RESEARCH REPORT

on

IDENTIFICATION OF LINEAR SYSTEMS

Prepared by

Thomas S. Englar

of
The Mathematical Sciences Group, Inc.
7100 Baltimore Avenue
College Park, Maryland

Under Contract NAS 12-583

for

OCTA
NASA/Electronics Research Center
Cambridge, Massachusetts

Preface

This report has been written in partial fulfillment of Contract NAS 12-583 carried out by The Mathematical Sciences Group with the support of OCTA at NASA's Electronics Research Center.

The goal of work performed under this contract is the production of a digital computer program capable of identifying the dynamic characteristics of a human operator from knowledge of input-output data.

The component programs have been written and are documented herein. A certain amount of experimentation has been done with self-generated data corrupted by additive noise and the results of this experimentation is also reported here.

We wish to take this opportunity to thank Dr. Richard Shirley for the assistance which he has rendered by his interest and suggestions.

TABLE OF CONTENTS

Introduction

This report describes how the identification problem has been approached in this work. Many of the details have been reported previously in Program Descriptions delivered to ERC. Several of these are included as report appendices to provide complete information about all aspects of the analysis and computation. The body report concerns itself with the broader aspects of the system, referring to the appendices for deeper study.

Chapter I describes the context of linear systems analysis in which the problem is formulated, making specific our assumptions and the conditions the identified system must satisfy.

Chapter II describes the mathematics involved in the two basic subdivisions of the system-obtaining the impulse response by projection onto a subspace, and obtaining a canonical realization from the impulse response by application of the B. L. Ho algorithm.

Chapter III describes how these methods are mechanized as computational techniques.

Chapter IV describes our preliminary results in system identification.

Chapter I

Background and Problem Statement

The ultimate goal of this work is the identification of the human operator in the sense that we wish to obtain a linear constant dynamical system which best approximates the human input-output behavior in a particular job. In order to discuss the problem abstractly, we will assume that the system to be identified actually is a linear stationary dynamical system. By dynamical system we mean here a completely controllable, completely observable, finite dinensional system usually appearing as a set of differential equations relating the state $x(t)$ and the control $u(t)$ by

$$\dot{x} = Ax + Bu$$

and an algebraic relation relating the state and the output or vector of observables $y(t)$ by

$$y(t) = Cx(t)$$

Thus, our dynamical system can be represented by the triple of constant matrices $[C, A, B]$. As is well known, this representation is not unique since any similarity transformation S of A gives a new representation

$$[CS^{-1}, SAS^{-1}, SB].$$

We try to avoid this ambiguity by expressing the system in some canonical form.

This set of equations has the solution

$$y(t) = Ce^{tA}x(0) + \int_0^t Ce^{(t-\tau)A}Bu(\tau)d\tau.$$

Also characterizing this dynamical system is its impulse response

$$H(t) = Ce^{tA}B,$$

or its transfer function

$$H(s) = C(sI-A)^{-1}B = \mathcal{L}\,H(t) = \left[\frac{p_{ij}(s)}{q_{ij}(s)}\right].$$

The input-output relations sometimes appear in the form of an integral equation

$$y(t) = \int_0^t H(t-\tau)u(\tau)d\tau.$$

In all practical cases we define an additional variable $z(t)$ which is the state-dependent output $y(t)$ corrupted by some "noise" $v(t)$:

$$z(t) = y(t) + v(t).$$

These remarks serve to delineate the context in which our problem is stated.

<u>Problem 1</u>: Given $\{z(t), u(t)\}$, defined on the interval $[0, T]$, obtain a minimal realization $[\hat{C}, \hat{A}, \hat{B}]$ such that

$$\int_0^T \left\| z(t) - \int_0^t \hat{C}e^{(t-\tau)\hat{A}}\hat{B}u(\tau)d\tau \right\|^2 dt$$

is minimal.

This problem does not consider the effect upon $z(t)$ of initial conditions on the state at time zero. Therefore, it will provide a good operating procedure only if $x(0)$ is zero. Unfortunately, it is impossible to place a human operator, such as a pilot, in zero-state condition. Furthermore, such a technique would limit applications of the program, since there are great advantages to examining some part of a long data run rather than only its initial phase. For instance, it allows the system, human or machine, to have a break-in or warmup period before taking data for analysis. Furthermore, one could wish to examine sequential data blocks in a long run to determine possible low frequency nonstationarity.

The most straightforward assumption which will enable such operations is:

Assumption: The system to be identified is asymptotically stable.

In addition we will proceed on the basis that the eigenvalues, initial conditions, and inputs are such that there exists a time $t_1 < T$ such that for computation purposes

$$y(t) = \int_0^t H(t-\tau)u(\tau)d\tau \quad \text{for} \quad t_1 \leq t \leq T.$$

We now state the problem to be solved.

Problem 2: Given functions $\{z(t), u(t)\}$ defined on the interval $[0, T]$, obtain a minimal realization $[\hat{C}, \hat{A}, \hat{B}]$ such that

$$\sigma^2 = \int_{t_1}^T ||z(t) - \int_0^t \hat{C}e^{\tau\hat{A}}\hat{B}u(t-\tau)d\tau||^2 dt$$

is minimal.

The norm $||\cdot||$ used here is the usual Euclidean norm in finite dimensional space.

In what follows the mathematical methods, their numerical implementation, and recent numerical experiments will be described and analysed in some detail.

Chapter II

Mathematical Methodology

Involved in Problem 2 are two distinct subproblems, solutions to which we have programmed separately. The first is the definition of an approximating kernel $\hat{H}(t)$ such that σ^2 of Problem 2, is minimal.

The second is the definition of a system $[\hat{C}, \hat{A}, \hat{B}]$ such that, approximately,

$$\hat{H}(t) = \hat{C}e^{t\hat{A}}\hat{B} .$$

1. Obtaining $\hat{H}(t)$:

Without loss of generality, we restrict our discussion to scalar kernel functions $\hat{h}(t)$.

The method used is basically a Rayleigh-Ritz procedure. However, important modifications in both the theory and the numerical techniques are implied by the fact that we are performing what, from an engineering viewpoint, might be called a second-level approximation problem. What is really desired is an approximation $\hat{h}(t)$ which minimizes

$$\varepsilon^2 = \int_0^\infty \|\hat{h}(t) - h(t)\|^2 dt .$$

But our problem constraints are such that we must be satisfied with solving Problem 2.

Problem 2 is mathematically equivalent (see Appendix A, sec. VII-4), under the restrictions about stability which we have hypothesized, to minimizing

$$\int_0^{t_1}\int_0^{t_1} (\hat{h}(\tau) - h(\tau))Q(\tau,s)(\hat{h}(s) - h(s))dsd\tau = ||\hat{h} - h||_Q^2 .$$

Here

$$Q(\tau,s) = \int_{t_1}^{T} u(t-\tau)u(t-s)dt$$

is a nonnegative definite symmetric kernel which is singular if $u(t)$ is a band-limited function.

If nothing else, the digital implementation which we use would serve to band-limit $u(t)$ by the sampling theorem. Fortunately the singularity of Q does not seem to be a serious practical problem. The nonsingularity of Q is a measure of the amount of information about $h(t)$ which is present in $z(t)$. This is independent of noise, of course, and our experiments with noise-free data indicate that our present signal

$$u(t) = \frac{1}{4} + \sum_{k=1}^{10} c_k \sin k\omega t \ , \quad |c_k| = 1 \tag{2.1}$$

is adequate for our purposes.

Returning therefore, to our Rayleigh-Ritz procedure for Problem 2, we assume that a set of functions $\{\ell_i(t)\}_o^K$ are available such that for each $h(t)$ of interest, there exists a linear combination

$$\hat{h}(t) = \sum_{k=o}^{K} \beta_k \ell_k(t)$$

with $||\hat{h}(t) - h(t)||$

satisfactorily small.

This representation of $\hat{h}(t)$ being decided upon, σ^2 is minimized with respect to the vector

$$\beta = \begin{bmatrix} \beta_o \\ \beta_1 \\ \cdot \\ \cdot \\ \cdot \\ \beta_K \end{bmatrix} .$$

That is we compute

$$\hat{z}(t) = \int_0^t h(\tau)u(t-\tau)d\tau = \sum_{k=o}^{K} \beta_k \int_0^t \ell_k(\tau)u(t-\tau)d\tau$$

and minimize

$$\int_{t_1}^{T} \|\hat{z}(t) - z(t)\|^2 dt \tag{2.2}$$

by manipulating β.

Defining a new set $\{f_i(t)\}_o^K$ of functions by

$$f_i(t) = \int_0^t \ell_i(\tau)u(t-\tau)d\tau$$

we find that the equation to be solved in the least square sense is

$$\sum_{k=o}^{K} \beta_k f_k(t) = z(t) \qquad t_1 \leq t \leq T .$$

Under very general conditions on $\{\ell_i\}_o^K$ and $u(\cdot)$ (Appendix A, section VII-3) we can show that $\{f_k(t)\}_o^K$ is a linearly independent set and there exists, therefore, a unique minimum of (2.2).

The result on linear independence cannot be stated briefly, but for

$$u(t) = \sum_{k=1}^{M} a_k \sin k\omega t, \quad \prod_{k=1}^{M} a_k \neq 0$$

then, usually, the set $\{f_i\}_o^K$ will be linearly independent if

$$K + 1 \leq 2M$$

and $\{\ell_i\}_o^K$ is linearly independent.

A more detailed discussion of the effects of K and M is contained in Appendix A section VII-4. However, $M = 10$ appears to be adequate for our purposes if we can indeed obtain a satisfactory approximation with the given $K + 1$ functions $\{\ell_i(t)\}_o^K$.

This is rather a serious stumbling block at present for reasons associated with the numerical computations. These are set forth in Chapter III.

A complete description of the program used to obtain $\hat{h}(t)$ is contained in Appendix A.

2. Obtaining $[\hat{C}, \hat{A}, \hat{B}]$ from $\hat{h}(t)$:

The B. L. Ho method is used to obtain the system representation from the impulse response. This is done either directly from the Taylor coefficients of $\hat{h}(t)$ or indirectly by using the values of $\hat{h}$ at fixed time intervals to generate a system representation $[\hat{C}, \hat{\Phi}, \hat{\Gamma}]$ of the discrete system and then taking the logarithm of that system to obtain the continuous system $[\hat{C}, \hat{A}, \hat{B}]$. The present description will be confined to the single-input, single-output case because this is where our experience lies. In section 1. of this chapter, this restriction was made without loss of generality. Here it is a serious restriction, and programs are being modified to handle the multi-input, multi-output situation.

A complete proof of the single-input, single-output Ho procedure may be found in Appendix B, section VII-1. The method will be outlined only briefly here.

A sequence $\{a_k\}$ is said to be of rank less than or equal to n if it can be generated from n-vectors c and b and an n^{th} order matrix A by the rule

$$a_k = cA^{k-1}b \ .$$

The B. L. Ho procedure takes a sequence (of finite rank), determines its rank, and exhibits the matrices $[c, A, b]$.

For $h(t) = ce^{tA}b$, the sequence

$$h_k = h((k-1)\delta) = ce^{(k-1)\delta A}b = c(e^{\delta A})^{k-1}b$$

satisfies the above condition and the Ho procedure will therefore give a discrete system similar to $[c, e^{\delta A}, b]$. This can then be transformed to a continuous system similar to $[c, A, b]$.

On the other hand, if we expand $h(t)$ in its Taylor Series

$$h(t) = \sum_{k=o}^{\infty} \frac{a_k}{k!} t^k$$

then $a_k = cA^{k-1}b$

forms a sequence which satisfies the given condition and leads directly to a system similar to

$$[c, A, b] \ .$$

Generation of a_k from $\hat{h}(t)$ involves high order differentiation which is well known to be a poorly-conditioned operation on experimental data. Both procedures are available; however, we have obtained better results with the sampled impulse response than with the Taylor Series even for low order systems and expect this to hold even more strongly for higher-order system.

The program implementing the B. L. Ho procedure (MICARE) is described in appendix B, the system logarithm program (CPC) is described in Appendix C. These two virtually complete the procedure; we have omitted the very simple routines describing how the sampled impulse response is obtained from the coefficients $\{\beta_i\}_o^K$. Input to MICARE is a sequence as described above.

Chapter III

Computation

The mathematics described in Chapter II is very straightforward and the implementation should be very simple. This turns out to be untrue because of computational difficulties, especially in the presence of noise.

We first consider the noise-free case and examine the first problem: What should be the set of basis function $\{\ell_i(t)\}_o^K$?

1. The Approximating Set

By our fundamental assumption, all h(t) under consideration will decay to zero. It was felt therefore that the functions of the set $\{\ell_i\}$ should also satisfy this condition. This ruled out fourier approximation and the usual polynomial approximations.

Several sets of appropriate functions appear in the engineering literature (see W. H. Kautz, Transient Synthesis in the Time Domain, IRE Transactions-Circuit Theory, September 1954, pp 29-39). Of these, the laguerre functions were chosen for two major reasons. They can be generated economically by using their recursion relations and the analysis of their approximations properties has been very clearly performed (J. W. Head, Approximations to Transients by Means of Laguerre Series, Proc. Cambridge Philosophical Society, October 1956, pp 640-651).

A few facts about the laguerre functions will make the subsequent discussion more readily understood.

For arbitrary (real positive) p, the first few functions are:

$$\ell_o(t) = \sqrt{2p}\; e^{-pt}$$

$$\ell_1(t) = \sqrt{2p}\; e^{-pt}(2pt - 1)$$

$$\ell_2(t) = \sqrt{2p}\; e^{-pt}(2p^2t^2 - 4pt + 1)$$

$$\ell_3(t) = \sqrt{2p}\; e^{-pt}(\tfrac{4}{3}\, p^3t^3 - 6p^2t^2 + 6pt - 1)$$

$$\ell_4(t) = \sqrt{2p}\; e^{-pt}(\tfrac{2}{3}\, p^4t^4 - \tfrac{16}{3}\, p^3t^3 + 12p^2t^2 - 8p + 1)$$

$$\ell_5(t) = \sqrt{2p}\; e^{-pt}(\tfrac{4}{15}\, p^5t^5 - \tfrac{10}{3}\, p^4t^4 + \tfrac{40}{3}\, p^3t^3 - 20p^2t^2 + 10pt - 1)\ .$$

The initial value is $\pm\sqrt{2p}$, $\ell_k(t)$ has k relative extrema of decreasing magnitude, and $\ell_k(t)$ for $p = 1$ is computationally zero at $2k + 7$. The most serious oscillations of ℓ_k occur near zero, where $\ell_k(t)$ behaves, to first order, like $e^{-(2k+1)pt}$. Table I shows the percentage error in Simpson's Rule integration of $e^{-\lambda t}$ for various numbers of integration intervals per time constant. (To avoid confusion here, by integration interval, we mean the interval between function evaluations, which is half of what is usually called the integration interval in Simpson's Rule.)

Assuming that we wish to integrate with a relative error of about 10^{-4}, we see that the integration interval δ must satisfy

$$\delta < \frac{1}{2.7p(2K+1)}\ .$$

In addition, to satisfy the decay property, $\ell_k(t) \approx 0$ for $t > t_1$ we must have $t_1 > 2K + 7$ for $p = 1$. Since p represents a linear time scaling, we may solve these relations for $p = 1$ and then modify the integration interval by a factor of $\frac{1}{p}$. This means that

$$\delta < \frac{1}{2.7(2K+1)}\,, \qquad t_1 > 2K + 7\,.$$

In the computer program, the parameters determining t_1 are δ and INTST, the number of points omitted from fitting, by the relation

$$t_1 = (\mathrm{INTST}-1)*\delta$$

Putting these together we find that

$$\frac{1}{2.7(2K+1)} \geqslant \delta \geqslant \frac{2K+7}{(\mathrm{INTST}-1)}\,.$$

Solving this for K, and δ gives the following table

K	INTST	δ
0	19	.37
1	73	.123
2	150	.074
3	250	.0525
4	366	.041
5	510	.034
6	670	.028
7	856	.025
8	955	.022

At this point the hard facts of computer size intrude. We are at present limited to consideration of the function at 1600 points. It seems wasteful to devote less than half of these to the fitting interval.

In the light of all these factors, we have chosen

$$K = 6$$

$$\delta = .025$$

$$\mathrm{INTST} = 800$$

as our working parameter set.

For completeness we must also ask if this integration interval is adequate to integrate the input satisfactorily.

For reasons which are explained in Appendix A, section 4, the fundamental period appearing in the input should equal the fitting interval $T - t_1$. Therefore, the shortest period will be $\frac{T-t_1}{10}$ and have 80 points used for integration. The following table shows relative error in integrating sinusoids by Simpson's Rule, showing that we are easily within our desired error of 10^{-4}.

Intervals/period	Relative error
4	4.7%
8	.23%
12	$4.3 \cdot 10^{-4}$
16	$1.3 \cdot 10^{-4}$

The selection of parameters having been made, we must examine the systems which can be approximated satisfactorily. For this we refer to Head's paper, op.cit., to find that for arbitrary α and p,

$$e^{-\alpha t} = \frac{\sqrt{2p}}{\alpha+p} \sum_{k=o}^{\infty} \left(\frac{p-\alpha}{p+\alpha}\right)^k \ell_k(t)$$

Of course p, the eigenvalue of the laguerre functions is positive (or has positive real part) in our application, so this series is convergent iff α has positive real part, i.e., if our fundamental assumption of asymptotic stability is satisfied. However, we limit the series to seven terms; therefore, to satisfy our arbitrary desire for 10^{-4} relative error (approximately four significant digits) we must have

$$\left|\frac{\alpha-p}{\alpha+p}\right|^6 \approx 10^{-4} .$$

This implies that

$$\left|\frac{\alpha-p}{\alpha+p}\right| \simeq 10^{-\frac{2}{3}} \simeq 0.215 \ .$$

The points α which satisfy

$$\left|\frac{\alpha-p}{\alpha+p}\right| = r$$

lie on a circle of radius

$$\frac{2r|p|}{1-r^2}$$

and center

$$p\,\frac{1+r^2}{1-r^2} \ .$$

Unfortunately this doesn't cover nearly the desired area in the complex plane. For instance, in Figure 1, we show two circles to indicate the types of regions we could consider.

The preceding analysis has led us to an impasse which tells us that under the existing conditions we cannot approximate the desired spectrum of functions with a fixed set of laguerre functions. To illustrate, to encompass both $\alpha = 10$ and $\alpha = 0.1$, the best choice of p is 1 and the value of r will be $\frac{9}{11}$. In order to obtain 10^{-4} error, nearly 50 terms would be needed, requiring $\delta < .004$ and at the same time a fit interval of 100 seconds (25000 points).

This is clearly out of the question. During our period of testing the effects of noise we will confine our attention to systems which can be adequately approximated, in the noise free situation, by a single, low-order laguerre fit. After the noise problem is sufficiently understood, the basis set can be expanded to cover more of the region of interest. A set of perhaps twenty roots could be chosen in the complex plane so as to minimize the fit error for any system in our region. On the other hand several sets of laguerre functions could be used.

Figure 2 shows how four sets of laguerre functions could cover most of the desired region while staying within our computational capabilities. Figure 3 shows an alternate configuration which while covering fewer oscillatory roots, blankets the real roots extremely well.

Figures 2 and 3 are only approximate of course, since when distinct eigenvalues of laguerre functions are involved, a reevaluation of the working parameters must be made.

We now turn to the problems of the integration procedures involved in forming $\{f_i(t)\}_o^K$.

2. Integration Methods:

Trapezoidal integration was used initially but proved inaccurate. A procedure designed to convolve a tabulated function with laguerre functions was programmed and tested but was found to be no more accurate than trapezoidal integration because it required taking differences of large numbers. The integration finally chosen and now in use is Simpson's Rule.

Our computational object in the beginning was to be able to identify all eigenvalues with real parts between -10 and -0.1 and "reasonable" imaginary parts. To obtain satisfactory integration accuracy we should have an integration interval of about 0.03 and should have a total fit interval $[t_1,T]$ of length about 30.

The integration interval is compatible with that previously determined by the laguerre functions. The total fit interval of $800 \cdot 0.025 = 20$ seconds in less than the three time constants which would be ideal but does provide two time constants for the worst case (-0.1 eigenvalue).

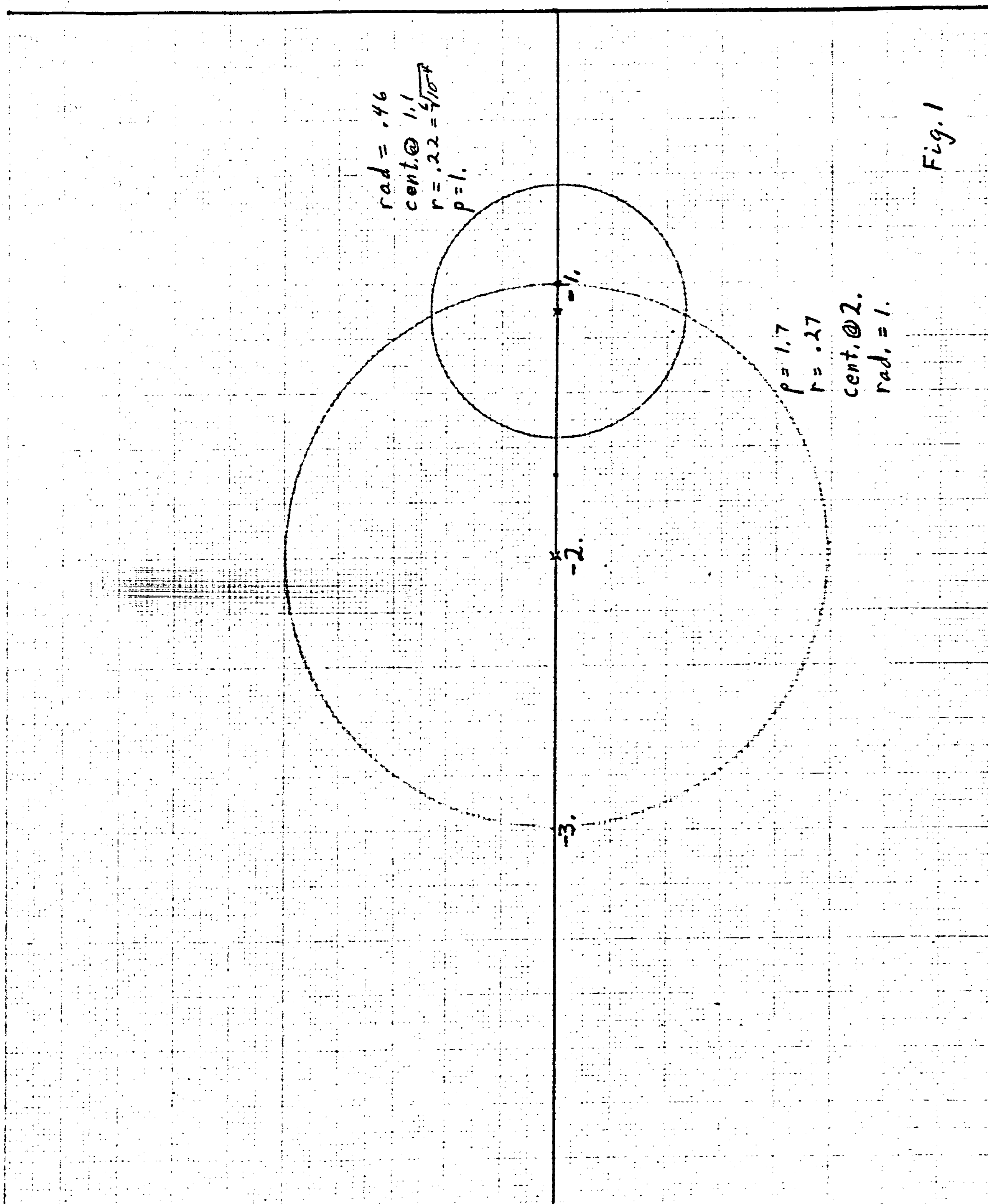


Fig. 1

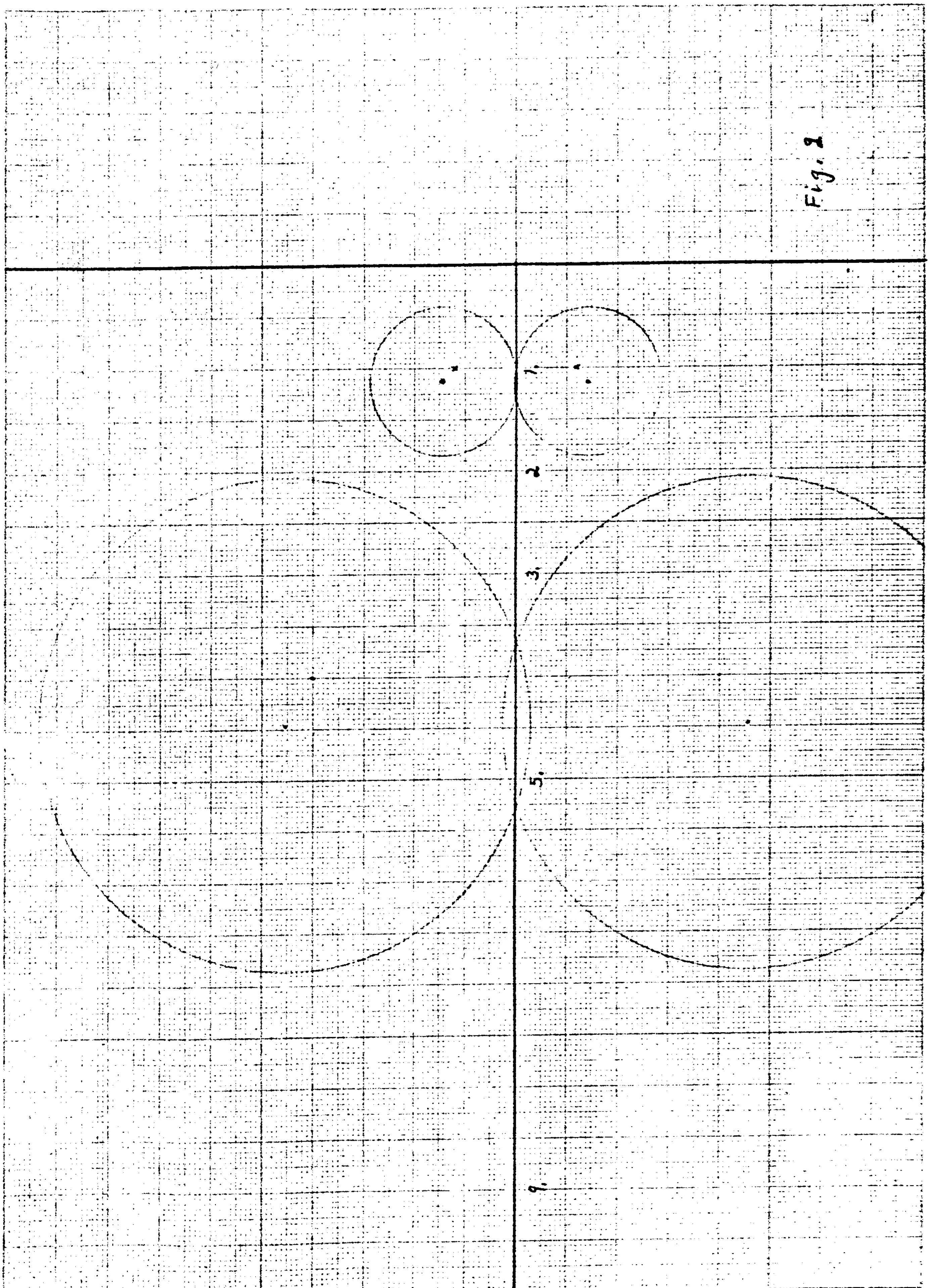


Fig. 1
1.
2
3.
5.
9.

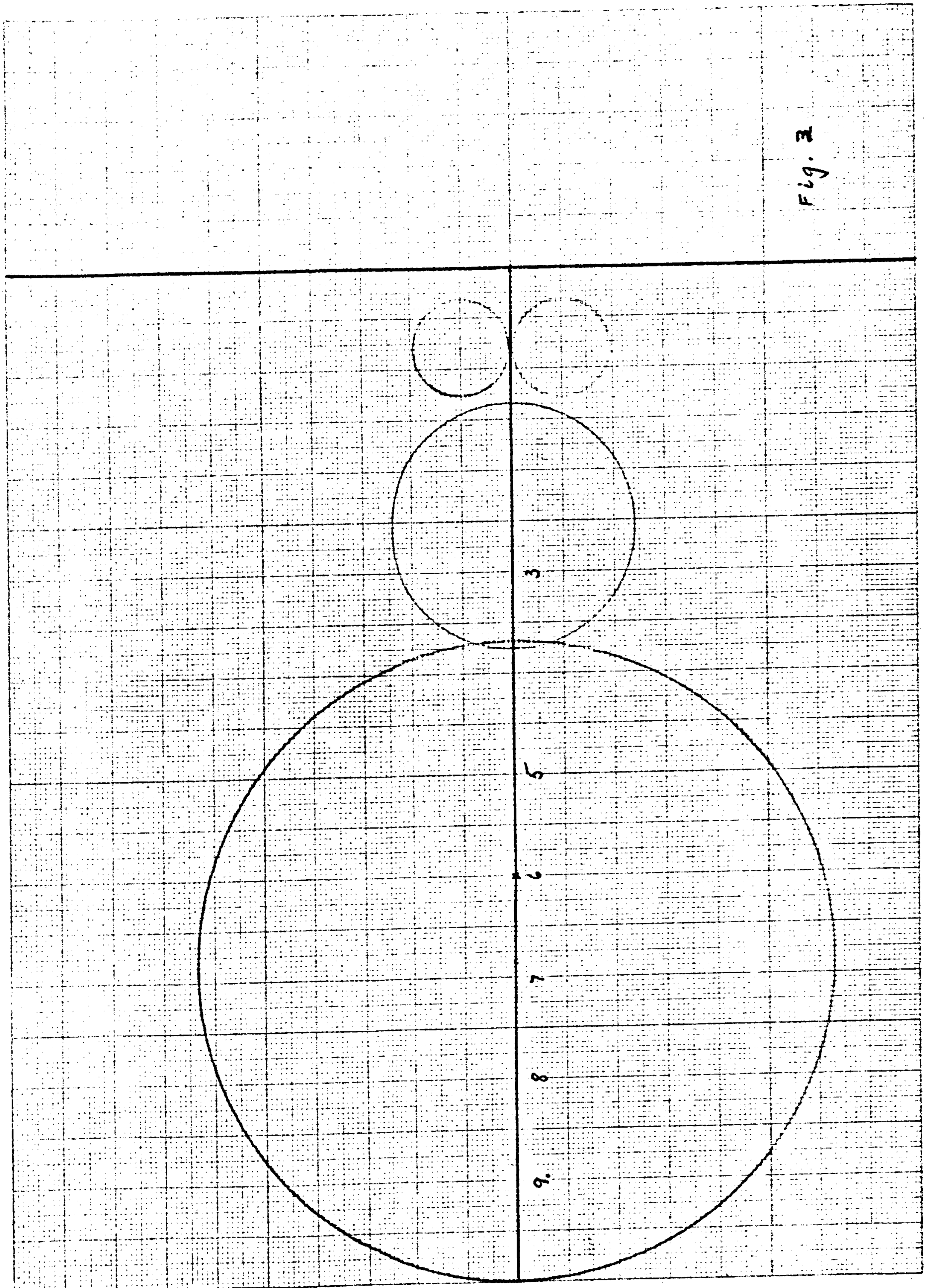


Fig. 3
3
5
6
7
8
9.

Number of intervals per time constant	Relative error
1	5.0 e-3
1.1	3.5 e-3
1.2	2.5 e-3
1.3	1.8 e-3
1.4	1.4 e-3
1.5	1.0 e-3
1.6	8.1 e-4
1.7	6.4 e-4
1.8	5.1 e-4
1.9	4.1 e-4
2.	3.4 e-4
2.1	2.8 e-4
2.2	2.3 e-4
2.3	1.9 e-4
2.4	1.6 e-4
2.5	1.4 e-4
2.6	1.2 e-4
2.7	1.0 e-4
2.8	9.0 e-5
2.9	7.7 e-5
3.	6.8 e-5

Table I

Relative error in integrating $e^{-\lambda t}$ by Simpson's Rule.

Chapter IV

Numerical Results

The results reported here were designed to give an estimate of the effects noise will have on the identification. In order to isolate these effects, the first system chosen was one which can be represented exactly by the laguerre functions. The system is

$$[1,\ 0] \quad \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad ,$$

having transfer function $\frac{1}{(s+1)^2}$ and kernel function

$$h(t) = te^{-t} \ .$$

For $p = 1$,

$$h(t) = \beta_o \ell_o(t) + \beta_1 \ell_1(t)$$

with $\beta_o = \beta_1 = \frac{1}{2\sqrt{2}} \simeq 0.353555.$

The complete set of parameters used in the Fit Program and in the Ho program appear in Table 1.

The problem was run first with no noise $(N/S = 0)$. This gave excellent results

$$\beta_o = .35355505$$

$$\beta_1 = .35355541$$

$$\beta_2 = -.11E-6$$

$$\beta_3 = .28E-6$$

$$\beta_4 = -.26E{-6}$$

$$\beta_5 = \ \ .89E{-7}$$

$$\beta_6 = -.26E{-6} \ .$$

Just how good these results are may be seen from the taylor coefficients $\{s_k\}$. These should be $s_k = (-1)^{k+1}k$ and in spite of the difficulty of computing derivatives we obtained

$$s_o = - \ \ .2E{-5}$$

$$s_1 = \ \ 1.00002$$

$$s_9 = \ \ 9.034$$

$$s_{16} = -16.77$$

Using $N/S = 0.5$, we found that results had a fairly large dispersion, indicating that using 800 points for fitting is not really adequate. In order to average over a larger number of points and to avoid drawing conclusions from a single noise burst, we ran five noise bursts.

The dispersion of the results were, in fact, so much larger than we had expected, that some additional checks were performed to verify program performance. Among these was a demonstration of linearity, done by fitting noise alone. This showed that the dispersions were indeed caused by the projections on the fitting functions $\{f_i(t)\}_o^6$ of the noise.

The actual computation of the eigenvalues was the most sensitive part of the process. Impulse responses and characteristic polynomials were usually obtained with fair accuracy.

Information about the impulse responses is summarized in Table 2. Burst 2 is undoubtedly the best, being virtually indistinguishable from the actual when graphed. Bursts 1 and 5 are the worst, Burst 1 having the lowest peak and Burst 5 having the highest initial and terminal errors. Nevertheless, the impulse responses obtained are not too bad. Figures 1 - 3 show the impulse responses for Bursts, 1, 3, and 5, together with the impulse responses of the associated realizations. Notice that the realizations depart from the fit in the second half of the interval. This occurs because of current space limitations in the ANALYSIS Program, these will be removed soon, enabling us to fit over the full range, rather than only over the first 2.3 seconds.

Table 3 shows the eigenvalues, characteristic coefficients, and input coefficicents (B vector) for the five realizations. Figure 4 shows the roots in the complex plane.

These roots are hardly good approximations to the actual roots, even though the fit impulse responses are, except for the initial value on 5, consistently in error by less than 10% of peak value. Part of this problem is caused by the coincident roots which are sensitive to the characteristic coefficients. For instance in Burst 2, the impulse response and the characteristic coefficients are in error by less than 2%, but the eigenvalues are individually wrong by 25%. Since coincident roots are not expected in practice, this particular problem need not concern us to much. In addition, we can expect some assistance from realizations using the larger interval mentioned above. Larger intervals, we know from experience, will tend to bring the roots, for this realization, closer to one, thus giving better eigenvalues. We might mention that in Bursts 2 and 4 the realization showed less tendency to depart from the fit response.

Although determination of system poles is a most important task, we must also be able to show the system zeroes. For this the last two columns of Table 3 are helpful. Naturally Burst 2 is the best.

When the noise to signal ratio was increased to one, all errors in $\{\beta_k\}_o^6$, $\{s_k\}$, and the impulse response increased linearly.

Table 4 gives the eigenvalues which were obtained from the five noise bursts and from the averaged β_k's of the five bursts.

The averaged impulse response for $N/S = 1$ appears in Figure 5. Again we are led to the conclusion that the procedures are working well and that we can obtain quite reasonable results even in the presence of low signal/noise ratios, but that 800 points is insufficient.

It is very clear here that the overall tendency of a noisy signal is to "spread" the impulse response so that the peaks are lower. In general this will probably tend to move the eigenvalues closer to the imaginary axis and to reduce the d.c. gain. It certainly tends to do that here. This is the only observed effect that cannot be removed by using more data.

Comparing Tables 3 and 4 with respect to linearity, we find b_1 and b_2 very linear (doubled noise, doubled error). This is because b_1 and b_2 are almost completely dependent, linearly, on the first and second sample points of the impulse response. The characteristic coefficients tend to look at the global impulse response and are almost but not quite linear, the eigenvalues obtained from them do not, of course, have linear behavior.

In all this work the noise was obtained from a digital pseudo-random noise generator. N/S was an input quantity and the noise standard deviation was set equal to

$$\frac{N}{S} \|z\|$$

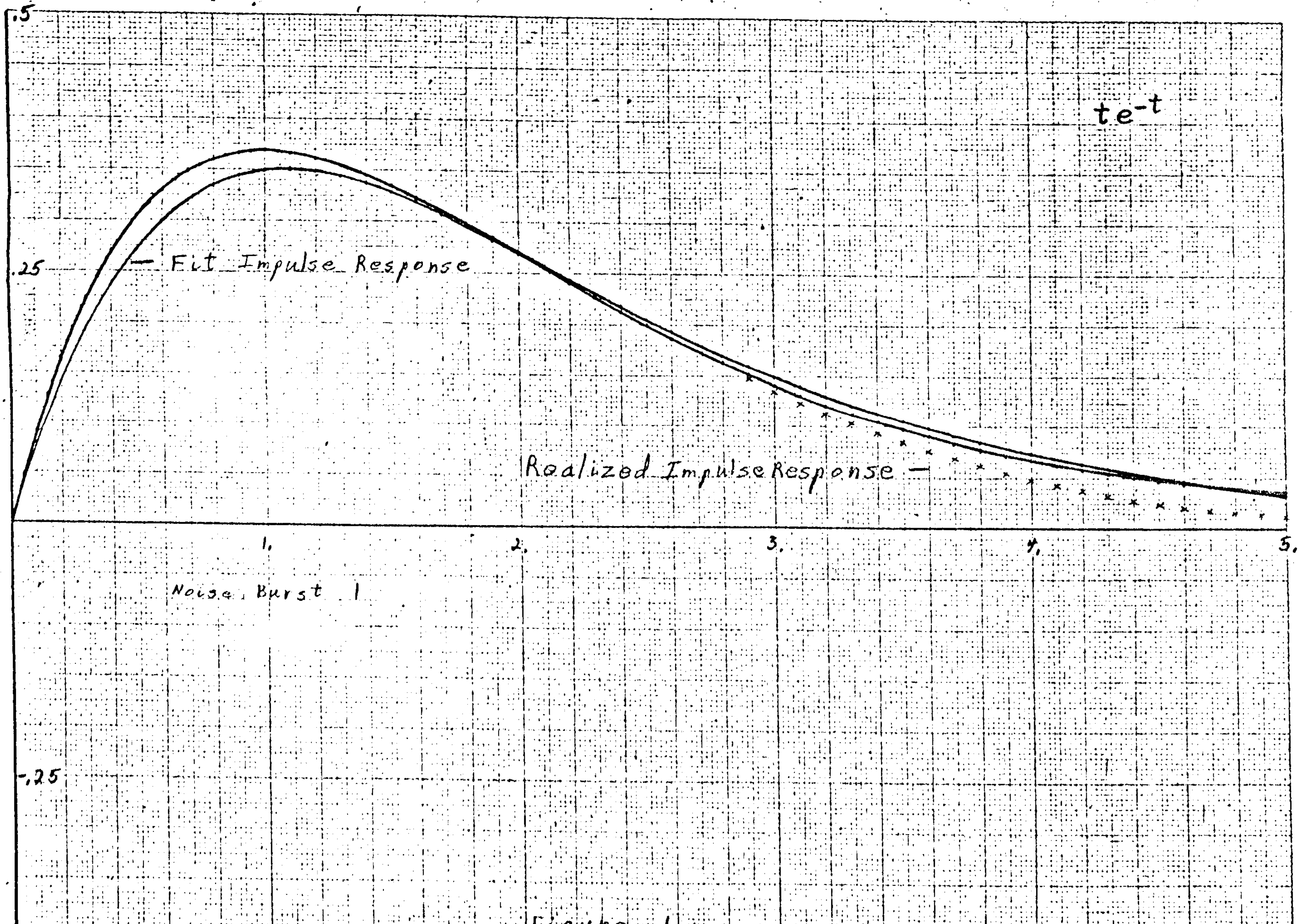
where for input

$$\sum_{k=1}^{10} c_k \sin k\omega t ,$$

having steady state output

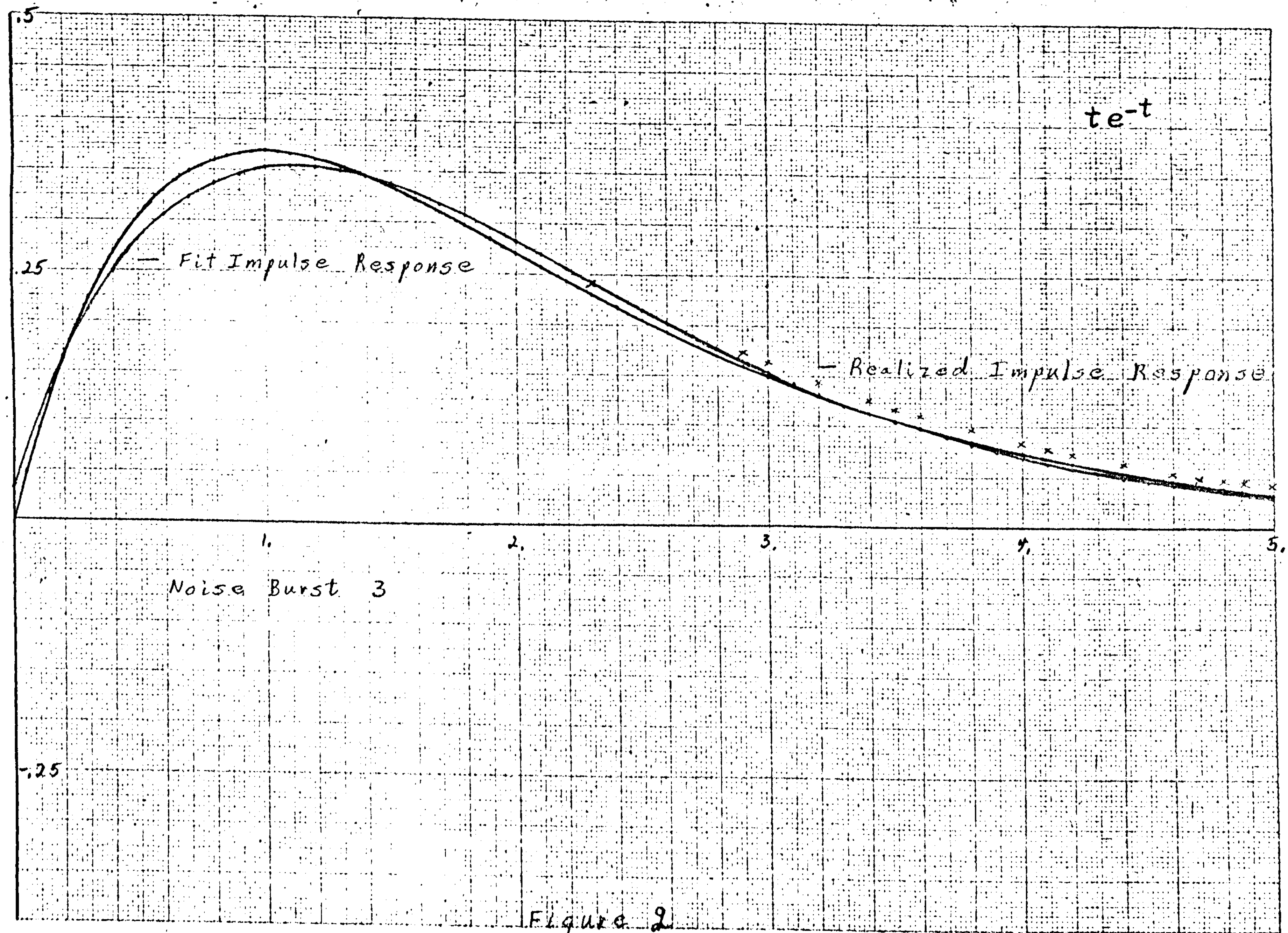
$$z(t) = \sum_{k=1}^{10} (a_k \cos k\omega t + b_k \sin k\omega t) ,$$

$$\|z\|^2 = \sum_{k=1}^{10} (a_k^2 + b_k^2) .$$



te^{-t}
Fit Impulse Response
Realized Impulse Response
Noise Burst 1
.5
.25
-.25
1.
2.
3.
4.
5.

Figure 1


.5
.25
-.25
1.
2.
3.
4.
5.
te^{-t}
Fit Impulse Response
Realized Impulse Response
Noise Burst 3
Figure 2

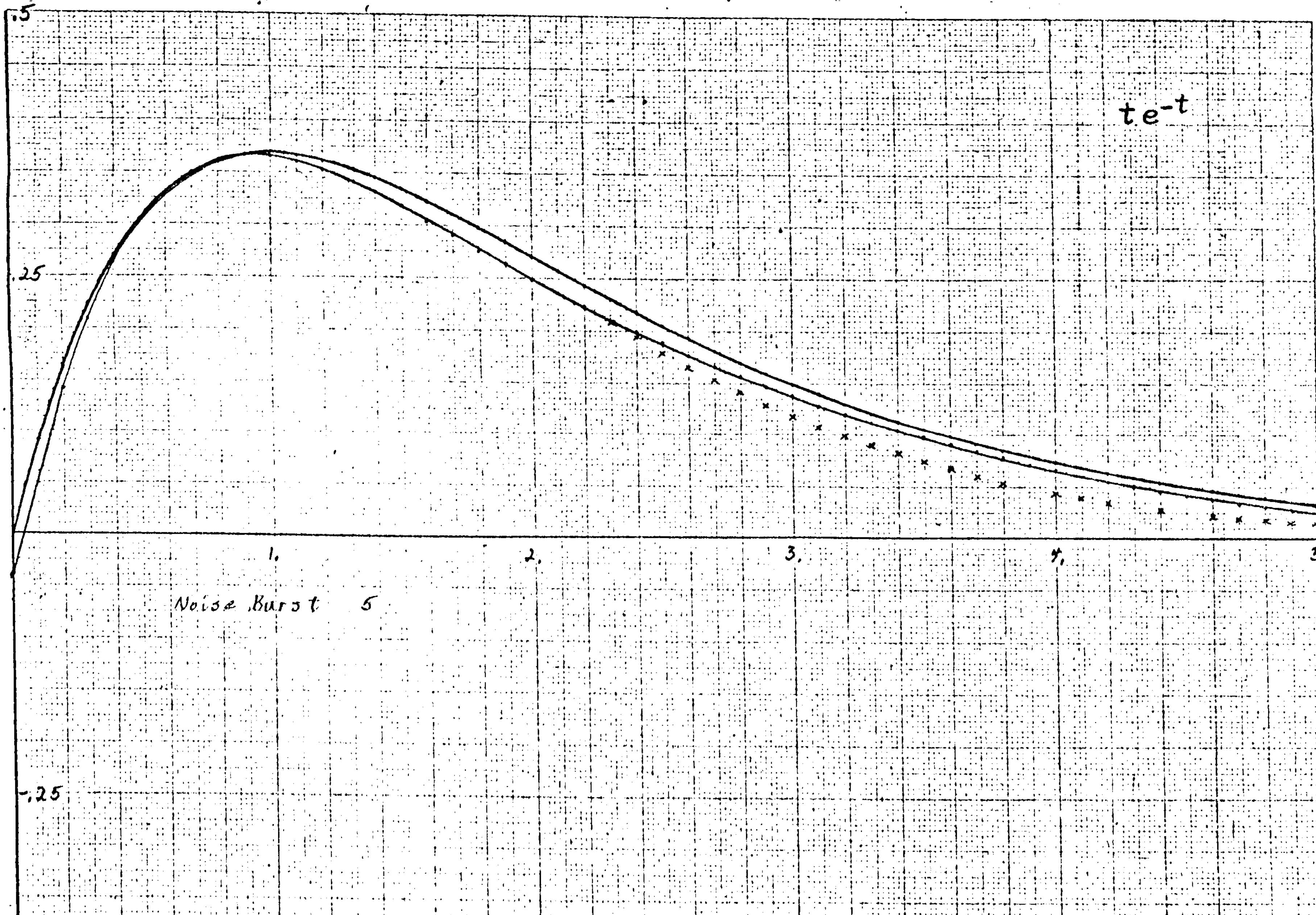
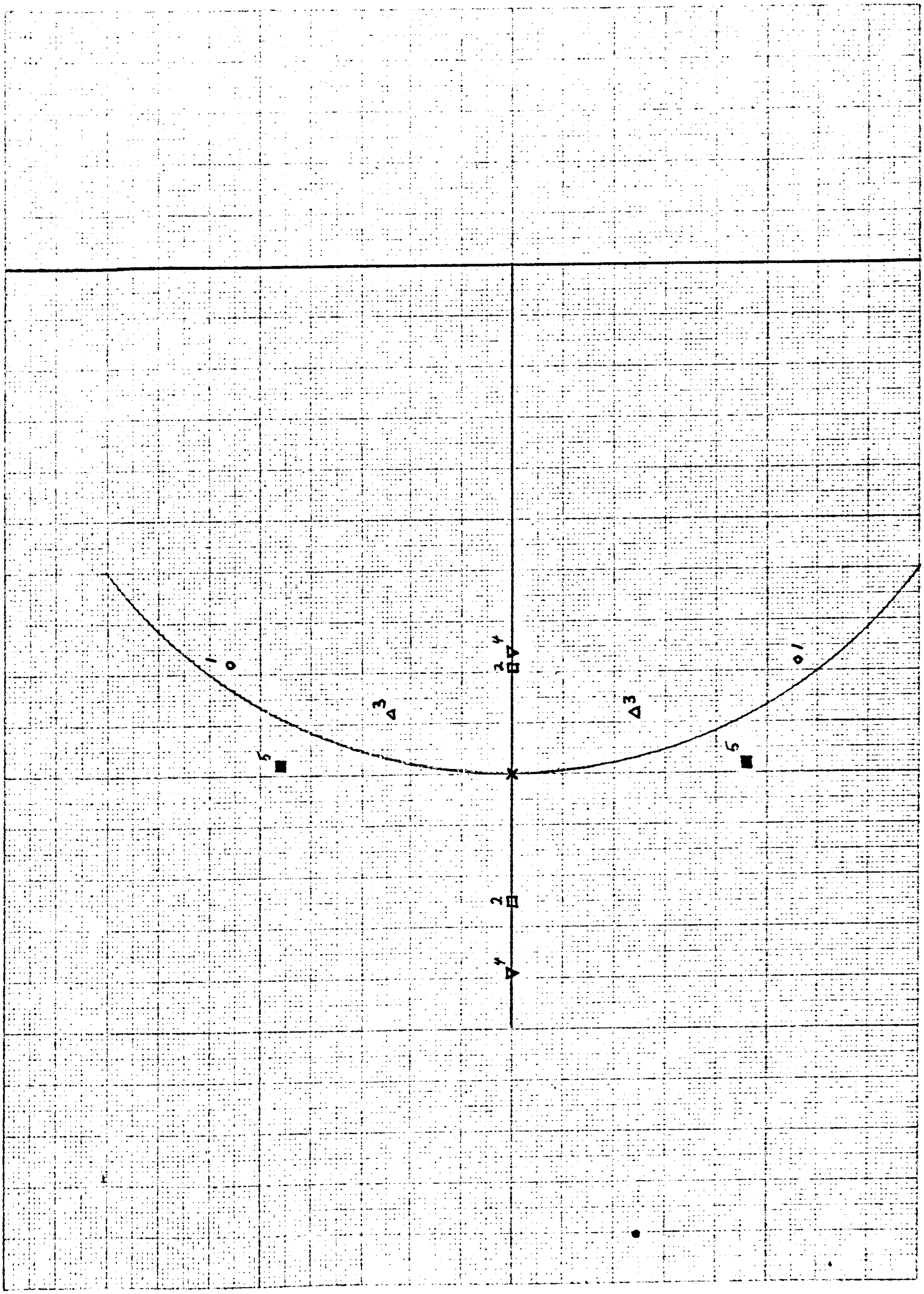
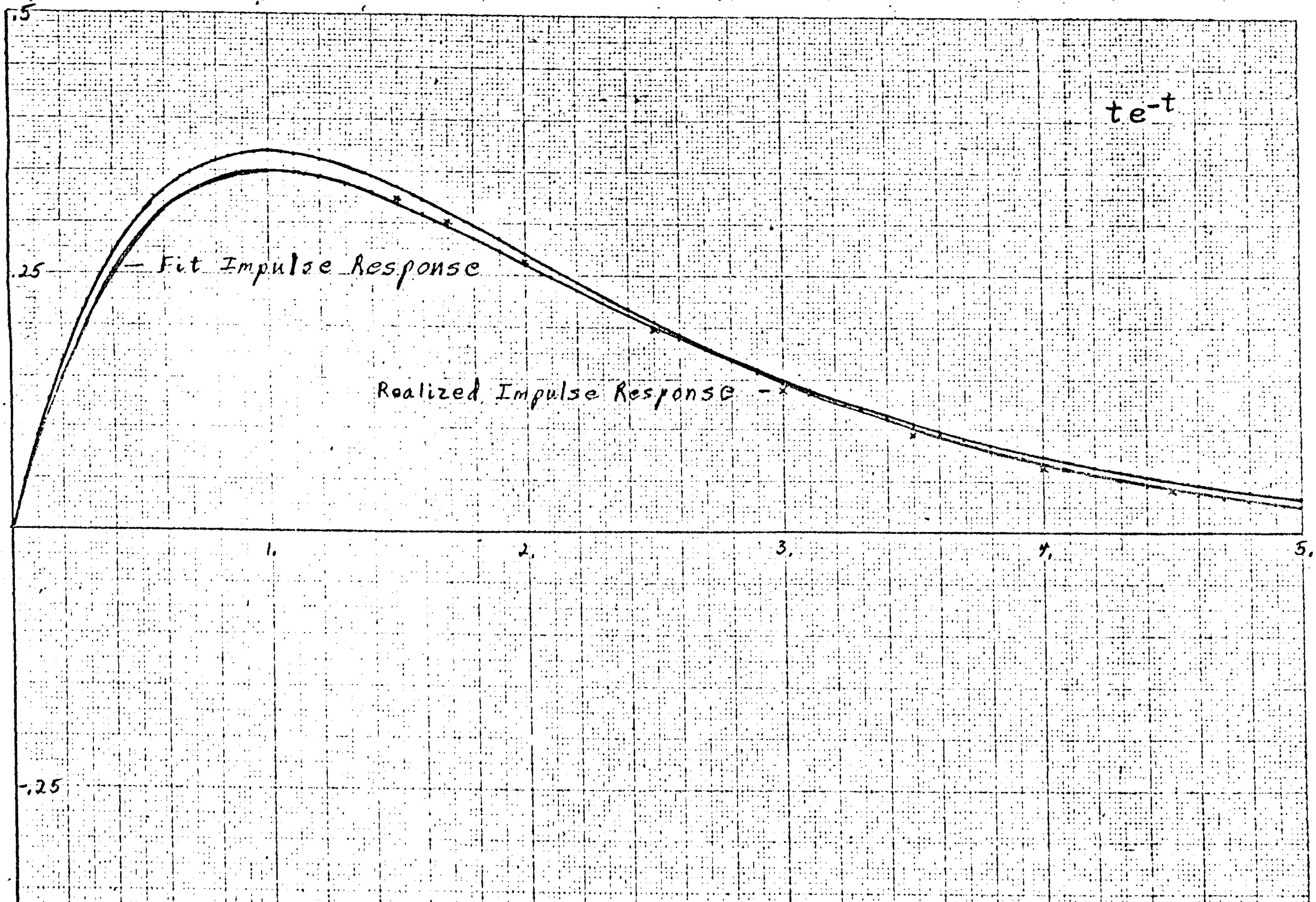


Figure 3





te^{-t}
Fit Impulse Response
Realized Impulse Response
.5
.25
-.25
1.
2.
3.
4.
5.

Figure 5

Fit Program Parameters

$K = 6$	(Order of laguerre approximation)
$\delta = .025$	(Integration step size)
$p = 1.$	(Eigenvalue of laguerre functions)
$N = 1600$	(Number of steps, input)
$N = 1599$	(Number of steps, used)
INTST = 800	(First point fitted, input)
INTST = 801	(First point fitted, used)
IORFOS = 10	(Number of sines in forcing function)
N/S = .5	(Noise/signal ratio)
$T_f = \frac{1}{2}$ (Fit Interval)	(Period of lowest frequency forcing term)

* * *

Analysis Program Parameters

$\delta = 0.1$	(Impulse response sampling interval)
$N = 49$	(Number of points used)
NST = 23	(Starting dimension of Hankel matrix)

Table 1

	Actual	Burst 1	2	3	4	5
y(0)	0	.0044	-.0003	.032	-.0011	-.041
y(.8)	.359	.334	.360	.337	.356	.361
y(.9)	.366	.344	.366	.346	.359	.366
y(1.0)	.368	.349	.368	.352	.358	.365
y(1.1)	.366	.351	.366	.354	.354	.360
y(1.2)	.361	.349	.361	.353	.347	.352
y(4.8)	.040	.038	.041	.036	.029	.030

Table 2

	λ_1	λ_2	$\lambda_1\lambda_2$	$\lambda_1 + \lambda_2$	b_1	b_2
Actual	-1	-1	1	-2	0	1
Burst 1	-.78 + .56i	-.78 - .56i	.93	-1.56	.004	.80
2	-.79	-1.25	.99	-2.04	-.0003	1.01
3	-.88 + .24i	-.88 - .24i	.83	-1.76	.032	.79
4	-.76	-1.39	1.05	-2.14	-.001	1.04
5	-.98 + .46i	-.98 - .46i	1.18	-1.96	-.041	1.10

Table 3

	λ_1	λ_2	$\lambda_1\lambda_2$	$\lambda_1+\lambda_2$	b_1	b_2
Actual	-1	-1	1	-2	0	1
Burst 1	-.52 + .79i	-.52 - .79i	.89	-1.04	.009	.60
2	-.71	-1.37	.98	-2.08	-.0006	1.03
3	-.68 + .49i	-.68 - .49i	.70	-1.36	.06	.59
4	-.72	-1.55	1.12	-2.27	-.002	1.08
5	-.97 + .67i	-.97 - .67i	1.38	-1.93	-.08	1.20
Average	-.91 + .37i	-.91 - .37i	.97	-1.82	-.002	.90

Table 4

Appendix A

The Fit Program

I. Purpose

The purpose of this program is to generate the coefficients $\{\beta_i\}_0^K$ in a finite expansion

$$\sum_{i=0}^{K} \beta_i \ell_i(t) \tag{1}$$

for the impulse response of an asymptotically stable, linear, stationary dynamical system. The functions $\{\ell_i(t)\}$ being used now in the program are the laguerre functions [1], but the modular construction of the program permits changes to different function sets.

The data on which the program works is the input function $u(t)$ to the unknown system and the output $z(t)$ which is the system response corrupted by noise. Here $t \in [0,T]$.

The problem is solved by assuming the impulse response to be represented in the form (1).

This function then is convoluted with the input to produce an output which is a function of the finite vector

$$\beta = [\beta_o, \cdots, \beta_K]'.$$

This is compared with the actual output function z over a subinterval $[t_1,T]$ to allow the effect of initial conditions to decay and a least square solution obtained for β.

The actual mechanization works with discretized functions $\{u_i\}$ and $\{z_i\}$, $u_i = u((i-1)\delta)$.

In addition to the vector β, the first few coefficients of the taylor expansion of (1) are printed.

The program described here works in a testing mode where the input and output sequences are generated internally from a known system. The deck described here uses a generalized inverse routine to solve for β. Other versions of the program, easily obtained from this one by modification, get the input-output sequences from externally generated cards and obtain β by inverting the normal matrix.

This program has the capability of iterating on P, the eigenvalue of the Laguerre functions, which is a free parameter in the expansion, to achieve a minimum of the fitting error. At present this is not in use (see V, Restrictions and Comments) but can be activated by removing the

GO TO 203

between EFN 404 and 510. (see VI, Procedure, and the listing in Appendix 1).

II. Operations - Input

1. The first input card has format

(3I10, 2E10.2)

it contains

N = number of subdivisions in the interval of interest [0,T]. Maximum 1600.

K = order of approximation. Maximum 19.

KS = number of Taylor coefficients desired. Maximum 29.

DEL = δ = Length of a subdivision.

$$\delta = \frac{T}{N}$$

Suggested range is $0.02 \leq \delta \leq 0.1$.

TSCALE = A parameter for scaling the time interval. Usually 1.

2. The second input card has format

(E13.8)

it contains

STDEV = the noise to signal ratio desired in the output, (self-generated-data operation). The input used is a sum of sinusoids, hence the noiseless output y(t) is a sum of sines and cosines. The square root of the sum of the squares of the coefficients is defined to be the norm of the output, $||y||$. STDEV * $||y||$ is the standard deviation of the noise added to $y(t)$.

3. The third input card has format

(I5)

it contains

INTST = the number of subdivisions ignored in the least square fitting. We allow INTST*DEL = t_1 time for transients to subside. INTST and N must satisfy N-INTST < 800.

4. The fourth input card has format

(7I10)

it contains 7 fixed point ones in that format. This card has purely historical significance.

5. The fifth input card has format

(I2)

it contains

NCASE = 1 if the run should terminate.

= 0 if another data set should be read.

Language is FORTRAN IV, no tapes are used.

III. Printout

The output $z(t) = y(t) + v(t)$, where $y(t)$ is the noiseless system response to the input and $v(t)$ is noise.

$y(t) = \sum_{k=1}^{10} (a_k \cos \frac{k}{2} t + b_k \sin \frac{k}{2} t)$. We define $||y|| = \left(\sum_{k=1}^{10} (a_k^2 + b_k^2) \right)^{1/2}$

this is printed as OUTPUT NORM = ____ .

STDEV is printed as NOISE TO SIGNAL RATIO = _____ .

The noise mean and standard deviation are printed.

The number of points (=N+1) in the interval [0,T] is printed.

The number of β coefficients (=K+1) is printed.

The number of taylor coefficients (=KS+1) is printed.

T (=N*DEL) is printed.

DEL the time increment, is printed.

The scaled time increment DEL*TSCALE is printed.

INTST is printed.

The rank of the matrix used to solve for β is printed as

RANK = ____ .

The K + 1 components of β are printed.

The KS + 1 taylor coefficients are printed.

P, the eigenvalue of the laquerre functions, is printed.

ERR, the experimental, relative standard deviation of the error,

$$ERR = \frac{1}{||y||}\left[\frac{\sum\limits_{i\,=\,INTST}^{N}\sum\limits_{j=0}^{K}[\beta_j f_j(t_i) - z(t_i)]^2}{N-INTST}\right]^{1/2}$$

is printed. Here

$$f_j(t) = \int_0^t \ell_j(z)u(t-\tau)d\tau .$$

IV. Subroutines

The modular construction of the program expresses itself in a relatively small MAIN and a large number of subroutines.

1. Subroutine GENIO. The purpose of subroutine GENIO is to compute the input vector $\{u_i\}_1^{N+1}$ and the output vector $\{z_i\}_1^{N+1}$.

$$u(t) = \sum_{k=1}^{10} c_k \sin \frac{k}{2} t$$

where $c_1 = c_2 = c_3 = c_5 = c_7 = c_9 = 1$

and $c_4 = c_6 = c_8 = c_{10} = -1$.

The sign changes are designed to minimize the effects of initial transients on the fitting procedures.

The noiseless output $y(t)$ is composed of only the equilibrium solution. The initial transient is omitted. There are two reasons for this. In the first place it is a more honest procedure since a better fit can be obtained if the correct transient is present and we must assume that we do not know the initial state of our system. Secondly it saves considerable machine time. This gives us

$$y(t) = \sum_{k=1}^{10} (a_k \cos \frac{k}{2} t + b_k \sin \frac{k}{2} t) ,$$

where a_k and b_k depend upon c_k and the system whose response is desired.

GENIO computes the OUTPUT NORM, $||y|| = \left[\sum_{k=1}^{10} (a_k^2 + b_k^2)\right]^{1/2}$

and forms SD = $||y||$*STDEV.

GENIO contains a random number generator and a sample v_i with mean zero and standard deviation SD is added to each sample y_i ($i \geq$ INTST) to form the noisy output z_i.

2. Function PHI1(T). Computes the number $\ell_o(T)$, the value at T of the first laquerre function.

$$\ell_o(t) = \sqrt{2p}\ e^{-pt} .$$

3. Function PHI2(T). Computes the number $\ell_1(T)$, the value at T of the second laguerre function.

$$\ell_1(t) = \sqrt{2p}\ e^{-pt}\ (2pt - 1).$$

4. Subroutine RCSN. This subroutine obtains $\ell_{k+1}(t)$ from $\ell_k(t)$ and $\ell_{k-1}(t)$ by the following recursion relation

$$\ell_{k+1}(t) = \frac{2pt - 2k - 1}{k+1}\ \ell_k(t) - k\ell_{k-1}(t)\ .$$

5. Subroutine FKSUB. This subroutine generates the functions

$$f_k(t) = \int_0^t \ell_k(\tau)u(t-\tau)d\tau\ .$$

In the actual mechanization, it forms equal matrices F and FP with elements

$$F(i,j) = \int_0^t f_{j-1}(\tau)u(t-\tau)d\tau$$

where $t = (\text{INTST} + 1 - i) * \text{DEL}$.

FKSUB calls functions PHI1 and PHI2 and subroutine RCSN.

6. Subroutine GINV2. This subroutine takes the matrix FP constructed in FKSUB and overwrites it with the transpose of $(FP)^+$, the pseudo inverse of FP.

The rank of FP is printed from GINV2; it should be equal to $k + 1$ in virtually all cases.

7. Function DOT. GINV2 requires the function DOT to compute inner products of the columns of FP.

8. Subroutine CHECK. This subroutine computes ERR, the fitting error.

Given the matrix F conducted in FKSUB, the vector β computed in MAIN, and the output vector z, it computes

$$||F\beta - z||^2 = \sum_{i=1}^{N+1-INTST} ((F\beta)_i - z_{INTST+1-i})^2$$

The experimental standard deviation of the error is computed from this

$$\frac{||F\beta - z||}{N-INTST} .$$

9. Subroutine DKPHI. This subroutine computes

$$\left. \frac{d}{dt} \ell_k(t) \right|_{t=0}$$

for $k = 0, \ldots, K$. These quantities are used in calculating the Taylor coefficients of the estimated impulse response.

10. Function BCOF. This function computes the binomial coefficients for use in DKPHI.

Several comments are in order concerning the subroutines.

When using experimental data entered from cards, we retain GENIO for the sake of convenience, but its purpose is solely to read cards.

Converting to use of the normal matrix rather than the generalized inverse requires considerable effort, including much use of double precision. Such a deck is available.

When changing from the laguerre functions to a different data set CHECK, GENIO, GINV2 would be retained unchanged. FKSUB would be somewhat modified, DKPHI would be completely altered, and the other routines more or less drastically changed, dependent upon the function set.

V. Restrictions and Comments

Dimension restrictions have been noted under II Input.

The program appears to be operating correctly, but as presently written it cannot be said to operate as well as expected. In the noise-free case, oscillatory systems with imaginary parts greater than about 2. do not yeild good fits. In the noisy case, even with 800 points to fit over, the approximation is not good enough to produce accurate results in the MICARE program (MSG PD-67-104). The β vector averaged from several distinct trials seems to do reasonably well. More information on the results can be obtained from a forthcoming MSG Technical Note.

The iteration on P to minimize ERR is not being used because it has proved ineffective in treating noisy data.

When fitting exact data, the iteration was extremely helpful in obtaining accurate information about the impulse response. However, the variations leading to this improvement were about 10^{-4} or 10^{-5} of $\|y\|$. Therefore at reasonable noise levels, this iteration was virtually useless.

VI. Procedure

Circled numbers, e.g. 26 are external formula numbers in the FORTRAN source program.

401 Initialize for iteration on P:

Set P

Read N,K,KS,DEL,TSCALE,STDEV,INST

Make N and INTST odd numbers, INTST $\geq$ 5.

Scale DEL.

Call GENIO to form $\{u_i\}$, $\{z_i\}$, and $\|y\|$.

551 Print N + 1, K + 1, KS + 1, T, DEL, TSCALE, T*TSCALE, DEL*TSCALE, INTST

Call FKSUB to form the equal matrices F and FP with

$$F(i,j) = \int_0^t \ell_i(\tau)\ u(t-\tau)d\tau$$

where $t = (\text{INTST} + j - 1)*\ \text{DEL}$.

Call GINV2 to obtain the pseudo-inverse and rank of FP.

23 Compute β as

$$\beta = (FP)^+ z$$

where only the components z_i of z from $i = \text{INTST}$ to $i = N + 1$ are used.

Call CHECK to obtain the standard deviation of the fit

$$\frac{\|F\beta - z\|}{N - \text{INTST}} .$$

29 Normalize this error by dividing by $||y||$

$$ERR = \frac{||F\beta - z||}{||y||\ (N-INTST)} \ .$$

Print β

Compute the Taylor coefficients

Print N, K, T, DEL

Print the Taylor coefficients.

Print P and ERR.

TO TO 203

(This omits P iteration for ERR minimization)

The error minimization is done by fitting a quadratic in P through the smallest three available errors. There is only one error return, when the second derivative is negative, i.e., when the function appears to have no minimum locally.

* * * * * * *

203 Read NCASE

If NCASE = 0, to to 401

STOP

VII. Mathematical Analysis

1. The Procedure.

Given the linear stationary dynamical system

$$\dot{x} = Fx + Gu$$
$$y = Hx \qquad (7.1)$$
$$z = y + v ,$$

where v is observational noise, we know that the output can be written as

$$z(t) = He^{tF}x(0) + \int_0^H He^{(t-\tau)F}Gu(\tau)d\tau + v .$$

By a change of variable, this can be rewritten as

$$z(t) = He^{tF}x(0) + \int_0^t He^{\tau F}Gu(t-\tau)d\tau + v .$$

From a knowledge of $z(t)$ and $u(t)$ only on some interval $[0,T]$, we want to obtain an estimate $\hat{h}(t)$ of

$$h(t) = He^{tF}G .$$

In order to do this, lacking knowledge of $x(0)$, we assume that F is asymptotically stable and that there exists a $t_1 < T$ such that in

$[t_1,T]$, $He^{tF}x(0)$ is very small compared with

$$\int_0^t He^{\tau F}Gu(t-\tau)d\tau \quad .$$

That is, we assume that on $[t_1,T]$,

$$z(t) = \int_0^t He^{\tau F}Gu(t-\tau)d\tau + v(t) \quad ,$$

and we then try to determine $\hat{h}(t)$ such that

$$\sigma^2 = \int_{t_1}^{T} [\int_0^t \hat{h}(\tau)u(t-\tau)d\tau - z(t)]^2 dt \tag{7.2}$$

is minimum.

Basically we use a Rayleigh-Ritz technique, that is we select a set of functions $\{\ell_i(t)\}$, which are "suitable" and represent $\hat{h}$ by linear combinations of the ℓ_i

$$\hat{h}(t) = \sum_{i=0}^{K} \beta_i \ell_i(t) \quad .$$

This reduces the problem to determining β such as to minimize σ^2 .

$$\int_0^t \hat{h}(\tau)u(t-\tau)d\tau = \sum_{i=0}^{K} \beta_i \int_0^t \ell_i(\tau)u(t-\tau)d\tau \quad .$$

We call the integrals above new functions

$$f_i(t) = \int_0^t \ell_i(\tau)u(t-\tau)d\tau \quad . \tag{7.3}$$

Then it is the (nonorthogonal) basis set $f_i(t)$ upon which we will project $z(t)$ to determine β. We are fitting the function $z(\cdot)$ on $[t_1,T]$ with the expansion

$$\sum_{h=0}^{K} \beta_i f_i(\cdot) \quad .$$

Naturally we are interested in the linear independence of $\{f_i\}_o^K$. In addition we should determine whether or not the system (7.1) can be uniquely determined from a knowledge of only z and u. These two questions are intimately connected as the development in 2 will show.

Assuming the functions $\{f_i\}_o^K$ to be independent, however, we can proceed.

Rewriting 7.2 in terms of the $f_i(t)$ gives

$$\sigma^2 = \int_{t_1}^{T} z(t) - \sum_{i=0} \beta_i f_i(t) \quad ^2 dt \ , \tag{7.2a}$$

which is then solved for the minimizing β vector.

2. Numerical Implementation.

A) The convolution integration in 7.3 is performed by Simpson's Rule, obtaining $f_i(t)$ at N+2 - INTST points on $[t_1, T]$. To expedite the mechanization, we insure an odd number of points on the interval $[0,t_1]$ by making INTST odd, and we make the number of points at which f_i is computed even by making N odd.

B) (7.2a) is minimized by using a generalized inverse routine to solve the linear finite system

$$[f_{ij}]\beta = z_i$$

where $f_{ij} = f_j(i\delta)$ and $z_i = z(i\delta)$.

3. <u>Linear Independence of</u> $\{f_i(t)\}_o^K$.

In order to investigate this we will consider only $u(t)$ of the type which we use, i.e.

$$u(t) = \sum_{k=1}^{M} c_k \sin \frac{k}{2} t , \qquad |c_k| = 1 . \tag{7.4}$$

We further assume that all $\ell_i(t)$ are impulse responses of asymptotically stable, linear stationary dynamical systems; this is in fact a <u>sine qua non</u> for being "suitable" to our problem. Because we are looking only at steady-state output $z(t)$, $t \geqslant t_1$, after initial transients have subsided, the analysis is somewhat simpler. For any asymptotically stable system (7.1) the steady-state output $y(t)$ for input $\sin \frac{k}{2} t$ is

$$y(t) = A_k \sin \frac{k}{2} t + B_k \sin \frac{k}{2} t. \tag{7.5}$$

Since the $\ell_i(t)$ are impulse responses, $f_i(t)$ may be thought of as the output of a linear dynamical system to the input $u(t)$ and therefore is the sum of terms like (7.5).

Therefore we have

Lemma: A necessary condition for the function $\{f_i(t)\}_o^K$ to be independent is that in (7.4), $M \geqslant \frac{K+1}{2}$.

Proof: $\{f_i(t)\}_o^K$ is a set of vectors from the $2M$ dimensional space spanned by

$$\{\sin \frac{k}{2} t \ , \ \cos \frac{h}{2} t\}_1^M$$

therefore if $K + 1 > 2M$, the set is linearly dependent.

In fact, we can write the vector

$$f = \begin{bmatrix} f_o \\ f_1 \\ - \\ - \\ - \\ f_k \end{bmatrix}$$

as

$$f = Av \tag{7.4}$$

where

$$v = \begin{bmatrix} \sin \frac{1}{2} t \\ \cos \frac{1}{2} t \\ \sin \frac{1}{2} t \\ - \\ - \\ - \\ \cos \frac{M}{2} t \end{bmatrix}$$

and A is a constant matrix. Then $\{f_i\}_o^K$ is linearly dependent if there exists a constant vector $p \neq 0$ such that

$$p'f = 0 \quad .$$

Since A is $(K+1)$ by $2M$ it is clear that such a vector exists if $K + 1 > 2M$.

It is tempting to hypothesize that the $\{f_i\}_o^K$ are linearly independent if $M \geqslant \frac{K+1}{2}$ and the set $\{\ell_i\}_o^K$ is linearly independent. Unfortunately this is not true.

Counterexample:

$$\ell_o = e^{-\lambda t}$$

and

$$\ell_1 = \frac{(\eta-\lambda)(\mu^2+1)}{(\eta-\mu)(\lambda^2+1)} e^{-\mu t} + \frac{(\lambda-\mu)(\eta^2+1)}{(\eta-\mu)(\lambda^2+1)} e^{-\eta t}$$

have the same steady-state response to $\sin t$, i.e., $f_o(t) \approx f_1(t)$ for t large.

Since this implies that the systems

$$H = 1 \qquad F = -\lambda \qquad G = 1$$

and

$$H = [1,1] \qquad F = \begin{bmatrix} -\mu & 0 \\ 0 & -\eta \end{bmatrix} \qquad G = \begin{bmatrix} \dfrac{(\eta-\lambda)(\mu^2+1)}{(\eta-\mu)(\lambda^2+1)} \\ \dfrac{(\lambda-\mu)(\eta^2+1)}{(\eta-\mu)(\lambda^2+1)} \end{bmatrix}$$

have the same steady-state response to $u(t) = \sin t$, it is clear that we do have problems also in determining the system uniquely solely from input-output information.

Both questions can be answered easily however with the help of the following.

Corollary: Let $h(t)$ be the impulse response of a c.c. - c.o., asymptotically stable linear stationary dyanamical system. Let

$$\mathcal{L}\, h(t) = \frac{p(s)}{q(s)} \quad .$$

Then the steady-state response $f(t)$ of the system to $u(t)$, that is

$$f(t) \approx \int_0^t h(t-\tau)u(\tau)d\tau \qquad \text{for large} \quad t \ ,$$

is zero if and only if $u(t)$ satisfies the homogeneous differential equation represented in the frequency domain by

$$p(s) \ ,$$

i.e., $\mathcal{L}^{-1}(p(s))\ u(t) = 0$.

Proof: This is a corollary to the much more general theorem by Leonard Weiss [1].

Applying this to our case, we take the Laplace transforms of $\{\ell_i\}_o^K$, $\{\frac{p_i}{q_i}\}_o^K$ and compute

$$\frac{p(s)}{q(s)} = \sum_{i=0}^{K} a_i \frac{p_i}{q_i} \quad .$$

If $\deg p(s) < 2M$ then the functions $\{f_i\}_o^K$ form a linearly independent set. In particular:

Case 1: The laguerre functions,

$$\frac{p_{i(s)}}{q_i(s)} = \frac{p_{i(s)}}{(s+1)^{i+1}} .$$

Therefore $\deg p(s) < K + 1$, hence $2M \geq K + 1$ is both necessary and sufficient for linear independence.

Case 2: The Kautz function [2].

For the Kautz functions $\deg p \leq \deg p_K < K + 1$ for K odd and $\deg p = \deg p_{K+1} = K + 2$ for K even. In any case then, we have the same result, $2M \geq K + 1$ is both necessary and sufficient for linear independence.

Case 3: Arbitrary pole selection.

If we select

$$\left.\begin{aligned} \ell_{2i} &= e^{-\lambda_i(t)} \cos w_i t \\ \ell_{wi+1} &= e^{-\lambda_i(t)} \sin w_i t \end{aligned}\right\} \quad \text{for } i = 0,n;\ w_i \neq 0 ,$$

and

$$\ell_i = e^{-\lambda_i t} \qquad \text{for } i = 2n+2,\ldots,K$$

with $w_i \neq w_j$ for $i \neq j$ and $\lambda_i \neq \lambda_j$ for $i \neq j$, i, $j \geq 2n + 2$ then $\deg p(s) < K + 1$. Again we have that $2M \geq K + 1$ is both necessary and sufficient for linear independence.

4. Uniqueness of Identification.

We wish to determine what system estimates

$$\hat{h}(t) = \sum_{i=0}^{K} \beta_i \ell_i(t)$$

can be obtained with fixed M and a set $\{\ell_i\}_o^K$, $K + 1 \leq 2M$, such that $\{f_i\}_o^K$ are linearly independent.

The counterexample in (3) can help our thinking about the problem. Letting $\lambda = 1$, $\mu = 2$, $\eta = 3$, we find that the systems

$$H_1 = 1 \qquad F_1 = -1 \qquad G_1 = 1$$

and

$$H_2 = [1,1] \qquad F_2 = \begin{bmatrix} -2 & 0 \\ 0 & -3 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 5 \\ -5 \end{bmatrix}$$

have the same response to $u(t) = \sin t$. However they have impulse responses

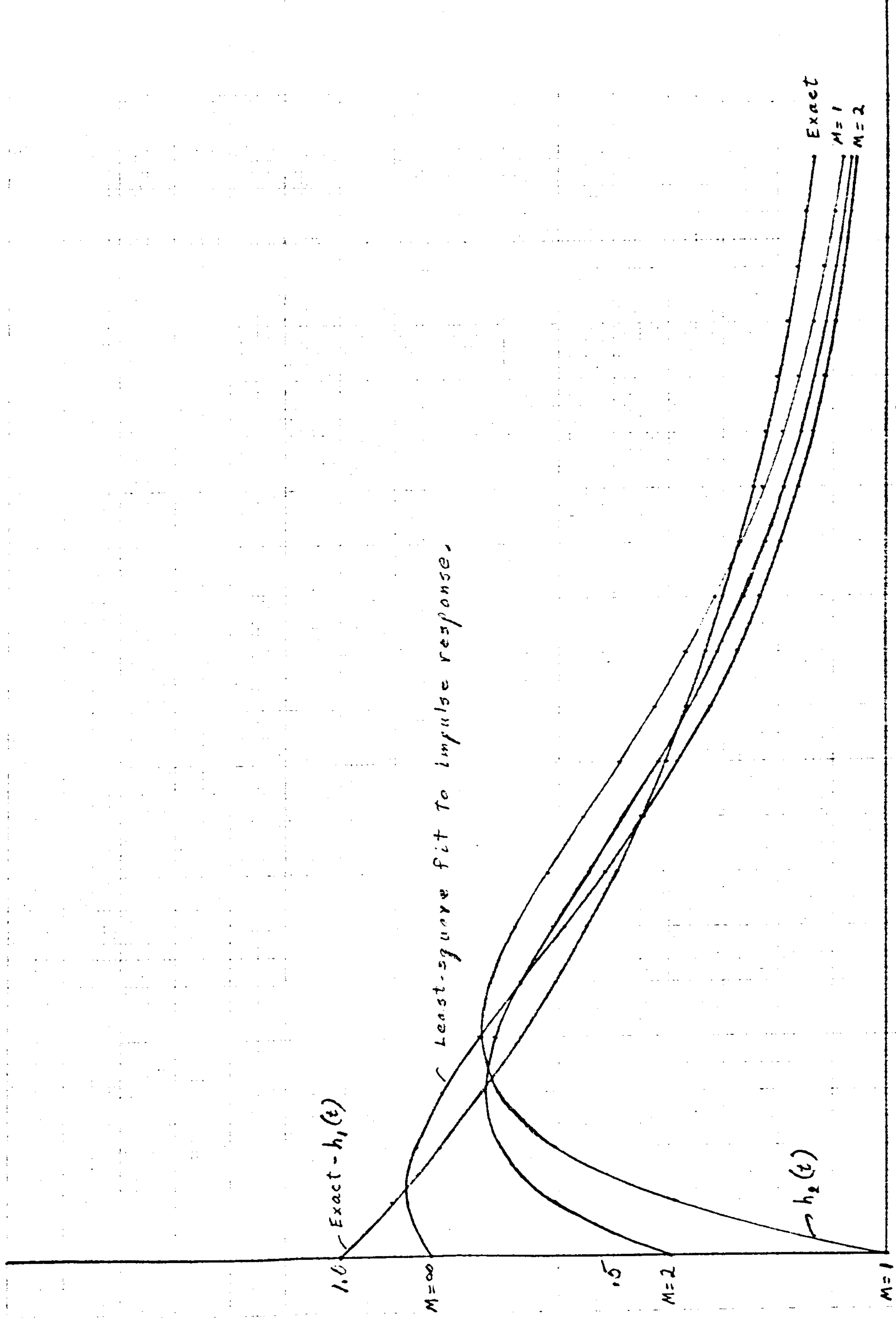
$$h_1(t) = e^{-t} \text{ , } h_1(s) = \frac{1}{s+1}$$

and

$$h_2(t) = 5e^{-2t} - 5e^{-3t} \text{ , } h_2(s) = \frac{5}{(s+2)(s+3)} \text{ .}$$

Figure 1 shows $h_1(t)$ and $h_2(t)$.

In their expansions in $\frac{1}{s}$ we have


Least-square fit to impulse response,
Exact - $h_1(t)$
$h_2(t)$
Exact
M=1
M=2
.5
1.0
1.5
2.0
1.0
M=∞
.5
M=2
M=1

$$h_1(s) \approx [1, -1, 1, -1, 1, \ldots]$$

$$h_2(s) \approx [0, 5, -25, 95, -325, \ldots] \quad .$$

This shows that we can get an exact fit of the input-output relations and be very far wrong in the impulse response. We attempt to circumvent the problem by increasing M. For instance if in the previous example, we let $u(t) = \sin t + \sin 2t$, then we obtain the algebraic system

$$\begin{bmatrix} \frac{2}{5} & \frac{3}{10} \\ -\frac{1}{5} & -\frac{1}{10} \\ \frac{2}{8} & \frac{3}{13} \\ \frac{2}{8} & \frac{2}{13} \end{bmatrix} \begin{bmatrix} \beta_o \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{5} \\ -\frac{2}{5} \end{bmatrix} .$$

Here β_o and β_1 are respectively the coefficients of the functions $\ell_o = e^{-2t}$ and $\ell_1 = e^{-3t}$ which will minimize (7.2). The optimal β_i are

$$\beta_o = 4.01572$$
$$\beta_1 = 3.62098$$

and the impulse response appears in Fig. 1.

The most unfortunate aspect of the procedure is that the error

$$\varepsilon^2 = \int_0^\infty (h(t) - \hat{h}(t))^2 dt$$

is not a monotonic function of σ^2, in 7.2, for fixed K. For instance in this case the vector which minimizes ε^2 is

$$\beta_o = 3\frac{1}{3}$$
$$\beta_1 = 2.5 \quad .$$

The impulse response for this fit appears in Fig. 1 also.

Note, in fact, that ε^2 does not necessarily decrease for fixed M as K increases. In fact we can obtain a better ε^2 fit with $\beta_o = \frac{3}{2}$, $\beta_1 = 0$, which is the minimum σ^2 fit for $K = 0$, $M = 1$ than by minimizing σ^2 for $K = 1$, $M = 1$.

Remark:

$$\int_0^\infty (e^{-t} - \beta_o e^{-2t} - \beta_1 e^{-3t})^2 dt$$
$$= \frac{15\beta_o^2 + 24\beta_o\beta_1 - 40\beta_o + 10\beta_1^2 - 30\beta_1 + 30}{60} \quad .$$

Now we see that there are two aspects of the uniqueness question. Let us take an asymptotically stable system (7.1) of order n and record its steady-state output for $2M \geq n$. Then there is only one system of order n which will give that output.

On the other hand if the eigenvalues are unknown and we use some arbitrary set of functions $\{\ell_i(t)\}_o^K$ then it is not necessarily true that we are fitting the impulse response more closely as K increases with M remaining fixed, even though the functions $\{f_i(t)\}_o^K$ are linearly independent. In fact, if the charaeteristic polynomial of the unknown nth order system has no roots in common with the $\ell_i(t)$, then when $n + K + 1 > 2M$, we can make $\sigma^2 = 0$ and still have a large ε^2.

Before attempting any conclusions about uniqueness or operational procedures we should obtain a better idea of the mathematical principles which underlie the process we are using. That will be done for a slightly idealized variation in the development which follows. The idea may be stated easily. Instead of minimizing $\|L\beta - h\|$, where

$$L = [\ell_o(t), \ldots, \ell_K(t)] ,$$

we are minimizing $\|L\beta - h\|_Q$, where Q is a non-negative definite symmetric kernel.

What our program does is to minimize

$$\sigma^2 = \int_{t_1}^{T} [F\beta - z(t)]^2 dt ,$$

where F is the $K + 1$ - component new vector with

$$F_i = \int_0^t u(t-\tau)\ell_{i-1}(\tau)d\tau .$$

Using the definition of F we can rewrite σ^2 as

$$\int_{t_1}^{T} \int_0^t [\beta L(\tau)-h(\tau)]u(t-\tau) \int_0^t u(t-s)[(s)\beta-h(s)]dsd\tau dt .$$

We now assume explicitly that $L(t) = 0 = h(t)$ for $t \geqslant t_2$ and that $t_1 \geqslant t_2$. Interchanging integrals then gives us

$$\int_0^{t_2}\int_0^{t_2} [\beta L(\tau)-h(\tau)] \int_{t_1}^{T} u(t-\tau)u(t-s)dt\ [L(s)\beta-h(s)]dsd$$

$$= \int_0^{t_2}\int_0^{t_2} [\beta L(\tau)-h(\tau)]Q(\tau,s)[L(s)\beta-h(s)]dsd\tau \quad .$$

$Q(\tau,s)$ is clearly non-negative definite symmetric. Furthermore, with

$$u_M(t) = \sum_{k=1}^{M} \sin k\,\omega\, t \quad ,$$

if $T - t_1$ is a multiple of $\frac{2\pi}{\omega}$, then the components of $u(t)$ are orthogonal, and ε^2 associated with $u_{M+1}(t)$ is less than $u_M(t)$. (This follows from the fact that the eigenfunctions of $Q_M(s,t)$ with nonzero eigenvalues are orthogonal and coincide with a subset of the eigenfunctions of $Q_{M+1}(s,t)$.)

One thing that is not clear from this is the speed with which

$$\|L\beta - h\|_Q \to \|L\beta - h\| \quad .$$

Treated as a periodic function, each component of L has a discontinuity at zero and therefore has considerable high frequency power. In fact because of this discontinuity, we cannot prove simply that

$$\|L\beta - h\|_Q \to \|L\beta - h\|$$

and we cannot expect convergence better than $\frac{1}{M}$.

We can draw some recommendations from this analysis for use in our operational procedures.

1) The input should contain a constant.

2) The lowest frequency, w , appearing in u should be such that

$$T - t_1 = \delta(N + 1 - \text{INTST})$$

is a multiple of $\frac{2\pi}{\omega}$.

3) It might be a good idea to try using some $\ell_i(t)$ which are zero at $t = 0$, to avoid the discontinuity.

It is interesting that when

$$u = \frac{1}{4} + \sum_{k=1}^{M} \sin k\, w\, t$$

then the procedure, in effect, takes the Mth order approximant $\hat{L}$ of L and minimizes $||\hat{L}\beta - h||$.

References

[1] Leonard Weiss, "On a Question Related to the Control of Linear Systems," IEEE Transactions on Automatic control vol. AC-9, Number 2, April 1964.

[2] William H. Kautz, Transient Synthesis in the Time Domain, IME Transactions-Circuit Theory, September, 1954.

```
 04/29/68                       80/80 LIST
$JOB  MM1    MATHAI-RZ3 U31142 U418  T418  9106  C 010 005
'N HDG  THIS PROGRAM REFERENCES MATH-PACK LIBRARY
' ASG A=$MATHP
' XQT CUR
 IN A
'IT FOR MAIN
C       GIVEN THE INPUT-OUTPUT DATA OF A PHYSICAL SYSTEM, THIS PROGRAM
C       APPROXIMATES THE COEFFICIENTS OF THE IMPULSE RESPONSE FUNCTION
C       PROGRAM INPUT
C       U                 SYSTEM INPUT DATA
C       Z                 SYSTEM OUTPUT DATA
C       N                 INTERVAL SIZE - N+1 DEFINES THE NO. OF POINTS USED
C                         MAX. VALUE OF N IS 500
C       K                 K+1 DENOTES THE ORDER OF THE LEAST SQUARES FIT
C                         MAX. VALUE OF K IS 29
C       KS                KS+1 DEFINES THE NUMBER OF SK COEF. DESIRED
C                         MAX. VALUE OF KS IS 29
C       DEL               ACTUAL TIME INCREMENT OF DATA POINTS
C       TSCALE            SCALING FACTOR USED ON TIME
C                         TO SCALE TIME BETWEEN 0-1 USE TSCALE=FINAL TIME
C       MAX               MAX NO. OF ITERATIONS ALLOWED IN EVALUATING THE INVERSE
C       ELEAST            ACCURACY LEVEL DESIRED IN THE INVERSE
C       KPF               =1 PRINT F MATRIX
C                         =0 DO NOT PRINT
C       KPZ               =1 PRINT Z VECTOR
C                         =0 DO NOT PRINT
C       KPF1              =1 PRINT INVERSE OF F MATRIX
C                         =0 DO NOT PRINT
C       KPI               =1 PRINT IDENTITY MATRIX TO TEST INVERSE
C                         =0 DO NOT PRINT
C       KPITER            =1 PRINT NO. OF ITERATIONS AND MAX. ERROR IN INVERSE
C                         =0 DO NO PRINT
C       KPRES             =1 PRINT RESIDUALS = ZVEC-FMAT*BETA
C                         =0 DO NOT PRINT
C       KPB               =1 PRINT BETA VECTOR
C                         =0 DO NOT PRINT
        COMMON/TEN/U(1603)
        COMMON /SCALE/ P
        COMMON  /NORM/  ERR,INTST
        COMMON/FK/F(801,20),PHI(2,1603),DELT,NP1,KP1,FP(801,20)
        DIMENSION AFLAG(20),ATEMP(20)
        DIMENSION SK(30),BETA(20),Z(1603),UNIT(20,20)
        DIMENSION DPHI(20)
        DIMENSION E(3),D(3)
        DOUBLE PRECISION PHI,BETA,SK
        INTEGER COUNT
C       FORMATS
    100 FORMAT(3I10,2E10.2)
    101 FORMAT(I10,D10.2)
    102 FORMAT(7I10)
    200 FORMAT(1H1,45H DYNAMICAL SYSTEM MODELING OF HUMAN OPERATORS///
      118H I - LINEAR MODELS//////62H NO. OF INPUT-OUTPUT POINTS USED IN
      2LEAST SQUARES FIT - N+1 = ,I5//40X,22H ORDER OF FIT - K+1 = ,I3//
      328X,34H NO. OF SK COEF. DESIRED - KS+1 = ,I3///////)
    201 FORMAT(3X,30H SIZE OF TIME INTERVAL USED = ,F10.5//33H TIME INCREM
      1ENT OF DATA POINTS = ,F10.5//2X,31H SCALING FACTOR USED ON TIME =
      2,F10.5//9X,24H SCALED TIME INTERVAL = ,F10.5//8X,25H SCALED TIME I
```

```
  3NCREMENT = ,F10.5//////)
202 FORMAT(51H MAX. NO. OF ITERATION ALLOWED TO OBTAIN INVERSE = ,I5//
  114X,37H ACCURACY LEVEL DESIRED IN INVERSE = ,E10.2)
150 FORMAT(1H1,27H F MATRIX - PRINTED BY ROWS)
151 FORMAT(///4H ROW,I3//(5E20.8))
152 FORMAT(1H1,18X,1HI,16X,4HZ(I)/(10X,I10,E20.8))
153 FORMAT(1H1,28H F INVERSE - PRINTED BY ROWS)
154 FORMAT(1H1,34H IDENTITY MATRIX - PRINTED BY ROWS)
155 FORMAT(1H1,18X,1HK,13X,7HBETA(K)/(10X,I10,D20.8))
156 FORMAT(1H1,25H SK VECTOR OF SCALED TIME//19X,1HI,15X,5HSK(I)/
  1(10X,I10,E20.8))
157 FORMAT(///18H NO. OF ITERATIONS,I5///11H MAX. ERROR,7X,D20.8)
158 FORMAT(1H1,27H SK VECTOR OF ORIGINAL TIME//19X,1HI,15X,5HSK(I)/
  1(10X,I10,E20.8))
159 FORMAT(1H1,18X,1HI,11X,9HRESIDUALS)
160 FORMAT(10X,I10,E20.8)
161 FORMAT(//,5X,8H INTST= ,I5)
162 FORMAT ( E13.8 )
401 CONTINUE
    COUNT = -1
    IMAX = 3
    E(3) = +.1E+30
    D(3) = +.1E+30
    P = .4132223
    P=2.0
    P = .5
    P=4.0
    P=2.7
    P=1.4641
    P = 1.1
    P=1.0
    READ 100 ,N,K,KS,DEL,TSCALE
    READ 162 , STDEV
    READ 103, INTST
103 FORMAT(I5)
    N = N/2
    N=2*N-1
    INTST = INTST/2
    INTST = 2*INTST + 1
    IF (INTST.LE.5) INTST=5
      DELT = DEL/TSCALE
    READ 102 ,KPF,KPZ,KPFI,KPI,KPITER,KPRES,KPB
     DEL  =  DELT*TSCALE
C   GENIO SUBROUTINE GENERATES THE SYSTEM INPUT-OUTPUT DATA FOR A TEST CA
    CALL GENIO (N,DEL,U,Z,STDEV,SSS,INTST)
551 CONTINUE
C   THE LEAST SQUARE SOLUTION IS OBTAINED BY SOLVING THE MATRIX
C   EQUATION        FMAT*BETA=ZVEC
C   COMPUTE FMAT
    DELT=DEL/TSCALE
    T2=1.0/DELT
    T1=DEL*N
    TT=DELT*N
    NP1=N+1
    KP1=K+1
    KSP1=KS+1
    PRINT 200,NP1,KP1,KSP1
```

```
      PRINT 201 ,T1,DEL,TSCALE,TT,DELT
      PRINT 161 , INTST
      CALL FKSUB
      NR=NP1+1-INTST
      CALL GINV2(FP,UNIT,AFLAG,ATEMP,801,NR,KP1)
C     COMPUTE BETA VECTOR
   23 DO 71 I=1,KP1
      BETA(I)=0.0
      DO 70 J=1,NR
      L=INTST-1+J
      BETA(I)=BETA(I)+FP(J,I)*Z(L)
   70 CONTINUE
   71 BETA(I)=T2*BETA(I)
C     COMPUTE ERROR IN LEAST SQUARES FIT
      CALL CHECK(BETA,F,Z,DELT,NP1,KP1)
   29 CONTINUE
      ERR=ERR/SSS
      PRINT 155,(L,BETA(L),L=1,KP1)
C     COMPUTE SK VECTOR
      DO 300 I = 1 , KSP1
        IM1 = I - 1
        CALL DKPHI ( KP1 , IM1 , DPHI )
        SK(I) = 0.
        DO 301 IX1 = 1 , KP1
          SK(I) = SK(I) + BETA(IX1) * DPHI(IX1)
  301 CONTINUE
  300 CONTINUE
      DO 95 I=1,KSP1
      IM1=I-1
   95 SK(I)=SK(I)/(TSCALE**IM1)
  402 FORMAT( 1H1,/,( 5X , 3HSK(,I2,3H)= ,E15.8  , 5X ,4HRSK(,I2,3H)= ,
     1         E15.8) )
      PRINT 403 , N , K , T1 , DEL
  403 FORMAT (1H1,//,9X,2HN=,I4,9X,2HK=,I4,9X,14HTIME INTERVAL=,F8.5 ,
     1         9X,15HTIME INCREMENT=,F8.5 )
  404 FORMAT(///,(5X,3HSK(,I2,3H)= ,D15.8))
      PRINT 404, (I,SK(I),I=1,KSP1)
      PRINT 550, P, ERR
      GO TO 203
      IF ( COUNT ) 510 , 520 , 530
  510 E(1) = ERR
      D(1) = P
      PRINT 550, P, ERR
  550 FORMAT(//5X, 6H   P = , E15.8,5X, 6H ERR = , E15.8 )
      P = 1.1*P
      COUNT = COUNT + 1
      GO TO 551
  520 E(2) = ERR
      D(2) = P
      COUNT = COUNT + 1
      PRINT 550, P, ERR
      IF ( E(1) .LT. E(2) )  GO TO 540
      IMIN = 2
      P = 1.1*P
      GO TO 551
  540 IMIN = 1
      P = .8*P
```

```
      GO TO 551
  530 PRINT 550 ,    P   , ERR
      IF ( ERR .GT. E(IMAX) ) GO TO 203
      E(IMAX) = ERR
      D(IMAX) = P
        IDMIN = 1
       DMIN   =   D(1)
        IDMAX = 1
       DMAX   =   D(1)
       DO  580  IX1   =    2,3
       IF  (D(IX1) .GT . DMIN )  GO TO 581
        IDMIN   = IX1
       DMIN   =   D(IX1)
  581   CONTINUE
       IF  (D(IX1) .LT. DMAX ) GO TO 580
       IDMAX    =   IX1
       DMAX   =   D(IX1)
  580   CONTINUE
       IMIN   =    1
       EMIN   =   E(1)
       IMAX = 1
        EMAX   =   E(1)
        DO  582   IX1   =   2,3
        IF   ( E(IX1) .GT. EMIN)    GO TO 583
       IMIN = IX1
       EMIN    =    E(IX1)
  583   CONTINUE
       IF (E(IX1) .LT. EMAX ) GO TO 582
        IMAX = IX1
       EMAX   = E(IX1)
  582  CONTINUE
       RELERR = ( EMAX - EMIN )/ EMIN
       IF ( RELERR .LT. 0.05 ) GO TO 203
      X = -E(1) * ( D(2)**2 - D(3)**2 ) + E(2)* ( D(1)**2 - D(3)**2 )
     1   -E(3) * ( D(1)**2 - D(2)**2 )
      Y =           ( E(1) * ( D(2) - D(3) ) - E(2) * ( D(1) - D(3) ) +
     1              E(3) * ( D(1) - D(2) ) )
      X1 =      ( ( D(1)-D(2)) * ( D(1)-D(3) ) * ( D(2)-D(3) ) )
      IF ( (Y/X1) .GE. 0 ) GO TO 552
      PRINT 553
  553 FORMAT(28H  SECOND DERIVITIVE NEGATIVE   )
      GO TO 203
  552 CONTINUE
      Y = -.5*X / Y
      IF ( (     DMIN  .LT. Y ) .AND. ( Y .LT.     DMAX   ) ) GO TO 560
        X =   ( -X*D(2)/Y     +    X )/X1
      IF ( X .GE. 0. ) GO TO 570
      P = 1.1*DMAX
      GO TO 551
  570 P = .9*DMIN
      GO TO 551
  560 P = Y
      GO TO 551
  203 CONTINUE
      READ 400 , NCASE
  400 FORMAT(I2)
      IF ( NCASE .EQ. 0 ) GO TO 401
```

```
STOP
END
```

```
'IT FOR SUB1
      SUBROUTINE CHECK(BETA,F,Z,DELT,NP1,KP1)
      COMMON  /NORM/  ERR,INTST
      DIMENSION BETA(20),F(801,20),Z(1603)
      DOUBLE PRECISION BETA,ERROR,TERM
      ERROR=0.0
      DO 1  I = INTST,NP1
      TERM=0.0
      L1=I+1-INTST
      DO 2 K=1,KP1
    2 TERM=TERM+F(L1,K)*BETA(K)
      TERM=DELT*TERM
      ERROR = ERROR + ( Z(I) - TERM ) ** 2
    1 CONTINUE
      ERROR = SQRT(ERROR/(NP1-INTST))
      ERR = ERROR
      RETURN
      END
```

04/29/68   80/80 LIST
'IT FOR SUB2

```
      SUBROUTINE GENIO (NP,H,DU,DZ,STDEV,SSS,INTST)
C     DU              GENERATED INPUT DATA
C     H               TIME INCREMENT USED IN GENERATING DATA
C                     NP=1 CORRESPONDS TO THE 0-TH POINT
C     NP              NP+1 DENOTES THE NO. OF POINTS DESIRED
C     SUBROUTINE TO GENERATE SYSTEM INPUT-OUTPUT DATA
C     DZ              GENERATED OUTPUT DATA
      DIMENSION DU(1603),DZ(1603)
       DIMENSION AS(15),AC(15)
      DIMENSION S(15),C(15)
      SS = 0
      FPER=(NP+1-INTST)
      FPER=FPER*H/12.
      FFREQ=2.*3.1415926/FPER
      AC0=.25
      IORFOS=10
      DO 5 K=1,IORFOS
      XK = K
      XOM=FFREQ*XK
C     FOR 1/(S+1)**2
      DENOM = XOM**4.+2.*XOM**2.+1.
      AS(K)=(1.-XOM**2.)/DENOM
      AC(K)=-2.*XOM/DENOM
      SS = SS + AS(K)**2. + AC(K)**2.
    5 CONTINUE
      SSS = SQRT(SS)
      PRINT 100 , SSS , STDEV
  100 FORMAT(///,5X,14HOUTPUT NORM = , E15.8,///,5X,25HNOISE TO SIGNAL
     1RATIO = ,E15.8,///)
  101 FORMAT(///,5X,13HNOISE MEAN = ,E15.8,///,5X,17HNOISE ST. DEV. = ,
     1E15.8,///)
  102 FORMAT(///,5X,5HNOISE,//,(I5,2X,E15.8))
      SD = SSS*STDEV
      IF (NANA.EQ.381) GO TO 6
      IA=1
      KA=2**18+3
      NANA=381
      CA = 2.**35
    6 CONTINUE
      PRINT 997,IA
  997 FORMAT(5X,48HSTARTING   INTEGER   FOR   THE   NOISE   BURST =,I11)
      IB=NP+2-INTST
      IB=IB/2
      DO 4 KB=1,IB
      IA = ABS(IA*KA)
      T1=FLOAT(IA)/CA
      IA = ABS(IA*KA)
      T2=FLOAT(IA)/CA
      T9=SQRT(-2.*ALOG(T1))
      T8=6.28318531*T2
      KC=2*KB+INTST-1
      DZ(KC)=SD*T9*SIN(T8)
      DZ(KC-1)=SD*T9*COS(T8)
    4 CONTINUE
      XMEAN = 0.
      NP1 = NP + 1
```

```
      NPP1=NP+1
      DZ(1) = SD
      X=0.
      Y=0.
      GO TO 3
    3 CONTINUE
      PRINT 102, (I,DZ(I),I=800,900)
      V=0.
      W=0.
      DO 27 I=INTST,NPP1
      V=V+DZ(I)
      W=W+DZ(I)**2.
   27 CONTINUE
      AVE=NPP1-INTST+1
      V=V/AVE
      W=SQRT(W/AVE)
      PRINT 101, V,W
      DO 28 I=INTST,NPP1
      DZ(I)=DZ(I)-V
   28 CONTINUE
      DO 1 I=1,NPP1
      TI = (I-1)*H
      DO 10    K = 1, 10
      XK = K
      XOM=FFREQ*XK
      IF(I.LT.INTST) GO TO 10
      C(K)=COS(XOM*TI)
   10 S(K)=SIN(XOM*TI)
      DU(I) = S(1)+S(2)+S(3)-S(4)+S(5)-S(6)+S(7)-S(8)+S(9)-S(10)+.25
      X=X+DZ(I)
      Y=Y+DZ(I)**2.
      IF(I.LT.INTST)  GO TO 1
      DZ(I) = AS(1)*S(1)+AS(2)*S(2)+AS(3)*S(3)-AS(4)*S(4)+AS(5)*S(5)
     1 -AS(6)*S(6)+AS(7)*S(7)-AS(8)*S(8)+AS(9)*S(9)-AS(10)*S(10)
     2 +AC(1)*C(1)+AC(2)*C(2)+AC(3)*C(3)-AC(4)*C(4)+AC(5)*C(5)
     3 -AC(6)*C(6)+AC(7)*C(7)-AC(8)*C(8)+AC(9)*C(9)-AC(10)*C(10)+DZ(I)
     4+AC0
    1 CONTINUE
      X=X/NPP1
      Y=Y/NPP1
      Y=SQRT(Y)
      PRINT 101, X,Y
      RETURN
      END
```

```
'IT FOR SUB3
      SUBROUTINE FKSUB
      COMMON/TEN/U(1603)
      COMMON  /NORM/  ERR,INTST
      COMMON/FK/F(801,20),PHI(2,1603),DELT,NP1,KP1,FP(801,20)
      DOUBLE PRECISION PHI
      DO 1 I=1,NP1
      T=(I-1)*DELT
      PHI(1,I) = PHI1(T)
      PHI(2,I) = PHI2(T)
    1 CONTINUE
      L1=NP1-INTST+1
      L1=L1/2
      DO 2 K=1,KP1
      KM1=K-1
      DO 3 L=1,L1
      J1=INTST+2*L-4
      C1=0
      C2=0
      B1=0
      B2=0
      DO 4 J2=3,J1,2
      J3=J1-J2+3
      C1=C1+PHI(1,J2)*U(J3)
      C2=C2+PHI(1,J2)*U(J3+1)
      B1=B1+PHI(1,J2+1)*U(J3-1)
      B2=B2+PHI(1,J2+1)*U(J3)
    4 CONTINUE
      J4=2*L-1
      J5=2*L
      F(J4,K)=(PHI(1,1)*U(J1+2)+PHI(1,J1+2)*U(1)+4.*B1+
     12.*C1+4.*PHI(1,2)*U(J1+1))/3.
      FP(J4,K)=F(J4,K)
      F(J5,K)=(PHI(1,1)*U(J1+3)+PHI(1,J1+2)*U(2)+4.*B2+2.*C2
     1+4.*PHI(1,2)*U(J1+2))/3.
     2+(5.*PHI(1,J1+3)*U(1)+8.*PHI(1,J1+2)*U(2)-PHI(1,J1+1)*U(3))/12.
      FP(J5,K)=F(J5,K)
    3 CONTINUE
      N = K
      DO 5 IJ = 1 , NP1
        T = (IJ - 1 ) * DELT
        CALL RCSN ( PHI , IJ  , N , T )
    5 CONTINUE
    2 CONTINUE
  100 FORMAT(5X,6D20.8)
      RETURN
      END
```

```
'IT FOR SUB4
      SUBROUTINE GINV2 (A,U,AFLAG,ATEMP,MR,NR,NC)
C     THIS ROUTINE CALCULATES THE GENERALIZED INVERSE OF A
C          AND STORES THE TRANSPOSE OF IT IN A
C       MR=FIRST DIMENSION NO. OF A.
C       NR = NO. ROWS IN A
C       NC = NO. COLUMNS IN A
C       U IS THE BOOKKEEPING MATRIX.
C       AFLAG AND ATEMP ARE TEMPORARY WORKING STORAGE.
      DIMENSION A(801,20),U(20,20),AFLAG(25),ATEMP(25)
      DO 10 I=1,NC
      DO 5 J = 1 , NC
    5 U(I,J) = 0.0
   10 U(I,I) = 1.0
      FAC = DOT(MR,NR,A,1,1)
      FAC = 1.0/SQRT(FAC)
      DO 15 I=1,NR
   15 A(I,1)=A(I,1)*FAC
      DO 20 I=1,NC
   20 U(I,1)=U(I,1)*FAC
      AFLAG(1)=1.0
C     DEPENDENT COL TOLERANCE FOR N BIT FLOATING POINT FRACTION
      N=27
      TOL = (10. * 0.5**N)**2
      DO100 J=2,NC
      DOT1 = DOT(MR,NR,A,J,J)
      JM1 = J-1
      DO 50 L=1,2
      DO 30 K=1,JM1
   30 ATEMP(K)=DOT(MR,NR,A,J,K)
      DO 45 K=1,JM1
      DO 35 I=1,NR
   35 A(I,J)=A(I,J)-ATEMP(K)*A(I,K)*AFLAG(K)
      DO 40 I=1,NC
   40 U(I,J)=U(I,J)-ATEMP(K)*U(I,K)
   45 CONTINUE
   50 CONTINUE
      DOT2 = DOT(MR,NR,A,J,J)
      IF((DOT2/DOT1)-TOL) 55,55,70
   55 DO 60 I=1,JM1
      ATEMP(I)=0.0
      DO 60 K=1,I
   60 ATEMP(I)=ATEMP(I)+U(K,I)*U(K,J)
      DO 65 I=1,NR
      A(I,J)=0.0
      DO 65 K=1,JM1
   65 A(I,J)=A(I,J)-A(I,K)*ATEMP(K)*AFLAG(K)
      AFLAG(J)=0.0
      FAC = DOT(NC,NC,U,J,J)
      FAC = 1.0/SQRT(FAC)
      GO TO 75
   70 AFLAG(J) = 1.0
      FAC= 1.0/SQRT(DOT2)
   75 DO 80 I=1,NR
   80 A(I,J) = A(I,J)*FAC
      DO 85 I=1,NC
   85 U(I,J) = U(I,J)*FAC
```

```
100 CONTINUE
    DO 130 J=1,NC
    DO 130 I=1,NR
    FAC=0.0
    DO 120  K=J,NC
120 FAC=FAC+A(I,K)*U(J,K)
130 A(I,J) = FAC
    RANK=0
    DO 132 J=1,NC
    RANK=RANK+AFLAG(J)
132 CONTINUE
    PRINT 133, RANK
133 FORMAT(//,3X,7HRANK = ,1E15.8)
    RETURN
    END
```

```
'IT FOR SUB5
      FUNCTION DOT(MR,NR,A,JC,KC)
C     COMPUTES THE INNER PRODUCT OF COLUMNS JC AND KC OF MATRIX A
      DIMENSION A(801,20)
      DOT = 0.0
      DO 5 I = 1,NR
    5 DOT= DOT + A(I,JC)* A(I,KC)
      RETURN
      END
```

```
'IT FOR SUB6
      SUBROUTINE DKPHI( KP1 , I , DPHI )
      DIMENSION DPHI(1)
      COMMON /SCALE/ P
C        THIS ROUTINE CALCULATES THE I DERIVATIVE OF THE (J-1)
C        LAGUERRE FUNCTION AND STORES IT IN DKPHI(J)
      DO 1 J = 1 , KP1
         IJ = J - 1
         IP1 = I + 1
         SUM = 0.
          DO 2 K = 1 , IP1
            IX2 = K - 1
            IX4 = I - IX2
            TERM = BCOF(I,IX2) * BCOF(IJ,IX4) * ( 2. ** (-K+1) )
            SUM = SUM + TERM
C
    2     CONTINUE
         IX = ( IJ + I ) / 2
         IX = IX * 2
         SIGN = - 1.
         IF ( IX .EQ. ( IJ+I) ) SIGN = 1.
         DPHI(J) = SIGN * SQRT ( 2. * P ) * ( 2. * P ) * * I   * SUM
    1 CONTINUE
C
      RETURN
      END
```

```
'IT FOR SUB7
      FUNCTION BCOF ( I , J )
      XI = I
      XJ = J
      XIMJ = I-J
      P = 1.
      IF ( J .GT. I ) GO TO 3
      IF ( I .EQ. 0 ) GO TO 2
      IF ( J .EQ. 0 ) XJ = 1.
      IF ( I .EQ. 0 ) XI = 1.
      IF ( (I-J) .EQ. 0 ) XIMJ = 1.
        DO 1 K = 1 , I
          P = P * ( XI / (XIMJ * XJ) )
          IF ( (I-K) .GT. 0 )  XI = XI - 1.
          IF ( (J-K) .GT. 0 )  XJ = XJ - 1.
          IF ( (I-J-K) .GT. 0 )  XIMJ = XIMJ - 1.
    1   CONTINUE
    2 CONTINUE
      BCOF = P
      RETURN
    3 CONTINUE
      BCOF = 0.
      RETURN
      END
```

04/29/68 80/80 LIST

'IT FOR SUB8

```
      FUNCTION PHI1(T)
C     *****  DEFINE THE FIRST ORTHOGONAL FUNCTION *****
      COMMON /SCALE/ P
      PHI1 = SQRT( 2. * P)  *  EXP( -P * T )
      RETURN
      END
```

```
'IT FOR SUB9
      FUNCTION PHI2(T)
C     ***** THIS SUBROUTINE DEFINES THE SECOND ORTHOGONAL FUNCTION ****
      COMMON /SCALE/ P
      PHI2 = SQRT( 2. * P )  * ( 2.*P*T - 1. ) * EXP( -P * T )
      RETURN
      END
```

```
'IT FOR SUB10
      SUBROUTINE RCSN ( PHI , I    , N , T )
C     ***** THIS SUB. DEFINES THE RECURSION FORMULA FOR THE GENERATION
C     ***** OF HIGHER ORDER ORTHOG. FNS   *****
      COMMON /SCALE/ P
      DIMENSION PHI(2,1603)
      DOUBLE PRECISION PHI,TEMP
      XN = N
        TEMP =(( 2.*P*T - 2.*XN - 1.) * PHI(2,I) - XN * PHI(1,I) )
     1          / ( XN + 1. )
        PHI(1,I) = PHI(2,I)
        PHI(2,I) = TEMP
      RETURN
      END
```

Appendix B

The Subroutine Micare

SUBROUTINE MICARE (SSUBR, N, TOLI, NST, H3, IDERK)

I. Purpose

We are given the N vector $S = \{s_k\}$ and wish to find an r-dimensional, constant linear dynamical system, $[c, \Phi, \gamma]$ in companion form, with $c = [1, 0, \ldots, 0]$ such that, approximately,

$$c\Phi^{k-1} = s_k \qquad k = 1, \ldots, N \ .$$

This is the primary task of subroutine MICARE - the implementation of the B. L. Ho procedure.

In addition, however, it calls subroutine CPC (see MSG PD-67-102) in order to obtain an r-dimensional, constant linear dynamical system $[c, A, b]$ in companion form, with $c = [1, 0, \ldots, 0]$ such that

$$c\ e^{k\sigma A}\ b = s_{k+1}, \qquad k = 0, \ldots, N-1.$$

Essentially CPC finds the continuous-time system $[c, A, b]$ from which the discrete system $[c, \Phi, \gamma]$ arises. This is under the assumption that the input vector s is the discretized (at interval σ) time history of the impulse response of some linear constant dynamical system.

It can happen that the vector s contains the leading coefficients of the expansion in powers of $1/s$ of a transfer function (the laplace transform of the impulse response). This is, in fact, the originally planned mode of operation for the procedure. In such a case the call to CPC is superfluous. The application for which MICARE was written usually requires the use of CPC, however, and furthermore CPC provides the eigenvalues of Φ, so no provision was made for avoiding the call to CPC.

II. Operations - Calling Sequence

The vector s is entered in the array SSUBR: N, the dimension of s, is given in N; TOL1 is a zero tolerance used in subroutine RAKAR, for details see MSG PD-67-103; NST gives the starting dimension of the square Hankel matrix

$$H = \begin{bmatrix} s_1 & s_2 & s_3 & - \\ s_2 & s_3 & s_4 & - \\ - & - & - & \end{bmatrix}$$

used in the B. L. Ho procedure (see Procedure and Mathematical Analysis below); the discretizing interval σ is given in H3; the maximum rank allowed, r, is given in IDERK.

Language is FORTRAN IV, no tapes are used.

The dimensions in MICARE and its required subroutines allow for N to be 50, the Hankel matrix to have dimension 20, and r (contained in IDERK) to be 15.

III. Printout

A fair amount of intermediate printout is given because it was required in the original application.

The zero tolerance, TOL1, is printed.

The input vector S is printed.

The dimension of the Hankel matrix which was used for computing the realization is printed as KM1.

The vector $\hat{S}$ as computed from the realization is printed as ESTIMATED VECTOR.

If a larger-than-expected error between $\hat{S}$ and S is encountered, see VI and VII for details, a print IFLAG is made indicating the component in which the difference occurred.

After the system is put in companion form, $\hat{S}$ is recomputed to establish the accuracy of the similarity transformation.

The output coefficients $c = [1, 0, \ldots, 0]$ are printed.

The system matrix Φ in companion form is printed.

The input coefficients γ are printed.

The program then transfers control to CPC which itself generates output (see MSG PD-67-104) terminating in the logarithm system $[c, A, b]$ in companion form.

Control returns to MICARE which, if IFLAG was not printed, will print a statement that the realization was successful. If IFLAG was printed it returns to increase the order of the Hankel matrix. If this is not possible, a message is printed. If the matrix was enlarged because of an IFLAG print but the rank did not increase, the message,

NIX EQUALS ONE AND RANK EQUALS RK

will be printed.

IV. Subroutines

The matrix decomposition routine RAKAR (see MSG PD-67-102)

The system logarithm routine CPC (see MSG PD-67-103).

The $\hat{S}$ generating routine SVCAP.

The inner product function DOT (for RAKAR).

The polynomial root solver MULLER.

The inversion routines MATINV and MINV.

V. Restrictions and Comments

None

VI. Procedure

Circled numbers, e.g., 26 are external formula numbers in the FORTRAN source program.

Set the maximum dimension of the Hankel matrix to N/2.

Put NST $\rightarrow$ K, 0 $\rightarrow$ NIX, 0 $\rightarrow$ RK, 0 $\rightarrow$ IFLAG.

$\downarrow$

Θ If IFLAG $\neq$ 0, go to 60 .

8 Set up the K-dimensional Hankel matrix H and call RAKAR for the rank RANK. If RANK = RK, go to 18 .

$\downarrow$

25 Put 0 $\rightarrow$ NIX, RANK $\rightarrow$ RK.

Using other output from RAKAR, define T_R and T_L such that

$$T_L \; H \; T_R = I_r$$

where I_r is a RANK-order identity matrix. If the order K of H is not maximum, go to 26 ; otherwise print

RANK NOT STABILIZED BUT WE HAVE REACHED MAXIMUM DIMENSION.

Then set K + 1 $\rightarrow$ K and go to 35 .

26 K + 1 $\rightarrow$ K

If K $\leq$ max dimension, go to Θ .

RETURN

* * * * * * * *

18 If NIX $\neq$ 0, go to 19

$\downarrow$

If RK = 0, go to 30

$\downarrow$

35 Put $K - 1$ KM1 and print this number which is the dimension of the H used for the representation.

Set up the matrix and vector

$$H^* = \begin{bmatrix} s_2 & s_3 & s_4 & - \\ s_3 & s_4 & s_5 & - \\ - & - & - & \end{bmatrix}, \quad h = \begin{bmatrix} s_1 \\ s_2 \\ \cdot \\ \cdot \\ \cdot \\ s_{KM1} \end{bmatrix}$$

and compute

$$\Phi^* = T_L \, H^* \, T_R \quad ,$$

$$c^* = h' \, T_R \, ,$$

and

$$\gamma^* = T_L \, h \quad .$$

$\downarrow$

Call SVCAP to produce $\hat{S} = \{\hat{s}_j\}$ where $\hat{s}_{j+1} = c^* \, \Phi^{*j} \, b^*$, $j = 0, \ldots, N-1$.

Let

$$\text{EPSIL} = \max_{1 \leq j \leq 2K-1} \left(10^{-7}, \ \frac{2\,\hat{s}_j - s_j}{1 + s_j} \right) .$$

If RANK is governed only by a small TOLI, that is, not constrained by IDERK, then EPSIL should be reasonably small.

$\downarrow$

We now check, for $j = 2K, \ldots, N-1$, if $\left(\text{EPSIL} - \dfrac{|\hat{s}_j - s_j|}{1 + |s_j|} \right)$ is always positive. If it is not, then for the first index, L, for which it is negative, we set

$$IFLAG = \frac{L + 2}{2} .$$

This IFLAG is the dimension of the smallest Hankel matrix which will include the offending term and thus give a more accurate representation, either in terms of rank, if that is free, or in more evenly distributed error.

Notice a print of IFLAG indicates an error in matching the 2*IFLAG-1 or 2*IFLAG-2 element of S.

↓

Whether or not IFLAG is printed we now prepare to put the system in companion form. If the system order is one, we ship this transformation, going to 38 .

↓

Because the system is a minimal realization of a transfer function, it is completely controllable and completely observable.

We thereform form the matrix

$$T = \begin{bmatrix} c^* \\ c^* \Phi^* \\ - \\ - \\ - \\ c^* \Phi^{*RANK-1} \end{bmatrix}$$

invert it and form

$$\Phi = T \Phi^* T^{-1} ,$$

$$c = c^* T^{-1} ,$$

and

$$\gamma = T \gamma^* .$$

(If T is singular, a message to that effect is printed and we RETURN.)

We compute $\hat{S}$ by the companion form system and print it.

↓

38 We print c, Φ, and γ.

↓

The matrix Φ is put exactly in companion form by putting true zeroes and ones in the proper places. The same is done for $c = (1, 0, \ldots, 0)$.

↓

CPC is called.

↓

(If K = max dimension, then we have previously printed a maximum dimension message so we RETURN.)

↓

If IFLAG $\neq 0$, go to 26.

↓

300 Print message that realization is good and RETURN

* * * * * * * * * *

60 IFLAG is the desired dimension of H. If IFLAG − K > 0, go to 26.

0 → IFLAG, 1 → NIX

* * * * * * * * * * * * * *

19 This path is taken if an error occurred in approximating S by $\hat{S}$ (IFLAG $\neq$ 0) but when H was increased to the proper dimension the rank did not increase. This could have occurred because IDERK constrained the rank or simply because in the context of higher dimensional vectors with larger norms, the error simply was not significant. Go to 25 .

* * * * * * * *

30 This path is taken if the rank (therefore H) is zero. If K is not yet at the max dimension, go to 26 .

↓

Print NULL MATRIX and then RETURN.

* * * * * * * *

VII. Mathematical Analysis

1. The B. L. Ho Procedure.

Definition: An infinite matrix is said to have rank r if the maximum rank of any finite submatrix is r.

Proposition 1: Let $[c, A, b]$ be an n^{th} order c.c and c.o. stationary system, with impulse response $c\phi(t)b$. Denote $\phi(\delta)$ by ϕ. Let $H = [h_{ij}]$, where

$$h_{ij} = c\phi((i+j-2)\delta)b = c\phi^{i+j-2}b \ ,$$

be an infinite order matrix. Then rank $H = n$.

Proposition 2: Let $[c, A, b]$ be an n^{th} order c.c. and c.o. stationary system with impulse response $c\phi(t)b = f(t)$. Represent $f(t)$ in its taylor's series expansion

$$\sum_{k=0}^{\infty} \frac{a_k}{k!} t^k \ .$$

Let $H = [h_{ij}]$, where

$$H_{ij} = a_{i+j-2} \, .$$

Then rank $H = n$.

<u>Proof</u>: Clearly $a_k = cA^k b$, since $a_k = f^{(k)}(0)$. Therefore

$$h_{ij} = cA^{i+j-2}b.$$

Also the matrices

$$W_\phi = [b, \phi b, \cdots , \phi^{n-1}b]$$

and

$$W_A = [b, Ab, \cdots , A^{n-1}b]$$

are both nonsingular by complete controllability, as are the comparable observability matrices. These remarks reduce the two propositions to one.

We shall prove proposition 2.

The $n \times m$ matrix $(m \geq n)$

$$W = [b, Ab, A^2b, \cdots]$$

has rank n, as does the $m \times n$ matrix M,

$$M' = [c', A'c', A'^2c', \cdots] \, .$$

Let ν_{ij} denote the elements of MN. Then

$$\nu_{ij} = cA^{i-1}A^{j-1}b = cA^{i+j-2}b \, .$$

That is $MN = H$.

Sylvester's inequality states that

$$\text{rank } M + \text{rank } N - n \leq \text{rank } MN \leq \min(\text{rank } M, \text{ rank } N) ,$$

in this case

$$n \leq \text{rank } MN \leq n .$$

Therefore for all $m \geq n$, rank $H = n$. †

<u>Remark</u>: Let $F(s) = \mathcal{L} f(t) = \frac{p(s)}{q(s)}$.

Then deg p < deg q. If $F(s)$ is expanded in powers of s^{-1},

$$F(s) = \sum_{k=0}^{\infty} \frac{a_k}{s^{k+1}} ,$$

then the a_k are the previously defined taylor coefficients of $f(t)$. This follows, of course, from the fact that

$$\mathcal{L} t^k = \frac{k!}{s^k} .$$

<u>Proposition 3</u>: Let $h = [h_{ij}]$ be an (infinite) hankel matrix (i.e. $h_{ij} = v_{i+j-2}$ for some sequence $\{v_k\}$) with n the maximum rank of any submatrix. Then there exists a triple $[c, A, b]$ such that

$$h_{ij} = cA^{i+j-2}b = v_{i+j-2} .$$

<u>Lemma</u>: For such an H, the first n rows $\{R_i\}_1^n$ are linearly independent.

Proof of Lemma: Since every (n+1)-rowed submatrix has determinant zero, the first n+1 rows are linearly dependent. Therefore there exists a number $r \leq n$ such that $R_1, R_2, \cdots, R_r$ are linearly independent and

$$R_{r+1} = \sum_{k=0}^{r-1} a_k R_{k+1} .$$

From the cyclic character of a Hankel matrix, we see that

$$R_{h+q+1} = \sum_{k=0}^{r-1} a_k R_{k+q+1} ,$$

and therefore every row can be expressed interms of the first r rows. It follows that $r = n$. †

Proof of Proposition 3: Let $[a_o, a_1, \cdots, a_{n-1}]$ be the vector defined in the proof of the lemma. Then

$$v_k = cA^k b$$

where $$c = (1, 0, \cdots, 0)$$

$$b' = (v_o, v_1, \cdots, v_{n-1})$$

and A is the companion-form matrix with last row

$$[a_o, a_1, \cdots, a_{n-1}] .$$ †

Our conclusion from these three propositions is that a hankel matrix has finite rank n iff its sequence is generated by an n^{th} order dynamical system.

2. Computation.

Let H be a hankel matrix of rank n and let S be its first n^{th} order principal submatrix

$$S = \begin{bmatrix} v_o & \cdots & v_{n-1} \\ & \cdots & \\ v_{n-1} & \cdots & v_{2n-2} \end{bmatrix} .$$

By an extension of the lemma, this has rank n.

Compute nonsingular matrices L and R such that

$$LSR = I_n.$$

It follows that $S^{-1} = RL$

Let

$$S^* = \begin{bmatrix} v_1 & \cdots & v_n \\ & \cdots & \\ v_n & \cdots & v_{2n-1} \end{bmatrix}$$

denote the second n^{th} order principal submatrix of H, and let

$$b' = [v_o, v_1, \cdots, v_{n-1}] .$$

We know that

$$S^* = AS$$

where A is the matrix defined in proving proposition 3.

Compute

$$c^* = b'R ,$$

$$b^* = Lb ,$$

and $A^* = LS^*R = LASR.$ Then

$$c^*b^* = b'RLb = (1, 0, \cdots, 0)\ b = v_o$$

and

$$c^*A^{*k}b^* = b'R(LASR)^kLb = b'RL(ASRL)^kb = cA^kb .$$

To provide additional smoothing we compute with

$$S = \begin{bmatrix} v_o & \cdots & v_{m-1} \\ & \cdots & \\ v_{m-1} & \cdots & v_{2m-2} \end{bmatrix}$$

The first m^{th} order principal submatrix of H, where m is much larger than $n = \text{rank } H$.

We find matrices L and R of rank n such that

$$LSR = I_n$$

and

$$SR = L' .$$

<u>Lemma</u>: $SRLS = S$.

Proof: Since $SR = L'$ and rank L = rank S, L is nonsingular on range S, therefore the fact that

$$L(SRLS - S) = LS - LS = 0 ,$$

implies that $SRLS - S = 0$.

Let

$$S^* = \begin{bmatrix} v_1 & \cdots & v_{m+1} \\ & \cdots & \\ v_{m+1} & \cdots & v_{2m-1} \end{bmatrix}$$

be the second m^{th} order principal submatrix of H.

We can define an m by m matrix $\hat{A}$ such that $S^* = AS$. In the 3×3 case, if

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ a & b & c \end{bmatrix}$$

then

$$\hat{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & . \\ 0 & 0 & 1 & 0 & 0 & . \\ a & b & c & 0 & 0 & . \\ 0 & a & b & c & 0 & . \\ 0 & 0 & a & b & c & . \\ . & . & . & . & . & \end{bmatrix}$$

to the size required.

The important thing is that b is the first column of S and $\hat{A}^k b$ is b_{k+1}, the $(k+1)^{st}$ column of S (or the first m rows of the $(k+1)^{st}$ column of H if $k \geq m$).

We compute

$$c^* = b'R \; ,$$

$$b^* = Lb \; ,$$

$$A^* = LS^*R = L\hat{A}SR \; .$$

Then

$$c^*b^* = b'RLb = v_o$$

by the lemma.

$$c^*A^*b^* = b'RL\hat{A}SRLb = b'RL\hat{A}b$$

by the lemma.

But $\hat{A}b$ is the second column of S so $b'RL\hat{A}b = v_1$, again by using the lemma.

In general

$$c^*A^{*k}b^* = b'R(L\hat{A}SR)^k Lb = b'RL(\hat{A}SRL)^k b.$$

By induction we can show that

$$(\hat{A}SRL)^k b = b_{k+1} .$$

Since $b_{k+1} \in$ range (S), it follows by the lemma that

$$b' RL b_{k+1} = v_k . \qquad \dagger$$

Remark: Notice that RL need not be the generalized inverse of S but must satisfy only

$$SRLS = S.$$

3. Mechanization.

Starting with H of dimension NST we find matrices T_L, T_R such that

$$T_L \, H \, T_R = I$$

where I is an n-dimensional identity, T_L and T_R are saved.

Increasing the dimension of H by one we replace T_L and T_R by their new values if the rank increases.

If the rank is unchanged, either because of the constraint IDERK or because the rank is the same within the tolerance TOL1, we use the T_L and T_R from the previous dimension KM1 as follows.

The matrix H^* of dimension KM1 is formed

$$H^* = \begin{bmatrix} s_2 & s_3 & s_4 & \cdot \\ s_3 & s_4 & s_5 & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} .$$

Then the system matrix is $\phi^* = T_L \, H^* \, T_R$, the output vector is $c^* = [s_1, \cdots , s_{KM1}] T_R$, and the input vector is

Appendix C

Program CPC

SUBROUTINE CPC (S, IRANK, B, C, DT)

Purpose: We are given the $n \times n$ companion form matrix ϕ, vectors G and H, and a time increment δ. We wish to find an $n \times n$ companion form matrix A and vectors B and C $(C = [1, 0, 0, \ldots 0])$ such that

$$Ce^{k\delta A}B = H\phi^k G, \quad k = 0,1,\ldots .$$

Basically we wish to find the logarithm of ϕ.

Operations - Calling Sequence: The matrix ϕ is entered in the array S and the output matrix A will be returned in the array S. The dimension of ϕ is contained in IRANK. The vectors G and H are in B and C respectively and the output vectors B and C will be in the arrays B and C. DT contains δ.

The dimensions in CPC, MULLER and MATINV are fixed at 15 except for the vector of coefficients of the characteristic polynomial which is fixed at 16.

Language is FORTRAN IV, no tapes are used.

Printout: A fair amount of intermediate output is given because it was required in the original application.

The roots of the characteristic polynomial are printed.

The number of complex roots is printed.

The real diagonal form $T^{-1}\phi T$ of ϕ, as computed, is printed.

The continuous system is printed in real diagonal form and finally the continuous system (C, A, B) is printed in companion form.

Subroutines: A polynomial root finder, MULLER is called once.

A matrix inversion routine, MATINV is called twice.

Restrictions and Commentary:

1) Naturally ϕ must be nonsingular.

2) ϕ cannot have repeated eigenvalues. In practice this is not a very serious restriction. Numerical difficulties may occur when roots are close to each other.

3) Early in the program, eigenvalues $\lambda = x + iy$ are assumed to be real and positive if they satisfy

$$\frac{|y|}{10^{-7} + |x|} < 10^{-7}$$

Theoretically this is a vulnerable point. If there is a complex pair of ϕ with small imaginary part, trouble can occur. However, this is essentially covered by the restriction that roots must be distinct. Perhaps more important, a complex pair in F can, for proper values of the time increment, give rise to a coincident pair of negative eigenvalues of ϕ. However, we do not expect this to occur because good engineering practice will dictate that the time increment used to generate ϕ will be selected less than half the natural period.

Besides which the condition is highly improbable under any circumstances.

4) This program, because of the application which evoked it, assumes that the pair $[H, \phi]$ is completely observable. This is clear from the output form of C and A.

Procedure: Since ϕ is given in companion form, the characteristic polynomial is immediately available. This is factored to obtain the eigenvalues of ϕ. If the eigenvalue $\lambda = x + iy$ satisfies

$$\frac{|y|}{10^{-7} + |x|} < 10^{-7}$$

the eigenvalue is taken as real and positive, otherwise as complex.

We set up a complex n-vector with the complex roots first and the real roots last.

The eigenvalues of ϕ are printed. The number of complex roots is printed.

The generalized Vandermonde matrix T is constructed which transforms ϕ to its real diagonal form, R.

T is inverted to form T^{-1}.

HT and ϕT are formed.

$T^{-1}G$ and $T^{-1}\phi T$ are formed.

$T^{-1}\phi T$ is printed. The computation and subsequent printout of $T^{-1}\phi T$ is done purely as a numerical check since $T^{-1}\phi T$ will be assumed to have the correct real diagonal form R and its computed value destroyed after printing.

$M = \log R$ is constructed and printed. Following this, the matrix

$$S = \begin{bmatrix} HT \\ HTM \\ \vdots \\ HTM^{n-1} \end{bmatrix}$$

is formed, and finally the desired matrices

$$C = HTS^{-1}$$

$$A = SMS^{-1}$$

$$B = ST^{-1}G$$

are printed.

Mathematical analysis:

1) Real diagonal form and generalized Vandermonde:

If a matrix ϕ has only real eigenvalues, its real diagonal form Λ is its diagonal form and the matrix T transforming to Λ is the Vandermonde

$$T^{-1}\phi T = \Lambda.$$

Where $t_{ij} = \lambda_j^{i-1}$.

If there is a single complex pair $a \pm bi$ then we take

$$T = \begin{bmatrix} 1 & 0 \\ a & b \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{a}{b} & \frac{1}{b} \end{bmatrix}$$

$$\phi = \begin{bmatrix} 0 & 1 \\ -(a^2 + b^2) & 2a \end{bmatrix}$$

and

$$T^{-1}\phi T = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}.$$

We call this the real diagonal form for this ϕ. In general, if there are r complex roots, the real diagonal form for ϕ is the direct sum of r such 2×2 matrices and an $(n - r)$-dimensional diagonal matrix. The jth column of the generalized Vandermonde T corresponding to a real root λ_j is $t_{ij} = \lambda_j^{i-1}$. The columns, say 1 and 2, corresponding to the pair $\lambda_1 = a + ib$ and $\lambda_2 = a - ib$ are

$$t_{i1} = Re(\lambda_1^{i-1})$$

$$t_{i2} = Im(\lambda_1^{i-1}).$$

The first such column starts

$$1,\ a,\ a^2 - b^2,\ a^3 - 3ab^2,\ \ldots$$

the second such column starts

$$0,\ b,\ 2ab,\ 3a^2b - b^3,\ \ldots$$

2) Logarithm of the real diagonal form.

Let R denote the real diagonal form.

The logarithm of the diagonal part of R is very simple being the diagonal matrix M whose elements are the logarithms of the (positive real) diagonal elements of R.

The rest of R is the direct sum of 2×2 matrices of the form

$$\begin{bmatrix} a & b \\ -b & a \end{bmatrix}.$$

The logarithm of this matrix is

$$\begin{bmatrix} \log(a^2 + b^2) & \tan^{-1} \frac{b}{a} \\ -\tan^{-1} \frac{b}{a} & \log(a^2 + b^2) \end{bmatrix}.$$

The nondiagonal part of M is the direct sum of such 2×2 matrices.

As is well known, the logarithm is not uniquely defined. Naturally we take the smallest value of the imaginary part which will give the correct

$$
\gamma^* = T_L \begin{bmatrix} s_1 \\ s_2 \\ \cdot \\ \cdot \\ \cdot \\ s_{KM1} \end{bmatrix} .
$$

The impulse response $\hat{S}_k = c^* \Phi^{*k-1} \gamma^*$ of this system is compared with S_k, the system is put in companion form, and the logarithm system computed.

If a reasonable approximation between S and $\hat{S}$ was found, we RETURN. If a term was too much in error, the dimension of H is increased to include that term in the next system. This proceeds until a good fit is obtained or the S vector is exhausted.

VIII. Appendices

Attached are listings of: a main program used to generate data and call MICARE; MICARE; DOT; RAKAR; SVCAP; MATINV; MINV; CPC; MULLER; and the output produced by the data. Notice that the system generating S need not be stable.

```
      DIMENSION S(50)
      N = 39
      TOL1 = .0CCC1
      NST = 18
      H3 = .1
      IDERK = 3
      DO 2 I = 1, N
      XI = I
      T = (XI - 1.)*H3
      S(I) = EXP(2.*T) + 10.*(EXP(-2.*T))
    2 CONTINUE
      CALL MICARE (S,N,TOL1,NST,H3,IDERK)
      END
```

PROGRAM SHCULD END WITH A STCP, RETURN CR TRANSFER STATEMENT
RETURN STATEMENT SIMULATED.

SOURCE ERROR 276, LEVEL 1. WARNING CNLY.

,14

```
      SUBROUTINE MICARE (SSUBR, N,TOL1,NST,H3,IDERK)
C      THIS SUBROUTINE COMPUTES A MINIMAL COMPANION FORM (TRIVIALLY
C     1OBSERVABLE) REALIZATION OFA DYNAMICAL SYSTEM. INPUT SSUBR IS
C     1 FIRST N TERMS OF THE IMPULSE RESPONSE EXPANDED IN POWERS OF  1/S.
C     1METHOD OF B.L. HO
C
      DIMENSION SSUBR(50), S(20,20), RT(20,20), RCODE(20), TRR(20,20),
     1    TRL(20,20), B(20), C(20), SCAP(50)
C     NWR IS SYSTEM NUMBER FOR WRITEUNIT
      NWR = 3
      IDIM = 20
        NIX  =   0
       IFLAG = 0
        RK = 0
      WRITE (NWR, 4)
    4 FORMAT(1H1)
      WRITE (NWR,302)   TOL1
  302 FORMAT(//,5X,12HTOLERANCE = ,E15.8 )
      WRITE (NWR, 5)
    5 FORMAT(//10X, 12HINPUT VECTOR    /)
      WRITE (NWR, 6) (SSUBR (I), I = 1, N)
    6    FORMAT (6E20.8)
C
C     CHANGE N TO MAXIMUM DIMENSION
       N = N/2
      M3=2*N
C
C      BEGIN MAIN LOOP
      DO 26 I = NST , N
C      GO TO 60 IF CERTAIN VALUES OF I SHOULD BE DELETED
       IF (IFLAG) 8,8,60
    8   K = I
C
C      SET UP THE FIRST MATRIX FROM THE SSUBR VECTOR
       DO 10 L = 1,I
       DO 10 M = 1,I
      L5 = L+M
   10 S(L,M) = SSUBR(L5-1)
C
C     GO TO RAKAR FOR RANK,  ORTHONORMALIZED S, AND  ORTHONORMALIZING
C    1   MATRIX RT.
      TOL2 = TOL1
      RANK = IDERK
      CALL RAKAR (S,RT,RCODE,RANK,IDIM,K,K,TOL2)
C
C      IF RANK IS INCREASING, CONTINUE
       IF (RANK-RK) 18,18,25
C
C       IF S IS A NULL MATRIX DETERMINE IF SSUBR VECTOR IS EXHAUSTED
C
   18     IF (NIX) 20 , 20 , 19
   19  WRITE(NWR,7)
        GO TO 25
    7  FORMAT(36H  NIX EQUALS ONE AND RANK EQUALS RK.)
   20     IF (RK) 30, 30, 35
```

```
C       IF SSUBR IS A NULL VECTOR PRINT MESSAGE AND RETURN
   30      IF (N-I) 31, 31, 26
   31    WRITE (NWR, 32)
   32 FORMAT (///10X, 12HNULL MATRIX      )
         RETURN
C
C       IF RANK IS STABLE WE CONSTRUCT A REALIZATION AND START BY
C     1    CONSTRUCTING THE  SECOND MATRIX FROM THE  SSUBR VECTOR
   35      KM1 = K - 1
      WRITE (NWR,303)  KM1
  303 FORMAT ( // , 5X , 5HKM1= ,I5 , // )
      DO 40 L = 1, KM1
      DO 40 M = 1, KM1
      L5 = L+M
   40 S(L,M) = SSUBR(L5)
      IRANK = RK + .1
C
C     PUT THE SYSTEM MATRIX IN S, INPUT COEFFICIENTS IN B, OUTPUT
C     1    COEFFICIENTS IN C.
      DO 42 L = 1, IRANK
       B(L) = 0.
      DO 42 M = 1, KM1
      RT(L,M) = 0.
      DO 43 M1 = 1, KM1
   43   RT(L,M) = RT(L,M) + TRL (L,M1) * S(M1,M)
   42   B(L) = B(L) + TRL(L,M) * SSUBR(M)
      DO 41 L = 1, IRANK
      C(L) = 0
       DO 41 M = 1, IRANK
      S(L,M) = 0
      DO 44 M1 = 1, KM1
   44   S(L,M) = S(L,M)   + RT (L,M1) * TRR(M1,M)
   41   C(L) = C(L) + SSUBR(M) * TRR(M,L)
C
C      COMPUTE THE S-ESTIMATE VECTOR
       CALL             SVCAP(S,B,C,N,IRANK,IDIM,SCAP)
   81    WRITE (NWR, 84)
   84 FORMAT (// 10X, 16HESTIMATED VECTOR    /)
      WRITE (NWR, 6) (SCAP(L), L = 1, M3)
      IF (N-K) 96,96,97
   97    EPSIL = 0.
          M8 = 2*K  - 1
         DO 47   L=1,M8
      STLDA = 2.*ABS(SSUBR(L) - SCAP (L))
   47 EPSIL=AMAX1(EPSIL,STLDA/(1.+ABS(SSUBR(L))))
C
C     EPSIL IS THE MAXIMUM RELATIVE ERROR OF THE  THEORETICALLY ZERO
C     1    ERRORS
           EPSIL = AMAX1(EPSIL,1.E-7)
        M8  =  M8  +1
        DO   50   L  =  M8,M3
      STLDA = ABS ( SSUBR (L) - SCAP (L))
       ERROR = STLDA/(1. + ABS (SSUBR(L)))
   48      IF(ERROR-EPSIL) 50,50,51
   51    IF(IFLAG) 52,52,50
   52  IFLAG = (L+2)/2
```

```
C
C        A PRINT OF IFLAG INDICATES AN ERROR IN MATCHINT THE
C        ( 2* IFLAG - 1 ) OR ( 2 * IFLAG - 2 )  COMPONENT OF THE INPUT
C        VECTOR.
      WRITE(NWR,45)IFLAG
        GO TO 372
   45 FORMAT(7HOIFLAG=I2)
   50   CONTINUE
C
C       FORM TRANSFORMATION MATRIX
  372 IF(IRANK-1) 38,38,96
   96   DO 53 M1 = 1, IRANK
   53   RT(1,M1) = C(M1)
   49  DO 54 L = 2, IRANK
      DO 54 M = 1, IRANK
       RT(L,M) = C.
     DO 54 M1 = 1, IRANK
   54  RT(L,M) = RT(L,M) + RT(L - 1,M1)* S(M1,M)
C    THIS HAS FORMED THE MATRIX TRANSFORMING TO COMPANION FORM.
      DO 71 L = 1, IRANK
     RCODE (L) = C.
     DO 71 M=1,IRANK
      TRL (L,M) = 0.
     DO 7C M1 = 1, IRANK
   70  TRL(L,M) = TRL(L,M) + RT(L,M1)* S(M1,M)
   71   RCODE (L) = RCODE (L) + RT(L,M)*B(M)
C
C       OBTAIN THE INVERSE OF THE TRANSFORMING MATRIX
     RANK = IRANK
     TOL2 = TOL1
     CALL RAKAR ( RT,S, B, RANK , IDIM,IRANK,IRANK,TCL2)
     IRK = RANK + .1
       IF (IRANK-IRK) 62,62,63
C
C      THE TRANSFORMATION MATRIX IS SINGULA. PRINT MESSAGE AND RETURN
   63     WRITE (NWR, 64)
   64 FORMAT(///10X, 32HTRANSFORMATION MATRIX SINGULAR          )
      RETURN
C
C      THE TRANSFORMATION MATRIX IS NOW FOUND
   62     DO 67 L = 1, IRANK
      DO 67 M = 1, IRANK
      TRR (L,M) = 0.
      DO 67 M1 = 1, IRANK
   67     TRR(L,M) = TRR(L,M) + S(L,M1)* RT(M,M1)
C
C       FINALLY PUT A,B,C IN COMPANION FORM
       DO 76 L = 1, IRANK
       B(L) = C.
       DO 76 M = 1, IRANK
      S(L,M) = 0.
        DO 77 M1 = 1, IRANK
   77        S(L,M) = S(L,M) + TRL(L,M1)* TRR(M1,M)
   76     B(L) = B(L) + C(M)*TRR(M,L)
      DO 78 L = 1, IRANK
       C(L) = B(L)
```

```
 78          E(L) = RCCDE (L)
C        CCMPUTE SIGNAL ESTIMATE FRCM CCMPANICN FCRM.
         CALL             SVCAP (S,B,C, N, IRANK, ICIM, SCAP)
 82      WRITE (NWR, 86)
 86 FORMAT (//10X,37HVECTCR FRCM CCMPANICN FCRM OPERATICN        /)
     WRITE (NWR,6) (SCAP(L), L = 1,M3)
 38        WRITE (NWR, 39)
 39 FORMAT(//10X, 20HOUTPUT CCEFFICIENTS         /)
     WRITE (NWR,6)(C(L), L = 1, IRANK)
     WRITE (NWR, 37)
 37 FORMAT (//10X, 14HSYSTEM MATRIX    /)
      DO 36 L = 1, IRANK
    WRITE (3,101)
101       FORMAT ( // )
 36       WRITE (NWR,6)(S(L,LL), LL = 1, IRANK)

      WRITE (NWR,34)
 34 FORMAT(//10X, 19HINPUT CCEFFICIENTS         /)
     WRITE (NWR,6)(B(L), L = 1, IRANK)
    IRNKM1 = IRANK - 1
    IF (IRNKM1) 304,304,305
305 DO 98 IX1 = 1, IRNKM1
    C(IX1 + 1) = 0
C
      DO 98 IX2 = 1 , IRANK
C
      S(IX1,IX2) = C.
      IF ( ( IX1+1) .EQ. IX2 )  S(IX1,IX2) = 1.
C
 98 CONTINUE
304 CONTINUE
C
    CALL CPC ( S , IRANK , B , C , H3 )
       IF (K-N)  91,3CC,55
 91  IF(IFLAG) 3CC,3CC,26
 60       IF(IFLAG-I) 11,11,26
 11      IFLAG = C
     NIX = 1
      GO TC 8
 25         L = 1
       RK = RANK
      NIX = 0
       DC 95 M = 1, K
       IF(RCODE(M)) 95,95,27
 27         DO 28 M1 = 1,I
       TRL(L,M1) = S(M1,M)
 28         TRR(M1,L) = RT(M1,M)
       L = L + 1
 95           CONTINUE
      IF (I-N)26,90,9C
 90 WRITE(NWR,1CC)
100 FORMAT(//10X,60HRANK NCT STABILIZED BUT WE HAVE REACHEC MAXIMUM DI
   1MENSICN         )
      K = K + 1
      GO TO 35
 26       CONTINUE
```

```
   55         RETURN
  300    WRITE(NWR,301)
  301    FORMAT( 115H   THIS REALIZATION IS SUCCESSFUL, ALL COEFFICIENTS
     1      HAVE BEEN MATCHED BEFORE REACHING THE MAX DIMENSION           )
          RETURN
          END
```

```
      FUNCTION DCT(ICIM,NC,A,I,J)
      DIMENSION   A(ICIM,IDIM)
       DOT  =    C
      DO  1     K=1,NC
 1      DOT = COT + A(K,I)*A(K,J)
       RETURN
        END
```

```
      SUBROUTINE RAKAR ( S , RT , RCODE , RANK , IDIM,NC,NR,TOL1)
      DIMENSION XIP(20)
      DIMENSION S(IDIM,IDIM), RT(IDIM,IDIM), RCODE(IDIM)
C        SET UP IDENTITY MATRIX
  100 DO 10  I =1,NC
       DO 11  J = 1,NC
   11 RT(I,J) = 0
       RCODE(I) = 0
   10  RT(I,I) =  1.
C     IDENTITY MATRIX HAS BEEN SET UP.
C
C       FIND FIRST NONZERO COLUMN OF INPUT
       DO 20       L =  1,NC
      K = L
      XMAG= DOT(IDIM,NR,S,L,L)
       IF (XMAG)  20,20,21
   20        RCODE(L) = 0
C         NULL MATRIX EXIT
C
         RETURN
   21  XMAG =   1./SQRT(XMAG)
C       FIRST NONZERO COLUMN AND ITS NORMALIZING FACTOR HAVE
C        BEEN FOUND
C
C       NORMALIZE THE VECTOR
       DO 15     I  =  1,NC
   15   RT(I,K)   = RT(I,K)*XMAG
      DO 16 I=1,NR
   16   S(I,K)   =   S(I,K)*XMAG
       RCODE(K) = 1.
      RK = 1.
C          VECTOR HAS BEEN NORMALIZED AND THE INDEPENDENCE
C        INDICATOR HAS BEEN SET
C
C       PREPARE TO START MAIN LOOP
      KA=K+1
C
C       START MAIN LOOP OF GRAM-SCHMIDT PROCESS
      DO 50 J=KA,NC
C       FIND PREORTOGONALIZED LENGTH OF NEXT (JTH) VECTOR
      XMAG=DOT (IDIM,NR,S,J,J)
         JM1 = J- 1
C       L CONTROLS THE DOUBLE ORTHOGONALIZATION
       DO 40    L   = 1,2
C      K RUNS OVER THE PREVIOUSLY DETERMINED BASIS VECTORS
        DO 30   K   =  1,JM1
C       XIP(K)  IS THE INNER PRODUCT OF THE PRESENT (JTH) VECTOR
C       WITH THE KTH ORTHONORMALIZED VECTOR
   30 XIP(K)=DOT(IDIM,NR,S,J,K)
C      ORTHOGONALIZE THE JTH VECTOR
      DO 40  K = 1,JM1
      DO 45   I = 1,NC
   45  RT(I,J) = RT(I,J) - XIP(K)*RT(I,K)*RCODE(K)
      DO 40 I=1,NR
         S(I,J)   =   S(I,J)   -XIP(K)*S(I,K)*RCODE(K)
```

```
   40    CONTINUE
C        FIND LENGTH OF JTH VECTOR AFTER ORTHOGONALIZATION
C
      XIP(1)=DOT (IDIM,NR,S,J,J)
C        DETERMINE IF LENGTH IS SIGNIFICANT
      IF (XIP(1)/XMAG-TOL1) 42,42,60
C     IT IS SIGNIFICANT AT 60. IT IS NOT AT 42
   42 DO 43 I=1,NR
   43 S(I,J )=0
      GO TO 50
   60  RCODE(J) = 1.
      RK = RK + 1.
      XMAG  =   1./SQRT(XIP(1))
C        NORMALIZE THE ORTHOGONALIZED VECTOR
      DO   70     I = 1,NC
   70  RT(I,J)   =   RT(I,J)*XMAG
      DO 51 I=1,NR
   51    S(I,J)   =   S(I,J)*XMAG
      IF (RK-RANK) 50,52,52
   52 TOL1 = 1.
  50     CONTINUE
C       COMPLTE RANK
          RANK = C
         DC 75    L= 1,NC
   75  RANK = RANK + RCODE(L)
        RETURN
         END
```

```
      SUBROUTINE SVCAP (A,B,C,N,IRANK, IDIM, SCAP)
      DIMENSION SCAP(50), VEC(50), V1(50)
      DIMENSION A(IDIM,IDIM), B(IDIM), C(IDIM)
      DO 5 L = 1, IRANK
    5   V1(L) = B(L)
           M3 = 2 * N
        DO 45 L = 1, M3
          SCAP (L) = C
          DO 46 M =          1, IRANK
         VEC (M) = 0
        SCAP (L) = SCAP (L) + C(M) * V1(M)
       DO 46 M1 = 1, IRANK
   46  VEC(M) = VEC(M) + A(M,M1) * V1(M1)
       DO 45 M = 1, IRANK
   45   V1(M) = VEC(M)
       RETURN
       END
```

```
      SUBROUTINE MATINV ( A , AINV , DETA , N )
      DIMENSION A(15,15) , AINV(15,15)
      DIMENSION C(125) , WA1(15) , WA2(15)
C
      K= 1
      DO 1 J = 1 , N
C
        DO 1 I = 1 , N
C
        C(K) = A(I,J)
        K = K + 1
C
    1 CONTINUE
C
      CALL MINV ( C , N , DETA , WA1 , WA2 )
C
      K=1
      DO 2 J = 1 , N
C
        DO 2 I = 1 , N
C
        AINV(I,J) = C(K)
        K= K + 1
C
    2 CONTINUE
      RETURN
      END
```

```
C        SUBROUTINE MINV                                                 MIN
C                                                                        MIN
C        ...............................................................MIN
C                                                                        MIN
C                                                                        MIN
C        PURPOSE                                                         MIN
C           INVERT A MATRIX                                              MIN
C                                                                        MIN
C        USAGE                                                           MIN
C           CALL MINV(A,N,D,L,M)                                         MIN
C                                                                        MIN
C        DESCRIPTION OF PARAMETERS                                       MIN
C           A - INPUT MATRIX, DESTROYED IN COMPUTATION AND REPLACED BY   MIN
C               RESULTANT INVERSE.                                       MIN
C           N - ORDER OF MATRIX A                                        MIN
C           D - RESULTANT DETERMINANT                                    MIN
C           L - WORK VECTOR OF LENGTH N                                  MIN
C           M - WORK VECTOR OF LENGTH N                                  MIN
C                                                                        MIN
C        REMARKS                                                         MIN
C           MATRIX A MUST BE A GENERAL MATRIX                            MIN
C                                                                        MIN
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                   MIN
C           NONE                                                         MIN
C                                                                        MIN
C        METHOD                                                          MIN
C           THE STANDARD GAUSS-JORDAN METHOD IS USED. THE DETERMINANT    MIN
C           IS ALSO CALCULATED. A DETERMINANT OF ZERO INDICATES THAT     MIN
C           THE MATRIX IS SINGULAR.                                      MIN
C                                                                        MIN
C     ..................................................................MIN
C                                                                        MIN
      SUBROUTINE MINV(A,N,D,L,M)                                         MIN
      DIMENSION A(1),L(1),M(1)                                           MIN
C                                                                        MIN
C        ...............................................................MIN
C                                                                        MIN
C        IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE   MIN
C        C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION       MIN
C        STATEMENT WHICH FOLLOWS.                                        MIN
C                                                                        MIN
C     DOUBLE PRECISION A,D,BIGA,HOLD                                     MIN
C                                                                        MIN
C        THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS     MIN
C        APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS       MIN
C        ROUTINE.                                                        MIN
C                                                                        MIN
C        THE DOUBLE PRECISION VERSION OF THIS SUBROUTINE MUST ALSO       MIN
C        CONTAIN DOUBLE PRECISION FORTRAN FUNCTIONS.  ABS IN STATEMENT   MIN
C        10 MUST BE CHANGED TO DABS.                                     MIN
C                                                                        MIN
C        ...............................................................MIN
C                                                                        MIN
C        SEARCH FOR LARGEST ELEMENT                                      MIN
C                                                                        MIN
```

```
      D=1.0                                                              MI
      NK=-N                                                              MI
      DO 80 K=1,N                                                        MI
      NK=NK+N                                                            MI
      L(K)=K                                                             MI
      M(K)=K                                                             MI
      KK=NK+K                                                            MI
      BIGA=A(KK)                                                         MI
      DO 20 J=K,N                                                        MI
      IZ=N*(J-1)                                                         MI
      DO 20 I=K,N                                                        MI
      IJ=IZ+I                                                            MI
   10 IF( ABS(BIGA)- ABS(A(IJ))) 15,20,20                                MI
   15 BIGA=A(IJ)                                                         MI
      L(K)=I                                                             MI
      M(K)=J                                                             MI
   20 CONTINUE                                                           MI
C                                                                        MI
C        INTERCHANGE ROWS                                                MI
C                                                                        MI
      J=L(K)                                                             MI
      IF(J-K) 35,35,25                                                   MI
   25 KI=K-N                                                             MI
      DO 30 I=1,N                                                        MI
      KI=KI+N                                                            MI
      HOLD=-A(KI)                                                        MI
      JI=KI-K+J                                                          MI
      A(KI)=A(JI)                                                        MI
   30 A(JI) =HOLD                                                        MI
C                                                                        MI
C        INTERCHANGE COLUMNS                                             MI
C                                                                        MI
   35 I=M(K)                                                             MI
      IF(I-K) 45,45,38                                                   MI
   38 JP=N*(I-1)                                                         MI
      DO 40 J=1,N                                                        MI
      JK=NK+J                                                            MI
      JI=JP+J                                                            MI
      HOLD=-A(JK)                                                        MI
      A(JK)=A(JI)                                                        MI
   40 A(JI) =HOLD                                                        MI
C                                                                        MI
C        DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS         MI
C        CONTAINED IN BIGA)                                              MI
C                                                                        MI
   45 IF(BIGA) 48,46,48                                                  MI
   46 D=0.0                                                              MI
      RETURN                                                             MI
   48 DO 55 I=1,N                                                        MI
      IF(I-K) 50,55,50                                                   MI
   50 IK=NK+I                                                            MI
      A(IK)=A(IK)/(-BIGA)                                                MI
   55 CONTINUE                                                           MI
C                                                                        MI
C        REDUCE MATRIX                                                   MI
C                                                                        MI
```

```
      DO 65 I=1,N                                                          MIN
      IK=NK+I                                                              MIN
      IJ=I-N                                                               MIN
      DO 65 J=1,N                                                          MIN
      IJ=IJ+N                                                              MIN
      IF(I-K) 60,65,60                                                     MIN
   60 IF(J-K) 62,65,62                                                     MIN
   62 KJ=IJ-I+K                                                            MIN
      A(IJ)=A(IK)*A(KJ)+A(IJ)                                              MIN
   65 CONTINUE                                                             MIN
C                                                                          MIN
C        DIVIDE ROW BY PIVOT                                               MIN
C                                                                          MIN
      KJ=K-N                                                               MIN
      DO 75 J=1,N                                                          MIN
      KJ=KJ+N                                                              MIN
      IF(J-K) 70,75,70                                                     MIN
   70 A(KJ)=A(KJ)/BIGA                                                     MIN
   75 CONTINUE                                                             MIN
C                                                                          MIN
C        PRODUCT OF PIVOTS                                                 MIN
C                                                                          MIN
      D=D*BIGA                                                             MIN
C                                                                          MIN
C        REPLACE PIVOT BY RECIPROCAL                                       MIN
C                                                                          MIN
      A(KK)=1.0/BIGA                                                       MIN
   80 CONTINUE                                                             MIN
C                                                                          MIN
C        FINAL ROW AND COLUMN INTERCHANGE                                  MIN
C                                                                          MIN
      K=N                                                                  MIN
  100 K=(K-1)                                                              MIN
      IF(K) 150,150,105                                                    MIN
  105 I=L(K)                                                               MIN
      IF(I-K) 120,120,108                                                  MIN
  108 JQ=N*(K-1)                                                           MIN
      JR=N*(I-1)                                                           MIN
      DO 110 J=1,N                                                         MIN
      JK=JQ+J                                                              MIN
      HOLD=A(JK)                                                           MIN
      JI=JR+J                                                              MIN
      A(JK)=-A(JI)                                                         MIN
  110 A(JI) =HOLD                                                          MIN
  120 J=M(K)                                                               MIN
      IF(J-K) 100,100,125                                                  MIN
  125 KI=K-N                                                               MIN
      DO 130 I=1,N                                                         MIN
      KI=KI+N                                                              MIN
      HOLD=A(KI)                                                           MIN
      JI=KI-K+J                                                            MIN
      A(KI)=-A(JI)                                                         MIN
  130 A(JI) =HOLD                                                          MIN
      GO TO 100                                                            MIN
  150 RETURN                                                               MIN
      END
```

```
      SUBROUTINE CPC ( S , IRANK , B , C , DT )
      DIMENSION S(20,20) , B(15) , C(15) , CCOF(16) ,ROOTR(15),ROOTI(15)
      DIMENSION RR(15) , RI(15) , T(15,15) , TINV(15,15),XM(15,15)
       NWR = 3
      DO 1 I = 1 , IRANK
C
        CCOF(I) = -S(IRANK,I)
C
    1 CONTINUE
C
      CCOF(IRANK+1) = 1.
C
      CALL MULLER ( CCOF , IRANK , ROOTR , ROOTI )
C
      NNZRO = 0
      J=0
C
      DO 2 I = 1 , IRANK
C
      X = (ABS(ROOTI(I)))/(1.E-7 + ABS(ROOTR(I)))
      IF ( X .LE. 1.E-7 ) GO TO 3
        IF ( ROOTI(I) .EQ. 0. ) GO TO 3
C
        NNZRO = NNZRO + 1
        RR(NNZRO) = ROOTR(I)
        RI(NNZRO) = ROOTI(I)
        GO TO 2
C
    3   CONTINUE
        K = IRANK - J
        RR(K) = ROOTR(I)
        RI(K) = 0.
        J = J + 1
    2 CONTINUE
C
      IR = NNZRO / 2
      WRITE(NWR,100) (I,ROOTR(I) , ROOTI(I) , I = 1 , IRANK )
  100 FORMAT ( /// , 7X , 33HROOT     REAL PART          CMPLX PART , / ,
     1         ( 5X , I5 , 3X , E15.8 , 3X , E15.8 ) )
      WRITE (3,101)  NNZRO
  101  FORMAT (2X,27H NUMBER OF COMPLEX ROOTS = ,  I5 , // )
C
      IF ( IR .EQ. 0 ) GO TO 7
C
      DO 5 J = 1 , IR
        IX1 = 2*J  - 1
        IX2 = 2*J
        T( 1,IX1) = 1.
        T( 1,IX2) = 0.
C
    5 CONTINUE
C
      DO 4 I = 2 , IRANK
C
        DO 6 J = 1 , IR
```

```
C
          J1 =   2*J   - 1
          J2 =   2*J
          T(I,J1) = RR(J1)* T(I-1,J1) - ABS( RI(J1) ) * T(I-1,J2)
          T(I,J2) = RR(J1)* T(I-1,J2) + ABS( RI(J1) ) * T(I-1,J1)
C
    6   CONTINUE
C
    4 CONTINLE
C
    7 CONTINLE
      IF ( NNZRO .GE. IRANK ) GC TC 24
C
      M1 = 2 * IR + 1
C
      DO 8 I = M1 , IRANK
C
        DO 9 J = 1 , IRANK
C
          T(J,I) = RR(I )**(J-1)
C
    9   CONTINUE
C
    8 CONTINLE
C
   24 CONTINLE
      CALL MATRIX INVERSICN
C
C
      CALL MATINV ( T , TINV , DET , IRANK )
      DO 10 I = 1 , IRANK
C
        ROOTR(I) = C.
        DO 1C J = 1 , IRANK
          XM(I,J) = O.
          ROCTR(I) = ROCTR(I) + C(J) * T(J,I)
C
          DO 10 K = 1 , IRANK
C
            XM(I,J) = XM(I,J) + S(I,K) * T(K,J)
C
   10 CONTINLE
      DO 11 I=1 , IRANK
C
      ROOTI(I) = C.
        DO 11 J = 1 , IRANK
        S(I,J) = C
      ROOTI(I) = ROOTI(I) + TINV(I,J) * B(J)
C
          DO 11 K = 1 , IRANK
C
            S(I,J) = S(I,J) + TINV(I,K) * XM(K,J)
   11 CONTINLE
      WRITE (NWR,1C4)
      WRITE (NWR,1C2)  (RCCTR(I),I=1,IRANK)
      WRITE (3,103)
      WRITE (3,112)
```

```
      DO 12 I = 1 , IRANK
  112 FORMAT  (50H   COMPUTED REAL DIAGONAL FORM FOR DISCRETE SYSTEM)
      WRITE (3,102)  (S(I,J) , J = 1,IRANK)
   12 CONTINUE
      WRITE (3,103)
      WRITE (NWR,105)
      WRITE (NWR,102)  (ROOTI(I),I=1,IRANK)
      WRITE(3,103)
  102 FORMAT ( 6E20.8 )
  103 FORMAT ( / )
C
      IF ( NNZRO .GE. IRANK ) GO TO 25
      DO 13 I = M1 , IRANK
C
        DO 14 J = 1 , IRANK
C
          S(I,J) = 0.
C
   14   CONTINUE
          S(I,I) = ALOG ( ABS ( RR(I) ) ) / DT
C
   13 CONTINUE
C
      IF ( IR .EQ. 0 ) GO TO 17
C
   25 CONTINUE
      DO 15 I = 1 , IR
C
        J1 = 2 * I - 1
        J2 = 2 * I
C
        DO 16 J = 1 , IRANK
C
        S(J1,J) = 0.
        S(J2,J) = 0.
C
   16   CONTINUE
C
        S(J1,J1) = ( ALOG ( SQRT ( RR(J1) **2 + RI(J1)**2 ) ) ) / DT
        S(J2,J2) = S(J1,J1)
      S(J1,J2) = ATAN2(ABS(RI(J2)),RR(J2))/DT
   15   S(J2,J1) = - S(J1,J2)
   17 WRITE(3,104)
      WRITE (NWR,102) (ROOTR(K),K=1,IRANK)
  104 FORMAT ( // , 21H   OUTPUT COEFFICIENTS )
      WRITE (3,106)
  106 FORMAT(/,5X,40HCONTINUOUS SYSTEM IN REAL DIAGONAL FORM )
      DO 18 I = 1 , IRANK
C
      WRITE(NWR,102) ( S(I,J),J=1,IRANK)
C
   18 CONTINUE
      WRITE (3,103)
      WRITE (3,105)
  105 FORMAT(20H   INPUT COEFFICIENTS )
      WRITE (NWR,102)  (ROOTI(K),K=1,IRANK)
```

```
C
      DO 19 I = 1 , IRANK
C
        TINV(1,I) = ROOTR(I)
   19 CONTINUE
C
      DO 20 I = 2 , IRANK
C
        DO 20 J = 1 , IRANK
C
          TINV(I,J) = C
C
            DO 20 K = 1 , IRANK
C
              TINV(I,J) = TINV(I,J) + TINV(I-1,K) * S(K,J)
C
   20 CONTINUE
C
C         CALL MATRIX INVERSION
C
      CALL MATINV ( TINV , T , DET , IRANK )
      DO 21 I= 1 , IRANK
        C( I) = C.
        DO 21 J = 1 , IRANK
C
          XM(I,J) = 0.
          C(I) = C(I) + ROOTR(J) * T(J,I)
C
          DO 21 K = 1 , IRANK
C
            XM(I,J) = XM(I,J) + S(I,K) * T(K,J)
C
   21 CONTINUE
C
      DO 22 I = 1 , IRANK
C
        B(I) = 0.
C
          DO 22 J = 1 , IRANK
C
            S(I,J) = C.
            B(I) = B(I) + TINV(I,J) * ROCTI(J)
C
            DO 22 K= 1 , IRANK
C
              S(I,J) = S(I,J) + TINV(I,K) * XM(K,J)
C
   22 CONTINUE
C
      WRITE (3,104)
      WRITE (NWR,1C2)  (C(I),I=1,IRANK)
      WRITE (3,1C7)
  107 FORMAT (/,5X,37H CONTINUOUS SYSTEM IN COMPANION FORM )
      DO 23 I = 1 , IRANK
C
      WRITE (NWR,1C2)  (S(I,J),J=1,IRANK)
```

```
      C
         23 CONTINUE
            WRITE (3,103)
            WRITE (3,105)
            WRITE (NWR,102)   (B(I),I=1,IRANK)
      C
            RETURN
            END
```

```
      SUBROUTINE MULLER(COE,N1,ROOTR,ROOTI)
C MULLER
      DIMENSION COE(16),ROOTR(15),ROOTI(15)
C
C        COEFS IN ORDER OF INCREASING POWERS OF Z
C
      NUP = ( N1 + 1 ) / 2
      DO 20 I = 1 , NUP
      J = N1 + 2 - I
      CSV = COE(I)
      COE(I) = COE(J)
   20 COE(J) = CSV
      N2=N1+1
      N4=0
      I=N1+1
   19 IF(COE(I))9,7,9
    7 N4=N4+1
      ROOTR(N4)=0.
      ROOTI(N4)=0.
      I=I-1
      IF(N4-N1)19,37,19
    9 CONTINUE
   10 AXR=0.8
      AXI=0.
      L=1
      N3=1
      ALP1R=AXR
      ALP1I=AXI
      M=1
      GOTO99
   11 BET1R=TEMR
      BET1I=TEMI
      AXR=0.85
      ALP2R=AXR
      ALP2I=AXI
      M=2
      GOTO99
   12 BET2R=TEMR
      BET2I=TEMI
      AXR=0.9
      ALP3R=AXR
      ALP3I=AXI
      M=3
      GOTO99
   13 BET3R=TEMR
      BET3I=TEMI
   14 TE1=ALP1R-ALP3R
      TE2=ALP1I-ALP3I
      TE5=ALP3R-ALP2R
      TE6=ALP3I-ALP2I
      TEM=TE5*TE5+TE6*TE6
      TE3=(TE1*TE5+TE2*TE6)/TEM
      TE4=(TE2*TE5-TE1*TE6)/TEM
      TE7=TE3+1.
      TE9=TE3*TE3-TE4*TE4
```

```
      TE10=2.*TE3*TE4                                              HPR
      DE15=TE7*BET3R-TE4*BET3I                                     HPR
      DE16=TE7*BET3I+TE4*BET3R                                     HPR
      TE11=TE3*BET2R-TE4*BET2I+BET1R-DE15                          HPR
      TE12=TE3*BET2I+TE4*BET2R+BET1I-DE16                          HPR
      TE7=TE9-1.                                                   HPR
      TE1=TE9*BET2R-TE10*BET2I                                     HPR
      TE2=TE9*BET2I+TE10*BET2R                                     HPR
      TE13=TE1-BET1R-TE7*BET3R+TE10*BET3I                          HPR
      TE14=TE2-BET1I-TE7*BET3I-TE10*BET3R                          HPR
      TE15=DE15*TE3-DE16*TE4                                       HPR
      TE16=DE15*TE4+DE16*TE3                                       HPR
      TE1=TE13*TE13-TE14*TE14-4.*(TE11*TE15-TE12*TE16)             HPR
      TE2=2.*TE13*TE14-4.*(TE12*TE15+TE11*TE16)                    HPR
      TEM = SQRT(TE1*TE1+TE2*TE2)
      IF(TE1)113,113,112                                           HPR
  113 TE4=SQRT(.5*(TEM-TE1))
      TE3=.5*TE2/TE4                                               HPR
      GO TO 111                                                    HPR
  112 TE3=SQRT(.5*(TEM+TE1))
      IF(TE2)110,200,200                                           HPR
  110 TE3=-TE3                                                     HPR
  200 TE4=.5*TE2/TE3                                               HPR
  111 TE7=TE13+TE3                                                 HPR
      TE8=TE14+TE4                                                 HPR
      TE9=TE13-TE3                                                 HPR
      TE10=TE14-TE4                                                HPR
      TE1=2.*TE15                                                  HPR
      TE2=2.*TE16                                                  HPR
      IF(TE7*TE7+TE8*TE8-TE9*TE9-TE10*TE10)204,204,205             HPR
  204 TE7=TE9                                                      HPR
      TE8=TE10                                                     HPR
  205 TEM=TE7*TE7+TE8*TE8                                          HPR
      TE3=(TE1*TE7+TE2*TE8)/TEM                                    HPR
      TE4=(TE2*TE7-TE1*TE8)/TEM                                    HPR
      AXR=ALP3R+TE3*TE5-TE4*TE6                                    HPR
      AXI=ALP3I+TE3*TE6+TE4*TE5                                    HPR
      ALP4R=AXR                                                    HPR
      ALP4I=AXI                                                    HPR
      M=4                                                          HPR
      GO TO 99                                                     HPR
   15 N6=1                                                         HPR
   38 IF(ABS(HELL)+ABS(BELL)-1.E-20)18,18,16
   16 TE7=ABS(ALP3R-AXR)+ABS(ALP3I-AXI)
      IF(TE7/(ABS(AXR)+ABS(AXI))-1.E-7)18,18,17
   17 N3=N3+1                                                      HPR
      ALP1R=ALP2R                                                  HPR
      ALP1I=ALP2I                                                  HPR
      ALP2R=ALP3R                                                  HPR
      ALP2I=ALP3I                                                  HPR
      ALP3R=ALP4R                                                  HPR
      ALP3I=ALP4I                                                  HPR
      BET1R=BET2R                                                  HPR
      BET1I=BET2I                                                  HPR
      BET2R=BET3R                                                  HPR
      BET2I=BET3I                                                  HPR
```

```
      BET3R=TEMR                                                    FPR
      BET3I=TEMI                                                    FPR
      IF(N3-100)14,18,18                                            FPR
   18 N4=N4+1                                                       FPR
      ROOTR(N4)=ALP4R                                               FPR
      ROOTI(N4)=ALP4I                                               FPR
      N3=0                                                          FPR
   41 IF(N4-N1)30,37,37                                             FPR
   37 RETURN                                                        HPR
   30 IF(ABS(ROOTI(N4))-1.E-5)10,10,31
   31 GO TO(32,10),L                                                FPR
   32 AXR=ALP1R                                                     FPR
      AXI=-ALP1I                                                    FPR
      ALP1I=-ALP1I                                                  FPR
      M=5                                                           HPR
      GO TO 99                                                      FPR
   33 BET1R=TEMR                                                    FPR
      BET1I=TEMI                                                    FPR
      AXR=ALP2R                                                     FPR
      AXI=-ALP2I                                                    FPR
      ALP2I=-ALP2I                                                  FPR
      M=6                                                           FPR
      GO TO 99                                                      FPR
   34 BET2R=TEMR                                                    FPR
      BET2I=TEMI                                                    FPR
      AXR=ALP3R                                                     FPR
      AXI=-ALP3I                                                    FPR
      ALP3I=-ALP3I                                                  FPR
      L=2                                                           HPR
      M=3                                                           FPR
   99 TEMR=CCE(1)                                                   FPR
      TEMI=0.0                                                      FPR
      DO100I=1,N1                                                   FPR
      TE1=TEMR*AXR-TEMI*AXI                                         FPR
      TEMI=TEMI*AXR+TEMR*AXI                                        FPR
  100 TEMR=     TE1+CCE(I+1)                                        FPR
      HELL=TEMR                                                     FPR
      BELL=TEMI                                                     FPR
   42 IF(N4)102,103,102                                             FPR
  102 DO101I=1,N4                                                   FPR
      TEM1=AXR-ROOTR(I)                                             HPR
      TEM2=AXI-ROOTI(I)                                             FPR
      TE1=TEM1*TEM1+TEM2*TEM2                                       FPR
      TE2=(TEMR*TEM1+TEMI*TEM2)/TE1                                 FPR
      TEMI=(TEMI*TEM1-TEMR*TEM2)/TE1                                FPR
  101 TEMR=TE2                                                      FPR
  103 GO TO(11,12,13,15,33,34),M                                    FPR
      END
```

```
TOLERANCE =  C.C9999999E-C4

INPUT VECTOR
0.11000000E C2      C.94087101E C1      0.81950250E 01      0.7310235
0.63320591E C1      0.65211695E C1      0.69719975E 01      0.7702636
0.11930355E C2      C.14206473E C2      0.17052747E 02      0.2058340
0.36871470E C2      C.44924891E C2      0.54781304E 02      0.6683628
0.12159271E C3      0.14848053E C3      0.18132739E 03      0.2214515
0.40345355E C3      C.49276929E C3      0.60186163E 03      0.7351087
0.13394381E C4      C.16359904E C4      0.19982008E 04

KM1=     18

ESTIMATEC VECTOR
0.10999999E C2      0.94087097E 01      0.81950239E 01      0.7310233
0.63320526E C1      C.65211610E C1      0.69719867E 01      0.7702622
0.11930326E C2      C.14206438E C2      0.17052703E 02      0.2058335
0.36871376E C2      C.44924778E C2      0.54781169E 02      0.6683612
0.12159240E C3      C.14848015E C3      0.18132695E 03      0.2214510
0.40345258E C3      0.49276812E C3      0.60186017E 03      0.7351070
0.13394348E C4      C.16359864E C4

VECTOR FRCM COMPANICN FORM CPERATION
0.11000001E C2      C.94087117E 01      0.81950267E 01      0.7310237
0.63320677E C1      0.65211816E C1      0.69720145E 01      0.7702659
0.11930407E C2      C.14206541E C2      0.17052833E 02      0.2058351
0.36871699E C2      C.44925161E C2      0.54781672E 02      0.6683675
0.12159365E C3      C.14848171E C3      0.18132889E 03      0.2214534
0.40345725E C3      0.49277394E C3      0.60186742E 03      0.7351160
0.13394523E C4      C.16360081E C4

OUTPUT COEFFICIENTS
0.09999999E C1      C.23841858E-C6

SYSTEM MATRIX

0.23841858E-C6      C.99999976E C0

-0.99999988E CC      C.20401334E C1

INPUT COEFFICIENTS
0.10999999E C2      0.94087096E 01

ROOT     REAL PART        CMPLX PART
  1      0.81873053E CC    -C.
  2      0.12214029E C1    -C.
NUMBER OF COMPLEX RCOTS =       C
```

```
OUTPUT COEFFICIENTS
    0.09999999E 01      0.09999999E 01

 COMPUTED REAL DIAGONAL FORM FOR DISCRETE SYSTEM
    0.12214028E 01     -0.29802322E-07
    0.59604645E-07      0.81873055E 00

INPUT COEFFICIENTS
    0.10000043E 01      0.99999949E 01


OUTPUT COEFFICIENTS
    0.09999999E 01      0.09999999E 01

   CONTINUOUS SYSTEM IN REAL DIAGONAL FORM
    0.20000014E 01      0.
    0.                 -0.20000027E 01

INPUT COEFFICIENTS
    0.10000043E 01      0.99999949E 01

OUTPUT COEFFICIENTS
    0.09999999E 01     -0.

    CONTINUOUS SYSTEM IN COMPANION FORM
    0.14901161E-07      0.99999999E 00
    0.40000085E 01     -0.12964010E-05

INPUT COEFFICIENTS
    0.10999999E 02     -0.18000007E 02
THIS REALIZATION IS SUCCESSFUL, ALL COEFFICIENTS             HAVE BEEN MATCHED
```

exponential. Our justification for this is again the assumption that the δ used to generate ϕ was smaller than half the smallest natural period appearing in the spectrum of A.

3) Companion form.

An nth order matrix A is said to be in companion form if

$$a_{i,i+1} = 1 ,$$

the characteristic polynomial of A is

$$x^n + \sum_{j=o}^{n-1} a_{n,j+1} x^j ,$$

and all other a_{ij}, besides the last row and the first upper diagonal, are zero.

It is easy to show that if the matrix

$$S = \begin{bmatrix} H \\ H\phi \\ \cdot \\ \cdot \\ \cdot \\ H\phi^{n-1} \end{bmatrix}$$

is nonsingular, i.e. if $[H, \phi]$ is completely observable, then

$$HS^{-1} = [1, 0, \cdots, 0]$$

and $S\phi S^{-1}$ is in companion form.

Appendices: Attached are listings of a main program to call CPC, the data used by that main, and the output from CPC produced by that data.

Listings of CPC, MULLER, MATINV, and MINV appear in APPENDIX B - MICARE.

```
      DIMENSION S(15,15),B(15),C(15)
C     THIS IS THE MAIN FOR CPC- K IS IRANK
C
    1 READ 100,K,DT
      READ 101,(B(I),I=1,K),(C(I),I=1,K)
      READ 101,((S(I,J),J=1,K),I=1,K)
      PRINT 102
      DO 10 I=1,K
      PRINT 103,(S(I,J),J=1,K)
      PRINT 108
   10 CONTINUE
      PRINT 108
      PRINT 104, (B(I),I=1,K)
      PRINT 105, (C(I),I=1,K)
      PRINT 107,DT
      CALL CPC (S,K,B,C,DT)
      GO TO 1
  100 FORMAT (I10,E20.8)
  101 FORMAT (5E14.8)
  102 FORMAT (23X,14HINPUT MATRIX S ,///)
  103 FORMAT (1P6E20.8)
  104 FORMAT (15H0INPUT VECTOR B,1P4E20.8)
  105 FORMAT (15H0INPUT VECTOR C,1P4E20.8)
  107 FORMAT ( 3H0DT ,1PE20.8)
  108 FORMAT (1H )
      END
         3    1.
.0025037473    .9735          -3.85          1.
     0
     0              1.              0                                 0
     1.        -13.096271     -15.948172     -6.7669062
```

INPUT MATRIX S

.00000000E 00	1.00000000E 00	.00000000E 00
.00000000E 00	.00000000E 00	1.00000000E 00
-1.30962710E 01	-1.59481720E 01	-6.76690620E 00

INPUT VECTOR B	2.50374730E-03	9.73500000E-01	-3.85000000E 00
INPUT VECTOR C	1.00000000E 00	.00000000E 00	.00000000E 00

DT 1.00000000E 00

ROOT	REAL PART	CMPLX PART
1	-.22011641E 01	.83270654E 00
2	-.22011641E 01	-.83270654E 00
3	-.23645780E 01	-.89988780E-17

NUMBER OF COMPLEX ROOTS = 2

OUTPUT COEFFICIENTS

.10000000E 01	.00000000E 00	.10000000E 01

COMPUTED REAL DIAGONAL FORM FOR DISCRETE SYSTEM

-.22011641E 01	.83270654E 00	.23283064E-09
-.83270654E 00	-.22011641E 01	-.30559022E-09
-.93132257E-09	-.11641532E-09	-.23645780E 01

INPUT COEFFICIENTS

-.62175809E 00	.12982055E 01	.62426183E 00

OUTPUT COEFFICIENTS

.10000000E 01	.00000000E 00	.10000000E 01

CONTINUOUS SYSTEM IN REAL DIAGONAL FORM

.85586399E 00	.27799295E 01	.00000000E 00
-.27799295E 01	.85586399E 00	.00000000E 00
.00000000E 00	.00000000E 00	.86059955E 00

INPUT COEFFICIENTS

-.62175809E 00	.12982055E 01	.62426183E 00

OUTPUT COEFFICIENTS

.10000000E 01	.90949470E-12	-.90949470E-12

CONTINUOUS SYSTEM IN COMPANION FORM

.00000000E 00	.10000000E 01	-.90949470E-12
-.36379788E-10	-.15461410E-10	.10000000E 01
.72811123E 01	-.99336237E 01	.25723275E 01

INPUT COEFFICIENTS

.25037472E-02	.36140188E 01	.10989349E 02