to the system consists of a series of first-order curves, or straight lines connecting equally spaced input points. Also assume that the input at t=o is zero. If $e_i(o)$ is not zero, then the input discontinuity at the origin can be accounted for by including a summation of the form of (8). A first-order curve connecting input data points can be written as

$$e_i(t) = \sum_{n=o}^{k} \left[\frac{e_i(nT+T) - e_i(nT)}{T} \; (t-nT)\, u(t-nT) \right.$$
$$\left. - \frac{e_i(nT+T) - e_i(nT)}{T} \; (t-nT-T)\, u(t-nT-T) \right] . \qquad (16)$$

Note that the first term of (16) is

$$e_i(t) = \frac{e_i(T) - e_i(o)}{T} \; tu(t) - \frac{e_i(T) - e_i(o)}{T} \; (t-T)u(t-T) , \qquad (17)$$

which is a ramp from t=o to t=T and is equal to $e_i(T)$ for $t > T$. Equation (17) is plotted in Figure 7.

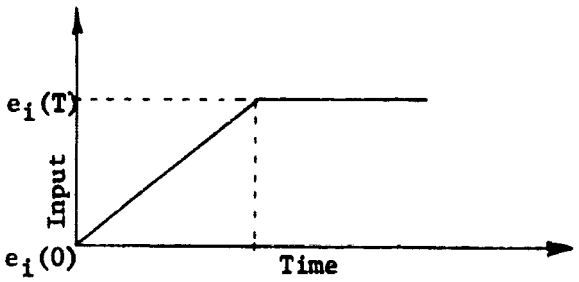

Figure 7..Typical First-order Input Curve

The mth term in (16) is of the same form as (17) and expresses the input from t=mT to t=mT+T along a straight line having slope $\frac{e_i(mT+T) - e_i(mT)}{T}$. The Laplace transform of the input is

$$E_i(s) = \frac{e_i(T) - e_i(o)}{s^2T} - \frac{e_i(T) - e_i(o)}{s^2T}\epsilon^{-sT} + \frac{e_i(2T) - e_i(T)}{s^2T}\epsilon^{-sT}$$

$$- \frac{e_i(2T) - e_i(T)}{s^2T}\epsilon^{-2sT} + \ldots + \frac{e_i(kT+T) - e_i(kT)}{s^2T}\epsilon^{-kTs}$$

$$= \frac{e_i(T) - e_i(o)}{s^2T} + \frac{e_i(2T) - 2e_i(T) + e_i(o)}{s^2T}\epsilon^{-sT}$$

$$+ \ldots + \frac{e_i(kT+T) - 2e_i(kT) + e_i(kT-T)}{s^2T}\epsilon^{-kTs} \quad . \qquad (18)$$

The Laplace transform of the output can be written as

$$E_o(s) = E_i(s)G(s) = \frac{e_i(T)-e_i(o)}{T}R(s) + \frac{e_i(2T)-2e_i(T)+e_i(o)}{T}R(s)\epsilon^{-sT}$$

$$+ \ldots + \frac{e_i(kT+T) - 2e_i(kT) + e_i(kT-T)}{T}R(s)\epsilon^{-kTs} \quad , \qquad (19)$$

where R(s) is the Laplace transform of the response of the system to a ramp input of unit slope and is given by

$$R(s) = \frac{G(s)}{s^2} \quad . \tag{20}$$

In the time domain, the output becomes

$$e_o(t) = \sum_{n=o}^{k} \left[\frac{e_i(nT+T) - 2e_i(nT) + e_i(nT-T)}{T}\right] R(t-nT) \tag{21}$$

where $e_i(o) \triangleq e_i(-T) \triangleq o$. Equation (21) is valid for $kT < t < kT+T$, thus

$$e_o(kT) = \sum_{n=o}^{k} \left[\frac{e_i(nT+T) - 2e_i(nT) + e_i(nT-T)}{T}\right] R(kT-nT). \tag{22}$$

Equation (21) is also valid for $t < kT$ since $R(t-nT) = o$ for $t < nT$.

Equation (22) can be rearranged as

$$e_o(kT) = \sum_{n=o}^{k} \left[\frac{R(nT)-2R(nT-T)+R(nT-2T)}{T}\right] \cdot e_i(kT+T-nT) \tag{23}$$

where $R(-T) = R(-2T) = o$. Equation (23) expresses the output as a linear combination of the inputs. In some cases the series of coefficients in (23) can be truncated as the coefficients approach a constant value with increasing n.

The coefficient sequence approaches a constant if

$\lim_{t\to\infty} [R(t)-2R(t-T)+R(t-2T)] = C$. By the Final Value Theorem

$$\lim_{t\to\infty} [R(t)-2R(t-T)+R(t-2T)] = \lim_{s\to o} s[R(s)-2R(s)\epsilon^{-sT}+R(s)\epsilon^{-2sT}]$$

$$= \lim_{s\to o} [A(s)-2A(s)\epsilon^{-sT}+A(s)\epsilon^{-2sT}]$$

$$= \lim_{s\to o} G(s)\left[\frac{1-2\epsilon^{-sT} + \epsilon^{-2sT}}{s}\right] = C. \tag{24}$$

As in the zero -order method, if G(s) has only roots with negative real parts, then

$$\lim_{s\to o} G(s) = K_1 \quad , \tag{25}$$

and (24) becomes

$$K_1 \lim_{s\to o}\left[\frac{1-2\epsilon^{-sT}+\epsilon^{-2sT}}{s}\right] = K_1 \lim_{s\to o}\left[2T\epsilon^{-sT}-2T\epsilon^{-2sT}\right] = o = C. \tag{26}$$

If G(s) has a single pole at the origin of the s-plane then

$$\lim_{s\to o} sG(s) = K_1 \quad , \tag{27}$$

and (24) becomes

$$K_1 \lim_{s\to o}\left[\frac{1-2\epsilon^{-sT}+\epsilon^{-2sT}}{s^2}\right] = K_1 \lim_{s\to o}\left[\frac{2T\epsilon^{-sT}-2T\epsilon^{-2sT}}{2s}\right]$$

$$= K_1 \lim_{s\to o}\left[\frac{-2T^2\epsilon^{-sT}+4T^2\epsilon^{-2sT}}{2}\right] = K_1T^2 = C \ . \tag{28}$$

Conditions for the coefficient sequence in (23) to converge to a constant are that G(s) contain at most one pole at the origin and the remaining poles have negative real parts. If G(s) has one pole at the origin, then a recursive formula analogous to (23) is

$$\epsilon_o(kT) = \sum_{n=o}^{k}\left[\frac{R(nT)-2R(nT-T)+R(nT-2T)}{T} - K_1T\right] e_i(kT+T-nT)$$

$$+ K_1T \sum_{n=o}^{k} e_i(kT+T-nT) \ . \tag{29}$$

The coefficient sequence in the first term of (29) does converge to zero for large n, thus can be truncated after an appropriate number of terms.

The methods given in (8) or (15), and (23) or (29) present an accurate and easy approach to simulating a linear system if the system's response to either a step or a ramp input is known. From these responses a series of coefficients can be computed to be used in either (8), (15), (23), or (29).

There are several sources of error present in this class of methods. One is produced by approximating the input with either a series of step-functions or a series of first-order curves connecting the input points. Another source of error is introduced by truncating the series of coefficients. If a large number of these coefficients are needed before they converge to zero, then computer round-off error becomes significant. This can be reduced by increasing the update time, thus decreasing the number of additions and multiplications required per update.

Example

The zero- and first-order transient response methods are applied to the third-order system

$$G(s) = \frac{s+8}{s^3+5s^2+12s+8}$$

$$A(t) = \mathcal{L}^{-1}\left\{A(s)\right\} = \mathcal{L}^{-1}\left\{\frac{G(s)}{s}\right\} = \mathcal{L}^{-1}\left\{\frac{s+8}{s(s^3+5s^2+12s+8)}\right\}$$

$$= 1 - 1.4\epsilon^{-t} + \epsilon^{-2t}\ (.4\cos 2t - .3\sin 2t)$$

$$R(t) = \mathcal{L}^{-1}\left\{\frac{G(s)}{s^2}\right\} = \mathcal{L}^{-1}\left\{\frac{s+8}{s^2(s^3+5s^2+12s+8)}\right\}$$

$$= t-1.375 + 1.4\epsilon^{-t} + \epsilon^{-2t}(-.025 \cos 25 + .175 \sin 2t) .$$

The coefficients for the recursions in (8) and (23) become

$$[A(nT) - A(nT-T)] = 1.4\epsilon^{-nT}(\epsilon^{T}-1) + \epsilon^{-2nT} (.4 \cos 2nT - .3 \sin 2nT)$$

$$- \epsilon^{-2(nT-T)}(.4 \cos 2(nT-T) - .3 \sin 2(nT-T)) , \text{ and}$$

$$[R(nT) - 2R(nT-T) + R(nT-2T)] / T = 1.4\,\epsilon^{-nT}\left(\frac{1-2\epsilon^{T}+\epsilon^{2T}}{T} \right)$$

$$+\frac{\epsilon^{-2nT}}{T}(-.025\cos 2nT+.175\sin 2nT)-\frac{2\epsilon^{-2(nT-T)}}{T}(-.025\cos 2(nT-T)$$

$$+.175\sin 2(nT-T))+\frac{\epsilon^{-2(nT-2T)}}{T}(-.025\cos 2(nT-2T)+.175\sin 2(nT-2T))$$

which can be evaluated for any desired T. With T = .1, the above sets of coefficients are calculated from the step and ramp responses and are plotted in Figure 8. Had the transfer function not been known, the step and ramp responses could have been obtained from shake tests or other means.

Using the calculated zero- and first-order coefficients, the system's output for an input $t\epsilon^{-1.5t}$ was computed using either (8) or (23) as appropriate. The results are listed in Table 5 and plotted in Figure 9. Although the zero-order method was introduced prior to the writing of this treatise, the first-order method is presented for the first time.

TABLE 5...Third-order System by Transient Response Methods with T= 0.1 Seconds.

Time	Analytic Solution	Zero-order Transient Response	First-order Transient Response
0.0	.0	.0	.0
0.1	.0001 6449	.0004 6434	.0001 5239
0.2	.0012 6963	.0022 6544	.0012 2195
0.3	.0040 6374	.0060 2106	.0039 6167
0.4	.0090 0563	.0120 4342	.0088 3762
0.5	.0162 5180	.0203 5288	.0160 1303
0.6	.0256 9028	.0307 2776	.0253 8223
0.7	.0370 0047	.0427 7007	.0366 2964
0.8	.0497 2117	.0559 7357	.0492 9762
0.9	.0633 1614	.0697 8577	.0628 5206
1.0	.0772 3077	.0836 5928	.0767 3925
1.1	.0909 3712	.0970 9071	.0904 3109
1.2	.1039 6613	.1096 4706	.1034 5768
1.3	.1159 2817	.1209 8093	.1154 2798
1.4	.1265 2313	.1308 3628	.1260 4014
1.5	.1355 4205	.1390 4678	.1350 8334
1.6	.1428 6261	.1455 2889	.1424 3342
1.7	.1484 4012	.1502 7143	.1480 4398
1.8	.1522 9605	.1533 2326	.1519 3494
1.9	.1545 0519	.1547 8040	.1541 7980
2.0	.1551 8295	.1547 7339	.1548 9285

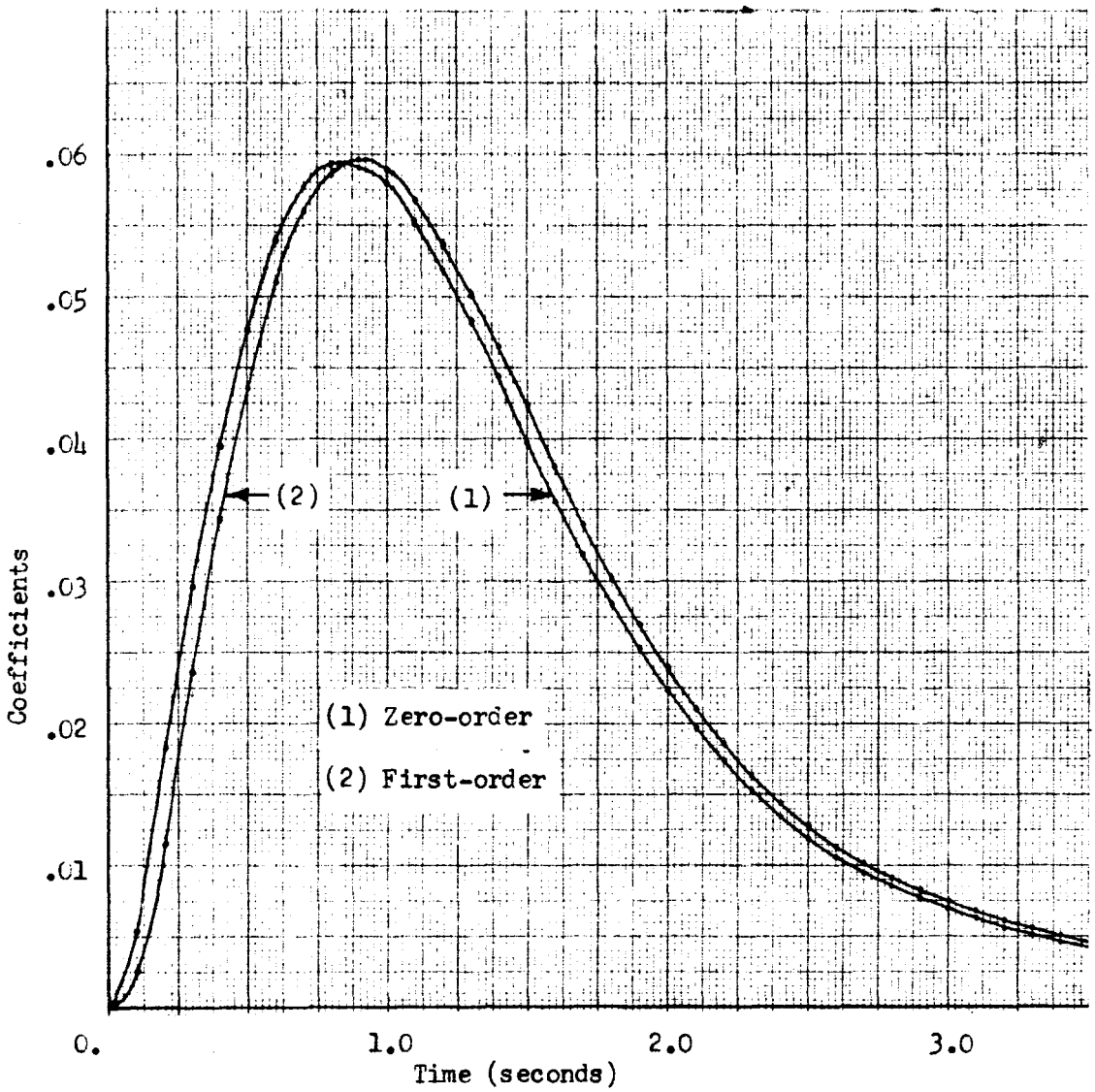

Figure 8...Transient Response Coefficients for Third-order System with T= 0.1 Seconds.

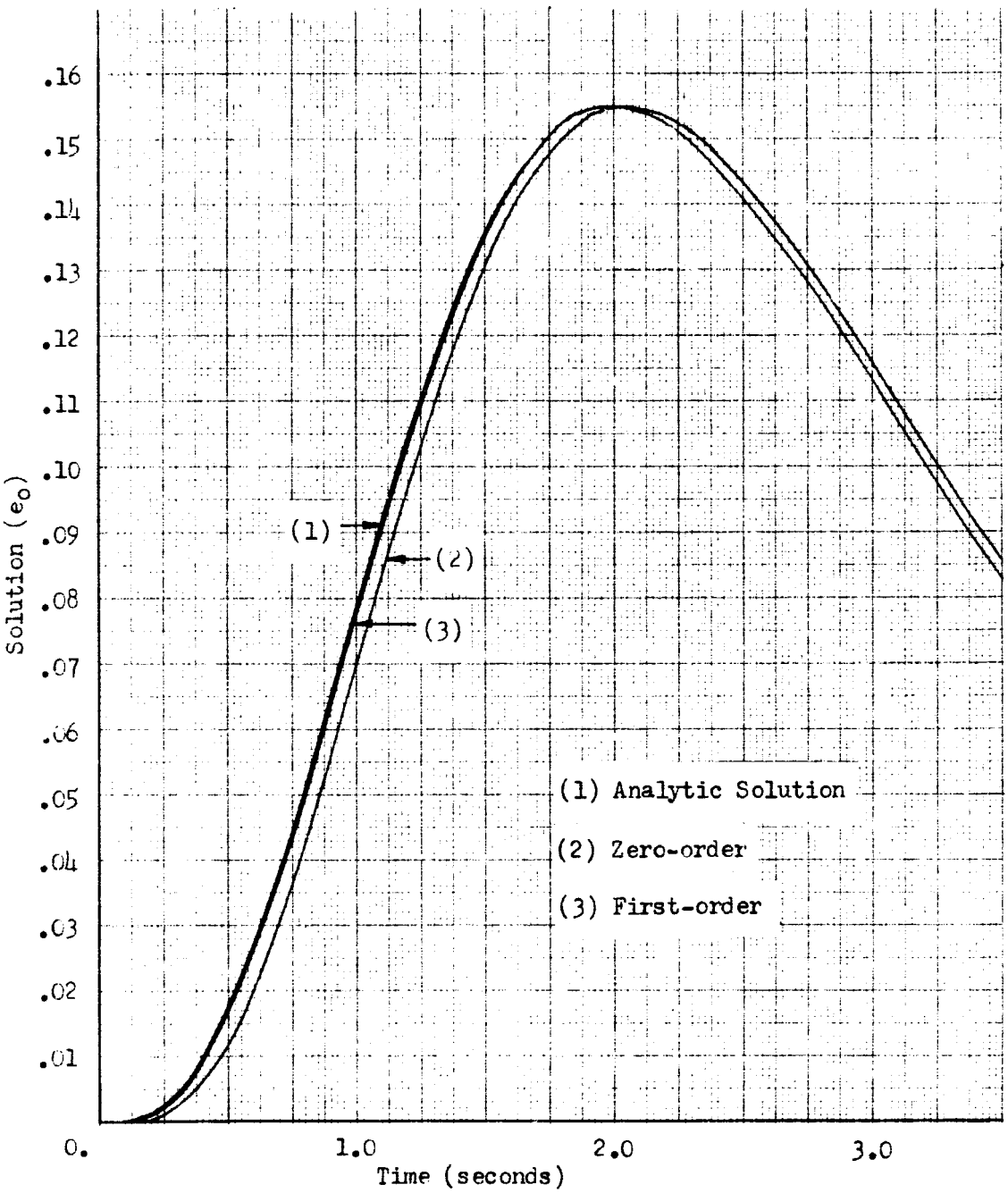

Figure 9...Solution of Third-order System Using the Transient Response Methods with T= 0.1 Seconds.

VI. CONCLUSIONS

Several methods suitable for simulating a physical system with a digital computer have been examined in detail. Each class of methods was first derived and then illustrated with one or more examples.

The two main examples considered were a third-order linear system, chosen to demonstrate technique application, and a ninth-order quasi-linear system, chosen to demonstrate accuracy and speed of computation. Results of all examples are plotted in graphic form, and results of the third-order example are also listed in tabular form. Some of the more significant results of this survey are:

(1) a fourth-order Runge-Kutta method can be expected to yield the greatest degree of accuracy;

(2) a transient response recursion can be executed very rapidly, particularly if the update time is not small with respect to natural time constants of the system;

(3) the discrete form, state variable, and transient response methods are applicable only for systems of linear differential equations with constant coefficients;

(4) error can best be controlled if a multi-point iteration formula is used, which can allow for a variable update time;

(5) the Runge-Kutta and state-variable methods are the easiest to program, since no prior computations are required;

(6) the discrete and transient response methods are the easiest to program once the initial calculations have been performed; and

(7) the transient response methods require less amount of information

about the system, prior to simulation, than any of the other methods.

The various methods are given in Appendices A through E for reference.

BIBLIOGRAPHY

1. Milne, W. E., Numerical Solution of Differential Equations, John Wiley and Sons, Inc., New York, 1953.

2. Ralston, Anthony, A First Course in Numerical Analysis, McGraw-Hill Book Company, New York, 1965.

3. Milne, W. E., Numerical Calculus, Princeton University Press, Princeton, New Jersey, 1949.

4. Conte, S. D., Elementary Numerical Analysis, McGraw-Hill Book Company, New York, 1965.

5. Rosser, J. B., A Runge-Kutta For All Seasons, Mathematics Research Center Technical Summary Report No. 698, The University of Wisconsin, Madison, Wisconsin, 1966.

6. Tustin, A., A Method of Analyzing the Behavior of Linear Systems in Terms of Time Series, Journal I.E.E., Vol. 94, Part II-A, May 1947.

7. Fryer, W. D. and Schultz, W. C., A Survey of Methods for Digital Simulation of Control Systems, Report XA-1681-F-1, Cornell Aeronautical Laboratory, Inc., Buffalo, New York, July 1964.

8. Gibson, J. E., Nonlinear Automatic Control, McGraw-Hill Book Company, New York, 1963.

9. Blakey, J. and Hutton, M., Engineering Mathematics, Blackie and Son Limited, Glasgow, 1960.

10. Monroe, A. J., Digital Processes for Sampled Data Processes, John Wiley and Sons, Inc., New York, 1962.

11. DeRusso, P. M., Roy, R. J., and Close, C. M., State Variables for Engineers, John Wiley and Sons, Inc., New York, 1965.

12. Chen, W. H., The Analysis of Linear Systems, McGraw-Hill Book Company, Inc., New York, 1963.

13. Carroll, C. C., and White, R., Several Methods for the Evaluation of Transient Response, IEEE Transaction on Education, Vol. E-9, No. 4, December 1966.

14. Lowry, J. L., et al., An Analysis of a Single Axis Platform, Technical Report Contract NAS8-20004, Auburn Research Foundation, Auburn, Alabama, December 1966.

APPENDIX A RUNGE-KUTTA FORMULAS

1. Second-Order Methods

$$\begin{cases} \alpha_2 = 1/2 \\ k_1 = hf(t_o, x_o) \\ k_2 = hf(t_o + \frac{1}{2}h, x_o + \frac{1}{2}k_1) \\ x_1 = x_o + k_2 \end{cases}$$

$$\begin{cases} \alpha_2 = 1 \\ k_1 = hf(t_o, x_o) \\ k_2 = hf(t_o + h, x_o + k_1) \\ x_1 = x_o + \frac{1}{2}(k_1 + k_2) \end{cases}$$

$$\begin{cases} \alpha_2 = 2/3 \\ k_1 = hf(t_o, x_o) \\ k_2 = hf(t_o + \frac{2}{3}h,\ x_o + \frac{2}{3}k_1) \\ x_1 = x_o + \frac{1}{4}(k_1 + 3k_2) \end{cases}$$

2. Third-Order Methods

$$\begin{cases} \alpha_2 = 1/2,\ \alpha_3 = 3/4 \\ k_1 = hf(t_o, x_o) \\ k_2 = hf(t_o + \frac{1}{2}h,\ x_o + \frac{1}{2}k_1) \\ k_3 = hf(t_o + \frac{3}{4}h,\ x_o + \frac{3}{4}k_2) \\ x_1 = x_o + \frac{1}{9}(2k_1 + 3k_3 + 4k_3) \end{cases}$$

$$\begin{cases} \alpha_2 = 1/2,\ \alpha_3 = 1 \\ k_1 = hf(t_o, x_o) \\ k_2 = hf(t_o + \frac{1}{2}h,\ x_o + \frac{1}{2}k_1) \end{cases}$$

$$\begin{cases} k_3 = hf(t_o+h,\ x_o-k_1,\ +2k_2) \\ x_1 = x_o + \frac{1}{6}(k_1+4k_2+k_3) \end{cases}$$

3. Fourth-Order Methods

$$\begin{cases} \alpha_2 = 1/2,\ \alpha_3 = 1/2,\ \alpha_4 = 1 \\ k_1 = hf(t_o, x_o) \\ k_2 = hf(t_o+\frac{1}{2}h,\ x_o+\frac{1}{2}k_1) \\ k_3 = hf(t_o+\frac{1}{2}h,\ x_o+\frac{1}{2}k_2) \\ k_4 = hf(t_o+h,\ x_o+k_3) \\ x_1 = x_o+\frac{1}{6}(k_1+2k_2+2k_3+k_4) \end{cases}$$

$$\begin{cases} \alpha_2 = 1/3,\ \alpha_3 = 2/3,\ \alpha_4 = 1 \\ k_1 = hf(t_o, x_o) \\ k_2 = hf(t_o+\frac{1}{3}h, x_o+\frac{1}{3}k_1) \\ k_3 = hf(t_o+\frac{2}{3}h, x_o-\frac{1}{3}k_1+k_2) \\ k_4 = hf(t_o+h,\ x_o+k_1-k_2+k_3) \\ x_1 = x_o+\frac{1}{8}(k_1+3k_2+3k_3+k_4) \end{cases}$$

APPENDIX B MULTI-POINT FORMULAS

1. One-point formula

$$x_1 = x_o + h\dot{x}_o \quad \Big\} \text{ Iterative Formula}$$

2. Two-point formulas

$$x_1 = x_o + \frac{h}{2}(\dot{x}_o + \dot{x}_1) \quad \Big\} \text{ Iterative Formula}$$

$$x_2 = x_o + 2h\dot{x}_1 \quad \Big\} \text{ Predictor Formula}$$

3. Three-point formulas

$$\left.\begin{aligned} x_1 &= x_o + \frac{h}{12}(5\dot{x}_o + 8\dot{x}_1 - \dot{x}_2) \\ x_2 &= x_o + \frac{h}{3}(\dot{x}_o + 4\dot{x}_1 + \dot{x}_2) \end{aligned}\right\} \text{ Iterative Formulas}$$

$$\left.\begin{aligned} x_3 &= x_o + \frac{3h}{4}(\dot{x}_o + 3\dot{x}_2) \\ x_4 &= x_o + \frac{4h}{3}(2\dot{x}_o - 4\dot{x}_1 + 5\dot{x}_2) \end{aligned}\right\} \text{ Predictor Formulas}$$

4. Four-point formulas

$$\left.\begin{aligned} x_1 &= x_o + \frac{h}{24}(9\dot{x}_o + 19\dot{x}_1 - 5\dot{x}_2 + \dot{x}_3) \\ x_2 &= x_o + \frac{h}{3}(\dot{x}_o + 4\dot{x}_1 + \dot{x}_2) \\ x_3 &= x_o + \frac{3h}{8}(\dot{x}_o + 3\dot{x}_1 + 3\dot{x}_2 + \dot{x}_3) \end{aligned}\right\} \text{ Iterative Formulas}$$

$$\left.\begin{aligned} x_4 &= x_o + \frac{h}{32}(2\dot{x}_1 - \dot{x}_2 + 2\dot{x}_3) \\ x_5 &= x_o + \frac{5h}{24}(-11\dot{x}_o + 55\dot{x}_1 - 65\dot{x}_2 + 45\dot{x}_3) \\ x_6 &= x_o + 3h(-3\dot{x}_o - 12\dot{x}_1 - 15\dot{x}_2 + 8\dot{x}_3) \end{aligned}\right\} \text{ Predictor Formulas}$$

5. Five-point formulas

$$\left.\begin{aligned} x_1 &= x_o + \frac{h}{720}(251\dot{x}_o + 646\dot{x}_1 - 264\dot{x}_2 + 106\dot{x}_3 - 19\dot{x}_4) \\ x_2 &= x_o + \frac{h}{90}(29\dot{x}_o + 124\dot{x}_1 + 24\dot{x}_2 + 4\dot{x}_3 - \dot{x}_4) \\ x_3 &= x_o + \frac{3h}{80}(9\dot{x}_o + 34\dot{x}_1 + 24\dot{x}_2 + 14\dot{x}_3 - \dot{x}_4) \end{aligned}\right\} \text{ Iterative Formulas}$$

$$x_4 = x_o + \frac{2h}{45}(7\dot{x}_o + 32\dot{x}_1 + 12\dot{x}_2 + 32\dot{x}_3 + 7\dot{x}_4) \quad \} \text{ Iterative Formula}$$

$$\left.\begin{aligned}
x_5 &= x_o + \frac{5h}{144}(19\dot{x}_o - 10\dot{x}_1 + 120\dot{x}_2 - 70\dot{x}_3 + 85\dot{x}_4) \\
x_6 &= x_o + \frac{3h}{10}(11\dot{x}_o - 44\dot{x}_1 + 96\dot{x}_2 - 84\dot{x}_3 + 41\dot{x}_4) \\
x_7 &= x_o + \frac{7h}{720}(1301\dot{x}_o - 5894\dot{x}_1 + 11256\dot{x}_2 - 9674\dot{x}_3 + 3731\dot{x}_4) \\
x_8 &= x_o + \frac{8h}{45}(206\dot{x}_o - 944\dot{x}_1 + 1716\dot{x}_2 - 1424\dot{x}_3 + 491\dot{x}_4)
\end{aligned}\right\} \text{Predictor Formulas}$$

APPENDIX C DISCRETE SUBSTITUTIONS

n	s^{-n}	
	Tustin	z-form
1	$\frac{T}{2}\left(\frac{z+1}{z-1}\right)$	$\frac{T}{2}\left(\frac{z+1}{z-1}\right)$
2	$\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^2$	$\frac{T^2}{12}\ \frac{(z^2+10z+1)}{(z-1)^2}$
3	$\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^3$	$\frac{T^3}{2}\ \frac{z(z+1)}{(z-1)^3}$
4	$\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^4$	$\frac{T^4}{6}\ \frac{z(z^2+4z+1)}{(z-1)^4}$
5	$\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^5$	$\frac{T^5}{24}\ \frac{z(z^3+11z^2+11z+1)}{(z-1)^5}$
6	$\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^6$	$\frac{T^6}{120}\ \frac{z(z^4+26z^3+66z^2+26z+1)}{(z-1)^6}$
7	$\left[\frac{T}{2}\left(\frac{z+1}{z-1}\right)\right]^7$	$\frac{T^7}{720}\ \frac{z(z^5+57z^4+302z^3+302z^2+57z+1)}{(z-1)^7}$

APPENDIX D STATE VARIABLE METHOD

K_{ij} = Residue of $G_j(s)$ at λ_i = Elements of K matrix

$$E = \begin{bmatrix} \epsilon^{\lambda_1 h} & \cdots & 0 \\ 0 & \epsilon^{\lambda_1 h} & \cdots \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \epsilon^{\lambda_n h} \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{1}{h\lambda_1^2}(\epsilon^{\lambda_1 h} - 1) + \frac{\epsilon^{\lambda_1 h}}{\lambda_1} & \cdots & 0 \\ 0 & \frac{1}{h\lambda_2^2}(\epsilon^{\lambda_2 h} - 1) + \frac{\epsilon^{\lambda_2 h}}{\lambda_2} & \cdots \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \frac{1}{h\lambda_n^2}(\epsilon^{\lambda_n h} - 1) + \frac{\epsilon^{\lambda_n h}}{\lambda_n} \end{bmatrix} K$$

$$D = \begin{bmatrix} \frac{1}{h\lambda_1^2}(\epsilon^{\lambda_1 h} - 1) - \frac{1}{\lambda_1} & \cdots & 0 \\ 0 & \frac{1}{h\lambda_2^2}(\epsilon^{\lambda_2 h} - 1) - \frac{1}{\lambda_2} & \cdots \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \frac{1}{h\lambda_n^2}(\epsilon^{\lambda_n h} - 1) - \frac{1}{\lambda_n} \end{bmatrix} K$$

$$\underline{Y}(t_o + h) = E\underline{Y}(t_o) + C\underline{V}(t_o) + D\underline{V}(t_o + h)$$

$$e_o(t_o + h) = \sum_{i=1}^{n} \underline{Y}(t_o + h)$$

APPENDIX E TRANSIENT RESPONSE METHODS

Zero-Order Method

Zero poles at the origin

$$C(nT) = A(nT) - A(nT-T)$$

$$e_o(kT) = \sum_{n=0}^{k} C(nT) \cdot e_i(kT-nT)$$

One pole at the origin

$$C(nT) = A(nT) - A(nT-T) - K_1T$$

$$e_o(kT) = \sum_{n=0}^{k} C(nT) \cdot e_i(kT-nT) + K_1T \sum_{n=0}^{k} e_i(kT-nT)$$

First-Order Method

Zero poles at the origin

$$C(nT) = [R(nT) - 2R(nT-T) + R(nT-2T)]/T$$

$$e_o(kT) = \sum_{n=0}^{k} C(nT) \cdot e_i(kT + T - nT)$$

One pole at the origin

$$C(nT) = [R(nT) - 2R(nT-T) + R(nT-2T)]/T - K_1T$$

$$e_o(kT) = \sum_{n=0}^{k} C(nT) \cdot e_i(kT + T - nT) + K_1T \sum_{n=0}^{k} e_i(kT + T - nT) .$$