NASA CR-152340

SYSTEMS TECHNOLOGY, INC.

2672 BAYSHORE FRONTAGE ROAD • MOUNTAIN VIEW, CALIFORNIA 94043 • PHONE [415] 961-4674

Technical Report No. 1140-1-I

THE ANALYSIS OF DELAYS IN SIMULATOR DIGITAL COMPUTING SYSTEMS

Volume One

Formulation of an Analysis Approach Using a Central Example Simulator Model

R. K. Heffley
W. F. Jewell
R. F. Whitbeck
T. M. Schulman

February 1980

Contract NAS2-10106

National Aeronautics and Space Administration
Ames Research Center
Moffett Field, California 94035

ABSTRACT

The effects of spurious delays in realtime digital computing systems are examined. In this, the first of a two volume series, various sources of spurious delays are defined and analyzed using, as a central example, an extant simulator system. A specific analysis procedure is set forth, and four cases are viewed in terms of their time and frequency domain characteristics. Numerical solutions are obtained for three single rate one- and two-computer examples, and the analysis problem is formulated for a two-rate, two-computer example. In the second volume of this series a separate approach to the two-computer, multirate problem is examined. At the conclusion of Volume One the results of the analyses from both volumes are discussed in terms of their potential value for fidelity metrics or simulator problem detection and correction aids.

FOREWORD

The research reported here was performed under NASA Contract NAS2-10106. The NASA technical monitor was Mr. William B. Cleveland, and the Systems Technology, Inc., project engineer was Mr. Robert K. Heffley. Work on the project was conducted during the period from January 1979 through October 1979.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Concluded)

LIST OF ABBREVIATIONS

ADC	Analog-to-digital conversion
AHS	Airborne Hardware Simulator
DAC	Digital-to-analog conversion
DDC	Digital-to-digital conversion (also D-to-D)
I/O	Input/Output
NASA	National Aeronautics and Space Administration
ZOH	Zero order hold
1819	Sperry 1819 Digital Flight Control Computer
8400	EAI 8400 General Purpose Digital Computer

LIST OF SYMBOLS

a	Engine response constant
a_o	Difference equation coefficient
a_1	Difference equation coefficient
b_1	Difference equation coefficient
b_2	Difference equation coefficient
c	Sampled quantity, f^T
d	Flight path displacement
d_c	Commanded flight path displacement
$\dot{d}$	Flight path rate, $\frac{d}{dt}$ (d)
$\ddot{d}$	Flight path acceleration $\frac{d^2}{dt^2}$ (d)
e	2.71 ...
f	Argument of sampled quantity
G	Transfer function
j	$\sqrt{-1}$
K	Gain in flight control computer
M	Data hold operator
M_o	Zero-order hold
M_2, M_3	Zero-order holds in Case 4
n	Finite difference equation index, also a sampling factor in multirate analysis
p	Discrete transform, in Case 4 $p \triangleq z^6 = e^{-sT}$
R	Input variable
s	Laplace transform
t	Time
T	Sampling period

LIST OF SYMBOLS (Concluded)

W, W_*	Delay and advance operator matrices
x	Input variable
X	Input variable
y	Output variable
Y	Output variable
z	Discrete transform $= e^{-sT}$
z_n	Discrete transform, $e^{-sT/n}$
Z_w	Aerodynamic heave damping derivative
Z_{δ_T}	Throttle control derivative
π	3.14 ...
ω	Frequency

SUBSCRIPTS

c	Command

SUPERSCRIPTS

T	Sampled quantity

SECTION I

INTRODUCTION

The objectives of this report are to examine the effects of various spurious delays in real-time digital computing systems and to explore useful simulation fidelity metrics and procedures for obtaining them. In this effort, the emphasis is placed on digital computing systems involving two computers in which there may be several forms of delays along with the complications due to multirate or multiloop operation.

The work reported here is an initial effort to deal with delays in digital simulator systems. Therefore considerable space is devoted to the categorization of sources of spurious delays and lags and the formulation of analysis procedures. Although several non-trivial examples are considered, they are not particularly pathological so far as system fidelity is concerned. These examples do provide good illustrations, however, of how simulator computers can produce a variety of artifacts and how those artifacts may be analyzed.

Particular kinds of spurious digital effects considered include those associated with computer timing (the sequence of instructions and operations), digital computer algorithms, and computer interfacing. Each of these aspects is examined using a central example of a mathematical model representing a portion of an aircraft and digital flight control computer combination. This same model is used to demonstrate the formulation of a technique capable of handling two simulator computers running at different frametimes. All of these above items are included in this, the first volume of a two-volume series.

The second volume presents an alternative procedure for examining the multirate or multiloop digital computer problem. It is convenient to put this second procedure in a separate volume because it does not follow the central example of Volume One. Nevertheless both methods will be discussed in the concluding section of this, the first volume.

One particularly noteworthy accomplishment made during this program was the development of analysis software which permits the computation and plotting of digital system frequency response and steady-state time response. Although the software is not documented in this report, its potential role in simulation validation or verification is illustrated.

SECTION II

TECHNICAL APPROACH

The purpose of this section is to describe the approach taken in the analysis of the effects of delays in simulator computing systems.

The main concern of this report is the analysis of simulator computer systems composed of two digital processors operating in series or within a feedback loop. This concern is reflected in the discussion of spurious digital effects, the definition of an example simulator model, and the description of an extant simulator system to be used with that model. The technical approach thus formed is concluded with a brief treatment of how the analysis cases will be formulated in the next section.

A. DESCRIPTION OF SPURIOUS DIGITAL EFFECTS

Several forms of spurious digital effects connected with lags and delays are addressed in this study, but roundoff and truncation effects are specifically excluded. The particular digital effects of interest include the following:

- Transport delay
- Data skewness
- Algorithmic delay and distortion
- Multiloop anomalies (two or more synchronized digital processors operating at nearly the same sample rates)
- Data exchange between two or more digital processors.

Each of these effects is described below along with an indication of the appropriate analysis approach.

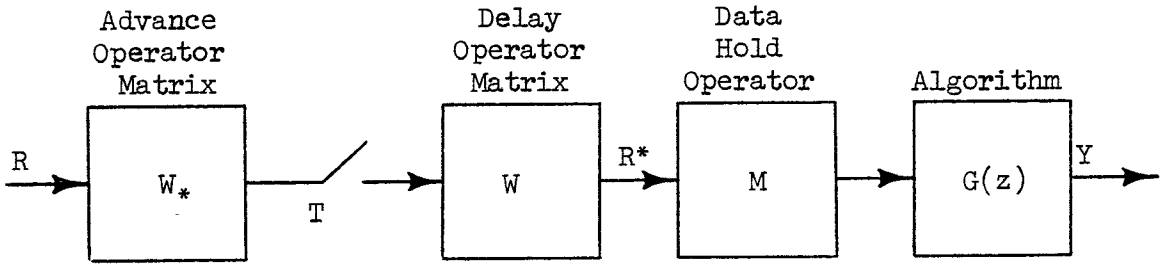
1. Transport Delay

Transport delay is the pure time delay associated with a digital system. That is, in a simulation system involving an independent digital processor, we must be willing to tolerate a delay of at least one frame time before all algorithms implemented in that particular digital processor are updated and their outputs delivered in response to a set of input samples. The process of updating and holding a sample for one frame time T is modeled with a zero order hold:

$$M_o(s) = \frac{1 - e^{-sT}}{s} \qquad (1)$$

2. Data Skewness

Data skewness is the time delay associated with serially transmitted data. For example, input data skewness is due to multiplexing the analog-to-digital converters (ADC). Thus the data input to the digital computer are sampled at different points in time. A vector linear model for data skewness is shown below. The vector form of the switch decomposition analysis method (Ref. 1) can be used to model data skewness. The technique is briefly described in the following paragraphs.



Vector Linear Model for Data Skewness

Suppose the vector signal, R, in Fig. II-1 consists of p components, R_1, R_2, ..., R_p. Let T be the frame time. Suppose R_1 is sampled first and sent to its storage register T/N seconds after the start of a frame. Then T/N seconds later R_2 is sampled and sent to its storage register, and the process continues until the p components have been stored once in each frame time. Then the vector R in the model of Fig. II-1 has the components

$$R = [R_1, R_2, \ldots, R_p]' \tag{2}$$

where the prime superscript denotes transposition. The output vector R* is given by

$$R^* = W\,(W_* R)^T \tag{3}$$

where superscript T denotes the impulsive sampling operator. The delay operator matrix W is given by

$$W = W(s) = \mathrm{diag}\left[e^{-sT/N},\ e^{-(2/N)sT},\ \ldots,\ e^{-(p/N)sT}\right] \tag{4}$$

and the advance operator matrix by

$$W_* = W_*(s) \triangleq W'(-s) \tag{5}$$

where the prime superscript again denotes transposition†. W_* and W thus model data skewness while permitting the mathematical convenience afforded by the common vector impulsive sampling switch. The transport delay which is common to all of the stored data is modeled by the data hold operator M(s).

---

† Although $W'(-s) = W(-s)$ in this special case where each component of R is sampled only once in each frame time, transposition within each of the diagonal elements of W(-s) is necessary if each component of R is sampled more than once in each frame time. Vector switch decomposition requires the z-transform of functions which are advanced in time by some fraction, Δ, of the least common sampling interval. Ref. 1 contains additional details.

Alternatively, in the example given above, R_1 need not have been sampled (or transferred) immediately following the start of a frame. Suppose instead that R_1 is sampled at $(L+1)T/N$ seconds and that subsequent samples are transmitted serially T/N seconds later as before. The delay operator matrix W is alternatively given by

$$W_{(alt)} = \text{diag}\left[e^{-[(L+1)/N]sT},\ e^{-[(L+2)/N]sT},\ \ldots,\ e^{-[(L+p)/N]sT}\right] \quad (6)$$

where $0 < (L+p)/N \leq 1$ and the advance operator matrix by

$$W_{*_{(alt)}} = W'_{(alt)}(-s) \quad (7)$$

The vector linear model of data skewness described above can also be used to analyze computational and output skewness. Computational skewness is due to the digital computer serially processing the input data used to compute the output of a particular transfer function. Output skewness is due to the serial transmission output data. Both computational and output data skewness are usually negligible because the processing times are small fractions of the total frame time T (i.e., microseconds versus milliseconds).

3. Algorithmic Delay

Algorithmic delay is most commonly the delay associated with the method used to transform a continuous transfer function, G(s), into a discrete transfer function, G(z). There are numerous methods available for performing the transformation and hence there are numerous forms of G(z) for any given G(s).

To demonstrate how the algorithm G(z) can be used to describe the same G(s), but with varying delay, an example of scalar numerical integration will be used. The Laplace transform of a zero order hold in $M_o(s)$ followed by a continuous integration operator in the real world prototype for G(s) is

$$M_o(s)G(s) = \frac{1 - e^{-sT}}{s^2} \quad (8)$$

where T is the frame time. This transfer function represents the particular timing case in which the input, computation, and output occur simultaneously, i.e., there is no delay due to input/output skewness. The sampled-data reconstruction and continuous integration operations represented by Eq. 8 can be advanced in the time domain by a time ΔT (where Δ is the numerical advance ratio) in order to overcome part of the delay contributed by the zero order hold, part of the delay introduced by computational timing, and part of the disparity between the computed derivative and the state. The Laplace transform of the operations thus advanced in time is $M_o(s)G(s)e^{\Delta Ts}$. A corresponding family of familiar elementary numerical integration algorithms can be represented by $G(z,\Delta)$, the z-transform of the discrete time sequence resulting from sampling (with period T) the reconstruction and integration operation advanced in time†. $G(z,\Delta)$ is called the advanced z-transform (Refs. 2, 3, and 5) and, for the pure integration example, is given by

$$G(z,\Delta) = \frac{T[\Delta z + (1 - \Delta)]}{z-1} \tag{9}$$

If we identify the z-transform of the sequence of discrete inputs to $G(z,\Delta)$ in Eq. 9 as $X(z)$ and the z-transform of the sequence of discrete outputs as $Y(z,\Delta)$, then the numerical recursion equation in the discrete time domain corresponding to $Y(z,\Delta) = G(z,\Delta)\, X(z)$ is

$$y_{n+1} = y_n + T\left[\Delta\, x_{n+1} + (1 - \Delta)\, x_n\right] \tag{10}$$

This recursion formula represents a family of numerical integration algorithms, each of which can be identified by specifying a particular advance ratio, Δ. FORTRAN statements corresponding to various values of Δ are listed in Table 1. The question which now arises is, which is best? We can determine which is best by a direct frequency response comparison between the ideal integrator, $G(s) = 1/s$, and any of the possible digital algorithms, $\frac{1}{T}G(z,\Delta)$. But first we must define the computer <u>timing</u>.

---

† For the sake of brevity we will redefine $G(z,\Delta) \triangleq M_o G(z,\Delta)$.

TABLE 1

FORTRAN STATEMENTS
FOR NUMERICAL INTEGRATION ALGORITHMS

Advance Ratio	FORTRAN Statement	Common Name†
0	Y = Y + T * XPAST	Euler
0.5	Y = Y + T * (X/2 + XPAST/2)	Trapezoidal
1	Y = Y + T * X	Rectangular
1.5	Y = Y + T * (3*X/2 - XPAST/2)	Second order Adams

where XPAST is the first past value of the integrand X

† Names of numerical integration algorithms vary depending upon the textbook source and the context in which the algorithm is used. The most precise label includes a statement of the recursion equation and the computational timing.

As we mentioned, the recursion formula (Eq. 10) represents only one very special input/output timing case. Namely, Eq. 10 applies to an essentially simultaneous input of y, computation of x, and output of x. As we shall see in Subsection C, this particular timing situation is not commonly used in simulation. Rather there is usually a full frame time delay added because output of data follows all computations — either at the end of the frame or just after the start of the next frame. These timing situations are illustrated in Fig. 1.

While computer timing will be addressed in the general analysis procedure prescribed in Subsection D, let us proceed by considering $G(z,\Delta)$ frequency response for the two timing cases mentioned. For simultaneous input/output, $G(z,\Delta)$ applies; and for a one frame time delay between input and output, $G(z,\Delta)z^{-1}$ applies.

The algorithmic delays associated with the various advance ratios can be demonstrated by computing the frequency responses of the various $G(z,\Delta)$. This can be done by substituting

$$z = e^{j\omega T} = \cos \omega T + j \sin \omega T \tag{11}$$

into Eq. 9. The frequency response is:

For simultaneous input/output,

$$\frac{1}{T} G(e^{j\omega T}, \Delta) = (\Delta - 0.5) - j(0.5 \cot \frac{\omega T}{2}) \tag{12}$$

and for output following input by T,

> The same function but <u>multiplied by $e^{-j\omega T}$</u>
> (hence amplitude is unaffected, but phase is decreased by ωT radians).

The frequency responses obtained from Eq. 12 are compared to a continuous integrator in Fig. 2, and, in addition to the four values of Δ defined in Table 1, a fifth value of Δ is considered, i.e.,

$$\Delta = 0.5 + \frac{1}{\pi} = 0.818310 \tag{13}$$

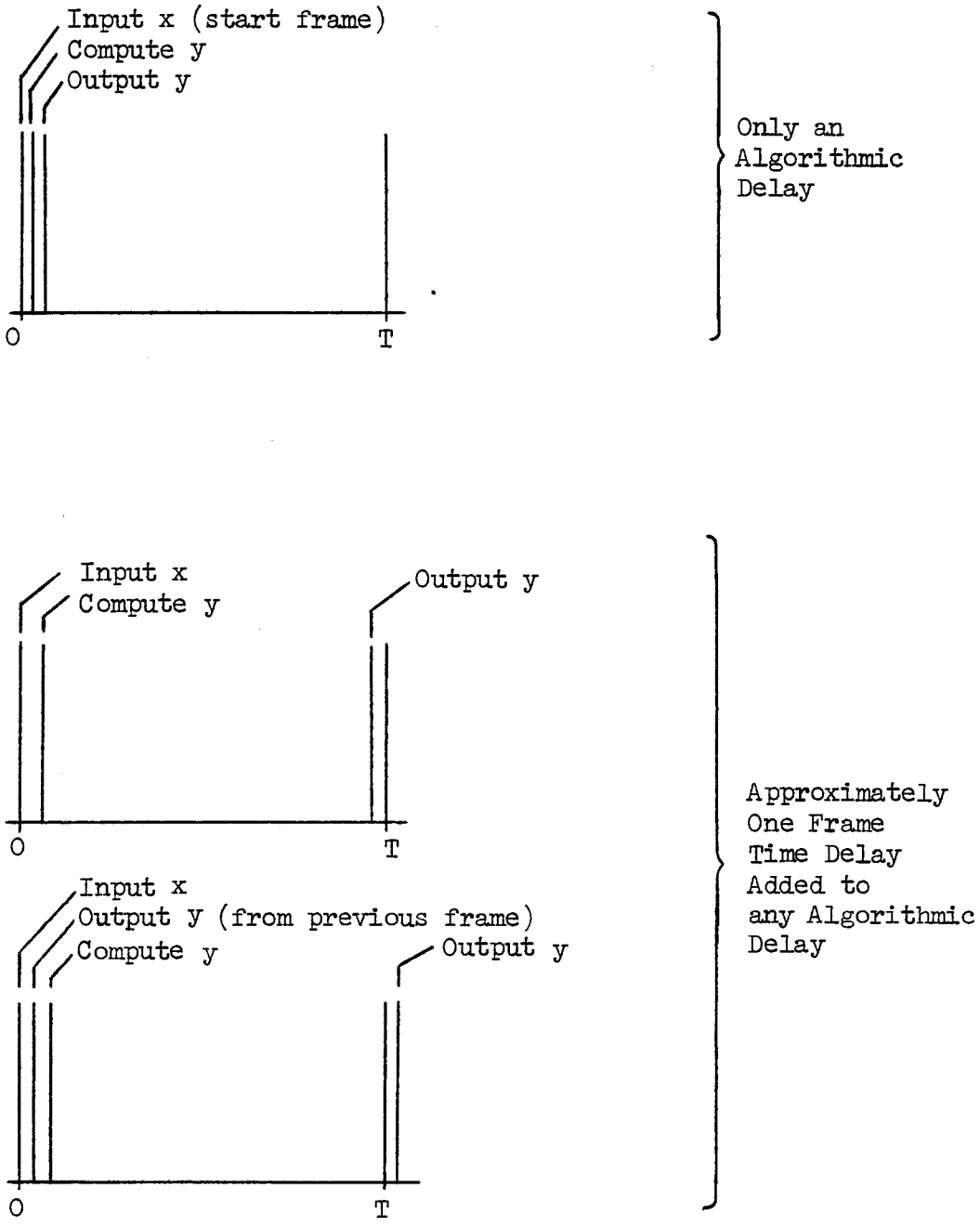


Figure 1. Various Computational and I/O Timing Situations

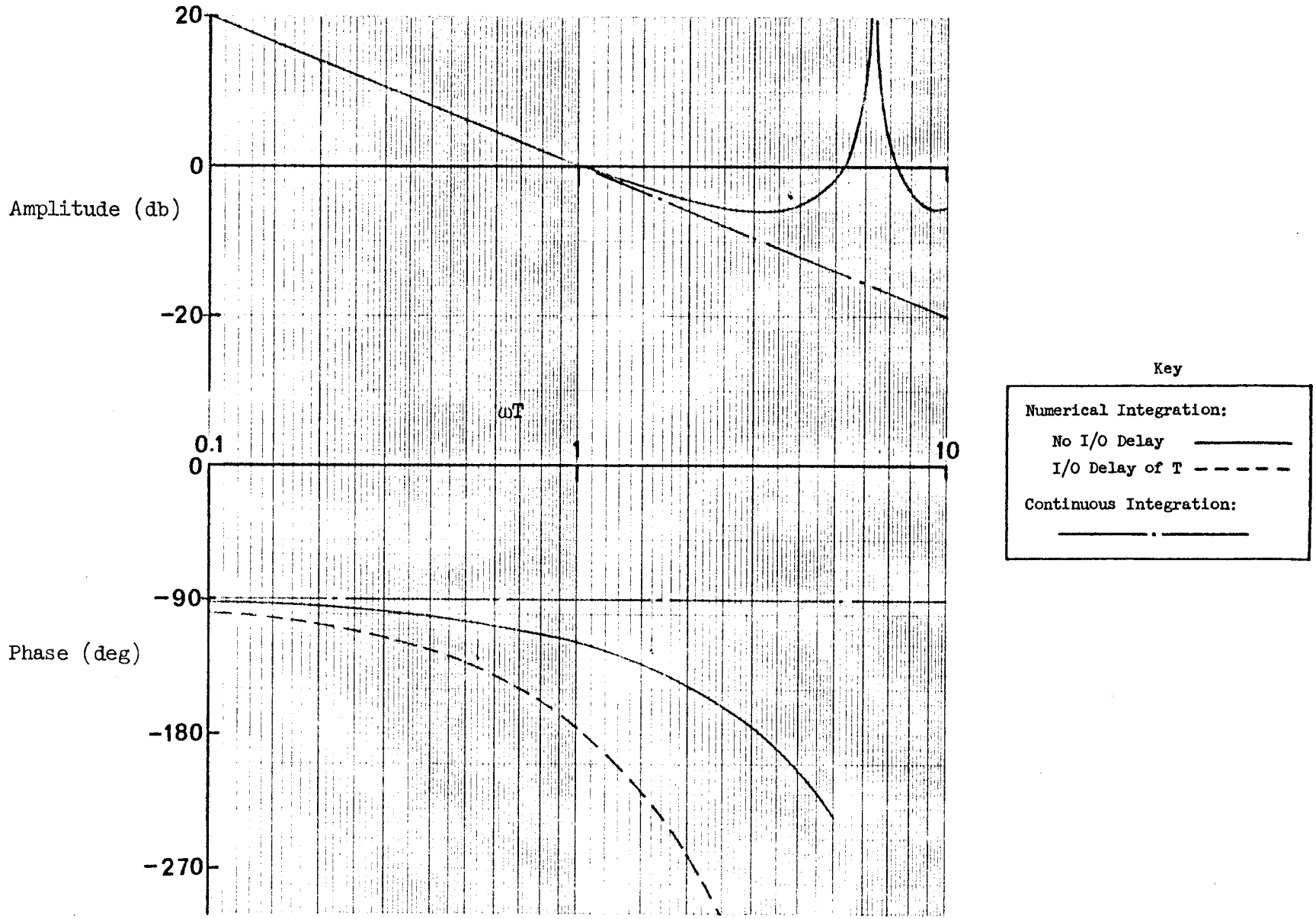


Figure 2. Numerical Integration Frequency Response
a. $\Delta = 0$ (Euler)

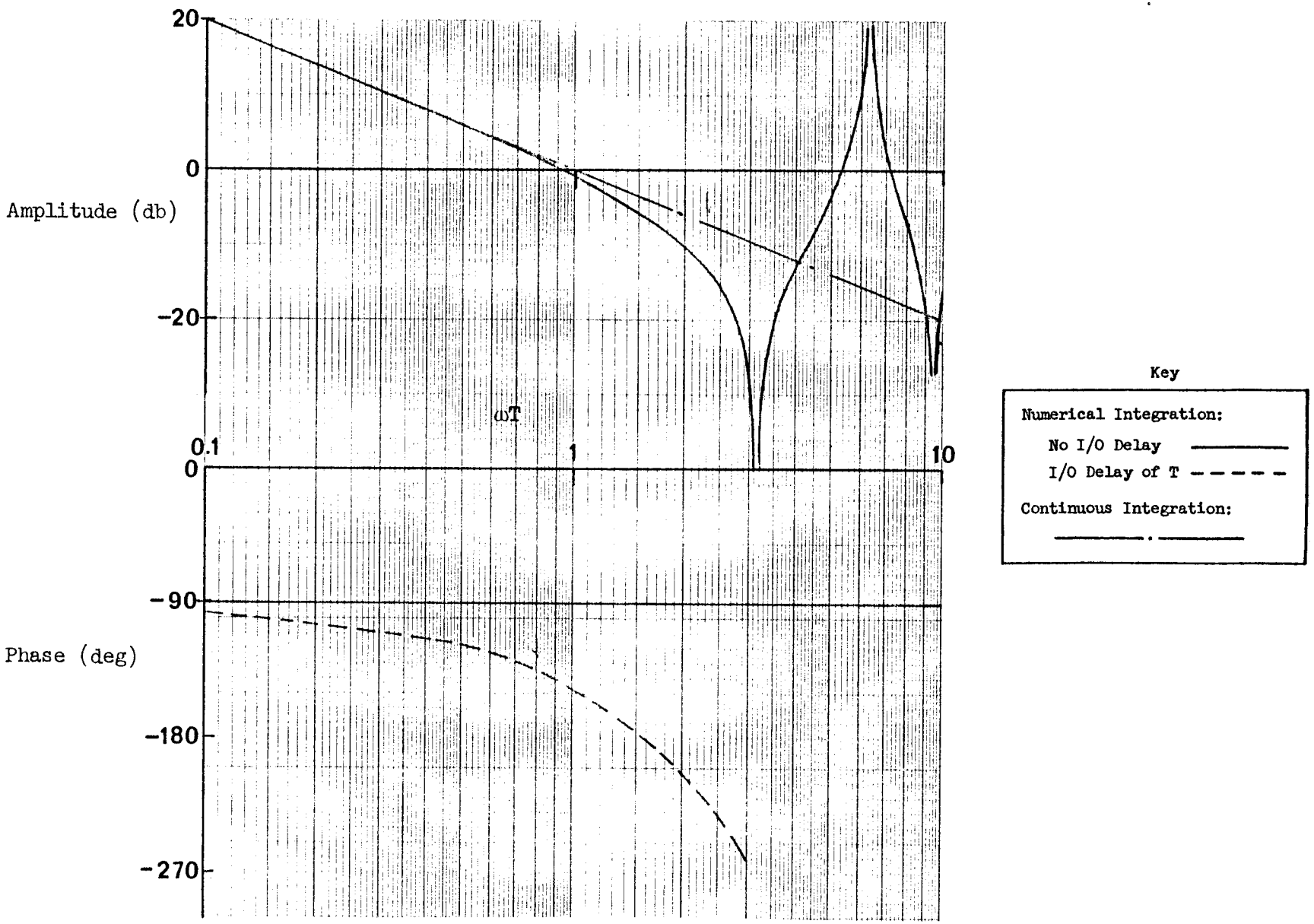


Figure 2 (Continued)
b. $\Delta = 0.5$ (Trapezoidal)

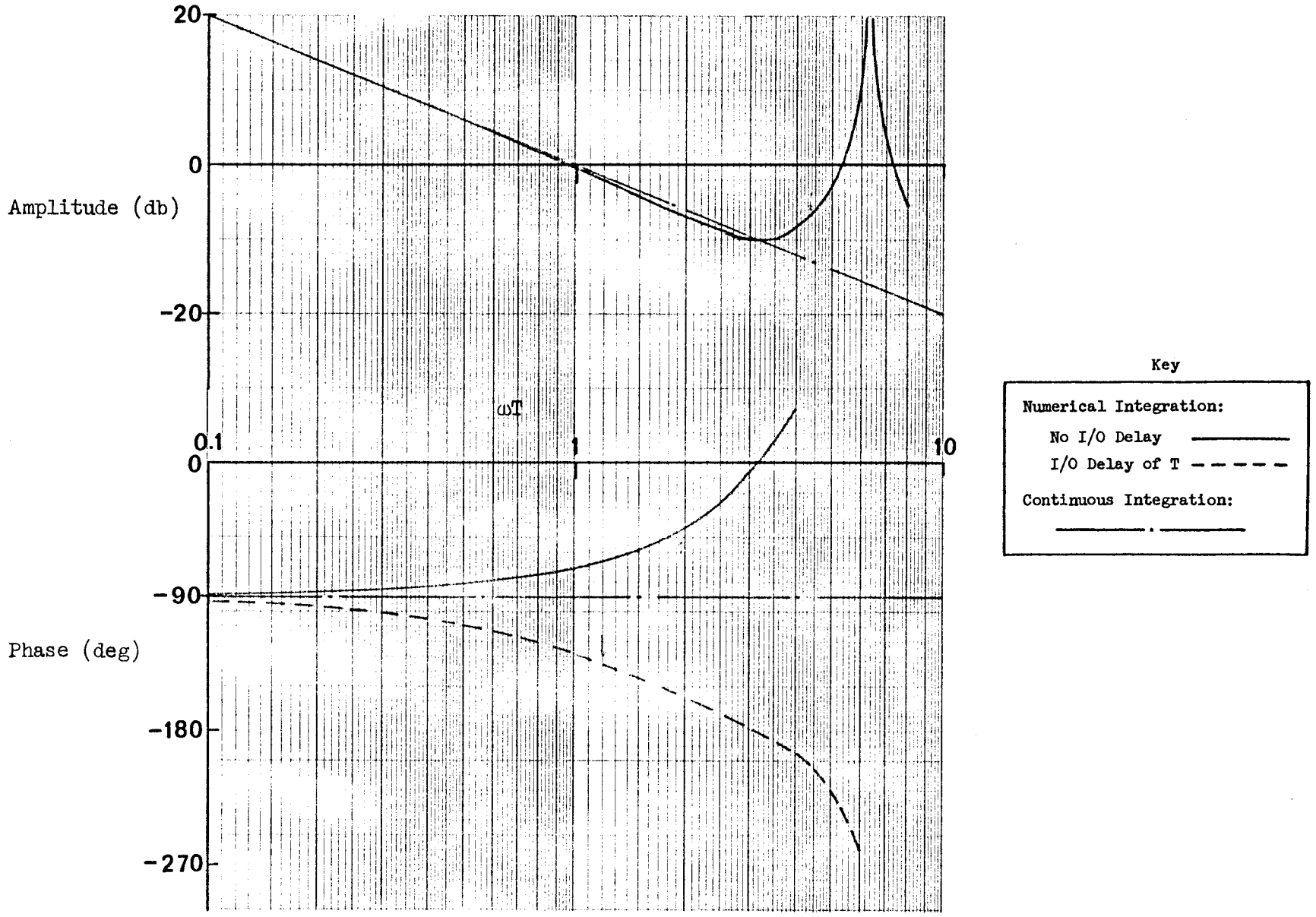


Figure 2 (Continued)

c. $\Delta = 0.5 + \frac{1}{\pi}$

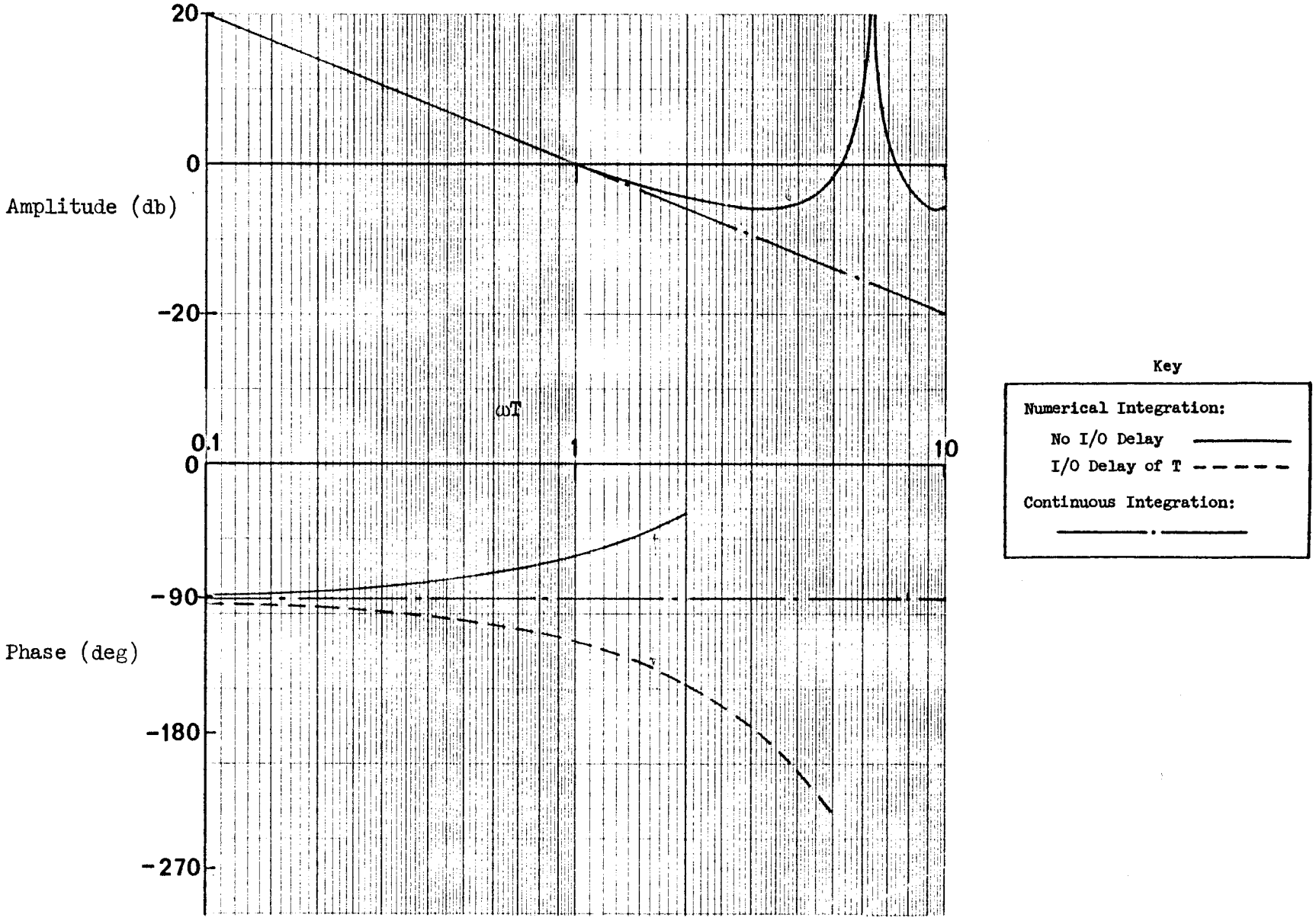


Figure 2 (Continued)
d. $\Delta = 1$ (Rectangular)

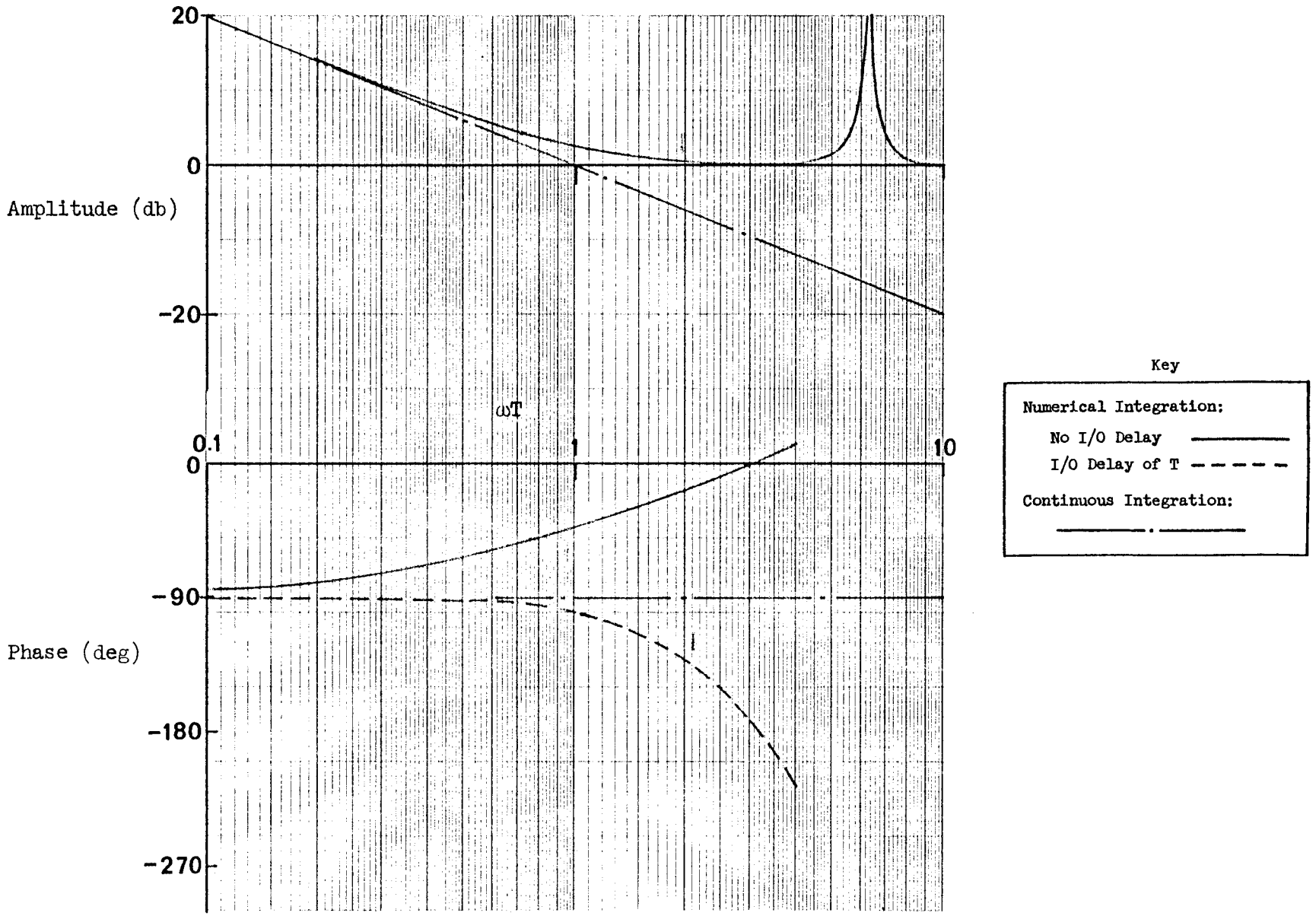


Figure 2 (Concluded)
e. $\Delta = 1.5$ (Second Order Adams)

This value of Δ is obtained by equating the amplitude ratio of Eq. 12 to that of the continuous integration at $\omega T = \pi$ (the so-called folding frequency).

What does Fig. 2 tell us about the various numerical integration algorithms? First, for ωT less than 0.3, all combinations of Δ and input/output timing are about equal and compare well to an ideal continuous integrator. Beyond ωT equal to 0.3 the amplitude and phase distortions can become significant, and there can be an advantage gained by a judicious choice of the advance ratio, Δ. Clearly, some schemes are best when no delay is involved, others are better with a delay. We shall summarize this shortly.

One additional factor which frequently enters the simulation picture is double integration. The most notable example is the double integration of acceleration in order to get displacement. This can involve two single integration algorithms but only one delay if output follows input by one frame time, T. Following the procedure developed for a single integration, the frequency response of two numerical integrations in series with an input/output delay of T can be computed by:

$$\frac{1}{T} G_1(z,\Delta_1) \cdot G_2(z,\Delta_2)\, z^{-1} \qquad (14)$$

The plotted results for two possible combinations of Δ_1 and Δ_2 are shown in Fig. 3. The case of $\Delta_1 = 1.5$ and $\Delta_2 = 0.5$ corresponds to the commonly used second order Adams and trapezoidal combination. The other case, $\Delta_1 = 1$ and $\Delta_2 = 0.5 + \frac{1}{\pi}$, was suspected to be a reasonable approach based on the compensating features observed in their single integration response (Fig. 1).

A summary of numerical integration performance for all cases considered is given in Table 2. This performance is based on amplitude and phase deviations from ideal continuous integrations (6 db and 45 deg, respectively).

For an integration without other sources of delay a trapezoidal appears the best of those considered (good to $\omega T = 2.3$) with a delay of T, the so-called second order Adams looks good (to $\omega T = 2$). However, for a double

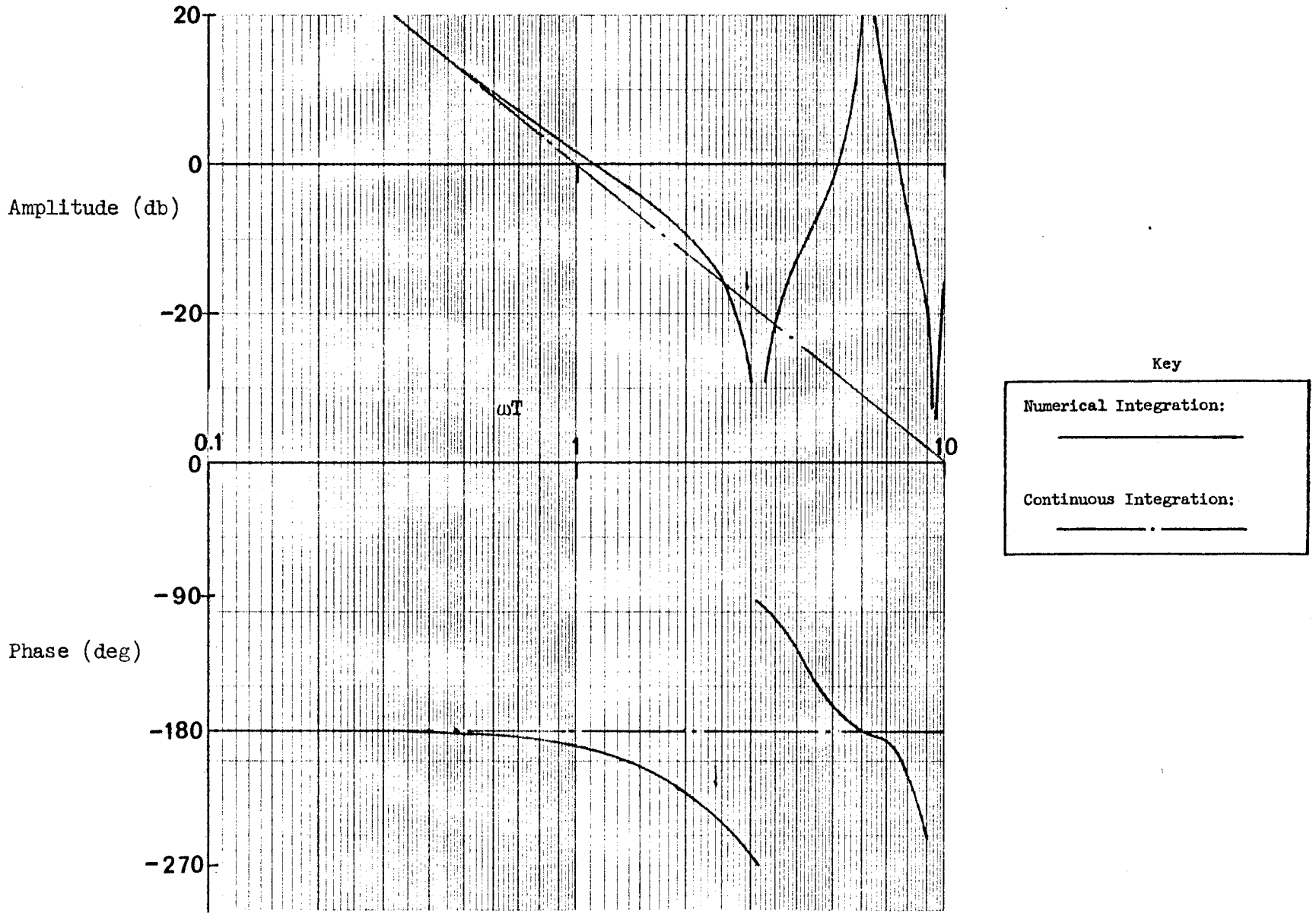


Figure 3. Numerical Double Integration Frequency Response
a. $\Delta_1 = 1.5$ and $\Delta_2 = 0.5$

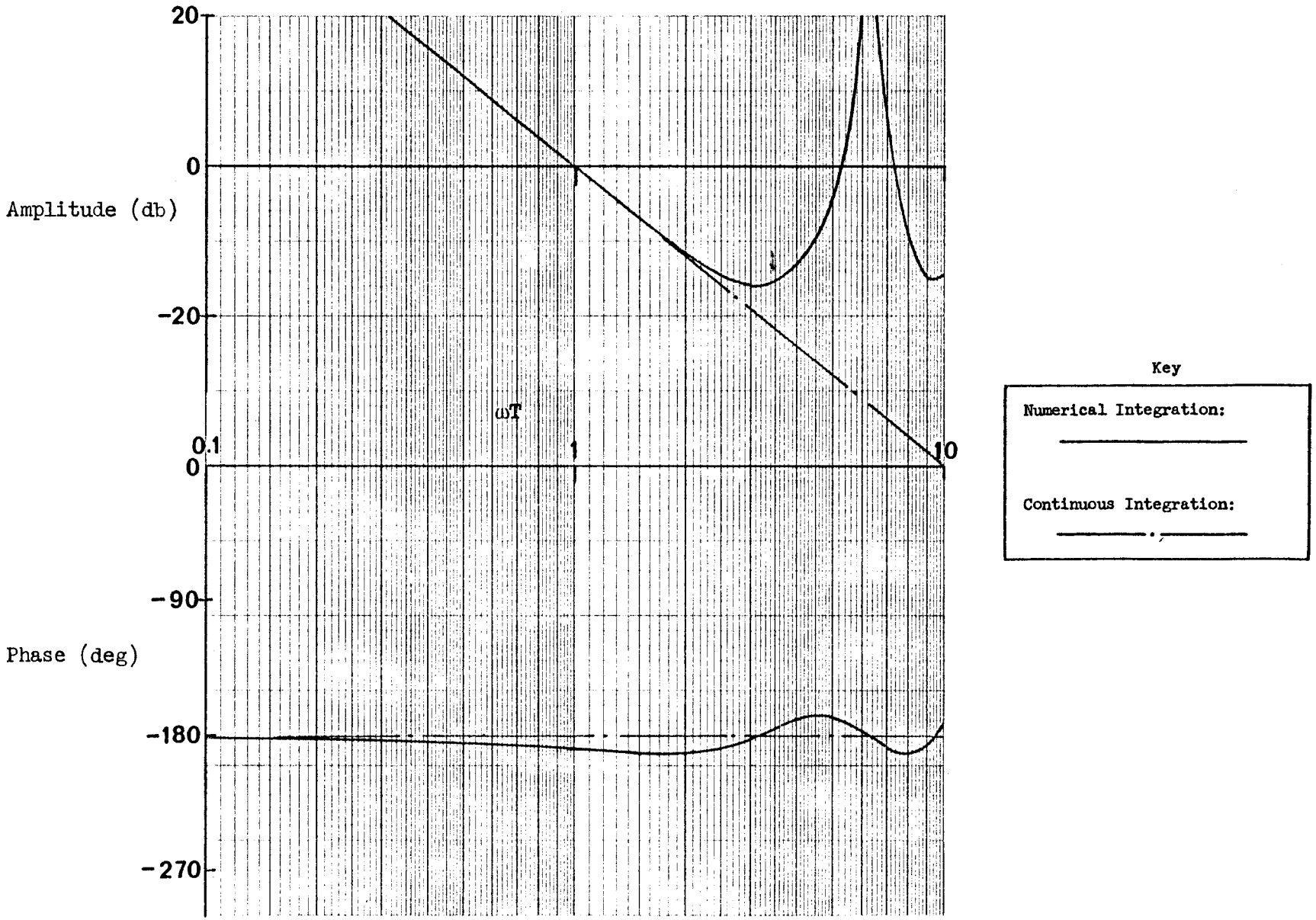


Figure 3 (Concluded)

b. $\Delta_1 = 1$ and $\Delta_2 = 0.5 + \frac{1}{\pi}$

TABLE 2

SUMMARY OF NUMERICAL INTEGRATION ALGORITHM PERFORMANCE

Case		ωT Performance			Suitability
		Amplitude*	Phase† (no delay)	Phase† (T delay)	
Single Integration	$\Delta = 0$ (Euler)	3.7	1.6	0.6	$\omega T < 1.6$, no delay
	$\Delta = 0.5$ (trapezoidal)	2.3	∞	0.8	$\omega T < 2.3$, no delay#
	$\Delta = 0.5 + \frac{1}{\pi}$	4.4	2	1.2	$\omega T < 2$, no delay
	$\Delta = 1$ (rectangular)	3.7	1.6	1.6	$\omega T < 1.6$, delay or not
	$\Delta = 1.5$ (Adams)	2.0	1.0	2.1	$\omega T < 2$, delay#
Double Integration	$\Delta = 1.5, 0.5$ (Adams plus trapezoidal)	2.9	—	2.4	$\omega T < 2.4$, delay
	$\Delta = 1, 0.5 + \frac{1}{\pi}$ (rectular plus $\Delta = 0.81831$)	3.4	—	> 10	$\omega T < 3.4$, delay#

* The value of ωT below which there is less than 6 db amplitude deviation from the ideal continuous response.

† The value of ωT below which there is less than 45 deg phase deviation from the ideal continuous response.

\# Most suitable for a particular situation based on performance.

integration with a delay of T, a combination composed of rectangular ($\Delta = 1$) and $\Delta = 0.5 + \frac{1}{\pi}$ is better than the above two (to $\omega T = 3.4$).

4. Multiloop or Multirate

Multiloop or multirate architecture is often used to conserve digital computer resources by updating different elements of a simulation at different rates. This, in turn, creates a requirement for evaluating the artifacts introduced by the multirate environment — a task which tends to be complicated by the use of a set of "modifying rules."

One tool which can be applied to the multirate situation is vector switch decomposition — already mentioned in connection with data skewness. This requires the two (or more) rates to be classified in terms of a least common factor with vector switch arrays having sizes equal to the ratios of frame times and the least common factor. For example, consider two rates, 50 msec and 70 msec. The least common factor is 350 msec. Therefore the two vector switch arrays involved will have the dimensions 5 and 7.

The vector switch decomposition approach is currently believed to be appropriate for the solution and analysis of the two-computer, two-rate central example which will be described in the next section.

A second general method for formulating and analyzing multirate systems is presented in Volume Two. The method is then applied to the second order system shown in Fig. 4. Stability criteria for the special case of $N = 2$ are also discussed. Also it may be possible to use the methods described in Ref. 6 to develop diagnostic tools for analyzing the stability of extant multirate systems (e.g., Ref. 7).

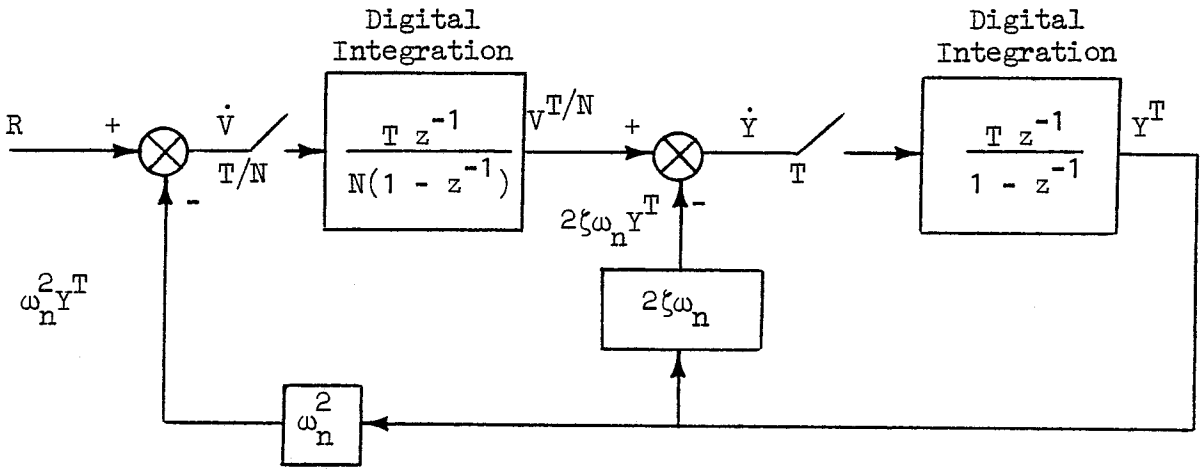


Figure 4. A Simple Two-Rate Representation of a Second Order System which is to be Simulated in a Digital Computer

5. Frame Slip

Frame slip is associated with data transfer between two or more independent asynchronous digital processors. Each independent computer may have a slightly different frame time from the others in the network. For example, one computer might have a frame time of 40 msec and another, 39 msec. Every so often (about every 1.56 sec), the faster computer will have executed one more full frame than the slower computer. This constitutes the effect called "frame slip," which can produce undesirable transport delay jump phenomena in the dynamic system being simulated.

The transport delay jump phenomenon is illustrated by an example of an actual case study presented in Ref. 8. The example applies all of the foregoing models for data skewness and computation delay to analyze two interconnected but independent digital processors which exhibit "frame slip."

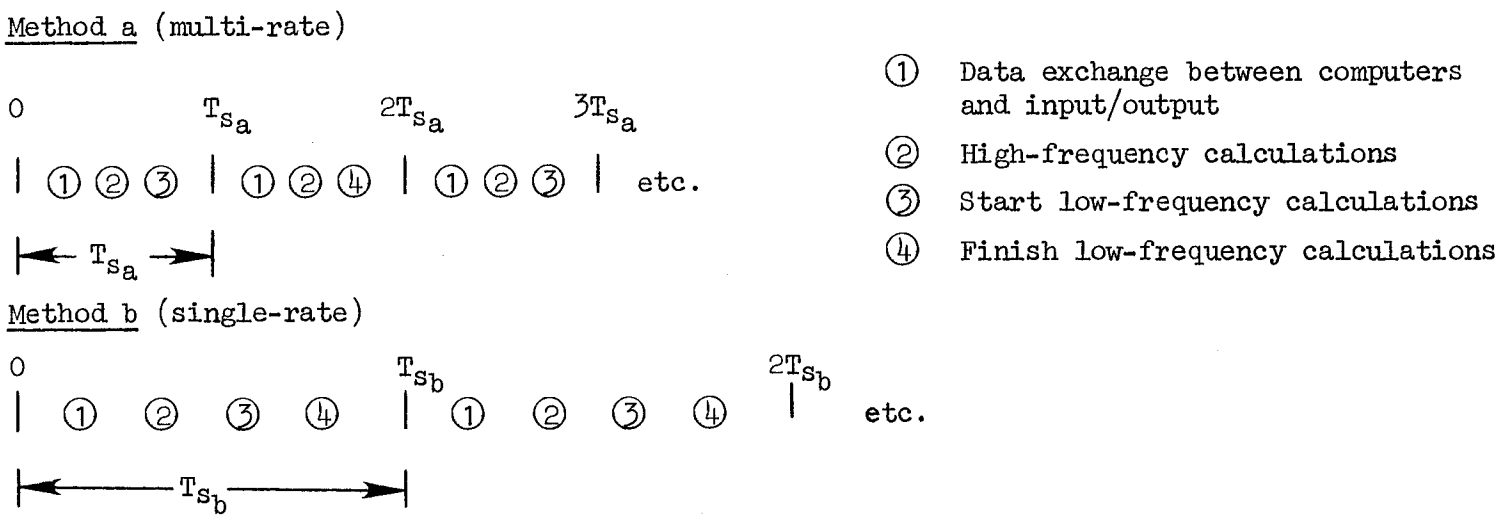
6. Data Exchange

Data exchange analysis involves using several of the tools described above. These tools can be used to perform tradeoffs of competing data exchange methods.

Figure 5 shows two possible methods. It is assumed that the simulation computer frame time is T_s and the control computer frame time is T_c. Method a uses a frame time, T_{s_a}, by implementing the data exchange, input/output, and calculations in a multirate environment. The advantage of method a is that the values of T_{s_a} and T_c can be the same (or possibly only slightly different). The disadvantage is that the aircraft equations of motion are subject to anomalies associated with a multirate simulation. Method b uses a longer frame time, T_{s_b}, and requires only a single-rate. The values of T_{s_b} and T_c are unequal and the aircraft equations of motion are implemented in a larger frame time.

This concludes our description of the various spurious digital effects to be analyzed. The next subsection describes a central example that will be used to demonstrate these effects.

Two Possible Time Lines for the Simulation Computer:



One Possible Time Line for the Control Computer:

Figure 5. Two Possible Methods of Data Exchange Between Two Computers

B. AN ILLUSTRATIVE SIMULATOR MODEL EXAMPLE

The simulator software example presented in Fig. 6 will be used as a central example to illustrate as many as possible of the potential spurious digital effects listed in the previous subsection. The plant shown within the general purpose simulator computer block is a reduced order model of the flight path dynamics of a V/STOL-type aircraft. The digital autopilot in the special purpose guidance and control computer block regulates flight path and provides suitable compensation to improve the flight path dynamics.

The continuous system descriptions contained in each of the two computer blocks in Fig. 6 represent the equations of motion from which the digital computer programmer typically works. Clearly, the systems diagrammed here could be programmed in a number of ways, one of which is presented in Table 9. In this example the airframe, propulsion system, and autopilot are "called" in subroutines, and the states are integrated (and differentiated) in typical fashion for real time simulation. There is an intent, of course, to show a direct parallel to the simulator programming practices commonly used at Ames Research Center.

The simulator model example shown here provides a convenient and flexible point of departure from which to explore digital-processor-related lags and delays and their effects. As mentioned previously, these include two-computer interactions, multiloop (within a computer), multirate (between computers), algorithmic effects, and timing effects.

The specific use of the software example in exploring the various digital effects listed in Subsection A is diagrammed below. The main features to be studied can be lumped into the autopilot block (G_1), airframe/propulsion block (G_2), autopilot frame time (T_1), or airframe/propulsion frame time (T_2), i.e.,

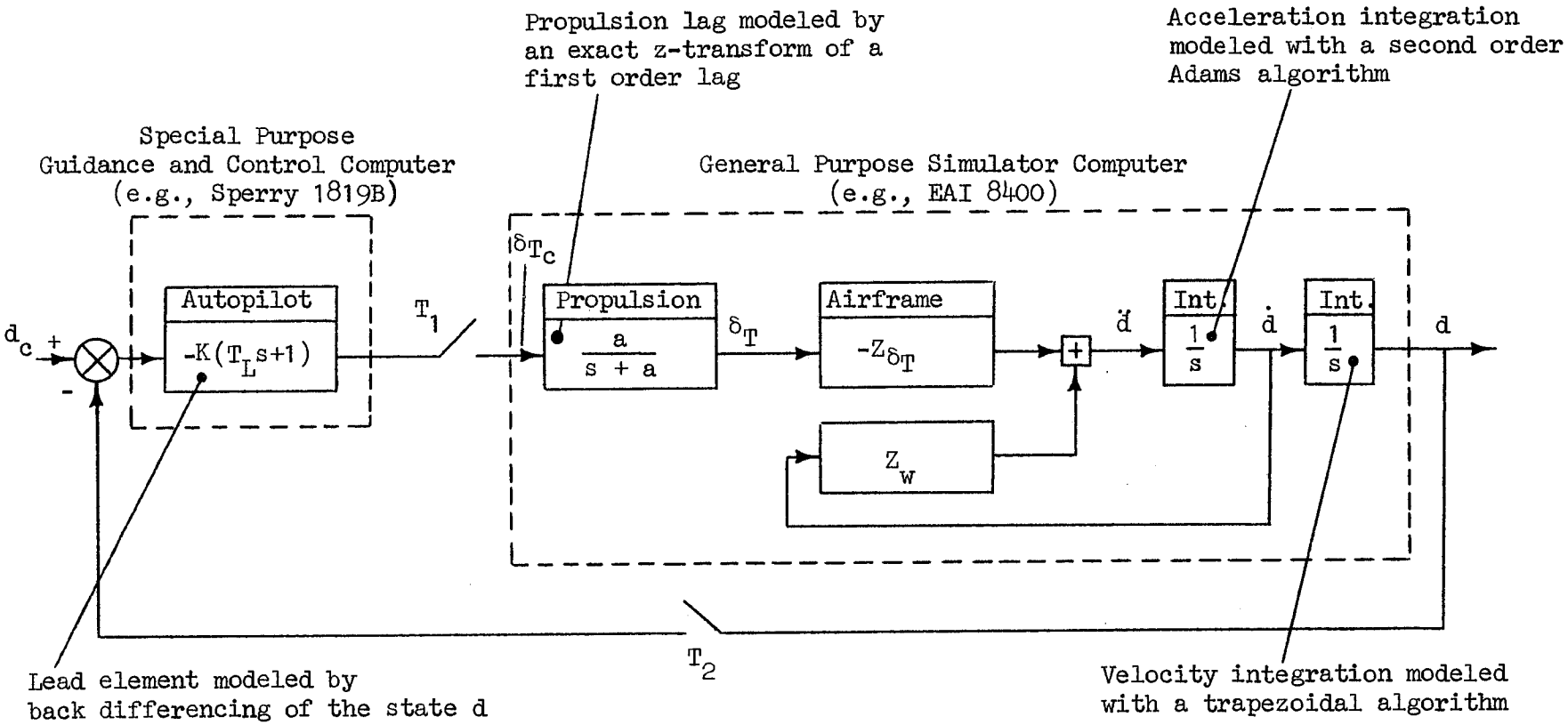
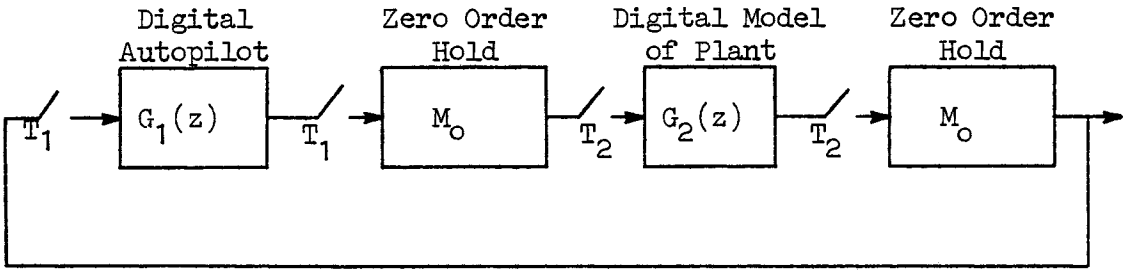
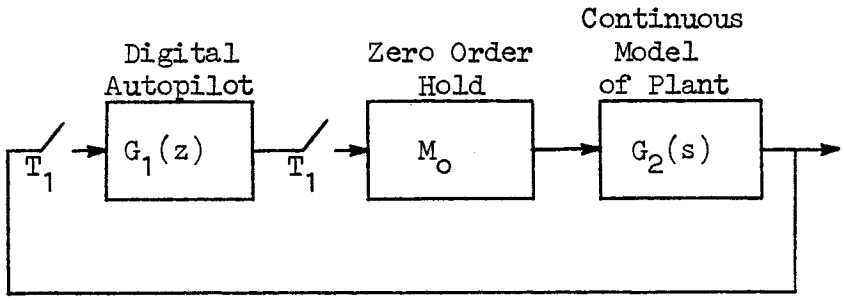


Figure 6. Diagram of an Illustrative Simulator Model Example — Automatic Glide Slope Control via Throttle

General Simulator Computer Example



Actual Airborne System Being Simulated
(A standard for simulator validity)



The frame-time-related effects include:

$T_1 \doteq T_2$ (frame slip)

$T_1 \neq T_2$ (large difference in frame times)

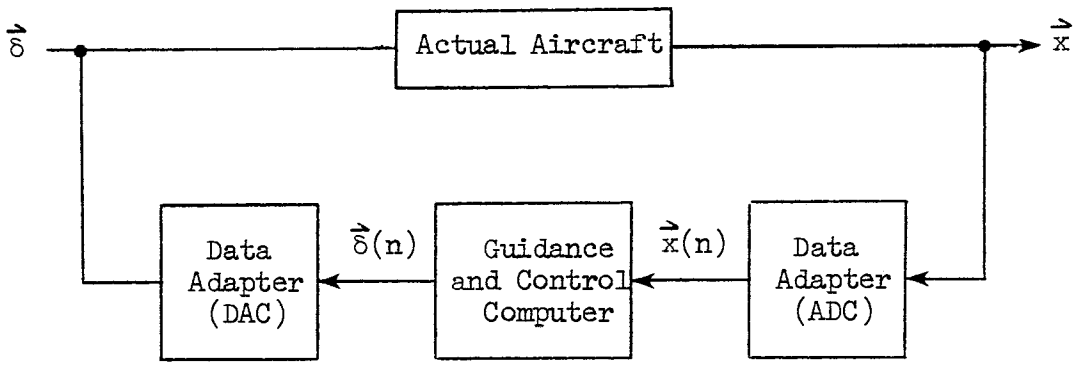
Finally, the use of multiple T_2 within the airframe/propulsion block will be studied (see Ref. 7).

At this point we have described the model-related aspects of our analysis example. Let us now go on to the next section which addresses the specific hardware and software related features as characterized by an existing simulator computer configuration.

C. FUNCTIONAL DESCRIPTION OF AN EXTANT SIMULATOR COMPUTING SYSTEM

The hardware and software configuration which will be used to define a realistic context for our study of spurious digital effects consists of the Sperry 1819B flight control computer and the EAI 8400 general purpose digital computer. This combination forms the Ames Research Center V/STOLAND simulator — A system widely and routinely used by the Ames simulator community. Most important, it is a prime example of a two-computer/two-rate simulator system. The purpose of this subsection is to define the features of the V/STOLAND simulator needed for system analysis.

Let us begin with the Sperry 1819B portion of the system. Figure 7 is a function block diagram of the V/STOLAND system used to control an actual aircraft. The system is essentially a digitally controlled continuous process. Input to and output from the digital computer (i.e., ADC and DAC) is provided by the Data Adapter (Ref. 9). The Sperry 1819B digital computer (Ref. 10) is used to implement the required guidance and control equations. Features of the Data Adapter and 1819B digital computer will be discussed shortly.



$\vec{x}$ = vector of aircraft states, navigation aids, and control commands
$\vec{\delta}$ = vector of aircraft controls

Figure 7. Architecture of V/STOLAND Controlling the Actual Aircraft

The guidance and control equations are usually developed in the continuous domain (i.e., the s-plane) by using a continuous model of the aircraft. The guidance and control equations are then transformed into the discrete domain (e.g., by Tustin transform, z-transform, etc.) and finally implemented in the 1819B.

The role of the EAI 8400 digital computer is to simulate the actual aircraft and its flight dynamics. In addition, a mockup of the aircraft cab is provided, and sometimes actual hardware is used (e.g., control servos). The aircraft simulation computer is interfaced with the guidance and control computer and both computers have access to the aircraft cab (i.e., controls, servos, displays, and switches). The basic architecture of this system is shown in Fig. 8. There are many additional components of the system shown in Fig. 8 that have been omitted because they are not relevant to the subjects addressed herein. References 9 through 12 should be consulted for additional details.

In general the simulation and guidance and control computers operate at different frame times, and they are not synchronized. Figure 9 contains time lines for both computers. The numbers on the time lines indicate the functions performed within a single frame time. How and when each computer communicates with the various simulation components is also shown in Fig. 9. Tables 3 and 4 contain expanded definitions of the functions performed by the two computers. Table 5 contains some of the key parameters of both computers.

Both the EAI 8400 and the 1819B are capable of multirate operation, although they are shown as single-rate computers in Fig. 9. The 1819B has three loops and their frame times are fixed at 25 ms, 50 ms, and 100 ms. Both input and output in the 1819B occurs every 25 ms. The EAI 8400 is capable of two loops. The frame time of the fast loop is set as low as possible but its minimum is usually above 50 ms (52 ms is representative of a simple model like the UH-1H, but 70 to 75 ms is required for more complex models like the Augmentor Wing Jet STOL Research Aircraft). Input/output (I/O) in the EAI 8400 is always programmed in the fast loop.

Note that both input and output are performed at the beginning of the frame time in the simulation computer. Because the I/O is so fast compared

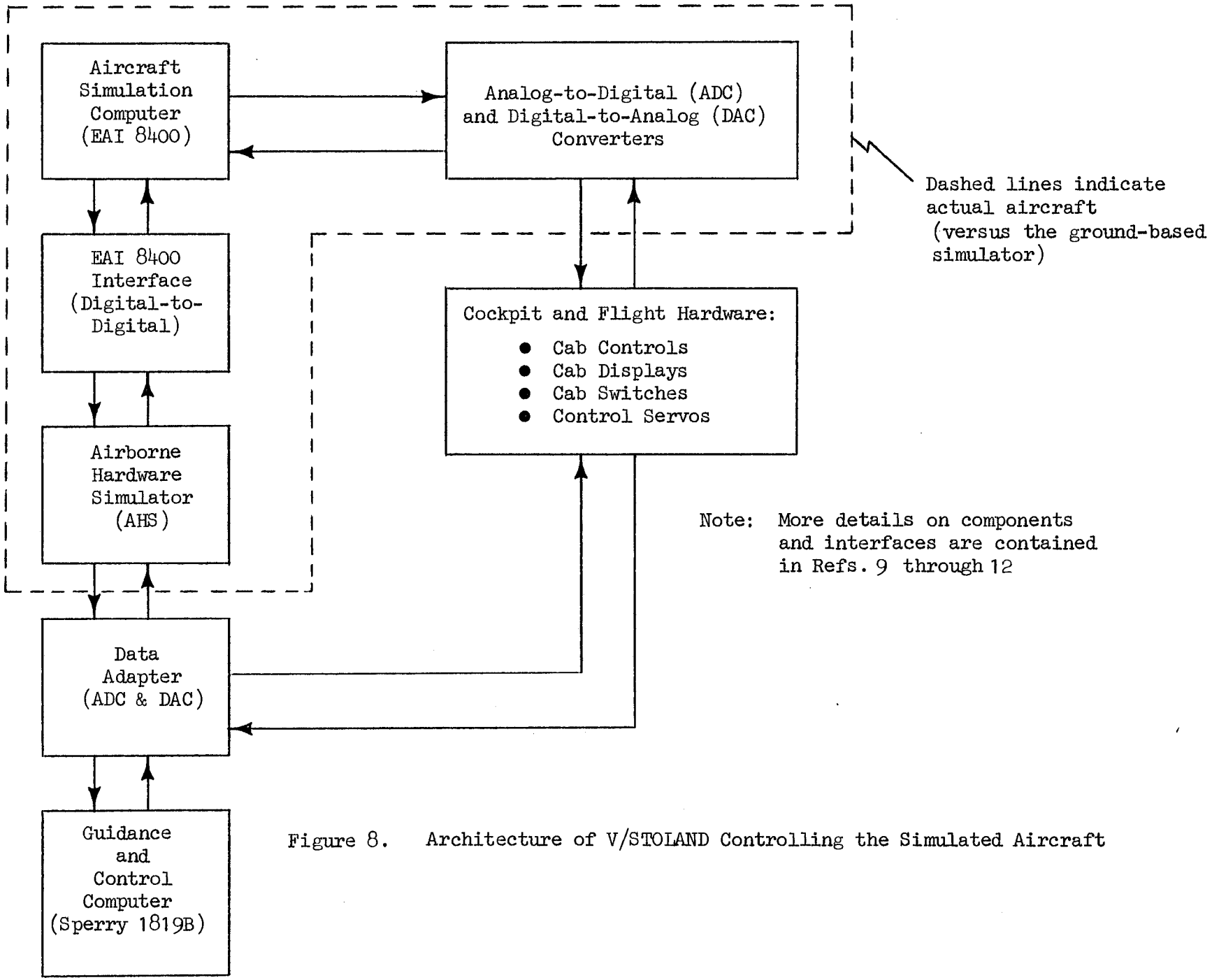


Figure 8. Architecture of V/STOLAND Controlling the Simulated Aircraft

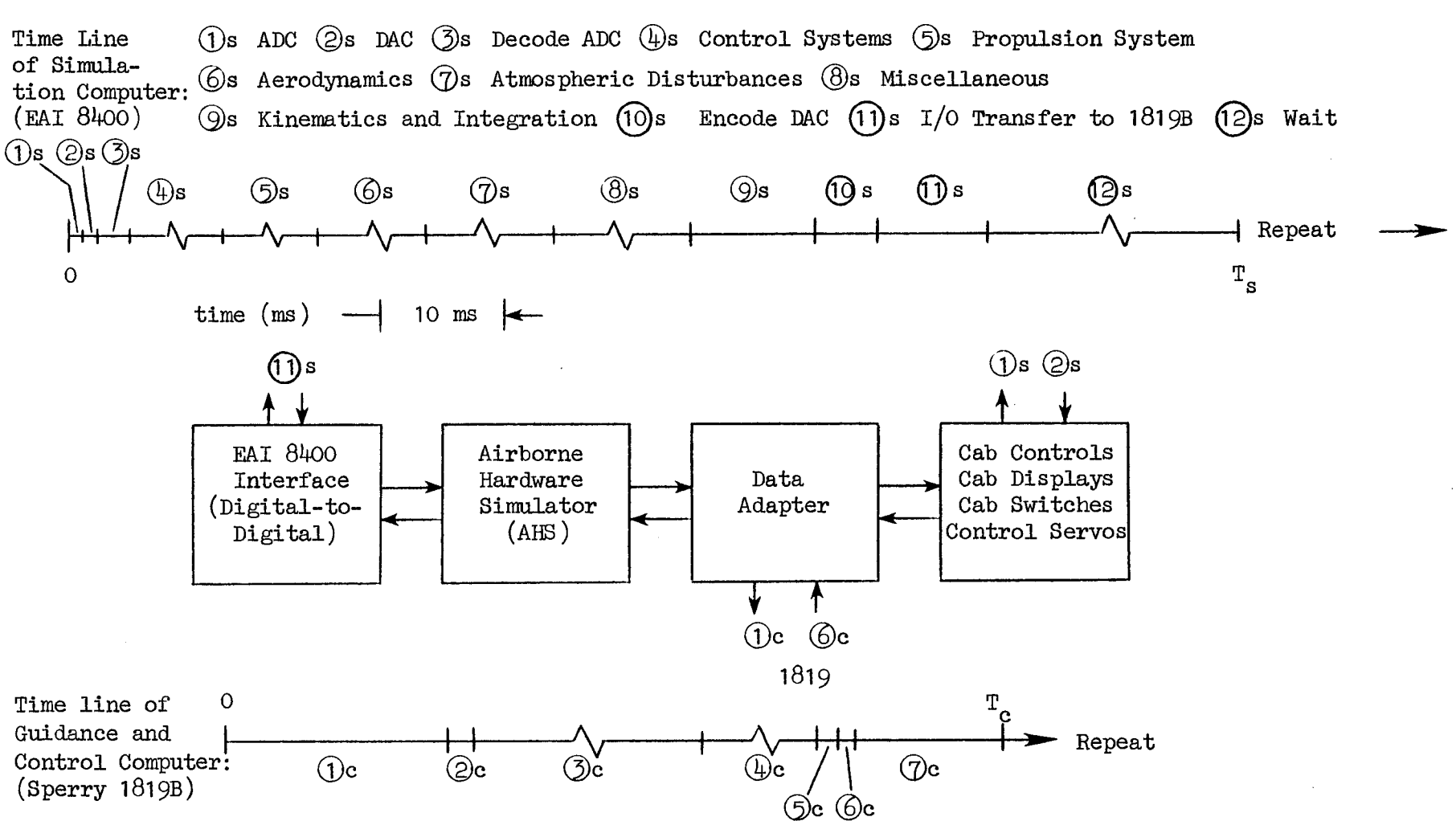


Figure 9. Time Lines of Simulation and Guidance and Control Computers

TABLE 3

ORDER OF EXECUTION AND TIME REQUIREMENTS OF REALTIME FUNCTIONS PERFORMED IN THE AIRCRAFT SIMULATION COMPUTER (EAI 8400)

Number	Functional Description	Time Required
1	Analog-to-digital conversion (ADC) of 32 input variables (5 μs per variable)	0.16 ms
2	Digital-to-analog conversion (DAC) of 88 output variables (5 μs per variable)	0.44 ms
3	Decode analog input (SADC)	2.0 ms
4	Control system simulation (CONTR2)	— *
5	Propulsion system simulation (ENGINE)	— *
6	Aerodynamics (AERO2)	— *
7	Atmospheric disturbances (WINDC)	— *
8	Any special-purpose processing (SHOW, UTIL2, etc.)	— *
9	Kinematics and integration (SMART)	7.8 ms
10	Encode analog output (SDAC)	4.0 ms
11	Encode digital-to-digital output, initiate and wait for I/O transfer to 8400 interface, wait for I/O completion, and decode digital-to-digital input	7.0 ms
12	Wait for end of time frame	—
		$\sum = T_s$ (typically 75 ms)

* Time required is a function of model complexity.

TABLE 4

ORDER OF EXECUTION AND TIME REQUIREMENTS OF REALTIME FUNCTIONS PERFORMED IN THE GUIDANCE AND CONTROL COMPUTER (SPERRY 1819B)

Number	Functional Description	Time Required
1	Input 64 analog variables and 24 digital variables (224 μs per analog variable)	14.5 ms
2	Decode inputs	1.4 ms
3	Miscellaneous housekeeping functions	—
4	Guidance and control calculations	—
5	Encode outputs	1.1 ms
6	Output 24 analog variables and 15 digital variables (125 μs for each set of three analog variables)	1.0 ms
7	Wait for end of time frame	—
		$\sum = T_c$ (50 ms)

TABLE 5

KEY PARAMETERS OF THE SIMULATION
AND GUIDANCE AND CONTROL COMPUTER

Parameter	Simulation (EAI 8400)	Guidance and Control (Sperry 1819B)
Word Size	32 bits (floating point)	18 bits (fixed point)
Execution Times*		
Load	4.2 μs	2.4 μs
Store	4.2	2.4
Add	5.3†	2.4
Subtract	5.3†	2.4
Multiply	8.1†	7.4
Divide	11.4†	8.0
Compare	4.2	1.6

* All times are in microseconds (μs - 10^{-6} sec)

† Includes clear and load

to the total frame time (5 μs per variable for ADC and DAC versus 50 to 75 ms total frame time) there is little I/O data skewness. After decoding the input (Step 3) the control systems, propulsion system, aerodynamics, and atmospheric disturbance models are executed (Steps 4 through 7). The times required to execute these models are, of course, functions of their complexity. Any special purpose processing such as performance calculations or strip chart outputs is done in Step 8. The aircraft equations of motion are integrated in Step 9 in the subroutine SMART. This subroutine is used by all aircraft simulations and takes 7.8 ms to execute. Encoding the outputs (DAC, Step 10) takes about 4 ms. I/O transfer between the simulation and the guidance and control computers (Step 11) is a multiphased process. The transfer from the EAI 8400 to the Sperry 1819B takes place as follows:

a. The digital output data is encoded

b. The digital output data is sent to the EAI 8400 interface ("D-to-D box")

c. The D-to-D box sends the data to the Airborne Hardware Simulator (AHS)

d. The AHS converts the digital data to analog signals in order to simulate actual sensors in the aircraft (e.g., Navaids, gyros, etc.)

e. The Data Adapter converts the AHS analog signals to digital data (ADC) and transfers the data to the 1819B. The Data Adapter ADC's are multiplexed and it takes approximately 224 μs per variable to perform the conversion and transfer the data to the memory of the 1819B.

The transfer from the 1819B to the EAI 8400 is accomplished by Steps a through e performed in reverse order. Two important differences are the speed and method of data conversion performed by the Data Adapter transferring data from the 1819B to the AHS (DAC). It takes only 125 μs to convert and transfer three variables to the AHS. That is, three variables are converted in parallel. Thus there is virtually no skewness in the output data. The simulation computer must wait for the entire I/O process to take place before proceeding, which takes approximately 7 ms.

There is an interesting difference between the data input transfers to the 1819B in the simulation versus the actual aircraft. In the simulation, data from the EAI 8400 is sampled at a very high rate by the D-to-D box and sent to the AHS (it is stored there and waits for sampling and transfer to the 1819B by the Data Adapter). Thus there is virtually no input skewness in the data sent to the 1819B. In the actual aircraft, however, it takes 224 μs to sample each input variable. Thus input data skewness could be significant, especially if the variables used in a particular algorithm were widely separated in the input queue.

This completes the description of system features which will be used to form a realistic context for analysis. We shall conclude now with a brief statement of how the analysis will be presented.

D. FORMULATION OF ANALYSIS CASES

The key to analyzing systems composed of digital processors (and continuous elements) lies in correctly formulating the problem. The following paragraphs describe a simple, systematic method which is sufficiently general to accept multi-rate or multi-loop digital systems composed of more than one processor. After studying the cases to be presented in Section III, the reader should have a clear understanding of the formulation procedure. Spectral or time-domain analysis can then be performed using the methods given in Refs. 1 or 2.

A four-step outline of the formulation procedure is given in Table 6 and elaborated below.

1. Obtain Essential System Description

The first and most important step is the acquisition of essential system information. Note that for any digital element two pieces of information are required — program instructions and digital processor timing.

The program instructions can be in a variety of forms. For simplicity we shall assume that the instructions are in FORTRAN since it is the language with which the reader is most likely to be familiar. Also

TABLE 6

FORMULATION OF DIGITAL PROCESSOR ANALYSIS

1. Obtain Essential System Description
   - Program instructions
   - Digital processor
   - Differential equations of continuous elements
2. Derive Transform Domain Properties
   - Convert program instructions to difference equations
   - Convert difference equations to z-domain
   - Convert differential equations to s-domain
3. Construct System Block Diagram
   - Sample continuous signals (ADC)
   - Sample and zero-order hold discrete signals (DAC)
4. Manipulate Block Diagram
   - Formulate as open-loop system
   - Lump elements into F(s), G(z), hold

we shall assume that the program is segmented into subroutines which are called sequentially, a common practice with the Ames Research Center simulator facilities. Figure 10 shows a sample set of program instructions in this form which also will be used in subsequent cases.

```
SUBROUTINE CONTROL
E=DC-D
EDOTHT=(E-EP)/DT
EP=E
DELTC=RK*(TL*EDOTHT+E)
RETURN
END
```

(DC, D, and EP are the inputs; DELTC is the output.)

Figure 10. Example of FORTRAN Program Instructions

The digital processor timing information which is required includes computer frame time (cycle time for real-time operation), points in time within the frame when data are input and output. When possible it is convenient to realize the actual timing diagram by assuming that (1) all data input occurs at the beginning of the frame (this is the convention at Ames Research Center) and (2) all data output occurs just after the data input, i.e., there is effectively one frame time interval between input and output generated using that input. Figure 11 shows an actual timing diagram (for the Sperry 1819 discussed previously) and an idealized timing diagram as we have defined it.

If the system in question involves continuous elements then the linearized ordinary differential equations are required as part of the essential system description.

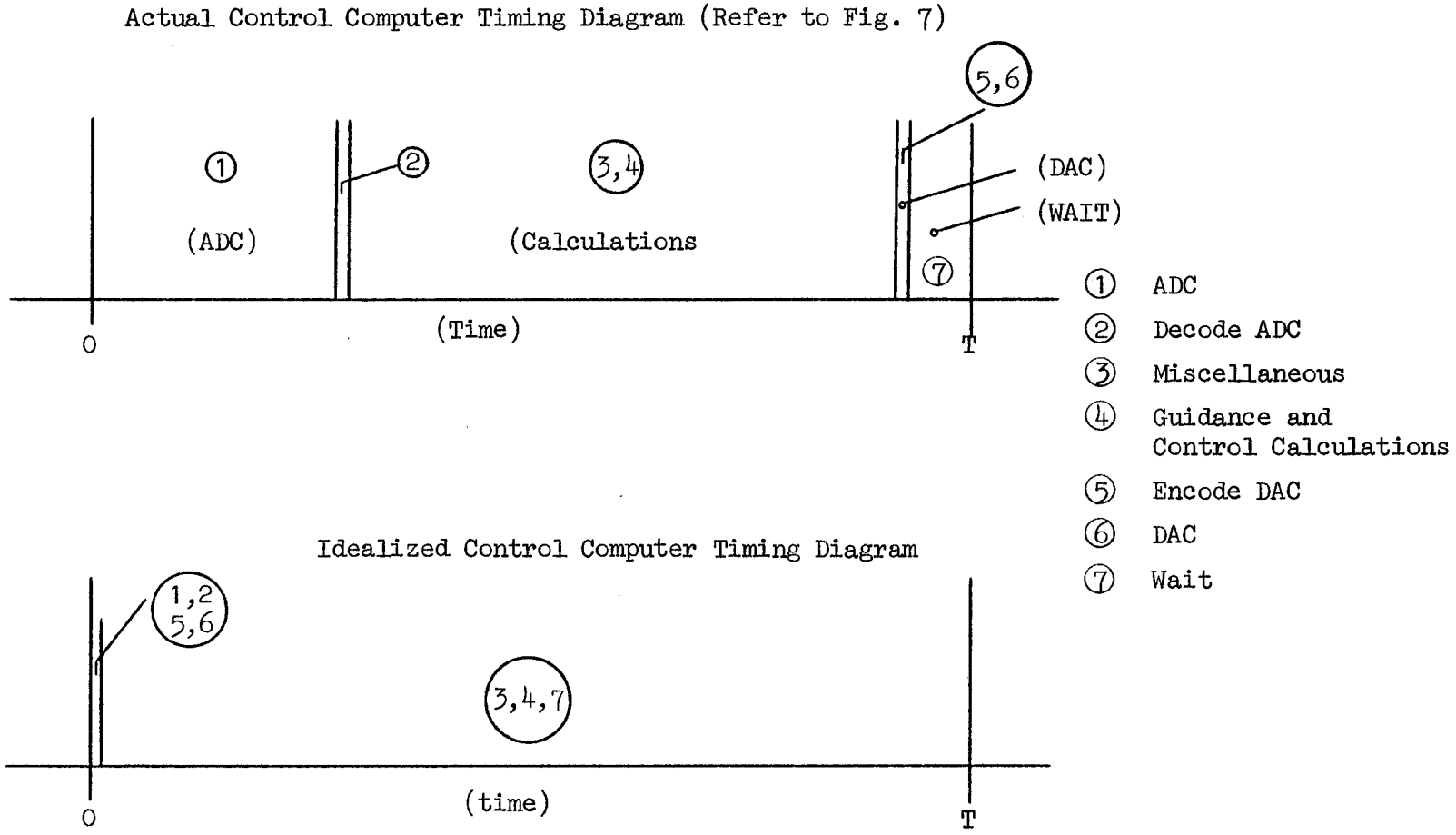


Figure 11. Example of Actual Versus Idealized Timing Diagrams

2. Derive Transform Domain Properties

This step involves converting discrete system components to linear finite difference equations and then to the z-domain. Also, for any continuous elements, the linear differential equations must be converted to the s-domain. (We shall cover only the former of these steps.)

The transformation of discrete elements requires careful attention in order to get the correct effects of digital processor timing as well as the correct implementation of program instructions and algorithms. The following procedure is suggested as a simple, mechanical routing which should give consistent and correct results if followed to the letter. The reader is cautioned that the main pitfalls lie in correct handling of the multi-processor situation — the single processor is relatively uncomplicated.

The first issue to address is indexing of difference equation terms. The rules offered here are:

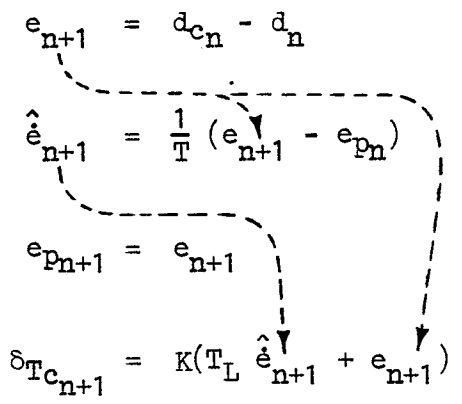
(a) Consider a given frame interval in which one or more subroutines will be executed by a given single digital processor.

(b) Label all input to the frame interval, either data from outside sources or variables computed during the previous frame, as "n". (Avoid thinking in terms of "past," "present," "future," etc.)

(c) Label all computations contained in the sequence of program instructions during the given frame as "n+1."

(d) Apply the same rules to each additional processor. Do not be concerned about inter-processor synchronization. The task at hand is to develop transfer relationships within each separate processor.

For the example set of instructions given earlier in Fig. 8 the corresponding difference equations are:



(d_c is an input at the start of the frame and d comes from the earlier frame, therefore the index n)

(e was calculated in this frame but e_p came from the earlier frame)

(The past value storage is handled in the same manner as other calculations)

Thus we have translated the FORTRAN instructions line-by-line strictly in accord with our rules.

The next step is to write the z-transform equations using the index labels to determine the exponent of z, i.e.,

$$ze = d_c - d$$

$$z\hat{\dot{e}} = \frac{1}{T}(ze - e_p)$$

$$ze_p = ze$$

$$z\delta T_c = K(T_L z\hat{\dot{e}} + ze)$$

A transfer function can be formed now by direct manipulation of these z-domain equations, i.e.,

$$\frac{e}{d_c - d} = \frac{1}{z}$$

$$\frac{\hat{\dot{e}}}{e} = \frac{1}{T}\frac{z - 1}{z}$$

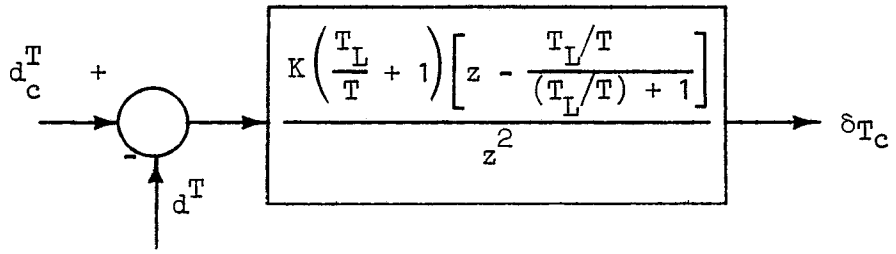
$$\frac{\delta T_c}{e} = \frac{K\left[T_L\, z\left(\frac{1}{T}\,\frac{z-1}{z}\right) + z\right]}{z}$$

$$= \frac{K\left(\frac{T_L}{T} + 1\right)\left[z - \frac{T_L/T}{(T_L/T) + 1}\right]}{z}$$

$$\text{or} \qquad \delta T_c = \frac{K\left(\frac{T_L}{T} + 1\right)\left[z - \frac{T_L/T}{(T_L/T) + 1}\right]}{z^2}\,(d_c - d)$$

3. Construct System Block Diagram

First, let us directly form the portion of the block diagram given by the transform domain description developed above:



(Note that the $d_c(z)$ and $d(z)$ in turn imply sampled d_c and d or, in our notation, d_c^T and d^T.)

This z-domain block diagram interpretation of the program instructions prepares us for constructing the overall digital processor architecture. In effect we must add an I/O description consisting of:

(a) A sampler on the input running at the processor frame time — representing ADC or DDC.

and (b) A sample and zero-order-hold on the output — the output buffer register and DAC or DDC.

The overall digital processor block diagram is shown in Fig. 12.

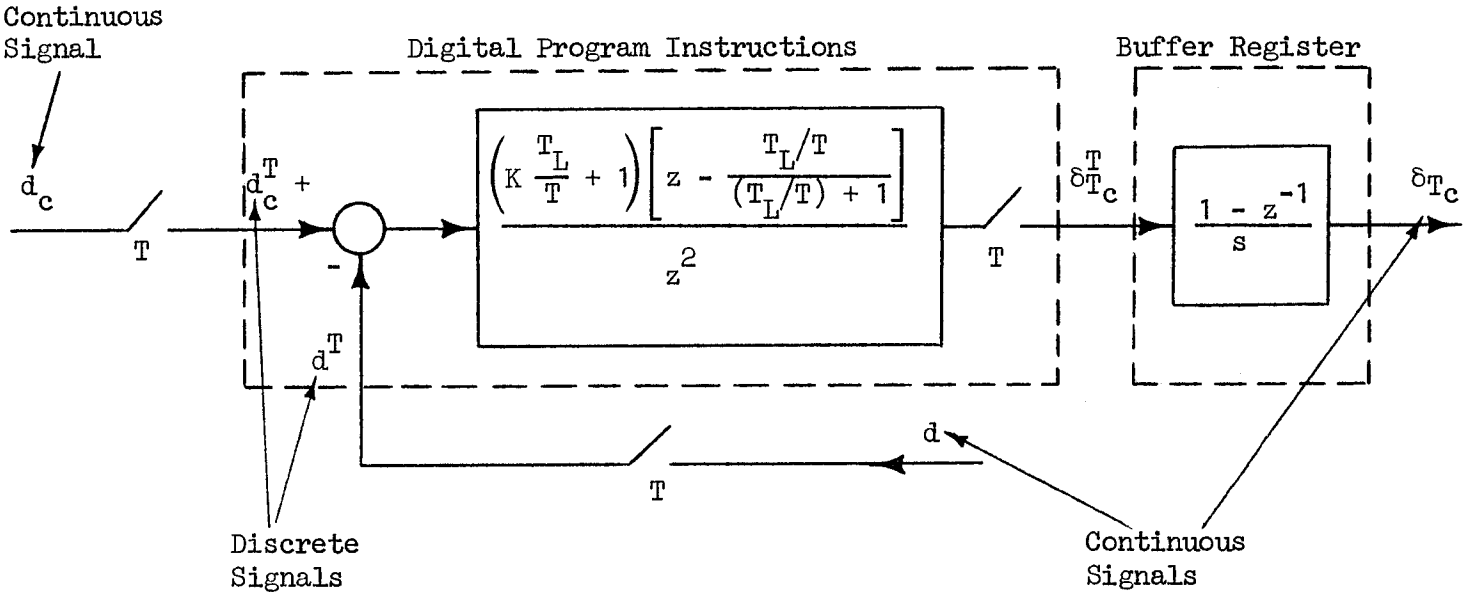


Figure 12. Digital Processor Block Diagram

4. Manipulate Block Diagram

The last step is to manipulate the block diagram into an open loop form so that input-output transfer relationships can be examined. In the simple example considered above this step is essentially done since we are not completing the feedback loop between δ_{T_c} and d. This will be handled, however, in the subsequent examples given in Section III.

SECTION III

PRESENTATION OF ANALYSIS RESULTS

In terms of the central example previously defined, the digital effects are viewed in both the time and frequency domains. This permits a high degree of insight and allows application of a number of candidate system metrics. The cases considered are listed in Table 7 and cover a number of possible simulator mechanizations including the actual system simulated. Except for the multirate analyses both in Case 4 and in Volume Two, generalized results have been computed and plotted using representative numerical values. This involved some software development of PDP-10 and HP-97 programs. These new programs were were based on the method outlined in the appendix.

The main objective of the set of examples presented is to demonstrate the analysis procedures discussed previously and to illustrate how various simulator formulations can affect the resulting system dynamics.

TABLE 7

EXAMPLES ANALYZED

Case 1	The system to be simulated — a digitally controlled continuous plant
Case 2	Simulation of controller and plant using a single digital computer
Case 3	Simulation of plant but controlled by a second digital computer — frametimes common and synchronous
Case 4	Same as Case 3 but with dual frametimes

Case 1: The System to be Simulated — A Digitally Controlled Continuous Plant

This system provides a point of reference for the various simulator implementations to follow. It consists of a digital controller which provides propulsion commands based on flight path position measurements and a continuous plant composed of simple propulsion and airframe dynamics. The essential elements are described in Table 8.

A block diagram of the above system is shown in Fig. 13. This characterization, in turn, allows us to systematically construct an input-output expression appropriate for analyzing overall system response. In this case we choose to consider the closed-loop response of the actual flight path, d, relative to the command, d_c, thus:

$$d = (GM_o)\left[1 + G_1^T (GM_o)^T\right]^{-1} G_1 d_c^T \qquad (15)$$

Since $G_1^T = G_1(z)$ it is only necessary to convert the control computer program instructions to a z-domain representation. The term $(GM_o)^T$ is a discrete representation of the continuous plant (i.e., engine and flight path dynamics). It can be evaluated by using partial fraction expansions and the z-transform, as described and demonstrated in Ref. 1. For higher order systems, however, we recommend using the technique described in the appendix of this report in conjunction with an automatic data processor.

In Eq. 15 note that d is a <u>continuous</u> variable but that d_c^T is a sampled, or <u>discrete</u>, variable. In order to evaluate the fidelity of the system shown in Fig. 13 we want to obtain the continuous frequency response of the output, d, to the input, d_c. This can be done using the techniques described in Ref. 1. The result is

$$\frac{d}{d_c}(jb) = \sum_{i=0}^{\infty} (A_i + jB_i) \qquad (16)$$

$$A_i + jB_i = \left[\frac{G(s)M_o(s)}{T}\right]_{s=j\omega_i} \left\{\left[1 + G_1(z)(GM_o)^T\right]^{-1} G_1(z)\right\}_{z=e^{jbT}} \qquad (17)$$

where $\omega_i = b + (2\pi/T)\, i$

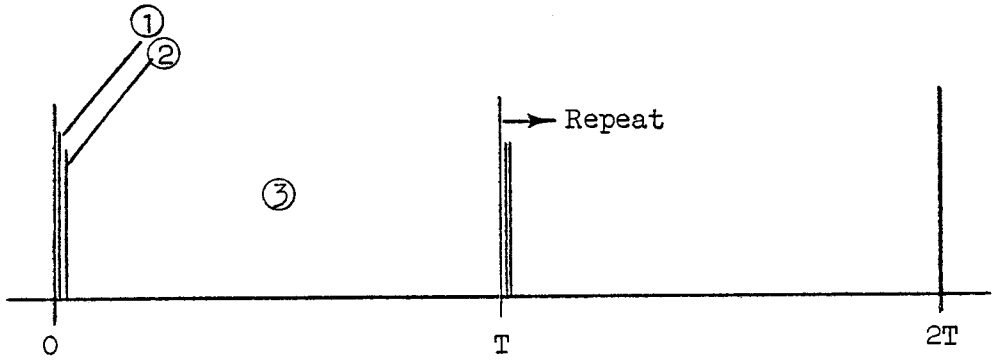
TABLE 8

ESSENTIAL SYSTEM DESCRIPTION ELEMENTS FOR CASE 1

1. Control Computer FORTRAN Program:

```
SUBROUTINE CONTROL
E=DC-D
EDOTHT=(E-EP)/DT
EP=E
DELTC=RK*(TL*EDOTHT+E)
RETURN
END
```

2. Control Computer Timing Diagram:



① d_{c_n}, d_n input to flight control system computer

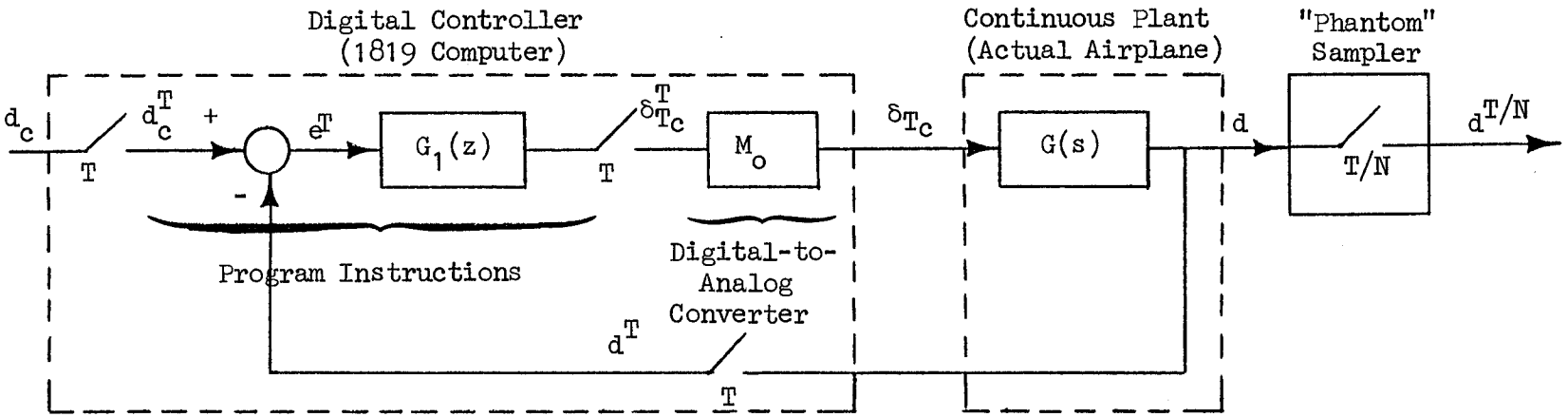
② $\delta^T_{T_{c_n}}$ output to a DAC (which is modeled by a zero order data hold); the output of the DAC is input to the engine

③ "Control" executed to compute "n+1" values

3. Airframe and Engine Equations of Motion (from Fig. 6)

$$\dot{\delta}_T = -a\,\delta_T + a\,\delta_{T_c}$$

$$\ddot{d} = +Z_w\,\dot{d} - Z_{\delta_T}\,\delta_T$$



$$\delta_{T_c}^T = G_1^T d_c^T - G_1^T (GM_o)^T \delta_{T_c}^T$$

$$d = GM_o \left[1 + G_1^T (GM_o)^T\right]^{-1} G_1 d_c^T$$

Figure 13. Case 1 Functional Block Diagram

The frequency response obtained from evaluating Eqs. 16 and 17 would represent the continuous output resulting from exciting the system shown in Fig. 13 with an input $d_c = \sin(bt)$. Some special-purpose software was written to evaluate and plot Eq. 17.

The continuous, steady-state time response, d, resulting from a sinusoidal input of $d_c = \sin(bt)$ can be calculated using the phantom sampler concept discussed in Ref. 1. The appropriate equations are:

$$d^{T/N}(t) = \sum_{n=0}^{N-1} A_n \sin \omega_n t + B_n \cos \omega_n t \tag{18}$$

$$A_n + jB_n = \left[\frac{1}{N}(GM_o)^{T/N}\right]_{z=e^{j\omega_n T/N}} \left\{\left[1 + G_1(z)(GM_o)^T\right]^{-1} G_1(z)\right\}_{z=e^{jbT}} \tag{19}$$

where $\omega_n = b + (2\pi/T)\, n$

and N = an integer multiple of the basic frame time T shown in Fig. 13.

The phantom sample variable $d^{T/N}(t)$ will <u>exactly</u> match the continuous variable d at the sampling instances T/N. Thus even for large values of T an exact plot of d can be obtained by choosing a large enough value for N.

<u>Evaluation of System Elements</u>

First, let us describe the digital control computer in the z-domain. Applying the established rules, we label each computational step as "n+1," i.e.,

$$e^T_{n+1} = d^T_{c_n} - d^T_n$$

$$\dot{e}^T_{n+1} = \frac{1}{T}\left(e^T_{n+1} - e^T_{p_n}\right)$$

$$e^T_{p_{n+1}} = e^T_{n+1}$$

$$\delta^T_{T_{c_{n+1}}} = K(T_L \hat{e}^T_{n+1} + e^T_{n+1})$$

Transforming to the z-domain ($n+1 \longrightarrow z$, $n \longrightarrow z^0$, $n-1 \longrightarrow z^{-1}$, etc.) (the superscript T is dropped and the variables are understood to be in the z-domain, i.e., $e(z) = e$, etc.):

$$ze = d_c - d$$

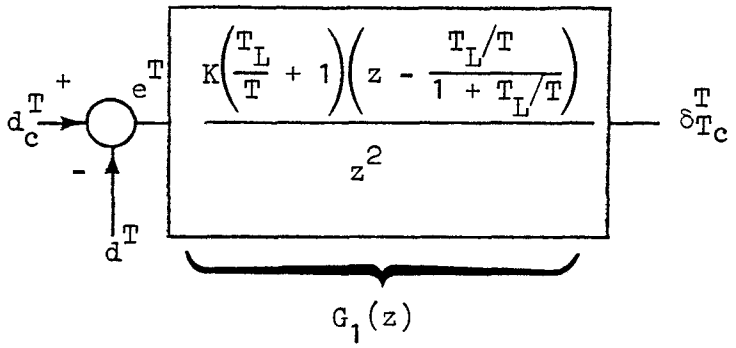
$$z\hat{e} = \frac{1}{T}(z-1)\, e$$

$$z\delta_{T_c} = K\left[T_L \frac{1}{T}(z-1) + z\right]e$$

$$= K\left(\frac{T_L}{T} + 1\right)\left(z - \frac{T_L/T}{1 + T_L/T}\right)e$$

$$= K\left(\frac{T_L}{T} + 1\right)\left(z - \frac{T_L/T}{1 + T_L/T}\right)z^{-1}\,(d_c - d)$$

This, in turn, leads to the following representation:

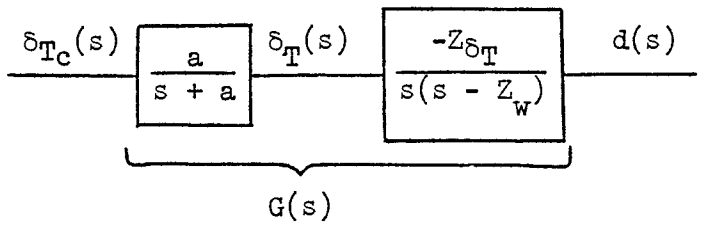


The continuous plant can be transformed to the s-domain in an analogous manner starting with the differential equations, i.e.,

$$s\,\delta_T(s) = -a\,\delta_T(s) + \delta_{T_c}(s)$$

$$s^2 d(s) = Z_w d(s) - Z_{\delta_T}\,\delta_T(s)$$

or



Finally, the zero order hold, M_o, is:

$$M_o = \frac{1 - z^{-1}}{s}$$

The term $(GM_o)^T$ can be evaluated using either the technique presented in the appendix or a partial fraction expansion technique such as described in Refs. 1 and 2 and shown below.

$$(GM_o)^T = \left[\frac{(1-z^{-1})}{s} \frac{a}{(s+a)} \frac{(-Z_{\delta_T})}{s(s-Z_w)}\right]^T$$

$$= aZ_{\delta_T}\left[\frac{a+Z_w}{a^2Z_w^2} + \frac{T/aZ_w}{(z-1)} + \frac{(z-1)/[a^2(a+Z_w)]}{(z-e^{-aT})} - \frac{(z-1)/[Z_w^2(a+Z_w)]}{z-e^{Z_wT}}\right]$$

For the set of numerical values stated earlier, i.e.,

$$K = 0.3^2\sqrt{2} = 0.127$$

$$a = 1.0$$

$$Z_{\delta_T} = -1.0$$

$$Z_w = -0.3$$

$$T = 0.05$$

$$(GM_o)^T = \frac{0.26498\times10^{-4}(z+0.26363)(z+3.6720)}{(z-0.95123)(z-0.98511)(z-1)}$$

Combining this result with G_1,

$$\left[1 + G_1^T(GM_o)^T\right]^{-1} = \frac{z^2(z-0.95123)(z-0.98511)(z-1)}{(z-0.95143)[0.022316;0.0073396]_z[-0.99987;0.99275]_z^\dagger}$$

Using these results, and Eqs. 16 to 19, it is possible to analyze d/d_c in both the frequency and time domains. The resulting frequency response is shown in Fig. 14. Note that the aliasing involved in this frequency response is substantially attenuated. Physically, the continuous airplane dynamics are filtering the effects of the digital flight control system by at least two integrations — three, counting the engine dynamics. This will not be the case when those integrations are performed digitally as we shall see.

The phase angle shows an ever increasing roll-off — characteristic of a transport delay. This is a direct result of the digital flight control computer situated between d_c and d.

The high attenuation of digital effects is also apparent in Fig. 15 in the steady-state time histories resulting from a sinusoidal command, d_c.

Case 2: Simulation of Controller and Plant Using a Single Digital Computer

This case represents a simple simulator mechanization using a single general purpose digital computer, e.g., an EAI 8400. The digital control functions, ordinarily in a separate computer, are contained in the simulator computer along with the propulsion and airframe models.

Table 9 summarizes the essential information required, namely a listing of controller and plant subroutines and a timing diagram. Figure 16 shows the block diagram formed from this information and which permits us

---

† This is a shorthand notation for a pair of complex roots, i.e., in the z-plane

$$[a;b]_z \triangleq [z^2 + 2\,ab\,z + b^2]$$

or in the s-plane

$$[\zeta;\omega]_s \triangleq [s^2 + 2\,\zeta\omega\,s + \omega^2]$$

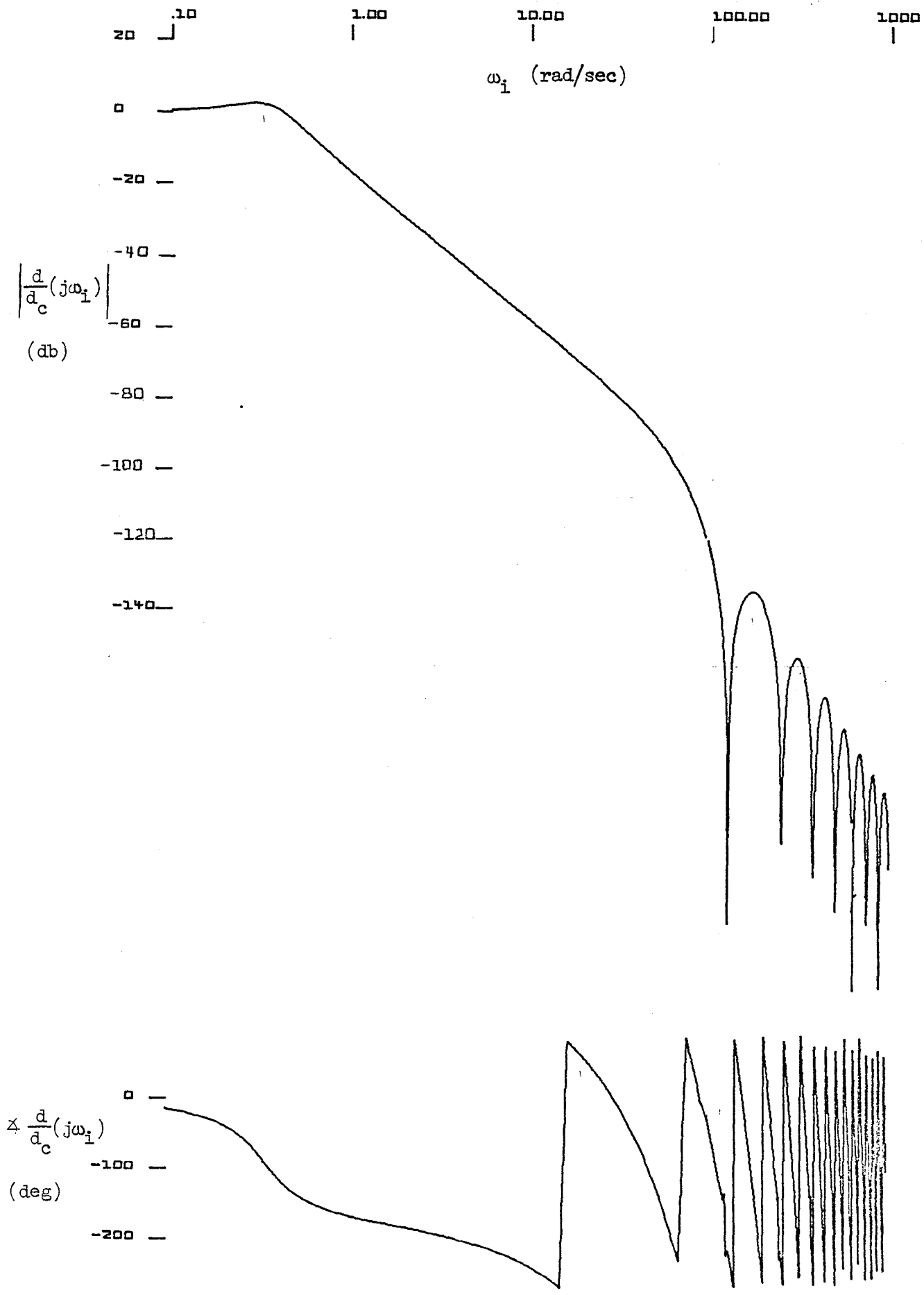
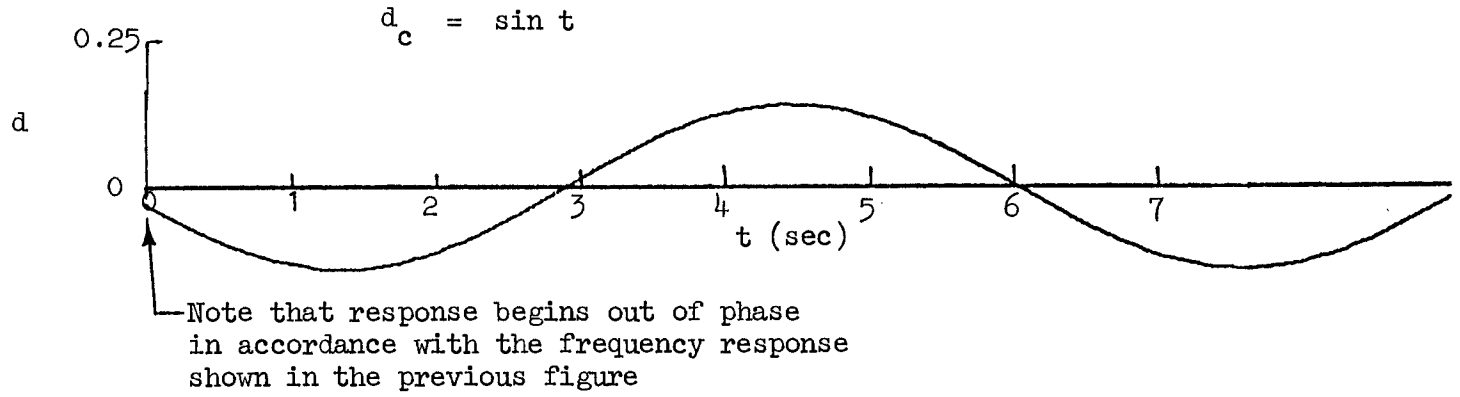


Figure 14. Case 1 Frequency Response of d/d_c



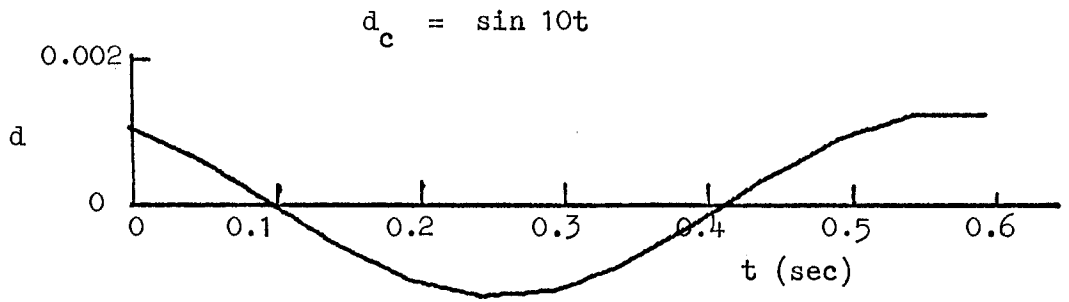


Figure 15. Case 1 Response of d to Sinusoidal Inputs of d_c

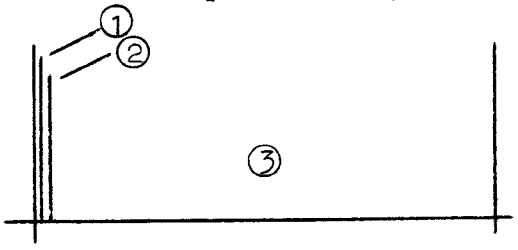
TABLE 9

ESSENTIAL SYSTEM DESCRIPTION ELEMENTS FOR CASE 2

1. Simulator Computer FORTRAN Program

```
SUBROUTINE CONTROL
E=DC-D
EDOTHT=(E-EP)/DT
EP=E
DELTC=RK*(TL*EDOTHT+E)
RETURN
END

SUBROUTINE ENGINE
DELT=EXP(-A*DT)*DELT+(1-EXP(-A*DT))*DELTC
RETURN
END

SUBROUTINE AERO
DDDOT= ZW*DDOT-ZDT*DELT
RETURN
END

SUBROUTINE SMART
TIME=TIME+DT
DDOT=DDOT+DT/2*(3*DDDOT-DDDOTP)
DDDOTP=DDDOT
D=D+DT/2*(DDOT+DDOTP)
DDOTP=DDOT
RETURN
END
```

2. Simulator Computer Timing Diagram



① d_{c_n} input to digital computer

② d_n output to visual and motion devices

③ "Control," "Engine," "Aero," and "Smart" executed

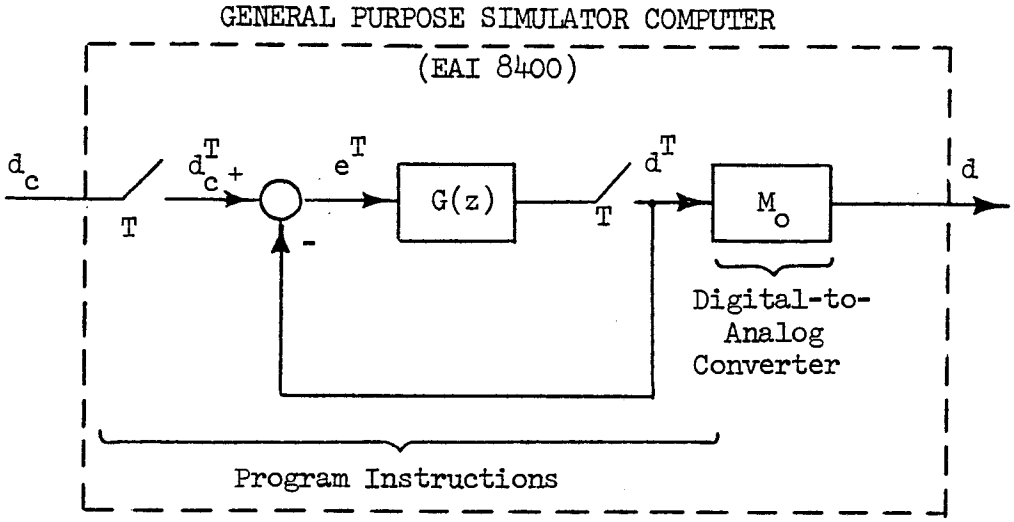


Figure 16. Case 2 Functional Block Diagram

to express the output, d, in terms of the sampled command, d_c^T, i.e.,

$$d = M_o[1 + G(z)]^{-1} G(z) d_c^T$$

Evaluation of System Elements

The procedure for evaluating the overall system transfer relationships will again follow the rules set forth in Section II.D and exercised in the first example. The main difference will be the more extensive set of digital computer instructions to be handled.

First, the FORTRAN program is translated into the difference equations shown in Table 10. Next the z-domain transfer functions are obtained as shown in Table 11. Hence,

TABLE 10

CASE 2 DIFFERENCE EQUATIONS

$$e_{n+1} = d_{c_n} - d_n$$

$$\hat{\dot{e}}_{n+1} = \frac{1}{T}\left(e_{n+1} - e_{p_n}\right)$$

$$e_{p_{n+1}} = e_{n+1}$$

$$\delta T_{c_{n+1}} = K\left(T_L\, \hat{\dot{e}}_{n+1} + e_{n+1}\right)$$

$$\delta T_{n+1} = e^{-aT}\, \delta T_n + \left(1 - e^{-aT}\right) \delta T_{c_{n+1}}$$

$$\ddot{d}_{n+1} = Z_w\, \dot{d}_n - Z_{\delta T}\, \delta T_{n+1}$$

$$t_{n+1} = t_n + T$$

$$\dot{d}_{n+1} = \dot{d}_n + \frac{T}{2}\left(3\, \ddot{d}_{n+1} - \ddot{d}_{p_n}\right)$$

$$\ddot{d}_{p_{n+1}} = \ddot{d}_{n+1}$$

$$d_{n+1} = d_n + \frac{T}{2}\left(\dot{d}_{n+1} + \dot{d}_{p_n}\right)$$

$$\dot{d}_{p_{n+1}} = \dot{d}_{n+1}$$

TABLE 11

CASE 2 z-DOMAIN EQUATIONS

$$ze = d_c - d \qquad zt = t + T$$

$$z\hat{\dot{e}} = \frac{1}{T}(ze - e_p) \qquad z\dot{d} = \dot{d} + \frac{T}{2}(3z\,\ddot{d} - \ddot{d}_p)$$

$$ze_p = ze \qquad z\ddot{d}_p = z\ddot{d}$$

$$z\delta_{T_c} = K(T_L\, z\hat{\dot{e}} + ze) \qquad zd = d + \frac{T}{2}(z\dot{d} + \dot{d}_p)$$

$$z\delta_T = e^{-aT}\,\delta_T + (1 - e^{-aT})\, z\,\delta_{T_c} \qquad z\dot{d}_p = z\dot{d}$$

$$z\ddot{d} = Z_w\,\dot{d} - Z_{\delta_T}\, z\,\delta_T$$

And, after making substitutions

$$e = \frac{d_c - d}{z} \qquad \frac{\ddot{d}}{\delta_T} = \frac{-Z_{\delta_T}\, z(z - 1)}{z(z - 1) - Z_w \frac{T}{2}(3z - 1)}$$

$$\delta_{T_c} = \frac{K\left(\frac{T_L}{T} + 1\right)\left(z - \frac{T_L/T}{1 + T_L/T}\right)}{z}\, e \qquad \frac{\dot{d}}{\ddot{d}} = \frac{\frac{T}{2}(3z - 1)}{(z - 1)}$$

$$\frac{d}{\dot{d}} = \frac{T}{2}\frac{(z + 1)}{(z - 1)}$$

$$\delta_T = \frac{(1 - e^{-aT})z}{z - e^{-aT}}\,\delta_{T_c}$$

$$G(z) = \frac{\delta_{T_c}}{d_c - d} \cdot \frac{\delta_T}{\delta_{T_c}} \cdot \frac{\ddot{d}}{\delta_T} \cdot \frac{\dot{d}}{\ddot{d}} \cdot \frac{d}{\dot{d}}$$

$$= \frac{K\left(\frac{T_L}{T} + 1 \quad z - \frac{T_L/T}{1 + T_L/T}\right)}{z^2} \cdot \frac{(1 - e^{-aT})z}{(z - e^{-aT})}$$

$$\cdot \frac{(-Z_{\delta_T})\, z(z - 1)}{z(z - 1) - Z_w \frac{T}{2}(3z - 1)} \cdot \frac{\frac{T}{2}(3z - 1)}{(z - 1)} \cdot \frac{T}{2} \frac{(z + 1)}{(z - 1)}$$

Substituting numerical values

$$G(z) = \frac{2.4442\times10^{-4}(z-0.95238)(z-0.33333)(z+1)}{(z-0.985113)(z-1)(z-0.951229)(z-0.0076133)}$$

and

$$\frac{G(z)}{1 + G(z)} = \frac{2.4442\times10^{-4}(z-0.95238)(z-0.33333)(z+1)}{(z-0.95142)(z-0.0075306)[-0.99987;0.99243]}$$

Analysis Results

The resulting frequency and time responses are shown in Figs. 17 and 18. Note that as we would expect the amplitude of the aliases are significantly higher than for Case 1 and that the digital effects are prominent in the time histories. The somewhat surprising result is that the overall effective delay is not as great as in the actual computer/aircraft system (Case 1) as reflected by the less rapid loss in phase. It is suspected that this is a result of the engine response algorithm (a direct z-transform of a first order lag) adding an excess of phase advance. This matter was not explored here but could be using the technique illustrated in Section II.A.3 to analyze algorithmic delay.

Case 3: Simulation of a Plant But Controlled by a Second Digital Computer — Frame Times Common and Synchronous

This case represents the easiest two-computer configuration to analyze, namely, two computers in series but synchronized to a common frame time.

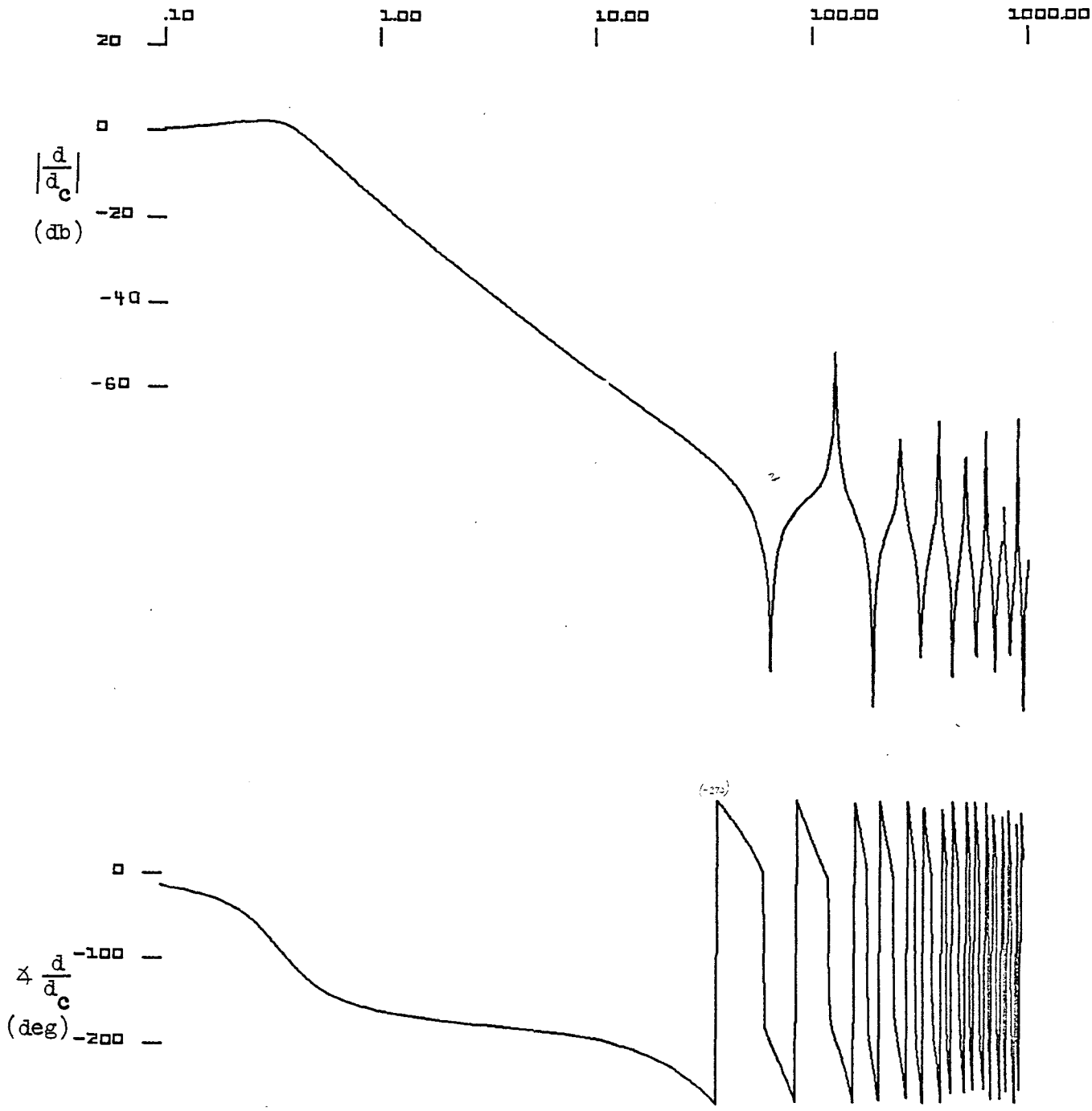


Figure 17. Case 2 Frequency Response of d/d_c

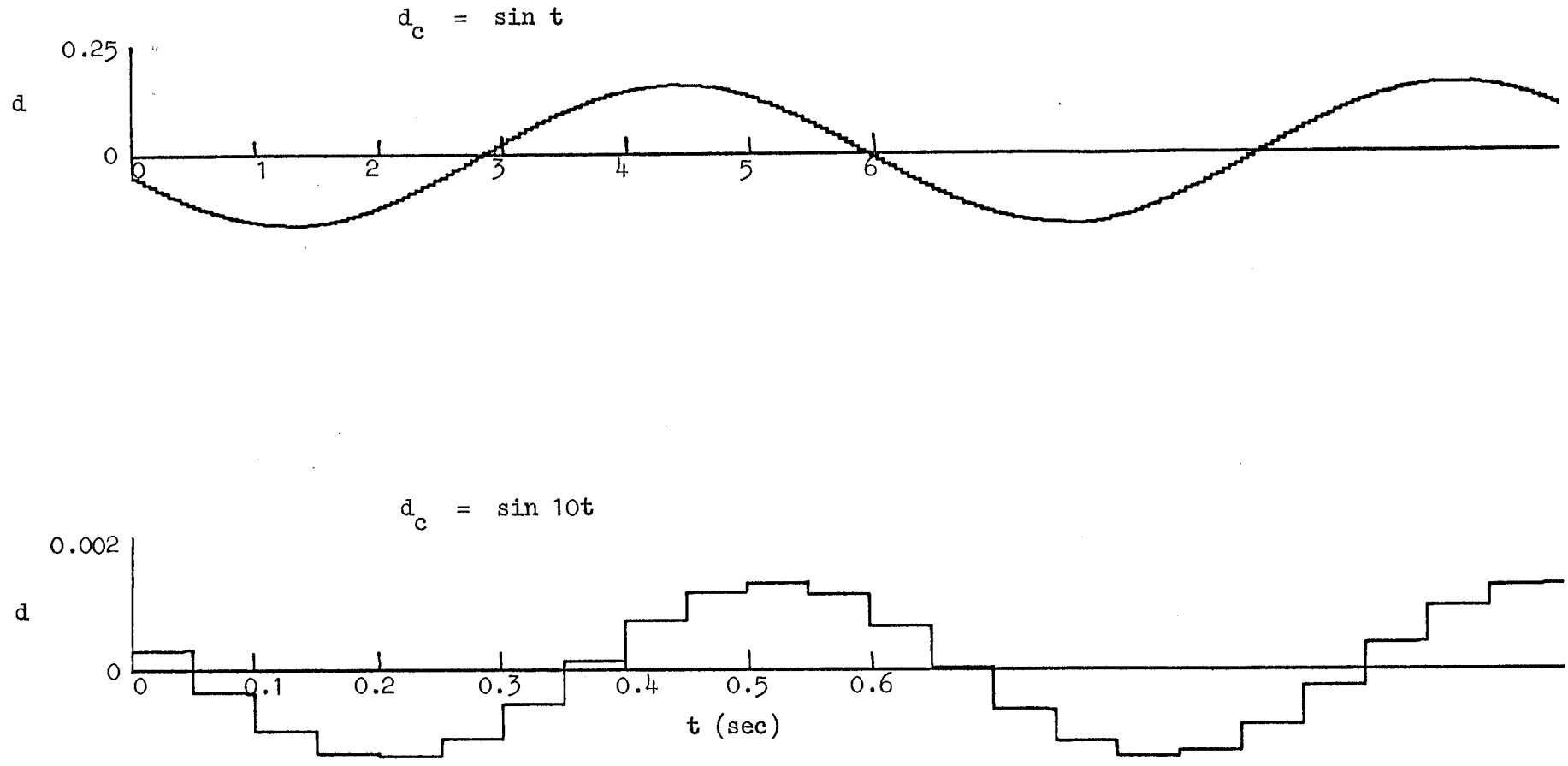


Figure 18. Case 2 Response of d to Sinusoidal Inputs of d_c

The essential elements are given in Table 12 and Fig. 18. Note that the digital program instructions are identical to those in Case 2 but they are divided between the two computers — the control function in one computer and the remaining airframe and propulsion models in the other.

Evaluation of System Elements

The important aspect in this case is handling correctly the interaction between the two computers. As an aid to bookkeeping, Fig. 20 diagrams the events occurring in the two computers during the frame in which the "n+1" computations are made. Corresponding to this Table 13 lists the difference equations taken one-computer-at-a-time as suggested in Section II.D, and Table 14 lists the z-domain equations, hence:

$$G(z) = \frac{\delta T_c}{d_c - d} \cdot \frac{\delta_T}{\delta T_c} \cdot \frac{\ddot{d}}{\delta_T} \cdot \frac{\dot{d}}{\ddot{d}} \cdot \frac{d}{\dot{d}} = G_2^T M_o^{T^1} G_1^T$$

$$= \frac{K\left(\frac{T_L}{T} + 1\right)\left(z - \frac{T_L/T}{1 + T_L/T}\right)}{z^2} \cdot \frac{(1 - e^{-aT})}{(z - e^{-aT})}$$

$$\cdot \frac{(-Z_{\delta_T})\, z(z - 1)}{z(z - 1) - Z_w \frac{T}{2}(3z - 1)} \cdot \frac{\frac{T}{2}(3z - 1)}{(z - 1)} \cdot \frac{T}{2}\frac{(z + 1)}{(z - 1)}$$

Finally, substituting numerical values:

$$G(z) = \frac{2.4442\times10^{-4}(z-0.95238)(z-0.33333)(z+1)}{(z-0.985113)(z-1)(z-0.951229)(z-0.0076133)z}$$

and

$$\frac{G(z)}{1 + G(z)} = \frac{2.4442\times10^{-4}(z-0.952381)(z-0.33333)(z+1)}{(z-0.951421)(z-0.0060569)(z+0.013666)[-0.999868;0.992590]_z}$$

Compare the closed loop denominator with Case 2.

TABLE 12

ESSENTIAL SYSTEM DESCRIPTION ELEMENTS FOR CASE 3

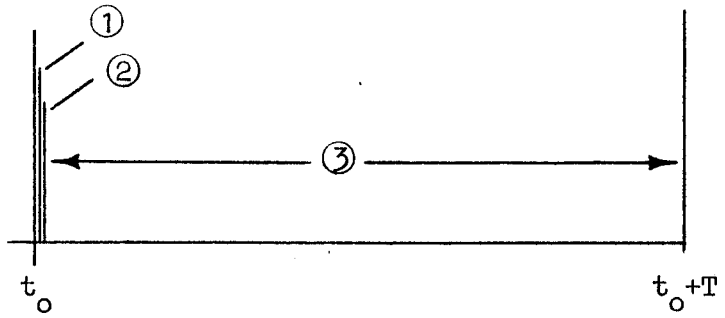
1. Control Computer FORTRAN Program

```
SUBROUTINE CONTROL
E=DC-D
EDOTHT=(E-EP)/DT
EP=E
DELTC=RK*(TL*EDOTHT+E)
RETURN
END
```

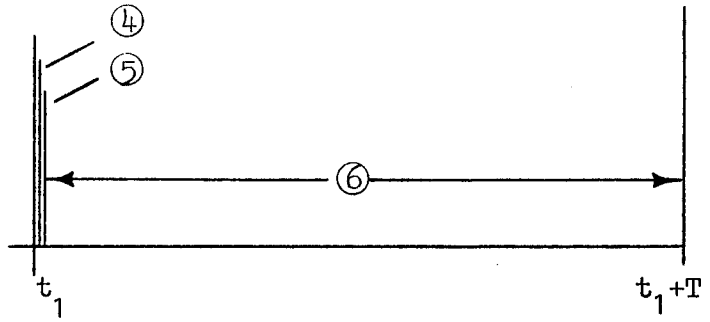
2. Simulator Computer FORTRAN Program

```
SUBROUTINE ENGINE
DELT=EXP(-A*DT)*DELT+(1-EXP(-A*DT))*DELTC
RETURN
END

SUBROUTINE AERO
DDDOT=ZW*DDOT-ZDT*DELT
RETURN
END

SUBROUTINE SMART
TIME=TIME+DT
DDOT=DDOT+DT/2*(3*DDDOT-DDDOTP)
DDDOTP=DDDOT
D=D+DT/2*(DDOT+DDOTP)
DDOTP=DDOT
RETURN
END
```

3. Control Computer Timing Diagram

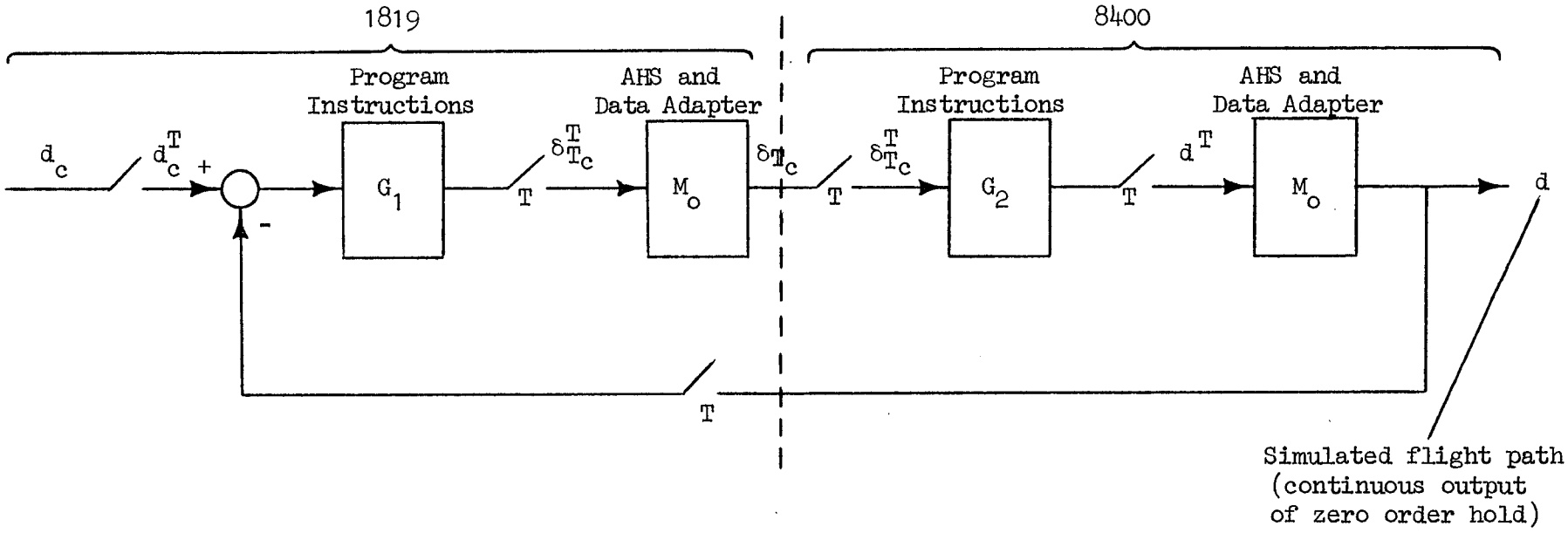


4. Simulator Computer Timing Diagram



① d_{c_n}, d_n input to control computer
② δT_{c_n} output to buffer register
③ Execute "Contr2"
④ δT_{c_n} input to simulator computer
⑤ d_n output to visual and motion devices
⑥ "Engine," "Aero," and "Smart" executed

Two-Computer, Single Frame Time Simulation



$$d = M_o \left(I + G_2^T M_o^T G_1^T M_o^T\right)^{-1} G_2^T M_o^T G_1^T d_c^T$$

Note: $M_o^T = 1$

Therefore $$d = M_o \left(I + G_2^T G_1^T\right)^{-1} G_2^T G_1^T d_c^T$$

Figure 19. Case 3 Functional Block Diagram

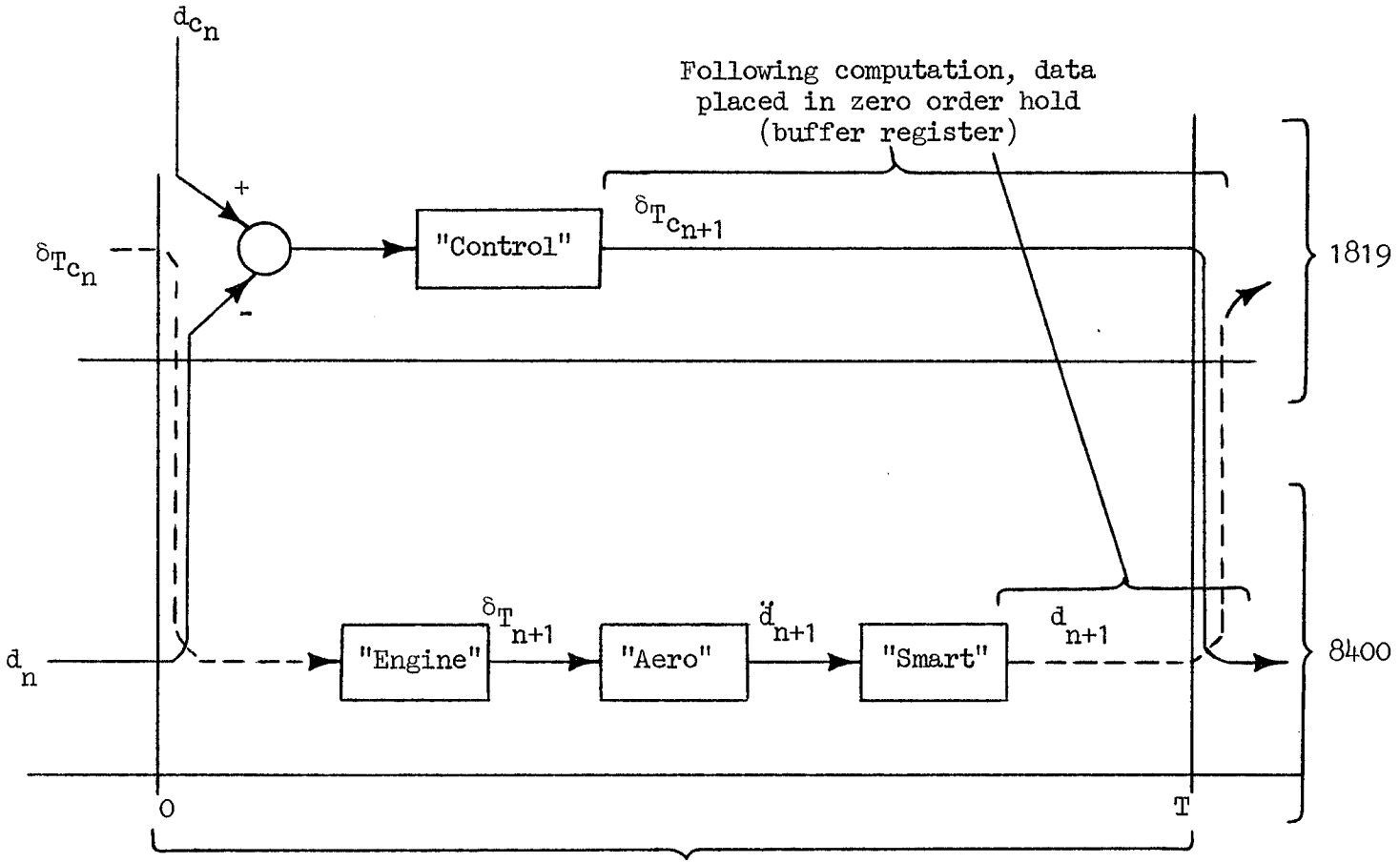


Figure 20. Two-Computer Timing Diagram Required for Formulating Analysis

TABLE 13

CASE 3 DIFFERENCE EQUATIONS

Flight Control Computer (1819)

$$e_{n+1} = d_{c_n} - d_n$$

$$\hat{\dot{e}}_{n+1} = \frac{1}{T}\left(e_{n+1} - e_{p_n}\right)$$

$$e_{p_{n+1}} = e_{n+1}$$

$$\delta T_{c_{n+1}} = K\left(T_L \hat{\dot{e}}_{n+1} + e_{n+1}\right)$$

Simulation Computer (8400)

$$\delta T_{n+1} = e^{-aT} \delta T_n + \left(1 - e^{-aT}\right) \underline{\delta T_{c_n}}$$

Note that δT_c is passed from the previous frame!

$$\ddot{d}_{n+1} = Z_w \dot{d}_n - Z_{\delta T} \delta T_{n+1}$$

$$t_{n+1} = t_n + T$$

$$\dot{d}_{n+1} = \dot{d}_n + \frac{T}{2}\left(3\ddot{d}_{n+1} - \ddot{d}_{p_n}\right)$$

$$\ddot{d}_{p_{n+1}} = \ddot{d}_{n+1}$$

$$d_{n+1} = d_n + \frac{T}{2}\left(\dot{d}_{n+1} + \dot{d}_{p_n}\right)$$

$$\dot{d}_{p_{n+1}} = \dot{d}_{n+1}$$

TABLE 14

CASE 3 z-DOMAIN EQUATIONS

$$ze = d_c - d \qquad zt = t + T$$

$$z\hat{\dot{e}} = \frac{1}{T}(ze - e_p) \qquad z\dot{d} = \dot{d} + \frac{T}{2}(3z\,\ddot{d} - \ddot{d}_p)$$

$$ze_p = ze \qquad z\ddot{d}_p = z\ddot{d}$$

$$z\delta_{T_c} = K(T_L\, z\hat{\dot{e}} + ze) \qquad zd = d + \frac{T}{2}(z\dot{d} + \dot{d}_p)$$

Note the loss of z compared to Case 2

$$z\delta_T = e^{-aT}\,\delta_T + (1 - e^{-aT})\,\delta_{T_c} \qquad z\dot{d}_p = z\dot{d}$$

$$z\ddot{d} = -Z_w\,\dot{d} - Z_{\delta_T}\, z\,\delta_T$$

And, after making substitutions

$$e = \frac{d_c - d}{z} \qquad \frac{\ddot{d}}{\delta_T} = \frac{-Z_{\delta_T}\, z(z-1)}{z(z-1) - Z_w \frac{T}{2}(3z-1)}$$

$$\delta_{T_c} = \frac{K\left(\frac{T_L}{T} + 1\right)\left(z - \frac{T_L/T}{1 + T_L/T}\right)}{z}\, e \qquad \frac{\dot{d}}{\ddot{d}} = \frac{\frac{T}{2}(3z-1)}{(z-1)}$$

$$\delta_T = \frac{(1 - e^{-aT})}{z - e^{-aT}}\,\delta_{T_c} \qquad \frac{d}{\dot{d}} = \frac{T}{2}\frac{(z+1)}{(z-1)}$$

Analysis Results

The frequency and time responses for Case 3 are shown in Figs. 21 and 22. Note that the frequency response amplitude is very close (but not identical) to Case 2. The phase loss is more rapid, however, owing to the excess z in the denominator — a result of linking two digital processors in series.

Case 4. Digital Simulation of Plant Using a Separate Flight Control Computer — Both Running at Different Frame Times

This case approaches a typical simulator situation involving two interacting digital computers but operating at different frame times. The same simulation computer programs assumed for Case 3 will apply for this case.

A vector switch decomposition approach will be used to handle the multirate aspect. In order to minimize the dimensions of the vector switching a 3:2 ratio will be used for the two frame times, i.e.,

$$T_1 = 75 \text{ msec (8400 frame time)}$$

$$T_2 = 50 \text{ msec (1819 frame time)}$$

Thus the common sample time is 150 msec.

The essential system elements consist of the program instructions from Case 3 (Table 12), the timing diagram shown in Fig. 23, and the block diagram shown in Fig. 24.

Evaluation of System Elements

Case 4 demonstrates how the multirate system is handled using a vector switch decomposition approach. The following shows how the various components in Fig. 24 are evaluated in explicit terms thus permitting the frequency or time domain analyses demonstrated in previous cases. Note the effect of sampling at T/2 and T/3.

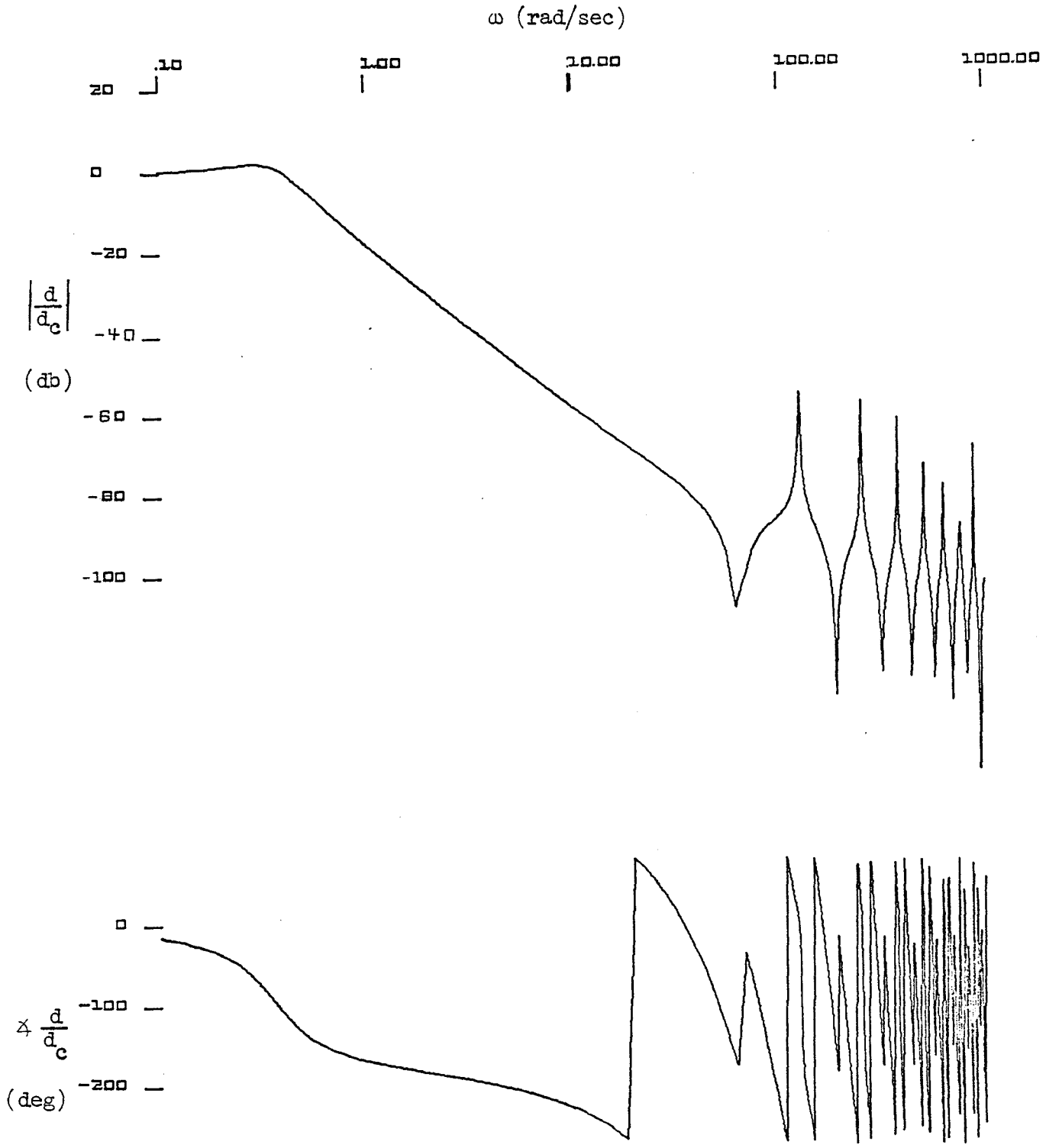


Figure 21. Case 3 Frequency Response of d/d_c

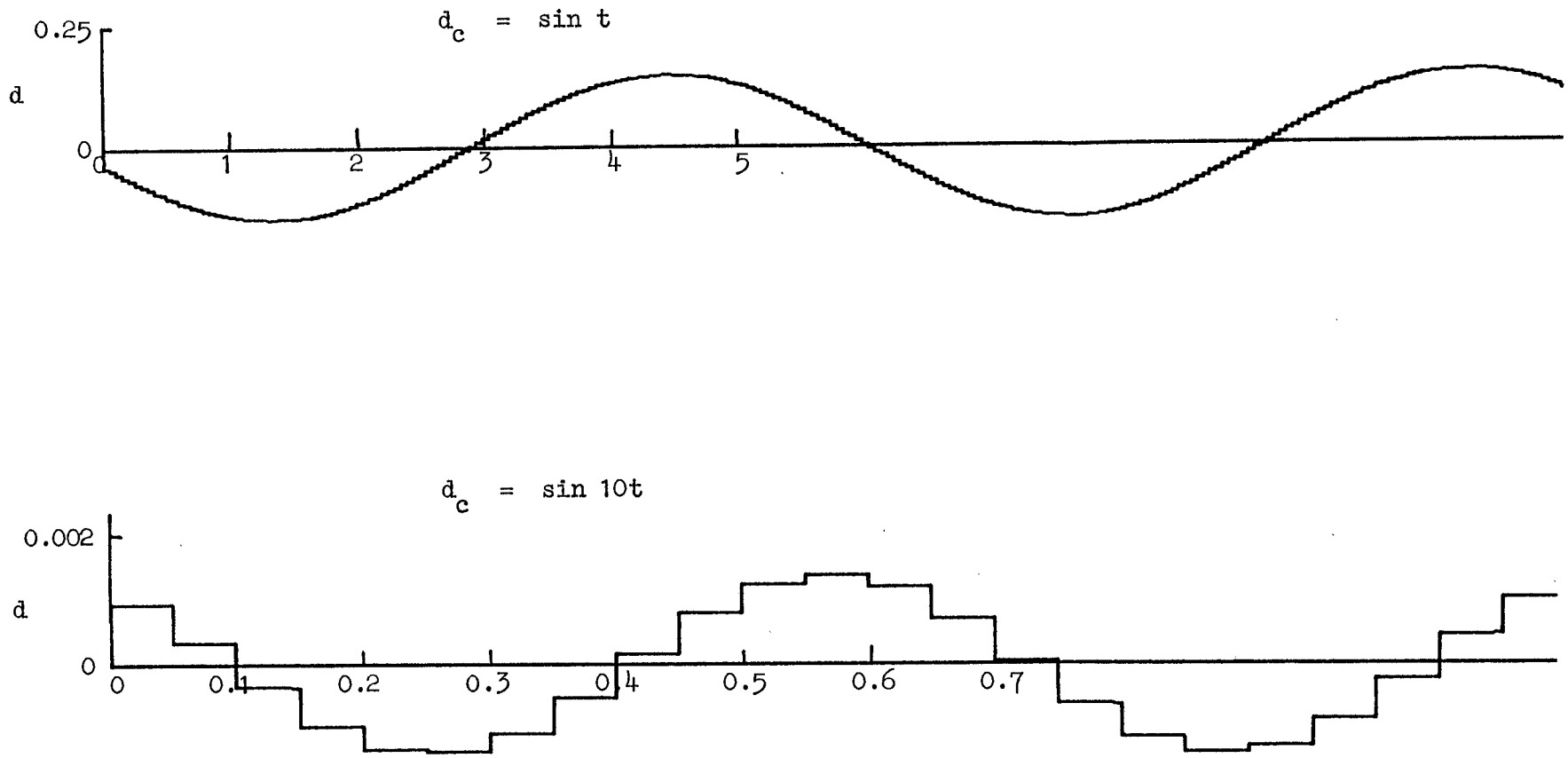


Figure 22. Case 3 Response of d to Sinusoidal Inputs of d_c

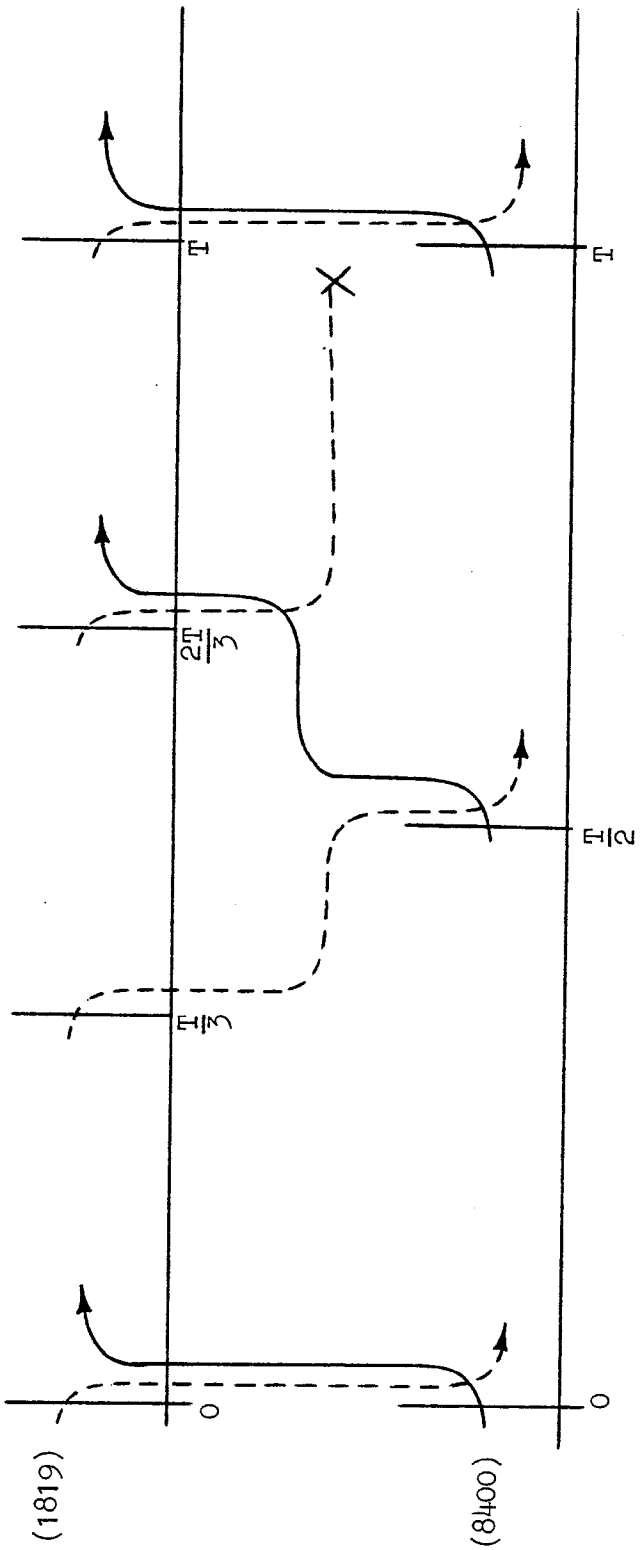
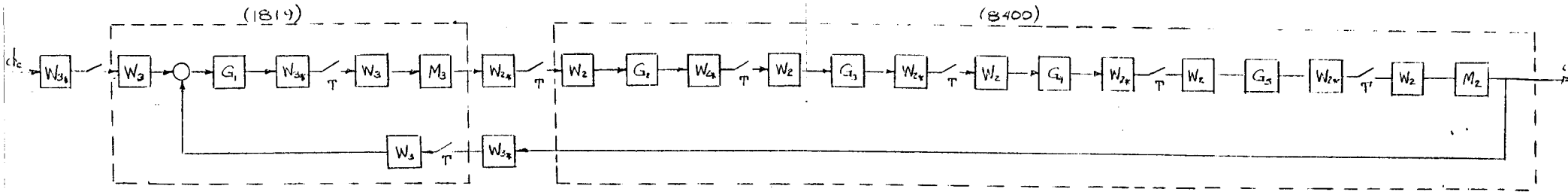


Figure 23. Data Transfer Between the Two Computers in Case 4
(to be handled with vector switch decomposition)



$$d = M_2 W_2 \left[A^T\right]^{-1} B^T \left(W_{3*}\, d_c\right)^T$$

where $$B^T = \left(W_{2*}\, G_5\, W_4\right)^T \left(W_{2*}\, G_4\, W_4\right)^T \left(W_{2*}\, G_3\, W_4\right)^T \left(W_{2*}\, G_2\, W_2\right)^T \left(W_{2*}\, M_3\, W_3\right)^T \left(W_{3*}\, G_1\, W_3\right)^T \left(W_{3*}\, M_2\, W_2\right)^T$$

and $$A^T = I + B^T \left(W_{3*}\, M_2\, W_2\right)^T$$

Figure 24. Case 4 Functional Block Diagram

Seven elements must be evaluated in formulating this case. Two involve vector switching of zero-order data holds with solutions based on relatively simple arguments. The remaining five elements require the more formal evaluation of a convolution integral using the method of residues as summarized in Table 15. The explicit transfer function quantities involved here are:

$$z_2 \triangleq e^{(sT)/2}$$

$$z_3 \triangleq e^{(sT)/3}$$

$$W_2 = [1 \; z_2^{-1}]$$

$$W_{2_*} = \begin{bmatrix} 1 \\ z_2 \end{bmatrix}$$

$$W_3 = [1 \; z_3^{-1} \; z_3^{-2}]$$

$$W_{2_*} = \begin{bmatrix} 1 \\ z_3 \\ z_3^2 \end{bmatrix}$$

$$G_1^{T/3} = \frac{K\left(\frac{3T_L}{T} + 1\right)\left(z_3 - \frac{3T_L/T}{1 + 3T_L/T}\right)}{z_3}$$

$$G_2^{T/2} = \frac{\left[1 - e^{-(aT)/2}\right]z_2}{z_2 - e^{-(aT)/2}}$$

$$G_3^{T/2} = \frac{-z_{\delta_T} z_2 (z_2 - 1)}{z_2 (z_2 - 1) - z_w \frac{T}{4} (3z_2 - 1)}$$

$$G_4^{T/2} = \frac{\frac{T}{4} (3z_2 - 1)}{z_2 - 1}$$

TABLE 15

SUMMARY OF EVALUATION BY METHOD OF RESIDUES†

$$\left[W_{a_*}(z_n)\, G(z_n)\, W_a(z_n) \right]^T \equiv C(z)$$ (a matrix where vector switch decomposition is involved)

where $z_n \equiv e^{(sT)/n}$ and $z = e^{sT}$

$$C_{ik}(z) = \frac{1}{2\pi j} \int_{\Gamma} f_{ik}(z_n)\, dz_n$$

where $f_{ik}(z_n) \equiv \left[W_{a_*}\, G\, W_a \right]_{ik} \frac{z}{[z - z_n^n]_{z_n}}$

and $C_{ik}(z) = \sum \text{Res.}\, [f_{ik}(z_n)]$

with

$$\underset{(z_n = a)}{\text{Res.}}\, [f_{ik}(z_n)] = \frac{1}{(m-1)!} \lim_{z_n \to a} \left\{ \frac{d^{m-1}}{dz_n^{m-1}} \left[(z_n - a)^m f_{ik}(z_n) \right] \right\}$$

where m is the number of poles at $z_n = a$.

For the case where m = 1

$$\underset{(z_n = a)}{\text{Res.}}\, [f_{ik}(z)] = \lim_{z_n \to a} \left[(z_n - a)\, f_{ik}(z_n) \right]$$

† The technique summarized here was adapted from Ref. 2 (vis. Eqs. 9.18 or 9.19 on p. 227).

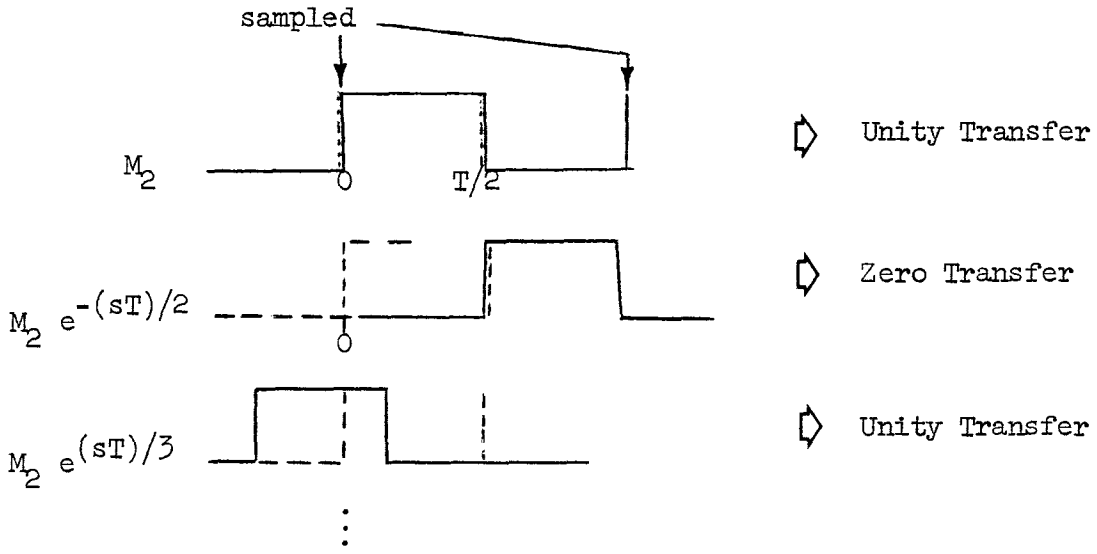
$$G_5^{T/2} = \frac{\frac{T}{4}(z_2 + 1)}{(z_2 - 1)}$$

(1) Buffer Register for the 8400

$$(W3_* \; M_2 \; W_2)^T = \left\{ \begin{bmatrix} 1 \\ e^{(sT)/3} \\ e^{(2sT)/3} \end{bmatrix} \left(\frac{1 - e^{-(sT)/2}}{s} \right) \begin{bmatrix} 1 & e^{-(sT)/2} \end{bmatrix} \right\}^T$$

$$= \left\{ \left(\frac{1 - e^{-(sT)/2}}{s} \right) \begin{bmatrix} 1 & e^{-(sT)/2} \\ e^{(sT)/3} & e^{-(sT)/6} \\ e^{(2sT)/3} & e^{(sT)/6} \end{bmatrix} \right\}^T$$

This represents the output of T/2 zero-order hold which is shifted in time and sampled at T (or 0).



Note that sampling at T will miss the input to the T/2 zero order hold if the input is delayed from 0^+ to T/2 or if it is shifted ahead T/2 to T^-. Thus,

$$(W_{3*}\ M_2\ W_2)^T = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(2) Buffer Register for the 1819

In a manner similar to that above,

$$(W_{2*}\ M_3\ W_3)^T = \left\{ \begin{bmatrix} 1 \\ e^{(sT)/2} \end{bmatrix} \frac{1 - e^{-(sT)/3}}{s} \begin{bmatrix} 1 & e^{-(sT)/3} & e^{-(2sT)/3} \end{bmatrix} \right\}^T$$

$$= \left\{ \frac{1 - e^{-(sT)/3}}{s} \begin{bmatrix} 1 & e^{-(sT)/6} & e^{-(2sT)/3} \\ e^{(sT)/2} & e^{(sT)/6} & e^{-(sT)/6} \end{bmatrix} \right\}^T$$

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

(3) "Control" Subroutine

$$W_{3*}\ G_1^{T/3}\ W_3 = \begin{bmatrix} 1 & z_3^{-1} & z_3^{-2} \\ z_3 & 1 & z_3^{-1} \\ z_3^2 & z_3 & 1 \end{bmatrix} \frac{a_o z_3 + a_1}{z_3}$$

where $$a_o \equiv K\left(\frac{3T_L}{T} + 1\right)$$

and $$a_1 \equiv K\frac{3T_L}{T}$$

Evaluating this by the method of residues:

$$\left[W_{3_*}\ G_1^{T/3}\ W_3\right]^T = \begin{bmatrix} a_o & 0 & a_1\ p^{-1} \\ a_1 & a_o & 0 \\ 0 & a_1 & a_o \end{bmatrix}$$

where $p \triangleq z^6 = e^{-sT}$

(4) "Engine" Subroutine

$$W_{2_*}\ G_2^{T/2}\ W_2 = \begin{bmatrix} 1 & z_2^{-1} \\ z_2 & 1 \end{bmatrix} \frac{\left[1 - e^{-(aT)/2}\right] z_2}{z_2 - e^{-(aT)/2}}$$

and $$\left[W_{2_*}\ G_2^{T/2}\ W_2\right]^T = \frac{(1 - e^{-aT})}{(p - e^{-aT})} \begin{bmatrix} p & e^{-(aT)/2} \\ e^{-(aT)/2}p & p \end{bmatrix}$$

(5) "Aero" Subroutine

$$W_{2_*}\ G_3\ W_2 = \begin{bmatrix} 1 & z_2^{-1} \\ z_2 & 1 \end{bmatrix} \underbrace{\frac{(-Z_{\delta_T})\ z_2\ (z_2 - 1)}{z_2\ (z_2 - 1) - Z_w \frac{T}{4}\ (3z_2 - 1)}}_{(z_2 - b_1)(z_2 - b_2)}$$

and

$$\left[W_{2_*}\ G_3\ W_2\right]^T =$$

$$\frac{p}{(p-b_1^2)(p-b_2^2)}\begin{bmatrix} p - \dfrac{b_1^2(1-b_2)-b_2^2(1-b_1)}{(b_1-b_2)} & \dfrac{b_1(b_1-1)-b_2(b_2-1)}{b_1-b_2} - \dfrac{b_1b_2}{z} \\ z\left[\dfrac{b_1(b_1-1)-b_2(b_2-1)}{b_1-b_2}\right]-b_1b_2 & p - \dfrac{b_1^2(1-b_2)-b_2^2(1-b_1)}{(b_1-b_2)} \end{bmatrix}$$

(6) Numerical Integration Subroutines

$$W_{2_*}\ G_4\ W_2 = \begin{bmatrix} 1 & z_2^{-1} \\ z_2 & 1 \end{bmatrix} \frac{\frac{T}{4}(3z_2 - 1)}{z_2 - 1}$$

$$W_{2_*}\ G_5\ W_2 = \begin{bmatrix} 1 & z_2^{-1} \\ z_2 & 1 \end{bmatrix} \frac{\frac{T}{4}(z_2 + 1)}{(z_2 - 1)}$$

and

$$\left[W_{2_*}\ G_4\ W_2\right]^T = \frac{T}{(p-1)} \begin{bmatrix} \frac{3p-1}{4} & \frac{1}{2} \\ \frac{p}{2} & \frac{3p-1}{4} \end{bmatrix}$$

$$\left[W_{2_*}\ G_5\ W_2\right]^T = \frac{T}{p-1} \begin{bmatrix} \frac{p+1}{4} & \frac{1}{2} \\ \frac{p}{2} & \frac{p+1}{4} \end{bmatrix}$$

In order to calculate the continuous frequency response of d/d_c we would have to evaluate the mixed-domain expression,

$$\frac{d}{d_c}(j\omega_n) = \frac{M_2 W_2}{T}\Bigg|_{s=j\omega_n}\left\{\left[A^T\right]^{-1} B^T\right\}\Bigg|_{p=e^{j\omega_n T}} W_{3_*}\Bigg|_{s=jb}$$

$$\equiv x_n + j\, y_n$$

where b is the fundamental frequency and ω_n is the n-th alias,

$$\omega_n = b + \frac{2\pi}{T} n, \quad n = 0, 1, 2, \text{ etc.}$$

The steady-state output, d_{ss}, due to a sine wave input, $d_c = \sin bt$, is given by

$$d_{ss} = \sum_{n=0}^{\infty} (x_n \sin \omega_n t + y_n \cos \omega_n t)$$

The elements of the A^T and B^T matrices could be evaluated at each value of ω_n and then multiplied together, or pre-multiplied and then evaluated at each ω_n. Either way, however, would require the aid of a digital computer.

SECTION IV

CONCLUSIONS AND RECOMMENDATIONS

As a result of the work reported, we have established a general procedure for evaluating a particular simulator implementation against any given standard such as the actual system counterpart. This evaluation can be made in either the frequency or time domain.

Although the simulator implementations illustrated were non-trivial, they were nevertheless uncomplicated — by order and linearity — in comparison to many simulator models utilized at Ames Research Center and other simulator facilities. But the real limiting factor in handling more complex systems is in the availability of analysis software. The software developed during this project represents a starting point. Based on the techniques cited in Ref. 1, this software can handle multi-processor, single-rate systems of almost any order. The key limitation is, of course, the single-rate aspect, a subject we shall discuss shortly.

Let us now consider the value of the foregoing analysis results to the simulator user under the following specific headings:

- Metrics for system fidelity
- Problem detection and correction
- Recommended experiments and analyses.

A. METRICS FOR SYSTEM FIDELITY

The main metrics utilized in examining the cases of Section III were direct frequency and time domain comparisons. In particular, input-output gain and phase plots were used for frequency domain portraits, and a steady state response to a sine wave command (transients were allowed to die out) was used for a time domain measure. Both of these are primary metrics for considering system fidelity but are overly general. Clearly, an exact match of frequency or time domain responses constitutes a claim for fidelity. But this by itself is not of much value to the simulator user — exact

matches simply are not to be expected. The real question is, therefore, How close a match must one have in system response for acceptable fidelity?

We have found that the examples analyzed in the previous section did, in fact, exhibit discernible differences in frequency response. Time response differences were less clear (except where digital versus continuous time histories were involved). The frequency response differences were not particularly crucial, however. The delays involved were not sufficiently long to induce instabilities or effects otherwise perceptible to the pilot. Therefore it is difficult to infer response features, e.g., amplitude or phase, significant to a given level of fidelity. Consequently the examples are not particularly pathological insofar as system fidelity is concerned. Nevertheless some specific observations from the cases analyzed may be worth emphasizing.

Case 1 served as a baseline — a system which was to be simulated on a digital computer. Although a continuous-discrete hybrid, Case 1 exhibited digital effects which were highly attenuated in terms of d/d_c response (the first alias was down more than 120 db). When simulated on one or two digital computers (Cases 2 and 3) the amplitude of the first alias was much higher (down 50 to 60 db) the result of which was a noticeable digital effect on the time history of d.

The specific attenuation of the first alias could be used as a metric for what is commonly referred to as "graininess" or "digital appearance" of a particular variable. What must be determined is the level of acceptability versus a given alias amplitude attenuation relative to the pilot's various thresholds of perception (motion, visual, etc.).

The phase characteristics for the cases analyzed, as expected, reflected the overall loop delays. The most apparent feature was the rate of change of phase lag with respect to frequency. The gradient of only the initial phase roll-off up to 180 deg should be necessary for quantifying the effective overall loop delay unless the simulation is attempting to represent something such as high frequency rotor dynamics which might interact with digital artifacts.

In order to quantify specific metrics for judging piloted simulator fidelity, whether frequency or time domain, it will be necessary to develop correlations between specific fidelity-related features and a qualitative rating of goodness. This was not possible for the systems considered here. They were far too innocuous. It is suggested that attention be focused on specific cases where digital effects can be more critical, e.g., visual systems, motion systems, rotary-wing modeling, etc.

The matter of algorithmic effects was addressed briefly in Section II with regard to numerical integration. There frequency response analysis was used to infer the most suitable numerical integration algorithm for a given situation. It could be seen that the critical determinant was I/O timing or whether the integrand was based on calculations in the same frame or in the previous frame. Where the integral and integrand were taken simultaneously, a trapezoidal algorithm was the clear choice. Where a single frame delay was involved, a second-order Adams algorithm has the correct phase relationship to compensate for the staleness of the integrand. But for a double integration a new numerical integration algorithm was synthesized which would better match phase amplitude response better than the commonly used second order Adams-trapezoidal combination. The FORTRAN form of this new recursion equation is:

```
Y = Y + DT*X    (first integration-rectangular algorithm)
Z = Z + 0.818310 * DT * Y
        + 0.181690 * DT * YP    (second integration)
```

where X is the computed derivative of Y

Y is the computed derivative of Z

YP is the past value of X

DT is the frame time.

(Note for comparison: the second-order Adams algorithm is

```
Y = Y + 1.5 * DT * X - 0.5 * DT * XP
```

and the trapezoidal algorithm is

```
Z = Z + 0.5 * DT * Y + 0.5 * DT * YP)
```

B. PROBLEM DETECTION AND CORRECTION

One particularly useful means of employing the analysis methods presented in these two volumes is in problem detection and correction. A variety of possible approaches exist.

- Time response comparison of the desired simulator implementation to the actual simulator implementation
- Frequency response comparison of the simulator implementation to the actual system
- Determination of simulator system stability — via either z-domain or T-transition matrix (really equivalent but two different formulation approaches can be taken)
- Frequency response comparison of competing simulator implementations, especially regarding algorithms, timing, and fast loop/slow loops.

The key to problem detection and correction is in having effective means of checking simulator model implementations. One means is simply preparing independent check cases which can be compared with the simulation, and this is the nature of the suggestions listed above.

Normally check cases, if they are prepared, are done so in the continuous domain. We suggest that check cases be made to reflect the precise nature of the continuous-discrete system being simulated. In addition, we suggest check cases for the system as it is to be implemented on simulator computers.

This latter step is presently impractical with regard to the overall simulator system — it is normally far too complex and the existing analysis software too inadequate. Nevertheless certain critical elements could be isolated. Among these are landing gear, high frequency aerodynamic or control system features, or propulsion system features.

It is recommended that attention be given to developing and refining procedures for problem detection and correction as suggested above. This will entail development of software accessible to simulator users and programmers and the exercising of this software to establish routine procedures.

Perhaps the most difficult aspect of problem detection and correction software is in handling multirate or multiloop simulator systems. Although no such software capability was developed under this program, two multirate analysis methods were examined which have potential software applications.

In this volume the vector switch decomposition method was used to formulate a multirate system analysis. The formulation was straightforward, though tedious, and the software required to evaluate frequency response is not regarded to be a big step beyond that used in the single rate analyses. The aspect which is of concern, however, is the great amount of manual manipulation required in setting up the check cases and hence the room for introducing errors.

The transition matrix approach to analyzing multirate or multiloop systems is explored in Volume Two of this report. This method has its own special potential for problem detection and correction. It may be possible to generate a transition matrix directly from a simulator program. Once in hand, the transition matrix could be used to measure pole locations, system stability, and examine frequency response.

C. RECOMMENDED LABORATORY EXPERIMENTS AND FURTHER ANALYSES

The present results suggest several tasks which might be considered for future research. These tasks include verification of theoretical results using simulator hardware, further work on multirate or multiloop analysis, development of specific fidelity metrics, and application of analysis methods produced thus far to critical simulation areas.

Theoretical Verification

Perhaps the most immediate job is to verify that the results obtained analytically are valid and meaningful for actual simulator hardware. Since the analysis cases presented in this volume were centered around existing Ames Research Center hardware, namely the Sperry 1819 and EAI 8400 computers, it makes sense to follow through using this hardware.

The specific checks to back up the theoretical development should include time response and frequency response matching to verify:

- Computer interfacing assumptions
- Logic in translating program instructions to discrete domain transfer functions

Using the cases already established (Cases 1 through 3) the theoretical verification suggested here should require little effort.

Further Development of Multirate/Multiloop Analysis

The analysis Case 4 presented in Section III is a point of departure for development of multirate/multiloop analysis tools. To this end, the existing time and frequency response software could be extended. Also the hardware verification mentioned above should be applied to any such extension of analysis tools.

Development of Specific Fidelity Metrics

Data are required in order to relate frequency response features to particular fidelity aspects. Acquisition of such data should be centered around known fidelity problems. Two areas in which such problems exist are modeling of rotary-wing aerodynamics and simulator subsystems (e.g., motion, visual, etc.).

Rotary-wing simulation presents two difficult fidelity problems. First there is the question of fidelity of the system as it is modeled (e.g., do the high frequency rotor modes of an eight- or nine-degree-of-freedom continuous model provide the necessary fidelity to the vehicle being simulated?). Second, is the question of fidelity of the model as it is implemented in the simulator (e.g., are the continuous rotor modes accurately implemented in the digital simulation?). Neither of these fidelity questions can be addressed without also addressing the other, and this confounds the strictly digital aspects.

The increased complexity of the current generation of external-view visual and motion-platform devices has in several cases actually reduced

their fidelity (Refs. 13 and 14). Seemingly small simulation artifacts (such as delays in visual field updating for computer generated images and spurious motion cues due to motion limiting functions) have negated their intended benefit in a pilot-vehicle closed loop task.

Other simulator subsystems are involved in the closed-loop pilot vehicle training system, such as the control "feel" system, the vehicle dynamics computer, and the visual and motion computers. Contrary to the opinion of Caro in Ref. 15, the vehicle dynamics computers are often not adequate in terms of computational update rates as recent studies on multirate-sampled systems for the Office of Naval Research, Air Force, and NASA have dramatically shown (e.g., Ref. 16). In fact, the extra sophistication in representing complex aerodynamics has taken its toll by introducing serious computational artifacts. Such problems are currently being attacked by NTEC and AFHRL using the ASPT facility (Ref. 17), but the proposed solutions— low pass (1 Hz) drive filters — will not solve the basic problem and will further reduce motion fidelity at frequencies where it is needed most!

REFERENCES

1. Whitbeck, Richard F., and L. G. Hofmann, Analysis of Digital Flight Control Systems with Flying Qualities Applications. Volume II: Technical Report, AFFDL-TR-78-115, September 1978 (Volume I is the Executive Summary).

2. Ragazzini, J. R., and G. F. Franklin, Sampled-Data Control Systems, McGraw-Hill, New York, 1958.

3. Jury, E. I., Sampled-Data Control Systems, John Wiley, New York, 1958.

4. Tou, J. T., Digital and Sampled-Data Control Systems, McGraw-Hill, New York, 1959.

5. Smith, Jon M., Mathematical Modeling and Digital Simulation for Engineers and Scientists, John Wiley and Sons, New York, 1977.

6. Hildebrand, F. B., Advanced Calculus for Engineers, Prentice-Hall, Inc., New York, 1949.

7. Sinacori, John B., Robert L. Stapleford, Wayne F. Jewell, and John M. Lehman, Researcher's Guide to the NASA Ames Flight Simulator for Advanced Aircraft (FSAA), NASA CR-2875, February 1977.

8. Hofmann, L. G., and R. F. Whitbeck, Hybrid System Frequency Response Case Study Example, Systems Technology, Inc., Working Paper No. 503-32, January 1978.

9. Anon., V/STOLAND Data for NASA/Ames Research Center, Performance Specification VSTL 5442-1033, Data Adapter, Part No. 4008174-204, Sperry Flight Systems, Phoenix, Arizona, December 1974.

10. Anon., Programmers Reference Manual, 1819B Digital Computer, Sperry Flight Systems, Pub. No. 61-0204-00-01, June 1976.

11. Lidén, Sam, V/STOLAND Digital Avionics System for UH-1H, Final Report, NASA CR-152179, October 1978.

12. Lidén, Sam, V/STOLAND Input/Output Specification and Program, Sperry Flight Systems, Pub. No. 71-0565-00-02, April 1978.

13. Orlansky, Jesse, and Joseph String, Cost-Effectiveness of Flight Simulators for Military Training, Vol. I: Use and Effectiveness of Flight Simulators, Institute for Defense Analyses, Paper P-1275, Aug. 1977.

14. Martin, Elizabeth, and Wayne L. Waag, Contributions of Platform Motion to Simulator Training Effectiveness: Study I — Basic Contact, AFHRL, TR-78-15, June 1978.

15. Caro, Paul W., Some Factors Influencing Air Force Simulator Training Effectiveness, AFOSR, Rept. No. HumPRO-TR-77-2, Mar. 1977.

16. Whitbeck, R. F., Simulation Error Analysis, Systems Technology, Inc., Working Paper No. 510-44, Mar. 1979.

17. Ricard, G. L., et al., Compensation for Transport Delays Produced by Computer Image Generation Systems, NAVTRAEQUIPCEN IH-297/AFHRL-TR-78-46, June 1978.

APPENDIX

COMPUTATION OF A DISCRETE TRANSFER FUNCTION, G(z), FROM A CONTINUOUS TRANSFER FUNCTION, G(s)

The technique described below has been used to compute G(z) based on a particular G(s). That is,

$$G(z) = [G(s)\ M_o]^T \qquad \text{(A-1)}$$

where M_o is a zero order data hold.

The algorithm was implemented on an HP-67 hand calculator which allowed up to a third order system for G(s) and a PDP-10 computer which allowed up to a 20th order system for G(s).

The first step is to express G(s) in polynomial form:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{A_N s^N + A_{N-1} s^{N-1} + \ldots + A_o}{s^M + B_{M-1} s^{M-1} + \ldots + B_o} \qquad \text{(A-2)}$$

Next express Eq. A-2 as a system of first order differential equations in a state space form. That is,

$$\underline{\dot{y}} = F\underline{y} + G\underline{x} \qquad \text{(A-3)}$$

where the underscore indicates a vector.

The result is

$$\begin{Bmatrix} \dot{y} - A_M \dot{x} \\ \dot{v}_1 \\ \dot{v}_2 \\ \vdots \\ \dot{v}_{M-1} \end{Bmatrix} = \begin{bmatrix} -B_{M-1} & 1 & & & \\ -B_{M-2} & & 1 & & \\ -B_{M-3} & & & 1 & \\ \vdots & & & & \ddots \\ -B_o & & & & & 1 \end{bmatrix} \begin{Bmatrix} y - A_M x \\ v_1 \\ v_2 \\ \vdots \\ v_M \end{Bmatrix}$$

$$+ \begin{Bmatrix} A_{M-1} - B_{M-1} A_M \\ A_{M-2} - B_{M-2} A_M \\ \vdots \\ A_o - B_o A_M \end{Bmatrix} \{x\} \qquad \text{(A-4)}$$

The discrete representation of Eq. A-3, assuming a zero-order data hold, is

$$\underline{y}_{n+1}^T = \Phi \, \underline{y}_n^T + \Gamma \, \underline{x}_n^T \qquad \text{(A-5)}$$

where

$$\Phi = e^{FT} \qquad \text{(A-6)}$$

$$\Gamma = \left[\int_o^T e^{F\lambda} \, d_\lambda \right] G \qquad \text{(A-7)}$$

$$T = \text{frame time in seconds} \qquad \text{(A-8)}$$

Taking the z-transform of Eq. A-5 we obtain

$$\left[z \, I - \Phi \right] \begin{Bmatrix} Y(z) - A_M X(z) \\ v_1(z) \\ \vdots \\ v_{M-1}(z) \end{Bmatrix} = \Gamma \{ X(z) \} \qquad \text{(A-9)}$$

The desired transfer function, G(z), is computed from Eq. A-9 as follows:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{A_M \Delta + N_x^{Y - A_M X}}{\Delta} \qquad \text{(A-10)}$$

where Δ is the characteristic equation and $N_x^{Y - A_M X}$ is the numerator for $(Y - A_M X)/X$.

The matrices Φ and Γ can both be computed using Taylor series expansions:

$$\Phi = \sum_{o}^{\infty} F^n \frac{T^n}{n!} \qquad \text{(A-11)}$$

$$\Gamma = \left[\sum_{o}^{\infty} F^n \frac{T^{(n+1)}}{(n+1)!}\right] G \qquad \text{(A-12)}$$

Computing F^n is fairly simple due to the sparseness of the F matrix defined in Eq. A-4. In fact recursion equations can be used to compute the first column of F^{n+1}:

$$f_{i,1}^{n+1} = \sum_{j=1}^{M} f_{i,j}^{n} (-B_{M-j}); \quad i = 1,M \qquad \text{(A-13)}$$

where $f_{i,j}^{n}$ is the i,j element of the F^n matrix. All other columns of F^{n+1} are obtained with a "push-down" stack:

$$f_{i,j+1}^{n+1} = f_{i,j}^{n} \; ; \; \begin{matrix} i = 1,M \\ j = 1,M-1 \end{matrix} \qquad \text{(A-14)}$$

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-152340		

4. Title and Subtitle	5. Report Date
THE ANALYSIS OF DELAYS IN SIMULATOR DIGITAL COMPUTING SYSTEMS. Volume One: Formulation of an Analysis Approach Using a Central Example Simulator Model	February 1980
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
R. K. Heffley, W. F. Jewell, R. F. Whitbeck, and T. M. Schulman	STI-TR-1140-1-I
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
Systems Technology, Inc. 2672 Bayshore-Frontage Road, Suite 505 Mountain View, California 94043	NAS2-10106
	13. Type of Report and Period Covered

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Ames Research Center Moffett Field, California 94035	Final Report

15. Supplementary Notes

16. Abstract

The effects of spurious delays in realtime digital computing systems are examined. In this, the first of a two volume series, various sources of spurious delays are defined and analyzed using, as a central example, an extant simulator system. A specific analysis procedure is set forth, and four cases are viewed in terms of their time and frequency domain characteristics. Numerical solutions are obtained for three single rate one- and two-computer examples, and the analysis problem is formulated for a two-rate, two-computer example. In the second volume of this series a separate approach to the two-computer, multirate problem is examined. At the conclusion of Volume One the results of the analyses from both volumes are discussed in terms of their potential value for fidelity metrics or simulator problem detection and correction aids.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Digital Simultaion Time Lag Flight Simulators Algorithms Digital Techniques Numerical Analysis	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	100	

*For sale by the National Technical Information Service, Springfield, Virginia 22161