STATE-SPACE TECHNIQUES FOR DIGITAL SIMULATION OF DYNAMIC SYSTEMS

by

RICHARD CARL BURKHARDT

A Thesis Submitted to the Faculty of the DEPARTMENT OF ELECTRICAL ENGINEERING
In Partial Fulfillment of the Requirements For the Degree of MASTER OF SCIENCE
In the Graduate College THE UNIVERSITY OF ARIZONA

1969
STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

John V. Wait
Associate Professor of Electrical Engineering

July 18, 1968
ACKNOWLEDGEMENTS

Sincere gratitude must be extended to Dr. John V. Wait for his help and guidance throughout the preparation of this thesis. His numerous suggestions were invaluable.

Thanks must also be given to David B. Leeds and Robert E. May of Control Data Corporation for their cooperation and understanding to help make these results possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF ILLUSTRATIONS.</td>
<td>vii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 History of the Simulation Methods.</td>
<td>1</td>
</tr>
<tr>
<td>1.2 The Basic Approach to the Problem.</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Summary of Results</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER II PRESENTATION OF THE TWO METHODS.</td>
<td>5</td>
</tr>
<tr>
<td>2.1 The General Approach</td>
<td>5</td>
</tr>
<tr>
<td>2.2 The Hurt-Fowler Approach</td>
<td>5</td>
</tr>
<tr>
<td>2.3 The State-Space Method</td>
<td>9</td>
</tr>
<tr>
<td>CHAPTER III DISCUSSION OF THE EXAMPLE PROBLEMS AND RESULTS.</td>
<td>19</td>
</tr>
<tr>
<td>3.1 The Linear Simulation</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Results from the Linear Simulation</td>
<td>21</td>
</tr>
<tr>
<td>3.3 The State-Space Solution for the Non-Linear Problem</td>
<td>29</td>
</tr>
<tr>
<td>3.4 The Hurt-Fowler Solution for the Non-Linear Problem</td>
<td>31</td>
</tr>
<tr>
<td>3.5 Non-Linear Simulation Results</td>
<td>34</td>
</tr>
<tr>
<td>CHAPTER IV CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK.</td>
<td>39</td>
</tr>
<tr>
<td>4.1 Problem Considerations</td>
<td>39</td>
</tr>
<tr>
<td>4.2 Improvements in the Simulation Method</td>
<td>40</td>
</tr>
<tr>
<td>4.3 Other Input Approximations</td>
<td>41</td>
</tr>
<tr>
<td>4.4 System Errors</td>
<td>43</td>
</tr>
<tr>
<td>4.5 Conclusion</td>
<td>44</td>
</tr>
<tr>
<td>APPENDIX I. SIMULATION PROGRAM CONCEPTS</td>
<td>46</td>
</tr>
<tr>
<td>A1.1 Basic Program Approach</td>
<td>46</td>
</tr>
<tr>
<td>A1.2 Non-Linear Program Modifications</td>
<td>48</td>
</tr>
<tr>
<td>A1.3 Provisions for Problem Extension</td>
<td>48</td>
</tr>
</tbody>
</table>
APPENDIX II  INPUT DATA FORMATS FOR SIMULATION PROGRAMS

A2.1 General Approach .......................... 51
A2.2 Linear Problem Formats ................. 51
A2.3 Non-Linear Problem Formats ............ 53

LIST OF REFERENCES ........................ 55
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Equations for Calculating System Constant Matrices</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Equations for Calculation of Simulation Expressions</td>
<td>18</td>
</tr>
</tbody>
</table>
LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>General Example System</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Sampler Separated System</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Input Approximation Concept</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>Impulse Responses of Input Approximations</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Linear Example System</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Plot of Peak Error vs Sample Period</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>for Linear Simulation, Step Input of 1.0</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Plot of Log Peak Error vs Log $\omega_n/\omega_{in}$</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>for Linear Simulation, Method I, Sine Inputs</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Plot of Log Peak Error vs Log $\omega_n/\omega_{in}$</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>for Linear Simulation, Method I, Sine Inputs</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Plot of Log Peak Error vs Log $\omega_n/\omega_{in}$</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>for Linear Simulation, Method II, Sine Inputs</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>Plot of Log Peak Error vs Log $\omega_n/\omega_{in}$</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>for Linear Simulation, Method IIIa, Sine Inputs</td>
<td></td>
</tr>
<tr>
<td>3.7</td>
<td>Plot of Log Peak Error vs Log $\omega_n/\omega_{in}$</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>for Linear Simulation, Method IV, $\zeta = .05$</td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>Non-Linear Example System</td>
<td>30</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.9</td>
<td>Plot of Peak Error vs Sample Period for Non-Linear Simulation, Step Input of 1.0</td>
<td>35</td>
</tr>
<tr>
<td>3.10</td>
<td>Plot of Peak Error vs Sample Period for Non-Linear Simulation, Sine Input of sin 2t</td>
<td>37</td>
</tr>
<tr>
<td>3.11</td>
<td>Plot of Error vs Time for Non-Linear Simulation, Sine Input of sin 2t (T=.1 seconds)</td>
<td>38</td>
</tr>
<tr>
<td>A1.1</td>
<td>Basic Linear Program Flow.</td>
<td>47</td>
</tr>
<tr>
<td>A1.2</td>
<td>Basic Non-Linear Program Flow.</td>
<td>49</td>
</tr>
</tbody>
</table>
This thesis presents three state-space methods for simulating dynamic systems. The simulation algorithms are derived from the system matrix first-order differential equations using a general state-variable method for designing discrete-time approximations to continuous-time systems. A FORTRAN computer program was developed for implementing the technique.

The methods were first used for the simulation of a general linear second-order system. They were then adapted to the simulation of a second-order non-linear example, and the results compared to those obtained from the Hurt-Fowler (IBM) method.
CHAPTER I

INTRODUCTION

1.1 History of the Simulation Methods

Until recently, real-time simulation of dynamic systems has commonly been done on an analog computer because most simulation required some form of integration and "slow" digital computers could not perform the integration in real-time. The major drawback of the analog computer was difficulty in reducing point by point data from the solution as displayed on an oscilloscope or some other device. The accuracy of the analog computer was lost in the transfer of actual data. Poor problem turn around time and the inability of the analog computer to do digital jobs associated with the main simulation problem were also drawbacks. The availability of floating-point arithmetic on the digital computer eliminates the variable scaling required on an analog machine. The advent of "fast" digital computers provided a help toward this end, but some techniques still need to be perfected to supplement conventional numerical integration methods in order to improve computing speed and convenience of problem programming.
Tustin's method for discrete approximation of continuous systems was one of the first advances in digital computation methods and is still in wide use (Sage and Smith 1966). Recently, the method introduced by Hurt and Fowler for non-linear simulations provided a significant advance in the field (Fowler 1965, Hurt 1964). A description of this basic method is presented in Chapter II.

1.2 The Basic Approach to the Problem

The state variable approach that is the basis for the simulation program developed here was derived from a set of algorithms designed for linear systems (Giese 1967, Shultz and Melsa 1967, Wait 1966). Discussion of ways to simulate fixed, linear systems, making errors only in the input approximation can be found in the List of References (IBM Report E20-0029.0, Wait 1966). Comparative empirical studies of how well various zero-order and first-order hold input approximations work in simulating a general second-order system for sine and step inputs are presented in Chapter III. A second-order system with a limiter was selected as an example for easy comparison to the example used by Hurt and Fowler (Hurt 1964).

The simulation programs were written for a basic linear case and then adapted to the non-linear example. The program was originally written in FORTRAN II for an IBM 7072/1401 system and then modified for use on a CDC
3200 system. The results that are presented here were obtained using the 3200 system.

Five input approximations obtained from realizable and non-realizable zero- and first-order holds are used for the system as discussed in Chapter II. Test inputs of sines and steps were used in the program examples.

The concept of this program is to provide a state-space method for simulating systems of the type in the problem examples with provisions for extension to more general classes of non-linear systems. Specifically, given any fixed (time invariant) linear system specified in state variable form, the program finds discrete-time algorithms for the system.

1.3 Summary of Results

Five types of input approximations were considered; Method I, an interpolative first-order hold, Method II, an extrapolative first-order hold, Method IIIa, an interpolative zero-order hold, Method IIIb, an extrapolative zero-order hold, Method IV, an average step.

For the linear program, a comparison was made between results from the five methods and Laplace transform-derived solutions for two linear second-order systems. The natural frequency of each system was unity and the damping constants were .707 (critical damping) and .05 (Q=10). As shown in Chapter III, good results were obtained
for both step and sine inputs. The results from Method IIIb were nearly identical to those of the transform solution for step inputs. Method I gave the best results for sine inputs.

In the non-linear example problem, the five state-space methods and the Hurt-Fowler method were compared to a solution of the system with the state-space simulation using a sample period \( T \) of 0.001 seconds, and the first-order interpolative input approximation. Method IIIa gave the best results for both step and sine inputs. For sampling periods less than 0.2 seconds with sine inputs, there was no significant difference between the Hurt-Fowler method and the state-space Methods I, II, and IV. At \( T = 0.2 \) seconds, the error for the Hurt-Fowler method still increased linearly (vs sample period) but the error for the state-space methods (except Method IIIa) increased with a steeper slope.

The results are described in more detail in Chapter III.

The major contribution of this work is to show that the five input approximation methods can be applied to piecewise-linear models of non-linear systems to yield a simulation scheme that derives the simulation algorithms directly from the state equations of the piecewise-linear system without the detailed root-locus matching analysis required by the Hurt-Fowler method (IBM Report).
CHAPTER II

PRESENTATION OF THE TWO METHODS

2.1 The General Approach

The intent of this chapter is to present the basic Hurt-Fowler approach to the simulation of a non-linear system using Z-transform techniques and the state variable approach upon which the computer simulation program is based. Any attempt at contrasting the two methods will be reserved for Chapter III, where the results from the two simulation approaches will be presented and compared.

2.2 The Hurt-Fowler Approach

Basically, the Z-transform approach treats the system first as a pure linear system and solves for the complete S-domain system transfer function. This transfer function is then transformed to the Z-domain. Then the system is considered to be composed of system blocks. The S-domain transfer function for each system block is found and individually transformed to the Z-domain. Gain constants, $k_i$, are added to one or more system blocks. A complete Z-domain system transfer function is obtained from these individual transfer functions. The denominators of the two complete Z-domain system transfer functions are equated.
yielding like poles and a solution for the gain constants, \( k_i \); the \( k_i \) are chosen to provide a suitable match between the Z- and S-domain root loci.

To illustrate this approach, consider the example of a system with a limiter in the forward loop as in Fig. 2.1. The limiter behaves as a variable gain element of gain \( k \) such that when the system is in the non-limiting mode, \( k = 1 \), and when limiting occurs \( k \) is such that \( k = k_L \). First, disregarding the limiter, the complete system transfer function is

\[
\frac{x_5(s)}{x_1(s)} = \frac{G_1(s)G_2(s)}{1 + G_1(s)G_2(s)H(s)} \quad (2.1)
\]

The Z-transform of equation 2.2 yields

\[
\frac{x_5(z)}{x_1(z)} = \frac{G_1(s)G_2(s)}{1 + G_1(s)G_2(s)H(s)} \quad (2.2)
\]

This transfer function has correct roots and steady state gain for impulse inputs; any other inputs would require an adjustment which would alter the numerator.

Next, the Z-transform of the individual system blocks is taken yielding the system of Fig. 2.2 which is sampler separated to facilitate knowledge of when limiting occurs. Now a supplemental gain constant may be added to each system block.
Fig. 2.1 General Example System
Fig. 2.2 Sampler Separated System
\[ G_1^*(z) = k_1 G_1(z) \]
\[ G_2^*(z) = k_2 G_2(z) \]  

(2.3)

The calculations can now proceed one transfer function at a time to allow a check if \( x_3 \langle k_L \) before \( x_4 \) is calculated. The complete system transfer function for this configuration is

\[ \frac{x_5(z)}{x_1} = \frac{k_1 k_2 G_1(z) G_2(z)}{1 + k_1 k_2 G_1(z) G_2(z) H(z)} \]  

(2.4)

It must be noted that Equations 2.2 and 2.4 are not equivalent. The roots and steady-state gain will agree if the denominators are equated; yielding a solution for \( k_1 \) and \( k_2 \). The different equations derived from this modified transfer function will give accurate solutions for digital simulations of many non-linear systems involving simple non-linearities such as limiter, deadspace, or hysteresis.

2.3 The State-Space Method

The state-space approach that is implemented in the program described in Chapter III, is based on switching from one linear fixed continuous-time system to another when the piecewise linear approximation of the system non-linearity has a change in slope. Given such a system as described in Equations 2.5 and 2.6, one may formulate a solution scheme using a discrete-time approximation.
\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) + E \\
y(t) &= Cx(t) + Du(t)
\end{align*}
\] (2.5) (2.6)

where the variables are:

\[
\begin{align*}
x(t), & \text{ a } k \times 1 \text{ state vector time function} \\
u(t), & \text{ a } n \times 1 \text{ input vector time function} \\
x(t), & \text{ a } m \times 1 \text{ output vector time function} \\
A, & \text{ a } k \times k \text{ constant matrix} \\
B, & \text{ a } k \times n \text{ constant matrix} \\
C, & \text{ a } m \times k \text{ constant matrix} \\
D, & \text{ a } m \times n \text{ constant matrix} \\
E, & \text{ a } k \times 1 \text{ constant vector}
\end{align*}
\]

Solving Equation 2.5 yields a typical solution of a first-order differential equation; initial condition response plus forced response and an added forced response type term due to the E vector.

Consider that E is a vector like the matrix, D, of piecewise-constant inputs. The complete system solution is

\[
x(t) = \Phi(t)x(0+) + \int_0^t \Phi(t-v)Bu(v)dv + \int_0^t \Phi(t-v)Edv
\] (2.7)
where the state transition matrix, $\Phi(t)$, is defined as

$$\Phi(t) = e^{At} = I + At + \frac{A^2 t^2}{2!} + \ldots$$

(2.8)

at discrete times, $t = nT$,

$$x(nT + T) = \Phi(nT + T) x(0^+) + \int_{0}^{T+nT} \Phi(nT+T-v)Bu(v)dv$$

$$+ \int_{0}^{T+nT} \Phi(nT+T-v)vEdv$$

(2.9)

factoring out the $\Phi(T)$ since $\Phi(nT+T-v) = \Phi(T)\Phi(nT-v)$

$$x(nT + T) = \Phi(T) \left[ \Phi(nT) x(0^+) + \int_{0}^{nT} \Phi(nT-v)Bu(v)dv$$

$$+ \int_{0}^{nT} \Phi(nT-v)vEdv \right]$$

$$+ \int_{nT}^{nT+T} \Phi(nT+T-v) \left[ Bu(v) + E \right] dv$$

(2.10)

the bracketed term is $x(nT)$ so

$$x(nT + T) = \Phi(T) x(nT) + \int_{nT}^{nT+T} \Phi(nT+T-v) \left[ Bu(v) + E \right] dv$$

(2.11)
Equation 2.11 is exact along one segment of the given piecewise linear non-linearity provided the integral could be evaluated directly, but in the digital simulation the value must be estimated based on the knowledge of \( u(t) \) only at discrete sample times. There will also be error due to the fact that the \( A \) and \( E \) matrices change between calculation times.

Five input approximations are to be considered here with the constraint that no more than two input values are used in any algorithm. Figure 2.3 illustrates the input approximation concept. The input approximations correspond to a non-realizable first-order hold (Method I), a realizable first-order hold (Method II), a non-realizable zero-order hold (Method IIIa), a realizable zero-order hold (Method IIIb), and a non-realizable averaged step (Method IV). The impulse response of each is indicated in Fig. 2.4.

Consider the following discussion for approximating \( u(t) \) in the interval \( nT < t < nT + T \), by Method I, as an example for all five cases. With the case corresponding to an interpolative first-order hold, knowledge is required of \( u(nT) \), and \( u(nT + T) \) at \( t = (n+1)T \). In the time interval, disregarding the \( E \) term,

\[
\frac{d}{dt} u(t) - \frac{u_{n+1} - u_n}{T} = u(t) - u_{n+1} = \frac{u_{n+1} - u_n}{T} (t - nT) \quad (2.12)
\]

if one defines, \( t = nT + T - a \), \( dt = da \), \( a = nT + T - t \) then

\[
u(t) = u_n + \frac{u_{n+1} - u_n}{T} (T - a)
\]
Fig. 2.3 Input Approximation Concept
Fig. 2.4  Impulse Responses of Input Approximations
if \( J \) is defined as

\[
J = \int_{0}^{nT+T} \Phi(nT+T-v)Bu(v)dv
\]  

(2.14)

let \( t = v \) in the above

\[
J = \int_{0}^{nT+T-a} \Phi(nT+T-a)Bu(nT+T-a)(-da)
\]

\[
= \int_{0}^{T} \Phi(a)Bu(nT+T-a)da
\]  

(2.15)

then substitute the estimated \( u \)

\[
J = \int_{0}^{T} \Phi(a)Bu[nT+1-(u_{n+1}-u_n)]a da
\]  

(2.16)

where

\[
k_1 = \int_{0}^{T} \Phi(a)da = \Lambda^{-1}[\Phi(T)-I]
\]

and

\[
k_2 = \int_{0}^{T} \Phi(a)da = (\Lambda^{-1})^2[I-(I-\Lambda^{T})\Phi(T)]
\]  

(2.17)
With a similar development of the $E$ term, the final solution is:

$$x_{n+1} = \Phi(t)x_n + k_1Bu_{n+1} + k_1\frac{E}{t} - k_2\frac{u_{n+1} - u_n}{t}$$

(2.13)

The equations for calculating the $k_i$ matrices are presented in Table 2.1. Solutions for the other methods, similar in form to Equation 2.18, are given in Table 2.2.

Now, by switching algorithms in the system corresponding to changes in the $A$ matrix for different segments of the non-linearity, one may simulate the non-linear system with the state-space approach.
### Table 2.1 Equations for Calculating System Constant Matrices (Wait 1966)

<table>
<thead>
<tr>
<th>Constant Matrix</th>
<th>Integral Expression</th>
<th>Matrix Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>$\int_0^T \Phi(v)dv$</td>
<td>$\Lambda^{-1}[\Phi(T) - I]$</td>
</tr>
<tr>
<td>$k_2$</td>
<td>$\int_0^T v\Phi(v)dv$</td>
<td>$(\Lambda^{-1})^2[I - (I - \Lambda T)\Phi(T)]$</td>
</tr>
<tr>
<td>$k_3$</td>
<td>$\int_0^{T/2} \Phi(v)dv$</td>
<td>$\Lambda^{-1}[\Phi(T/2) - I]$</td>
</tr>
<tr>
<td>$k_4$</td>
<td>$\int_{T/2}^T \Phi(v)dv$</td>
<td>$\Lambda^{-1}[\Phi(T)\Phi(T/2)]$</td>
</tr>
</tbody>
</table>
### Table 2.2 Equations for Calculation of Simulation Expressions (Wait 1966)

**BASIC SYSTEM EQUATION:**

\[
x(nT+T) = x(n^) + l_1 u(nT-T) + l_0 u(nT) + l_1 u(nT+T) + k_1 E
\]

<table>
<thead>
<tr>
<th>METHOD</th>
<th>(B_1)</th>
<th>(B_2)</th>
<th>(B_3)</th>
</tr>
</thead>
</table>
| I      | 0      | \(k_1\) | \[
\begin{bmatrix}
k_1 - k_2  \\
T
\end{bmatrix}
\] |
| II     | \[
\begin{bmatrix}
k_2 - k_1  \\
T
\end{bmatrix}
\] | \[
\begin{bmatrix}
k_1 - k_2  \\
T
\end{bmatrix}
\] | 0 |
| IIIa   | 0      | 0      | \(k_1\) |
| IIIb   | 0      | \(k_1\) | 0 |
| IV     | 0      | \(k_2\) | \(k_3\) |
CHAPTER III

DISCUSSION OF THE EXAMPLE PROBLEMS AND RESULTS

3.1 The Linear Simulation

The state-space approach discussed in Chapter II was implemented in a Fortran computer program that used all five input approximations. Initially written for a linear problem, the program was used to simulate some second-order systems with both step and sine inputs. For a general second-order system, the overall system transfer function is of the form,

\[ R(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \]  

(3.1)

where \(\omega_n\) is the natural frequency of the system and \(\zeta\) is the damping factor. The cases considered used a natural frequency of unity and the system configured as in Fig. 3.1.

A theoretical solution for the system response to the input, \(u\), may be derived using Laplace transforms.

The constant matrices \(A\) and \(B\), for the state-space approach are derived in Equation 3.2.
Fig. 3.1 Linear Example System
The system matrices are set up from the above equations.

\[
\begin{align*}
\dot{x}_2 &= Sx_1, & \Rightarrow & & \dot{x}_1 = x_2 \\
\dot{x}_2 &= \frac{u - x_1}{S + 2\zeta} & \Rightarrow & & \dot{x}_2 = u - x_1 - 2\zeta x_2
\end{align*}
\]

The system matrices are set up from the above equations.

\[
A = \begin{bmatrix} 0 & 1 \\ -1 & -2\zeta \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

(3.3)

3.2 Results from the Linear Simulation

The simulation was performed for two systems corresponding to values of \(\zeta = .05\) and \(\zeta = .707\). The inputs used were a unit step, and unit amplitude sines of frequencies \(.1, .5, 1.0,\) and \(2.0\) radians/seconds. Sample periods of \(T = .01, .05, .1, .15,\) and \(.2\) seconds were used. The comparison shown with the error as a difference between a Laplace transform solution and the state-space simulation in Fig. 3.2 is for the step input, a plot of peak error vs sample period for the five input approximations and both values of \(\zeta\).

From Fig. 3.2, it is apparent that the small value of damping constant \((\zeta = .05)\) gave less accurate results. This was to be expected since the frequency response for the narrow band filter has a rapid amplitude change versus frequency near the natural frequency. Even though the step
Fig. 3.2 Plot of Peak Error vs Sample Period for Linear Simulation, Step Input of 1.0.
input corresponds to zero input frequency, the close proximity of the system open loop poles to the $j\omega$ axis dictates a more distorted transient response. The plot was linear for all methods indicating a predictable error. The system using the Method IIIb input approximation (extrapolative zero-order hold) gave excellent results for all cases. Error in the method was independent of sample period and, considering computer accuracy, negligible. The systems using Methods I, II, and IV (interpolative first-order hold, extrapolative first-order hold, and averaged step) gave identical peak errors as the equations in Table 2.2 predict for a step input.

Figures 3.3-3.7 show the comparison for sine inputs; representative plots of the log of the peak error vs $\log \omega_n/\omega_{in}$ for the two values of $\zeta$ and four different sample periods.

As for the step input results, the system with a small damping constant gave less accurate results with a predictable increase in error for the case where $\omega_n/\omega_{in}$ is unity. Results for the systems using Method IIIa (interpolative zero-order hold) and Method IIIb were not as good as for the other methods. The peak errors were identical even though the results from Method IIIa led the theoretical response and those from Method IIIb lagged. This is predictable since Method IIIa has an inherent half-period advance and Method IIIb a half-period delay. The Method I
Fig. 3.3 Plot of Log Peak Error vs Log $\omega_n/\omega_{in}$ for Linear Simulation, Method I, Sine Inputs, $\zeta = 0.707$
Fig. 3.4 Plot of Log Peak Error vs Log $\omega_n/\omega_{in}$ for Linear Simulation, Method I, Sine Inputs, $\zeta = .05$
Fig. 3.5 Plot of Log Peak Error vs Log $\frac{\omega_n}{\omega_{in}}$ for Linear Simulation, "method II, Sine Inputs, $\zeta = .707$
FIG. 3.6 Plot of log peak error vs log ω/ω_n in for linear simulation, Method III, Sino Inputs, Y" = .707
Fig. 3.7 Plot of Log Peak Error vs Log $\omega_n/\omega_{in}$ for Linear Simulation, Method IV, $\zeta = .05$
input approximation gave the best results followed by Method IV and Method II. The error from Method II can be, in part, attributable to the inherent one period delay for the method. The greater the weighting given the knowledge of the future input value, the less was the error.

3.3 The State-Space Solution for the Non-Linear Problem

The state-space program was adapted to a non-linear second-order system with a limiter in the forward path. The specific example chosen was that used by Hurt and Fowler for their initial discussions as shown in Fig. 3.8 (Hurt 1964, IBM Report). In the state-space approach, the contrast between the limiting and non-limiting modes corresponds to a change in the A matrix and the E vector. If the system is not limiting, the linear case, then,

\[
A = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 5 \end{bmatrix}, \quad E = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\] (3.4)

\[
A = \begin{bmatrix} 0 & 0 \\ -5 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 5 \end{bmatrix}, \quad E = \begin{bmatrix} 2_1 \\ 0 \end{bmatrix}
\] (3.5)

The A matrix is singular and the \( E_1 \) element of E is \( ^\dagger 1 \) depending on the limit level. With the A matrix singular, the scheme for finding \( \Phi \) will not work since \( A^{-1} \) is not defined. This problem can be avoided by giving the limit levels some small slope, e. Then, in Equation 3.5, \( A_{12} = e \), and \( E_1 = ^\dagger (1-e) \) for the limiting case.

The value of \( x_2 \) is checked following each calcu-
Fig. 3.3 Non-Linear Example System
lation, and if it is out of the linear region, the limiting algorithms are substituted for the next calculation; switching from one algorithm set to another as the solution for \( x_2 \) dictates.

### 3.4 The Hurt-Fowler Solution for the Non-Linear Problem

The Hurt-Fowler algorithms for the same system and a general value of \( T \) are derived in the following (Hurt 1964). If the system is in the linear region, \( x_1(s) = \frac{5}{s^2+2s+5} \) if \( u(s) = \frac{1}{s} \), a unit step,

\[
x_1(s) = \frac{5}{s(s^2+2s+5)} \quad (3.6)
\]

Taking the \( Z \)-transform of Equation 3.7 yields

\[
x_1(z) = \frac{z}{z-1} - \frac{2-e^{-T\cos(2T)}z-.5e^{-T\sin(2T)}z}{z^2-2e^{-T\cos(2T)}z+e^{-2T}} \quad (3.7)
\]

Divide by \( u(z) \)

\[
x_1(z) = \frac{z[1+.5e^{-T\sin(2T)}-e^{-T\cos(2T)}]}{z^2-2e^{-T\cos(2T)}z+e^{-2T}} \quad (3.8)
\]

\[
+ \frac{e^{-2T}-e^{-T\cos(2T)}+.5e^{-T\sin(2T)}}{z^2-2e^{-T\cos(2T)}z+e^{-2T}} \quad (3.9)
\]

Now considering the system as sampler separated, solve for the transfer function of each system block.
where a gain constant, $k$, has been introduced. The overall transfer function, after a feedback loop delay is introduced so to not require output before input is,

$$\frac{\frac{5}{s+2}}{z-e^{-2T}} \rightarrow \frac{kz}{z-e^{-2T}} \quad \frac{1}{s} \rightarrow \frac{Tz}{z-1}$$

(3.10)

In order to solve for $k_1$, equate coefficients of like powers of $z$ in the denominators of Equations 3.9 and 3.11. Solving about each transfer function for the difference equations yields

$$\frac{x_1(z)}{u} = \frac{kTz^2}{z^2 + (Tk-1-e^{-2T})z + e^{-2T}}$$

(3.11)

A comparison must be found for both systems that approximates the continuous system. After some study, it was found that both systems compared well for a sample period of .001 second. With the value of $T$ so small, theory predicts that the continuous system response will be very similar. For this reason, the state-space results from Method I for $T = .001$ seconds have been used as a standard for comparison.
To more easily see the correspondence of the system algorithms for the state-space and Hurt-Fowler methods, the state-space algorithms for Method IIIa and the difference equation set of 3.12 are shown with \( T = .1 \) seconds and \( e = .0001 \) in Equations 3.13 to 3.16.

For the Hurt-Fowler system in the linear mode,

\[
\begin{align*}
    x_1(nT) &= .1x_2(nT) + x_1(nT-T) \\
    x_2(nT) &= -.453x_1(nT-T) + .8187x_2(nT-T) + .453u(nT) 
\end{align*}
\]

(3.13)

in the limiting mode,

\[
\begin{align*}
    x_1(nT) &= x_2 + x_1(nT-T) \\
    x_2(nT) &= -.453x_1(nT-T) + .8187x_2(nT-T) + .453u(nT) 
\end{align*}
\]

(3.14)

For the state-space system in the linear mode,

\[
\begin{align*}
    x_1(nT) &= .9767x_1(nT-T) + .0899x_2(nT-T) + .0233u(nT) \\
    x_2(nT) &= -.4494x_1(nT-T) + .7962x_2(nT-T) + .4494u(nT) 
\end{align*}
\]

(3.15)

in the limiting mode,

\[
\begin{align*}
    x_1(nT) &= -.0362 + x_1(nT-T) \\
    x_2(nT) &= -.453x_1(nT-T) + .8187x_2(nT-T) + .453u(nT) 
\end{align*}
\]

(3.16)

An examination of the corresponding terms in the equations shows a great similarity in the \( x_2(nT) \) terms.
for both systems but the $x_1(nT)$ terms are quite dissimilar. In the linear mode, the Hurt-Fowler scheme uses the just calculated value for $x_2$, $x_2(nT)$, while the state-space method uses the past value, $x_2(nT-T)$. For the linear case, the state-space method uses a slight weighting of the input value in the calculation of $x_1(nT)$ while it does not appear in the Hurt-Fowler algorithm. In the limit mode, the state-space algorithm weights the limit level about one-third as much as does the Hurt-Fowler approach. It is quite possible that limiting would not occur at the same time in both systems.

3.5 Non-Linear Simulation Results

Each system (five state-space methods and Hurt-Fowler method) was given a unit step input with results as in Fig. 3.9. The plot shows peak error vs sample period for the six methods.

With the exception of the results for $T = .2$ seconds, state-space Methods I, II, IV and the Hurt-Fowler method had similar results. Method IIIb had predictably poorer results since only one current input value was known at a time. Method IIIa gave considerably better results than the others. This might be expected for step inputs because the system has knowledge of the next value of the input prior to its application to the system, and it assumes it to be a step. The improvement in this method, over the linear case, could be attributed to the fact that
Fig. 3.9 Plot of Peak Error vs Sample Period for Non-Linear Simulation, Step Input of 1.0
the non-linearity effectively attenuates the response making it more predictable by the interpolative hold tending to negate any lag over the system response.

Each system was given a unit amplitude sine input of frequency two radians/second. Figure 3.10 shows a plot of peak error vs sample period for the six methods. Figure 3.11 shows a time plot of the error for the solutions of Methods I, IIIa, and the Hurt-Fowler method for T = .1 seconds.

Rather unpredictably, the best results were achieved for Method IIIa. A possible reason for the results is that the Method IIIa system has an inherent lead over the continuous system response and the system of Methods I, II, IV, and the Hurt-Fowler method had a slight lag. This was apparent from the results since the greatest errors occurred when the change in slope was most rapid (zero crossings). The non-linearity produces some flattening at the peaks compared to the linear system response. This is more like a transient step response which may account for the good results of Method IIIa. As for the step input, Methods I, II, III, and the Hurt-Fowler method compare well for T = .15. The Hurt-Fowler peak error was linear for all the sample periods considered and hence more predictable.
Fig. 3.10 Plot of Peak Error vs Sample Period for Non-Linear Simulation, Sine Input of $\sin 2t$
Fig. 3.11 Plot of Error vs Time for Non-Linear Simulation, Sine Input of sin 2t (T=1 seconds)
CHAPTER IV

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

4.1 Problem Considerations

The results discussed in the previous chapters indicate that the state-space approach may be as accurate, if not more accurate, than the Hurt-Fowler approach for the non-linear example used. It must be pointed out that the non-linear example problem considered was one that had achieved good results for the Hurt-Fowler algorithms (IBM Report). The problem was not one that was designed to illustrate the accuracy of the state-space methods as some other problems might well have accomplished.

Simulation of this class of problems in real-time is still the desired goal of the system designer. The state-space approach provides a method that may well achieve this goal. The simulation program is designed for straightforward input of data derived from the equations that describe the system in the state-space. There is no need for special system analysis like the root-locus analysis used in the Hurt-Fowler approach. The availability, in the state-space method, of intermediate system states as well as the system output, may be quite desirable in system design for determining performance criteria and other pertinent characteristics.
The largest difficulty in the use of the basic computer program arises in the case of the particular example studied here. A limiter type of non-linearity causes the system to exhibit several distinct states. When on a limiting level, the system loop is effectively broken with the output the integral of a step input. The slope of the non-limiting portion acts as a gain element giving the system a linear type of response. Most other linear approximations of non-linearities are extensions of the change in gain concept. These other classes of curves may be easily implemented with the basic program intact.

4.2 Improvements in the Simulation Method

Any difficulty that may arise when the A matrix is singular, as in the example problem, can be avoided by using the Taylor series expansion for the state transition matrix. In the present program, the state transition matrix is evaluated by determining the system characteristic equation from the A matrix which yields the eigenvalues for the system. With the eigenvalues, Sylvester’s Expansion Theorem is used to find \( \Phi(t) \) (Schultz and Melsa, 1967). The matrix multiplication for the evaluation of the system states must involve the inverse of the A matrix as in Table 2.1. Using the Taylor series expansion, it has been shown that in all simulation expressions, there is no inverse term (Kuo and Kaiser 1966). While this adds the problem of determining the number of series terms to include, desired accuracy will
straightforwardly determine the number of terms. The resulting equations for the constant $k_i$ matrices are,

$$k_1 = \frac{I}{T} + \frac{A^2}{2T} + \frac{A^3}{3T} + \ldots$$  \hspace{1cm} (4.1)

$$k_2 = T^2 \left[ 1 - \frac{1}{2T} \right] \frac{I}{A} + A^T \left[ \frac{1}{2T} - \frac{1}{3T} \right] + A^2 \left[ \frac{1}{3T} - \frac{1}{4T} \right] + \ldots$$  \hspace{1cm} (4.2)

$$k_3 = \frac{T}{2} \frac{I}{A} + A^2 \left[ \frac{T}{2} \right] \frac{1}{2T} + A^2 \left[ \frac{T}{2} \right] \frac{1}{3T} + \ldots$$  \hspace{1cm} (4.3)

$$k_4 = \frac{T}{2} \frac{I}{A} + A \left[ \frac{2T}{3} \right] \frac{1}{2T} + A^2 \left[ \frac{7T}{8} \right] \frac{1}{3T} + \ldots$$  \hspace{1cm} (4.4)

With this change, the simulation program may be generalized for any piecewise-linear approximation of a system non-linearity.

4.3 Other Input Approximations

The input approximations for the cases under consideration in the program include only realizable and non-realizable zero- and first-order holds. It would seem that for some classes of continuous inputs, a higher order hold would provide a better approximation to the true character of the time varying input signal. Equation 4.5 (Kuo 1963) is the general power series expansion of some time-varying signal $e(t)$, in the interval between sampling instants $nT$ and $(n+1)T$. 
\[ e_n(t) = e(nT) + e'(nT)(t-nT) + e''(nT)(t-nT)^2 + \ldots \]  
\hfill (4.5)

where the first and second derivatives are estimated by

\[ e'(nT) = \frac{1}{T}(e(nT) - e(n-1)T) \]

\[ e''(nT) = \frac{1}{T^2}(e'(nT) - e''(n-1)T) \]  
\hfill (4.6)

The input approximation would correspond to that of a realizable second-order hold if the first three terms of the series are included in the computation. The resulting equation for \( e(t) \) in the interval \( nT \leq t \leq (n+1)T \) is

\[ e(t) - e(nT) = \frac{1}{T} \left[ e(nT) - e(n-1)T \right] (t-nT) \]

\[ + \frac{1}{T^2} \left[ e(nT) - 2e(n-1)T + e(n-2)T \right] (t-nT)^2 \]  
\hfill (4.7)

For simulation this would require knowledge of the input at three time instants. The realizable second-order hold would be an extrapolative approximation requiring knowledge of \( e(nT) \), \( e(n-1)T \), and \( e(n-2)T \). Two non-realizable situations could be created: an extrapolative-interpolative approximation requiring \( e(n+1)T \), \( e(nT) \), and \( e(n-1)T \), and an interpolative approximation requiring \( e(n+2)T \), \( e(n+1)T \), \( e(nT) \). A system could use a predictive approximation of the current input trends based on past knowledge to give values for \( e(n+1)T \) and \( e(n+2)T \), but the validity of the value for
e(n+2)T would be questionable at best. However, the use of the other two approximations for inputs would provide useful techniques for many continuous systems.

4.4 System Errors

Errors at the points where the system dictates a change from one piecewise-linear segment to another on the system non-linearity have not been considered in the previous study. Clearly, there will be errors, since the system state will, in general, cross a switching surface somewhere between calculation times. One possible approach might be to recalculate the system state value when a change of segment is detected with the sample period halved until no switching is indicated, or a minimum step size is used. The difficulty that might arise here, is that excessive computation time might be taken at segment changes.

Another possible approach would be to use a weighted average of state values calculated by the algorithms of adjacent regions, using a vector addition in the state-space defined by the states preceding and following the non-linearity in the system. As an example, consider the following; define the value of the state vector preceding the indication of crossing a switching surface to be $\mathbf{x}_L$, when the calculated state variables cross a switching surface we obtain the value $\mathbf{x}_0$. Next, the system state is recalculated using the algorithms which hold on the other side of the
switching surface, and call the resulting state $x^o_n$. It is possible that $x^o_n$ will not predict a switching even though $x^o_0$ did. If this is the case, one strategy would be to use the value for $x^o_0$ and proceed with the old algorithm set. If $x^o_n$ does also indicate a switching, then use a weighted average of the two state values. Just how this weighting should be done needs to be examined; it probably could be done based upon the relative distance of $x^o_0$ and $x^o_n$ from the switching surface. This scheme would not create the need for excessive recalculation at the crossing of a switching surface yielding a more real-time oriented system of simulation.

Errors in the simulation due to round-off, appear to be another important consideration for program improvement. Differences in results obtained on the two computer systems used for the program development, were noticeable in the fifth and sixth decimal places. The computer word length and floating point calculation methods should be considered when comparing results and in the desired overall system accuracy and speed.

4.5 Conclusion

Further generalization of the approach used here should also include other locations of the non-linearity in the system and provisions for higher order systems.

There are many fields using simulation where an
improved real-time all digital simulation technique would prove most beneficial. It is hoped that the results and suggestions presented here, will be means toward achieving that end.
APPENDIX I

SIMULATION PROGRAM CONCEPTS

A1.1 Basic Program Approach

The basic program was written for a linear system and then modified for use with the non-linear example problem. Matrices that describe the system are read as input data as well as desired sample sizes and inputs. After the input data is read, it is output for solution clarity and as a data check. All constant system matrices for a particular sample size are calculated for reference by the iteration portion of the program. There are five similar iterating loops that compute the simulated system states. These loops are composed of calls to various subroutines to perform matrix intermediate results.

More specifically, the basic program flow is indicated in the flow chart of Fig. A1.1. The basic system describing matrices, $A$ and $B$, are read in as data, along with any number of desired sample sizes $T_i$, and any inputs to be used with the sample sizes. The methods to be used as well as output formatting information are also read as data. The data formats are indicated in Appendix II. The system constant matrices $k_i$, and $A^{-1}$ are computed for reference by the iterating loops. The computation is done using subroutines outside of the main program. The five iterating
loops are similar in that they all obtain the system input computed for all required samples regardless of realizability, perform matrix multiplication using subroutines and obtain the eventual result as the system states. There are provisions for comparison routines and variable output formats in the loop.

A1.2 Non-Linear Program Modifications

The addition of the limiter in the system required some program modifications. The \( E \) matrix was included in the computation scheme and the product of \( k \) and \( E \) was added to each system equation. To allow for checking of the state prior to the limiter, the iteration loop was modified. If a limiting level was exceeded, a flag was set and limit algorithms were used for the next iteration and on for succeeding state changes. The limit algorithms were found like the other basic constant system matrices but using a modified \( A \) matrix to reflect the limiting condition. A slight slope was given to the limit levels to retain the non-singular character of the \( A \) matrix. An alternative to this method is discussed in Chapter IV.

The other aspects of the linear program are retained for this example problem. A basic flow chart is included in Fig. A1.2.

A1.3 Provisions for Problem Extension

The system dimensions may be increased as computer
A1.2 Basic Non-Linear Program Flow
size allows for the study of more complex problems. Calculation of the y vector was not done as a part of the program but all pertinent variables are available for its insertion into the iteration loop. As discussed in Chapter IV, future more general piecewise representations of non-linearities can be incorporated into the problem with relative ease once the basic system matrices are found.

Each user will undoubtedly have some modifications to make to the system. The program was designed with future uses in mind and those changes should be able to be made with relative ease.
A2.1 General Approach

The basic configuration is similar for both the linear and non-linear programs. The iteration loop for the non-linear program checks for changes in the state of the limiter. A third-order system is provided for in the linear program. A change in the program dimension statements is all that is necessary to increase the system order. The non-linear program is set up for problems discussed in Chapter III. Any desired routine for comparison to the results from the programs may be inserted after the iteration routines. The system is set up to generate steps, ramps, or sines as input variables.

A2.2 Linear Problem Formats

The formats for the data cards are indicated in the following list. More than one card of the same format may be required or can, for the indicated types, be optional with the user. The variables are defined in the order that they appear on the card.

Type I: FORMAT (I5,5X,I2,8X,I2,8X,I1)

NINC = number of program increments
NA - dimensions of A matrix
NB - dimension of B matrix
(one not specified by NA)
IT - number of sample sizes to be used
    with this system for each input

Type II: FORMAT (F10.0)
      RUCK - frequency of output printout-
          RUCK increments/printing

Type III: FORMAT (8F10.0)
        TR(I), I=1, IT - sample sizes, number
            specified by IT

Type IV: FORMAT (8F10.0)
       AM(L,I), I=1, NA - rows of A matrix,
           NA like formatted cards required

Type V: FORMAT (8F10.0)
      BM(L,I), I=1, NB - rows of B matrix,
          NA like formatted cards required

Type VI: FORMAT (5I1)
       M1,M2,M3A,M3B,N4 - method of input
           approximation flags, a 1 will execute
               the method, any combination is possible

Type VII: FORMAT (I1,9X,2F10.0)
        NUT - input type flag, 1 for step, 2
            for ramp, 3 for sine
        FAR - amplitude for step and sine inputs,
            slope for ramp input
CAR = frequency for sine input

One card for each input, and number of like cards followed by a blank card

A2.3 Non-Linear Problem Formats

The same discussion of the formats for the linear program may be applied to the non-linear program. The formats for the data cards are indicated in the following list.

Type I:  FORMAT (I5,5X,I2,3X,I2,8X,11)

NINC = number of program increments

NA = dimensions of A matrix

NB = dimension of B matrix

(one not specified by NA)

IT = number of sample sizes to be used with this system for each input

Type II:  FORMAT (4F10.0)

WK = gain of first-order transfer function block before limiter

WT = time constant of transfer function block before limiter

EPSIL = small slope given to limiter levels to keep A non-singular

SA = slope of linear portion of limiter
Type III: \[ \text{FORMAT (F10.0)} \]

\text{RUCK - frequency of output printout-}

\text{RUCK increments/printing}

Type IV: \[ \text{FORMAT (8F10.0)} \]

\text{TR(I), I=1, IT - sample sizes, number specified by IT}

Type V: \[ \text{FORMAT (5II)} \]

\text{MET1, MET2, MET3A, MET3B, MET4 - method of input approximation flags, a 1 will execute the method, any combination is possible}

Type VI: \[ \text{FORMAT (II, 9X, 2F10.0)} \]

\text{NUT - input type flag, 1 for step, 2 for ramp, 3 for sine}

\text{FAR - amplitude for step and sine inputs, slope for ramp input}

\text{CAR - frequency for sine input}

One card for each input, any number of like cards followed by a blank card.
LIST OF REFERENCES


