Walt Lowsbery

# DYNAMICS OF REAL–TIME DIGITAL SIMULATION

by

R. M. Howe

Ann Arbor, Michigan

June, 1986

# DYNAMICS OF REAL—TIME DIGITAL SIMULATION

## CHAPTER 1

## INTRODUCTION

### 1.1 Overview of Real—Time Digital Simulation

Simulation of physical systems using digital computers is playing an ever increasing role in all aspects of today's technological society. In general the basis for simulation resides in mathematical models of the systems being simulated. In the case of continuous dynamic systems the mathematical models take the form of nonlinear ordinary or partial differential equations. The simulation of these systems and hence the simulation of the corresponding mathematical models can be accomplished by numerical integration of the diffential equations.

When simulation involves inputs and outputs in the form of real—world continuous or discrete signals, it is necessary for the simulation to run in "real—time." In other words, the simulation must be capable of running fast enough on the digital computer that the computed outputs, in response to real—time inputs, occur at the exact time these outputs would take place in the real world. A typical example of this is the real—time flight simulator shown in Figure 1.1. It consists of a digital simulation of the airplane equations of motion interfaced to the outputs from and inputs to actual flight hardware that will ultimately be used to control the airplane in flight. The system in Figure 1.1 is known as a "hardware—in—the—loop" simulation. It can be used to check out the flight control system under a wide variety of simulated flight conditions, all under the safe and repeatable conditions of the laboratory. In many cases the "hardware" is a pilot, and the resulting simulation is used to evaluate the flying qualities of the airplane along with the effectiveness of the cockpit controls and displays. Alternatively, such a man—in—the—loop simulation can be used to train pilots to fly the airplane and handle emergency conditions, again in a safe and controlled laboratory environment.

Often the inputs and outputs for a hardware—in—the—loop simulation are in continuous or analog form. In this case the continuous inputs must be converted to digital form using A to D (analog to digital) converters, a single channel of which is shown in the figure. The output of the A to D converter is then a data sequence, usually

with a fixed time interval h between samples of the analog input. In this case h also becomes the step size in the numerical integration of the airplane equations of motion, as mechanized on the digital computer. The computer outputs in the form of digital data sequences are converted to continuous form using D to A converters, a single channel of which is also illustrated in Figure 1.1. When the hardware is a human operator in a piloted flight simulator, the computer inputs, converted from analog to digital signals, represent the control displacements generated by the pilot. The computer outputs, converted from digital to analog signals, represent cockpit instrument readings, inputs to the motion system (in the case of moving–base simulators), and inputs to any visual display system.

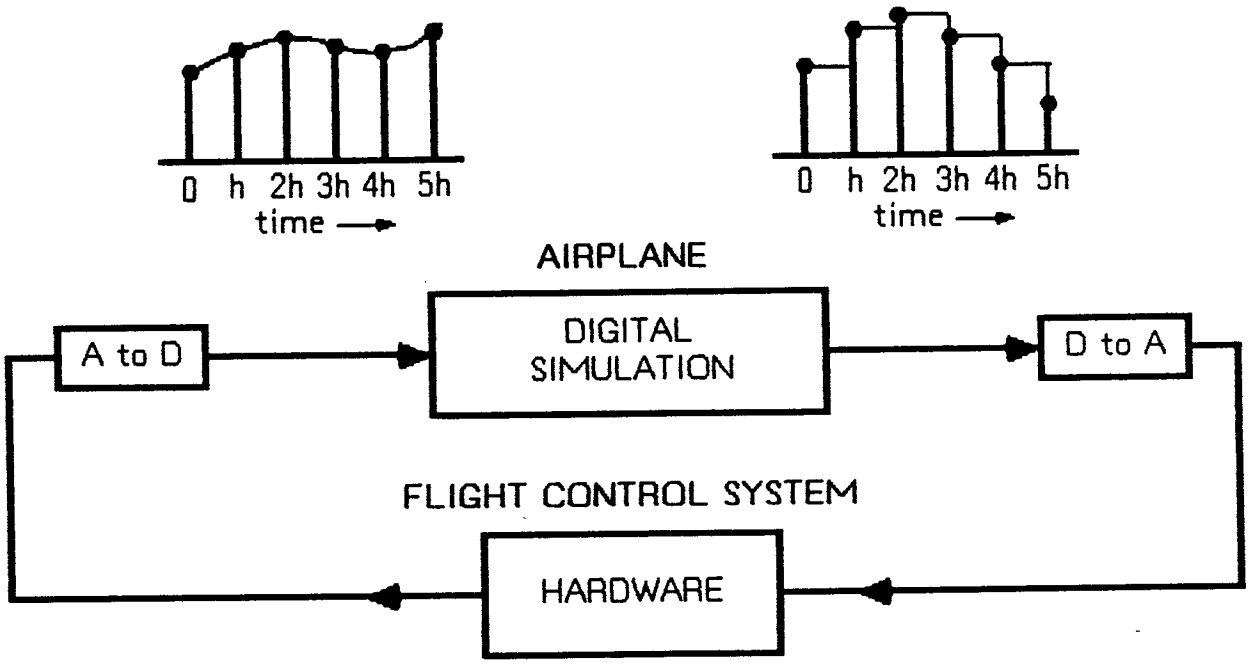



Figure 1.1. Hardware–in–the–loop flight simulation.

In addition to continuous inputs and outputs, the hardware in Figure 1.1 may have input/output signals which are digital data sequences, as in the case of a digital flight control system. Also, many of the input/output channels may be simple on–off signals, each of which represent the occurrence of a discrete event.

There are many other examples of digital simulation involving real—time inputs and outputs, including spacecraft simulators, land vehicle simulators, ship simulators, process control simulators, power plant simulators, etc. As in the flight simulator example of Figure 1.1, the hardware in the loop may be a human operator, in which case the simulator can be used for man—systems development and evaluation, or for the training of human operators.

In real—time digital simulation the numerical integration step size h is almost always fixed. This is necessary to insure that the computer will produce outputs at the end of each integration step which never occur later than real time. A fixed step size also insures compatibility with fixed sample—rate inputs and outputs. The mathematical step size h must of course be chosen large enough that the computer has sufficient available time during each step to finish the necessary calculations. On the other hand, the step size h cannot be made too large, or the errors due to the approximate numerical integration algorithms will cause the dynamic fidelity of the simulation to be unacceptable. Often these dynamic errors (known as integration truncation errors) can be reduced for a given step size h by choosing the proper integration algorithm, by using specialized integration methods, by running fast subsystems with sub—multiple step sizes, and by the use of other special techniques which will be covered in this text.

In order to assess the comparitive accuracy of different integration methods and to develop specialized integration algorithms, it is important to have a general method of quantitative evaluation of the dynamic errors in digital simulation. Unfortunately, there is no such method which provides generally useful results when the simulation involves nonlinear differential equations. Fortunately, however, many if not most dynamic systems can be approximated as quasi—linear, time—invarient systems by considering linearized perturbation equations with respect to steady—state or reference solutions over limited segments of time. For example, this is precisely the technique used to linearize the highly nonlinear equations of motion of an aircraft in order to develop airframe transfer functions in the design of flight control systems.

When the differential equations are linear and the integration step size is fixed, we can use the method of z transforms to obtain direct measures of simulation accuracy for specific integration algorithms.* These direct measures include the frac—

* See, for example, E.G. Gilbert, "Dynamic Error Analysis of Digital and Combined Digital—Analog Systems," *Simulation*, Vol. 6, No. 4, April, 1966, pp 241—257.

tional error in the characteristic roots of the digital simulation, and the fractional error in transfer function gain and the transfer function phase error of the digital simulation. These latter two error measures are particularly useful in determining the acceptability of digital system errors in hardware—in—the—loop simulations, since they can be translated directly into gain and phase—margin errors for the closed—loop simulation.

In this chapter we will introduce the conventional integration algorithms which are usually considered in real—time digital simulation. Each of the algorithms will be examined in regard to its suitability for use with real—time inputs. This will be followed by the development of the dynamic error measures described above. A brief introduction to z transforms and their application to digital simulation is presented in Chapter 2. In Chapter 3 we develop both exact and approximate asymptotic formulas for dynamic errors in applying conventional integration algorithms to the simulation of quasi—linear systems. This will allow us to compare the different integration methods and to establish accuracy/speed tradeoffs. Sampling, extrapolation and interpolation, and their application to real—time interface systems is covered in Chapters 4 and 5. Chapter 6 describes and analyzes special methods for simulating linear subsystems, including the state—transition method and Tustin's method. In Chapter 7 we introduce a modified form of Euler integration which is especially efficient in the simulation of linear subsystems but which also can be used in the simulation of some nonlinear systems. Chapter 8 describes a method for efficient handling of discontinuous nonlinear functions. Finally, in Chapter 9 we describe some special techniques for improving the performance of real—time digital simulations, including the use of multiple integration frame rates in simulating fast subsystems and the mixing of integration methods.

It should again be noted that all of the integration methods considered here use a fixed integration step size because of the real—time requirement. In non real—time simulation this is not necessary, and integration algorithms are often used which vary the step size during the solution based on some error criterion. Even in this case, however, the step size may remain relatively fixed over significant lengths of time, so that the dynamic analysis techniques developed here can be applied. It may also be true that the use of very efficient fixed—step integration algorithms may actually speed up many non real—time simulations in comparison with variable—step methods. Any

significant speed improvement will for the same accuracy, of course, save computational cost in the non real—time environment.

## 1.2 Introduction to Integration Algorithms

In this section we will introduce and interpret a number of integration methods which are candidates for use in real—time digital simulation. We will assume that the continuous dynamic system to be simulated is represented by the following nonlinear differential equation:

$$\frac{dX}{dt} = F[X, U(t)] \tag{1.1}$$

Here X is the state vector and U(t) is the input vector. For a qth—order dynamic system with m inputs the scalar state equations can be written as

$$\frac{dx_k}{dt} = f_k[x_1, x_2, \ldots, x_q, u_1(t), u_2(t), \ldots, u_m(t)], \quad k = 1, 2, \ldots, q \tag{1.2}$$

When solving Eq. (1.1) or (1.2) using numerical integration with a fixed integration step size h, we will be computing the vector solution X only at the discrete, equally—spaced times 0, h, 2h, 3h, .... . At the time $t = nh$ we denote the solution X(nh) by $X_n$. Similarly, we denote U(nh) by $U_n$. To illustrate the geometric interpretation of numerical integration we consider the simple first—order equation

$$\frac{dx}{dt} = f(t) \tag{1.3}$$

Given the value of x at $t = nh$, i.e., given $x_n$, we can express $x_{n+1}$, i.e., the value of x at $t = (n+1)h$, by the equation

$$x_{n+1} = x_n + \int_{nh}^{(n+1)h} f(t)\,dt \tag{1.4}$$

The integral in Eq. (1.4) is simply the area under the f(t) versus t curve between $t = nh$ and $t = (n+1)h$, as shown graphically in Figure 1.2. However, in our digital simulation we do not have a representation of the continuous time function f(t). Instead, we have only the representation of f at the discrete times (n-1)h, nh, ... , i.e., $f_{n-1}$, $f_n$, ... . Thus we can only approximate f(t) and hence the area under f(t) using the available values of f at the discrete times. The method used to approximate the area defines the type of numerical integration which will be used.

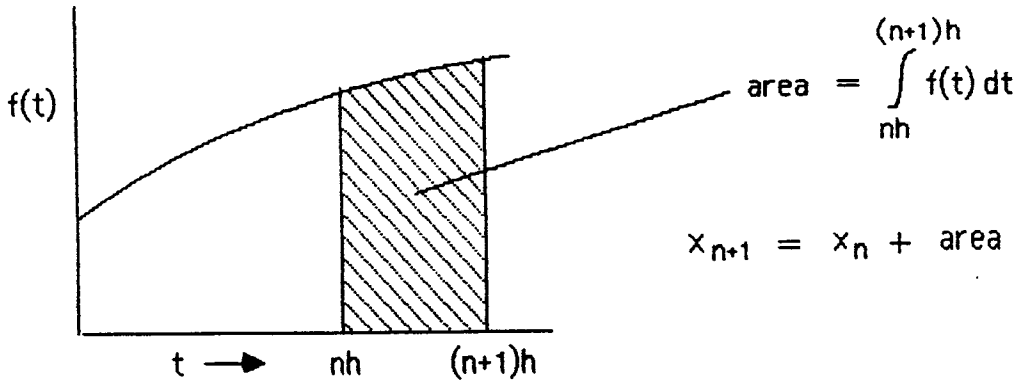



Figure 1.2. Graphical representation of one integration step.

The simplest integration algorithm is Euler or rectangular integration. In this method the area is approximated by the rectangle of height $f_n$ and width h, as shown in Figure 1.3. Thus the area is equal to $h f_n$ and the integration formula is given by

$$x_{n+1} = x_n + h f_n \tag{1.5}$$

It can be shown, as we shall see in the next chapter, that the dynamic errors produced by Euler integration vary as the first power of the integration step size h.

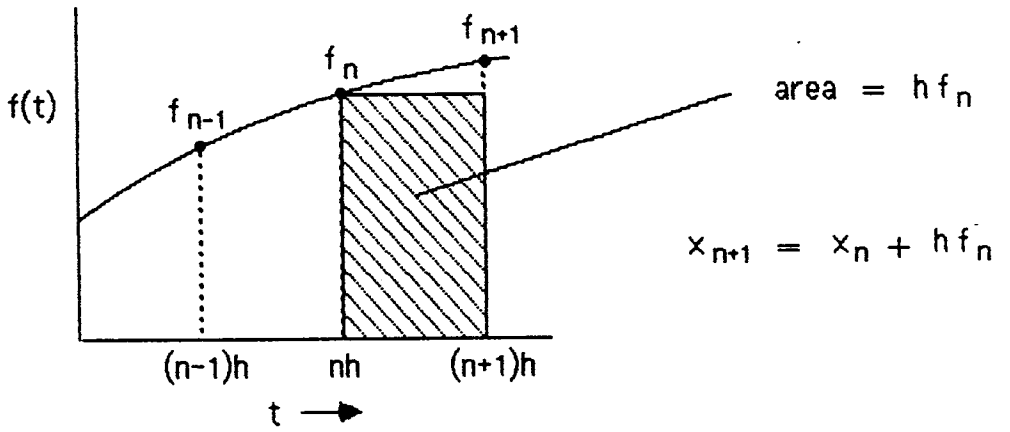



Figure 1.3. Euler integration for the equation $\dot{x} = f$.

A more accurate numerical integration formula is obtained by considering the the area under the trapezoid shown in Figure 1.4. If f in the state equation (1.3) is an explicit function of time t, then $f_{n+1}$ is known and the formula for trapezoidal in—

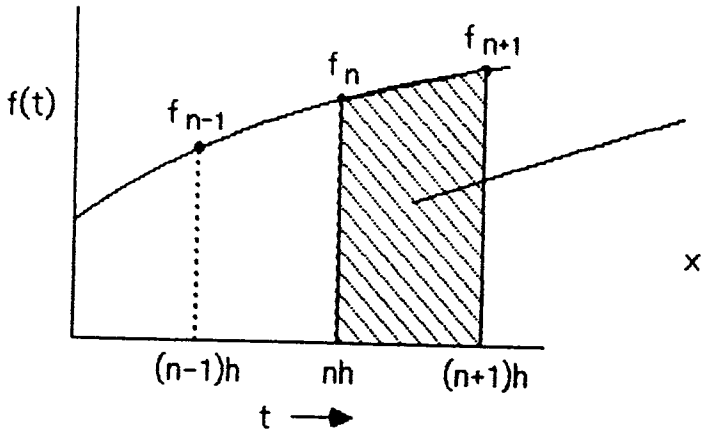



Figure 1.4. Trapezoidal integration for the equation $\dot{x} = f$.

tegration becomes

$$x_{n+1} = x_n + h\frac{f_n + f_{n+1}}{2} \tag{1.6}$$

On the other hand, if f in Eq. (1.3) is a function of the state x and an input u(t), i.e., if the first–order equation is given by

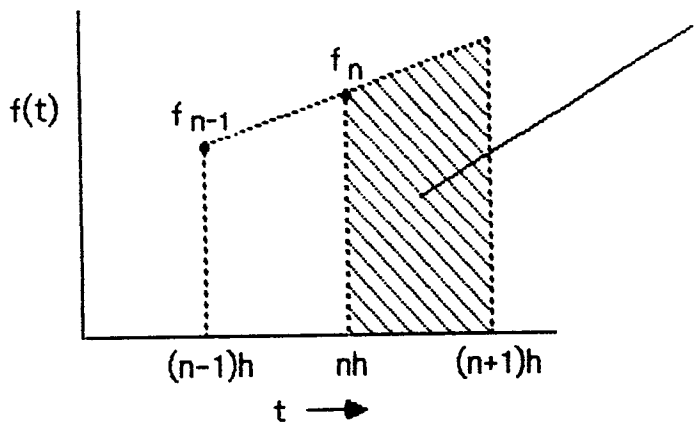
$$\frac{dx}{dt} = f[x,u(t)] \tag{1.7}$$

then $f_{n+1}$ in Eq. (1.6) will involve $x_{n+1}$. If f is a linear function of x, the resulting equation can be solved for $x_{n+1}$, as required. In this case the technique is known as Tustin's method for simulating linear systems. However, if f is a nonlinear function of x, the resulting implicit equation for $x_{n+1}$ can in general only be solved by some iterative technique. Because of uncertainties in the number of required iterations, especially in dealing with high order systems, implicit trapezoidal integration is generally deemed unsuitable for real–time simulation. It is used often in non real–time simulation, particularly in the case of "stiff" systems.*

The trapezoidal integration formula of Eq. (1.6) also forms the basis for both Runge–Kutta and Adams Moulton integration algorithms of second order, as we shall see later in this section. In Chapter 3 we will show that the dynamic errors produced by trapezoidal integration vary as the second power of the step size h.

The difficulties associated with implicit trapezoidal integration when solving nonlinear state equations can be avoided by basing the incremental area under the f versus t curve on a linear extrapolation using $f_n$ and $f_{n-1}$, as shown in Figure 1.5. The resulting integration algorithm is known as the second–order Adams–Bashforth

* Stiff systems are systems with eigenvalues ranging over many orders of magnitude.




$$\text{area} = h f_n + \frac{h}{2}(f_n - f_{n-1})$$

$$= \frac{h}{2}(3 f_n - f_{n-1})$$

$$x_{n+1} = x_n + \frac{h}{2}(3 f_n - f_{n-1})$$

Figure 1.5. AB–2 integration for the equation $\dot{x} = f$.

method, hereafter referred to as AB–2. The integration formula is the following:

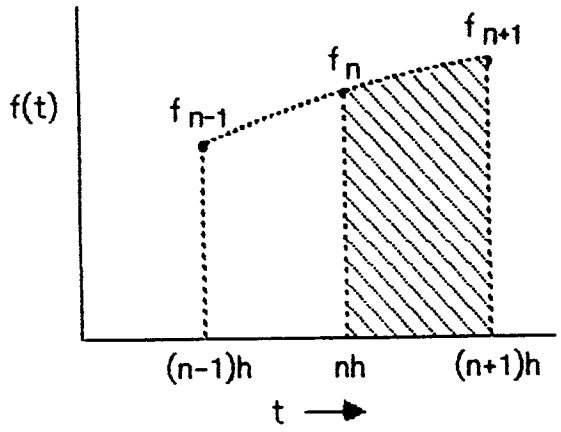
$$x_{n+1} = x_n + \frac{h}{2}(3 f_n - f_{n-1}) \tag{1.8}$$

As a second–order method, AB–2 integration produces errors proportional to $h^2$.

Implicit third order integration is illustrated in Figure 1.6. Here a parabola is passed through the three points $f_{n+1}$, $f_n$ and $f_{n-1}$ to define the incremental area from $t = nh$ to $t = (n+1)h$. This leads to the following formula:

$$x_{n+1} = x_n + \frac{h}{12}(5 f_{n+1} + 8 f_n - f_{n-1}) \tag{1.9}$$

As in the case of trapezoidal integration, this third–order implicit method requires iterations to solve for $x_{n+1}$ when f is a function of the state x. However, a parabola




$$\text{area} = \frac{h}{12}(5 f_{n+1} + 8 f_n - f_{n-1})$$

$$x_{n+1} = x_n + \frac{h}{12}(5 f_{n+1} + 8 f_n - f_{n-1})$$

Figure 1.6. Implicit third–order integration for the equation $\dot{x} = f$.

can be passed through the three points $f_n$, $f_{n-1}$ and $f_{n-2}$ to allow quadratic extrapolation over the interval from $t = nh$ to $t = (n+1)h$. The resulting incremental area leads to the third–order explicit algorithm known as AB–3 integration. From Figure 1.7 we see that the integration formula is

$$x_{n+1} = x_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2}) \tag{1.10}$$

The dynamic errors produced by both AB–3 integration and implicit third–order integration vary as $h^3$, as we shall see in Chapter 3.

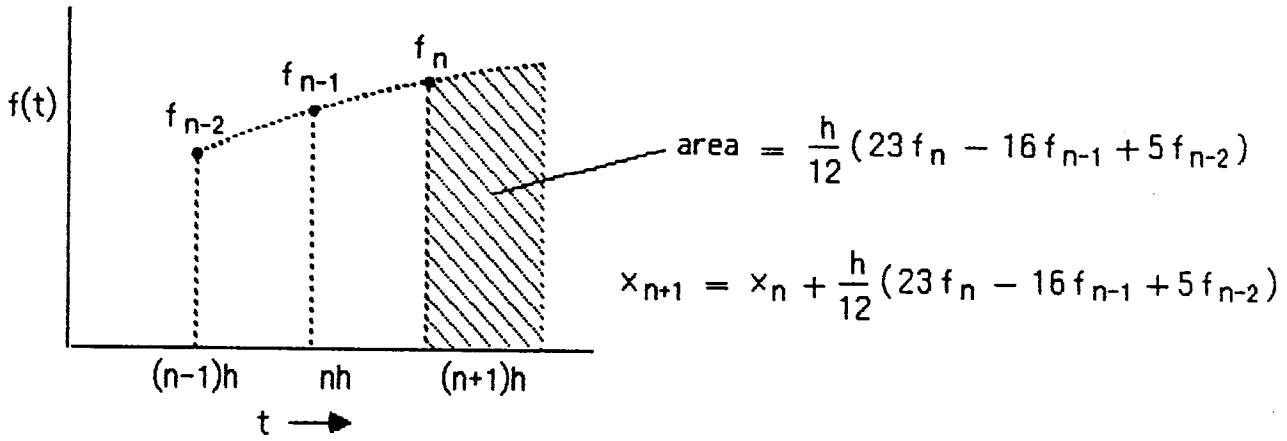



Figure 1.7. AB–3 integration for the equation $\dot{x} = f$.

A fourth–order implicit algorithm can be generated by passing a cubic through the four points $f_{n+1}$, $f_n$, $f_{n-1}$ and $f_{n-2}$ to define the incremental area from $t = nh$ to $t = (n+1)h$. This leads to the formula

$$x_{n+1} = x_n + \frac{h}{24}(9f_{n-1} + 19f_n - 5f_{n-1} + f_{n-2}) \tag{1.11}$$

The explicit AB–4 integration formula is obtained by passing a cubic through the four points $f_n$, $f_{n-1}$, $f_{n-2}$ and $f_{n-3}$. The resulting cubic extrapolation defines the incremental area from $t = nh$ to $t = (n+1)h$, which leads to the AB–4 formula

$$x_{n+1} = x_n + \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) \tag{1.12}$$

As expected, the dynamic errors produced by the fourth–order implicit integration and AB–4 integration vary as $h^4$.

The third–order formulas in Eq. (1.9) and (1.10) form the basis for the third–

order Adams–Moulton predictor–corrector integration algorithm, hereafter referred to as AM–3 integration. In the same way, Eqs. (1.11) and (1.12) form the basis for the AM–4 predictor–corrector algorithm.

In considering the Adams–Mouton algorithms, we will write the formulas for the solution of the vector state equation as given in (1.1). In the case of AM–2 integration, a predicted value, $\hat{X}_{n+1}$, for the next state is computed using the AB–2 formula of Eq. (1.8). $\hat{X}_{n+1}$ is then used to calculate the derivative $F_{n+1}$ in the trapezoidal formula of Eq. (1.6). Thus the AM–2 algorithm is given by

$$\hat{X}_{n+1} = X_n + \frac{h}{2}[3F(X_n, U_n) - F(X_{n-1}, U_{n-1})] \tag{1.13}$$

$$X_{n+1} = X_n + \frac{h}{2}[F(X_n, U_n) + F(\hat{X}_{n+1}, U_{n+1})] \tag{1.14}$$

The AM algorithms are called predictor–corrector methods, where the calculation of $\hat{X}_{n+1}$ using the AB algorithm is viewed as the predictor, and the calculation of $X_{n+1}$ using the implicit algorithm is viewed as the corrector.

In the same way, Eqs. (1.9) and (1.10) are used to mechanize the AM–3 algorithm. Thus we have

$$\hat{X}_{n+1} = X_n + \frac{h}{12}[23F(X_n, U_n) - 16F(X_{n-1}, U_{n-1} + 5F(X_{n-2}, U_{n-2})] \tag{1.15}$$

$$X_{n+1} = X_n + \frac{h}{12}[5F(\hat{X}_{n+1}, U_{n+1}) + 8F(X_n, U_n) - F(X_{n-1}, U_{n-1})] \tag{1.16}$$

for the AM–3 predictor–corrector formulas. From Eqs. (1.11) and (1.12) the following formulas are obtained for the AM–4 predictor–corrector algorithm:

$$\hat{X}_{n+1} = X_n + \frac{h}{24}[55F(X_n, U_n) - 59F(X_{n-1}, U_{n-1} + 37F(X_{n-2}, U_{n-2} - 9F(X_{n-3}, U_{n-3})] \tag{1.17}$$

$$X_{n+1} = X_n + \frac{h}{24}[9F(\hat{X}_{n+1}, U_{n+1}) + 19F(X_n, U_n) - 5F(X_{n-1}, U_{n-1} + F(X_{n-2}, U_{n-2})] \tag{1.18}$$

The Adams–Moulton predictor–corrector algorithms combine the accuracy of the implicit methods with the explicit nature of the AB predictor methods. We now turn to the consideration of some Runge–Kutta integration algorithms. We consider first a second–order Runge–Kutta (RK–2) method also known as Heun's method. As in the case of AM–2, we first compute an estimate, $\hat{X}_{n+1}$, for the next state. But in

RK−2 integration this estimate is computed using Euler integration. $\hat{X}_{n+1}$ is then used to calculate the derivative $F_{n+1}$ in the trapezoidal formula of Eq. (1.6). The algorithm is therefore given by the following two equations:

$$\hat{X}_{n+1} = X_n + hF(X_n, U_n) \tag{1.19}$$

$$X_{n+1} = X_n + \frac{h}{2}[F(X_n, U_n) + F(\hat{X}_{n+1}, U_{n+1})] \tag{1.20}$$

An interesting variation of RK−2 integration uses Euler integration to compute $\hat{X}_{n+1/2}$, an estimate of the state halfway through the next integration step. $\hat{X}_{n+1/2}$ is then used to compute the derivative $F_{n+1/2}$ at the half−integer step which, in turn, is used to compute $X_{n+1}$. The two equations for the algorithm are

$$\hat{X}_{n+1/2} = X_n + \frac{h}{2}F(X_n, U_n) \tag{1.21}$$

$$X_{n+1} = X_n + hF(\hat{X}_{n+1/2}, U_{n+1/2}) \tag{1.22}$$

The integration method represented by Eqs. (1.21) and (1.22) is better suited to real−time digital simulation than conventional RK−2 integration, as we shall see later in this section. For this reason we will designate it as "real−time RK−2."

Next we consider the RK−3 algorithm given by the following equations:

$$X_{n+1} = X_n + h\left[\frac{\frac{F_n + \hat{F}_{n+2/3}}{2} + \hat{F}_{n+2/3}}{2}\right] \tag{1.23}$$

or

$$X_{n+1} = X_n + \frac{h}{4}(F_n + 3\hat{F}_{n+2/3}) \tag{1.24}$$

where

$$F_n = F(X_n, U_n) \tag{1.25}$$

$$\hat{F}_{n+1/3} = F(X_n + \frac{h}{3}F_n, U_{n+1/3}) \tag{1.26}$$

$$\hat{F}_{n+2/3} = F(X_n + \frac{2h}{3}\hat{F}_{n+1/3}, U_{n+2/3}) \tag{1.27}$$

Here Euler integration is used to compute $\hat{X}_{n+1/3} = X_n + (h/3)F_n$ which, when substituted into Eq. (1.26), is used to calculate the derivative estimate at the one−third frame, $\hat{F}_{n+1/3}$. This derivative is then used with Euler integration to compute $\hat{X}_{n+2/3} = X_n + (2h/3)\hat{F}_{n+1/3}$ which, when substituted into Eq. (1.27), yields the derivative estimate at the two−thirds frame, $\hat{F}_{n+2/3}$. In Eq. (1.23) $F_n$ and $\hat{F}_{n+2/3}$ are then averaged to produce an improved estimate of $\hat{F}_{n+1/3}$, which in turn is averaged

with $\hat{F}_{n+2/3}$ to produce the final average derivative used in the integration formula of Eq. (1.24).

The equations used for the RK−4 integration algorithm are the following:

$$X_{n+1} = X_n + h\left[\frac{\frac{F_n + \hat{F}_{n+1/2}}{2} + \frac{\hat{F}_{n+1/2} + \hat{\hat{F}}_{n+1/2}}{2} + \frac{\hat{\hat{F}}_{n+1/2} + \hat{F}_{n+1}}{2}}{3}\right] \tag{1.28}$$

or

$$X_{n+1} = X_n + \frac{h}{6}(F_n + 2\hat{F}_{n+1/2} + 2\hat{\hat{F}}_{n+1/2} + \hat{F}_{n+1}) \tag{1.29}$$

where

$$F_n = F(X_n, U_n) \tag{1.30}$$

$$\hat{F}_{n+1/2} = F(X_n + \frac{h}{2}F_n, U_{n+1/2}) \tag{1.31}$$

$$\hat{\hat{F}}_{n+1/2} = F(X_n + \frac{h}{2}\hat{F}_{n+1/2}, U_{n+1/2}) \tag{1.32}$$

$$\hat{F}_{n+1} = F(X_n + h\hat{\hat{F}}_{n+1/2}, U_{n+1}) \tag{1.33}$$

Here the two half−frame derivative estimates, $\hat{F}_{n+1/2}$ and $\hat{\hat{F}}_{n+1/2}$, and the one−frame estimate, $\hat{F}_{n+1}$, are combined as shown in Eq. (1.28) to produce the grand average used in the integration formula in Eq. (1.29). The derivative estimates are calculated using Eqs. (1.30) through (1.33), which in turn involve three intermediate Euler integration steps. Although RK−4 is frequently used as the integration method in the numerical solution of differential equations, it is not well suited to real−time simulation, as we shall see next. Also, it should be noted that RK−2, 3 and 4 exhibit dynamic errors which vary, respectively, as the square, cube and fourth power of the integration step size h.

We now turn to considering the compatibilty of the various integration methods described in this chapter with real−time inputs. In the case of Euler integration, as well as the AB predictor algorithms, the formulas for computing the next state $X_{n+1}$ involve only current and past derivatives. When the derivative depends on an external input, as in Eq. (1.1) or Eq. (1.7), this means that only the current input $U_n$ and past inputs are required to implement the computation of the simulation output one frame later. This is illustrated at the top of Figure 1.8, which shows that for Euler, AB−2, AB−3, and AB−4 integration, the real−time input $U_n$ occurs at $t = nh$, and the state $X_n$, having been computed in the previous frame, is available as a real−time output at $t = nh$. Then, h seconds later, the computation of $X_{n+1}$ has been com−pleted and the real−time input $U_{n+1}$ is available to initiate the next computational frame. Clearly the inputs are never required before they occur in real time.

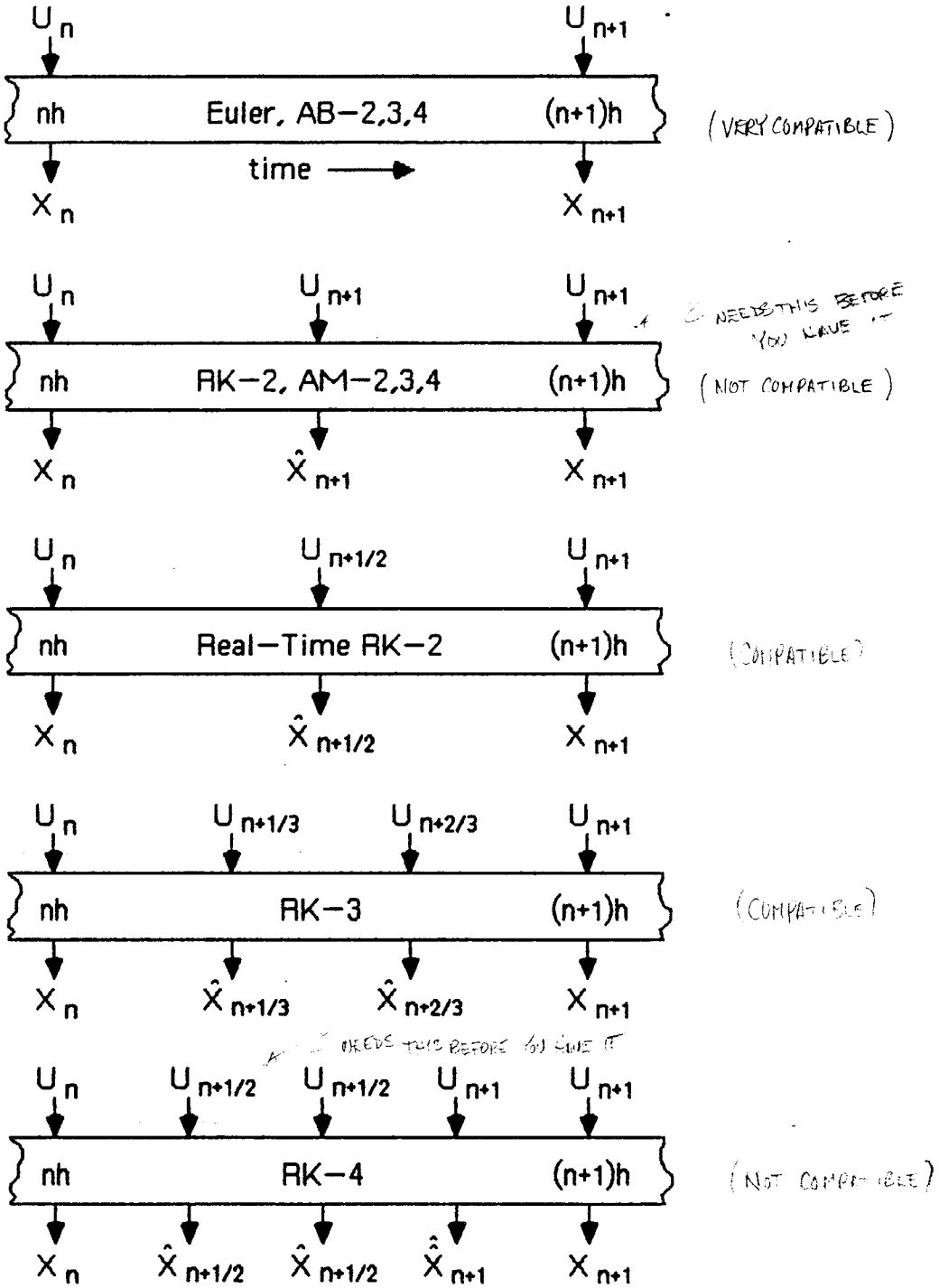



Figure 1.8. Input/output timing for integration algorithms.

On the other hand, RK−2 integration as defined in Eqs. (1.19) and (1.20), and the AM predictor−corrector algorithms, as defined in Eqs. (1.13) through (1.18), all require the input $U_{n+1}$ to implement the second and final evaluation of $X_{n+1}$. Thus $U_{n+1}$ is required before it is available in real time, as can be seen in the second dia−

gram from the top of Figure 1.8. Thus RK–2 and the AM algorithms are not compatible with real–time inputs. Only by estimating $U_{n+1}$ ahead in time based on $U_n$ and past values with extrapolation formulas can we use these algorithms correctly in a real–time simulation. However, the problem is eliminated in the real–time version of RK–2. From Eq. (1.22) we see that $U_{n+1/2}$, not $U_{n+1}$, is required as the input for the second half of the algorithm. From the middle diagram in Figure 1.8 it is apparent that $U_{n+1/2}$ is indeed available in real time, as required halfway through the integration frame.

Reference to Figure 1.8 shows that RK–3, as defined in Eqs. (1.24) through (1.27), is also compatible with real–time inputs. Thus $U_n$ is required and avaiable at the start of the first pass through the state equations, $U_{n+1/3}$ at the start of the second pass, and $U_{n+2/3}$ at the start of the third and final pass.

Finally, reference to Figure 1.8 and Eqs. (1.29) through (1.33) shows that RK–4 is not compatible with real time inputs. This is because $U_{n+1/2}$ is required at the start of the second of four passes through the state equations, a quarter frame before it becomes available. Similarly, $U_{n+1}$ is required at the start of the fourth pass through the equations, a quarter frame before it becomes available.

We conclude that of all the integration methods described thus far, only Euler, real–time RK–2, RK–3, and the AB predictor algorithms are compatible with real–time inputs.

## 1.3 Dynamic Error Measures

The final topic considered in this introductory chapter is the development of error measures used to compare different integration methods in the simulation of dynamic systems. As noted earlier, we will assume that we can linearize the non–linear state equations about some reference or equilibrium solution. Then the relationship between any scalar input–output pair can be represented in terms of the transfer function H(s) given by

$$H(s) = \frac{N(s)}{(s-\lambda_1)(s-\lambda_2)\cdots(s-\lambda_n)} \tag{1.34}$$

Here we have assumed a linearized system of order n with characteristic roots $\lambda_1$, $\lambda_2, \ldots, \lambda_n$. H(s) in Eq. (1.34) can be represented in terms of a partial–fraction expansion. Thus

$$H(s) = \frac{A_1}{(s-\lambda_1)} + \frac{A_2}{(s-\lambda_2)} + \cdots + \frac{A_n}{(s-\lambda_n)} \tag{1.35}$$

where

$$A_k = \lim_{s \to \lambda_k} (s - \lambda_k) H(s) \tag{1.36}$$

For the case where $\lambda_k$ and $\lambda_{k+1}$ represent a complex–conjugate pair the two resulting first–order transfer functions with complex conjugate $A_k$ and $A_{k+1}$ can be combined into a second–order transfer function. Thus the nth–order linearized system can be considered to be the sum of first and second–order linear subsystems. If we can determine the dynamic errors in simulating each of these subsystems, then by superposition we can determine the overall dynamic error in solving the quasi–linear version of the entire system.

We shall use two types of error measure in our analysis. The first is $e_\lambda$, the fractional error in characteristic root, defined as

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \tag{1.37}$$

Here $\lambda^*$ is the equivalent characteristic root in the digital solution, whereas $\lambda$ is the exact continuous–system root. For the case where $\lambda$ is complex, corresponding to an underdamped second–order subsystem with damping ratio $\zeta$ and damped frequency $\omega_d$, we define the damping–ratio error $e_\zeta$ and the fractional error in frequency $e_\omega$ with the following formulas:

$$e_\zeta = \zeta^* - \zeta, \qquad e_\omega = \frac{\omega_d^*}{\omega_d} - 1 \tag{1.38}$$

Another useful error measure in the case of complex roots is $e_{|\lambda|}$, the fractional magnitude of the root error. Thus

$$e_{|\lambda|} = \frac{|\lambda^* - \lambda|}{|\lambda|} \tag{1.39}$$

The second type of error measure is the fractional error in the transfer function for sinusoidal inputs. This can be determined from the z transform $H^*(z)$ of the digital system represented by the computer simulation. For sinusoidal input data sequences the digital–system transfer function is simply $H^*(e^{j\omega h})$, where $\omega$ is the input frequency, as we will show in Chapter 2 on z transforms. Then the fractional error in transfer function is given by

$$\frac{H^*(e^{j\omega h})}{H(j\omega)} - 1 = e_M + j e_A \tag{1.40}$$

Here $e_M$ and $e_A$ represent, respectively, the real and imaginary parts of the fractional error in transfer function. For error magnitudes small compared with unity $e_M$ and $e_A$ have a particularly significant meaning, which can be understood by writing $H^*$ and $H$ in polar form. Thus we let $H^* = |H^*| e^{jN^*}$ and $H = |H| e^{jN}$, where $|H^*|$ and $N^*$

are the gain and phase of the digital system, and |H| and N are the gain and phase of the continuous system, respectively. After substitution into Eq. (1.40), we obtain

$$\frac{H^*}{H} - 1 = \frac{|H^*|}{|H|} e^{j(N^*-N)} - 1 \cong \frac{|H^*|}{|H|} - 1 + j(N^*-N), \quad H^* \cong H \qquad (1.41)$$

Here we have used a first-order power series approximation for $e^{j(N^*-N)}$ and have approximated $(|H^*|/|H|)j(N^*-N)$ with $j(N^*-N)$, in both cases on the assumption that $H^* \cong H$ if the simulation is of reasonable accuracy. Comparison of Eq. (1.40) with Eq. (1.41) shows that

$$\left[\frac{H^*}{H} - 1\right]_{real} = e_M \cong \frac{|H^*|}{|H|} - 1 = \text{fractional error in gain} \qquad (1.42)$$

$$\left[\frac{H^*}{H} - 1\right]_{imag} = e_A \cong (N^*-N) = \text{phase error} \qquad (1.43)$$

We conclude that the real part, $e_M$, of the fractional error in transfer function corresponds approximately to the fractional error in digital transfer function gain, and the imaginary part, $e_A$, corresponds approximately to the phase error of the digital transfer function.

Another error measure for the digital transfer function is $e_H$, the fractional error in the transfer function magnitude. Thus

$$e_H = \frac{|H^*-H|}{|H|} = \sqrt{e_M^2 + e_A^2} \qquad (1.44)$$

It can be shown that the fractional peak error in sinusoidal output of the digital system is equal to $e_H$. By fractional peak error we mean the peak error each cycle divided by the peak output amplitude.

Errors in characteristic roots, as reflected by Eqs. (1.37), (1.38) or (1.39), will determine how well the transients generated in the digital simulation will match the exact solution for the continuous system. As noted earlier, the fractional error in transfer function gain and the transfer function phase error, as reflected by Eqs. (1.42) and (1.43), can be directly related to the errors in gain and phase margin produced by the digital computer in a hardware-in-the-loop simulation. In particular, any hardware-in-the-loop simulation will have a crossover frequency, i.e., the frequency $\omega_0$ at which the open loop system has unity gain. We recall that the phase margin is the amount by which the open-loop phase shift exceeds -180 degrees at the crossover frequency $\omega_0$. The phase margin must be positive for the closed-loop system to be stable, and the size of the phase margin is a direct measure of the relative stability of the closed-loop system. For the hardware-in-the-loop simulation

to exhibit relative stability which closely matches that of the system being simulated, the phase error in the digital—system transfer function at the frequency $\omega_0$ must be sufficiently small. Since $\omega_0$ is defined as the frequency for unity open—loop gain, any gain error in the digital—system transfer function will affect the crossover frequency itself. Thus for the hardware—in—the—loop simulation to exhibit a crossover frequency which closely matches that of the system being simulated, the gain error in the digital—system transfer function at the frequency $\omega_0$ must also be sufficiently small. Since the dominant frequency in the closed loop transient will be approximately equal to $\omega_0$, low gain error at that frequency in the digital simulation will insure an accurate representation of the closed—loop transient frequency, just as low phase error at $\omega_0$ insures an accurate representation of the damping for closed—loop transients.

For the above reasons we conclude that transfer function errors in real—time digital simulation are generally more important than characteristic root errors, although in Chapter 3 we will develope formulas for both types of error measures using the method of z transforms.

CHAPTER 2

# Z TRANSFORMS APPLIED TO DIGITAL SIMULATION

## 2.1 Introduction

The method of z transforms was originally developed for analysis of sampled–data control systems, i.e., dynamic systems utilizing a mixture of continuous and discrete time signals. This is just the case, for example, when a digital computer is used to control a continuous process. The z–transform method is equally useful in analyzing the performance of all–digital systems, and in particular in the determination of dynamic errors in digital simulation of dynamic systems. The dynamic errors will depend on the type of numerical integration algorithm and the integration step size. The method of z transforms permits us to examine these errors analytically and hence choose acceptable integration algorithms and step sizes. The method is restricted to linear systems with a fixed step size or sample period. We have already noted in Chapter 1 that nonlinear systems can often be linearized using perturbation equations, and that real–time digital simulation invariably requires the use of fixed integration step sizes. Thus the z transform requirements can be met.

## 2.2 Definition of the Z Transform

We will assume that the digital signals involved in digital simulation are data sequences, with the individual numbers in each data sequence considered to be equally spaced in time, as noted above. Thus the digital signal or data sequence representing a continuous time function f(t) will be denoted as the data sequence $\{f_n\}$, where the individual data–sequence number $f_n = f(nh)$, $n = 0, 1, 2, \ldots$ . Here h is the time interval between successive numbers in the data sequence. In a sampled–data system h is the sample interval. In a digital simulation h is the integration step size.

The z transform of the data sequence $\{f_n\}$ is by definition

$$Z\{f_n\} = F^*(z) = \sum_{n=0}^{\infty} f_n z^{-n} = f_0 + f_1 z^{-1} + f_2 z^{-2} + \cdots \qquad (2.1)$$

where z is a complex variable. We note the similarity of the z–transform definition to the Laplace transform F(s) of a continuous time time function f(t).* Thus

* For a discussion of both z and Laplace transforms, see Wilfred Kaplan, *Operational Methods for Linear Systems*, Addison Wesley Publishing Company, Inc., 1962.

$$F(s) = \int_0^\infty f(t)\, e^{-st}\, dt \tag{2.2}$$

The integral of the continuous time function in the Laplace transform is replaced by the summation of the discrete time series in the z transform.

For data sequences derived from exponential time functions the z transform series can be written in closed form. Consider the exponential time function

$$f(t) = e^{\sigma t} \tag{2.3}$$

and the equivalent data sequence

$$\{f_n\} = \{e^{\sigma nh}\} = \{(e^{\sigma h})^n\} = \{A^n\} \tag{2.4}$$

where

$$A = e^{\sigma h} \tag{2.5}$$

From Eq. (2.1) we see that

$$F^*(z) = \sum_{n=0}^{\infty} A^n z^{-n} = (1 - Az^{-1})^{-1} \tag{2.6}$$

Expansion of $(1 - Az^{-1})^{-1}$ using the binomial theorem easily proves the validity of Eq. (2.6). Thus we have shown that

$$Z\{A^n\} = (1 - Az^{-1})^{-1} = \frac{z}{z - A} \tag{2.7}$$

It also follows that if the z transform of a data sequence contains a term $z/(z-A)$, the corresponding data sequence is the exponential sequence, $\{e^{\sigma nh}\}$, i.e., equivalent to equally spaced samples from a continuous time function $e^{\sigma t}$ with a sample interval h. From Eq. (2.5)

$$\sigma = \frac{1}{h} \ln A \tag{2.8}$$

It is useful to consider the case of a complex exponential function

$$f(t) = e^{\sigma t}\, e^{j\omega t} = e^{(\sigma + j\omega)t} \tag{2.9}$$

We recall that together with the conjugate function $e^{(\sigma - j\omega)t}$ this can be used to represent a sinusoidal time function with frequency $\omega$ and exponentially varying amplitude $e^{\sigma t}$. Here the equivalent data sequence is

$$\{f_n\} = \{(e^{\sigma h} e^{j\omega h})^n\} = \{A^n\} \tag{2.10}$$

where A is now complex and given by

$$A = e^{\sigma h} e^{j\omega h} \tag{2.11}$$

But the complex number A can be written as

$$A = A_r + jA_i = |A| e^{j \tan^{-1}(A_i/A_r)} \tag{2.12}$$

where

$$|A| = \sqrt{A_r^2 + A_i^2} \tag{2.13}$$

Equating the right sides of Eqs. (2.11) and (2.12), we have

$$\sigma = \frac{1}{h} \ln |A| \quad , \qquad \omega = \frac{1}{h} \tan^{-1} \frac{A_i}{A_r} \tag{2.14}$$

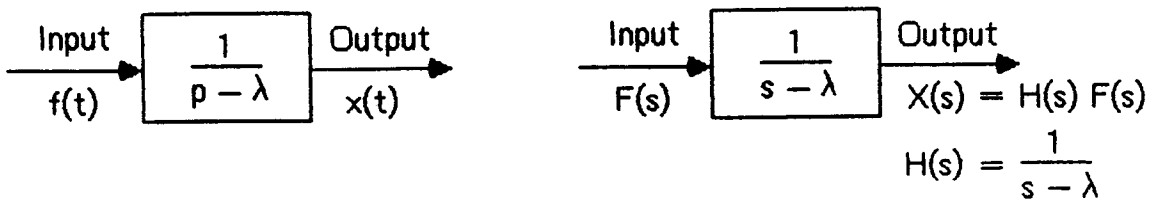
Thus a z transform given by $z/(z-A)$ where A is complex corresponds to a data sequence equivalent to equally spaced samples from a sinusoidal time function of frequency $\omega$ and amplitude $e^{\sigma t}$, with $\omega$ and $\sigma$ related to A by the formulas in Eq. (2.14). We note from both Eq. (2.8) for A positive real and Eq. (2.14) for A complex that for $|A| < 1$ the data sequence decreases exponentially, and for $|A| > 1$ the data sequence increases exponentially with time. For $A = 1$ the data sequence is always unity, i.e., $f_n = 1$ for all n. For $A = -1$ the data sequence alternates, i.e., $f_0 = +1$, $f_1 = -1$, $f_2 = +1$, $f_3 = -1$, etc., which represents samples from a unit amplitude sinusoid with frequency equal to one-half the sample frequency, as predicted by Eq. (2.14). Thus $\omega = \pi/h$ for $A = -1$ and hence the sinusoidal period $= 2\pi/\omega = 2h$.

## 2.3 Simple Example of a Linear Digital System

We have seen in the previous section how the z transform can be used to represent a data sequence. In this section we will see how the z transform can be used to represent a digital system. As a simple example, consider the continuous first-order system shown in Figure 2.1. Assume that we wish to simulate this system digitally using Euler integration.

$$\dot{x} = \lambda x + f(t) \tag{2.15}$$

In Chapter 1 we have seen that the numerical solution of Eq. (2.15) with Euler integration leads to the following difference equation:




a) Time—domain representation b) s—domain representation

Figure 2.1. First—order continuous system

$$x_{n+1} = x_n + h(\lambda x_n + f_n) \tag{2.16}$$

This is the difference equation that is mechanized by the digital simulation, starting at $t = 0$ (i.e., $n = 0$) with $x_0$ as the initial condition. The equation is now iterated for $n = 1, 2, 3, \ldots$, to produce the outputs $x_1, x_2, x_3, \ldots$, in response to the inputs $f_0, f_1, f_2, f_3, \ldots$. Thus the digital computer produces an output data sequence $\{x_{n+1}\}$ given from Eq. (2.16) by

$$\{x_{n+1}\} = (1 + \lambda h)\{x_n\} + h\{f_n\} \tag{2.17}$$

where $\{f_n\}$ is the input data sequence.

We now take the z transform of the data sequences appearing on both sides of Eq. (2.17). Consider first the z transform of $\{x_{n+1}\}$. From Eq. (2.1) this is given by

$$\begin{aligned} Z\{x_{n+1}\} &= x_1 + x_2 z^{-1} + x_3 z^{-2} + \cdots \\ &= z(x_0 + x_1 z^{-1} + x_2 z^{-2} + \cdots) - z x_0 \end{aligned}$$

or

$$Z\{x_{n+1}\} = zX^*(z) - z x_0 \tag{2.18}$$

Note the similarity of Eq. (2.18) to the formula for the Laplace transform of the derivative of a function. Thus

$$L\{df/dt\} = sF(s) - f_0 \tag{2.19}$$

Using Eq. (2.18), we can now take the z transform of Eq. (2.17) and obtain

$$zX^*(z) - z x_0 = (1 + \lambda h)X^*(z) + hF^*(z) \tag{2.20}$$

Solving for $X^*(z)$, we have

$$X^*(z) = \frac{z x_0}{z - 1 - \lambda h} + \frac{h}{z - 1 - \lambda h} F^*(z) \tag{2.21}$$

The first term on the right side of Eq. (2.21) depends only on the initial condition $x_0$ and is equivalent to the transient (complementary) solution in the case of a continuous system. It has exactly the form of Eq. (2.7) and therefore represents in the time domain an exponential data sequence $x_0\{(e^{\sigma h})^n\} = x_0\{A^n\}$, where $A = 1 + \lambda h$. From Eq. (2.8) we see that

$$\sigma = \frac{1}{h} \ln(1 + \lambda h) \tag{2.22}$$

We note that $\ln(1+x) = x - x^2/2 + x^3/3 - x^4/4 + \cdots$. Expanding the ln function in Eq. (2.22), we have

$$\sigma = \frac{1}{h}\left(\lambda h - \frac{1}{2}\lambda^2 h^2 + \cdots\right)$$

or

$$\lambda^* = \sigma \cong \lambda - \frac{1}{2}\lambda^2 h \quad , \quad |\lambda h| << 1 \tag{2.23}$$

For the continuous linear system the transient solution is given by $x_0 e^{\lambda t}$, so that a "perfect" digital solution would be a data sequence $x_0\{(e^{\lambda h})^n\}$, compared with our actual solution, $x_0\{(e^{\sigma h})^n\}$. Just as $\lambda$ is the characteristic root of the continuous system, so is $\sigma$ the equivalent characteristic root of the digital system. From now on we will designate the characteristic root of the digital system by $\lambda^*$, where $\lambda^* = \sigma$ in our example here. Replacing $\sigma$ with $\lambda^*$ in Eq. (2.23), subtracting $\lambda$ and dividing by $\lambda$, we obtain the following formula for $e_\lambda$, the fractional error in the characteristic root of our digital system.

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{1}{2}\lambda h \quad , \quad |\lambda h| << 1 \tag{2.24}$$

The above asymptotic formula is valid only for $h << 1/|\lambda|$, but this will of necessity be true if we are to have an accurate simulation. As expected for Euler integration, the root error varies as the first power of the integration step size, h. Thus the root error will decrease by a factor of two when we halve the step size.

We next consider the second term on the right side of Eq. (2.21), i.e., the term involving the input z transform, $F^*(z)$. The coefficient of $F^*(z)$ in this term is defined as the z transform, $H^*(z)$, of the digital system. Thus

$$H^*(z) = \frac{h}{z - 1 - \lambda h} \tag{2.25}$$

and when $x_0 = 0$,

$$X^*(z) = H^*(z)\, F^*(z) \tag{2.26}$$

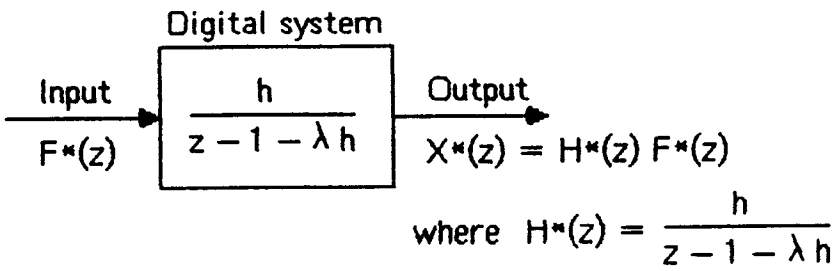



Figure 2.2. Digital system representing simulation of a first–order system using Euler integration.

Thus the system z transform, $H^*(z)$, times the input z transform, $F^*(z)$, yields the output z transform, as shown in Figure 2.2. The analogy with the Laplace transform representation of the continuous system, as shown in Figure 2.1a, is very evident.

The system z transform in Eq. (2.25) can be related to the corresponding data sequence in the time domain by dividing the denominator into the numerator, as shown below.

$$
\begin{array}{r}
hz^{-1} + h(1+\lambda h)z^{-2} + h(1+\lambda h)^2 z^{-3} + \cdots \\
z - 1 - \lambda h \,\overline{)\, h} \\
\underline{h - h(1+\lambda h)z^{-1}} \\
h(1+\lambda h)z^{-1} \\
\underline{h(1+\lambda h)z^{-1} - h(1+\lambda h)^2 z^{-2}} \\
h(1+\lambda h)^2 z^{-2} \\
\underline{h(1+\lambda h)^2 z^{-2} - h(1+\lambda h)^3 z^{-3}} \\
h(1+\lambda h)^3 z^{-3}
\end{array}
$$

Thus

$$H^*(z) = hz^{-1} + h(1+\lambda h)z^{-2} + h(1+\lambda h)^2 z^{-3} + \cdots \qquad (2.27)$$

From the z–transform definition in Eq. (2.1) we see that the data sequence $\{w_n\}$ for which $H^*(z)$ is the z transform is given by $w_0, w_1, w_2, \ldots$, where

$$w_0 = 0, \quad w_1 = h, \quad w_2 = h(1+\lambda h), \quad w_3 = h(1+\lambda h)^2, \quad w_n = h(1+\lambda h)^{n-1} \qquad (2.28)$$

I.e., when the z transform is expressed as a series in powers of $z^{-1}$, the coefficient of $z^{-n}$ is $w_n$, the data sequence value at time nh.

We next calculate the response of a digital system with z–transform $H^*(z)$ to a unit data point input defined by $f_0 = 1$, $f_1 = f_2 = f_n = 0$, $n \neq 0$. Then from the

z−transform definition in Eq. (2.1), $F^*(z) = 1$ and from Eq. (2.26), $X^*(z) = H^*(z)$. Thus we see that the z transform of the unit data point response of a digital system is the system z transform, $H^*(z)$. This provides an alternative method for determining the z transform of a digital system, namely, take the z transform of the system response sequence to a unit data point input. Earlier, in Eq. (2.25) we obtained the z transform of the system by dividing the response z transform, $X^*(z)$, by the input z transform, $F^*(z)$.

The unit data point input for a digital system is analagous to a unit impulse input $\delta(t)$ for a continuous system. The response data sequence, $\{w_n\}$, for the unit data point input is analagous to the weighting function, W(t), for the continuous system, which represents the response to the unit impulse input. Just as the z transform of the unit data point response sequence, $\{w_n\}$, is $H^*(z)$, the system z transform, so is the Laplace transform of the unit impulse response, W(t), equal to H(s), the Laplace transform (transfer function) of the continuous system.

It is useful to write Eq. (2.26) with the z transforms expressed in series form. Thus

$$\begin{aligned} X^*(z) &= x_0 + x_1 z^{-1} + x_2 z^{-2} + \cdots \\ &= (w_0 + w_1 z^{-1} + w_2 z^{-2} + \cdots)(f_0 + f_1 z^{-1} + f_2 z^{-2} + \cdots) \\ &= w_0 f_0 + (w_0 f_1 + w_1 f_0) z^{-1} + (w_0 f_2 + w_1 f_1 + w_2 f_0) z^{-2} + \cdots \\ &\quad + (w_0 f_n + w_1 f_{n-1} + \cdots + w_{n-1} f_1 + w_n f_0) z^{-n} + \cdots \end{aligned} \tag{2.29}$$

Equating coefficients of like powers of z on both sides of Eq. (2.29), we obtain

$$x_0 = w_0 f_0, \quad x_1 = w_0 f_1 + w_1 f_0, \cdots, \quad x_n = \sum_{k=0}^{n} w_k f_{n-k} \tag{2.30}$$

Since $f_{n-k} = 0$ for $k > 0$ (the input data point $f_n = 0$ for $n < 0$), the upper limit in in the summation in Eq. (2.30) can be changed from n to $\infty$. Thus the digital system response data point $x_n$ at time nh is given by

$$x_n = \sum_{k=0}^{\infty} w_k f_{n-k} \tag{2.31}$$

where $\{w_n\}$ is the system unit data point response sequence and $\{f_n\}$ is the input data sequence. The similarity to the superposition integral for the response x(t) of a continuous linear time−invarient system to an input f(t) is striking. Thus

$$f(t) = \int_0^{\infty} W(\tau)\, f(t-\tau)\, d\tau \tag{2.32}$$

where W(t) is the unit impulse function for the system.

Finally, we consider a sinusoidal data sequence input as represented by samples from $f(t) = A e^{j\omega t}$. Thus we let $f_n = A e^{j\omega nh} = A(e^{j\omega h})^n$. From Eq. (3.17) the response $x_n$ is given by

$$x_n = \sum_{k=0}^{\infty} w_k A(e^{j\omega h})^{n-k} = \left[\sum_{k=0}^{\infty} w_k (e^{j\omega h})^{-k}\right] A(e^{j\omega h})^n \tag{2.33}$$

Reference to Eq. (2.1) shows that

$$\left[\sum_{k=0}^{\infty} w_k (e^{j\omega h})^{-k}\right] = H^*(e^{j\omega h}) \tag{2.34}$$

Thus

$$x_n = H^*(e^{j\omega h}) f_n \tag{2.35}$$

where $f_n$ is a sinusoidal data sequence of frequency $\omega$. Eq. (2.35) shows that the response data sequence is also a sinusoid with the same frequency $\omega$ and a complex amplitude that is simply $H^*(e^{j\omega h})$ times the input amplitude A. Thus $H^*(e^{j\omega h})$ (i.e., the system z transform with z replaced by $e^{j\omega h}$) is the digital system transfer function for sinusoidal inputs. The magnitude and phase angle of the complex number $H^*(e^{j\omega h})$ represent the amplitude ratio and phase shift of the output data sequence with respect to the input sinusoidal data sequence.

Again, the analogy to continuous linear time-invarient systems is striking, i.e.,

$$x(t) = H(j\omega) f(t) \quad \text{for} \quad f(t) = A e^{j\omega t} \tag{2.36}$$

where x(t) is the response, $H(j\omega)$ is the transfer function for sinusoidal inputs, and $f(t) = A e^{j\omega t}$ is the sinusoidal input to the continuous system.

As a specific example of a digital transfer function for sinusoidal inputs, consider the simulation of the first-order linear system in Figure 2.1 using Euler integration. The z transform of the resulting digital system is given by Eq. (2.25). From Eq. (2.35) we see that the digital transfer function for sinusoidal inputs is obtained by replacing z with $e^{j\omega h}$ in $H^*(z)$. Thus

$$H^*(e^{j\omega h}) = \frac{h}{e^{j\omega h} - 1 - \lambda h} = \frac{h}{\cos\omega h - 1 - \lambda h + j \sin\omega h} \tag{2.37}$$

The continuous-system transfer function for sinusoidal inputs is obtained from Figure 2.1b by replacing s in H(s) by $j\omega$. Thus

$$H(j\omega) = \frac{1}{j\omega - \lambda} \tag{2.38}$$

For a given frequency $\omega$ and step size h we can now calculate the fractional error in the sinusoidal transfer function of the digital system. From Eqs. (2.37) and (2.38) this is given by

$$\frac{H^*(e^{j\omega h})}{H(j\omega)} - 1 = \frac{h(j\omega - \lambda)}{\cos\omega h - 1 - \lambda h + j\sin\omega h} - 1 \qquad (2.39)$$

In Chapter 1, Eq. (1.41), we showed that the real and imaginary parts of $H^*/H - 1$ represent, approximately, the fractional gain error and the phase error of the digital system transfer function. This in turn gives us a very meaningful measure of the dynamic accuracy of the digital simulation. In Chapter 3 we will show how simplified asymptotic formulas can be derived for these gain and phase errors, and in particular that the errors vary as the first power of the integration step size h in the case of Euler integration.

## 2.4 Stability of Digital Systems

We obtained the z transform of the digital system representing simulation by Euler integration of the first order system in Figure 2.1 by taking the z transform of the difference equation (2.16) and solving for $H^*(z) = X^*(z)/F^*(z)$. The same procedure can be used to obtain the z transform of any linear digital system described by one or more difference equations, leading to $H^*(z)$ of the following form:

$$H^*(z) = \frac{N^*(z)}{D^*(z)} = \frac{N^*(z)}{(z-A_1)(z-A_2)\cdots(z-A_n)} \qquad (2.40)$$

Here $N^*(z)$ and $D^*(z)$ are polynomials in z. $A_1, A_2, \ldots, A_n$ are the roots of $D^*(z)$, i.e., the poles of $H^*(z)$, and in general can be real or complex. $H^*(z)$ in Eq. (2.40) can be represented in terms of a partial fraction expansion, just as H(s) for the continuous linear system was represented in terms of a partial fraction expansion in Eq. (1.35). Thus we can write

$$H^*(z) = \frac{C_1}{(z-A_1)} + \frac{C_2}{(z-A_2)} + \cdots + \frac{C_n}{(z-A_n)} \qquad (2.41)$$

We conclude that the z transform for a linear digital system of any order can be expressed as the sum of z transforms of the form $z/(z-A)$. We have already seen that such z transforms correspond to exponential data sequences $\{A^n\}$, where for $|A| > 1$ the sequence diverges exponentially, and for $|A| < 1$ the sequences converges exponentially. In this case the exponential data sequences are terms in the unit data point

response sequence. Thus the overall unit data point response sequence is just the sum of the exponential data sequences corresponding to each term on the right side of Eq. (2.41). It is clear that if any of these exponential data sequences diverge, the unit data point response will diverge and the digital system is unstable. For all the exponential terms in the unit data point response to converge, the pole A associated with each term must have a magnitude less that unity, i.e., $|A| < 1$. We thus form the conclusion that a digital system with z transform poles given by $A_k$ will be stable if $|A_k| < 1$ for all k. In the complex z plane, $|A_k| < 1$ corresponds to the region inside the unit circle defined by $|z| = 1$, as shown in Figure 2.3. Z transform poles within this region all correspond to exponentially converging data sequences. Poles outside the unit circle in the z plane correspond to exponentially diverging data sequences. Poles on the unit circle correspond to data sequences of constant amplitude, except that repeated poles on the unit circle represent diverging data sequences.

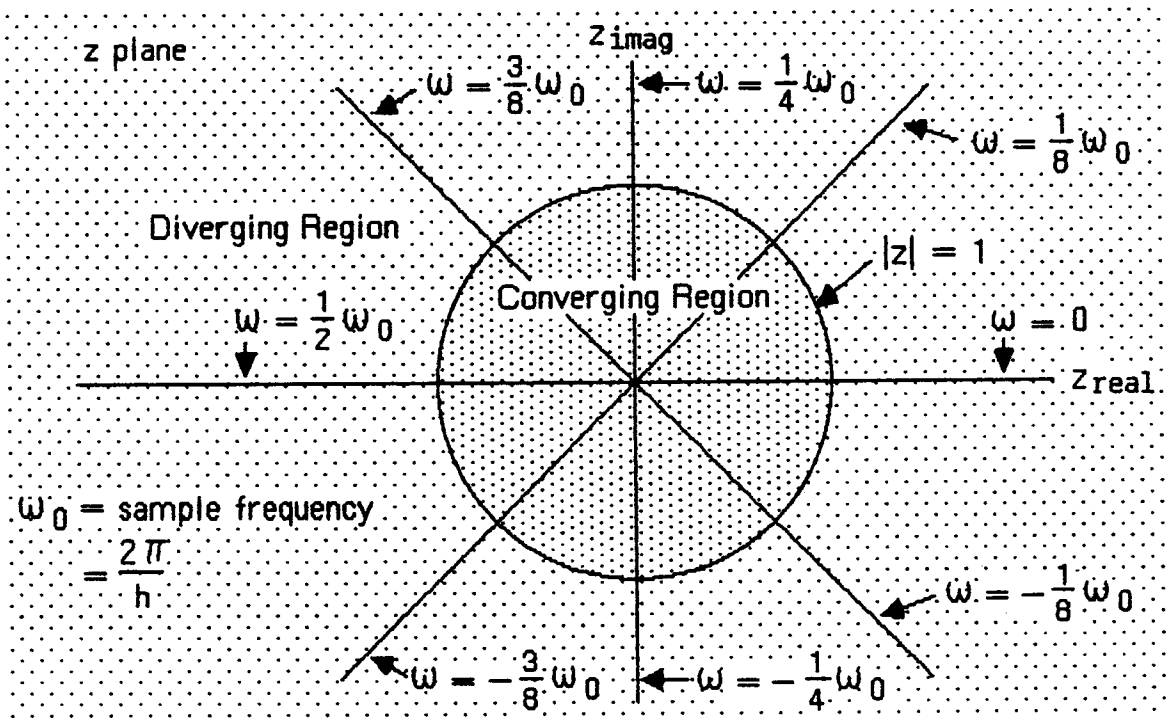



Figure 2.3. Geometric relationship between z transform poles and the stability and frequency of the corresponding data sequences.

Also shown in Figure 2.3 are the frequencies of the data sequences corresponding to the z transform poles, based on Eq. (2.14). Note that poles along the positive real axis correspond to zero frequency, i.e., a pure exponential data sequence. Poles along the negative real axis correspond to data sequences with a frequency equal to

one—half the sample frequency. Along the 45 degree line the frequency equals one—eighth the sample frequency; along the positive imaginary axis, one—fourth the sam—ple frequency; etc. Note that for every z transform pole in the upper—half of the z plane there must be a complex conjugate pole in the lower—half plane. These lower—half plane plane poles correspond to negative frequencies, which are present because of the Euler representation of sinusoids as $e^{j\omega t}$.

It is useful to compare regions in the z plane of Figure 2.3 with the corres—ponding regions of the s plane in the Laplace transform. Thus we see that the unit circle in the z plane corresponds to the imaginary axis in the s plane. Z transform poles must be inside the unit circle in the z plane for the digital system to be stable; Laplace transform poles must lie to the left of the imaginary axis for the continuous system to be stable. Pole locations corresponding to a constant frequency lie along straight line rays passing through the origin in the z plane, with the polar angle given by $\pi\omega/\omega_0$; pole locations corresponding to a constant frequency lie along horizontal lines in the s plane, with the frequency $\omega$ given by the intercept on the imaginary axis. Poles representing oscillatory time functions always occur in complex conjugate pairs in the Laplace transform for continuous systems; this is also true for digital systems for all frequencies except $\omega = \omega_0/2$, in which case a single pole along the the negative real axis represents a sinusoidal data sequence with a frequency equal to one half the sample frequency.

We now examine the stability of the digital system in Figure 2.2, which rep—resents simulation of a first—order linear system using Euler integration. For the system being simulated to be stable, it is clear that the characteristic root $\lambda$ must be negative. From the digital system z transform it is apparent that the single pole is given by

$$z_1 = 1 + \lambda h \tag{2.42}$$

As we vary the step size h from zero to infinity, the pole $z_1$ moves from +1 to mi—nus infinity, as shown in Figure 2.4. This plot in the z plane is analagous to the so—called root locus plot in the s plane for continuous closed—loop systems as the gain constant of the controller is varied from zero to infinity. In Figure 2.4 we see that the pole locus crosses the unit circle in the z plane for $h = -2/\lambda$. For $h > -2/\lambda$ the pole $z_1$ lies to the left of the point $z = -1$, and the simulation will be unstable. The corresponding data sequence will be an exponentially growing oscillation at one—half the sample frequency. In fact, for h in the range $-1/\lambda < h < -2/\lambda$ the pole $z_1$, although within the unit circle, lies on the negative real axis. This means that the corresponding stable data sequence is oscillatory at one—half the sample frequency, even though the continuous system being simulated has a non—oscillatory exponential transient. Figure 2.5 compares the data points from a series of digital solutions with

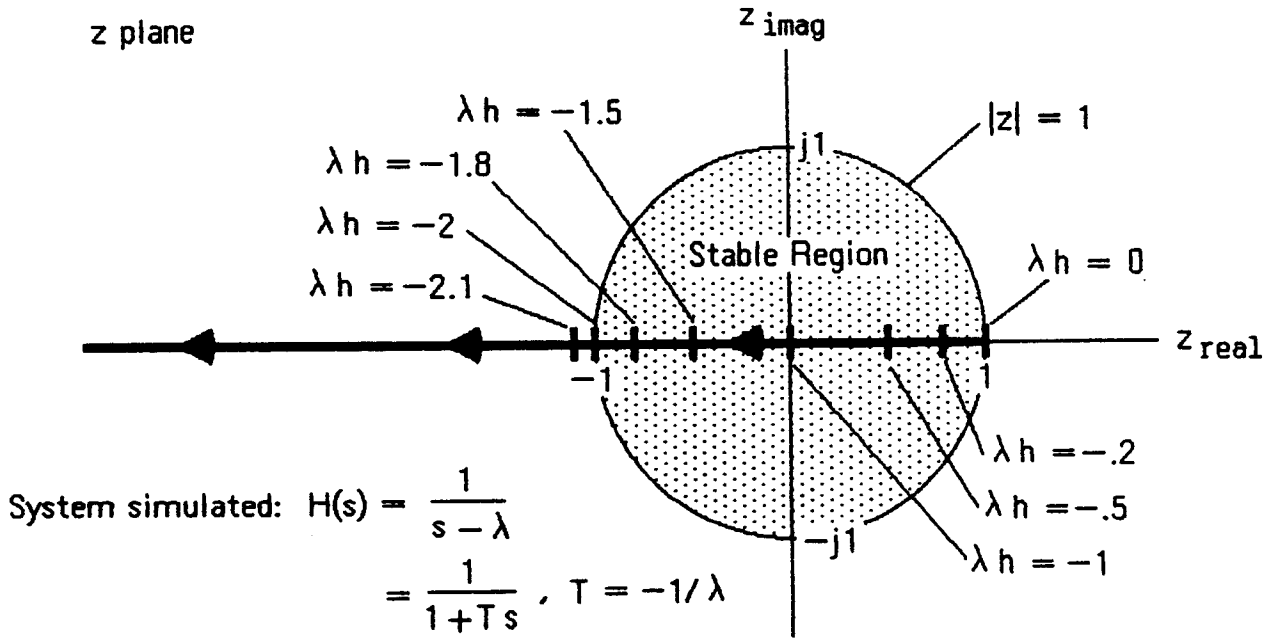
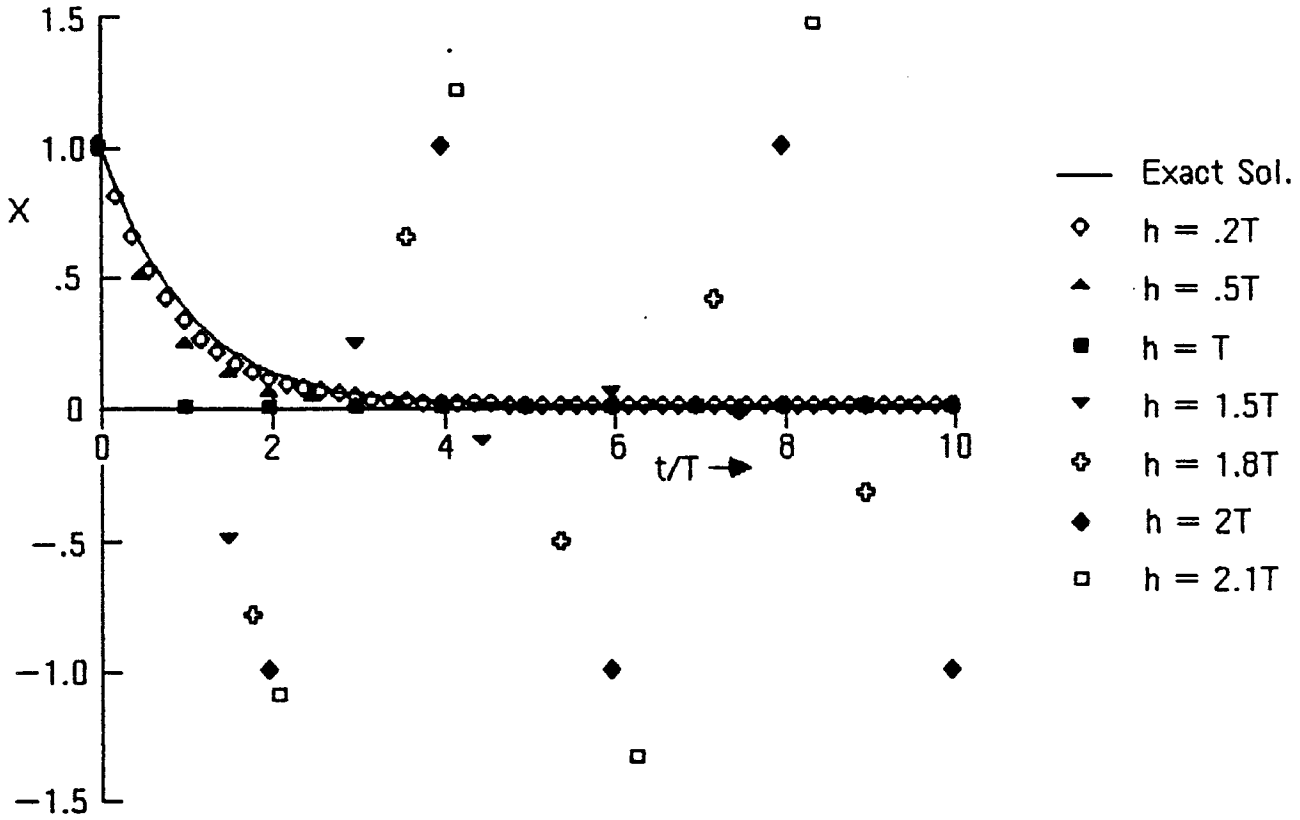



Figure 2.4. Root locus for the pole $z_1$ of the digital system when Euler integration is used to simulate a first–order linear system and $0 < h < \infty$.

the exact solution of the continuous system for x(0) = 1 and f(t) = 0. For each case the step size h corresponds to one of the values shown on the root locus plot in Figure 2.4. For h = .2T (i.e., $\lambda h = -.2$) the digital solution is fairly close to the exact exponential solution $e^{-t/T}$, falling somewhat below it because the characteristic root $\lambda^*$ of the digital system has a slightly larger in magnitude than the ideal root $\lambda$. Thus the digital solution decays somewhat more rapidly than the exact solution. For h = .5T (i.e., $\lambda h = -.5$) the difference between the digital and exact solutions is even more evident. For h = T the digital solution drops to 0 at the first integration step and remains there for all subsequent steps. In this case the pole $z_1 = 0$ and Eq. (2.22) shows that the equivalent characteristic root $\lambda^*$ ($= \sigma$) is also equal to zero. Thus the digital solution remains equal to a constant, namely zero.

For h = 1.5T and 1.8T, Figure 2.5 shows that the digital solutions are damped oscillations at one–half the sample frequency. In each case the corresponding pole $z_1$ in Figure 2.4 lies on the negative real axis, but inside the unit circle in the z plane. For h = 2T (i.e., $\lambda h = -2$) the pole in Figure 2.4 lies at $z_1 = -1$. This results in a digital solution which is an undamped oscillation at one–half the sample frequency. Finally, for h = 2.1T Figure 2.5 shows that the digital solution is a growing oscillation, again at one–half the sample frequency. The corresponding pole $z_1$ in Figure 2.4 lies on the negative real axis, but outside the unit circle in the z plane. This then accounts for the observed instability of the digital solution.




h = integration step size

Figure 2.5. Digital solutions using Euler integration for system with $H(s) = \frac{1}{1+Ts}$

Until now we have only considered the simulation of a first–order linear system using Euler integration, where stabilty of the resulting digital system has been determined by considering the pole $z_1$ of the system z transform. A second–order linear system, when simulated using Euler integration, will have a z transform with two poles, $z_1$ and $z_2$. These poles are related to the two characteristic roots, $\lambda_1$ and $\lambda_2$, of the second–order system by Eq. (2.42) for the case of both real and complex $\lambda$. We recall that $\lambda_1$ and $\lambda_2$ will be a complex conjugate pair when the second–order system is underdamped. For the digital simulation of the second–order system to be stable, both the z transform poles must lie within the unit circle in the z plane. From Eq. (2.42) the stability criterion becomes

$$|1 + \lambda h| < 1 \tag{2.43}$$

$|1 + \lambda h| = 1$ represents a unit circle centered at the origin in the $1 + \lambda h$ plane, or a unit circle centered at $-1$ in the $\lambda h$ plane. This circle is shown in Figure 2.6. It

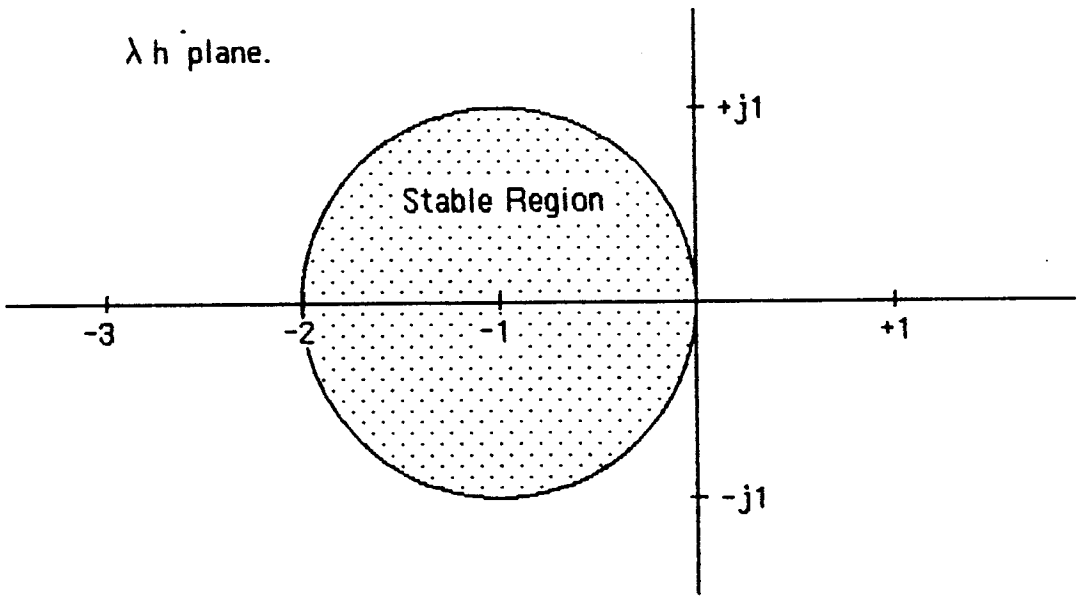



Figure 2.6. Stability region for Euler integration when simulating linear systems.

is the region within which $\lambda h$ must lie for all characteristic roots $\lambda$ when simulating any order linear system with Euler integration using a step size h.

## 2.5 Some Additional Z—Transform Formulas

Consider next the z transform of the data sequence $\{f_{n-k}\}$, which represents a data sequence delayed by kh with respect to the data sequence $\{f_n\}$. From the z—transform definition in Eq. (2.1) it follows that

$$Z\{f_{n-k}\} = z^{-k} F^*(z) \tag{2.44}$$

Thus the z transform for a digital system representing a time delay of k samples is simply $z^{-k}$. Again, it is interesting to compare these formulas with those for continuous systems. The Laplace transform of $f(t-t_d)$ is given by

$$L[f(t-t_d] = e^{-s t_d} F(s) \tag{2.45}$$

where F(s) is the Laplace transform of f(t). It follows that the Laplace transform of a continuous system representing a pure time delay, $t_d$, is $e^{-s t_d}$. Table 2.1 presents the z transforms for a number of commonly encountered data sequences, including those we have already derived in this chapter.

Table 2.1

## z Transforms

$$Z\{f_n\} = F^*(z) = \sum_{n=0}^{\infty} f_n z^{-n}$$

| $f_n$ | $F^*(z)$ |
|---|---|
| $1$ | $\frac{z}{z-1}$ |
| $n$ | $\frac{z}{(z-1)^2}$ |
| $n^2$ | $\frac{z(z+1)}{(z-1)^3}$ |
| $a^n$ | $\frac{z}{z-a}$ |
| $n a^n$ | $\frac{z\,a}{(z-a)^2}$ |
| $(n+1)\,a^{n+1}$ | $\frac{z^2 a}{(z-a)^2}$ |
| $a^n \sin nb$ | $\frac{z\,a \sin b}{z^2 - z\,2a\cos b + a^2}$ |
| $a^n \cos nb$ | $\frac{z(z - a\cos b)}{z^2 - z\,2a\cos b + a^2}$ |

CHAPTER 3

# TRANSFER FUNCTION AND CHARACTERISTIC ROOT ERRORS FOR FIXED–STEP INTEGRATION ALGORITHMS

## 3.1 Introduction

Error analysis of numerical integration algorithms is traditionally based on the residual terms in the Taylor series expansion representing the integral. Although such error measures do establish the dependence of the error on both the integration step size and the magnitude of the appropriate higher derivative, they do not lend much insight into the estimation of the solution errors when applying a given integration algorithm to a specific problem. Indeed, comprehensive error analysis can only be accomplished when the differential equations being integrated are linear, which is seldom the case in most practical applications.

Fortunately, as we have pointed out in Chapter 1, the nonlinear equations can often be linearized about some reference or equilibrium solution, at least for purposes of approximate error analysis. If, in addition, the integration step size is taken to be constant (this is invariably the case in real–time simulation of dynamic systems and can be approximately true over a number of steps when a variable–step method is used), then the z–transform theory introduced in Chapter 2 can be applied. This in turn allows analytic formulas to be developed for the errors in quasi–linear system characteristic roots and the gain and phase errors in quasi–linear system transfer functions. These specific error measures have already been defined in Chapter 1, Eqs. (1.37), (1.38), (1.39), (1.42) and (1.43). They will depend on the integration step size as well as the particular integration algorithm, and can be directly related in a meaningful manner to the overall solution accuracy in simulating dynamic systems.

In this chapter the characteristic root and transfer–function gain and phase errors are developed in convenient asymptotic form for a variety of integration algorithms. For single–pass integration algorithms we will show that the characteristic root errors and transfer function gain and phase errors can be easily written for any order linear system based on knowledge of the gain and phase errors of the algorithm in performing simple isolated integration. The frequency–domain methodology developed in this chapter not only allows quantitative comparison of the dynamic performance of various integration algorithms, but it also points the way to algorithm improvements in specific applications.

## 3.2 Error Measures for Euler Integration

We now illustrate the methodology for deriving the error measures defined in Chapter 1 by considering the simplist of all integration methods, Euler or rectangular integration, applied to the solution of the first-order linear system described by the state equation

$$\frac{dx}{dt} = \lambda x + u(t) \tag{3.1}$$

This is the example we considered earlier in Chapter 2 to illustrate the z-transform method. If we use Euler integration to solve Eq. (3.1) numerically, the following dif-equation is obtained:

$$x_{n+1} = x_n + h(\lambda x_n + u_n) \tag{3.2}$$

where h is the integration step size and $x_n = x(nh)$, $u_n = u(nh)$. Taking the z transform of Eq. (3.2), we have

$$zX^* - zx_0 = X^* + h(\lambda X^* + U^*) \tag{3.3}$$

For $x_0 = 0$ (zero initial condition) we obtain the following formula for the system z transform:

$$\frac{X^*}{U^*} = H^*(z) = \frac{h}{z - (1+\lambda h)} \tag{3.4}$$

As noted in Eq. (2.22), the pole, $z_1 = 1+\lambda h$, of $H^*$ is related to the equivalent continuous characteristic root, $\lambda^*$, by the formula

$$z_1 = e^{\lambda^* h} = 1+\lambda h \tag{3.5}$$

from which

$$\lambda^* = \frac{1}{h} \ln (1+\lambda h) \tag{3.6}$$

An asymptotic formula for $\lambda^*$ when $|\lambda h| << 1$ can be derived by letting $z = e^{\lambda^* h} = 1 + \lambda^* h + (\lambda^* h)^2/2 + \cdots$ in the denominator of Eq. (3.4). Setting the denominator equal to zero and retaining terms up to order $h^2$, we obtain

$$\lambda^* h - \lambda h \cong -\frac{(\lambda^* h)^2}{2} \cong -\frac{(\lambda h)^2}{2} \quad , \quad |\lambda h| << 1$$

This gives us the following formula for the fractional error in characteristic root:

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{\lambda h}{2} \quad , \quad |\lambda h| << 1 \tag{3.7}$$

This is just the result we obtained earlier in Eq. (2.24) by expanding the log function of Eq. (3.6) in a power series. Here we have obtained the asymptotic formula for $e_\lambda$ directly from the denominator of the system z transform. This is the method we will also use later in this chapter in the case of higher–order integration methods, where it is not possible to derive analytic formulas for the poles of the system z transform.

Next we derive the asymptotic formula for the fractional error in the transfer function for sinusoidal inputs. From Eq. (2.37) we have

$$H^*(e^{j\omega h}) = \frac{h}{e^{j\omega h} - 1 - \lambda h} \tag{3.8}$$

for the digital–system transfer function. From Eq. (2.38) the continuous–system transfer function is

$$H(j\omega) = \frac{1}{j\omega - \lambda} \tag{3.9}$$

From Eqs. (3.8) and (3.9) we can write the exact formula for $H^*/H - 1$, the fractional error in digital transfer function. In trigonometric form the formula is given in Eq. (2.39). To obtain a simple asymptotic expression when $\omega h << 1$, we expand the exponential function in the denominator of Eq. (3.8) in a power series, retaining terms of order $h^2$. In this way we obtain

$$H^* \cong \frac{1}{j\omega - \lambda - \omega^2 h/2} = \frac{1}{(j\omega - \lambda)\left[1 - \dfrac{\omega^2 h/2}{j\omega - \lambda}\right]} \cong \frac{1}{j\omega - \lambda}\left[1 - \frac{\omega^2 h/2}{j\omega - \lambda}\right] \quad , \omega h << 1 \tag{3.10}$$

Noting that $1/(j\omega - \lambda) = H$ from Eq. (3.9), we obtain the following formula for the fractional error in digital transfer function:

$$\frac{H^*}{H} - 1 \cong \frac{\omega^2 h/2}{j\omega - \lambda} \cdot \frac{-j\omega - \lambda}{-j\omega - \lambda} = -\frac{\omega\lambda}{2(\omega^2 + \lambda^2)}(\omega h) - j\frac{\omega^2}{2(\omega^2 + \lambda^2)}(\omega h)$$

Comparison with Eqs. (1.42) and (1.43) leads directly to the following asymptotic formulas for the transfer function gain and phase errors.

$$e_H = \frac{|H^*|}{|H|} - 1 \cong -\frac{\omega\lambda}{2(\omega^2 + \lambda^2)}(\omega h) \quad , \omega h << 1 \tag{3.11}$$

$$e_A = \angle H^* - \angle H \cong -\frac{\omega^2}{2(\omega^2 + \lambda^2)}(\omega h) \quad , \omega h << 1 \tag{3.12}$$

As expected, the transfer function gain and phase errors using Euler integration vary as the first power of the integration step size h.

Reference to Eq. (3.1) shows that when the characteristic root $\lambda = 0$, the linear first-order system represents a pure integrator. The corresponding Euler integrator transfer function, $H_I^*$, can be written directly in asymptotic form by setting $\lambda = 0$ in Eq. (3.10). In this way we obtain

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega(1+\frac{1}{2}j\omega h)} \quad , \quad \omega h << 1 \tag{3.13}$$

From Eq. (3.13) it is evident that Euler integration behaves approximately as an ideal integrator (transfer function $= 1/j\omega$) with an additional phase lag of $\omega h/2$, which corresponds to a time delay of h/2. $\lambda$

We next consider the case where the characteristic root (eigenvalue) is com- In terms of the undamped natural frequency $\omega_n$ and the damping ratio $\zeta$ the root can be written as

$$\lambda = -\zeta\omega_n + j\omega_n\sqrt{1-\zeta^2} \tag{3.14}$$

This root will of course be accompanied by a complex conjugate root, so that the root pair represents a second-order linear subsystem with real coefficients. Eq. (3.7) for is valid for complex roots as well as real roots. Substituting Eq. (3.14) for $\lambda$ into Eq. (3.7), we obtain the following formula for $\lambda^*$.

$$\lambda^* = \omega_n(e_r - \zeta) + j\omega_n\sqrt{1-\zeta^2}(e_\omega + 1) \tag{3.15}$$

where for Euler integration

$$e_r \cong (\frac{1}{2} - \zeta^2)\omega_n h \quad , \quad e_\omega \cong \zeta\omega_n h \quad , \quad \omega_n h << 1 \tag{3.16}$$

The imaginary part of $\lambda^*$ in Eq. (3.16) is simply $\omega_d^*$, the frequency of the equivalent digital root. Since $\omega_d = \omega_n\sqrt{1-\zeta^2}$, where $\omega_d$ is the frequency of the continuous-system root, it follows that $e_\omega$ in Eq. (3.15) is just the fractional error in the frequency of the digital root, as defined earlier in Eq. (2.6). For the specific case of Euler integration $e_\omega$ is given by Eq. (3.16).

To compute the equivalent damping ratio $\zeta^*$ of the digital root we must first determine $\omega_n^*$, the equivalent undamped natural frequency. Noting that $|\lambda^*|^2 = \omega_n^{*2}$, we can derive the following asymptotic formula for $\omega_n^*$ from Eq. (3.15).

$$\omega_n^* \cong \omega_n[1 - e_r\zeta + (1-\zeta^2)e_\omega] \quad , \quad |e_r| << 1 \quad , \quad |e_\omega| << 1 \tag{3.17}$$

Equating the real part of $\lambda^*$ in Eq. (3.15) to $-\zeta^*\omega_n^*$ and using Eq. (3.17) for $\omega_n^*$, we obtain the following asymptotic formula for the damping ratio error, $e_\zeta$.

$$e_\zeta = \zeta^* - \zeta \cong -(1-\zeta^2)(e_r + \zeta e_\omega) \quad , \quad |e_r| << 1 \quad , \quad |e_\omega| << 1 \qquad (3.18)$$

With the specific values of $e_r$ and $e_\omega$ given in Eq. (3.16) for Euler integration the asymptotic formula for the damping ratio error becomes

$$e_\zeta = \zeta^* - \zeta \cong \frac{1}{2}(\zeta^2 - 1)\omega_n h \quad , \quad \omega_n h << 1 \qquad (3.19)$$

From Eqs. (3.16) and (3.19) we see that errors in both the frequency and damping ratio vary as the first power of the integration step size h in the case of Euler integration.

For other integration algorithms the equation for $\lambda^*$ will have the same asymptotic form as that shown in Eq. (3.15) for Euler integration. Thus Eq. (3.18) can in general be used to determine the damping-ratio error $e_\zeta$ by substituting the formulas for $e_r$ and $e_\omega$, as derived for each specific integration method.

We next consider the transfer function errors when using Euler integration to simulate the second-order linear system given by

$$\frac{d^2x}{dt^2} + 2\zeta\omega_n\frac{dx}{dt} + \omega_n^2 x = u(t) \qquad (3.20)$$

The transfer function of the continuous system for sinusoidal inputs is given by

$$H(j\omega) = \frac{1}{\omega_n^2 - \omega^2 + j2\zeta\omega_n\omega} \qquad (3.21)$$

We could, of course, use the same procedure we employed earlier in analyzing the transfer function errors in the case of Euler integration of the first-order linear system, namely, write the difference equations, take the z-transform, and determine the digital transfer function from which the asymptotic formulas for the gain and phase errors can be derived. The error formulas can be determined more easily, however, by using the asymptotic formula of Eq. (3.13) for the integrator transfer function when Euler integration is used. Thus we can obtain the approximate digital transfer function for Euler integration of the second-order linear system by replacing $j\omega$ with $j\omega(1+j\omega h/2)$ in Eq. (3.21). In this way we obtain

$$H^* \cong \frac{1}{\omega_n^2 - \omega^2(1+j\omega h) + j2\zeta\omega_n\omega(1+j\omega h/2)} \quad , \quad \omega h << 1 \qquad (3.22)$$

From Eq. (3.22) the following asymptotic formula for the fractional error in transfer function follows directly:

$$\frac{H^*}{H} - 1 \cong \frac{\zeta\omega_n\omega + j\omega^2}{\omega_n^2 - \omega^2 + j2\zeta\omega_n\omega}\omega h \quad , \quad \omega h << 1 \tag{3.23}$$

After rationalization to obtain the real and imaginary parts of H*/H−1, the following formulas are obtained for the transfer−function gain and phase errors for Euler integration applied to a second−order linear system.

$$e_M \cong \frac{\zeta\frac{\omega}{\omega_n}\left[1 + \frac{\omega^2}{\omega_n^2}\right]}{\left[1 - \frac{\omega^2}{\omega_n^2}\right]^2 + \left[2\zeta\frac{\omega}{\omega_n}\right]^2}\omega h, \quad e_A \cong \frac{\frac{\omega^2}{\omega_n^2}\left[1 - 2\zeta^2 - \frac{\omega^2}{\omega_n^2}\right]}{\left[1 - \frac{\omega^2}{\omega_n^2}\right]^2 + \left[2\zeta\frac{\omega}{\omega_n}\right]^2}\omega h, \quad \omega h << 1 \tag{3.24}$$

Again we note that the transfer function gain and phase errors for Euler integration, applied here to the second−order system, are proportional to the first power of the step size h.

## 3.3 Error Measures for All Single−Pass Integration Algorithms

In the previous section we developed characteristic−root and sinusoidal transfer−function error formulas for Euler integration of first and second−order linear systems. We used procedures in developing these formulas which can be generalized to any single−pass integration method. By a single−pass method we mean an integration algorithm which requires only one evaluation of each state−variable derivative per overall integration step. Thus any of the Adams−Bashforth predictor algorithms represents a single−pass method, as does trapezoidal integration when used explicitly (Tustin's method). The Runge−Kutta algorithms represent multiple−pass methods. Although the Adams−Moulton predictor−corrector methods require two passes per integration step, it turns out that the asymptotic error formulas are identical to the equivalent single−pass implicit algorithms based on the corrector formulas.

With single−pass algorithms all of the asymptotic formulas for the characteristic root and transfer function errors can be written directly, once the asymptotic formulas for each individual integrator sinusoidal transfer function are known. Eq. (3.13) represents this formula in the case of Euler integration. As a next example we consider second−order Adams−Bashforth (AB−2) integration. From Eq. (1.8) the difference equation for simple integration, dx/dt = f(t), is given by

$$x_{n+1} = x_n + \frac{h}{2}(3f_n - f_{n-1}) \tag{3.25}$$

Based on $f_n$ and $f_{n-1}$ this algorithm assumes a linear extrapolation of u(t) from t = nh to t = (n+1)h. The area under this extrapolation is added to $x_n$ to compute $x_{n+1}$. Taking the z transform of Eq. (4.1) and assuming $x_0 = 0$, we have

$$zX^* = X^* + \frac{h}{2}(3 - z^{-1})F^* \tag{3.26}$$

from which the following formula is obtained for the AB−2 integrator z transform.

$$H_I^*(z) = \frac{\frac{h}{2}(3 - z^{-1})}{z - 1} \tag{3.27}$$

The integrator transfer function for sinusoidal input data sequences is given by

$$H_I^*(e^{j\omega h}) = \frac{\frac{h}{2}(3 - e^{-j\omega h})}{e^{j\omega h} - 1} \tag{3.28}$$

If we represent the exponential terms in Eq. (3.28) by power series and retain terms to order $h^3$, then the following asymptotic formula is obtained for the AB−2 integrator.

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega(1 - \frac{5}{12}(\omega h)^2} \quad , \quad \omega h << 1 \tag{3.29}$$

Comparison of Eq. (3.29) with the ideal integrator transfer function of $1/j\omega$ shows that the AB−2 integrator behaves approximately like an ideal integrator except for a fractional gain error equal to $(5/12)(\omega h)^2$. To order $h^2$ the AB−2 integrator phase error is zero.

For any kth−order numerical integration algorithm the asymptotic formula for the integrator transfer function will take the form

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega[1 + e_I(j\omega h)^k]} \quad , \quad \omega h << 1 \tag{3.30}$$

Comparison of Eq. (3.30) with Eq. (3.13) shows that for Euler integration, k = 1 and $e_I = 1/2$. Comparison of Eq. (3.30) with Eq. (3.29) shows that for AB−2, k = 2 and $e_I = 5/12$. Table 3.1 at the end of this chapter presents summary formulas for the integrator transfer function error coefficient $e_I$ for predictor, predictor−corrector, power series, and implicit integration algorithms up to order k=4.

For any single–pass integration algorithm it is easily seem that the z transform of the digital system which result when the algorithm is used to solve a linear system with transfer function H(s) is given by

$$H^*(z) = H[1/H_I^*(z)] \tag{3.31}$$

I.e., the argument s in H(s) is simply replaced by $1/H_I^*(z)$, since $H_I^*(z)$ is the digital equivalent to 1/s. In the same way, for any single–pass integration algorithm the transfer function for sinusoidal input data sequences in solving a system with transfer function $H(j\omega)$ is given by

$$H^*(e^{j\omega h}) = H[1/H_I^*(e^{j\omega h})] \tag{3.32}$$

In particular, let $H(s) = 1/(s-\lambda)$ and $H(j\omega) = 1/(j\omega-\lambda)$. Using Eq. (3.30) to represent $H_I^*$, the integrator transfer function, we obtain the following approximate formula for the digital transfer function for sinusoidal inputs.

$$H^*(e^{j\omega h}) \cong \frac{1}{j\omega - \lambda + j\omega e_I (j\omega h)^k} \quad , \omega h << 1 \tag{3.33}$$

Replacing $j\omega$ in Eq. (3.33) with $\lambda^*$ and solving for 1/H*, we obtain

$$\frac{1}{H^*(e^{\lambda^* h})} \cong \lambda^* - \lambda + \lambda^* e_I (\lambda^* h)^k \quad , |\lambda^* h| << 1 \tag{3.34}$$

Setting the right side of Eq. (3.34) equal to zero and solving for $\lambda^*$ determines the value of $\lambda^*$ which makes the denominator of $H^*(e^{\lambda^* h})$ vanish, i.e., the equivalent characteristic root of the digital system. Thus

$$\lambda^* - \lambda \cong -\lambda^* e_I (\lambda^* h)^k \cong -\lambda e_I (\lambda h)^k \quad , |\lambda^* h| << 1 \tag{3.35}$$

and

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -e_I (\lambda h)^k \quad , |\lambda h| << 1 \tag{3.36}$$

Eq. (3.36) represents an asymptotic formula for the fractional error in characteristic root for any single–pass algorithm, where $e_I$ is the integrator transfer–function error coefficient for the specific algorithm. We have already seen for Euler integration that k=1 and $e_I = 1/2$. In this case Eq. (3.36) yields $e_\lambda = -\lambda h/2$, which is precisely the result we derived earlier in Eq. (3.7). For AB–2 integration we found that k=2 and $e_I = 5/12$. From Eq. (3.36) we obtain $-(5/12)(\lambda h)^2$ as the asymptotic formula for $e_\lambda$, the fractional error in characteristic root.

It should be noted that Eqs. (3.35) and (3.36) are not generally valid for multiple-pass integration algorithms, since the overall z transform for the solution of a linear system cannot be obtained by substituting the reciprocal of the integrator z transform for s in H(s) to obtain H*(z). This is because during successive passes the integration formulas are based on intermediate evaluations of states and state derivatives, evaluations which depend on the particular state equations being solved.

From Eq. (3.33) it is easy to obtain the asymptotic formula for the fractional error in sinusoidal transfer function. If we factor $j\omega-\lambda$ from the denominator and note that $1/(j\omega-\lambda) = H(j\omega)$. we obtain the following formula:

$$\frac{H^*}{H} - 1 \cong - \frac{j\omega e_I (j\omega h)^k}{j\omega - \lambda} \tag{3.37}$$

To obtain the separate real and imaginary parts of $H^*/H-1$ we must rationalize the right side of Eq. (3.37). The result will depend on whether the order k of the integration algorithm is odd or even. For odd k we obtain the following formulas for $e_M$, the fractional error in transfer-function gain (i.e., the real part of $H^*/H-1$), and $e_A$, the transfer function phase error (i.e., the imaginary part of $H^*/H-1$).

$$e_M \cong (-1)^{\frac{k+1}{2}} \frac{\omega\lambda e_I}{\omega^2+\lambda^2}(\omega h)^k, \quad e_A \cong (-1)^{\frac{k+1}{2}} \frac{\omega^2 e_I}{\omega^2+\lambda^2}(\omega h)^k, \ \omega h \ll 1 \tag{3.38}$$

On the other hand, for k even we obtain

$$e_M \cong -(-1)^{\frac{k}{2}} \frac{\omega^2 e_I}{\omega^2+\lambda^2}(\omega h)^k, \quad e_A \cong (-1)^{\frac{k}{2}} \frac{\omega\lambda e_I}{\omega^2+\lambda^2}(\omega h)^k, \ \omega h \ll 1 \tag{3.39}$$

For k=1 and $e_I = 1/2$ it is seen that $e_M$ and $e_A$, as given by Eq. (3.38), agree with the asymptotic formulas derived earlier for Euler integration in Eqs (3.11) and (3.12).

Thus far in this section we have developed general formulas for the characteristic root and transfer function errors when using single-pass integration methods to simulate first-order linear systems. We now derive general formulas for these same error measures when simulating second-order linear systems with complex roots. To determine the characteristic root errors we can work directly with the formula already developed in Eq. (3.35) by letting the root $\lambda$ be complex in accordance with Eq. (3.14). For a given order k in the integration algorithm Eq. (3.35) can be rewritten in the form of Eq. (3.15), including formulas for the errors $e_r$ and $e_\omega$. Finally, Eq. (3.18) is used to determine the formula for the damping-ratio error $e_\zeta$. In terms of the integrator error coefficient $e_I$, the asymptotic formulas for $e_\omega$ and $e_\zeta$, the frequency and damping-ratio errors, respectively, are summarized for k = 1, 2, 3,

and 4 in the following equations:

First–order algorithm (k=1), $\omega_n h \ll 1$

$$e_\omega = \frac{\omega_d^*}{\omega_d} - 1 \cong 2\zeta e_I \omega_n h \quad , \quad e_\zeta = \zeta^* - \zeta \cong (\zeta^2 - 1) e_I \omega_n h \tag{3.40}$$

Second–order algorithm (k=2), $\omega_n h \ll 1$

$$e_\omega \cong (1 - 4\zeta^2) e_I (\omega_n h)^2, \quad e_\zeta \cong 2(\zeta - \zeta^3) e_I (\omega_n h)^2 \tag{3.41}$$

Third–order algorithm (k=3), $\omega_n h \ll 1$

$$e_\omega \cong 4(2\zeta^3 - \zeta) e_I (\omega_n h)^3, \quad e_\zeta \cong (1 - \zeta^2)(1 - 4\zeta^2) e_I (\omega_n h)^3 \tag{3.42}$$

Fourth–order algorithm (k=4), $\omega_n h \ll 1$

$$e_\omega \cong -(1 - 12\zeta^2 + 16\zeta^4) e_I (\omega_n h)^4, \quad e_\zeta \cong -2(1 - \zeta^2)(\zeta - 2\zeta^3) e_I (\omega_n h)^4 \tag{3.43}$$

We next consider the derivation of asymptotic formulas for the transfer function gain and phase errors when single–pass integration algorithms are used to solve an underdamped second–order linear system with sinusoidal input. In Section 3.2 we derived these formulas for Euler integration by substituting the asymptotic representation of the Euler integrator into the second–order transfer function $H(j\omega)$. We use the same procedure here by employing Eqs. (3.30) and (3.32), where $H(j\omega)$ is given by Eq. (3.21). Thus

$$H^*(e^{j\omega h}) \cong \frac{1}{\omega_n^2 - \omega^2[1 + 2e_I(j\omega h)^k] + j2\zeta\omega_n\omega[1 + e_I(j\omega h)^k]} \quad ; \omega h \ll 1$$

If we factor $\omega_n^2 - \omega^2 + j2\zeta\omega_n\omega$ from the denominator and set the factor equal to 1/H in accordance with Eq. (3.21), we obtain the following formula for the fractional error in transfer function.

$$\frac{H^*}{H} - 1 \cong \frac{2e_I(j\omega h)^k \left[\dfrac{\omega^2}{\omega_n^2} - j\zeta\dfrac{\omega}{\omega_n}\right]}{\left[1 - \dfrac{\omega^2}{\omega_n^2}\right]^2 + \left[2\zeta\dfrac{\omega}{\omega_n}\right]^2} \quad , \omega h \ll 1 \tag{3.44}$$

To obtain the separate real and imaginary parts of $H^*/H - 1$ we must rationalize the right side of Eq. (4.20). The result will depend on whether the order k of the integration algorithm is odd or even. For odd k we obtain the following formulas for the transfer function gain and phase errors, respectively:

$$e_M \cong (-1)^{\frac{k-1}{2}} \frac{2\zeta\frac{\omega}{\omega_n}\left[1+\frac{\omega^2}{\omega_n^2}\right]e_I}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^k, \quad \omega h<<1 \tag{3.45}$$

$$e_A \cong (-1)^{\frac{k-1}{2}} \frac{2\frac{\omega^2}{\omega_n^2}\left[1-2\zeta^2-\frac{\omega^2}{\omega_n^2}\right]e_I}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^k, \quad \omega h<<1 \tag{3.46}$$

For even k the formulas are

$$e_M \cong (-1)^{\frac{k}{2}} \frac{2\frac{\omega^2}{\omega_n^2}\left[1-2\zeta^2-\frac{\omega^2}{\omega_n^2}\right]e_I}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^k, \quad \omega h<<1 \tag{3.47}$$

$$e_A \cong -(-1)^{\frac{k}{2}} \frac{2\zeta\frac{\omega}{\omega_n}\left[1+\frac{\omega^2}{\omega_n^2}\right]e_I}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^k, \quad \omega h<<1 \tag{3.48}$$

Thus far in this section we have derived formulas for transfer-function gain and phase errors for single-pass integration algorithms in the simulation of first and second-order linear subsystems. It turns out that the transfer-function gain and phase errors in simulating any order of linear system are equal approximately to the sum of the gain and phase errors, respectively, for the individual subsystems. In particular, let the overall transfer function of the continuous system be given by

$$H(s) = \frac{A(s-\lambda_{n+1})(s-\lambda_{n+2})\cdots(s-\lambda_{n+m})}{(s-\lambda_1)(s-\lambda_2)\cdots(s-\lambda_n)} \tag{3.49}$$

Here $\lambda_1, \lambda_2, \ldots, \lambda_n$ represent the n characteristic roots of the linear system and $\lambda_{n+1}, \lambda_{n+2}, \ldots, \lambda_{n+m}$ represent the m zeros of H(s), where $m \leq n$. Using Eq (4.8) we can write the following formula for the digital transfer function for sinusoidal inputs

$$H^*(e^{j\omega h}) = \frac{A\,(1/H_I^* - \lambda_{n+1})(1/H_I^* - \lambda_{n+2}) \cdots (1/H_I^* - \lambda_{n+m})}{(1/H_I^* - \lambda_1)(1/H_I^* - \lambda_2) \cdots (1/H_I^* - \lambda_n)} \tag{3.50}$$

where $H_I^*$ is the digital integrator transfer function. Consider next the single factor $H_q^*$ in Eq. (4.26) which results from the qth characteristic root, i.e., let

$$H_q^* = \frac{1}{(1/H_I^* - \lambda_q)} \tag{3.51}$$

If we denote $e_{Mq}$ and $e_{Aq}$ as the real and imaginary parts of the fractional error in digital transfer function, i.e., $H_q^*/H_q - 1$ as defined in Eq. (1.40), then $H_q^*$ can be represented as

$$H_q^* = (1 + e_{Mq} + j\,e_{Aq})\,H_q \tag{3.52}$$

Here $H_q = (j\omega - \lambda_q)^{-1}$, the sinusoidal transfer function for the factor $(s - \lambda_q)^{-1}$ in Eq. (3.49). Similarly, let $H_{n+q}^*$ be the reciprocal of the factor $(1/H_I^* - \lambda_{n+q})$ in in the numerator of Eq. (3.50) and $H_{n+q}$ be the reciprocal of the factor $(j\omega - \lambda_{n+q})$ of $H(j\omega)$ in Eq. (3.49). Then we can write

$$H_{n+q}^* = (1 + e_{M(n+q)} + j\,e_{A(n+q)})\,H_{n+q} \tag{3.53}$$

where $e_{M(n+q)}$ and $e_{A(n+q)}$ are the real and imaginary parts of the fractional error in digital transfer function, i.e., $H_{n+q}^*/H_{n+q} - 1$. In terms of the individual transfer function factors defined above the overall digital transfer function of Eq. (3.50) can be written as

$$H^* = \frac{H_1^* H_2^* \cdots H_n^*}{H_{n+1}^* H_{n+2}^* \cdots H_{n+m}^*} \tag{3.54}$$

Substituting Eqs. (3.52) and (3.53) for the individual transfer-function factors on the the right side of Eq. (3.54) and neglecting terms above first order in the errors $e_M$ and $e_A$, we obtain the following expression for the fractional error in digital transfer function.

$$\frac{H^*(e^{j\omega h})}{H(j\omega)} - 1 \cong \sum_{q=1}^{n} (e_{Mq} + j\,e_{Aq}) - \sum_{q=n+1}^{n+m} (e_{Mq} + j\,e_{Aq}) \tag{3.55}$$

Comparison with Eqs. (1.42) and (1.43) yields the following formulas for the fractional gain error and the phase error, respectively, of the overall transfer function.

$$e_M = \frac{|H^*|}{|H|} - 1 \cong \sum_{q=1}^{n} e_{Mq} - \sum_{q=n+1}^{n+m} e_{Mq} \tag{3.56}$$

$$e_A = \angle H^* - \angle H \cong \sum_{q=1}^{n} e_{Mq} - \sum_{q=n+1}^{n+m} e_{Mq} \tag{3.57}$$

Eqs. (3.56) and (3.57) are completely general and quite useful. They state that the transfer-function gain and phase errors in digital simulation of any order linear system when using a single-pass integration algorithm are equal to the sum of the gain and phase errors, respectively, in simulation of the individual zero and pole factors. In terms of the individual integrator error coefficient, $e_I$, we have developed simple asymptotic formulas for $e_{Mq}$ and $e_{Aq}$ in Eqs. (3.38) and (3.39) for factors with real zeros or poles, and in Eqs. (3.45) through (3.48) for factors with complex zeros or poles.

## 3.4 Adams-Bashforth Predictor Algorithms

In the next several sections we consider a number of well known single-pass integration methods. For each method we determine the integrator transfer-function error coefficient, $e_I$, which, along with the order k of the method, allows us to use the formulas developed in the previous section for errors in characteristic roots and the transfer-function gain and phase errors in simulating any linear system.

First we consider Adams-Bashforth predictor methods up to 4th order. Euler integration could be termed AB-1, since it is first order (k = 1) and is based on a zero-order extrapolation of the state-variable derivative over the integration step, h. We have seen that $e_I = 1/2$ for Euler integration; this means that the characteristic root errors will be proportional to $\lambda h/2$, as is evident in Eqs. (3.7), (3.16) and (3.19). It also means that the transfer function errors will be proportional to $\omega h/2$, which is evident in Eqs. (3.11, (3.12), and (3.24).

AB-2 integration, represented in Eq. (3.25), is based on a first-order extrapolation from the current and past state-variable derivatives. We have seen that $e_I = 5/12$ and k=2 for AB-2 integration. This means that characteristic root errors will be proportional to $(5/12)(\lambda h)^2$, and transfer-function gain and phase errors will be proportional to $(5/12)(\omega h)^2$.

AB-3 integration, represented in Eq. (1.10), is based on a quadratic extrapolation from the current and past two state-variable derivatives. For integration of

the equation dx/dt = u(t) this results in the following formula:

$$x_{n+1} = x_n + \frac{h}{12}(23u_n - 16u_{n-1} + 5u_{n-2}) \tag{3.58}$$

Taking the z transform and solving for X*/U*, we obtain the AB−3 integrator z transform. Thus

$$H_I^*(z) = \frac{\frac{h}{12}(23z^2 - 16z + 5)}{z^3 - z^2} \tag{3.59}$$

The AB−3 integrator transfer function is simply $H_I^*(e^{j\omega h})$, which takes the following asymptotic form when the exponential functions are replaced by power series with terms retained up to order $h^4$.

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega[1 + \frac{3}{8}(j\omega h)^3]}, \quad \omega h << 1 \tag{3.60}$$

Comparison of Eq. (3.60) with Eq. (3.30) shows that $e_I = 3/8$ and $k = 3$ for AB−3 integration.

AB−4 integration, represented by Eq. (1.12), is based on a cubic extrapolation from the current and past three state−variable derivatives. Integration of the equation dx/dt = u(t) leads to the following formula:

$$x_{n+1} = x_n + \frac{h}{24}(55u_n - 59u_{n-1} + 37u_{n-2} - 9u_{n-3}) \tag{3.61}$$

Taking the z transform and solving for X*/U*, we obtain the AB−4 integrator z transform. Thus

$$H_I^*(z) = \frac{\frac{h}{24}(55z^3 - 59z^2 + 37z - 9)}{z^4 - z^3} \tag{3.62}$$

From the z transform in Eq. (3.63) we can derive the transfer function of the AB−4 integrator, which has the asymptotic form

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega[1 + \frac{251}{720}(j\omega h)^4]}, \quad \omega h << 1 \tag{3.63}$$

Comparing Eq. (3.63) with Eq. (3.30), we see that $e_I = 251/720$ and $k = 4$ for AB−4 integration.

Before considering additional single—pass integration methods, we will discuss briefly the extraneous roots which result from the AB predictor methods. Assume we use AB—2, for example, to solve the first order linear system of Eq. (3.1), which has the transfer function $H(s) = (s-\lambda)^{-1}$. According to Eq. (3.31) we can write the z transform of the resulting digital system by simply replacing s in H(s) with $1/H_I^*$, as given for AB—2 in Eq. (3.27). Thus we obtain

$$H^*(z) = \frac{\frac{h}{2}(3z-1)}{z^2 - (1+\frac{3\lambda h}{2})z + \frac{\lambda h}{2}} \tag{3.64}$$

We note that $H^*(z)$ has two poles because the denominator is a quadratic in z. One of these poles, $z_1$, corresponds to an equivalent characteristic root $\lambda^*$ which will almost equal the ideal root $\lambda$ when $|\lambda h| << 1$. The second pole, $z_2$, corresponds to a second characteristic root which is extraneous. It results from the additional state intro—duced because $u_{n-1}$, the past state—variable derivative, is included in the AB—2 in—tegration algorithm. For small step size ($|\lambda h| << 1$) it can be shown that $z_2 \cong \lambda h/2$ and corresponds to a rapidly decaying transient.

In Chapter 2, Figure 2.3, we observed that whenever a pole of $H^*(z)$ lies out—side the unit circle in the z plane, i.e., exceeds unity in magnitude, the corresponding characteristic root $\lambda^*$ will have a positive real part, leading to an unstable transient. When a pole of $H^*(z)$ lies on the unit circle, the system is neutrally stable. Thus all the poles of $H^*(z)$ must lie inside the unit circle in the z plane for the digital system to be stable. If Eq. (3.1) is to represent a stable continuous system, then $\lambda < 0$ and the dimensionless step size $\lambda h$ is negative. For $\lambda h = -1$, it is apparent from the denominator of Eq. (3.64) that $z = -1$ is a pole of $H^*(z)$. The corresponding $\lambda^*$ from Eq. (3.5) is equal to $j\pi/h$, which represents an undamped oscillation with frequency equal to one—half the integration frame rate 1/h. This corresponds to neutral stabil—ity due to the extraneous root. For any larger integration step size ($\lambda h < -1$) the extraneous root causes an unstable transient and the solution diverges, even though the first pole $z_1$, corresponding to the principal root, represents a stable solution.

For AB—2 simulation of a second—order subsystem there will be two extrane—ous roots in addition to the two principal roots. The locus of dimensionless step sizes $\lambda h$ for which a pole of $H^*(z)$ lies on the unit circle can be calculated by setting the denominator of Eq. (3.64) equal to zero with $|z| = 1$ and solving for $\lambda h$. In particular, we let $z = e^{j\theta}$ and vary the polar angle $\theta$ to generate the locus, obtaining the plot shown for AB—2 in Figure 3.1. For all characteristic roots $\lambda$ and step sizes h such that the product $\lambda h$ remains inside the AB—2 countour shown in the figure, the dig—ital system will be stable. For $\lambda h$ values lying outside the contour the digital system

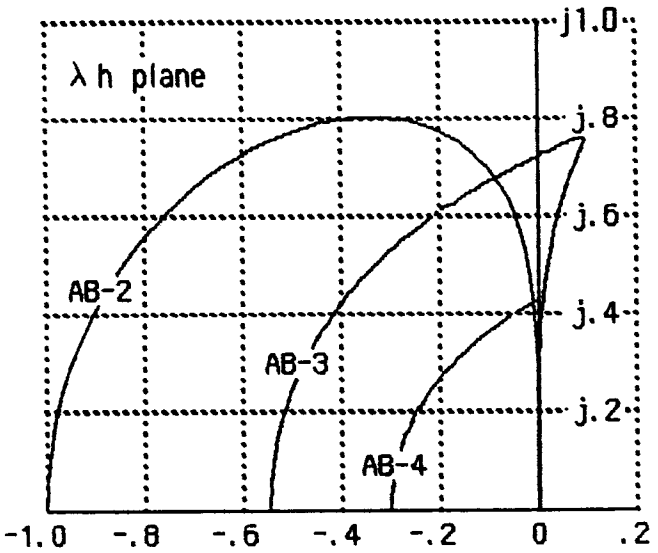



Figure 3.1. Stability boundaries for Adams—Bashforth integration.

is unstable. For $\lambda h$ values lying on the contour the digital system is neutrally stable. The frequency of oscillation is generally less than one—half the sample frequency. The one exception, noted above, is when $\lambda h = -1$, in which case the digital system exhibits neutral stability at exactly one—half the sample frequency.

Figure 3.1 also shows the stability boundaries for AB—3 and AB—4 integration. To obtain the stability boundary in the case of AB—3 integration, we first write the system z transform which results when AB—3 is used to solve a first—order system with characteristic root $\lambda$. Thus we replace $s$ in the transfer function $(s - \lambda)^{-1}$ with $H_I^*$. In this way we obtain

$$H^*(z) = \frac{\frac{h}{12}(23z^2 - 16z + 5)}{z^3 - (1 + \frac{23}{12}\lambda h)z^2 + \frac{16}{12}\lambda h z - \frac{5}{12}\lambda h} \qquad (3.65)$$

Here $H^*(z)$ has three poles because the denominator is a cubic in z. One of the poles corresponds to an equivalent characteristic root $\lambda^*$ which will almost equal the ideal root $\lambda$ when $|\lambda h| << 1$. The other two poles represent characteristic roots which are extraneous. They result from the past two state—variable derivatives which are used in the AB—3 algorithm. For small integration step sizes, i.e., $|\lambda h| << 1$, they correspond to very rapidly decaying transients. If we consider the case for which $z = -1$ is a pole of the z transform in Eq. (3.65) and solve for $\lambda h$, we find that $\lambda h = -6/11$.

Here $z = -1$ corresponds to an extraneous root and leads to an undamped transient at one half the sample frequency. For any larger integration step size ($\lambda h < -1$) the system becomes unstable.

In general, AB–3 integration will introduce two extraneous roots per state variable in the problem being solved. Figure 3.1 shows the stability boundary in the complex $\lambda h$ plane for AB–3 integration. The boundary is obtained by the same method used earlier for AB–2. Thus we set the denominator of Eq. (3.65) equal to zero with $z = e^{j\theta}$ and solve for $\lambda h$ as the polar angle $\theta$ is varied from 0 to $\pi$.

To obtain the stability boundary for AB–4 integration, we write the system z transform which results when AB–4 is used to simulate a first–order system with a characteristic root $\lambda$. Thus we replace s in the transfer function $(s - \lambda)^{-1}$ by $1/H_I^*$, as given for the AB–4 integrator in Eq. (3.62). In this way we obtain

$$H^*(z) = \frac{\frac{h}{24}(55z^3 - 59z^2 + 37z - 9)}{z^4 - (1 + \frac{55}{24}\lambda h)z^3 + \frac{59}{24}\lambda h z^2 - \frac{37}{24}\lambda h z + \frac{9}{24}\lambda h} \qquad (3.66)$$

Here we see that $H^*(z)$ has four poles, three of which correspond to extraneous roots. They result from the past three state–variable derivatives which are used in AB–4 integration. Again, for $|\lambda h| << 1$ they correspond to very rapidly decaying transients and do not cause significant errors. But they can cause instability as the integration step gets larger. If we set the denominator of $H^*(z)$ in Eq. (3.66) equal to zero for $z = -1$ and solve for $\lambda h$, we obtain $\lambda h = -3/10$. Thus AB–4 integration becomes unstable when the step size h is larger than $-1/\lambda$ (we recall that $\lambda$ must be negative for the continuous system to be stable). For $h = -1/\lambda$ the digital system will be neutrally stable with an undamped transient at one–half the sample frequency due to the extraneous pole at $z = -1$. For this same step size, which is only 30 percent of of the continuous system time constant, $-1/\lambda$, the principle root $\lambda^*$ of the AB–4 simulation is within one percent of the ideal root $\lambda$!

Figure 3.2 illustrates the effect of the unstable extraneous root when the integrator step size is too large. Shown in the figure is the response of a first–order linear system to a unit step input as computed using AB–4 integration with an integration step size $h = 0.36$. Since the system being simulated here has a unit time constant (i.e., $\lambda = -1$), $\lambda h = -0.36$. For $h = 0.30$ we have seen above that the extraneous root will produce neutral stability in this case. For the larger step size in Figure 3.2 the instability is very apparent. At the same time we note that the AB–4 solution is very close to the exact solution until such time as the small initial transient associated with the extraneous root grows to significant amplitude.

In general, AB–4 integration will introduce three extraneous roots per state

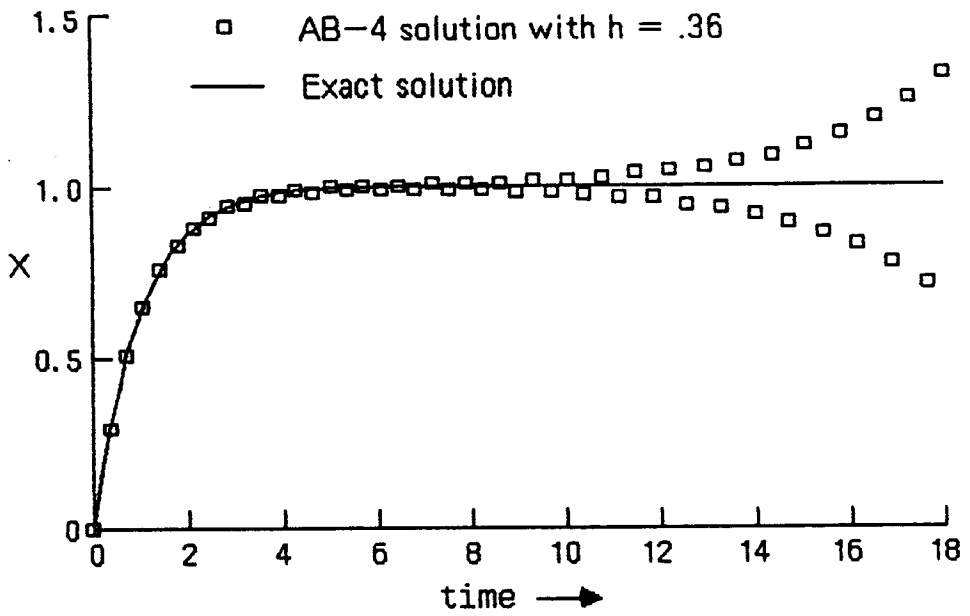



Figure 3.2. Unit step response of first−order system showing instability of AB−4 extraneous root. $H(s) = 1/(s+1)$.

variable. For complex $\lambda$ the AB−4 stability boundary is shown in the $\lambda h$ plane in Figure 3.1. The same method described earlier for AB−2 and AB−3 integration is used to compute the boundary. For values of $\lambda h$ along the left side of the stability region for all three methods, the instability results from an extraneous root. In the case of AB−3 integration, for values of $\lambda h$ along the right side of the stability region, the instability results from the principle root. Consider the case, for example, where $\lambda h$ lies on the imaginary axis, which corresponds to a second−order continuous system with zero damping. Such pure imaginary values of $\lambda h$ lie inside the stability region for AB−3 integration, at least for $|\lambda h| < \sim 0.7$. Thus AB−3 integration introduces positive damping into the simulation of a zero−damped system. This conclusion is consistant with the damping−ratio error formula of Eq. (3.42) along with the AB−3 integrator error coefficient, $e_I = 3/8$, as derived from Eq. (3.60).

From Figure 3.1 it is clear that the higher the order of the predictor integration algorithm, the more restrictive is the upper limit on integration step size based on stability considerations. For real−time simulations, where dynamic accuracy requirements are often moderate (0.1 to 1 percent), it is clear that AB−4 is unlikely to be a strong candidate algorithm due to instability considerations. This is especially true when simulating stiff systems, such as those containing controller subsystems with very small time constants. For adequate stability margin the AB−4 method requires the order of 4 integration steps per shortest time constant in the system. It is likely that such a small step size will provide much more accurate solutions than

are required. Yet an attempt to speed up the solution with even a modest increase in the step size will be likely to cause instability due to an extraneous root. In this case a lower order predictor method, or even a two–pass predictor–corrector algorithm, may prove to be a better compromise.*

From the above discussion it is clear that the higher order AB methods should be used with some care. However, if the characteristic roots of the linearized system are known and, in particular, the magnitude of the largest root is known, then the maximum safe step size can be estimated based on the stability plots in Figure 3.1.

Another disadvantage of the AB predictor methods is the startup problem, which results from the need in the initial integration step to specify the required past values of the state–variable derivatives. These past values can be computed by integrating backwards from the initial time the required number of steps, using an integration method such as Runge–Kutta which has no startup problem. This was in fact the way we produced the AB–4 solution in Figure 3.2. A simpler but less accurate alternative is to use Euler integration for the first step, AB–2 for the second, AB–3 for the third, etc., until the final order of AB algorithm is attained. Subsequent integration steps use that algorithm. In many real–time simulations the small transient startup errors introduced by this scheme are unimportant. It is not an appropriate startup method for the solution of two–point boundary–value problems, however.

### 3.5 Implicit Integration Algorithms

In this section we consider the three specific implicit integration methods that form the basis for the Adams–Moulton predictor–corrector algorithms of order two, three and four. We have already introduced the formulas for these implicit methods in Section 1.2 of Chapter 1. Here we will determine the integrator error coefficient, $e_I$, as defined by Eq. (3.30), for each method. This will, in turn, let us apply the formulas developed in Section 3.3 for the characteristic–root and transfer–function errors. Note that these implicit methods can be used as explicit, single–pass algorithms when applied to linear systems.

The first of the three implicit methods is trapezoidal integration, given earlier in Eq. (1.6). When used to solve $dx/dt = u$, it takes the form

$$x_{n+1} = x_n + \frac{h}{2}(u_{n+1} + u_n) \tag{3.67}$$

* See, for example, R.M. Howe, "Special Considerations in Real–Time Digita Simulation," *Proc. 1983 Summer Computer Simulation Conference*, P.O. Box 2228, La Jolla, California 92038, pp 64–71.

Taking the z transform of Eq. (3.67) and solving for $H_I^*$, we obtain

$$H_I^*(z) = \frac{\frac{h}{2}(z+1)}{z-1} \tag{3.68}$$

Replacing z with $e^{j\omega h}$ in Eq. (3.69), we obtain $H_I^*(e^{j\omega h})$, the trapezoidal integrator transfer function for sinusoidal inputs. Thus

$$H_I^*(e^{j\omega h}) = \frac{\frac{h}{2}(e^{j\omega h}+1)}{(e^{j\omega h}-1)} = \frac{\frac{h}{2}\left[\frac{e^{j\omega h/2}+e^{-j\omega h/2}}{2}\right]}{j\left[\frac{e^{j\omega h/2}-e^{-j\omega h/2}}{j2}\right]} = \frac{\frac{h}{2}\cos(\omega h/2)}{j\sin(\omega h/2)}$$

or

$$H_I^*(e^{j\omega h}) = \frac{h/2}{j\tan(\omega h/2)} \tag{3.69}$$

Noting that $\tan x = x + x^3/3 + \cdots$, we can rewrite Eq. (3.69) as follows:

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega[1-\frac{1}{12}(j\omega h)^2]}, \quad \omega h << 1 \tag{3.70}$$

Comparison with Eq. (3.30) shows that the integrator error coefficient $e_I = -1/12$ and $k = 2$. Earlier, based on Eq. (3.29), we determined that $e_I = 5/12$ for AB–2 integration. Thus it is clear that trapezoidal integration is five times more accurate than AB–2 integration for small step sizes. From Section 3.3 we know that this accuracy comparison extends as well to characteristic root errors and transfer function gain and phase errors. The trapezoidal method has the further advantager that it does not introduce any extraneous roots. However, it cannot be used as a single–pass method in the solution of nonlinear differential equations because of the implicit nature of the algorithm.

In Eq. (1.9) we introduced the third–order implicit algorithm, which takes the following form when used to solve the equation dx/dt = u:

$$x_{n+1} = x_n + \frac{h}{12}(5u_{n+1} + 8u_n - u_{n-1}) \tag{3.71}$$

Taking the z transform of Eq. (3.67) and solving for $H_I^*$, we obtain

$$H_I^*(z) = \frac{\frac{h}{12}(5z+8-z^{-1})}{z-1} \tag{3.72}$$

Again, the integrator transfer function for sinusoidal inputs is $H_I^*(e^{j\omega h})$, which has the following asymptotic form when the exponential functions are expanded in power series and terms to order $h^4$ are retained:

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega[1 - \frac{1}{24}(j\omega h)^3]}, \quad \omega h << 1 \tag{3.73}$$

Comparison of Eq. (3.73) with Eq. (3.30) shows that $e_I = -1/24$ and $k = 3$ for this third-order implicit integration algorithm. On the other hand Eq. (3.60) shows that $e_I = 3/8$ for the AB-3 predictor algorithm algorithm. Thus the third-order implicit method is nine times more accurate than the third-order predictor method. But the implicit method can only be used explicitly, i.e., without time-consuming iterations, when the system being simulated is linear. Also, it introduces an extraneous root for each state variable in the system being simulated. This is because the algorithm in Eq. (3.71) depends on a past derivative. Despite the presence of the extraneous root, however, instability only occurs for very large step sizes. For example, when this third-order implicit algorithm is used to simulate a system which has a negative real characteristic root $\lambda$, it is easy to show that the simulation becomes unstable only when $|\lambda h| > 6$. This is a very large step size indeed, and should not be a significant deterent in the selection of this third-order implicit integration method.

Finally, we consider the fourth-order implicit integration algorithm given in Eq. (1.11). When used to solve the equation $dx/dt = u$, it takes the form

$$x_{n+1} = x_n + \frac{h}{24}(9u_{n+1} + 19u_n - 5u_{n-1} + u_{n-2}) \tag{3.74}$$

Taking the z transform and solving for $H_I^*$, we obtain

$$H_I^*(z) = \frac{\frac{h}{24}(9z + 19 - 5z^{-1} + z^{-2})}{z - 1} \tag{3.75}$$

As noted previously, the integrator transfer function for sinusoidal inputs is given by $H_I^*(e^{j\omega h})$. When the exponential functions are expanded in power series and terms to order $h^5$ are retained, the following asymptotic formula for the integrator transfunction results:

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega[1 - \frac{19}{720}(j\omega h)^3]}, \quad \omega h << 1 \tag{3.76}$$

Comparison of Eq. (3.76) with Eq. (3.30) shows that $e_I = -19/720$ and $k = 4$ for this fourth–order implicit integration algorithm. For AB–4 integration we found in Eq. (3.63) that $e_I = 251/720$. Thus the fourth–order implicit method of Eq. (3.74) is 251/19 or 13.2 times more accurate than the AB–4 predictor method. However, it can only be used as an explicit algorithm in the simulation of linear systems. Also, it introduces two extraneous roots for each state variable in the system being simulated. This is because the algorithm in Eq. (3.74) utilizes two past derivatives. These extraneous roots do not, however, cause significant stability problems for large integration step sizes. For example, when the system being simulated has a negative real root $\lambda$, the simulation only becomes unstable when $|\lambda h| > 3$. This represents a step size equal to three times the time constant associated with the negative real root.

Figure 3.3 shows the stability boundaries in the $\lambda h$ plane for all three of the implicit integration algorithms considered in this section. The figure shows that the stability boundary for the second–order algorithm, which represents trapezoidal in–

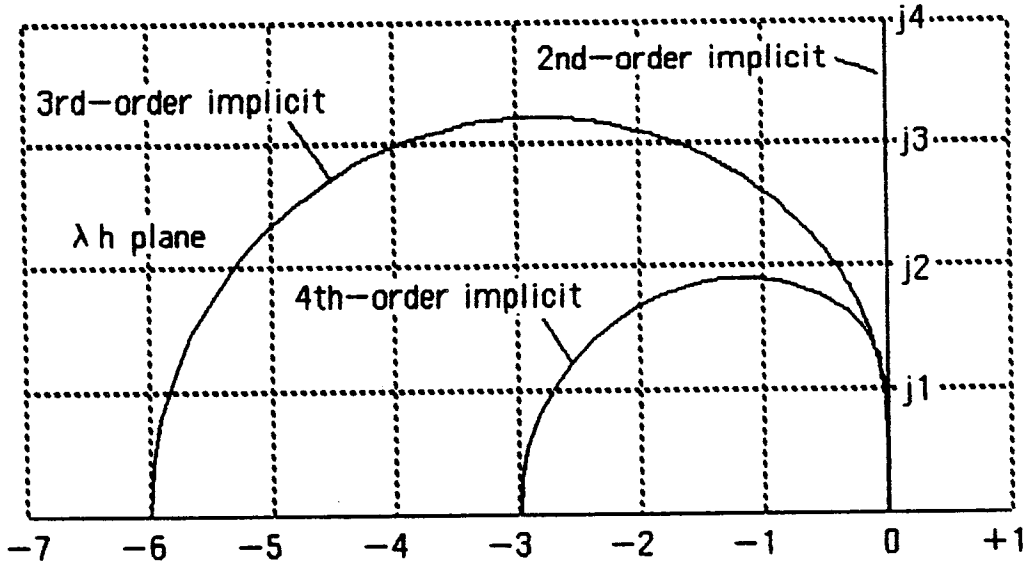



Figure 3.3. Stability boundaries for implicit integration methods.

tegration, is the imaginary axis. The stability region is therefore the entire left–half plane. Thus trapezoidal integration, when used to simulate a stable linear system, will always produce a stable solution regardless of the integration step size. For the third and fourth–order implicit algorithms the step size h must be such that $\lambda h$ remains within the region shown in the left–half plane for the digital solution to be stable.

### 3.6 Power Series Integration Algorithms

When the derivatives of the state variables are functions which can be differentiated analytically, then the following power series algorithm can be used for numerical integration of the state equation, $dx/dt = u$:

$$x_{n+1} = x_n + hu_n + \frac{h^2}{2!}u'_n + \frac{h^3}{3!}u''_n + \cdots \quad (3.77)$$

where $u' = du/dt$, $u'' = d^2u/dt^2$, etc. The order of the integration method depends on the number of terms utilized before the power series is truncated. Halin has been particularly successful in exploiting the power series method as an efficient means of numerical integration of nonlinear differential equations.* This type of integration algorithm is often used with a variable step size. However, we will assume here that the step size is fixed and that the nonlinear equations have been linearized so that we can use the method of z transforms to analyze the dynamic errors. The results of this analysis will give us insight into the dynamic performance of the power series method compared with other integration algorithms.

Taking the z transform of Eq. (3.77) and assuming that $x_0 = 0$, we obtain

$$zX^* = X^* + hU^* + \frac{h^2}{2!}U'^* + \frac{h^3}{3!}U''^* + \cdots \quad (3.78)$$

To derive the integrator transfer function for a sinusoidal input data sequence we replace z by $e^{j\omega h}$ and note that $u_n = e^{j\omega nh}$, $u'_n = j\omega e^{j\omega nh} = j\omega u_n$, $u''_n = (j\omega)^2 e^{j\omega nh} = (j\omega)^2 u_n$, etc. Then it follows that Eq. (3.78) becomes

$$e^{j\omega nh}X^* = X^* + h\left[1 + \frac{(j\omega h)}{2!} + \frac{(j\omega h)^2}{3!} + \cdots\right]U^*$$

Solving for $X^*/U^* = H_I^*$, the power series integrator transfer function, we obtain

$$H_I^*(e^{j\omega h}) = \frac{h\left[1 + \frac{(j\omega h)}{2!} + \frac{(j\omega h)^2}{3!} + \cdots + \frac{(j\omega h)^{k-1}}{k!}\right]}{e^{j\omega h} - 1} \quad (3.79)$$

where we have truncated the power series at the $h^{k-1}$ term. Expanding $e^{j\omega h}$ in a power series and retaining terms up to order $h^k$, we obtain the following asymptotic

* H.J. Halin, "Integration across Discontinuities in Ordinary Differential Equations Using Power Series," *Simulation* Vol. 32, No. 2, February, 1979, pp 33–45.

formula for the power series integrator transfer function:

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega[1 + \frac{(j\omega h)^k}{(k+1)!}]}, \quad \omega h << 1 \tag{3.80}$$

Comparison with Eq. (3.30) shows that the kth–order power series integration algorithm has an integrator error coefficient $e_I$ given by

$$e_I = \frac{1}{(k+1)!} \tag{3.81}$$

Since the power series integration method is a single–pass algorithm, the value of $e_I$ in Eq. (3.81) can be used directly in the asymptotic formulas of Section 3.3 for characteristic root errors, and for transfer function gain and phase errors.

### 3.7 Runge–Kutta Integration Algorithms

Next we consider the Runge–Kutta multiple–pass algorithms which were introduced in Section 1.2. We consider first a version of RK–2 known as Heun's method, which from Eqs. (1.19) and (1.20) takes the following form when solving the state equation given by (3.1).

$$X_{n+1} = X_n + \frac{h}{2}(\lambda X_n + U_n + \lambda \hat{X}_{n+1} + U_{n+1}), \text{ where } \hat{X}_{n+1} = X_n + h(\lambda X_n + U_n) \tag{3.82}$$

Here $\hat{X}_{n+1}$ is an estimate of $X_{n+1}$ based on Euler integration. Taking the z transform of Eq. (3.82) and solving for $X^*(z)$, we obtain

$$X^*(z) = \frac{z x_0}{z - (1 + \lambda h + \frac{\lambda^2 h^2}{2})} + H^*(z) F^*(z) \tag{3.83}$$

Here $H^*(z)$, the system z transform, is given by

$$H^*(z) = \frac{\frac{h}{2}(z + 1 + \lambda h)}{z - (1 + \lambda h + \frac{\lambda^2 h^2}{2})} \tag{3.84}$$

As noted in Eq.(2.22), the pole, $z_1 = 1 + \lambda h + \lambda^2 h^2/2$, of $H^*(z)$ is related to the the equivalent cgaracteristic root, $\lambda^*$, by the formula

$$\lambda^* = \frac{1}{h} \ln \left(1 + \lambda h + \frac{\lambda^2 h^2}{2}\right) \tag{3.85}$$

The asymptotic formula for $\lambda^*$ when $|\lambda h| << 1$ is obtained by letting $z = e^{\lambda^* h} = 1 + \lambda^* h + (\lambda^* h)^2/2 + \cdots$ in the denominator of Eq. (3.84). Setting the denominator equal to zero and retaining terms up to order $h^3$, we obtain

$$\lambda^* - \lambda + \frac{1}{2}(\lambda^* - \lambda)(\lambda^* + \lambda)h + \frac{1}{6}(\lambda^*)^3 h^2 = 0 \tag{3.86}$$

Next we assume that $|\lambda h| << 1$ and $|\lambda^* - \lambda| << |\lambda|$. Then the middle term on the left side of Eq. (3.86) is negligible and we have

$$\lambda^* - \lambda \cong -\frac{1}{6}\lambda^3 h^2, \quad |\lambda h| << 1 \tag{3.87}$$

This equation is valid for both real and complex $\lambda$. The fractional error in characteristic root, $e_\lambda$, is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{1}{6}\lambda^2 h^2, \quad |\lambda h| << 1 \tag{3.88}$$

As expected for RK–2 integration, the error varies as the square of the integration step size h. From Eqs. (3.29) and (3.36) we see that the comparable error for AB–2 integration is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{5}{12}\lambda^2 h^2, \quad |\lambda h| << 1$$

Thus the AB–2 root error is 2.5 times that of RK–2. On the other hand, the RK–2 algorithm requires two passes through the state equations per integration step. This means that RK–2 will generally take twice as long to execute each integration step when compared with AB–2. This results in double the step size and hence four times the root error, since we are dealing with a second–order method. The net effect is to make the AB–2 error equal to 2.5/4 or 5/8ths the root error obtained when using RK–2 integration.

We next consider the transfer function for sinusoidal inputs, $H_I^*(e^{j\omega h})$. This is obtained by substituting $e^{j\omega h}$ for z in Eq. (3.84). The asymptotic formulas for the transfer function gain and phase errors are obtained by representing $e^{j\omega h}$ with a power series and retaining terms to order $h^2$ in the numerator of Eq. (3.84), and $h^3$ in the denominator of Eq. (3.84). The resulting formula for $H_I^*(e^{j\omega h})$ takes the following form:

$$H_I^*(e^{j\omega h}) \cong \frac{1 + b + jc}{j\omega - \lambda + d + je} \tag{3.89}$$

where for RK–2 integration

$$b = \frac{\lambda}{2}h - \frac{1}{4}\omega^2 h^2, \quad c = \frac{\omega}{2}h, \quad d = -\left(\frac{\lambda^2}{2} + \frac{\omega^2}{2}\right)h, \quad e = -\frac{1}{6}\omega^3 h^2 \tag{3.90}$$

Next we factor $(j\omega - \lambda)$ out of the denominator of Eq. (3.89) and obtain

$$H_I^*(e^{j\omega h}) \cong \frac{1 + b + jc}{(j\omega - \lambda)\left[1 + b + jc - b - jc + \frac{d + je}{j\omega - \lambda}\right]}$$

$$= \frac{1}{(j\omega - \lambda)\left[1 - \frac{(b + jc)(j\omega - \lambda) - d - je}{(1 + b + jc)(j\omega - \lambda)}\right]}$$

or

$$H_I^*(e^{j\omega h}) \cong \frac{1}{j\omega - \lambda}\left[1 + \frac{(b + jc)(j\omega - \lambda) - d - je}{j\omega - \lambda}\right] \tag{3.91}$$

Here we have assumed that $|b| << 1$, $|c| << 1$, and $|d + je| << |j\omega - \lambda|$. Noting that $1/(j\omega - \lambda)$ is equal to $H(j\omega)$, we can write the following expression for the fractional error of the digital transfer function:

$$\frac{H^*}{H} - 1 \cong \frac{(b + jc)(j\omega - \lambda) - d - je}{j\omega - \lambda} \tag{3.92}$$

Using Eq. (3.90) for b, c, d and e, we obtain

$$\frac{H^*}{H} - 1 \cong \frac{\frac{1}{4}\lambda\omega^2 h^2 - j\frac{1}{12}\omega^3 h^2}{j\omega - \lambda} \cdot \frac{-j\omega - \lambda}{-j\omega - \lambda}$$

or

$$\frac{H^*}{H} - 1 \cong \frac{-\frac{1}{12}\omega^2 h^2(\omega^2 + 3\lambda^2) - j\frac{1}{6}\lambda\omega^3 h^2}{\omega^2 + \lambda^2}, \quad \omega h << 1 \tag{3.93}$$

We have seen in Eqs. (1.42) and (1.43) that the real and imaginary parts on the right

side of Eq. (3.93) are equal to the fractional gain error and the phase error, respectively, of the digital transfer function for sinusoidal inputs. Thus we have

$$\frac{|H^*|}{|H|} - 1 \;=\; \text{fractional error in gain} \cong e_M \cong -\frac{\omega^2 + 3\lambda^2}{12(\omega^2 + \lambda^2)}(\omega h)^2$$

$$\angle H^* - \angle H \;=\; \text{phase error} \cong e_A \cong -\frac{\omega\lambda}{6(\omega^2 + \lambda^2)}(\omega h)^2, \quad \omega h << 1 \tag{3.94}$$

As expected, the transfer function gain and phase errors, when using RK–2 integration to simulate a first–order system, vary as the second power of the integration step size h.

The characteristic root errors which result when RK–2 integration is used to simulate an underdamped second–order subystem can be computed from Eq. (3.87) by letting the root $\lambda$ be complex in accordance with Eq. (3.14). Thus we obtain the equivalent digital root $\lambda^*$ as defined in Eq. (3.15), with the parameters $e_r$ and $e_\omega$ given by the following asymptotic formulas:

$$e_r \cong \left(\frac{2}{3}\zeta^3 - \frac{1}{2}\zeta\right)(\omega_n h)^2, \quad e_\omega \cong \frac{1}{6}(1 - 4\zeta^2)(\omega_n h)^2, \quad \omega_n h << 1 \tag{3.95}$$

Here $e_\omega$ is the fractional error in the frequency of the characteristic root, as defined by Eq. (1.38). Substituting Eq. (3.95) into Eq. (3.18), we obtain the formula for the damping ratio error. Thus

$$e_\zeta = \zeta^* - \zeta = \frac{1}{3}(\zeta - \zeta^3)(\omega_n h)^2, \quad \omega_n h << 1 \tag{3.96}$$

Again, both the frequency and damping ratio errors vary as $h^2$.

Next we consider the transfer function for sinusoidal inputs when RK–2 integration is used to simulate the second–order system given by Eq. (3.20). The transfer function H(s) can be written as the sum of two first–order transfer functions in accordance with the following formula:

$$H(s) = \frac{1}{s^2 + 2\zeta\omega_n s + \omega_n^2} = \frac{j}{2\omega_n\sqrt{1 - \zeta^2}}[H_2(s) - H_1(s)] \tag{3.97}$$

where

$$H_1(s) = \frac{1}{s - \lambda_1}, \quad H_2(s) = \frac{1}{s - \lambda_2} \tag{3.98}$$

and the characteristic roots $\lambda_1$ and $\lambda_2$ are given by

$$\lambda_1 = -\zeta\omega_n + j\omega_n\sqrt{1-\zeta^2}, \quad \lambda_2 = -\zeta\omega_n - j\omega_n\sqrt{1-\zeta^2}, \tag{3.99}$$

From Eq. (3.97) it is clear that we can represent the z transform of a digital system that is simulating an underdamped second–order system in terms of the sum of the z transforms for two first–order system simulations. Thus

$$H^*(z) = \frac{j}{2\omega_n\sqrt{1-\zeta^2}}\left[H_2^*(z) - H_1^*(z)\right] \tag{3.100}$$

where $H_1^*(z)$ and $H_2^*(z)$ are the z transforms corresponding to the digital simulations of $H_1(s)$ and $H_2(s)$, respectively, as defined in Eq. (3.98). It follows that the error, $H^*(e^{j\omega h}) - H(j\omega)$, in the digital transfer function for sinusoidal inputs can be written in terms of the digital transfer function errors for the first–order subsys–systems, i.e.,

$$H^*(e^{j\omega h}) - H(j\omega) = \frac{j}{2\omega_n\sqrt{1-\zeta^2}}\Big(\left[H_2^*(e^{j\omega h}) - H_2(j\omega)\right] - \left[H_1^*(e^{j\omega h}) - H_1(j\omega)\right]\Big) \tag{3.101}$$

From Eq. (3.91) with $\lambda = \lambda_2$ we see that $H_2^*(e^{j\omega h}) - H_2(j\omega)$ takes the form

$$H_2^* - H_2 = \omega_n \frac{u_r + v_r\sqrt{1-\zeta^2} + j(u_i + v_i\sqrt{1-\zeta^2})}{(j\omega - \lambda_2)^2} \tag{3.102}$$

where for RK–2 integration

$$u_r = -\frac{\zeta\omega^2h^2}{4}, \quad v_r = 0, \quad u_i = -\frac{\omega^2h^2}{12}\left(\frac{\omega}{\omega_n}\right), \quad v_i = -\frac{\omega^2h^2}{4} \tag{3.103}$$

Similarly, for $\lambda = \lambda_1$ we obtain the following formula for $H_1^* - H_1$:

$$H_1^* - H_1 = \omega_n \frac{u_r - v_r\sqrt{1-\zeta^2} + j(u_i - v_i\sqrt{1-\zeta^2})}{(j\omega - \lambda_1)^2} \tag{3.104}$$

Substituting Eqs. (3.102) and (3.104) into Eq. (3.101) and noting that 1/H is equal to $(j\omega - \lambda_1)(j\omega - \lambda_2)$, we obtain the following formulas for the real part, $e_M$, and the imaginary part, $e_A$, of the fractional error in digital transfer function in simu–lating a second–order undamped linear system:

$$e_M \cong \frac{1}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}\left\{2\zeta\left[1+\frac{\omega^2}{\omega_n^2}\right]u_r+2\frac{\omega}{\omega_n}\left[2\zeta^2-1+\frac{\omega^2}{\omega_n^2}\right]u_i\right.$$

$$\left.-4\zeta\frac{\omega}{\omega_n}\left[1-\frac{\omega^2}{\omega_n^2}\right]v_r+\left[1-2\zeta^2-2\zeta^2\frac{\omega^2}{\omega_n^2}-\frac{\omega^4}{\omega_n^4}\right]v_i\right\} \qquad (3.105)$$

$$e_A \cong \frac{1}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}\left\{2\frac{\omega}{\omega_n}\left[1-2\zeta^2-\frac{\omega^2}{\omega_n^2}\right]u_r+2\zeta\left[1+\frac{\omega^2}{\omega_n^2}\right]u_i\right.$$

$$\left.+\left[2\zeta^2-1+2\zeta^2\frac{\omega^2}{\omega_n^2}+\frac{\omega^4}{\omega_n^4}\right]v_r+4\zeta\frac{\omega}{\omega_n}\left[\zeta^2-1\right]v_i\right\} \qquad (3.106)$$

When we substitute into Eqs. (3.105) and (3.106) the formulas for $u_r$, $u_i$, $v_r$, and $v_i$ given in Eq. (3.103) for RK–2 integration, we obtain the following asymptotic formulas for the RK–2 transfer function gain and phase errors in simulating a second–order system:

$$e_M = \frac{|H^*|}{|H|}-1 \cong \frac{\frac{1}{12}\left[-3+(2-4\zeta^2)\frac{\omega^2}{\omega_n^2}+\frac{\omega^4}{\omega_n^4}\right]}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^2, \quad \omega h \ll 1 \qquad (3.107)$$

$$e_A = \angle H^* - \angle H \cong \frac{\frac{1}{3}\zeta\frac{\omega}{\omega_n}\left[1+\frac{\omega^2}{\omega_n^2}\right]}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^2, \quad \omega h \ll 1 \qquad (3.108)$$

Note that the transfer function gain and phase errors, as given by Eqs. (3.107) and (3.108), vary as the square of h, the integration step size.

Before considering higher order RK integration methods, we examine the real–time RK–2 algorithm introduced in Chapter 1. Like conventional RK–2, real–time RK–2 is a two–step method that employs Euler integration for the first step. But the first step is actually a half–step, which utilizes Euler integration to compute the state half–way through the integration step. This estimate is then used to calculate

the derivative, which in turn forms the basis for the calculation of the state one full time-step later. Applying the real-time RK-2 method, as defined in Eqs. (1.21) and (1.22), to the first-order equation given by (3.1) leads to the following difference equation:

$$\hat{X}_{n+1/2} = X_n + \frac{h}{2}(\lambda X_n + U_n), \quad X_{n+1} = X_n + h(\lambda \hat{X}_{n+1/2} + U_{n+1/2}) \tag{3.109}$$

Using a sample period T = h/2, we can rewrite Eq. (3.109) as the following difference equation:

$$X_{n+2} = (1 + 2T\lambda + 2T^2\lambda^2)X_n + 2T^2\lambda U_n + 2TU_{n+1} \tag{3.110}$$

Taking the z transform, replacing T by h/2, and solving for $H^*(z) = X^*(z)/F^*(z)$, we obtain

$$H^*(z) = \frac{h(z + \frac{1}{2}\lambda h)}{z^2 - (1 + \lambda h + \frac{1}{2}\lambda^2 h^2)} \tag{3.111}$$

We determine the equivalent characteristic root $\lambda^*$ by setting the denominator of $H^*$ equal to zero with $z = e^{\lambda^* T} = e^{\lambda^* h/2}$. Then $z^2 = e^{\lambda^* h}$ and the denominator of $H^*$ in Eq. (3.111) becomes the same as the denominator of $H^*$ in Eq. (3.84) for the standard RK-2 method. Thus $\lambda^*$ for real-time RK-2 is the same as $\lambda^*$ for standard RK-2 and the characteristic root errors given in Eqs. (3.88), (3.95), and (3.96) when simulating first and second-order systems are therefore the same for both methods.

To obtain the digital transfer function for sinusoidal inputs we let $z = e^{j\omega T} = e^{j\omega h/2}$ in Eq. (3.111) for $H^*(z)$. We then follow the same procedure used earlier in Eqs. (3.89) to (3.94) to obtain the transfer function gain and phase errors. In this way the following formulas are obtained for real-time RK-2 simulation of a first-order linear subsystem:

$$\frac{|H^*|}{|H|} - 1 = \text{fractional error in gain} \cong e_M \cong \frac{\omega^2 - 3\lambda^2}{24(\omega^2 + \lambda^2)}(\omega h)^2$$

$$\angle H^* - \angle H = \text{phase error} \cong e_A \cong -\frac{\omega\lambda}{6(\omega^2 + \lambda^2)}(\omega h)^2, \quad \omega h << 1 \tag{3.112}$$

Comparison with Eq. (3.94) for standard RK-2 shows that the gain error for real-time RK-2 is less than half as large. This improved accuracy is doubtless due to the

fact that $u_n$ and $u_{n+1/2}$ serve as inputs for the nth integration frame in real-time RK-2, which means that the input is sampled at twice the integration frame rate. In standard RK-2 the inputs for the nth integration frame are $u_n$ and $u_{n+1}$, which represents an input sample rate equal to the integration frame rate.

The transfer function gain and phase errors in real-time RK-2 simulation of second-order linear subsystems are determined using the same procedure employed to obtain Eqs. (3.107) and (3.108) for standard RK-2 integration. The real-time RK-2 asymptotic formulas for transfer function gain error, $e_M$, and phase error, $e_A$, are given in Table 3.2 at the end of this chapter. Comparison with the formulas for standard RK-2 shows that the phase errors are identical, but that the gain error for real-time RK-2 is at least a factor of two smaller. Again, this can be ascribed to the doubled input sampling rate used in real-time RK-2 integration.

We next consider the RK-3 algorithm defined by Eqs. (1.24) through (1.27) in Chapter 1. When applied to the first-order linear system described by Eq. (3.1), the method yields the following difference equation:

$$X_{n+1} = X_n + \frac{h}{4}(\lambda X_n + 3\lambda \hat{X}_{n+2/3} + U_n + 3U_{n+2/3})$$

where

$$\hat{X}_{n+2/3} = X_n + \frac{2h}{3}(\lambda \hat{X}_{n+1/3} + U_{n+1/3}), \quad \hat{X}_{n+1/3} = X_n + \frac{h}{3}(\lambda X_n + U_n) \tag{3.113}$$

Using a sample period $T = h/3$, we can rewrite Eq. (3.113) as the following difference equation:

$$X_{n+3} = (1 + 3T\lambda + \frac{9}{2}T^2\lambda^2 + \frac{9}{2}T^3\lambda^3)X_n + (\frac{3}{4}T + \frac{9}{2}T^3\lambda^2)U_n + \frac{9}{2}T^2\lambda U_{n+1} + \frac{9}{4}T U_{n+2} \tag{3.114}$$

Taking the z transform, replacing T by h/3, and solving for $H^*(z) = X^*(z)/U^*(z)$, we obtain

$$H^*(z) = \frac{\frac{3}{4}hz^2 + \frac{1}{2}\lambda h^2 z + (\frac{1}{4}h + \frac{1}{6}\lambda^2 h^3)}{z^3 - (1 + \lambda h + \frac{1}{2}\lambda^2 h^2 + \frac{1}{6}\lambda^2 h^3)} \tag{3.115}$$

We determine the equivalent characteristic root $\lambda^*$ by setting the denominator of $H^*$ equal to zero with $z = e^{\lambda^* T} = e^{\lambda^* h/3}$. Then $z^3 = e^{\lambda^* h} = 1 + \lambda^* h + (\lambda^* h)^2/2 + \cdots$ in the denominator of Eq. (3.84). Retaining terms up to order $h^4$, we obtain the following formula for the characteristic rooterror:

$$\lambda^* - \lambda + \frac{1}{2}(\lambda^* - \lambda)(\lambda^* + \lambda)h + \frac{1}{6}(\lambda^* - \lambda)(\lambda^{*2} + \lambda^*\lambda + \lambda^2)h^2 + \frac{1}{24}(\lambda^*)^4 h^3 = 0 \tag{3.116}$$

Next we assume that $|\lambda h| << 1$ and $|\lambda^* - \lambda| << |\lambda|$. Then the middle terms on the left side of Eq. (3.116) are negligible and we have

$$\lambda^* - \lambda \cong -\frac{1}{24}\lambda^4 h^3, \quad |\lambda h| << 1 \tag{3.117}$$

This equation is valid for both real and complex $\lambda$. The fractional error in characteristic root, $e_\lambda$, is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{1}{24}\lambda^3 h^3, \quad |\lambda h| << 1 \tag{3.118}$$

As expected for RK–3 integration, the error varies as the cube of the integration step size h. From Eqs. (3.36) and (3.60) we see that the comparable error for AB–3 integration is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{3}{8}\lambda^3 h^3, \quad |\lambda h| << 1$$

Thus the AB–3 root error is 9 times that of RK–3. On the other hand, the RK–3 algorithm requires three passes through the state equations per integration step. This means that RK–3 will generally take three times as long to execute each integration step when compared with AB–3. This results in three times the step size and hence 27 times the root error, since we are dealing with a third–order method. The net effect is to make the AB–3 error equal to 9/27 or 1/3rd the root error obtained using RK–2 integration.

To obtain the digital transfer function for sinusoidal inputs we let $z = e^{j\omega T} = e^{j\omega h/3}$ in Eq. (3.115) for $H^*(z)$. We then follow the same procedure used earlier in Eqs. (3.89) to (3.94) to obtain the transfer function gain and phase errors. In this way the following formulas are obtained for RK–3 simulation of a first–order linear subsystem:

$$\frac{|H^*|}{|H|} - 1 = \text{fractional error in gain} \cong e_M \cong \frac{\omega\lambda - 2\frac{\lambda^3}{\omega}}{72(\omega^2 + \lambda^2)}(\omega h)^3 \tag{3.119}$$

$$\angle H^* - \angle H = \text{phase error} \cong e_A \cong -\frac{\omega^2 - 8\lambda^2}{216(\omega^2 + \lambda^2)}(\omega h)^3, \quad \omega h << 1$$

The characteristic root errors and transfer function errors for RK–3 simulation of underdamped second–order linear systems are obtained using the same procedures employed for RK–2 integration. The asymptotic error formulas are given in in Table 3.2 at the end of this chapter.

Finally, we consider the RK–4 integration algorithm defined by Eqs. (1.29) through (1.33) in Chapter 1. When applied to the first–order linear system described by Eq. (3.1), the method yields the following difference equation:

$$X_{n+1} = X_n + \frac{h}{6}(\lambda X_n + 2\lambda \hat{X}_{n+1/2} + 2\lambda \hat{\hat{X}}_{n+1/2} + \lambda \hat{X}_{n+1} + U_n + 4U_{n+1/2} + U_{n+1})$$

where

$$\hat{X}_{n+1/2} = X_n + \frac{h}{2}(\lambda X_n + U_n), \quad \hat{\hat{X}}_{n+1/2} = X_n + \frac{h}{2}(\lambda \hat{X}_n + U_{n+1/2}), \qquad (3.120)$$

$$\hat{X}_{n+1} = X_n + h(\lambda \hat{\hat{X}}_{n+1/2} + U_{n+1})$$

Using a sample period $T = h/2$, we can rewrite Eq. (3.120) as the following difference equation:

$$\begin{aligned} X_{n+2} = {} & (1 + 2T\lambda + 2T^2\lambda^2 + \frac{4}{3}T^3\lambda^3 + \frac{2}{3}T^4\lambda^4)X_n \\ & + (\frac{1}{3}T + \frac{2}{3}T^2\lambda + \frac{2}{3}T^3\lambda^2 + \frac{2}{3}T^4\lambda^3)U_n \\ & + (\frac{4}{3}T + \frac{4}{3}T^2\lambda + \frac{2}{3}T^3\lambda^2)U_{n+1} + \frac{1}{3}TU_{n+2} \end{aligned} \qquad (3.121)$$

Taking the z transform, replacing T by h/2, and solving for $H^*(z) = X^*(z)/U^*(z)$, we obtain

$$H^*(z) = \frac{\frac{h}{6}[z^2 + (4 + 2\lambda h + \frac{1}{2}\lambda^2 h^2)z + 1 + \lambda h + \frac{1}{2}\lambda^2 h^2 + \frac{1}{4}\lambda^3 h^3]}{z^2 - (1 + \lambda h + \frac{1}{2}\lambda^2 h^2 + \frac{1}{6}\lambda^2 h^3 + \frac{1}{24}\lambda^2 h^3)} \qquad (3.122)$$

As before, we determine the equivalent characteristic root $\lambda^*$ by letting $z = e^{\lambda^* T} = e^{\lambda^* h/2}$ in the denominator of $H^*(z)$ and setting the denominator equal to zero. Expanding $e^{\lambda^* h/2}$ in a power series and retaining terms to $h^5$, we can solve for the asymptotic formula for the charcateristic root error, $\lambda^* - \lambda$. In this way we obtain

$$\lambda^* - \lambda \cong -\frac{1}{120}\lambda^4 h^3, \quad |\lambda h| << 1 \qquad (3.123)$$

Again, this equation is valid for both real and complex $\lambda$. The fractional error in the

characteristic root, $e_\lambda$, is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{1}{120}\lambda^2 h^2, \quad |\lambda h| << 1 \tag{3.124}$$

As expected for RK–3 integration, the error varies as the fourth power of the integration step size. From Eqs. (3.36) and (3.63) we see that the comparable error for AB–4 integration is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{251}{720}\lambda^2 h^2, \quad |\lambda h| << 1$$

Thus the AB–3 root error is 251/6 or 41.83 times that of RK–4. On the other hand, the RK–4 algorithm requires four passes through the state equations per integration step. This means that RK–4 will in general take four times as long to execute each integration step compared with AB–4. This results in four times the step size and hence 256 times the root error, since we are dealing with a fourth–order method. The net effect is to make the AB–4 error equal to 41.83/256 or about 1/6th the root error obtained when using RK–4 integration.

To obtain the digital transfer function for sinusoidal inputs we let $z = e^{j\omega T} = e^{j\omega h/2}$ in Eq. (3.122) for $H^*(z)$. We then follow the same procedure used earlier in Eqs. (3.89) to (3.94) to obtain the transfer function gain and phase errors. In this way the following formulas are obtained for RK–4 simulation of a first–order linear subsystem:

The

$$\frac{|H^*|}{|H|} - 1 = \text{fractional error in gain} \cong e_M \cong \frac{\omega^2 - 5\lambda^2 - 30\dfrac{\lambda^4}{\omega^2}}{2880(\omega^2 + \lambda^2)}(\omega h)^4$$

$$\angle H^* - \angle H = \text{phase error} \cong e_A \cong \frac{\omega\lambda - 5\dfrac{\lambda^3}{\omega}}{720(\omega^2 + \lambda^2)}(\omega h)^4, \quad \omega h << 1 \tag{3.125}$$

The characteristic root errors and transfer function errors for RK–4 simulation of underdamped second–order linear systems are obtained using the same procedures employed for RK–2 integration. The asymptotic error formulas are given in in Table 3.2 at the end of this chapter.

We have noted in this section that Runge–Kutta algorithms enjoy a significant accuracy advantage over Adams–Bashforth predictor algorithms of the same order for a given integration step size. However, this advantage is more than offset by the additional RK execution time that results from multiple passes per integration step. Also, as noted in Chapter 1, RK algorithms are not compatible with real–time in–

puts. It should be noted, however, that AB algorithms can present startup problems, perform poorly in response to step inputs, and generate extraneous roots which can cause instability even for moderate integration step sizes. The RK algorithms suffer from none of these problems. Even though RK methods introduce no extraneous roots, it is nevertheless important to examine the region of stability in the $\lambda h$ plane when using RK integration in order to determine the maximum allowable step size. This is accomplished with the same procedure used in Section 3.4 for AB integration. Thus we calculate the values of $\lambda h$ for which a pole of $H^*(z)$ lies on the unit circle in the z plane. In particular, we let $z = e^{j\theta}$ in the denominator of $H^*(z)$ and solve for $\lambda h$ as the polar angle is varied between 0 and $\pi$. Application of this procedure to $H^*(z)$ as given in Eqs. (3.111), (3.115) and (3.122) for RK–2, 3 and 4, respectively, leads to the stability boundaries shown in Figure 3.4. Values of $\lambda h$ lying within the boundaries correspond to stable solutions.

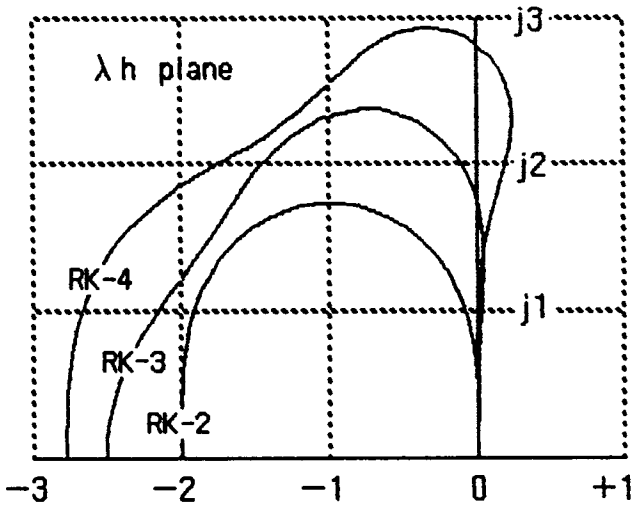



Figure 3.4. Stability boundaries for Runge–Kutta integration algorithms.

Comparison of Figure 3.4 with Figure 3.1 shows that the stability regions for RK integration are very much larger than those for AB integration. Note also that the higher the order of RK integration, the larger the stability region. This is exactly the reverse of AB integration, where the higher the order, the smaller the stability region. This same observation applies when the RK stability boundaries of Figure 3.4 are compared with those in Figure 3.3 for the implicit algorithms. Thus the second–order implicit method (trapezoidal integration) has a stability region which includes the entire left–half $\lambda h$ plane, whereas the fourth–order implicit integration shown

in Figure 3.3 exhibits a stability region that is comparable with that of RK–4.

## 3.8 Adams–Moulton Predictor–Corrector Algorithms

In Chapter 1 we noted that the AB predictor algorithms can be combined with implicit algorithms to mechanize the AM (Adams–Moulton) predictor–corrector algorithms. In this section we examine the AM–2, 3 and 4 integration methods with respect to characteristic root errors, and transfer function gain and phase errors. We will also consider the extraneous roots which these methods introduce, as well as the overall stability regions in the $\lambda h$ plane.

First we consider the second–order predictor–corrector algorithm, AM–2. In Eqs. (1.13) and (1.14) the general form of the equations is presented. When we apply these equations to the first–order linear system given in Eq. (3.1), we obtain the following single difference equation:

$$x_{n+1} = (1 + \lambda h + \frac{3}{4}\lambda^2 h^2)x_n - \frac{1}{4}\lambda^2 h^2 x_{n-1} + \frac{h}{2}u_{n+1} + (\frac{h}{2} + \frac{3}{4}\lambda h^2)u_n - \frac{1}{4}\lambda h^2 u_{n-1} \qquad (3.126)$$

Taking the z transform and solving for $H^*(z) = X^*(z)/U^*(z)$, we obtain

$$H^*(z) = \frac{\frac{h}{2}[z^2 + (1 + \frac{3}{2}\lambda h)z - \frac{1}{2}\lambda h)}{z^2 - (1 + \lambda h + \frac{3}{4}\lambda^2 h^2)z + \frac{1}{4}\lambda^2 h^2} \qquad (3.127)$$

The two poles of $H^*(z)$ (i.e., the two roots of the denominator), $z_1$ and $z_2$, determine the equivalent charcteristic roots. In particular, one of the two roots corresponds to the ideal characteristic root, $\lambda$, in accordance with the formula $z_1 = e^{\lambda^* h}$. Here $\lambda^*$, as before, represents the equivalent characteristic root of the digital system. Since the denominator of $H^*(z)$ in Eq. (3.127) is a quadratic in z, we can solve analytically for the two poles, $z_1$ and $z_2$, and thus determine the exact value of $\lambda^*$ as well as the exact value of the equivalent extraneous root. As we have noted in the previous sections, it is also convenient to determine the approximate formula for $\lambda^*$ when the step size h is small. As before, we do this by replacing z with $e^{\lambda^* h}$ in the denominator of Eq. (3.127), expanding $e^{\lambda^* h}$ in a power series, and solving for the root error $\lambda^* - \lambda$. When this is done, the asymptotic result agrees exactly with the formula obtained for 2nd–order implicit (i.e., trapezoidal) integration. Thus the fractional error in characteristic root, $e_\lambda$, in accordance with Eqs. (3.36) and (3.70),

is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong \frac{1}{12}\lambda^2 h^2, \quad |\lambda h| << 1 \tag{3.128}$$

Comparison with the AB–2 formula for $e_\lambda$ following Eq. (3.88) shows that the AB–2 root error is 5 times larger in magnitude. On the other hand, AM–2 is a two–pass method. This means that it will in general require twice the execution time per integration step. For second–order algorithms, doubling the step size means that the accuracy is reduced by a factor of four. The net result is that AB–2 will exhibit a characteristic root error that is 5/4 or 1.25 times that of AM–2. On this basis we conclude that AM–2 enjoys a slight edge as a second–order algorithm. But it does have the disadvantage as a real–time method of requiring the input one–half frame before it is available in real time, as pointed out in Figure 1.8.

From Eq. (3.127) we can obtain the exact transfer function for sinusoidal inputs when AM–2 is used to simulate a first–order subsystem, by replacing z with $e^{j\omega h}$. The approximate asymptotic formulas for transfer function gain and phase errors are obtained with the same procedures used in Eqs. (3.89) through 3.94) in the case of RK–2 integration. When this is done, the asymptotic formulas turn out to be identical, again, with those based on second–order implicit integration. Thus Eq. (3.39) with $e_I = -1/12$ and $k = 2$ yields the asymptotic formulas for the transfer function gain error $e_H$, and the phase error $e_A$ when using AM–2 to simulate a 1st–order linear subsystem. In fact it turns out that all of the AM–2, 3 and 4 asymptotic formulas for characteristic root and transfer function errors are identically the same as those for the implicit algorithms of Section 3.5. These in turn are given by the single–pass formulas of Section 3.3, with the integration error coefficients $e_I$ equal to $-1/12$, $-1/24$ and $-19/720$, for AM–2, 3 and 4 integration, respectively. For this reason it is not necessary to rederive the asymptotic error formulas for the AM predictor–corrector algorithms.

The exact formulas for characteristic root and transfer function errors are, of course, unique for each AM predictor–corrector algorithm. For AM–3 integration they can be obtained by applying Eqs. (1.15) and (1.16) to the first–order linear system described by Eq. (3.1). When this is done, the following equation is obtained for $H^*(z)$, the system z transform in AM–3 simulation of a first–order linear subsystem:

$$H^*(z) = \frac{\frac{h}{12}[5z^3 + (8 + \frac{115}{12}\lambda h)z^2 - (1 + \frac{20}{3}\lambda h)z + \frac{25}{12}\lambda h}{z^3 - (1 + \frac{13}{12}\lambda h + \frac{115}{144}\lambda^2 h^2)z^2 + (\frac{1}{12}\lambda h + \frac{20}{3}\lambda^2 h^2)z - \frac{25}{144}\lambda^2 h^2} \tag{3.129}$$

The three poles of $H^*(z)$ can be used to calculate the exact equivalent characteristic roots for the digital system. One of these roots, $\lambda^*$, corresponds to the ideal root, $\lambda$. The other two roots are extraneous. As noted above for AM–2, the asymptotic formula for the fractional error in characteristic root, $e_\lambda$, for AM–3 integration is identical to the formula derived earlier for the 3rd–order implicit algorithm. In accordance with Eqs. (3.36) and (3.73) the formula is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong \frac{1}{24}\lambda^3 h^3, \quad |\lambda h| << 1 \tag{3.130}$$

Comparison with the AB–3 formula for $e_\lambda$ following Eq. (3.118) shows that the AB–3 root error is 9 times larger in magnitude. On the other hand, AM–3 is a two–pass method. This means that it will in general require twice the execution time per integration step. For third–order algorithms, doubling the step size means that the accuracy is reduced by a factor of eight. The net result is that AB–3 will exhibit a characteristic root error that is 9/8 or 1.125 times that of AM–3. On this basis we conclude that AM–3 enjoys a slight edge as a third–order algorithm. Again, it has the disadvantage as a real–time method of requiring the input one–half frame before it is available in real time, as pointed out in Figure 1.8. It should also be noted that the AM–3 root error in Eq. (3.130) has the same magnitude (but opposite polarity) as the RK–3 root error in Eq. (3.118). Since RK–3 is a three–pass method, it will in general require 1.5 times longer to execute per integration step than AM–3 integration. This in turn means that RK–3 will exhibit a root error that is 1.5 cubed or 3.375 times that of AM–3.

From Eq. (3.129) we can obtain the exact transfer function for sinusoidal inputs when AM–3 is used to simulate a first–order subsystem, by replacing z with $e^{j\omega h}$. The approximate asymptotic formulas for transfer function gain and phase errors are, as noted above, identical with the errors for the third–order implicit algorithm in Section 3.5. Thus Eq. (3.38) applies, with $e_I = -1/24$ and $k = 3$.

Finally, we consider AM–4 simulation of the first–order system given by Eq. (3.1). The difference equation is obtained by applying Eqs. (1.17) and (1.18) to (3.1). Taking the z transform and solving for $H^*(z)$, we obtain

$$H^*(z) = \frac{\frac{h}{8}[3z^4 + (\frac{19}{3} + \frac{55}{8}\lambda h)z^3 - (\frac{5}{3} + \frac{59}{8}\lambda h)z^2 + (\frac{1}{3} + \frac{37}{8}\lambda h)z - \frac{9}{8}\lambda h}{z^4 - (1 + \frac{7}{6}\lambda h + \frac{55}{64}\lambda^2 h^2)z^3 + (\frac{5}{24}\lambda h + \frac{59}{64}\lambda^2 h^2)z^2 - (\frac{1}{24}\lambda h + \frac{37}{64}\lambda^2 h^2)z - \frac{9}{64}\lambda^2 h^2} \tag{3.131}$$

The four poles of $H^*(z)$ can be used to calculate the exact equivalent characteristic roots for the digital system. One of these roots, $\lambda^*$, corresponds to the ideal root, $\lambda$. The other three roots are extraneous. As noted for AM–2 and 3, the asymptotic formula for the fractional error in characteristic root, $e_\lambda$, for AM–4 integration is identical to the formula derived earlier for the 4th–order implicit algorithm. In accordance with Eqs. (3.36) and (3.76) the formula is given by

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong \frac{19}{720}\lambda^3 h^3, \quad |\lambda h| << 1 \tag{3.132}$$

Comparison with the AB–4 formula for $e_\lambda$ following Eq. (3.124) shows that the AB–4 root error is 251/19 or 13.2 times larger in magnitude. On the other hand, AM–4 is a two–pass method. This means that it will in general require twice the execution time per integration step. For fourth–order algorithms, doubling the step size means the accuracy is reduced by a factor of 16. The net result is that AB–4 will exhibit a characteristic root error that is 13.2/16 or 0.826 times that of AM–4. On this basis AB–4 enjoys a slight edge over AM–4 as a fourth–order integration algorithm. We also note that AB–4 is compatible with real–time inputs. However, it has the disadvantage of having a much smaller stability region in the $\lambda h$ plane than AM–4, as we shall see later in this section.

Comparison of Eq. (3.132) for AM–4 with Eq. (3.124) for RK–4 shows that the AM–4 root error is 19/6 or 3.167 times larger than the RK–4 root error. Since RK–4 is a four–pass method, however, it will in general take twice as long to execute per integration step as the two–pass AM–4 method. For fourth–order algorithms, doubling the step size means increasing the error by a factor of sixteen. The net result is that AM–4 will exhibit a root error which is 19/96 or 0.198 times that of RK–4 integration.

We have already noted that the asymptotic formulas for characteristic root errors, and for transfer function gain and phase errors, turn out for AM methods to be identical with the formulas developed in Sections 3.3 and 3.5 for the implicit integration methods of corresponding order. Thus Eqs. (3.41), (3.42) and (3.43) with $e_I$ equal to $-1/12$, $-1/24$ and $-19/720$ for AM–2, 3 and 4, respectively, give the formulas for the frequency and damping ratio errors in AM simulation of underdamped second–order systems. Similarly, the transfer function gain and phase errors are given for AM–3 by Eqs. (3.45) and (3.46), and for AM–2 and AM–4 by Eqs. (3.47) and (3.48).

We now examine the stability regions in the $\lambda h$ plane for AM integration. The method used to generate the stability boundaries is the same as that used previously for AB, implicit, and RK integration. Thus we let $z = e^{j\theta}$ (i.e., $|z|=1$) in the de–

nominator of Eqs. (3.127), (3.128) and (3.129) for AM–2, 3 and 4, respectively. The denominators are then set equal to zero and the dimensionless step size $\lambda h$ is determined for different values of the polar angle $\theta$. In this way the stability boundaries shown in Figure 3.5 are generated. Values of $\lambda h$ lying within the boundaries correspond to stable solutions.

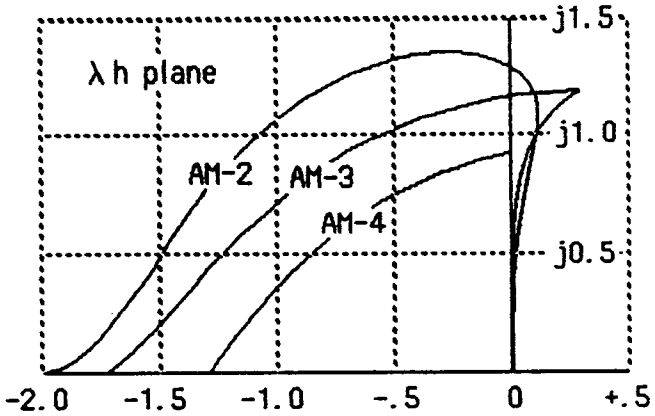



Figure 3.5. Stability boundaries for Adams–Moulton integration algorithms.

Comparison of Figure 3.5 with Figure 3.1 shows that the stability regions for AM integration are significantly larger than those for AB integration. In particular, consider the stability boundary when the characteristic root $\lambda$ is negative real, corresponding to a stable first–order linear subsystem. The AM–2 algorithm becomes unstable for $|\lambda h| > 2$, compared with $|\lambda h| > 1$ for AB–2 integration. For both algorithms the instability is due to the extraneous root. It is clear that AM–2 permits double the integration step size before instability results. For negative real $\lambda$, AM–3 becomes unstable for $|\lambda h| > 1.7288$, versus $|\lambda h| > 0.5556$ for AB–3; AM–4 becomes unstable for $|\lambda h| > 1.2848$ compared with $|\lambda h| > 0.3$ for AB–4.

On the other hand, comparison of Figure 3.5 with Figure 3.4 shows that the stability regions for AM integration are generally smaller than those for RK integration, especially for the higher–order algorithms. Thus the choice of the optimal integration method involves a number of tradeoffs, including dynamic accuracy needs, the stiffness of the system (i.e., the ratio of maximum to minimum characteristic–root magnitudes), and the requirements for real–time inputs and outputs. Weighting factors in these tradeoffs will become more clear as we consider additional methods and examples in the chapters to follow.

## 3.9 Single–Pass Adams–Moulton Integration Algorithms

An interesting modification of the two–pass Adams–Moulton integration formulas of the previous section utilizes derivatives based only on the predicted values of the state–variable derivatives. Thus the second–order algorithm given earlier in Eqs. (1.13) and (1.14) for conventional AM–2 is modified to the following formulas:

$$\hat{X}_{n+1} = X_n + \frac{h}{2}[3F(\hat{X}_n, U_n) - F(\hat{X}_{n-1}, U_{n-1})] \tag{3.133}$$

$$X_{n+1} = X_n + \frac{h}{2}[F(\hat{X}_n, U_n) + F(\hat{X}_{n+1}, U_{n+1})] \tag{3.134}$$

Since $F(\hat{X}_{n+1}, U_{n+1})$ is the only new derivative which must be evaluated for the nth integration frame, $F(\hat{X}_n, U_n)$ having been evaluated in the previous frame, the algorithm requires only one evaluation of the state variable derivatives per integration step. For this reason we call it a single–pass AM–2 method. It will frequently run approximately twice as fast as the conventional two–pass AM–2 algorithm. This is because the computation of the state–variable derivative functions is often the predominant element in the execution time for each integration step, as we have already observed in comparing the speed of single–pass and multiple–pass algorithms.

We next apply the single–pass AM method given by Eqs. (3.133) and (3.134) to the first–order linear system of Eq. (3.1). Taking the z transform of the resulting difference equation and solving for $X^*(z)/U^*(z)$, we obtain the following formula for the system z transform:

$$H^*(z) = \frac{\frac{h}{2}(z^3 + z^2)}{z^3 - (1 + 2\lambda h)z^2 + \frac{3}{2}\lambda h z - \frac{1}{2}\lambda h} \tag{3.135}$$

One of the three poles of $H^*(z)$ corresponds to the digital root, $\lambda^*$, which is approximately equal to the ideal characteristic root, $\lambda$. The other two poles of $H^*(z)$ correspond to extraneous roots. It turns out that the asymptotic formulas for the root errors for both real and complex $\lambda$ agree exactly with the formulas for 2nd–order implicit (trapezoidal) integration, as summarized in Sections 3.3 and 3.5. The asymptotic formulas for transfer function gain and phase errors also agree exactly with the equivalent formulas for 2nd–order implicit integration in Sections 3.3 and 3.5. Thus the single–pass AM–2 algorithm matches the accuracy of the conventional two–pass AM–2 as well as trapezoidal integration, at least for small integration step sizes.

The disadvantage of single–pass AM–2 lies with the extraneous roots and the instability they can cause if the step size is too large. The stability boundary in the

$\lambda h$ plane is obtained using the same methods employed in Section 3.4 for AB integration. Thus we set the denominator of $H^*(z)$ in Eq. (3.135) equal to zero and solve for $\lambda h$ with z equal to values on the unit circle (i.e., $|z| = 1$). In particular, for $z = -1$ we find that $\lambda h = -1/2$. The complete stability boundary for single-pass AM-2 is shown in Figure 3.6. Comparison with Figures 3.1 and 3.5 shows that the stability region for single-pass AM-2 is significantly smaller than the stability region for either AB-2 or standard (two-pass) AM-2. This is not unexpected, since single-pass AM-2 introduces two extraneous characteristic roots per state variable, as opposed to one extraneous root for AB-2 and AM-2. The instability beyond the upper boundary segment in Figure 3.6 is due to one of the extraneous roots.

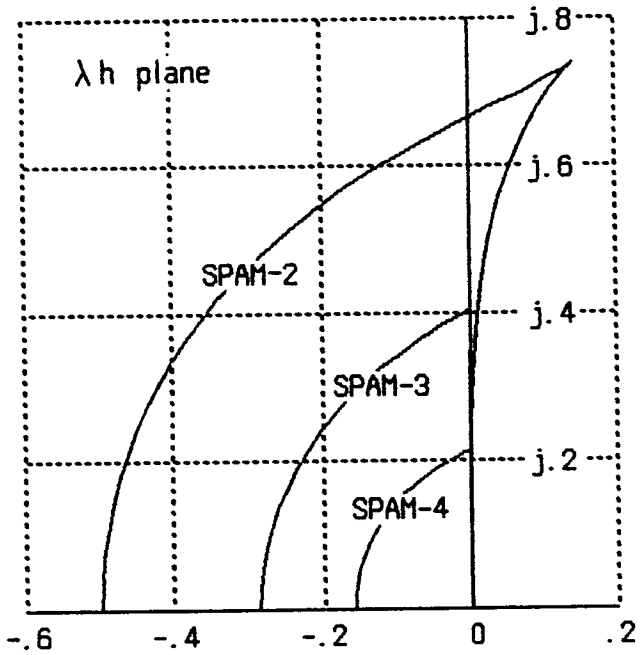



Figure 3.6. Stability boundaries for single-pass AM (SPAM) algorithms.

For third order single-pass Adams-Moulton Eqs. (1.15) and (1.16) are modified to the following form:

$$\hat{X}_{n+1} = X_n + \frac{h}{12}[23F(\hat{X}_n, U_n) - 16F(\hat{X}_{n-1}, U_{n-1} + 5F(\hat{X}_{n-2}, U_{n-2})] \quad (3.136)$$

$$X_{n+1} = X_n + \frac{h}{12}[5F(\hat{X}_{n+1}, U_{n+1}) + 8F(\hat{X}_n, U_n) - F(\hat{X}_{n-1}, U_{n-1})] \quad (3.137)$$

When we apply Eqs. (3.136) and (3.137) to the first-order system of Eq. (3.1), take

the z transform, and solve for $X^*(z)/U^*(z)$, we obtain the following formula for the system z transform:

$$H^*(z) = \frac{\frac{h}{12}(5z^4+8z^3-z)}{z^4-(1+\frac{28}{12}\lambda h)z^3+\frac{31}{12}\lambda h z^2-\frac{20}{12}\lambda h z+\frac{5}{12}\lambda h} \tag{3.138}$$

Since the denominator of $H^*(z)$ is fourth-order, whereas the system being simulated is first order, it is evident that single-pass AM-3 produces three extraneous roots per state variable. Figure 3.6 shows the stability boundary in the $\lambda h$ plane. When $\lambda$ is negative real, the system becomes unstable for $\lambda h < -2/7$, i.e., for an integration step size h greater than 2/7th the system time constant. Again it turns out that the asymptotic formulas for the characteristic root and transfer errors for single-pass AM-3 are identical with the equivalent formulas in Sections 3.3 and 3.5 for implicit 3rd-order integration and hence for conventional two-pass AM-3 integration.

From Eqs. (1.17) and (1.18) the difference equations for fourth-order Adams-Moulton integration become

$$\hat{X}_{n+1} = X_n + \frac{h}{24}[55F(\hat{X}_n, U_n) - 59F(\hat{X}_{n-1}, U_{n-1} + 37F(\hat{X}_{n-2}, U_{n-2} - 9F(\hat{X}_{n-3}, U_{n-3})] \tag{3.139}$$

$$X_{n+1} = X_n + \frac{h}{24}[9F(\hat{X}_{n+1}, U_{n+1}) + 19F(\hat{X}_n, U_n) - 5F(\hat{X}_{n-1}, U_{n-1} + F(\hat{X}_{n-2}, U_{n-2})] \tag{3.140}$$

When we apply Eqs. (3.139) and (3.140) to the first-order system given by Eq. (3.1), take the z transform, and solve for $X^*(z)/U^*(z)$, we obtain the following formula for the system z transform:

$$H^*(z) = \frac{\frac{h}{24}(9z^5+19z^4-5z^3+z)}{z^5-(1+\frac{64}{24}\lambda h)z^4+\frac{95}{24}\lambda h z^3-\frac{91}{24}\lambda h z^2+\frac{45}{24}\lambda h z-\frac{9}{24}\lambda h} \tag{3.141}$$

Since the denominator of $H^*(z)$ is fifth-order, it is evident that single-pass AM-4 introduces four extraneous roots per state variable in the system being simulated. Figure 3.6 shows the stability boundaries in the $\lambda h$ plane. When $\lambda$ is negative real, the system becomes unstable for $\lambda h < -3/19$. Once again the asymptotic formulas for single-pass AM-4 are identical with the equivalent formulas in Sections 3.3 and

3.4 for implicit 4th—order integration and hence for conventional two—pass AM—4 integration.

Comparison of the stability boundaries in Figure 3.6 for the single—pass AM algorithms with the stability boundaries in Figure 3.5 for the conventional two—pass AM algorithms shows that the single—pass algorithms have substanially smaller stability regions for algorithms of the same order. However, we must keep in mind that single—pass AM methods will usually execute twice as fast on a given digital computer as will two—pass AM methods. This in turn translates into half the mathematical step size, which means that the stability regions in Figure 3.6 should be doubled in size when making direct comparisons with the stability regions of Figure 3.5. For integration step sizes small enough to insure that the asymptotic error formulas are valid, single—pass AM methods of order 2, 3, and 4 will enjoy accuracy advantage factors of 4, 8, and 16, respectively, over conventional two—pass AM methods of the same order. Again, this results from the halved step size of single versus two—pass methods when solving differential equations dominated by complexity of the state variable derivatives.

## 3.10 Accuracy Summary for Real—Time Integration Methods

In this chapter we have developed methods for exact computation of the characteristic root errors, and the transfer function gain and phase errors, in simulating linearized versions of dynamic systems described by nonlinear differential equations. We have also developed approximate analytic formulas for the dynamic errors when the integration step size is sufficiently small. In each case the errors depend on the particular integration method being used, as well as the integration step size, h. In the case of characteristic roots, the errors vary as $|\lambda h|^k$, where k is the order of integration method and $\lambda$ is the eigenvalue. In the case of transfer functions for sinusoidal inputs of frequency $\omega$, the gain and phase errors vary as $(\omega h)^k$.

If the required accuracy of simulation is quite high, for example, dynamic errors of $10^{-4}$ or less, then the approximate asymptotic error formulas are valid. They are simple enough to be used directly to compare the dynamic performance of different integration methods, and to determine for a given method the required integration step size h to meet specific dynamic accuracy needs. On the other hand, if the dynamic accuracy requirements are more modest, for example, allowable fractional errors above $10^{-3}$, then it becomes important to assess the validity of the approximation formulas for the dynamic errors. In Figure 3.7 we attempt to do this by plotting the ratio of the exact eigenvalue error to the approximate value based on the asymptotic formulas for small step sizes. The ratio is plotted against dimensionless step size, $|\lambda h|$, for five different integration algorithms. The ratio should equal unity

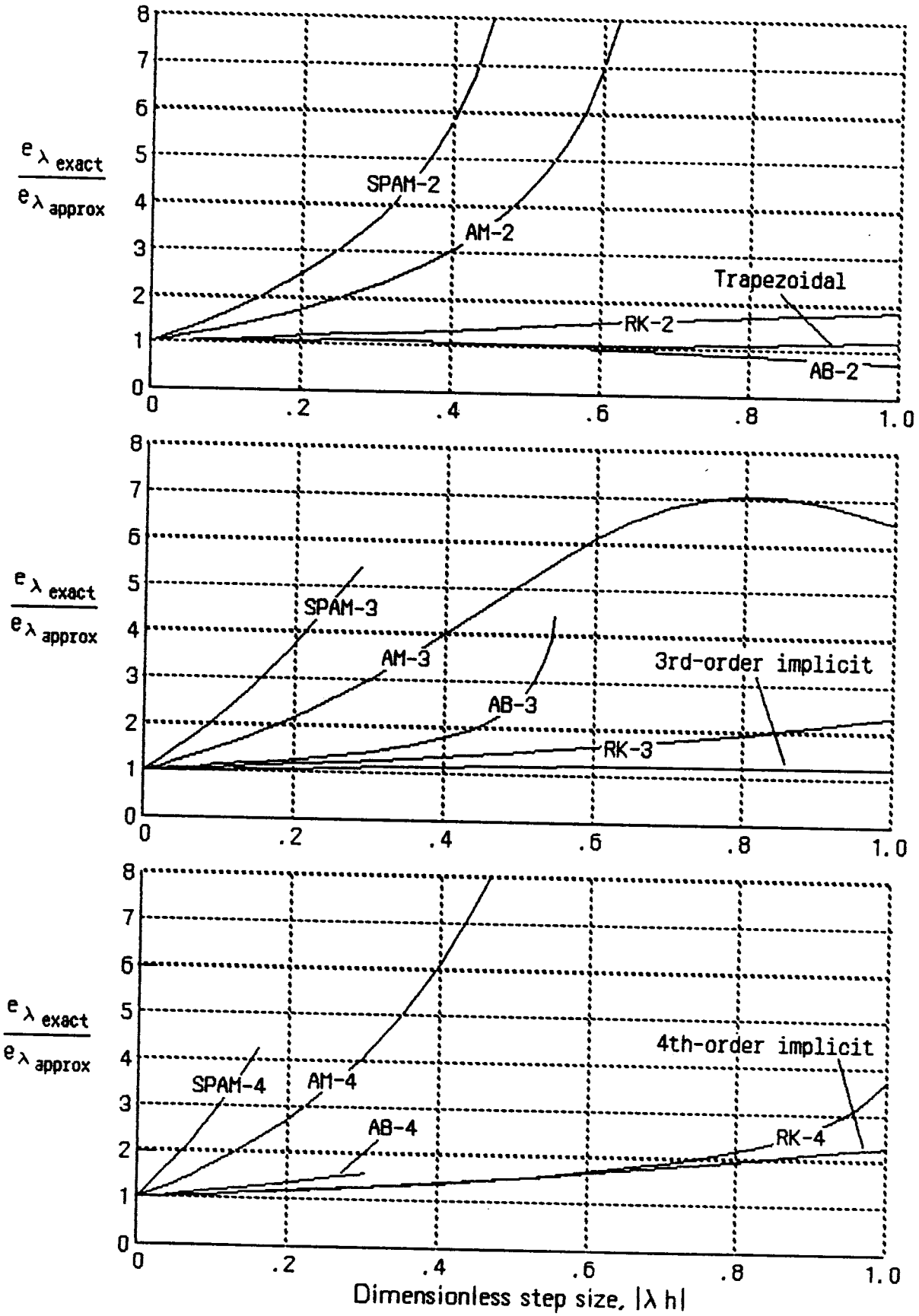



Figure 3.7. Ratio of exact to approximate characteristic root errors.

when the exact root error is equal to the value based on the asymptotic formula. In the figure it is evident that the exact root error generally exceeds the asymptotic error, especially for larger step sizes. This is particularly true in the case of the AM (Adams–Moulton) methods and the SPAM (single–pass Adams Moulton) methods. The plots in Figure 3.7 are terminated whenever the dimensionless step size $|\lambda h|$ exceeds the value beyond which the simulation becomes unstable due to an extraneous root. For example, the plot for AB–4 is not extended beyond $|\lambda h| = 3/10$. Reference to Figure 3.1 shows that instabilty results for larger step sizes, even though the fractional error in the principle root, as reflected by Figure 3.7, is equal to 1.578 times the asymptotic error, i.e., $1.578\,(251/720)(0.3)^4 = 0.00446$.

Figure 3.7 permits assessment of the validity of asymptotic formulas only in in the case of characteristic root errors. It turns out that the results for transfer function gain and phase errors are similar.*

Table 3.1 summarizes the values for the integrator error coefficient $e_I$ for all integration algorithms considered in this chapter except Runge–Kutta. All of the algorithms in Table 3.1 behave as single–pass methods in regard to the asymptotic formulas for characteristic root errors, and transfer function gain and phase errors. Thus the formulas developed in Section 3.3 for these various error measures in terms of the order k of the algorithm and the integrator error coefficient $e_I$ are valid for every method in Table 3.1. For each method the table references the numbers of the equations which give the asymptotic error formulas.

We noted in Section 3.7 that the asymptotic error formulas for the multiple–pass Runge–Kutta algorithms must be developed separately. These results are summarized in Table 3.2.

* For specific examples see R.M. Howe, "Special Considerations in Real–Time Digital Simulation," *Proc. 1983 Summer Computer Simulation Conference*, P.O. Box 2228, La Jolla, California 92038, pp 64–71.

## Table 3.1

### Error Summary for Single–Pass Integration Algorithms

$H_I^*(e^{j\omega h})$ = Integrator transfer function for sinusoidal inputs

In general, $H_I^*(e^{j\omega h}) \cong \dfrac{1}{j\omega[1 + e_I(j\omega h)^k]}$ , $\omega h \ll 1$

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -e_I(\lambda h)^k \ , \quad |\lambda h| \ll 1$$

| Integration algorithm | k | $e_I$ | Reference Eqs. for characteristic root errors | Reference Eqs. for transfer function errors |
|---|---|---|---|---|
| Euler | 1 | $\frac{1}{2}$ | (3.7), (3.16), (3.19) | (3.11–12), (3.24) |
| AB–2 | 2 | $\frac{5}{12}$ | (3.36), (3.41) | (3.39), (3.47–8) |
| AB–3 | 3 | $\frac{3}{8}$ | (3.36), (3.42) | (3.38), (3.45–6) |
| AB–4 | 4 | $\frac{251}{720}$ | (3.36), (3.43) | (3.39), (3.47–8) |
| Implicit | 2 | $-\frac{1}{12}$ | (3.36), (3.41) | (3.39), (3.47–8) |
| Implicit | 3 | $-\frac{1}{24}$ | (3.36), (3.42) | (3.38), (3.45–6) |
| Implicit | 4 | $-\frac{19}{720}$ | (3.36), (3.43) | (3.39), (3.47–8) |
| Power series | 2 | $\frac{1}{6}$ | (3.36), (3.41) | (3.39), (3.47–8) |
| Power series | 3 | $\frac{1}{24}$ | (3.36), (3.42) | (3.38), (3.45–6) |
| Power series | 4 | $\frac{1}{120}$ | (3.36), (3.43) | (3.39), (3.47–8) |
| AM–2, Single–pass AM–2 | | | Identical to Implicit, k = 2 | |
| AM–3, Single–pass AM–3 | | | Identical to Implicit, k = 3 | |
| AM–4, Single–pass AM–4 | | | Identical to Implicit, k = 4 | |

## Table 3.1

## Runge–Kutta Characteristic Root and Transfer Function Errors

First–Order System, RK–2

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{1}{6}(\lambda h)^2 \quad , \quad |\lambda h| << 1$$

$$e_M = \frac{|H^*|}{|H|} - 1 \cong -\frac{\omega^2 + 3\lambda^2}{12(\omega^2 + \lambda^2)}(\omega h)^2, \quad e_A = \angle H^* - \angle H \cong -\frac{\omega\lambda}{6(\omega^2 + \lambda^2)}(\omega h)^2$$

Second–Order System, RK–2

$$e_\omega = \frac{\omega_d^*}{\omega_d} - 1 \cong \frac{1}{6}(1 - 4\zeta^2)(\omega_n h)^2, \quad e_\zeta = \zeta^* - \zeta \cong \frac{1}{3}(\zeta - \zeta^3)(\omega_n h)^2$$

$$e_M \cong \frac{\frac{1}{12}\left[-3 + (2 - 4\zeta^2)\frac{\omega^2}{\omega_n^2} + \frac{\omega^4}{\omega_n^4}\right]}{\left[1 - \frac{\omega^2}{\omega_n^2}\right]^2 + \left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^2, \quad e_A \cong \frac{\frac{1}{3}\zeta\frac{\omega}{\omega_n}\left[1 + \frac{\omega^2}{\omega_n^2}\right]}{\left[1 - \frac{\omega^2}{\omega_n^2}\right]^2 + \left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^2$$

First–Order System, Real–Time RK–2

$$e_\lambda = \frac{\lambda^* - \lambda}{\lambda} \cong -\frac{1}{6}(\lambda h)^2 \quad , \quad |\lambda h| << 1$$

$$e_M = \frac{|H^*|}{|H|} - 1 \cong -\frac{\omega^2 + 3\lambda^2}{24(\omega^2 + \lambda^2)}(\omega h)^2, \quad e_A = \angle H^* - \angle H \cong -\frac{\omega\lambda}{6(\omega^2 + \lambda^2)}(\omega h)^2$$

Second–Order System, Real–Time RK–2

$$e_\omega = \frac{\omega_d^*}{\omega_d} - 1 \cong \frac{1}{6}(1 - 4\zeta^2)(\omega_n h)^2, \quad e_\zeta = \zeta^* - \zeta \cong \frac{1}{3}(\zeta - \zeta^3)(\omega_n h)^2$$

$$e_M \cong \frac{\frac{1}{24}\left[-3 + (2 - 4\zeta^2)\frac{\omega^2}{\omega_n^2} + \frac{\omega^4}{\omega_n^4}\right]}{\left[1 - \frac{\omega^2}{\omega_n^2}\right]^2 + \left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^2, \quad e_A \cong \frac{\frac{1}{3}\zeta\frac{\omega}{\omega_n}\left[1 + \frac{\omega^2}{\omega_n^2}\right]}{\left[1 - \frac{\omega^2}{\omega_n^2}\right]^2 + \left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^2$$

Table 3.1 (continued)

First–Order System, RK–3

$$e_\lambda \cong -\frac{1}{24}(\lambda h)^3, \quad e_M \cong -\frac{\omega\lambda - 2\frac{\lambda^3}{\omega}}{72(\omega^2+\lambda^2)}(\omega h)^3, \quad e_A \cong -\frac{\omega^2 - 8\lambda^2}{216(\omega^2+\lambda^2)}(\omega h)^3$$

Second–Order System, RK–3

$$e_\omega \cong \frac{1}{6}(2\zeta^3-\zeta)(\omega_n h)^3, \quad e_\zeta \cong \frac{1}{24}(1-\zeta^2)(1-4\zeta^2)(\omega_n h)^3$$

$$e_M \cong \frac{-\frac{1}{12}\zeta\frac{\omega}{\omega_n}\left[1+\frac{\omega^2}{\omega_n^2}\right]}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^3, \quad e_A \cong \frac{-\frac{1}{108}\left[7+5(2\zeta^2-1)\frac{\omega^2}{\omega_n^2}-2\frac{\omega^4}{\omega_n^4}\right]}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^3$$

First–Order System, RK–4

$$e_\lambda \cong -\frac{1}{120}(\lambda h)^4, \quad e_M \cong \frac{\omega^2-5\lambda^2-30\frac{\lambda^4}{\omega^2}}{2880(\omega^2+\lambda^2)}(\omega h)^4, \quad e_A \cong \frac{\omega\lambda - 5\frac{\lambda^3}{\omega}}{720(\omega^2+\lambda^2)}(\omega h)^4$$

Second–Order System, RK–4

$$e_\omega \cong -\frac{1}{120}(1-12\zeta^2+16\zeta^4)(\omega_n h)^4, \quad e_\zeta \cong -\frac{1}{30}(1-\zeta^2)(\zeta-2\zeta^3)(\omega_n h)^4$$

$$e_M \cong \frac{\frac{1}{960}\left[10\frac{\omega_n^2}{\omega^2}+40\zeta^2-5+(28\zeta^2-4)\frac{\omega^2}{\omega_n^2}-\frac{\omega^4}{\omega_n^4}\right]}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^4, \quad \omega h << 1$$

$$e_A \cong \frac{\frac{\omega_n}{240\omega}\left[5\zeta+(20\zeta^3-14\zeta)\frac{\omega^2}{\omega_n^2}+\frac{\omega^4}{\omega_n^4}\right]}{\left[1-\frac{\omega^2}{\omega_n^2}\right]^2+\left[2\zeta\frac{\omega}{\omega_n}\right]^2}(\omega h)^4, \quad \omega h << 1$$

CHAPTER 4

# DYNAMICS OF DIGITAL EXTRAPOLATION AND INTERPOLATION

## 4.1 Introduction

The estimation of future values of a time—dependent function based on current and past values of a data sequence representing the function is known as extrapolation. The estimation of function values between data points is known as interpolation. Both digital extrapolation and digital interpolation can play an important role in digital simulation of continuous dynamic systems. For example, digital interpolation can be used in the simulation of a pure time delay, often called a transport delay. Digital extrapolation can be used for predicting future values of a time—dependent function. In real time digital simulation this can be useful in compensating for delays in signals which are interfaced to the computer.

One of the most important applications of digital extrapolation and/or inter—polation occurs in the generation of a fast data sequence from a slow data sequence. The slow data sequence might represent the output from the simulation of a slow subsystem which is driving the simulation of a fast subsystem. Assume that the fast subsystem uses an integration frame rate which is an integer multiple N of the inte—gration frame rate used for the slow subsystem. From the data sequence output of the slow subsystem it is necessary to generate a data sequence which has a sample rate that is N times larger. This is necessary in order to provide inputs to the fast subsystem at its integration frame rate. The use of multiple integration frame rates within a digital simulation can often be used to improve the accuracy of a real—time digital simulation, or to speed up a non—real—time simulation in order to make it more economical to run. This is especially true in the case of many stiff systems, where the ratio of the largest to smallest eigenvalue magnitude values in the quasi—linear case is very large.

In this chapter we develop a number of digital extrapolation and interpolation algorithms and analyze their accuracy in the frequency domain in terms of gain and phase errors. Because the data sequences representing the digital signals are assumed to have a fixed sample period, the method of z transforms can be used for this analysis. The formulas for gain and phase provide a basis for selecting the most appropriate algorithm for a given application. The specific case of extrapolation or interploation to generate a fast data sequence from a slow data sequence is analyzed using a discrete

Fourier series to represent the periodic extrapolator/interpolator. When the slow data sequence is a sinusoid, this analysis provides formulas for the fundamental and harmonic components present in the fast data sequence. This information can in turn be used to estimate the dynamic errors in the fast data sequence.

## 4.2 First–Order Extrapolation based on $r_n$ and $r_{n-1}$

As a first example, let us consider digital extrapolation based on the current value, $r_n$, and the immediate past value, $r_{n-1}$, of a digital data sequence. Let h represent the time between samples. Then we can approximate the time function, r(t), represented by these two data points using the zeroth and first–order terms of a Taylor seies expansion about $t = nh$. Thus we let the approximation $\hat{r}(t)$ be given by

$$\hat{r}(t) = r_n + \frac{r_n - r_{n-1}}{h}(t - nh) \tag{4.1}$$

Here $(r_n - r_{n-1})/h$ is the numerical approximation to $\dot{r}_n$, as obtained from a backward difference. From Eq. (4.1) we see that $\hat{r}(nh) = r_n$ and $\hat{r}[(n-1)h] = r_{n-1}$, so that the approximation function $\hat{r}(t)$ passes through the data points $r_n$ and $r_{n-1}$.

Next assume that we wish to extrapolate ahead in time by a h seconds, where a represents a dimensionless extrapolation interval. From Eq. (4.1) the extrapolated time function $\hat{r}(nh + ah)$, which from now on we will designate as $\hat{r}_{n+a}$, is given by

$$\hat{r}(nh + ah) \equiv \hat{r}_{n+a} = r_n + a(r_n - r_{n-1}) \tag{4.2}$$

For $a = 1$ the prediction interval $a\,h = h$, and $\hat{r}_{n+a} = \hat{r}_{n+1}$, the extrapolated value for $r_{n+1}$ based on $r_n$ and $r_{n-1}$. Note that for $-1 < a < 0$, Eq. (4.2) represents interpolation between $r_{n-1}$ and $r_n$.

We now take the z transform of the difference equation represented by (4.2) in order to develop the formula for the extrapolator transfer function. Thus

$$R_a^*(z) = R^*(z) + a(1 - z^{-1})R^*(z) \tag{4.3}$$

where $R_a^*(z)$ is the z transform of the data sequence represented by $r_{n+a}$. The extrapolator z transform, $H_e^*(z)$, is given by

$$H_e^*(z) \equiv \frac{R_a^*(z)}{R^*(z)} = 1 + a(1 - z^{-1}) \tag{4.4}$$

The extrapolator transfer function for a sinusoidal input data sequence of frequency

$\omega$ becomes

$$H_e^*(e^{j\omega h}) = 1 + a(1 - e^{-j\omega h}) \tag{4.5}$$

An ideal extrapolator with prediction interval $ah$ has the transfer function $H_e(s) = e^{ahs}$. For sinusoidal inputs the ideal extrapolator transfer function is given by

$$H_e(j\omega) = e^{ja\omega h} \tag{4.6}$$

From Eqs. (4.5) and (4.6) we obtain the following formula for the fractional error in the digital extrapolator transfer function:

$$\frac{H_e^*(e^{j\omega h})}{H_e(j\omega)} - 1 = e^{-ja\omega h}[1 + a(1 - e^{-j\omega h})] \tag{4.7}$$

When the exponential functions in Eq. (4.7) are expanded as power series, the following formula is obtained for $H_e^*/H_e - 1$:

$$\frac{H_e^*}{H_e} - 1 = \frac{a(1+a)}{2}(\omega h)^2 + j\frac{a(1+3a+2a^2)}{6}(\omega h)^3 + \cdots \tag{4.8}$$

In general $H_e^*/H_e - 1$ in Eq. (4.8) will be a complex number which is small in magnitude when the extrapolator is reasonably accurate. Under these conditions we have shown in Eqs. (1.42) and (1.43) in Chapter 1 that the real part of $H_e^*/H_e - 1$ equals the fractional error in transfer function gain, and that the imaginary part equals the the phase error. Thus

$$\frac{|H_e^*|}{|H_e|} - 1 \equiv e_M \cong \frac{a(1+a)}{2}(\omega h)^2, \quad \omega h << 1 \tag{4.9}$$

$$\angle H_e^* - \angle H_e \equiv e_A \cong -\frac{a(1+3a+2a^2)}{6}(\omega h)^3, \quad \omega h << 1 \tag{4.10}$$

Although Eqs. (4.9) and (4.10) are approximations based on the dimensionless frequency $\omega h$ being small compared with unity, the formulas are reasonably accurate up to $\omega h = 0.5$ for $-1 < a < 1$. Of course Eq. (4.7) is exact and can be used to calcu- $e_M$ and $e_A$ for any specific $\omega h$ and $a$. From the asymptotic formulas in Eqs. (4.9) and (4.10) we note that the extrapolator gain error is proportional to $(\omega h)^2$ and the phase error is proportional to $(\omega h)^3$. Thus the predominant error will be a gain error for extrapolation based on $r_n$ and $r_{n-1}$. Figures 4.1 and 4.2 show exact plots of $e_M$ and $e_A$ versus $\omega h$ for $a = 0.5$ for a number of extrapolation algorithms, including the one we have just considered here.

From the extrapolator gain and phase errors, $e_M$ and $e_A$, we can estimate the effect of extrapolator performance as part of a digital simulation. In general, these frequency—domain results provide more useful error measures than results based on errors in the time domain for specific time—dependent functions, r(t).

## 4.3 First—Order Extrapolation from $r_n$, $\dot{r}_n$

Another method for first—order extrapolation is based on using $r_n$ and $\dot{r}_n$ instead of $r_n$ and $r_{n-1}$. This can always be done if $r_n$ is a state variable, since under these conditions the time derivative $\dot{r}_n$ is available from the state equation. In this case the extrapolation formula becomes

$$\hat{r}(t) = r_n + \dot{r}_n(t - nh) \tag{4.11}$$

From Eq. (4.11) it is clear that $\hat{r}(nh) = r_n$ and $\dot{\hat{r}}(nh) = \dot{r}_n$. If we let $t - nh = ah$, the prediction interval, then from Eq. (4.11) we obtain the following formula for the for the extrapolated data point, $\hat{r}_{n+a}$:

$$\hat{r}_{n+a} = r_n + ah\dot{r}_n \tag{4.12}$$

Taking the z transform of Eq. (4.12), we have

$$R_a^*(z) = R^*(z) + ah\dot{R}^*(z) \tag{4.13}$$

For a sinusoidal data sequence with $r_n = e^{j\omega nh}$, it follows that $\dot{r}_n = j\omega e^{j\omega nh} = j\omega r_n$ and Eq. (4.13) becomes

$$R_a^*(e^{j\omega h}) = (1 + ja\omega h)R^*(e^{j\omega h}) \tag{4.14}$$

Thus the digital extrapolator transfer function for sinusoidal inputs is given by

$$\frac{R_a^*(e^{j\omega h})}{R^*(e^{j\omega h})} \equiv H_e^*(e^{j\omega h}) = 1 + ja\omega h \tag{4.15}$$

Dividing $H_e^*$ in Eq. (4.15) by the ideal extrapolator transfer function $H_e$ of Eq. (4.6), we obtain the following formula for the fractional error in the extrapolator transfer function:

$$\frac{H_e^*(e^{j\omega h})}{H_e(j\omega)} - 1 = e^{-ja\omega h}(1 + ja\omega h) \tag{4.16}$$

Expanding the exponential function on the right side of Eq. (4.16) in a power series,

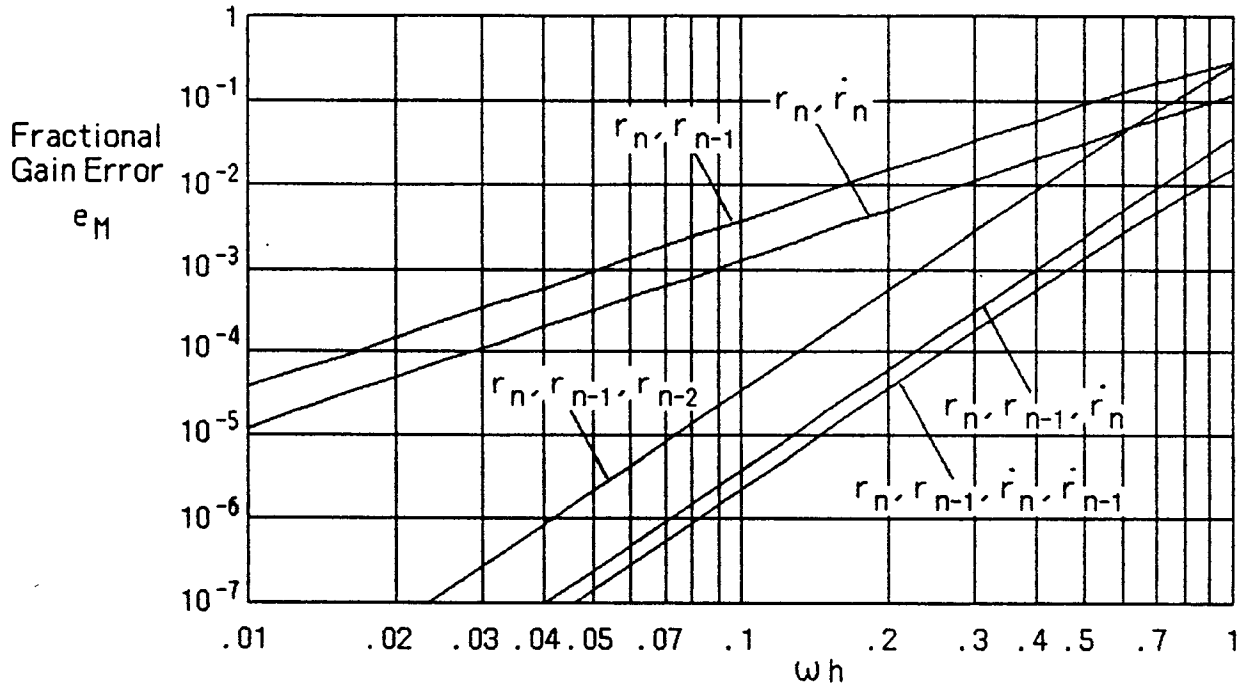



Figure 4.1. Fractional gain error versus demonsionless frequency for extrapolator algorithms; h = sample period; extrapolation interval a h = h/2.

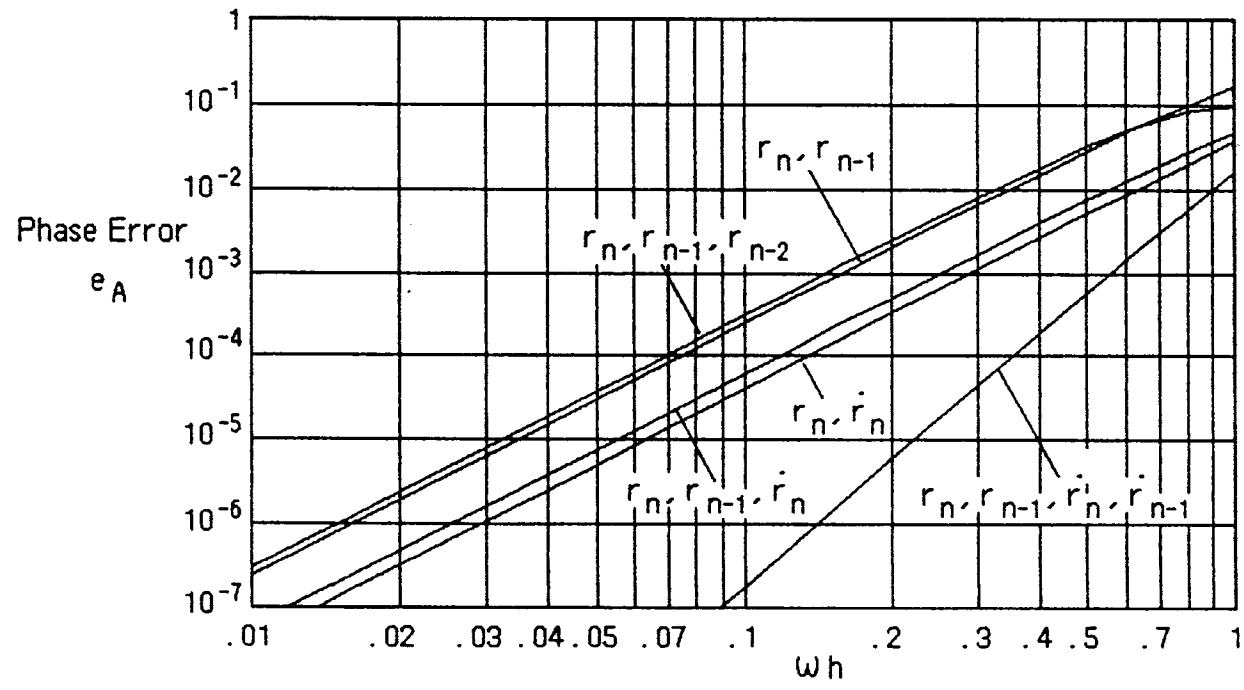



Figure 4.2. Phase error versus dimensionless frequency for extrapolator algorithms; h = sample period; extrapolation interval a h = h/2.

we obtain the formula

$$\frac{H_e^*}{H_e} - 1 = \frac{a^2}{2}(\omega h)^2 - j\frac{a^3}{3}(\omega h)^3 + \cdots \tag{4.17}$$

It follows that the asymptotic formulas for the fractional error in gain and the phase error are given by

$$\frac{|H_e^*|}{|H_e|} - 1 \equiv e_M \cong \frac{a^2}{2}(\omega h)^2, \quad \omega h << 1 \tag{4.18}$$

$$\angle H_e^* - \angle H_e \equiv e_A \cong -\frac{a^3}{3}(\omega h)^3, \quad \omega h << 1 \tag{4.19}$$

From Eqs. (4.18) and (4.19) we see that here again the first–order extrapolator gain error is proportional to $(\omega h)^2$ while the phase error is proportional to $(\omega h)^3$. Thus the gain error will predominate. Exact plots of $e_M$ and $e_A$ versus $\omega h$ for the $r_n, \dot{r}_n$ extrapolation algorithm considered here are shown in Figures 4.1 and 4.2 for $a = 0.5$. These plots plus a comparison of Eqs. (4.18) and (4.19) with Eqs. (4.9) and (4.10) show that extrapolation based on $r_n, \dot{r}_n$ is significantly more accurate than extrapolation based on $r_n, r_{n-1}$. This is further confirmed by Figure 4.3, which shows plots of the gain error coefficient, $e_M/(\omega h)^2$, versus dimensionless extrapolation interval a, where $0 < a < 1$ represents extrapolation and $-1 < a < 0$ represents interpolation.

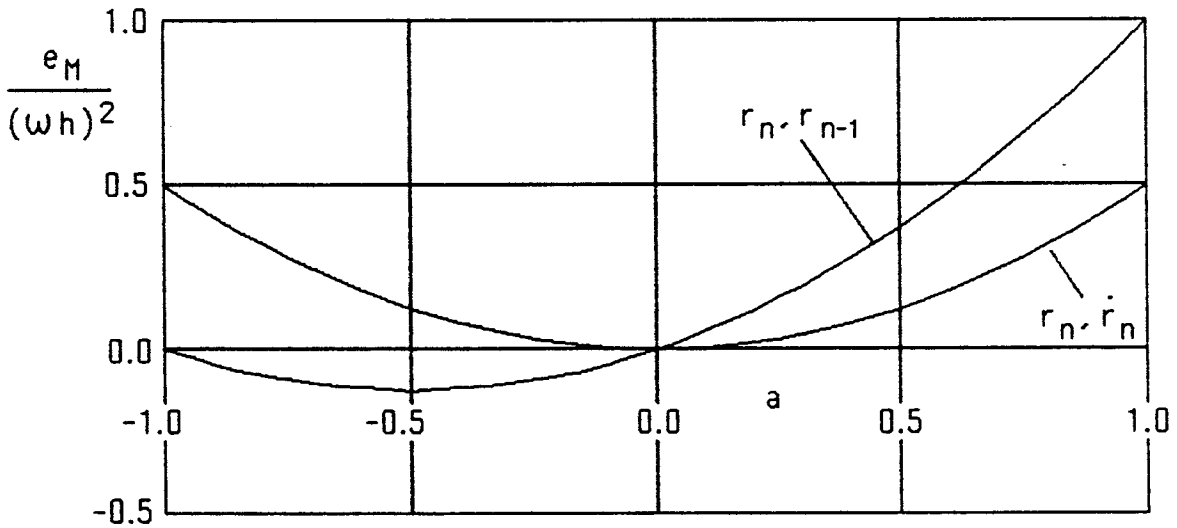



Figure 4.3. Gain error coefficient, $e_M/(\omega h)^2$, versus dimensionless extrapolation interval, a, for first–order extrapolation algorithms.

Figure 4.3 shows that whereas the $r_n, \dot{r}_n$ algorithm is superior for extrapolation, the $r_n, r_{n-1}$ algorithm is superior for interpolation.

It should be noted that both $r_n, r_{n-1}$ extrapolation and $r_n, \dot{r}_n$ extrapolation exhibit gains appreciably greater than unity when $\omega h$ exceeds unity. From the exact formula for $H_e^*$ in Eq. (4.5) it can be shown that the gain $|H_e^*|$ for a given prediction interval, a, peaks for $r_n, r_{n-1}$ extrapolation at the frequency given by $\omega h = \pi$. The corresponding peak gain is equal to $1 + 2a$. For the extrapolation based on $r_n, \dot{r}_n$ it is apparent from Eq. (4.15) that the gain is equal to $[1 + (a\omega h)^2]^{1/2}$. Ideally, the extrapolator gain should be equal to unity for all frequencies. In general, one does not expect significant frequency components above $\omega h = 1$ in the data sequence corresponding to $r_n$. But any such components can get amplified appreciably with the first-order extrapolation algorithms considered in this section.

## 4.4 Second-Order Extrapolation Based on $r_n, r_{n-1}, r_{n-2}$

In this and the following two sections we analyze several second-order extrapolation algorithms. First we consider extrapolation based on the data points $r_n, r_{n-1}$ and $r_{n-2}$. If a quadratic function $\hat{r}(t)$ is passed through these three points, the following formula results:

$$\hat{r}(t) = r_n + \frac{3r_n - 4r_{n-1} + r_{n-2}}{2h}(t - nh) + \frac{r_n - 2r_{n-1} + r_{n-2}}{2h^2}(t - nh)^2 \qquad (4.20)$$

Here the coefficient of $(t - nh)$ is a backward difference approximation to $\dot{r}_n$ and the coefficient of $(t - nh)^2$ is a backward difference approximation to $\ddot{r}_n$. If we substitute $t = nh + ah$ into Eq. (4.20), where $ah$ is the prediction interval, we obtain the extrapolation formula in terms of the dimensionless prediction interval a. Thus

$$\hat{r}_{n+a} = \frac{2 + 3a + a^2}{2} r_n - (2a + a^2) r_{n-1} + \frac{a + a^2}{2} r_{n-2} \qquad (4.21)$$

Note that for $-2 < a < 0$ Eq. (4.21) represents quadratic interpolation between $r_{n-2}$, $r_{n-1}$ and $r_n$.

Following the procedure used in Section 4.2, we take the z transform of Eq. (4.21) and let $z = e^{j\omega h}$ to obtain $H_e^*(e^{j\omega h})$, the extrapolator transfer function for sinusoidal input data sequences. Next we divide by $e^{ja\omega h}$, the ideal extrapolator transfer function $H_e(j\omega)$. In this way we obtain the following formula for the ratio $H_e^*/H_e$:

$$\frac{H_e^*}{H_e} = e^{-ja\omega h}\left[\frac{2 + 3a + a^2}{2} - (2a + a^2)e^{-j\omega h} + \frac{a + a}{2} e^{-j2\omega h}\right] \qquad (4.22)$$

Using power series expansions for the exponential functions in Eq. (4.22) and retaining terms to order $h^4$, we obtain the following approximate expression for the fractional error in extrapolator transfer function:

$$\frac{H_e^*}{H_e} - 1 \cong \frac{a(2+5a+4a^2+a^3)}{8}(\omega h)^4 + j\frac{a(2+3a+a^2)}{6}(\omega h)^3, \quad \omega h << 1$$

It follows that the asymptotic formulas for the fractional error in gain and the phase error are given by

$$\frac{|H_e^*|}{|H_e|} - 1 \equiv e_M \cong \frac{a}{8}(2+5a+4a^2+a^3)(\omega h)^4, \quad \omega h << 1 \tag{4.23}$$

$$\angle H_e^* - \angle H_e \equiv e_A \cong \frac{a}{6}(2+3a+a^2)(\omega h)^3, \quad \omega h << 1 \tag{4.24}$$

Here the phase error predominates for small $\omega h$, varying as $(\omega h)^3$; the fractional gain error is proportional to $(\omega h)^4$. For $a = 0.5$, Figures 4.1 and 4.2 show exact plots of $e_M$ and $e_A$ versus $\omega h$ in comparison with other extrapolation schemes.

## 4.5 Second–Order Extrapolation Based on $r_n, r_{n-1}, \dot{r}_n$

Next we consider extrapolation based on $r_n, r_{n-1}$ and $\dot{r}_n$. If $r_n$ is a state variable in the simulation, then $\dot{r}_n$ is available. Here the extrapolation is based on a quadratic which passes through $r_n$ and $r_{n-1}$ and matches the slope $\dot{r}_n$ at $t = nh$. Thus we obtain

$$\hat{r}_n(t) = r_n + \dot{r}_n(t - nh) + \frac{\dot{r}_n h - r_n + r_{n-1}}{h^2}(t - nh)^2 \tag{4.25}$$

Letting $t = nh + ah$ in Eq. (4.25), we obtain the following extrapolation formula:

$$\hat{r}_{n+a} = (1 - a^2)r_n + a^2 r_{n-1} + (a + a^2)h\dot{r}_n \tag{4.26}$$

For $-1 < a < 0$ Eq. (4.26) represents quadratic interpolation between $r_n$ and $r_{n-1}$. To obtain the fractional error in extrapolator transfer function we follow the same procedure used in Section 4.3 for the $r_n, \dot{r}_n$ extrapolator. Thus we take the z transform of Eq. (4.26), rewrite it for a sinusoidal input data sequence, and then solve for the extrapolator transfer function. In this way we obtain

$$H_e^*(e^{j\omega h}) = 1 - a^2 + a^2 e^{-j\omega h} + j(a + a^2)\omega h \tag{4.27}$$

Dividing by the ideal extrapolator transfer function, $H_e(j\omega) = e^{ja\omega h}$, we obtain the following formula for the ratio $H_e^*/H_e$:

$$\frac{H_e^*(e^{j\omega h})}{H_e(j\omega)} = e^{-ja\omega h}[1 - a^2 + a^2 e^{-j\omega h} + j(a + a^2)\omega h] \tag{4.28}$$

Expanding the exponential functions in power series, we obtain the formula for the the fractional error in extrapolator transfer function in series form. Thus

$$\frac{H_e^*}{H_e} - 1 = \frac{a^2}{24}(1 + 4a + 3a^2)(\omega h)^4 + j\frac{a^2}{6}(1 + a)(\omega h)^3 + \cdots$$

It follows that the asymptotic formulas for the fractional error in gain and the phase error are given by

$$\frac{|H_e^*|}{|H_e|} - 1 \equiv e_M \cong \frac{a^2}{24}(1 + 4a + 3a^2)(\omega h)^4, \quad \omega h << 1 \tag{4.29}$$

$$\angle H_e^* - \angle H_e \equiv e_A \cong \frac{a^2}{6}(1 + a)(\omega h)^3, \quad \omega h << 1 \tag{4.30}$$

Again the phase error predominates for small $\omega h$, varying as $(\omega h)^3$; the fractional gain error is proportional to $(\omega h)^4$. For $a = 0.5$, Figures 4.1 and 4.2 show exact plots of $e_M$ and $e_A$ versus $\omega h$ in comparison with other extrapolation schemes.

## 4.6. Second–Order Extrapolation Based on $r_n, \dot{r}_n, \dot{r}_{n-1}$

As in the previous section, this method is limited to cases where the derivatives of the data points are available or can be computed. This will always be true if $r_n$ is a state variable. In fact, for both predictor and predictor–corrector algorithms $\dot{r}_n$ and $\dot{r}_{n-1}$ will both be available in storage at the nth frame. Here the extrapolation formula is based on a quadratic which passes through the point $r_n$ and matches the slopes $\dot{r}_n$ and $\dot{r}_{n-1}$. The formula is given by

$$\hat{r}_n(t) = r_n + \dot{r}_n(t - nh) + \frac{\dot{r}_n - \dot{r}_{n-1}}{2h}(t - nh)^2 \tag{4.31}$$

Letting $t = nh + ah$ in Eq. (4.31), we obtain the following extrapolation formula:

$$\hat{r}_{n+a} = r_n + a\dot{r}_n h + \frac{a^2}{2}(\dot{r}_n - \dot{r}_{n-1})h \tag{4.32}$$

For $-1 < a < 0$ Eq. (4.32) represents quadratic interpolation between $t = (n-1)h$ and

Taking the z transform of Eq. (4.32) and rewriting it for a sinusoidal input data sequence, we can solve for the extrapolator transfer function. When this is then divided by the ideal extrapolator transfer function, $H_e(j\omega) = e^{ja\omega h}$, we obtain the following formula for the ratio $H_e^*/H_e$:

$$\frac{H_e^*(e^{j\omega h})}{H_e(j\omega)} = e^{-ja\omega h}[1 + ja\omega h + j\frac{a^2\omega h}{2}(1 - e^{-j\omega h})] \qquad (4.33)$$

Expanding the exponential functions in power series, we obtain the formula for the the fractional error in extrapolator transfer function in series form. Thus

$$\frac{H_e^*}{H_e} - 1 = \frac{a^2}{24}(2 + 6a + 3a^2)(\omega h)^4 + j\frac{a^2}{12}(3 + 2a)(\omega h)^3 + \cdots$$

It follows that the asymptotic formulas for the fractional error in gain and the phase error are given by

$$\frac{|H_e^*|}{|H_e|} - 1 \equiv e_M \cong \frac{a^2}{24}(2 + 6a + 3a^2)(\omega h)^4, \quad \omega h << 1 \qquad (4.34)$$

$$\angle H_e^* - \angle H_e \equiv e_A \cong \frac{a^2}{6}(3 + 2a)(\omega h)^3, \quad \omega h << 1 \qquad (4.35)$$

As in the case of the previous two second–order extrapolation algorithms, the phase error predominates here for small $\omega h$, varying as $(\omega h)^3$; the fractional gain error is proportional to $(\omega h)^4$. For $a = 0.5$, Figures 4.1 and 4.2 show exact plots of $e_M$ and $e_A$ versus $\omega h$ in comparison with other extrapolation algorithms.

The error coefficient, $e_A/(\omega h)^3$, is shown in Figure 4.4 as a function of the extrapolation interval $a$ for the three second–order algorithms considered here. It is again evident that the algorithms are more accurate for interpolation $(-1 < a < 0)$ than for extrapolation $(0 < a < 1)$. The algorithm based on $r_n$, $r_{n-1}$ and $\dot{r}_n$ is clearly the most accurate, whereas the algorithm based on $r_n$, $r_{n-1}$ and $r_{n-2}$ is generally the least accurate.

From the exact formula for $H_e^*$ in the case of $r_n$, $r_{n-1}$, $r_{n-1}$ extrapolation it can be shown that the gain $|H_e^*|$ peaks at the frequency given by $\omega h = \pi$ for $a > 0$. The corresponding peak gain is equal to $1 + 4a + 2a^2$. For extrapolation based on $r_n$, $r_{n-1}$, $\dot{r}_n$, as well as extrapolation based on $r_n$, $\dot{r}_n$, $\dot{r}_{n-1}$, the gain is less than this for $\omega h = \pi$, although it continues to increase for $\omega h > \pi$. In the unlikely event that the data sequence corresponding to $r_n$ has significant frequency components for which $\omega h > 1$, the second–order extrapolation algorithms considered here may amplify such components appreciably.

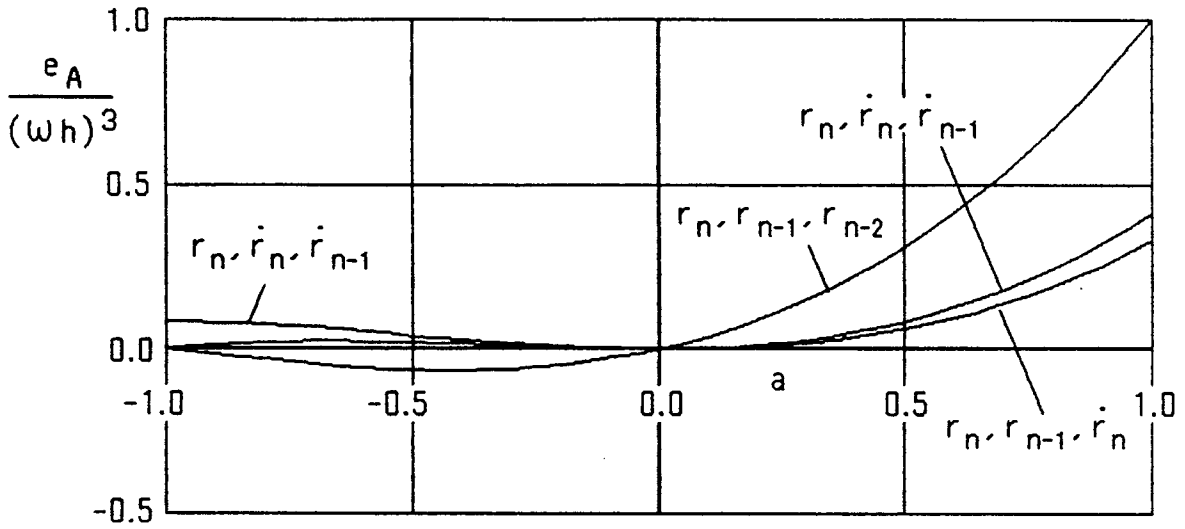



Figure 4.4. Phase error coefficient, $e_A/(\omega h)^3$, versus dimensionless extrapolation interval, a, for second–order extrapolation algorithms.

## 4.7. Third–Order Extrapolation Based on $r_n, \dot{r}_n, r_{n-1}, \dot{r}_{n-1}$

Although there are a number of possible third–order extrapolation algorithms, in this section we will consider only one, which is based on a cubic function that passes through the data points $r_n$ and $r_{n-1}$, and matches the slopes $\dot{r}_n$ and $\dot{r}_{n-1}$. In this case the formula for the extrapolated data point $\hat{r}_{n+a}$ is given by

$$\hat{r}_{n+a} = (1-3a^2-2a^3)r_n + (3a^2+2a^3)r_{n-1} + ah(1+a)^2\dot{r}_n + a^2h(1+a)\dot{r}_{n-1} \tag{4.36}$$

If $r_n$ is a state variable in the simulation, then $\dot{r}_n$ and $\dot{r}_{n-1}$ are available for the algorithm. For $-1 < a < 0$, Eq. (4.36) represents Hermite interpolation.* Following the same procedure used in Section 4.3 for $r_n, \dot{r}_n$ extrapolation, we can derive the digital transfer function for sinusoidal inputs, $H_e^*(e^{j\omega h})$. After dividing by the ideal extrapolator transfer function, given by $H_e(j\omega) = e^{ja\omega h}$, we obtain the following formula for the ratio $H_e^*/H_e$:

$$\frac{H_e^*}{H_e} = e^{-ja\omega h}[1+(3a^2+2a^3)(e^{-j\omega h}-1) + ja\omega h(1+a)(1+a+ae^{-j\omega h})] \tag{4.37}$$

The asymptotic formula for the fractional error in extrapolator transfer function is

* See, for example, Z. Kopal, *Numerical Analysis*, John Wiley and Sons, Inc., New York, 1981.

obtained by subtracting 1 from the right side of Eq. (4.37) and replacing the exponential functions with power series. Retaining terms up to order $h^5$, we obtain

$$\frac{H_e^*}{H_e} - 1 = -\frac{a(1+a)^2}{24}(\omega h)^4 + j\frac{a^2(1+a)^2(1+2a)}{60}(\omega h)^5 + \cdots$$

It follows that the asymptotic formulas for the fractional error in gain and the phase error are given by

$$\frac{|H_e^*|}{|H_e|} - 1 \equiv e_M \cong -\frac{a(1+a)^2}{24}(\omega h)^4, \quad \omega h << 1 \tag{4.38}$$

$$\angle H_e^* - \angle H_e \equiv e_A \cong \frac{a^2(1+a)^2(1+2a)}{60}(\omega h)^5, \quad \omega h << 1 \tag{4.39}$$

For this third-order extrapolation algorithm, the predominant error is the gain error, which is proportional to $(\omega h)^4$. Figures 4.1 and 4.2 show exact plots of $e_M$ and $e_A$ versus $\omega h$ in comparison with the other extrapolation algorithms considered in this chapter. Figure 4.4.5 shows a plot of the error coefficient $e_M/(\omega h)^4$ as a function of the dimensionless extrapolation interval a. In the case of this third-order algorithm the difference between the interpolation errors $(-1 < a < 0)$ and the extrapo-

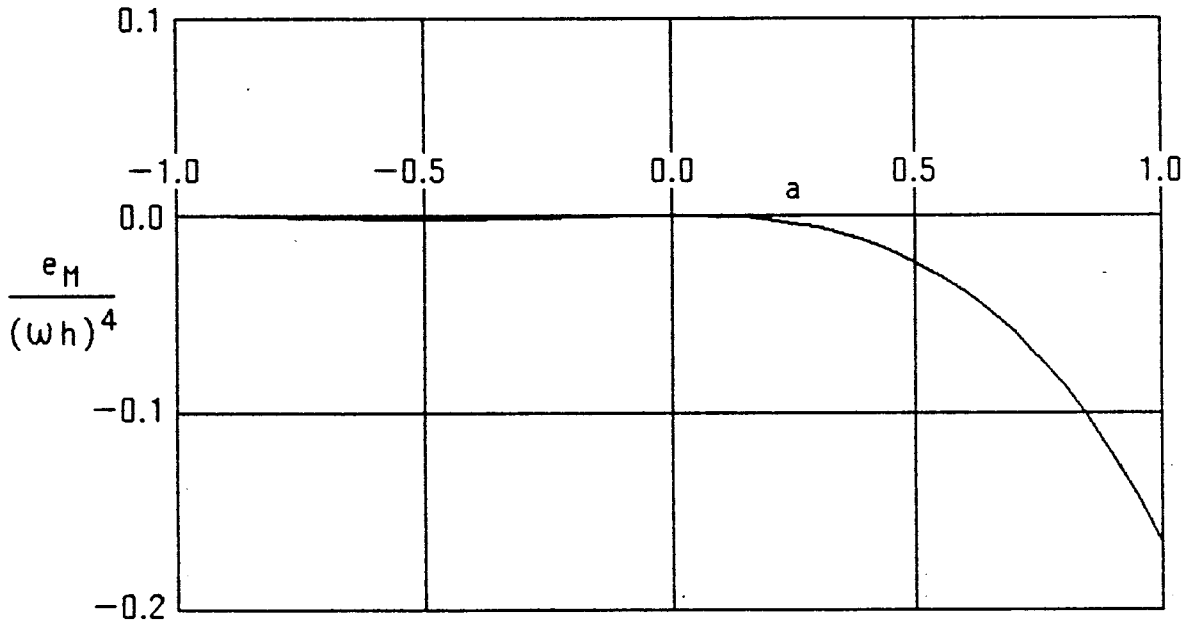



Figure 4.5. Gain error coefficient, $e_M/(\omega h)^4$, versus dimensionless extrapolation interval, a, for third-order algorithm.

polation errors ($0 < a < 1$) is very substantial. Thus the peak magnitude of gain error coefficient for interpolation in Figure 4.5, occurring at $a = -0.5$, is equal to 1/384. By comparison, the peak magnitude of gain error coefficient for extrapolation, which occurs at $a = 1$, is equal to 1/6.

From the exact formula for $H_e^*$ the transfer function gain $|H_e^*|$ can be determined for the case where the input frequency $\omega h$ is not small compared with unity. For $a = 1$ and $\omega h > 1$, the gain increases rapidly with increasing $\omega h$, e.g., the gain equals 10.976 for $\omega h = \pi$. Since the gain for $a < 1$ is roughly a function of $a^2$, it will be considerably less for smaller prediction intervals a. In the unlikely event that the data sequence corresponding to $r_n$ has significant frequency components for which $\omega h > 1$, the second-order extrapolation algorithms considered here may amplify such components appreciably.

## 4.8. Summary Tables of Extrapolation Formulas

The extrapolation formulas developed in Sections 4.2 through 4.7 are summarized in Table 4.1. Table 4.2 presents the asymptotic formulas for gain and phase errors of the corresponding extrapolator transfer functions.

**Table 4.1**

**Summary of Extrapolator Formulas**

$\hat{r}_{n+a} \equiv \hat{r}(nh + ah)$, where $ah$ = extrapolation interval

| Extrapolator Inputs | Extrapolator Formula for $\hat{r}_{n+a}$ |
|---|---|
| $r_n, r_{n-1}$ | $r_n + a(r_n - r_{n-1})$ |
| $r_n, \dot{r}_n$ | $r_n + ah\dot{r}_n$ |
| $r_n, r_{n-1}, r_{n-2}$ | $\frac{2 + 3a + a^2}{2} r_n - (2a + a^2) r_{n-1} + \frac{a + a^2}{2} r_{n-2}$ |
| $r_n, r_{n-1}, \dot{r}_n$ | $(1 - a^2) r_n + a^2 r_{n-1} + (a + a^2) h\dot{r}_n$ |
| $r_n, \dot{r}_n, \dot{r}_{n-1}$ | $r_n + a\dot{r}_n h + \frac{a^2}{2}(\dot{r}_n - \dot{r}_{n-1})h$ |
| $r_n, r_{n-1}, \dot{r}_n, \dot{r}_{n-1}$ | $(1 - 3a^2 - 2a^3) r_n + (3a^2 + 2a^3) r_{n-1} + ah(1 + a)^2 \dot{r}_n + a^2 h(1 + a)\dot{r}_{n-1}$ |

Table 4.2

Summary of Extrapolator Transfer Function Gain and Phase Errors

Note: all formulas are approximate based on $\omega h << 1$.

| Extrapolator Inputs | Fractional Gain Error, $e_M$ | Phase Error, $e_A$ |
|---|---|---|
| $r_n, r_{n-1}$ | $\frac{a(1+a)}{2}(\omega h)^2$ | $-\frac{a(1+3a+2a^2)}{6}(\omega h)^3$ |
| $r_n, \dot{r}_n$ | $\frac{a^2}{2}(\omega h)^2$ | $-\frac{a^3}{3}(\omega h)^3$ |
| $r_n, r_{n-1}, r_{n-2}$ | $\frac{a}{8}(2+5a+4a^2+a^3)(\omega h)^4$ | $\frac{a}{6}(2+3a+a^2)(\omega h)^3$ |
| $r_n, r_{n-1}, \dot{r}_n$ | $\frac{a^2}{24}(1+4a+3a^2)(\omega h)^4$ | $\frac{a^2}{6}(1+a)(\omega h)^3$ |
| $r_n, \dot{r}_n, \dot{r}_{n-1}$ | $\frac{a^2}{24}(2+6a+3a^2)(\omega h)^4$ | $\frac{a^2}{6}(3+2a)(\omega h)^3$ |
| $r_n, r_{n-1}, \dot{r}_n, \dot{r}_{n-1}$ | $-\frac{a^2(1+a)^2}{24}(\omega h)^4$ | $\frac{a^2(1+a)^2(1+2a)}{60}(\omega h)^5$ |

## 4.9. Extrapolation and Interpolation for Multiple Frame Rates

Consider digital simulation of a dynamic system where the integration step size is T. Assume that a fast subsystem within the simulation uses an integration frame-rate which is N time the basic frame-rate 1/T, so that the step size, h, for the fast subsystem is T/N. Let $\{r_n\}$ be a data sequence with sample period T which serves as an input to the fast subsystem, and assume that the fast-subsystem integration algorithm requires inputs $r_n, r_{n+1/N}, r_{n+2/N}, \cdots, r_{n+(N-1)/N}$ over each sample period T. The only way these data sequence values can be calculated from $r_n, r_{n-1}, r_{n-2}, \cdots,$ is through extrapolation. If $r_n$ is a state variable, then $\dot{r}_n, \dot{r}_{n-1}, \ldots,$ are also available for extrapolation. If $r_n$ is a state variable and $\dot{r}$ is integrated during the nth frame to obtain $r_{n+1}$ before the fast subsystem integration is mechanized, then $r_{n+1}$ is available as well as $r_n, r_{n-1}, \ldots, \dot{r}_n, \ldots$. In this case interpolation can be used to compute $r_n, r_{n+1/N}, r_{n+2/N}, \cdots$.

In Sections 4.2 through 4.7 we developed a number extrapolation formulas that are summarized in Table 4.1. We also developed asymptotic formulas for the ex-

trapolator transfer function gain and phase errors. These are summarized in Table 4.2. Here we will take advantage of these formulas in analyzing methods for providing extrapolated or interpolated inputs for multiple frame-rate digital simulations.

Figure 4.6 shows the extrapolation (or interpolation) process. The data sequence $\{r_n\}$ with a sample period T is generated by the slow simulation using an integration step size equal to T. The extrapolator, by means of the appropriate difference equation, generates from $\{r_n\}$ (and possibly $\{\dot{r}_n\}$) a data sequence $\{f_k\}$ which has a sample period $h = T/N$, $N = 2, 3, 4, \ldots$ . Then $\{f_k\}$ becomes the input to the fast-subsystem simulation, where $h = T/N$ is the integration step size.

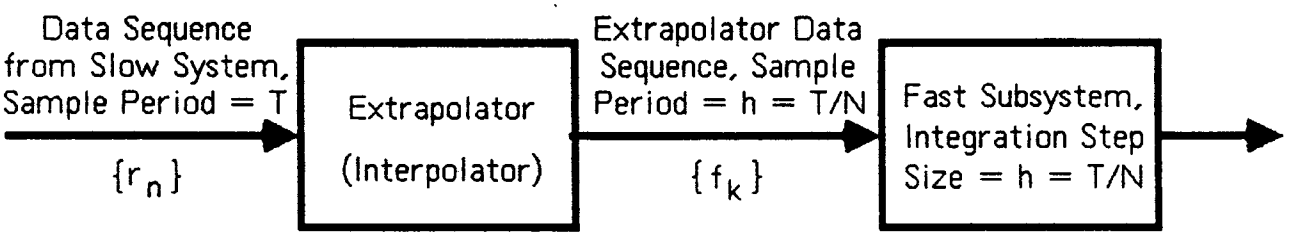



Figure 4.6. Extrapolator (or interpolator) for generating multiple frame-rate input to fast subsystem simulation.

For example, if $N = 4$ and we use first-order extrapolation, then we need to compute estimates for $r_{n+1/4}$, $r_{n+1/2}$ and $r_{n+3/4}$ from $r_n$ and $r_{n-1}$. These estimates, along with $r_n$, constitute 4 data points in the $\{f_k\}$ data sequence with sample period $h = T/4$. In Eq. (4.2) for first-order extrapolation based on $r_n$ and $r_{n-1}$, we let $a = 1/4$, $1/2$, and $3/4$, which leads to the following equations for the fast data sequence $\{f_k\}$:

$$
\begin{aligned}
f_k &= r_n \qquad , \; k = 4n \\
f_{k+1} &= r_n + \tfrac{1}{4}(r_n - r_{n-1}) \\
f_{k+2} &= r_n + \tfrac{2}{4}(r_n - r_{n-1}) \\
f_{k+3} &= r_n + \tfrac{3}{4}(r_n - r_{n-1}) \\
f_{k+4} &= r_{n+1} \\
f_{k+5} &= r_{n+1} + \tfrac{1}{4}(r_{n+1} - r_n) \\
f_{k+6} &= r_{n+1} + \tfrac{2}{4}(r_{n+1} - r_n) \\
&\vdots \qquad \vdots \qquad \vdots
\end{aligned}
\tag{4.40}
$$

Let us now write the formulas for the fast data sequence points for the general case where the frame multiple is N and $\hat{r}_{n+a}$ designates the extrapolated data point with extrapolation interval aT. Then the equations become

$$\begin{aligned} f_k &= \hat{r}_{n+0} \quad , \quad k = Nn \\ f_{k+1} &= \hat{r}_{n+1/N} \\ f_{k+2} &= \hat{r}_{n+2/N} \\ &\vdots \\ f_{k+N-1} &= \hat{r}_{(N-1)/N} \\ f_{k+N} &= \hat{r}_{n+1+0} \\ f_{k+N+1} &= \hat{r}_{N+1+1/N} \\ &\vdots \end{aligned} \tag{4.41}$$

It follows that

$$f_{k+m} = \hat{r}_{n+m/N} \quad , \quad m = 0,1,2, \ldots, N-1 \tag{4.42}$$

We next consider a sinusoudal input data sequence from the slow system. Thus we let

$$r_n = e^{j\omega nT} \tag{4.43}$$

The extrapolator response $\hat{r}_{n+a}$ to the sinusoidal input $r_n$ in Eq. (4.43) is simply

$$\hat{r}_{n+a} = H^*_{ea}\, e^{j\omega nT} \tag{4.44}$$

where $H^*_{ea}$ represents the extrapolator transfer function for sinusoidal inputs for the extrapolation interval a. Here we have designated this transfer function $H^*_{ea}$ rather than simply $H^*_e$ in order to indicate its dependence on the dimensionless extrapolation interval a. Substituting Eq. (4.44) into Eq. (4.42), we obtain

$$f_{k+m} = H^*_{em/N}\, e^{j\omega nT}, \quad m = 0,1,2, \ldots, N-1 \tag{4.45}$$

We note that

$$e^{j\omega nT} = e^{j\omega (n+a)T}\, e^{-j\omega aT} = \frac{e^{j\omega (n+a)T}}{H_{ea}(j\omega)} \tag{4.46}$$

Here $H_{ea}(j\omega) = e^{ja\omega T}$ is the ideal transfer function for the extrapolation interval a, first introduced as $H_e(j\omega)$ in Eq. (4.6), but now designated as $H_{ea}$ to indi-

cate its dependence on the dimensionless extrapolation interval a. Replacing T by Nh, n by k/N, and a by m/N in Eq. (4.46), we see that (n+a)T = (k+m)h. When Eq. (3.46) is then substituted into Eq. (4.45), we obtain

$$f_{k+m} = \frac{H^*_{em/N}}{H_{em/N}} e^{j\omega(k+m)h} \quad , \quad m = 0,1,2, \ldots, N-1 \tag{4.47}$$

For all integer values $k > 0$ let us rewrite Eq. (4.47) as

$$f_k = H^*_k e^{j\omega kh} \quad , \quad k = 0,1,2,3, \ldots \tag{4.48}$$

where

$$H^*_k = \frac{H^*_{em/N}}{H_{em/N}} \quad , \quad m = 0,1,2, \ldots, N-1, \quad k = m, N+m, 2N+m, \ldots \tag{4.49}$$

It is clear that $e^{j\omega kh}$ in Eq. (4.48) is simply a sinusoidal data sequence with sample period h, the integration step size for the fast subsystem. $H^*_k$ is equal to the ratio of $H^*_{em/N}$, the extrapolator transfer function, to $H_{em/N}$, the ideal extrapolator transfer function; for both transfer functions the dimensionless extrapolation interval is m/N. We have already derived formulas for this ratio in Sections 4.2 through 4.8 for a number of different extrapolation algorithms. From Eq. (4.49) we see that $H^*_k$ represents a periodic data sequence with period Nh, i.e., $H^*_{k+N} = H^*_k$. This means that we can represent $H^*_k$ by means of a discrete Fourier series.* Thus we let

$$H^*_k = \sum_{q=0}^{N-1} C_q e^{j\omega_0 kqh} = \sum_{q=0}^{N-1} C_q e^{j2\pi qk/N} \tag{4.50}$$

Here $H^*_k$ is given by the sum of N sinusoidal data sequences of frequency $0, \omega_0, 2\omega_0, \ldots, (N-1)\omega_0$, where $\omega_0 = 2\pi/T = 2\pi/Nh$, the frequency in radians per second of the fundamental component of the Fourier series.

To obtain the Fourier coefficients $C_q$, we multiply both sides of Eq. (4.50) by $e^{-j2\pi pk/N}$ and sum from $k = 0$ to $k = N-1$. Thus

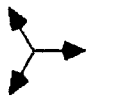
$$\sum_{k=0}^{N-1} H^*_k e^{-j2\pi pk/N} = \sum_{k=0}^{N-1}\sum_{q=0}^{N-1} C_q e^{j2\pi(q-p)k/N} \tag{4.51}$$

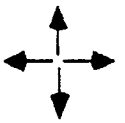
Interchanging the order of summation on the right side of Eq. (4.51), we obtain

*A.V. Oppenheim and R.V. Schafer, *Digita Signal Processing*, Prentiss–Hall, Engle–wood Cliffs, New Jersey, 1975.

$$\sum_{k=0}^{N-1} H_k^* e^{-j2\pi pk/N} = \sum_{q=0}^{N-1} C_q \sum_{k=0}^{N-1} e^{j2\pi(q-p)k/N} \qquad (4.52)$$

The exponential function $e^{j2\pi(q-p)k/N}$ is a complex number which can be represented as a unit vector in the complex plane. The polar angle of the unit vector is equal to $2\pi(q-p)k/N$, where both p and q are integers ranging between 0 and N−1. As k is indexed from 0 to N−1 it is easy to show that the sum of the resulting N unit vectors will add to zero except when q = p, in which case the sum is N. Figure 4.7 shows several specific cases for illustrative purposes. It follows that

$q-p = 1,\ N=3 \qquad \sum_{k=0}^{N-1} e^{j2\pi k/3} \Longrightarrow$  $= 0$

$q-p = 1,\ N=4 \qquad \sum_{k=0}^{N-1} e^{j2\pi k/4} \Longrightarrow$  $= 0$

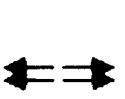
$q-p = 2,\ N=4 \qquad \sum_{k=0}^{N-1} e^{j2\pi k/2} \Longrightarrow$  $= 0$

Figure 4.7. Example vector representations of $\sum_{k=0}^{N-1} e^{j2\pi(q-p)k/N}$.

$$\sum_{k=0}^{N-1} e^{j2\pi(q-p)k/N} = N\ ,\ q = p$$
$$= 0\ ,\ q \neq p \qquad (4.53)$$

Therefore, the double summation on the right side of Eq. (4.52) reduces simply to $C_q N$ and the formula for the Fourier coefficients becomes

$$C_q = \frac{1}{N}\sum_{k=0}^{N-1} H_k^* e^{-j2\pi qk/N}\ ,\ q = 0,1,2,\ ...,\ N-1 \qquad (4.54)$$

In particular, $C_0$, the discrete Fourier series coefficient for the zero−frequency component of $H_k^*$, is given by

$$C_q = \frac{1}{N}\sum_{k=0}^{N-1} H_k^* \qquad (4.55)$$

Thus $C_0$ is simply the average of the N extrapolator transfer function ratios $H_0^*$, $H_1^*$, ... , $H_{N-1}^*$ corresponding, respectively, to the N extrapolation intervals 0, h, 2h, ... , (N−1)h.

Substituting the discrete Fourier series representation for $H_k^*$, as given in Eq. (4.50), into Eq. (4.48), we have

$$f_k = \sum_{q=0}^{N-1} C_q e^{j(q\omega_0 + \omega)kh} \tag{4.56}$$

This is the representation of the kth data point from the extrapolator output data sequence with sample period h. The extrapolator input is the sinusoidal data sequence given in Eq. (4.43) with sample period T (= Nh). The right side of Eq. (4.56) consists of N sinusoidal data sequences of frequency $\omega$, $\omega_0 + \omega$, $2\omega_0 + \omega$, $3\omega_0 + \omega$, . . . , $(N-1)\omega_0 + \omega$, where $\omega_0 = 2\pi/T$, the slow data−sequence sample frequency. When we let the input frequency take on negative as well as positive values, as required in the exponential representation of sinusoids, then the frequencies contained in $f_k$ become $\pm\omega$, $\omega_0 \pm \omega$, $2\omega_0 \pm \omega$, . . . , $(N-1)\omega_0 \pm \omega$. However, the amplitude $C_q$ of the frequencies $q\omega_0 \pm \omega$, where q = 1,2, ... , N−1, is normally small compared with the amplitude $C_0$ corresponding to the original input frequency $\omega$. To show this we let

$$H_k^* = 1 + e_{Mk} + j\, e_{Ak} \tag{4.57}$$

where $1 + e_{Mk}$ and $e_{Ak}$ represent, respectively, the real and imaginary parts of the extrapolator transfer function ratio $H_k^*$. For $|e_{Mk} + j\, e_{Ak}| << 1$ we have shown in Eqs. (1.41) and (1.42) that $e_{Mk}$ is approximately equal to the fractional error in gain of the transfer function and that $e_{Ak}$ is approximately equal to the phase error of the transfer function. In Sections 4.2 through 4.8 we have derived exact and appriximate formulas for $e_{Mk}$ and $e_{Ak}$ for a number of extrapolator algorithms, as summarized in Tables 4.1 and 4.2. In accordance with Eq. (4.49), k merely designates the appro−priate dimensionless extrapolation interval a. Thus

| | | |
|---|---|---|
| a = 0 | for | k = 0, N, 2N, 3N, . . . |
| a = 1/N | for | k = 1, N+1, 2N+1, 3N=1, . . . |
| a = 2/N | for | k = 2, N+2, 2N+2, 3N=2, . . . |
| ⋮ | | ⋮ |
| a = (N−1)/N | for | k = N−1, 2N−1, 3N−1, . . . |

Substituting Eq. (4.57) into Eq. (4.54), we obtain the following formula for the coefficients $C_q$ in the discrete Fourier series representation of $H_k^*$.

$$C_q = \frac{1}{N}\sum_{k=0}^{N-1}(1 + e_{Mk} + je_{Ak})\,e^{-j2\pi qk/N} \tag{4.58}$$

From Eq. (4.53) we see that this can be rewritten as

$$C_0 = 1 + \frac{1}{N}\sum_{k=0}^{N-1}(e_{Mk} + je_{Ak}) \tag{4.59}$$

and

$$C_q = \frac{1}{N}\sum_{k=0}^{N-1}(e_{Mk} + je_{Ak})\,e^{-j2\pi qk/N}\,,\quad q = 0,1,2,\ldots,N-1 \tag{4.60}$$

Thus the amplitudes $C_q$ of the harmonic frequencies $q\omega_0 \pm \omega$ present in the extrapolated fast data sequence are indeed small for small $e_{Mk}$ and $e_{Ak}$. The coefficient $C_0$ in Eq. (4.59) simply represents the effective extrapolator transfer function for the slow input data sequence of frequency $\omega$. It follows that

$$e_{Me} = \text{Extrapolator fractional gain error} \cong \frac{1}{N}\sum_{k=0}^{N-1} e_{Mk} \tag{4.61}$$

and

$$e_{Ae} = \text{Extrapolator phase error} \cong \frac{1}{N}\sum_{k=0}^{N-1} e_{Ak}\,,\quad \omega h << 1 \tag{4.62}$$

As a specific example, consider the case of first-order extrapolation based on $r_n$ and $r_{n-1}$. Letting $a = k/N$ in Eqs. (4.9) and (4.10), we obtain the following formulas for $e_{Mk}$ and $e_{Ak}$:

$$e_{Mk} \cong \frac{k/N}{2}(1 + k/N)(\omega h)^2,\qquad \omega h << 1 \tag{4.63}$$

$$e_{Ak} \cong -\frac{k/N}{6}[1 + 3k/N + 2(k/N)^2](\omega h)^2,\quad \omega h << 1 \tag{4.64}$$

The formulas for $e_{Me}$ and $e_{Ae}$, the extrapolator gain and phase errors, respectively, are obtained by substituting Eqs. (4.63) and (4.64) into Eqs. (4.61) and (4.62). The actual errors depend on the frame-rate multiple N. For $N \to \infty$ the summations in Eqs. (4.61) and (4.62) are replaced by the following integrals:

$$e_{Me} = \int_0^1 e_M(a)\,da\,, \qquad e_{Ae} = \int_0^1 e_A(a)\,da \tag{4.65}$$

Table 4.3 summarizes the multiple–frame extrapolator gain and phase errors for all of the extrapolator algorithms considered in this chapter. The table shows numerical values for the error coefficients when the frame multiple N = 2,3,4,5 and 6, as well as the asymptotic result for $N = \infty$. Note that the case of zero–order extrapolation is also included in the table. This is in effect equivalent to no extrapolation at all; i.e., the fast frame–rate system simply uses the same input $r_n$ from the slow system for each of the N fast system frames until $r_{n+1}$, the next input from the slow system, is available. In this case the extrapolation algorithm is given by

$$r(nT + aT) = \hat{r}_{n+a} = r_n \tag{4.66}$$

and the extrapolator transfer function $H_e^*(e^{j\omega T}) = 1$. After division by the ideal extrapolator transfer function, $H_e(j\omega) = e^{j\omega aT}$, and subtracting 1, we obtain the following expression for the fractional error of the zero–orde extrapolator:

$$\frac{H_e^*(e^{j\omega T})}{H_e(j\omega)} - 1 = e^{-j\omega aT} - 1 \cong -\frac{a^2}{2}(\omega T)^2 - ja(\omega T)\,, \quad \omega T << 1 \tag{4.67}$$

It follows that the formulas for the fractional error in gain and the phase errorare given approximately by

$$e_M \cong -\frac{a^2}{2}(\omega T)^2\,, \quad e_A \cong -a(\omega T)\,, \quad \omega T << 1 \tag{4.68}$$

These formulas are used in Eqs. (4.61) and (4.62) with a = k/N to compute the fractional gain error, $e_{Me}$, and the phase error, $e_{Ae}$, for zero–order extrapolation.

Next we consider the dynamic errors in generating the multiple frame–rate data sequence using interpolation instead of extraploation. We recall that the formulas for interpolation can be obtained from the formulas for extrapolation by simply replacing a with $a-1$ and n with $n+1$. When this is done, the results shown in Table 4.4 are obtained for the gain and phase errors of the multiple frame–rate interpolators. Again we have shown the data for different frame–rate multiples N, and the case of zero–order extrapolation has been included.

Comparison of the error coefficients in Tables 4.3 and 4.4 shows that the errors are generally much smaller when using interpolation rather than extrapolation. In particular, note that interpolation based on $r_{n+1}$, $r_n$ and interpolation based on $r_{n+1}$, $\dot{r}_{n+1}$, $r_n$, $\dot{r}_n$ in each case exhibit zero phase error. This is true because of the

Table 4.3

## Summary of Multiple–Frame Extrapolator Gain and Phase Errors

Note: N = frame multiple. All formulas are approximate based on $\omega h << 1$.

| Extrapolator Inputs | Gain Error Coefficient, $e_{Me}/(\omega h)^2$ | | | | | |
|---|---|---|---|---|---|---|
| | N = 2 | N = 3 | N = 4 | N = 5 | N = 6 | N = ∞ |
| $r_n$ (zero–order) | −.0625 | −.0926 | −.1094 | −.1200 | −.1273 | −.1667 |
| $r_n, r_{n-1}$ | .1875 | .2593 | .2969 | .3200 | .3356 | .4167 |
| $r_n, \dot{r}_n$ | .0625 | .0926 | .1094 | .1200 | .1273 | .1667 |
| | | | $e_{Me}/(\omega h)^4$ | | | |
| $r_n, r_{n-1}, r_{n-2}$ | .1758 | .2634 | .3127 | .3442 | .3569 | .4833 |
| $r_n, r_{n-1}, \dot{r}_n$ | .0195 | .0350 | .0445 | .0508 | .0553 | .0806 |
| $r_n, r_{n-1}, \dot{r}_n, \dot{r}_{n-1}$ | −.0117 | −.0199 | −.0248 | −.0281 | −.0303 | −.0431 |

| Extrapolator Inputs | Phase Error Coefficient, $e_{Ae}/(\omega h)$ | | | | | |
|---|---|---|---|---|---|---|
| | N = 2 | N = 3 | N = 4 | N = 5 | N = 6 | N = ∞ |
| $r_n$ (zero–order) | −.2500 | −.3333 | −.3750 | −.4000 | −.4167 | −.5000 |
| | | | $e_{Ae}/(\omega h)^3$ | | | |
| $r_n, r_{n-1}$ | −.1250 | −.1852 | −.2188 | −.2400 | −.2546 | −.3333 |
| $r_n, \dot{r}_n$ | −.0208 | −.0370 | −.0469 | −.0533 | −.0579 | −.0833 |
| $r_n, r_{n-1}, r_{n-2}$ | .1563 | .2222 | .2578 | .2800 | .2951 | .3750 |
| $r_n, r_{n-1}, \dot{r}_n$ | .0313 | .0494 | .0599 | .0667 | .0714 | .0972 |
| | | | $e_{Ae}/(\omega h)^5$ | | | |
| $r_n, r_{n-1}, \dot{r}_n, \dot{r}_{n-1}$ | .0094 | .0178 | .0232 | .0269 | .0295 | .0444 |

## Table 4.4

### Summary of Multiple–Frame Interpolator Gain and Phase Errors

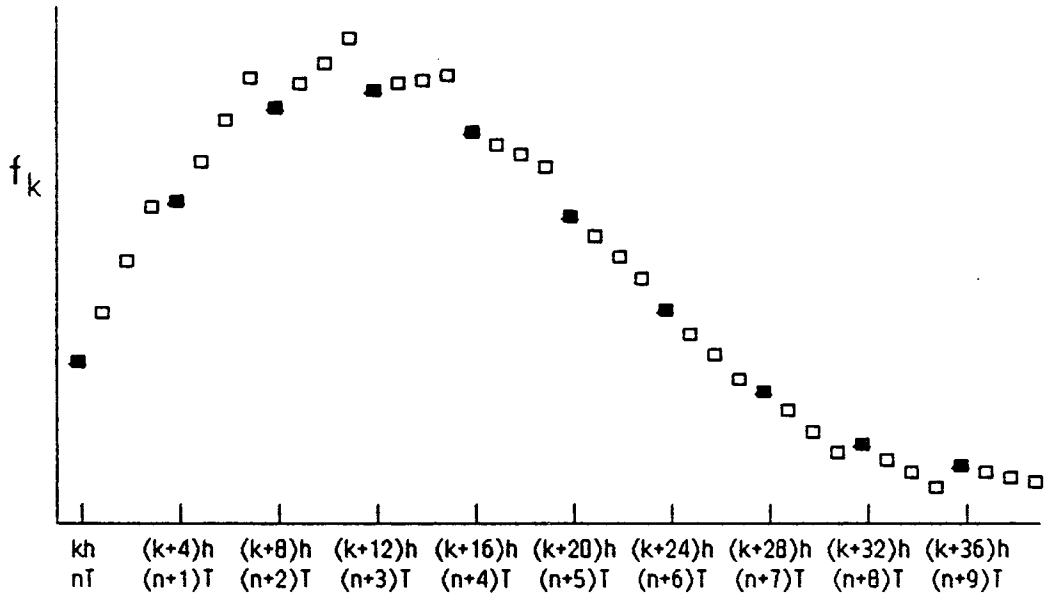
Note: N = frame multiple. All formulas are approximate based on $\omega h << 1$.

| Extrapolator Inputs | Gain Error Coefficient, $e_{Me}/(\omega h)^2$ | | | | | |
|---|---|---|---|---|---|---|
| | N = 2 | N = 3 | N = 4 | N = 5 | N = 6 | N = ∞ |
| $r_n$ (zero–order) | −.0625 | −.0926 | −.1094 | −.1200 | −.1273 | −.1667 |
| $r_{n+1}, r_n$ | −.0625 | −.0741 | −.0781 | −.0800 | −.0810 | −.0833 |
| | | | $e_{Me}/(\omega h)^4$ | | | |
| $r_{n+1}, r_n, r_{n-1}$ | −.01172 | −.01440 | −.01538 | −.01584 | −.01609 | −.01667 |
| $r_{n+1}, r_n, \dot{r}_n$ | −.00130 | −.00206 | −.00236 | −.00251 | −.00259 | −.00278 |
| $r_{n+1}, r_n, \dot{r}_{n+1}, \dot{r}_n$ | −.00130 | −.00137 | −.00138 | −.00139 | −.00139 | −.00139 |

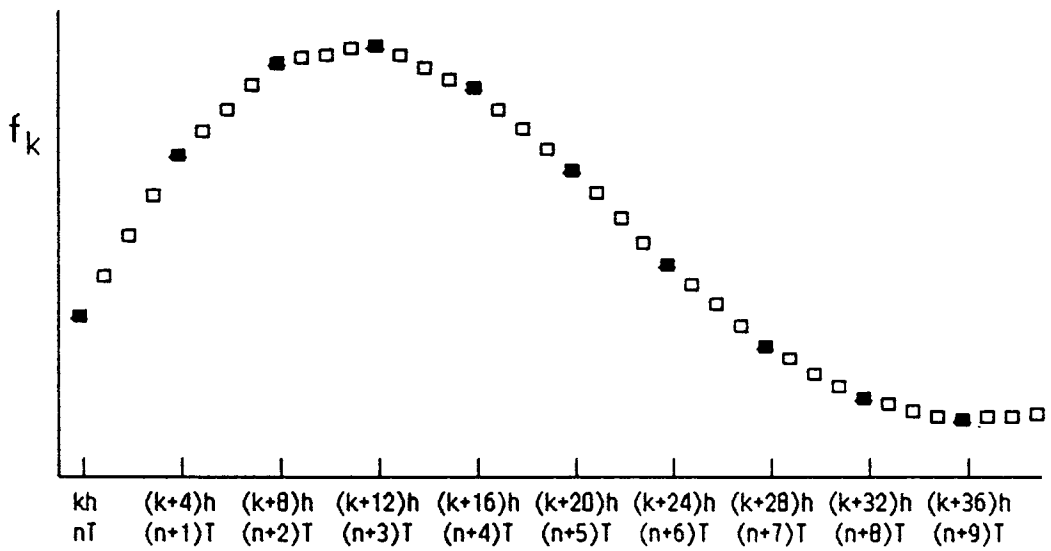
| Extrapolator Inputs | Phase Error Coefficient, $e_{Ae}/(\omega h)$ | | | | | |
|---|---|---|---|---|---|---|
| | N = 2 | N = 3 | N = 4 | N = 5 | N = 6 | N = ∞ |
| $r_n$ (zero–order) | −.2500 | −.3333 | −.3750 | −.4000 | −.4167 | −.5000 |
| | | | $e_{Ae}/(\omega h)^3$ | | | |
| $r_{n+1}, r_n$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $r_{n+1}, r_n, r_{n-1}$ | −.0313 | −.0370 | −.0391 | −.0400 | −.0405 | −.0417 |
| $r_{n+1}, r_n, \dot{r}_n$ | −.0104 | −.0123 | −.0130 | −.0133 | −.0135 | −.0139 |
| | | | $e_{Ae}/(\omega h)^5$ | | | |
| $r_{n+1}, r_n, \dot{r}_{n+1}, \dot{r}_n$ | 0 | 0 | 0 | 0 | 0 | 0 |

symmetry of the algorithms. Both algorithms, to be sure, require $r_{n+1}$ for interpolation over the nth frame. In a dynamic simulation where $r_n$ is a state variable, $r_{n+1}$ can in fact be made available for interpolation if the numerical integration of $\dot{r}$ to obtain $r_{n+1}$ is executed prior to executing the multiple-frame integration algorithm.

Figure 4.7 shown an example of the generation of a fast data sequence from a slow data sequence. In this example $N = 4$, i.e., 4 data points must be computed for the fast data sequence per data point of the slow data sequence. Figure 4.7a uses extrapolation based on $r_n$ and $r_{n-1}$ whereas Figure 4.7b uses interpolation based on $r_n$ and $r_{n+1}$. The superiority of interpolation is quite evident.




a). Fast data sequence generated by first—order extrapolation.




b). Fast data sequence generated by first—order interpolation.

Figure 4.7. Generation of a fast data sequence from a slow data sequence. N = 4.

CHAPTER 5

# DYNAMICS OF DIGITAL–TO–ANALOG AND ANALOG–TO–DIGITAL CONVERSION

## 5.1 Introduction

Real–time digital simulation often involves inputs and/or outputs in the form of analog (i.e., continuous) signals. Clearly this is true for the "hardware–in–the–loop" simulation shown in Figure 5.1, where the hardware requires continuous inputs and produces continuous outputs. In this case the continous outputs from the hardware must be converted to digital data sequences using A to D (analog–to–digital) converters, a single channel of which is shown in the figure. The computer outputs in the form of digital data sequences must be converted to continuous signals using D to A (digital–to–analog) converters. Again, a single channel is shown in Figure 5.1. For both A–to–D and D–to–A converters we have assumed that the sample period h is fixed, which is almost invariably the case in real–time, hardware–in–the–loop simulation.

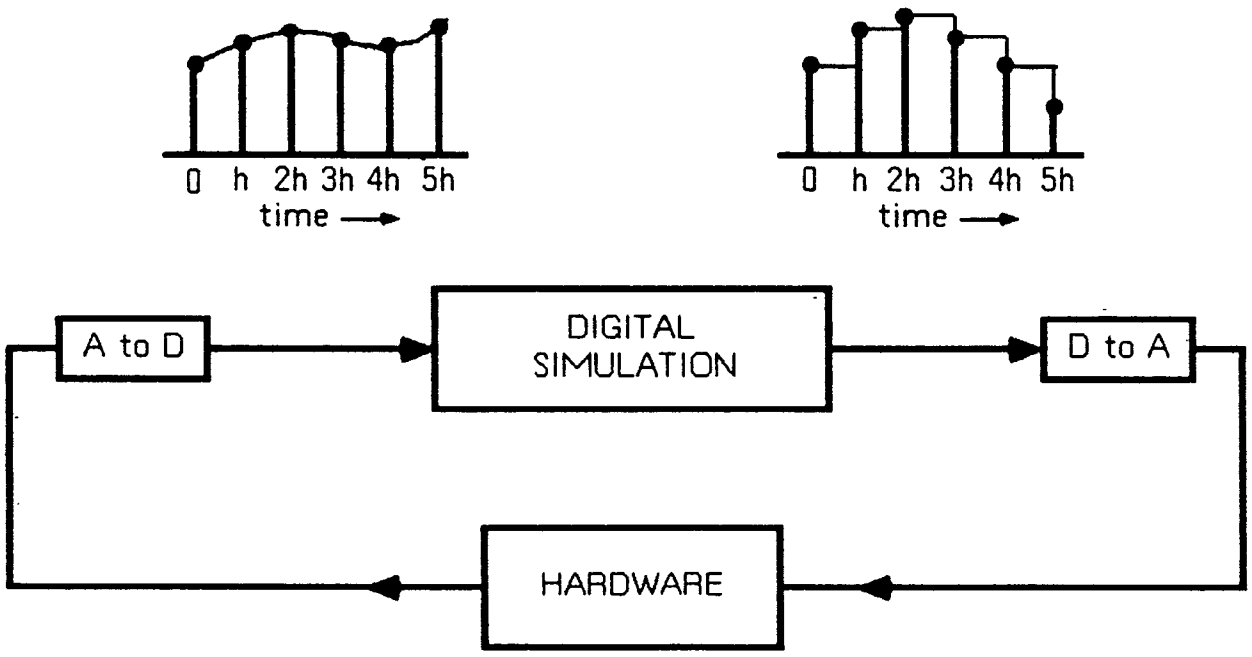



Figure 5.1. Hardware–in–the–loop simulation.

In this chapter we will examine the dynamics of the A–to–D and D–to–A conversion process. As in the previous chapters we will find this most convenient and meaningful in the frequency domain. In the next section we will consider digital–to–analog conversion using both zero and first–order extrapolation. The resulting DAC (digital–to–analog) characteristics will be examined in the frequency domain by deriving the DAC transfer function for sinusoidal input data sequences. This will lead in subsequent sections to the development of digital algorithms to compensate for the DAC transfer function gain and phase errors. The spectral characteristics of A–to–D converters will be examined in the last section of the chapter.

## 5.2 Definition of Zero and First–Order DAC Extrapolation

Assume that the digital computer produces a data sequence $\{r_n\}$ which is converted to a continuous signal using a double–buffered DAC. The DAC data word $r_n$ is loaded into the outer buffer register prior to time $nh$, where $h$ is the fixed time between data points. At $t = nh$, the data word $r_n$ is transferred to the inner buffer register so that the DAC output becomes the equivalent of $r_n$. Every $h$ seconds this process is repeated, which results in the staircase function shown in Figure 5.2. The

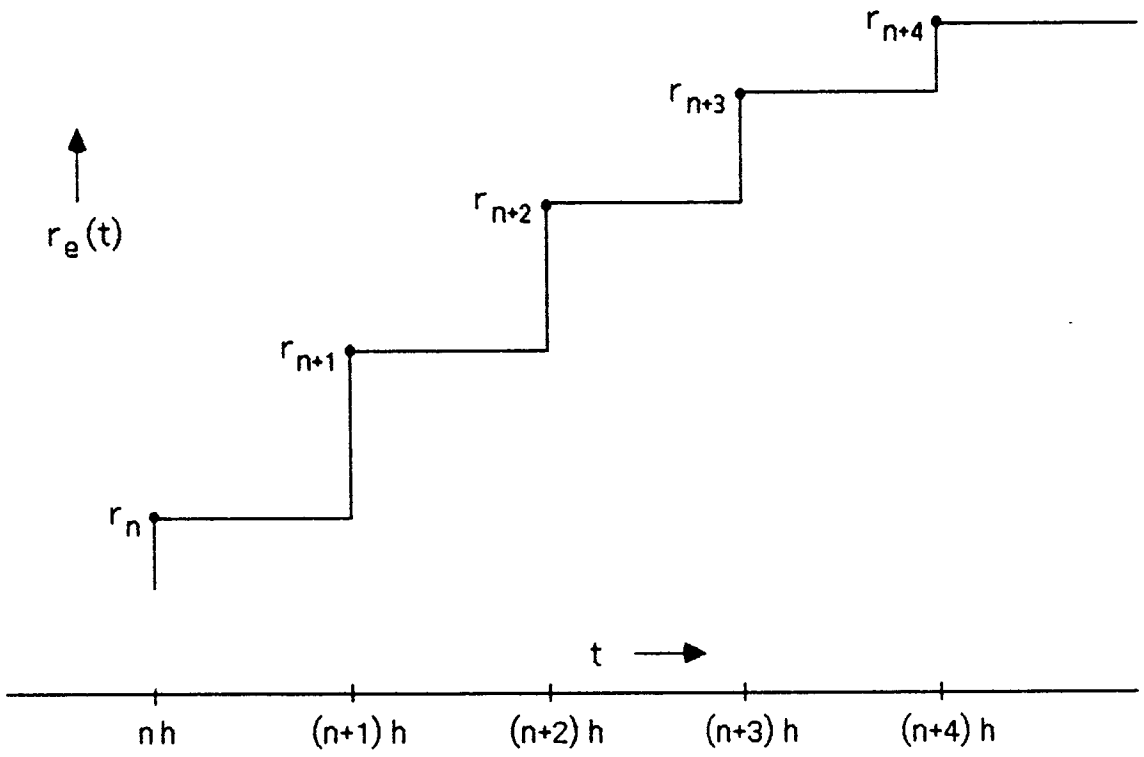



Figure 5.2. Zero–order DAC extrapolation.

DAC output $r_e(t)$ in this case represents a zero–order extrapolation from the input data sequence $\{r_n\}$. This is the most common method of mechanizing DAC outputs in real–time digital simulation.

An alternative is to use first–order extrapolation, which is illustrated in Figure 5.3. Here the DAC is used in combination with an analog integrator and storage device to produce a linearly varying output with slope based on the current $(r_n)$ and previous $(r_{n-1})$ data points.

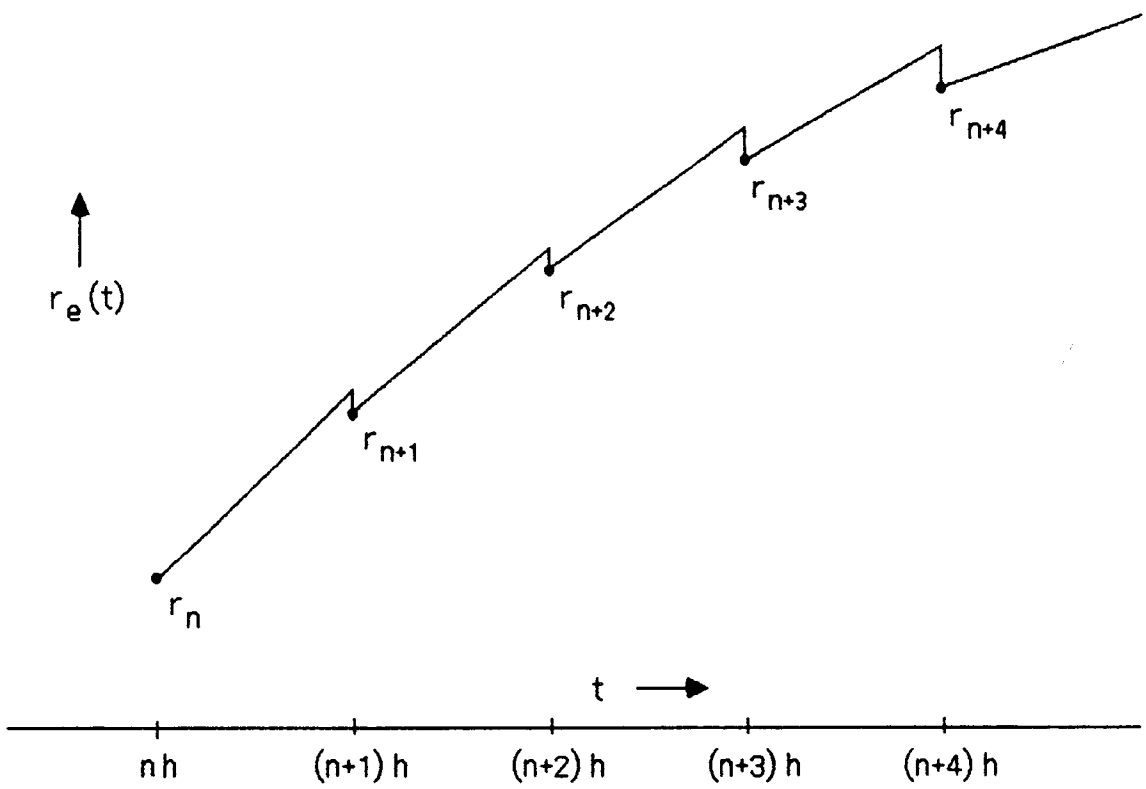



Figure 5.3 First–order DAC extrapolation.

## 5.3 Transfer Function for Zero–Order Extrapolating DAC

We will obtain the transfer function for the DAC which uses zero–order extrapolation by taking the Fourier transform of the DAC output, $r_e(t)$. First we define the unit data point response of the DAC, i.e., the response to the input

$$\begin{aligned} r_n &= 1 \ , \quad n = 0 \\ &= 0 \ , \quad n \neq 0 \end{aligned} \tag{5.1}$$

We define the corresponding DAC response as $h_e(t)$, which is shown in Figure 5.4 in the case of zero–order extrapolation. From the figure it is apparent that

$$\begin{aligned} h_e(t) &= 0, \quad t \leq 0 \\ &= 1, \quad 0 < t \leq h \\ &= 0, \quad t > h \end{aligned} \tag{5.2}$$

Then the DAC output $r_e(t)$ for any input data sequence can be represented as

$$r_e(t) = \sum_{n=1}^{\infty} r_n \, h_e(t-n\,h) \tag{5.3}$$

We now derive the frequency domain representation for the DAC output by taking the

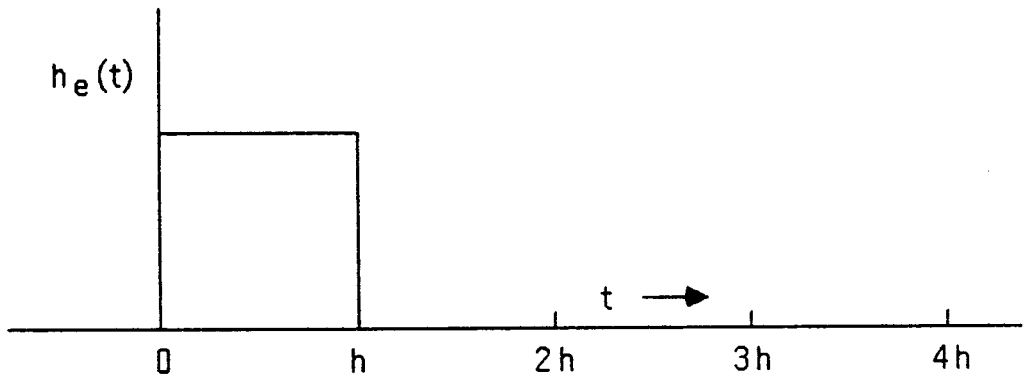



Figure 5.4. Unit data point response of zero–order DAC extrapolator

Fourier transform of $r_e(t)$. Thus

$$R_e(j\omega) = \int_{-\infty}^{\infty} r_e(t)\, e^{-j\omega t}\, dt = \int_{-\infty}^{\infty} \sum_{n=1}^{\infty} r_n \, h_e(t-n\,h)\, e^{-j\omega t}\, dt$$

or, interchanging the order of integration and summation,

$$R_e(j\omega) = \sum_{n=1}^{\infty} r_n \int_{-\infty}^{\infty} h_e(t-n\,h)\, e^{-j\omega t}\, dt \tag{5.4}$$

Here $\int_{-\infty}^{\infty} h_e(t-nh)\, e^{-j\omega t}\, dt$ is the Fourier transform of $h_e(t-nh)$. We recall the translation theorem of the Laplace transform, which states that

$$\mathcal{L}[f(t-t_d)] = e^{-st_d}\, \mathcal{L}[f(t)]$$

The equivalent theorem for the Fourier transform states that

$$\mathcal{F}[h_e(t-nh)] = e^{-j\omega nh}\, \mathcal{F}[h_e(t)]$$

Thus we can rewrite Eq. (5.4) as

$$R_e(j\omega) = \sum_{n=1}^{\infty} r_n\, e^{-j\omega nh} \int_{-\infty}^{\infty} h_e(t)\, e^{-j\omega t}\, dt$$

or

$$R_e(j\omega) = \sum_{n=1}^{\infty} r_n\, (e^{j\omega h})^{-n}\, H_e(j\omega) \tag{5.5}$$

where

$$H_e(j\omega) = \int_{-\infty}^{\infty} h_e(t)\, e^{-j\omega t}\, dt \tag{5.6}$$

We also recall that the z transform of the data sequence $\{r_n\}$ is defined as

$$R^*(z) = \sum_{n=1}^{\infty} r_n\, z^{-n}$$

Hence we can write Eq. (5.5) as

$$R_e(j\omega) = H_e(j\omega)\, R^*(e^{j\omega h}) \tag{5.7}$$

It follows that $H_e(j\omega)$ is the transfer function of the zero-order DAC extrapolator for a sinusoidal data sequence input. We can determine the formula for $H_e(j\omega)$ by substituting Eq. (5.2) into Eq. (5.6). Thus

$$H_e(j\omega) = \int_0^h e^{-j\omega t}\, dt = -\frac{1}{j\omega}\, e^{-j\omega t}\Big|_0^h = \frac{1 - e^{-j\omega h}}{j\omega} \tag{5.8}$$

An alternative expression for $H_e(j\omega)$ is obtained as follows:

$$H_e(j\omega) = \frac{e^{-j\omega h/2}}{\omega/2}\left[\frac{e^{j\omega h/2} - e^{-j\omega h/2}}{j2}\right] = \frac{h \sin(\omega h/2)}{(\omega h/2)}\, e^{-j\omega h/2} \tag{5.9}$$

Figure 5.5 shows plots of the magnitude (gain) and phase angle of $H_e(j\omega)$ versus dimensionless frequency $\omega h$ over the range $0 \leq \omega h \leq 2\pi$. Here $\omega h = 2\pi$ corresponds to $\omega = 2\pi/h$, the data sequence sample frequency in radians per second. In Figure 5.5 the gain is normalized by plotting $|H_e(j\omega)|/h$, which from Eq. (5.9) is given by $\sin(\omega h/2)/(\omega h/2)$. The phase angle is given by $-\omega h/2$. The resulting linear phase shift with frequency is apparent in Figure 5.5. This phase characteristic is equivalent to a pure time delay of h/2 seconds, which is also intuitively obvious in Figure 5.2 when one compares a smoothed curve through the staircase function with the data sequence driving the DAC.

If we make a power series expansion of the zero–order extrapolator transfer function in Eq. (5.8), the following formula is obtained:

$$\frac{H_e(j\omega)}{h} = 1 - \frac{1}{6}(\omega h)^2 + \frac{1}{120}(\omega h)^4 + \cdots + j\left[-\frac{1}{2}\omega h + \frac{1}{24}(\omega h)^3 + \cdots\right] \quad (5.10)$$

Ideally, the DAC extrapolator transfer function should be 1, i.e., input sinusoidal data sequences to the DAC should produce output sinusoids having the same amplitude and phase angle. Thus the fractional error in DAC transfer function is given by

$$\frac{H_e(j\omega)}{h} - 1 = e_H = e_M + je_A \quad (5.11)$$

where from Eq. (5.11)

$$e_M = -\frac{1}{6}(\omega h)^2 + \frac{1}{120}(\omega h)^4 + \cdots, \quad e_A = -\frac{1}{2}\omega h + \frac{1}{24}(\omega h)^3 + \cdots \quad (5.12)$$

In Eqs. (1.42) and (1.43) of Chapter 1 we have shown that for $\omega h \ll 1$, $e_M$ is approximately equal to the fractional error in DAC transfer function gain, and $e_A$ is approximately equal to the DAC transfer function phase error. Eq. (5.12) will form the basis for the zero–order extrapolator compensation algorithms which will be developed in Section 5.5.

## 5.4 Transfer Function for First–Order Extrapolating DAC

The unit data point response for the first–order DAC extrapolator illustrated in Figure 5.3 is shown in Figure 5.6. It is given by

$$h_e(t) = 0, \ t \leq 0; \qquad h_e(t) = 1 + t/h, \ 0 < t \leq h;$$
$$h_e(t) = 1 - t/h, \ h < t \leq 2h; \quad h_e(t) = 0, \ t > 2h \quad (5.13)$$

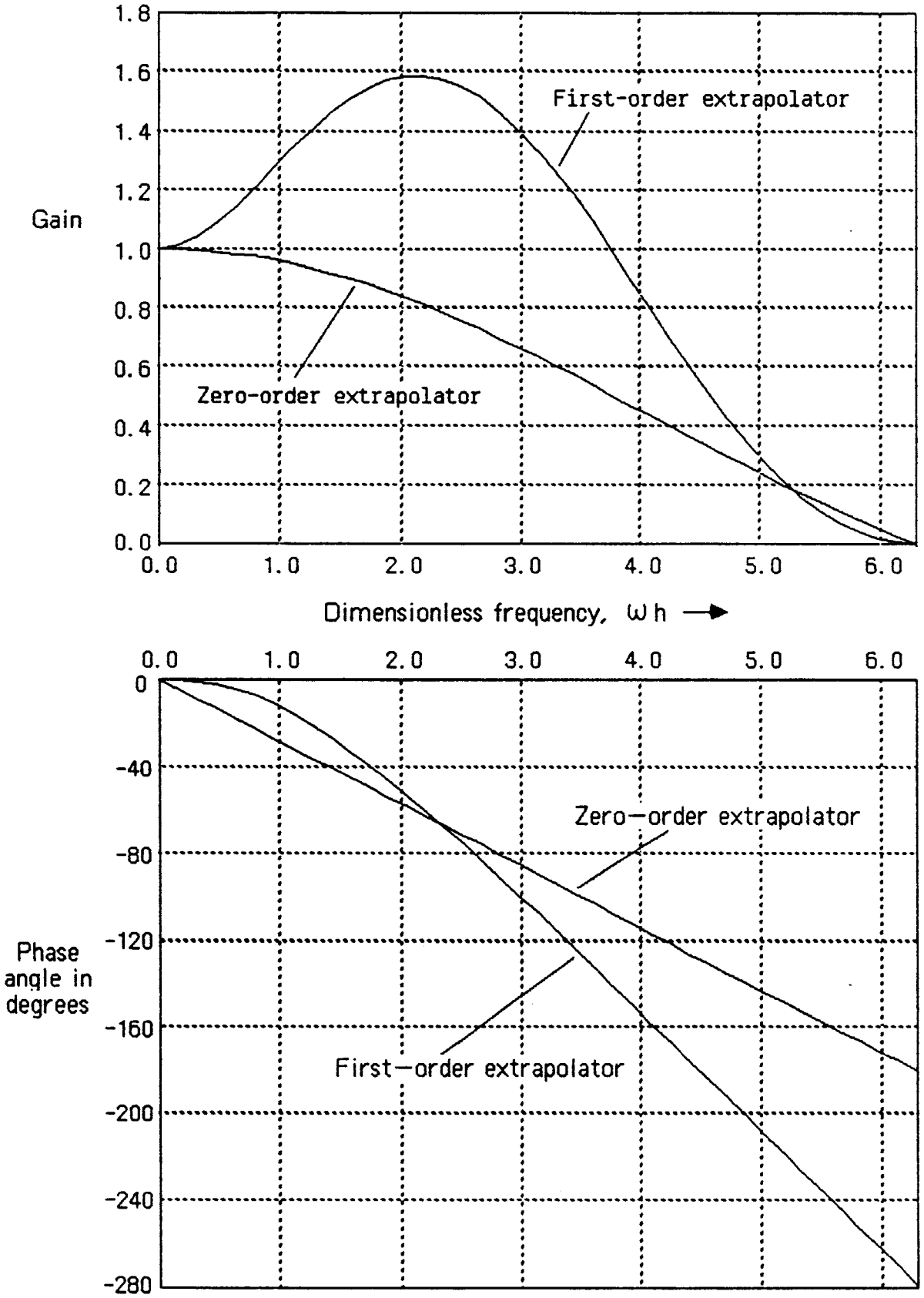



Figure 5.5. Frequency response of zero and first—order DAC extrapolators.

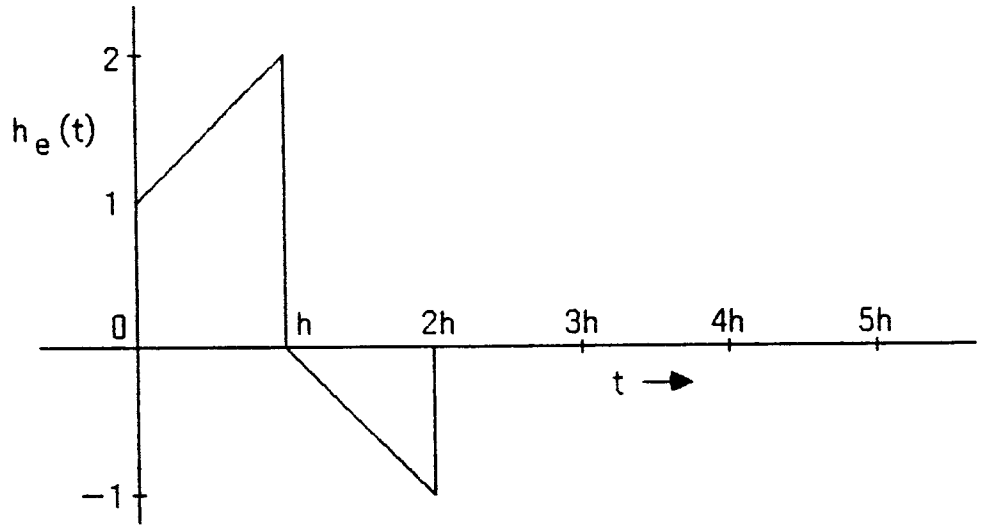



Figure 5.6. Unit data point response of first−order DAC extrapolator.

If we take the Fourier transform of Eq. (5.13) using Eq. (5.6), we obtain the following formula for the first−order extrapolator transfer function:

$$H_e(j\omega) = h^2(1 + j\omega h)\left[\frac{\sin(\omega h/2)}{\omega h/2}\right]^2 e^{-j\omega h} \tag{5.14}$$

Plots of the nomalized gain, $|H_e(j\omega)|/h^2$, and the phase angle of $H_e(j\omega)$ are shown as a function of dimensionless frequency $\omega h$ in Figure 5.5. Note that the gain exhibits a peak of approximately 1.6 at about one−third the sample frequency. By comparison, the transfer function gain for the zero−order extrapolator has no peaking. Also, at low frequencies the phase lag for the first−order extrapolator is quite small compared to the phase lag of the zero−order extrapolator; however, at higher frequencies the first−order extrapolator exhibits a larger lag.

If we make a power series expansion of the first−order extrapolator transfer function in Eq. (5.14), the following formula is obtained:

$$\frac{H_e(j\omega)}{h^2} = 1 + \frac{5}{12}(\omega h)^2 - \frac{59}{360}(\omega h)^4 + \cdots + j\left[-\frac{1}{3}(\omega h)^3 + \cdots\right] \tag{5.15}$$

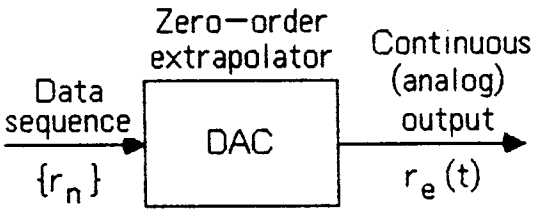
The real and imaginary parts of the fractional error in transfer function of the first−order extrapolating DAC are then given, respectively, by

$$e_M = \frac{5}{12}(\omega h)^2 - \frac{59}{360}(\omega h)^4 + \cdots, \quad e_A = -\frac{1}{3}(\omega h)^3 + \cdots \tag{5.16}$$

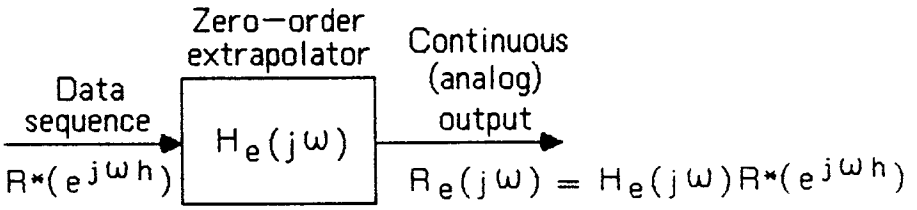
Comparison with Eq. (5.12) for the zero–order extrapolating DAC shows that for $\omega h << 1$ the gain error magnitude is 2.5 times larger when using first–order extrapolation, since $e_M$ represents approximately the fractional error in gain. For both schemes the gain varies as $h^2$. Since $e_A$ represents approximately the phase error, comparison of Eqs. (5.12) and (5.16) shows that for $\omega h << 1$ the phase error is much larger in the case of the zero–order extrapolating DAC. Specifically, the phase error in this case is proportional to the first power of the sample period h and, as pointed out in the previous section, will be the dominant cause of dynamic errors when using zero–order extrapolating DAC's.

## 5.5 Digital Compensation of Zero–Order Extrapolating DAC's

In this section we consider means for compensating the dynamic errors produced by zero–order extrapolation. Figure 5.7a shows a block diagram of the zero–order DAC extrapolation process with the digital input and continuous (analog) output represented in the time domain. In Figure 5.7b the input and output are represented in the frequency domain, with the output equal to the DAC transfer function times the input, in accordance with Eq. (5.7). Ideally we would like the DAC transfer function to be unity. The fractional deviation of the DAC transfer function from unity is represented by $e_H$, as defined in Eq. (5.11).



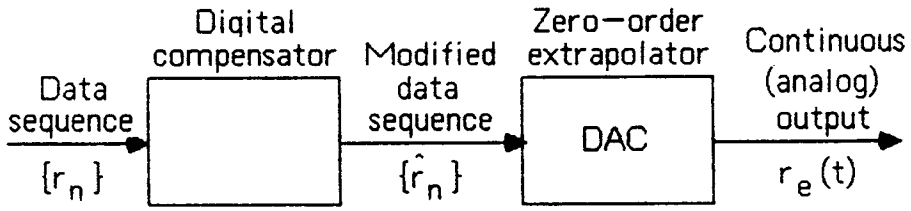

a) Time–domain representation



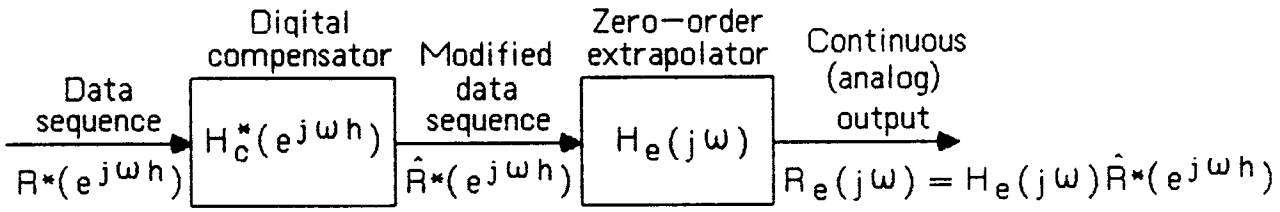

b) Frequency–domain representation

Figure 5.7. Uncompensated DAC

To reduce the DAC transfer function errors, we consider the addition of digital compensation of the DAC inputs. Figure 5.8a shows the time–domain representation. The data sequence $\{r_n\}$ from the digital simulation is converted to a modified data sequence $\{\hat{r}_n\}$ using the compensation algorithm. Then $\{\hat{r}_n\}$ becomes the input to the zero–order DAC extrapolator. Figure 5.8b shows the frequency–domain representation, where $H_c^*(e^{j\omega h})$ is the compensator transfer function. In Figure 5.8b we




a) Time–domain representation




b) Frequency–domain representation

Figure 5.7. DAC with digital compensation.

see that the Fourier transform, $R_e(j\omega)$, of the DAC output is given by

$$R_e(j\omega) = H_e(j\omega)H_c^*(e^{j\omega h})R^*(e^{j\omega h}) \tag{5.17}$$

In Eq. (5.11) we represented the DAC transfer function in the form

$$\frac{H_e(j\omega)}{h} = 1 + e_H \tag{5.18}$$

If we choose the digital compensation algorithm such that the compensator transfer function is given by

$$H_c^*(e^{j\omega h}) = 1 - e_H \tag{5.19}$$

then the combined compensator–DAC transfer function will be

$$\frac{H_e(j\omega)H_c^*(e^{j\omega h})}{h} = 1 - e_H^2 \tag{5.20}$$

The fractional error in the combined transfer function relating the continuous output $r_e(t)$ to the input data sequence $\{r_n\}$ will now be $-e_H^2$. This will result in significant accuracy improvement when $|e_H| \ll 1$. Note that Figures 5.7 and 5.8, as well as Eqs. (5.17), (5.19), and (5.20) apply in general, regardless of the type of DAC extrapolation. We now consider some specific examples.

### 5.5.1 DAC Compensation Using $r_n, r_{n-1}$

We have seen in Eq. (5.12) that the transfer function error $e_H$ for the zero-order DAC extrapolator is equal to $-j\omega h/2$ for $\omega h \ll 1$. According to Eq. (5.19) the extrapolator transfer function $H_c^*$ should be equal to $1+j\omega h/2$ for $\omega h \ll 1$. We can generate a transfer function proportional to $j\omega$ by using an algorithm which approximates differentiation. Thus we let the incremental compensation $\Delta r_n$ be given by the backward difference

$$\Delta r_n = r_n - r_{n-1} \tag{5.21}$$

Taking the z transform, we have

$$\Delta R^*(z) = (1 - z^{-1})R^*(z) \tag{5.22}$$

We define $\Delta H^* = \Delta R^*/R^*$ as the incremental compensator z transform. Thus

$$\Delta H^*(z) = \frac{\Delta R^*(z)}{R^*(z)} = 1 - z^{-1} \tag{5.23}$$

The incremental compensator transfer function for sinusoidal inputs is given by

$$\Delta H^*(e^{j\omega h}) = 1 - e^{-j\omega h} \tag{5.24}$$

Expanding the exponential function in a power series, we obtain

$$\Delta H^*(e^{j\omega h}) = j\omega h + \frac{1}{2}(\omega h)^2 - j\frac{1}{6}(\omega h)^3 + \frac{1}{24}(\omega h)^4 + \cdots \tag{5.25}$$

For $\omega h \ll 1$, $\Delta H^*$ in Eq. (5.25) clearly provides the required incremental transfer function proportional to $j\omega$. To make the compensator transfer function $H_c^* = 1 - e_H = 1 + j\omega h/2$ it is evident that we should let $H_c^* = 1 + \Delta H^*/2$. This in turn means

from Eq. (5.21) that the formula for the compensator output is the following:

$$\hat{r}_n = r_n + \frac{1}{2}\Delta r_n = r_n + \frac{1}{2}(r_n - r_{n-1}) \tag{5.26}$$

After taking the z transform and writing the compensator transfer function for sinusoidal inputs, we obtain

$$H_c^*(e^{j\omega h}) = \frac{3}{2} - \frac{1}{2}e^{-j\omega h} \tag{5.27}$$

From Eqs. (5.8) and (5.27) the formula for the combined compensator–DAC transfer function becomes

$$H_e(j\omega)H_c^*(e^{j\omega h}) = \frac{1 - e^{-j\omega h}}{j\omega}\left[\frac{3}{2} - \frac{1}{2}e^{-j\omega h}\right] \tag{5.28}$$

Expanding the exponential functions in power series and neglecting terms of order $h^4$ and higher, we obtain

$$\frac{H_e(j\omega)H_c^*(e^{j\omega h})}{h} \cong 1 + \frac{1}{3}(\omega h)^2 - j\frac{1}{4}(\omega h)^3, \quad \omega h << 1 \tag{5.29}$$

Comparison with Eq. (5.10) for the uncompensated DAC transfer function shows that now the fractional gain error predominates and is equal to $(\omega h)^2/3$ for $\omega h << 1$. The phase error has been reduced from $-\omega h/2$ to $-(\omega h)^3/4$ for $\omega h << 1$.

### 5.5.2 DAC Compensation Using $r_n, \dot{r}_n$

Next we let the incremental compensation used to generate a transfer function proportional to $j\omega$ be given directly by the derivative, $\dot{r}_n$, of the input to the DAC. This requires, of course, that $r_n$ is a state variable or is explicitly related to state variables such that $\dot{r}_n$ can be computed. Thus we let

$$\Delta r_n = \dot{r}_n h \tag{5.30}$$

from which the incremental compensator transfer function is given by

$$\Delta H^* = j\omega h \tag{5.31}$$

To obtain the desired compensator transfer function $H_c^* = 1 + j\omega h/2$ it is evident that we should let $H_c^* = 1 + \Delta H^*/2$ from which it follows that

$$\hat{r}_n = r_n + \frac{1}{2}\dot{r}_n h \tag{5.32}$$

The compensator transfer function becomes

$$H_c^*(e^{j\omega h}) = 1 + j\frac{1}{2}\omega h \tag{5.33}$$

as required. From Eqs. (5.8) and (5.33) the formula for the combined compensator–DAC transfer function becomes

$$H_e(j\omega)H_c^*(e^{j\omega h}) = \frac{1 - e^{-j\omega h}}{j\omega}\left[1 + j\frac{1}{2}\omega h\right] \tag{5.34}$$

Expanding the exponential function in a power series and neglecting terms of order $h^4$, we obtain

$$\frac{H_e(j\omega)H_c^*(e^{j\omega h})}{h} \cong 1 + \frac{1}{12}(\omega h)^2 - j\frac{1}{24}(\omega h)^3, \quad \omega h << 1 \tag{5.35}$$

Comparison with Eq. (5.29) for compensation based on $r_n$ and $r_{n-1}$ shows that in Eq. (5.35) the gain error, $(\omega h)^2/12$, is one–fourth as large. The comparison also shows that the phase error, $-(\omega h)^2/24$, is one sixth as large. Thus the DAC compensation based on $r_n$ and $\dot{r}_n$ is significantly more accurate than compensation based on $r_n$ and $r_{n-1}$.

### 5.5.3 DAC Compensation Using $r_n$, $r_{n-1}$ and $r_{n-2}$

When first–order compensation is used, we have seen in Eqs. (5.29) and (5.35) that the combined compensator–DAC transfer function error is proportional to $(\omega h)^2$. We can use a second–order compensation to eliminate this error. The procedure involves the generation of a transfer function proportional to $\omega^2$ by means of an algorithm which approximates the second time derivative. Thus we let the incremental compensation $\Delta r_n$ be given by the second–order difference

$$\Delta r_n = r_n - 2r_{n-1} + r_{n-2} \tag{5.36}$$

Here the right side of Eq. (5.36) is proportional to the numerical approximation for $\ddot{r}_{n-1}$. Taking the z transform and solving for $\Delta H^* = \Delta R^*/R^*$, we obtain

$$\frac{\Delta R^*(z)}{R^*(z)} = \Delta H^* = 1 - 2z^{-1} + z^{-2} \tag{5.37}$$

The incremental compensator transfer function for sinusoidal inputs is then given by

$$\Delta H^*(e^{j\omega h}) = 1 - 2e^{-j\omega h} + e^{-j2\omega h} \tag{5.38}$$

Expanding the exponential functions in power series, we have

$$\Delta H^*(e^{j\omega h}) = -(\omega h)^2 + j(\omega h)^3 + \cdots \tag{5.39}$$

From Eq. (5.29) we see that the desired incremental transfer function, $\Delta H_c^*$, is equal to $-(\omega h)^2/3$, i.e., $\Delta H^*/3$ in Eq. (5.39). Thus we let $\Delta \hat{r}_n$, the incremental compensation, be equal to $\Delta r_n/3 = (r_n - 2r_{n-1} + r_{n-2})/3$. This is added to the previous first-order compensation algorithm of Eq. (5.26) with the following result:

$$\hat{r}_n = r_n + \frac{1}{2}(r_n - r_{n-1}) + \frac{1}{3}(r_n - 2r_{n-1} + r_{n-2})$$

or

$$\hat{r}_n = \frac{11}{6} r_n - \frac{7}{6} r_{n-1} + \frac{1}{3} r_{n-2} \tag{5.40}$$

After taking the z transform, replacing z by $e^{j\omega h}$, and solving for the compensator transfer function, we obtain

$$H_c^*(e^{j\omega h} = \frac{11}{6} - \frac{7}{6} e^{-j\omega h} + \frac{1}{3} e^{-2j\omega h} \tag{5.41}$$

From Eq. (5.8) we have the following formula for the combined compensator-DAC transfer function:

$$H_e(j\omega) H_c^*(e^{j\omega h}) = \frac{1 - e^{-j\omega h}}{j\omega} \left[ \frac{11}{6} - \frac{7}{6} e^{-j\omega h} + \frac{1}{3} e^{-2j\omega h} \right] \tag{5.42}$$

Expanding the exponential functions in power series and neglecting terms of order $h^5$ and higher, we obtain

$$\frac{H_e(j\omega) H_c^*(e^{j\omega h})}{h} \cong 1 + \frac{3}{10}(\omega h)^4 + j\frac{1}{4}(\omega h)^3, \quad \omega h << 1 \tag{5.43}$$

With the second-order DAC compensator algorithm of Eq. (5.40 we have eliminated the gain error of $(\omega h)^2/3$ in Eq. (5.29). The predominant error for $\omega h << 1$ is now the phase error $(\omega h)^2/4$.

### 5.5.4 DAC Compensation Using $r_n, \dot{r}_n, r_{n-1}$

If $\dot{r}_n$ is available, which will be the case if $r_n$ is a state variable in the simulation, then a better approximation to $\ddot{r}_n$ than that given in Eq. (5.36) is one proportional to

$$\Delta r_n = \dot{r}_n h - r_n + r_{n-1} \tag{5.44}$$

The incremental compensator transfer function is given by

$$\Delta H^*(e^{j\omega h}) = j\omega h - 1 + e^{-j\omega h} \tag{5.45}$$

Expanding the exponential function in a power series, we obtain

$$\Delta H^*(e^{j\omega h}) = -\frac{1}{2}(\omega h)^2 + j\frac{1}{6}(\omega h)^3 + \cdots \tag{5.46}$$

In this case we apply the second-order compensation to the first-order compensation defined in Section 5.5.2 and based on $r_n$ and $\dot{r}_n$. From Eq. (5.35) we see that the desired incremental transfer function is equal to $-(\omega h)^2/12$. This can be obtained by making $\Delta H_c^* = \Delta H^*/6$, where $\Delta H^*$ is given by Eq. (5.46). This in turn means that the incremental compensation, $\Delta \hat{r}_n$, is equal to $\Delta r_n/6$, with $\Delta r_n$ defined in Eq. (5.44). This incremental compensation is then added to the previous first-order algorithm of Eq. (5.32) to produce the formula

$$\hat{r}_n = r_n + \frac{1}{2}\dot{r}_n h + \frac{1}{6}(\dot{r}_n h - r_n + r_{n-1})$$

or

$$\hat{r}_n = \frac{5}{6}r_n + \frac{1}{6}r_{n-1} + \frac{2}{3}\dot{r}_n h \tag{5.47}$$

From Eq. (5.47) the compensator transfer function is given by

$$H_c^*(e^{j\omega h}) = \frac{5}{6} + \frac{1}{6}e^{-j\omega h} + \frac{2}{3}j\omega h \tag{5.48}$$

From Eq. (5.8) we have the following formula for the combined compensator-DAC transfer function:

$$H_e(j\omega)H_c^*(e^{j\omega h}) = \frac{1-e^{-j\omega h}}{j\omega}\left[\frac{5}{6} + \frac{1}{6}e^{-j\omega h} + \frac{2}{3}j\omega h\right] \tag{5.49}$$

Expanding the exponential functions in power series and neglecting terms of order $h^5$ and higher, we obtain

$$\frac{H_e(j\omega)H_c^*(e^{j\omega h})}{h} \cong 1 + \frac{1}{45}(\omega h)^4 + j\frac{1}{36}(\omega h)^3, \quad \omega h \ll 1 \tag{5.50}$$

Comparison with Eq. (5.43) for compensation based on $r_n$, $r_{n-1}$ and $r_{n-2}$ shows that the compensation here, which is based on $r_n$, $\dot{r}_n$ and $r_{n-1}$, exhibits one-ninth the phase error and less than one-thirteenth the gain error for $\omega h \ll 1$. As we have seen in previous examples, the time derivative of the data sequence input, when available explicitly, should be used in preference to past values of the data sequence when implementing digital extrapolation or compensation.

### 5.5.5 DAC Compensation Using $r_n, \dot{r}_n, \dot{r}_{n-1}$

This second–order DAC–compensation method is based on $r_n, \dot{r}_n$ and $\dot{r}_{n-1}$. It is a convenient scheme if $r_n$ is a state variable and the integration is being performed using a predictor or predictor–corrector algorithm, since in this case $\dot{r}_{n-1}$ as well as $\dot{r}_n$ will be avaiable at the nth integration frame. Here we use $\dot{r}_n - \dot{r}_{n-1}$ to provide an incremental compensation proportional to the second derivative. Thus we let

$$\Delta r_n = (\dot{r}_n - \dot{r}_{n-1})h \tag{5.51}$$

Taking the z transform, replacing z by $e^{j\omega h}$, and solving for the incremental compensator transfer function, we obtain

$$\Delta H^*(e^{j\omega h}) = j\omega h(1 - e^{-j\omega h}) \tag{5.52}$$

Expanding the exponential function in a power series, we have

$$\Delta H^*(e^{j\omega h}) = -(\omega h)^2 + j\frac{1}{2}(\omega h)^3 + \cdots \tag{5.53}$$

As in the previous section, this second–order incremental compensation is added to the first–order compensation based on $r_n$ and $\dot{r}_n$. From Eq. (5.35) we see that the the desired incremental transfer function is given by $-(\omega h)^2/12$. This can be obtained by making $\Delta H_c^* = \Delta H^*/12$, with $\Delta H^*$ given by Eq. (5.53). Thus $\Delta \dot{r}_n$, the incremental compensation, is equal to $\Delta r_n/12$, with $\Delta r_n$ defined in Eq. (5.51). This incremental compensation is then added to the first–order algorithm of Eq. (5.32) to produce the formula

$$\hat{\dot{r}}_n = r_n + \frac{7}{12}\dot{r}_n - \frac{1}{12}\dot{r}_{n-1} \tag{5.54}$$

From Eq. (5.54) the compensator transfer function, when combined with the DAC transfer function of Eq. (5.8), results in

$$H_e(j\omega)H_c^*(e^{j\omega h}) = \frac{1 - e^{-j\omega h}}{j\omega}\left[1 + j\frac{1}{12}\omega h(7 - e^{-j\omega h})\right] \tag{5.55}$$

Expanding the exponential functions in power series and neglecting terms of order $h^5$ and higher, we obtain

$$\frac{H_e(j\omega)H_c^*(e^{j\omega h})}{h} \cong 1 + \frac{13}{360}(\omega h)^4 + j\frac{1}{24}(\omega h)^3, \quad \omega h << 1 \tag{5.56}$$

Comparison with Eq. (5.43) shows that the compensation here, which is based on $r_n, \dot{r}_n$ and $\dot{r}_{n-1}$, is considerably more accurate than compensation based on $r_n, r_{n-1}$ and

$r_{n-2}$. On the other hand, comparison with Eq. (5.50) shows that it is slightly less accurate than compensation based on $r_n$, $\dot{r}_n$ and $r_{n-1}$.

### 5.5.6 DAC Compensation Using $r_n$, $\dot{r}_n$ and $\ddot{r}_n$

The final DAC compensation method which we will consider is based on using $r_n$, $\dot{r}_n$ and $\ddot{r}_n$. To use this scheme requires that both $r_n$ and $\dot{r}_n$ are state variables. This will in fact be the case when $\ddot{r}_n$ represents an acceleration, which is integrated to obtain a velocity, $\dot{r}_n$, which is in turn is integrated to obtain the displacement $r_n$. Here the second time derivative is represented directly by $\ddot{r}_n$. The corresponding incremental compensator transfer function is $-\omega^2$. From Eq. (5.35) we see that the desired incremental transfer function is given by $-(\omega h)^2/12$. Thus we let the incremental compensation $\Delta \hat{r}_n = \ddot{r}_n h^2/12$, which is then added to the right side of Eq. (5.32) to produce the formula

$$\hat{r}_n = r_n + \frac{1}{2}\dot{r}_n + \frac{1}{12}\ddot{r}_n \tag{5.57}$$

From Eq. (5.57) the compensator transfer function, when combined with the DAC transfer function of Eq. (5.8), results in

$$H_e(j\omega)\,H_c^*(e^{j\omega h}) = \frac{1 - e^{-j\omega h}}{j\omega}\left[1 + j\frac{1}{2}\omega h - \frac{1}{12}(\omega h)^2\right] \tag{5.58}$$

Expanding the exponential function in a power series and neglecting terms of order $h^6$, we obtain

$$\frac{H_e(j\omega)\,H_c^*(e^{j\omega h})}{h} \cong 1 + \frac{1}{720}(\omega h)^4 - j\frac{1}{1440}(\omega h)^5, \quad \omega h << 1 \tag{5.59}$$

Comparison of Eq. (5.59) with equations representing the combined transfer function errors for the other second-order DAC compensators, as given in Eqs. (5.43), (5.50) and (5.56), shows that the scheme here, which is based on $r_n$, $\dot{r}_n$ and $\ddot{r}_n$, is by far the most accurate. Not only is the fractional gain error represented by the $(\omega h)^4$ term in the equation smaller, but the phase error is of order $(\omega h)^5$ and has a very small coefficient. In the other schemes the phase error is proportional to $(\omega h)^3$.

### 5.5.7 Summary of DAC Compensation Schemes

Table 7.1 summarizes the DAC compensation algorithms considered in Sections 5.5.1 through 5.5.7, as well as the asymptotic formulas for the gain and phase errors of the combined compensator–DAC transfer function for each algorithm. For the

Table 7.1

Zero–Order DAC Extrapolation; Summary of Compensator Formulas and Combined Compensator–DAC Transfer Function Errors

| Inputs | Formula | Transfer function error for $\omega h \ll 1$: Fractional Gain Error, $e_M$ | Transfer function error for $\omega h \ll 1$: Phase Error, $e_A$ |
|---|---|---|---|
| $r_n$ | $\hat{r}_n = r_n$ (no compensation) | $-\frac{1}{6}(\omega h)^2$ | $-\frac{1}{2}\omega h$ |
| $r_n, r_{n-1}$ | $\hat{r}_n = \frac{3}{2}r_n - \frac{1}{2}r_{n-1}$ | $\frac{1}{3}(\omega h)^2$ | $-\frac{1}{4}(\omega h)^3$ |
| $r_n, \dot{r}_n$ | $\hat{r}_n = r_n + \frac{1}{2}\dot{r}_n h$ | $\frac{1}{12}(\omega h)^2$ | $-\frac{1}{24}(\omega h)^3$ |
| $r_n, r_{n-1}, r_{n-2}$ | $\hat{r}_n = \frac{11}{6}r_n - \frac{7}{6}r_{n-1} + \frac{1}{3}r_{n-2}$ | $\frac{1}{10}(\omega h)^4$ | $\frac{1}{4}(\omega h)^3$ |
| $r_n, \dot{r}_n, r_{n-1}$ | $\hat{r}_n = \frac{5}{6}r_n + \frac{1}{6}r_{n-1} + \frac{2}{3}\dot{r}_n h$ | $\frac{1}{45}(\omega h)^4$ | $\frac{1}{36}(\omega h)^3$ |
| $r_n, \dot{r}_n, \dot{r}_{n-1}$ | $\hat{r}_n = r_n + \frac{7}{12}\dot{r}_n h - \frac{1}{12}\dot{r}_{n-1} h$ | $\frac{13}{360}(\omega h)^4$ | $\frac{1}{24}(\omega h)^3$ |
| $r_n, \dot{r}_n, \ddot{r}_n$ | $\hat{r}_n = r_n + \frac{1}{2}\dot{r}_n h + \frac{1}{12}\ddot{r}_n h^2$ | $\frac{1}{720}(\omega h)^4$ | $-\frac{1}{1440}(\omega h)^5$ |

two first–order schemes the predominant transfer function error is the gain error proportional to $(\omega h)^2$, with the $r_n, \dot{r}_n$ method clearly superior when it can be used. For the first three second–order schemes the predominant error is the phase error proportional to $(\omega h)^3$, with the $r_n, \dot{r}_n, r_{n-1}$ method superior, but with the $r_n, \dot{r}_n, r_{n-1}$ method about two–thirds as accurate. Finally, the $r_n, \dot{r}_n, \ddot{r}_n$ method is by far the most accurate, with a very small gain error proportional to $(\omega h)^4$ as the predominant error for the combined compensator–DAC transfer function.

Figure 5.9 shows plots of the gain and phase versus $\omega h$ for the compensator–DAC transfer function for $r_n, r_{n-1}$ compensation, $r_n, \dot{r}_n$ compensation, and no compensation. Note that both first–order compensation schemes cause peaking in the gain curves at between one–third and one–half the sample frequency, with less peaking in the case of $r_n, \dot{r}_n$ compensation. The superior phase characteristic of the

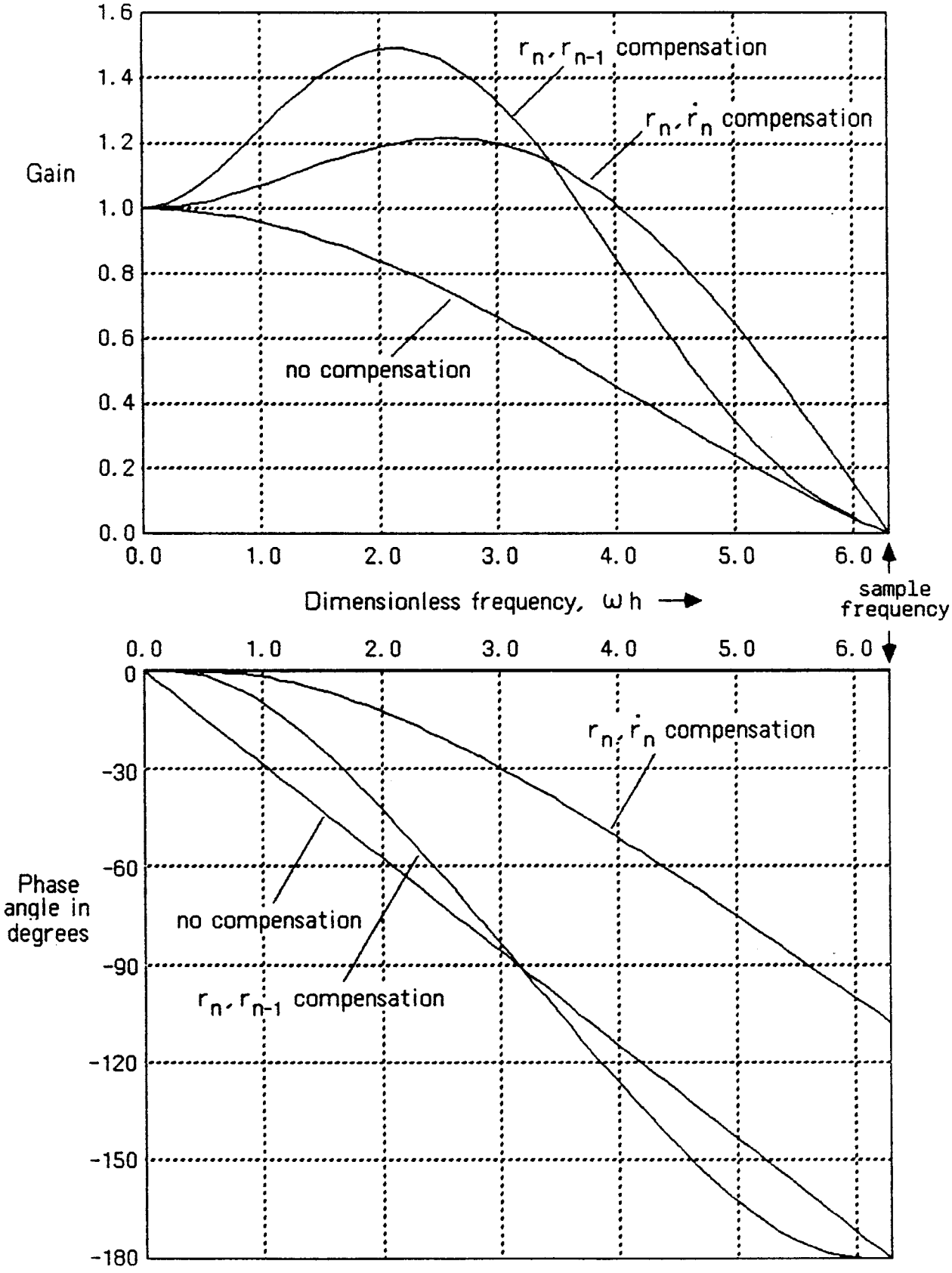



Figure 5.9. Frequency response of DAC with first–order compensation

$r_n, \dot{r}_n$ scheme is also evident in the figure. The plots in Figure 5.9 cover the entire frequency range out to the sample frequency $\omega_0 = 2\pi/h$, i.e., $\omega_0 h = 2\pi$; the formulas in Table 7.1 apply only for $\omega h << 1$.

Figure 5.10 shows plots of the gain and phase shift versus $\omega h$ for the compensator–DAC transfer function cases involving second–order algorithms. Note the very appreciable peaking in the gain curve for $r_n, r_{n-1}, r_{n-2}$ compensation, as well as the more modest peaking associated with $r_n, \dot{r}_n, r_{n-1}$ and $r_n, \dot{r}_n, r_{n-1}$ compensation. In all three cases the peaking occurs at roughly one–half the sample frequency. The peaking associated with $r_n, r_{n-1}, r_{n-2}$ compensation and the large accompanying phase lag could cause significant problems in a hardware–in–the–loop simulation if the output data sequence $\{r_n\}$ driving the compensator and DAC has significant frequency components near one half the sample frequency. Note in Figure 7.10 the excellent gain and phase performance of the compensation based on $r_n, \dot{r}_n$ and $\ddot{r}_n$.

To this point all of our frequency–domain considerations have focused on frequencies between zero and the sample frequency $\omega_0 = 2\pi/h$. As discussed in the next section, the DAC input data sequence will contain frequencies above $\omega_0$ as well. These will result in DAC output frequencies above $\omega_0$. For example, a sinusoidal data sequency at $\omega_0/2$, i.e., one–half the sample frequency, will have a fundamental component at $\omega_0/2$ and harmonics of the same amplitude at 3, 5, 7, ... times the fundamental frequency $\omega_0/2$. For $\omega = \omega_0/2$, we note that $\omega h/2 = \pi/2$ and according to Eq. (5.9) the DAC (without compensation) will have a gain $|H_e(j\omega)|/h$ given by $1/(\pi/2) = 2/\pi$. For $\omega = 3\omega_0/2$ the DAC gain will be $(2/\pi)/3$; for $\omega = 5\omega_0/2$ the DAC gain will be $(2/\pi)/5$; etc. Thus the DAC output consists of the fundamental plus odd harmonics, with the amplitude of each harmonic inversely proportional to its frequency. This is not an unexpected result when we recall that a data sequence input to the DAC at one–half the sample frequency produces a DAC output that is a square wave at one–half the sample frequency. A Fourier series expansion of a square wave contains the fundamental plus odd harmonics, with the am–amplitude of the harmonics inversely proportional to their frequency.

In general, a DAC with zero–order extrapolation, with or without compensation, will produce harmonics at frequencies above the sample frequency due to the discontinuities in the DAC output. The same will be true for a DAC using any order of extrapolation, but to a lesser degree when the input frequency is small compared with the sample frequency. If such discontinuities cause problems in the simulation, the DAC output can always be passed through a low–pass analog filter. However, the low–pass filter will itself introduce additional phase lag. The most effective solution for such a problem is to use a fast enough sample rate. This in turn requires a digital computer which has the speed necessary to achieve integration frame rates which are well beyond the highest significant frequencies in the simulation

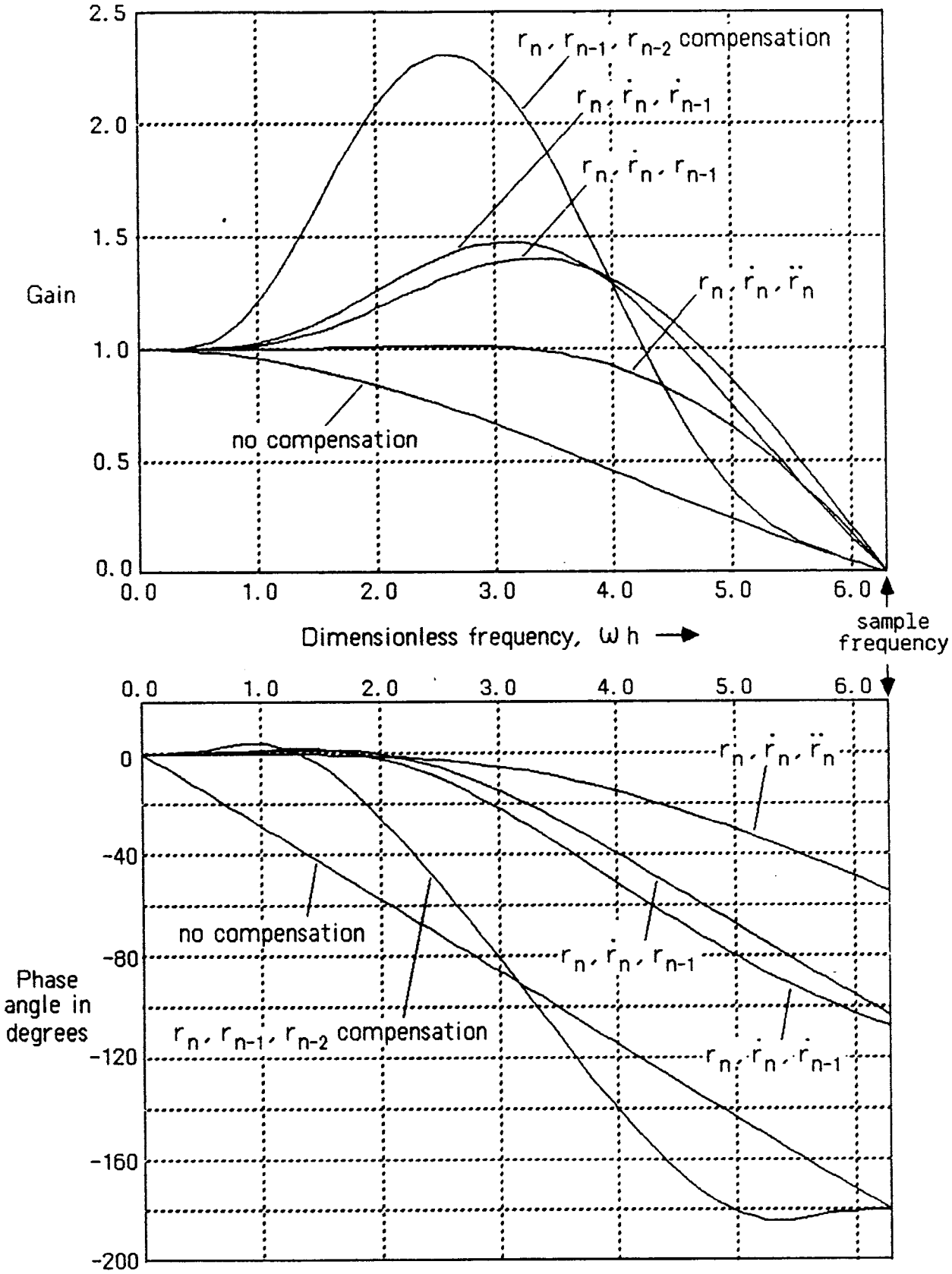



Figure 5.10. Frequency response of DAC with second—order compensation.

## 5.6 Dynamics of Analog–to–Digital Conversion

When a real–time digital simulation requires inputs from physical systems with continuous (analog) outputs, then conversion of the continuous signals to digital data sequences is required. This is accomplished for each continuous input by means of an ADC (analog–to–digital) converter). Each ADC requires a finite conversion time, usually short compared with the interval h between samples. Sometimes a single ADC is used for a number of analog input channels, in which case a multiplexer switch must be used to connect the ADC to successive analog input channels. Under these circumstances it is necessary to buffer each analog channel with a sample–hold circuit if time skew is to be avoided. Then all of the analog inputs can be sampled simultaneously and held at their sample value while the multiplexed ADC successively converts each channel to a digital signal.

With the advent of low–cost monolithic ADC's, it is now common to assign a single ADC to each analog channel, so that all conversions can proceed in parallel. Even in this case, however, a sample–hold circuit may be desirable so that the converted digital data in a given frame represents all analog signals sampled at a common time.

In this section we will examine the frequency spectrum which results when a continuous signal is sampled at a fixed sample rate. Consider the continuous signal f(t) shown in Figure 5.11a. Assume that f(t) is sampled every h seconds using a switch that is closed for $\alpha h$ seconds, where $0 < \alpha < h$ and $\alpha$ represents the switch duty cycle. Then the sampled signal f*(t) results, as shown in Figure 5.11b. We can represent f*(t) as follows:

$$f^*(t) = S(t)\, f(t) \tag{5.60}$$

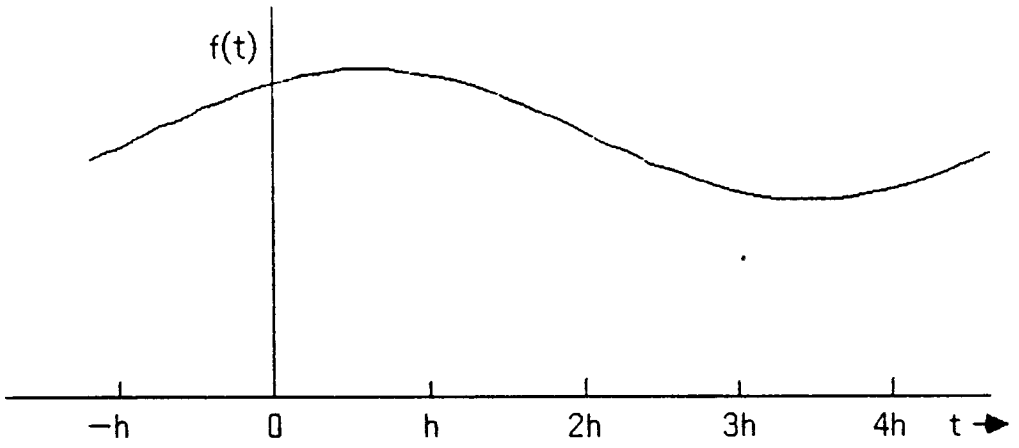
where the switching function S(t), shown in Figure 5.11c, is given by

$$\begin{aligned} S(t) &= 0 \ , \quad \tfrac{1}{2}\alpha h \leq |t-nh| \leq \tfrac{1}{2} h \\ &= 1 \ , \quad |t-nh| \leq \alpha h \end{aligned} \tag{5.61}$$

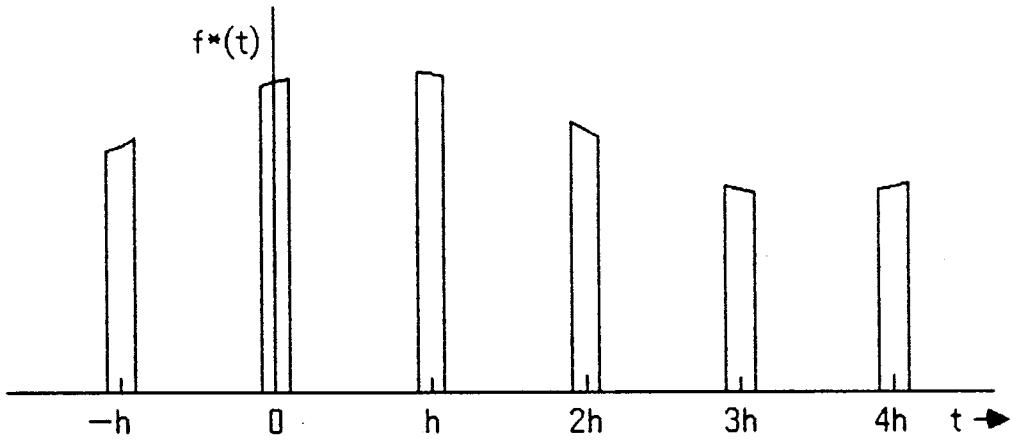
For $\alpha = 1$ the sampling process is continuous. As $\alpha \to 0$ the sampling process becomes instantaneous. For finite $\alpha$ we can expand the switching function in a Fourier series. Thus we let

$$S(t) = \sum_{k=-\infty}^{\infty} C_k e^{jk\omega_0 t} \tag{5.62}$$

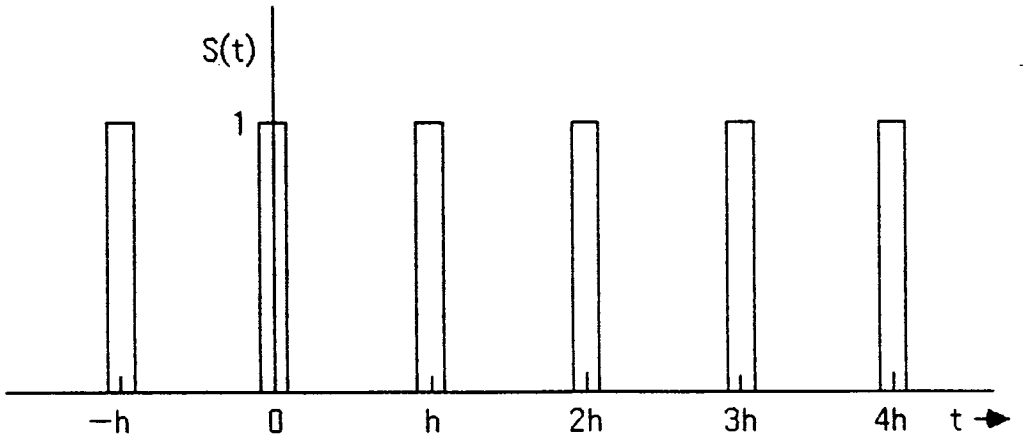
where $\omega_0 = 2\pi/h$, the sample frequency in radians per second. The Fourier coefficients are given by the formula




a. Continuous signal




b. Sampled signal




c. Switch function

Figure 5.11. Sampling process.

$$C_k = \frac{\omega_0}{2\pi} \int_{-\pi/\omega_o}^{+\pi/\omega_o} S(t)\, e^{jk\omega_0 t}\, dt \tag{5.63}$$

From Eq. (5.61) we have

$$C_k = \frac{\omega_0}{2\pi} \int_{-\alpha T/2}^{+\alpha T/2} S(t)\, e^{jk\omega_0 t}\, dt = \frac{1}{j2\pi k}\, e^{jk\omega_0 t}\Big|_{-\alpha\pi/\omega_o}^{+\alpha\pi/\omega_o}$$

or

$$C_k = \frac{e^{j\alpha\pi k} - e^{-j\alpha\pi k}}{j2\pi k} = \frac{\sin \alpha\pi k}{\pi k} \tag{5.64}$$

For $\alpha\pi k \ll 1$, i.e., $\alpha \ll 1/\pi k$, $C_k$ becomes

$$C_k \cong \alpha, \quad \alpha \ll 1/\pi k \tag{5.65}$$

Thus for a very short switch duty cycle, $\alpha$, the Fourier coefficients for the fundamental frequency $\omega_0$ and the lower harmonics ($k = \pm 2, \pm 3$, etc.,) are all equal approximately to $\alpha$.

Substituting Eq. (5.62) into Eq. (5.60), we obtain

$$f^*(t) = \sum_{k=-\infty}^{\infty} C_k\, f(t)\, e^{jk\omega_0 t} \tag{5.66}$$

To obtain the frequency domain representation of $f^*(t)$ we take the Fourier transform of $f^*(t)$. Thus

$$F^*(j\omega) = \int_{-\infty}^{\infty} f^*(t)\, e^{-j\omega t}\, dt = \int_{-\infty}^{\infty} \sum_{k=-\infty}^{\infty} C_k\, f(t)\, e^{jk\omega_0 t}\, e^{-j\omega t}\, dt$$

or

$$F^*(j\omega) = \sum_{k=-\infty}^{\infty} C_k \int_{-\infty}^{\infty} f(t)\, e^{-j(\omega - k\omega_0)t}\, dt \tag{5.67}$$

But the Fourier transform of $f(t)$ is given by

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-j\omega t}\, dt \tag{5.68}$$

Thus we can write Eq. (5.67) as

$$F^*(j\omega) = \sum_{k=-\infty}^{\infty} C_k F(j\omega - jk\omega_0) \tag{5.69}$$

Eq. (5.69) shows that the spectrum $F^*(j\omega)$ of the sample function $f^*(t)$ is a linear combination of the original spectrum $F(j\omega)$ and the original spectrum shifted by $k\omega_0$, $k = \pm 1, \pm 2, \ldots$. Furthermore, for near instantaneous sampling ($\alpha << 1$), all the translated spectra will have essentially the same amplitude ($C_k \cong \alpha$) until k becomes quite large in magnitude.

Figure (5.12) shows this result graphically, where for simplicity only the magnitudes $|F(j\omega)|$ and $|F^*(j\omega)|$ are plotted versus $\omega$. Note that for frequencies $\omega > \omega_0/2$ the spectrum $|F(j\omega)|$ of the continuous signal f(t) "folds back" on itself in the spectrum $|F^*(j\omega)|$ for the sample signal $f^*(t)$. This means that frequencies

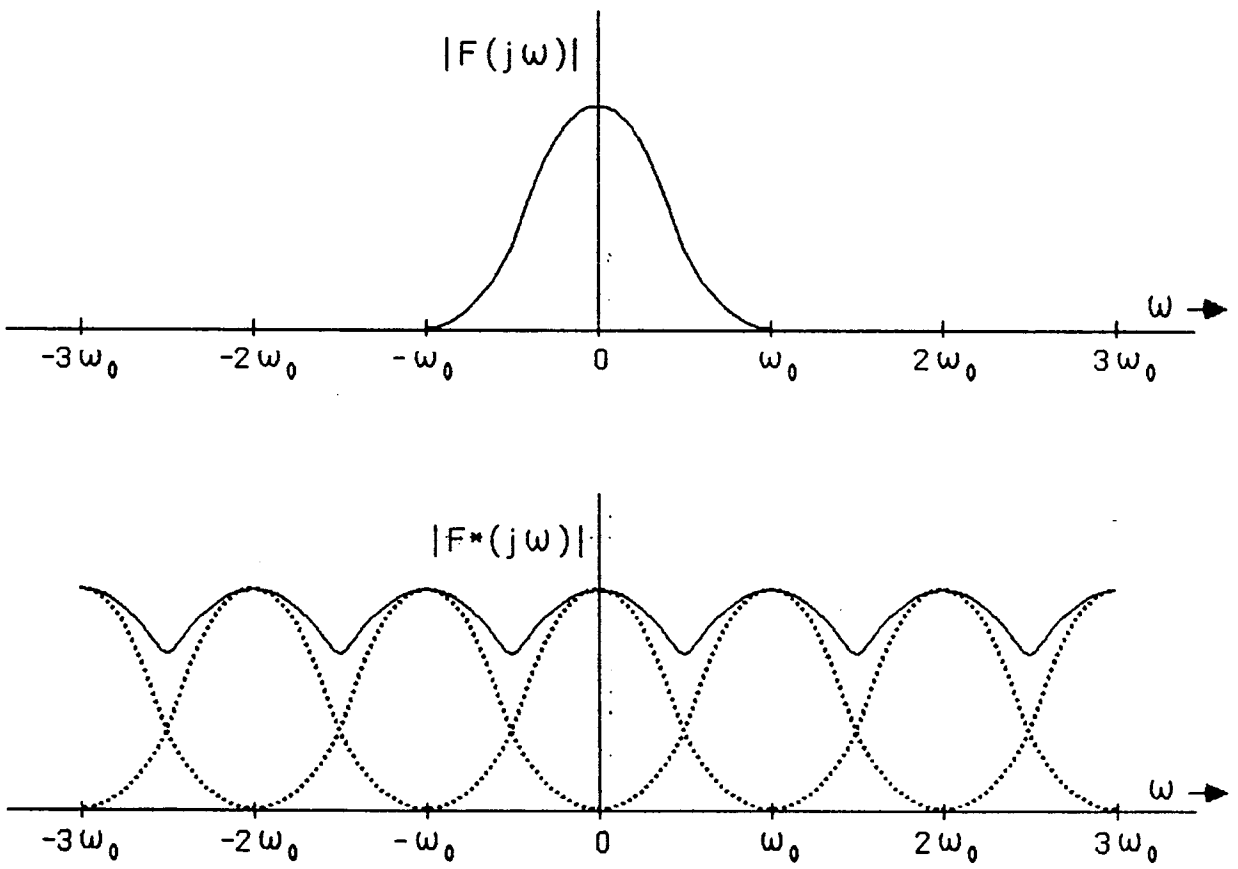



Figure 5.12. Frequency spectrum, $|F(j\omega)|$, of the continuous signal and frequency spectrum, $|F^*(j\omega)|$, of the sampled signal.

contained in f(t) which are larger than one—half the sample frequency will show up as frequencies below one—half the sample frequency in f*(t). In fact, a frequency in f(t) equal to $\omega_0$, the sample frequency, will be converted to zero frequency, i.e., a constant offset or bias in f*(t).

This foldback phenomenon, called aliasing, can be prevented by passing f(t), the continuous signal, through a low—pass analog filter before sampling. This is an important consideration in interfacing a real—time digital simulation to analog inputs. If the analog signals have significant frequency content above one—half the sample frequency, which, for example, would be the case if the analog signals contain high—frequency noise, then the signals should be passed through a low—pass filter before being sampled for A to D conversion.

Determining the order and cutoff frequency for the low—pass filter involves a tradeoff. The lower the cutoff frequency $\omega_c$, which must be below $\omega_0/2$, the larger the accompanying phase lag. In addition, frequencies in f(t) which are below $\omega_0/2$ but above $\omega_c$ will be lost. Also, the higher the order n of the low—pass filter, the larger the accompanying phase lag for a given $\omega_c$. On the other hand, the larger the order n and the lower the cutoff frequency $\omega_c$, the higher the attenuation of frequencies in f(t) above $\omega_0/2$.

An alternative to passing f(t) through an analog filter before sampling and A to D conversion is to perform the sampling and A to D conversion at a higher frame rate, one that is an integer multiple N of the original frame rate. The resulting fast data sequence is turned into a data sequence with the original frame rate by utilizing a simple digital algorithm. Thus every N successive samples from the fast data sequence are averaged to produce a single sample, which results in a data sequence at the original frame rate. The multiple frame—rate sampling followed by the averaging algorithm constitutes a low—pass digital filter. The effective time delay of this filter is h/2, where h is the period of the output data sequence. This is because the averaging algorithm must wait until all N samples over the interval h are received in order to compute the average, which represents a smoothed data value at the midpoint of the interval h.

Note that a sinusoidal data sequence $\{r_n\}$ can be viewed as a sample function r*(t) derived from a continuous signal $r(t) = A \sin \omega t$, where the duty cycle $\alpha \to 0$. Then the single spectral frequency $\omega$ contained in f(t) will be transformed in r*(t) to the frequencies $\omega \pm k\omega_0$, as shown in Figure 5.13. For frequencies above $\omega_0$ contained in r*(t) and hence $\{r_n\}$, output DAC's driven by $\{r_n\}$ will respond in accordance with the DAC transfer function $H_e^*(j\omega)$ or the compensated DAC transfer function $H_c^* H_e^*$. In the DAC frequency response curves shown in Figures 5.5, 5.9 and 5.10 the gain was plotted only up to $\omega = \omega_0$, the sample frequency. Reference to Figure 5.13 shows that the sinusoidal data sequence input to the DAC or compensat—

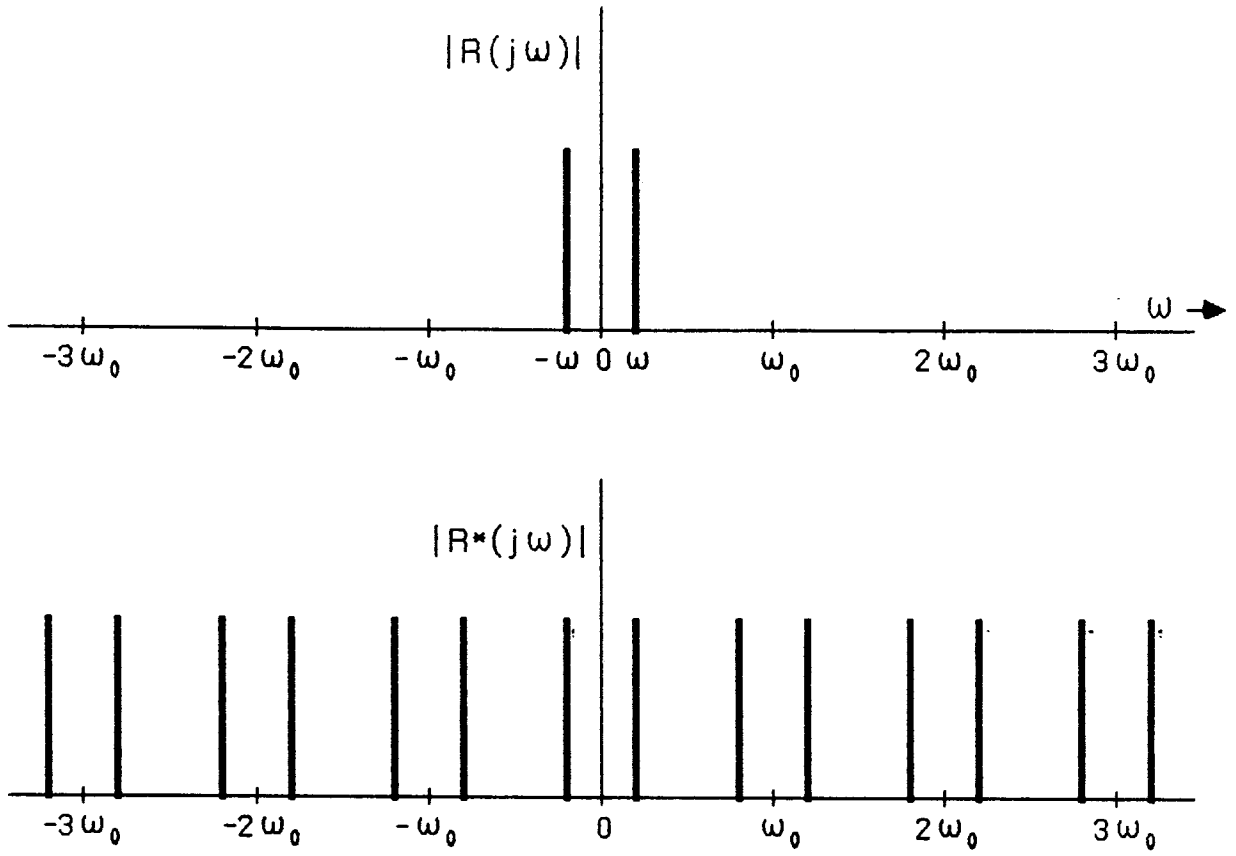



Figure 5.13. Spectrum $|R(j\omega)|$ of a continuous sinusoid of frequency $\omega$, and spectrum $|R^*(j\omega)|$ of the sinusoid when sampled at frequency $\omega_0$.

ed DAC will also contain frequencies above $\omega_0$ to which the DAC will respond. Reference to the various DAC and compensated DAC gain formulas in Sections 5.3, 5.4, and 5.5 shows that in general the gain falls off with increasing frequency (see comments in the last twp paragraphs of Section 5.5).