LOW COST ANTI-JAM DIGITAL DATA-LINKS
TECHNIQUES INVESTIGATIONS

Telecommunication and Control Systems Laboratory
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843

Final Report for Phase III
Contract F-33615-75-C-1011
For the period 1 March 1978 through 15 April 1979

Approved for public release; distribution unlimited.

MAY 1979

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OHIO 45433
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

John W. Mayhan
Project Engineer

Charles C. Gauder
Chief, Avionic Communications Branch
Avionics Laboratory

FOR THE COMMANDER

Raymond E. Sifferd, COL, USAF
Chief, System Avionics Division
Avionics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFH/WA/AAAD, W-PAB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.
Low Cost Anti-Jam Digital Data-Links Techniques Investigations, Volume III.

Approved for public release; distribution unlimited.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Signal Processing
Interference Concealment
Optimum Demodulation
Recursive Maximum Likelihood Demodulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report documents the final phase of research under the subject contract. Previous results showed that the Minimum Probability of Error recursive detector for colored plus white noise, tracks the colored noise and subtracts it from the data. The present effort investigated the effects on optimum detector performance of carrier phase estimation. A good characterization of the effects was obtained.
PREFACE

From 1971 through 1973, a new sampled-data processing technique for digital signals subject to colored multiplicative noise was developed and subsequently patented by the Principal Investigator, at NASA Langley Research Center. In 1974, a contract was issued by the Air Force Avionics Laboratory to determine if the same technique which provided processing gain against diffuse Doppler-spread multipath perturbations could be applied to anti-jam processing.

Anti-jam processing algorithms were produced under the 1974 contract, as well as a Monte Carlo simulation package for performance evaluation. Between 1976 and 1978, substantial evaluation of the algorithms was performed and documented, under an extension of the contract.

A final extension of the contract, through April 1979 served to support investigation of means for implementing carrier phase estimation and bit synchronization with the detection algorithms. This report documents those results and gives recommendations for further research.

LIST OF RESEARCH PERSONNEL

Dr. John H. Painter Principal Investigator
Dr. C. June Yoon Research Associate
Mr. Joel N. Holyoak Research Associate
Mr. Randall C. Reininger Research Assistant
Ms. Lanette Dockall Secretarial Assistant
Mr. Paul Painter Graphics

The Telecommunication and Control Systems Laboratory
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843
(713) 845-7441
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>1</td>
</tr>
<tr>
<td>II.</td>
<td>2</td>
</tr>
<tr>
<td>III.</td>
<td>24</td>
</tr>
<tr>
<td>IV.</td>
<td>29</td>
</tr>
<tr>
<td>V.</td>
<td>35</td>
</tr>
<tr>
<td>VI.</td>
<td>38</td>
</tr>
</tbody>
</table>

## I. INTRODUCTION

## II. COHERENT DETECTION WITH PHASE ESTIMATION

1. Signal and Channel Model | 2
2. Joint Detection With Phase Estimation | 5
3. The I-Q Data Model With Phase Estimation | 9
4. Detector Structure and Algorithms | 13
5. The Phase Estimator | 17
6. The Loop Filter Mechanization | 19

## III. ON THE EXISTANCE OF NON-COHERENT TRACKING DETECTORS

## IV. SIMULATION RESULTS

## V. COMPLETE RECEIVER ALGORITHMS

1. A Proposed Bit Synchronization Algorithm | 35
2. The Complete Algorithm | 36

## VI. CONCLUSION

## APPENDIX A - The Closed-Form Error-Rate Program

## APPENDIX B - The Monte-Carlo Simulation Program

## REFERENCES
<table>
<thead>
<tr>
<th>Figure</th>
<th>Illustration</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Physical Channel and Receiver Models</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>I-Q Carrier Demodulator</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Data Generating Model</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Data Model</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Data Generator Model for Phase Estimation</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Compound Detector and Phase Estimation</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>Tracking Filter</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Tangent Phase-Locked Loop</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>Discrete-Time Phase Estimator</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>Kalman Filter</td>
<td>26</td>
</tr>
<tr>
<td>11</td>
<td>Kalman Filter</td>
<td>27</td>
</tr>
<tr>
<td>12</td>
<td>Simulation Results</td>
<td>32</td>
</tr>
<tr>
<td>13</td>
<td>Complete Algorithm Diagram</td>
<td>37</td>
</tr>
</tbody>
</table>
SECTION I
INTRODUCTION

This report documents further research under the subject contract whose previous results have been reported in [1,4]. The basic technical problem is that of optimum discrete-time recursive detection of binary signals subject to additive colored and white noise. Previous results showed that the Minimum Probability of Error detector is one which tracks the colored noise and subtracts it from the received data. The related question of identification of the statistics of the colored interfering process was extensively investigated in Reference 1.

The research effort, documented herein, was pointed toward several related questions. First, it was desired to investigate the problem of simultaneous estimation of the carrier phase references required by the coherent detection algorithm. It was desired to specifically determine the method for measuring phase and also the augmentation of the detection algorithm required to operate with imperfect phase estimates.

Next, it was desired to investigate the possibility of non-coherent detection with interference tracking, with application to Frequency-Shift-Keying and Differential Phase-Shift-Keying.

A third area of interest was to determine a method for obtaining bit synchronization for the interference-tracking detection algorithms. This would then lead to assembly of a complete algorithm for the so-called IDEI (Integrated Detection, Estimation, Identification) receiver.

Finally, it was desired to obtain Monte Carlo evaluation of the augmented detector, operating in an environment of colored plus white additive noise.

All of the desired areas are investigated below. An expected result is that the coherent detector performance is degraded when carrier phase is estimated from the received data. An unexpected result is that a non-coherent version of the interference-tracking detection algorithm does not exist.

Recommendations are given on further research which may lead to improved performance of the complete IDEI receiver.
SECTION II
COHERENT DETECTION WITH PHASE ESTIMATION

1. SIGNAL AND CHANNEL MODEL

Figure 1 shows the overall model of the signal channel and signal processor. A continuous-time signal, \( s(t,m) \), is transmitted through the channel.

\[
s(t,m) = A(t;m) \cos[\omega_c t + \phi(t;m)]
\]

(1)

In (1), \( A(\ ) \) and \( \phi(\ ) \) are the envelope and phase functions, respectively. \( m \) denotes a digital symbol, which in the present work is restricted to the binary alphabet, \{0,1\}. Any arbitrary signal waveform may be represented in the form of (1).

The signal is subjected to additive colored and white noise, as per the figure. Then the bandpass signal plus noise process is translated to baseband in two separate channels, using coherent product detection with sinusoidal reference signals which are in phase and in phase quadrature with the unmodulated carrier signal. Following the I-Q demodulation, the two low-pass signal components of the signal vector are sampled to produce a discrete-time vector. The discrete-time signal is then processed further to recover the message symbol decisions, \( \hat{m} \).

The I-Q product demodulators require reference sinusoids having precise phase references, matched to the phase (zero) of the unmodulated carrier signal. Since this phase is A Priori unknown, the phase reference must be provided by the signal processor, itself, by phase estimation from the received data vector. The reference phase, so produced, is generally a function of time, \( \phi_0(t) \), as shown in Figure 2.

Since the signal phase is A Priori unknown, the signal model of (1) may be augmented with a random (or stochastic) phase term \( \phi_\Delta \) as

\[
s(t,m) = A(t;m) \cos[\omega_c t + \phi(t;m) + \phi_\Delta]
\]

\[
= s_i(t;m) \cos \omega_c t - s_q(t;m) \sin \omega_c t
\]

(2)

where

\[
s_i(t;m) = A(t;m) [\cos \phi(t;m) \cos \phi_\Delta - \sin \phi(t;m) \sin \phi_\Delta]
\]

\[
s_q(t;m) = A(t;m) [\sin \phi(t;m) \cos \phi_\Delta + \cos \phi(t;m) \sin \phi_\Delta]
\]

(3)
\[ s(t;m) = A(t;m) \cos[\omega_c t + \phi(t;m)] \]

\[ m \in \{0, 1, \ldots, M-1\} \]

- Coded or Uncoded
- \( A() \), \( \phi() \): Arbitrary Modulation

**Figure 1. Physical Channel and Receiver Models**
are the in-phase and quadrature low-pass components of the band-pass \( s(t;m) \).

The I-Q components of \( s(t;m) \) form a vector

\[
\begin{bmatrix}
\delta_i(t;m) \\
\delta_q(t;m)
\end{bmatrix} =
\begin{bmatrix}
\cos \phi_d & -\sin \phi_d \\
\sin \phi_d & \cos \phi_d
\end{bmatrix}
\begin{bmatrix}
A(t;m) \cos \phi(t;m) \\
A(t;m) \sin \phi(t;m)
\end{bmatrix} = \delta(t;m)
\]

Likewise, the additive colored and white noises may be written in terms of I-Q components as

\[
y(t) =
\begin{bmatrix}
y_i(t) \\
y_q(t)
\end{bmatrix} ;
n(t) =
\begin{bmatrix}
n_i(t) \\
n_q(t)
\end{bmatrix}
\]

where \( y(t) \) is the low-pass I-Q colored interference vector and \( n(k) \) is the I-Q data vector, \( z(t) \) may then be written as

\[
z(t) = s(t;m) + y(t) + n(t)
\]
The problem of detecting the digital symbol, $m$, in the presence of colored noise, white noise, and unknown signal phase is essentially the problem of processing $z(t)$ to make an optimum decision on $m$. This problem is analyzed in some detail below.

2. JOINT DETECTION WITH PHASE ESTIMATION

It is desired to reformulate the discrete-time recursive detection problem of [1] for the present case where the signal phase is unknown and time-varying. At this point it is still assumed that the symbol epoch, or timing, is known. The decision problem is based on processing the discretized I-Q data vector of (6). That is, a sequence of samples, $z(t_k)$ is processed recursively over the period of the binary symbol, $m$. Bit decision is made at the end of the symbol period. As in [1], decision direction is to be used from symbol to symbol, in order to preclude a processor size which would grow exponentially with symbol sequence length.

The assumed data generating model is that of Figure 3, wherein $z(k), n(k), a(k;m), \text{ and } y(k)$ are the sampled versions of $z(t), n(t), a(t;m), \text{ and } y(t)$, respectively, and $k$ is sample number. The colored interference process, $y(k)$, is generated from zero-mean, white, Gaussian, unit-variance noise (a two-vector), $W(k)$, which is independent of the channel noise, $n(k)$. The true structure of the $y(k)$ generator is the set $\{r, \phi, \lambda\}$ which may also be unknown. The problem of joint identification of $\{\Gamma, \phi, \lambda\}$ has been treated in Reference 1.

The decision on $m$ is to be made according to the maximum A Posteriori Probability (MAP) strategy. That is, a decision statistic, $S^T$ is to be formed recursively from the set of all data samples, $z(k)$, taken in sequence during the symbol period. Let $Z(k)$ denote the $2-K$ vector of $K$ samples of the I-Q data during the period.

$$Z(k) = [z(K), z(K-1), ..., z(1)]^T$$

$$= [z(K), ..., z(1)]$$

(7)
Figure 3. Data Generating Model

The MAP decision statistic is the probability

\[ S^1(K,m) = p(m|Z(K)) \]  \hspace{1cm} (8)

The decision rule is that the detected symbol, \( \hat{m} \), is that one for which \( S^1(K,\hat{m}) \) is maximum.

Assuming that the A Priori probability of transmitted symbols, \( p(m) \), is known, maximization of \( S^1(K,m) \) is obtained by just maximizing the Maximum Likelihood (ML) statistic, \( S(K,m) \), where

\[ S^1(K,m) = \frac{p(m)}{p(Z(K))} \cdot p(Z(K)|m) \]

\[ S(K,m) = p(Z(K)|m) \]  \hspace{1cm} (9)

Now, the signal, \( z(k;m) \), is a function of an unknown phase process, \( \phi_\delta(k) \), as per Eq. (4). Thus, define a K-vector, \( \phi(K) \), as
\[ \Phi(K) = [\phi_\delta(K), \phi_\delta(K-1), \ldots, \phi_\delta(1)]^T \]

\[
= \left[ \begin{array}{c}
\phi_\delta(K) \\
\vdots \\
\phi_\delta(K-1)
\end{array} \right] 
\tag{10}
\]

The unknown phase process, \( \Phi(K) \), is imbedded in the problem by using the composite detection approach, as

\[ p(Z(K)|m) = \int \cdots \int p(Z(K), \Phi(K)|m) d\phi_\delta(K) \cdots d\phi_\delta(1) \tag{11} \]

The ML decision statistic, \( S(K,m) \) is to be generated in recursive form. Thus, the argument of the integral in (11) is manipulated to obtain a recursive form.

We have

\[ p(Z(K), \Phi(K)|m) = \\
= p(Z(K), Z(K-1), \phi_\delta(K), \phi(K-1)|m) \\
= p(Z(K), \phi_\delta(K)|Z(K-1), \phi(K-1), m) \cdot \\
p(Z(K-1), \phi(K-1)|m) \\
= p(Z(K), \phi_\delta(K)|Z(K-1), \phi(K-1), m) \cdot \\
p(\phi(K-1)|Z(K-1), m) \cdot p(Z(K-1)|m) \tag{12} \]

Then,

\[ p(Z(K)|m) = \int \cdots \int p(Z(K), \phi_\delta(K)|Z(K-1), \phi(K-1), m) \cdot \\
p(\phi(K-1)|Z(K-1), m) \cdot p(Z(K-1)|m) d\phi_\delta(K) \cdots d\phi_\delta(1) \tag{13} \]

and

\[ S(K,m) = S(K-1, m)Q(K) \tag{14} \]

where

\[ Q(K) = \int \cdots \int p(Z(K)|\phi_\delta(K), \phi(K-1), Z(K-1), m) \cdot \\
p(\phi_\delta(K)|\phi(K-1), Z(K-1), m) \cdot p(\phi(K-1)|Z(K-1), m) \cdot \\
d\phi_\delta(K) \cdots d\phi_\delta(1) \tag{15} \]
Now, let us define $\hat{\phi}(\ell)$ to be the conditional-mean estimate of $\phi(\ell)$, given the data $z(k)$ for $k = 1, 2, \ldots, \ell$, and given the symbol, $m$. Then, $\hat{\phi}(\ell)$ maximizes $p(\phi(\ell)|Z(\ell), m)$. Now, it is assumed that the gradients of $p(z(K)|\phi(K), \hat{\phi}(K-1), Z(K-1), m)$ and of $p(\phi(K)|\hat{\phi}(K-1), Z(K-1), m)$, with respect to $\phi(K-1), \ldots, \phi(1)$, evaluated in the neighborhood of $\hat{\phi}(K-1)$, are sufficiently small so that the approximation may be made

$$Q(K) = \int p(z(K)|\phi(K), \hat{\phi}(K-1), Z(K-1), m) \cdot p(\phi(K)|Z(K-1), \hat{\phi}(K-1), m) d\phi(K)$$

This approximation says that the functions $p(z(K)|\phi(K), \hat{\phi}(K-1), Z(K-1), m)$, viewed as functions of the $\phi(K-1), \ldots, \phi(1)$, are sufficiently "flat" that $p(\phi(K-1)|Z(K-1), m)$ appears as a multi-dimensional delta function, centered at the co-ordinates, $\hat{\phi}(K-1), \ldots, \hat{\phi}(1)$. The multiple integral then simply evaluates the argument at those coordinates, analagous to "sifting" with a delta function.

Physically, the approximation means the following. If a sufficiently accurate conditional-mean estimate may be obtained for the phase process, $\phi(1), \ldots, \phi(K-1)$, then the density, $p(\phi(K-1)|Z(K-1), m)$, will have a very small variance about the mean estimate. Thus, the density $p(\phi(K-1)|Z(K-1), m)$ will be so highly concentrated that the densities, $p(z(K)|\phi(K), \hat{\phi}(K-1), Z(K-1), m)$ and $p(\phi(K)|Z(K-1), \hat{\phi}(K-1), m)$ will be flat by comparison. Thus, the accuracy of the approximation depends entirely on the availability of a very good phase estimate.

Similarly, now define $\hat{\phi}_\delta(\ell)$ to be the one-stage conditional-mean prediction of $\phi_\delta(K)$, given the previous data, $Z(\ell-1)$, the previous conditional mean estimate, $\hat{\phi}(\ell-1)$, and the symbol, $m$. As previously, assume that $\hat{\phi}_\delta(\ell)$ is sufficiently accurate so that $p(z(\ell)|\hat{\phi}_\delta(\ell), \hat{\phi}(\ell-1), Z(\ell-1), m)$ is flat, by comparison, in the neighborhood of $\hat{\phi}_\delta(\ell)$. This, then, yields the final approximation

$$Q(K) = p(z(K)|\hat{\phi}_\delta(K), \hat{\phi}(K-1), Z(K-1), m)$$

The recursive decision statistic is then

8
\[ S(K,m) = \prod_{k=1}^{K} Q(k) \]

\[ = \prod_{k=1}^{K} p(z(k)|\hat{\phi}(k), \hat{\phi}(k-1), Z(k-1), m) \quad (18) \]

It is seen from (17) and (18) that the recursive detector must form the conditional probability function, \( p(z(k)|\hat{\phi}(k), \hat{\phi}(k-1), Z(k-1), m) \), at each sample time (number) \( k \). Moreover, operating in parallel with the decision circuitry, and furnishing recursive phase estimates to it, is a conditional-mean phase estimator-predictor. The estimator produces the estimates

\[ \hat{\phi}(k) = E\{\hat{\phi}(k)|Z(k), m\} \]

The problem of conditional-mean estimation of the phase of a sinusoid in Gaussian noise is a non-linear estimation problem without a known general solution. However, the first-order approximate solution is known and is a phase-locked loop [2]. The closely related approximate Maximum A Posteriori Probability estimator is also a phase-locked loop [3]. Given the symbol, \( m \), and, hence, the corresponding signal waveform, \( s(t;m) \), the bandpass received data, \( z(t) \), consists of a sine wave of unknown (random) phase, imbedded in additive colored plus white Gaussian noise. Thus, the available solution to the estimation problem indicated by (14) is the decision-directed phase-locked loop. Note that the PLL is only the approximate solution to (19) for the case where the phase-estimation error is quite small. Thus, the optimality of the detection algorithm of (18) will depend on the phase estimation accuracy which may be realized in practice using the PLL.

3. THE I-Q DATA MODEL WITH PHASE ESTIMATION

In order to proceed with the detection and phase estimation algorithms, the discrete-time I-Q data generation model must be extended beyond that of equation (6) and Figure 3. Under the assumption that the I-Q demodulating reference sinusoid phases are estimated, the model changes somewhat. Let the physical model be shown in Figure 4.
In Figure 4, the transmitted signal with unknown phase is

\[ s'(t,m) = A \cos[\omega_c t + \phi(t,m) + \phi_\delta(t)] \] (20)

where \( \phi(t,m) \) is the angle modulation waveform, containing the symbol, \( m \).

The unknown, possibly time-varying, phase term is \( \phi_\delta(t) \). The additive, zero-mean, Gaussian colored and white noises are respectively,

\[ y'(t) = y'_i(t) \cos\omega_c t - y'_q(t) \sin\omega_c t \]
\[ n'(t) = n'_i(t) \cos\omega_c t - n'_q(t) \sin\omega_c t \] (21)

where the \( i \) and \( q \) subscripts denote "in-phase" and "quadrature" low-pass components, respectively.

The product detector reference sinusoids are

\[ s_{ri}(t) = 2 \cos[\omega_c t + \hat{\phi}_\delta(t)] \]
\[ s_{rq}(t) = -2 \sin[\omega_c t + \hat{\phi}_\delta(t)] \] (22)
where $\hat{\phi}_\delta(t)$ is the phase estimate of $\phi(t)$, provided by the phase-locked loop. The usual problem of the phase-locked loop responding to the low frequency portion of the modulation $\phi(t,m)$ may be encountered, depending on the exact form of the modulation.

Now define,

$$\hat{\phi}_\delta(t) - \phi(t) \triangleq \varepsilon(t)$$  \hspace{1cm} (23)

It may be shown that the low-pass I-Q data vector has the form

$$\begin{bmatrix} z_i(t) \\ z_q(t) \end{bmatrix} = \begin{bmatrix} \cos\varepsilon(t) & \sin\varepsilon(t) \\ -\sin\varepsilon(t) & \cos\varepsilon(t) \end{bmatrix} \begin{bmatrix} \cos\phi(t,m) \\ \sin\phi(t,m) \end{bmatrix} + \begin{bmatrix} y_i(t) \\ y_q(t) \end{bmatrix} + \begin{bmatrix} n_i(t) \\ n_q(t) \end{bmatrix}$$  \hspace{1cm} (24)

where

$$\begin{bmatrix} y_i(t) \\ y_q(t) \end{bmatrix} = \begin{bmatrix} \cos\hat{\phi}_\delta(t) & \sin\hat{\phi}_\delta(t) \\ -\sin\hat{\phi}_\delta(t) & \cos\hat{\phi}_\delta(t) \end{bmatrix} \begin{bmatrix} y'_i(t) \\ y'_q(t) \end{bmatrix}$$

$$\begin{bmatrix} n_i(t) \\ n_q(t) \end{bmatrix} = \begin{bmatrix} \cos\hat{\phi}_\delta(t) & \sin\hat{\phi}_\delta(t) \\ -\sin\hat{\phi}_\delta(t) & \cos\hat{\phi}_\delta(t) \end{bmatrix} \begin{bmatrix} n'_i(t) \\ n'_q(t) \end{bmatrix}$$  \hspace{1cm} (25)

With $n'(t)$ white, Gaussian, zero-mean with variance $\sigma_r^2$, then $n(t)$ is also white, zero-mean, with variance $\sigma_r^2$. This is because the multiplying matrix is a rotation matrix. However, $n(t)$ is not Gaussian, in general. For time periods which are short compared to the reciprocal bandwidth of $\hat{\phi}_\delta(t)$, $n(t)$ appears approximately Gaussian. With $\chi'(t)$ colored, Gaussian, zero-mean, with variance $\sigma_y^2$, $\chi(t)$ is zero-mean with variance $\sigma_y^2$. $\chi(t)$ is not Gaussian and may be of slightly greater bandwidth than $\chi'(t)$, if the variation of $\hat{\phi}_\delta(t)$ is not small.

The new data model of (24) may be written in three equivalent forms, and in discrete time, as
\[ z(k) = H[\phi_\delta(k)]H[\phi_\delta(k)]\hat{\phi}(k;m) + \chi'(k) + n'(k) \quad (26a) \]
\[ z(k) = H[\varepsilon(k)]\hat{\phi}(k;m) + \chi(k) + n(k) \quad (26b) \]
\[ z(k) = H[k;m]\hat{\phi}(k) + \chi(k) + n(k) \quad (26c) \]

where in (26)

\[ H[\varepsilon(k)] = \begin{bmatrix} \cos\varepsilon(k) & \sin\varepsilon(k) \\ -\sin\varepsilon(k) & \cos\varepsilon(k) \end{bmatrix} \quad A(k;m) = \begin{bmatrix} \cos\phi(k;m) \\ \sin\phi(k;m) \end{bmatrix} \]

\[ H[k;m] = \begin{bmatrix} \cos\phi(k;m) & -\sin\phi(k;m) \\ \sin\phi(k;m) & \cos\phi(k;m) \end{bmatrix} \quad \varphi(k) = \begin{bmatrix} \cos\varepsilon(k) \\ -\sin\varepsilon(k) \end{bmatrix} \]

\[ H[\phi_\delta(k)] = \begin{bmatrix} \cos\phi_\delta(k) & -\sin\phi_\delta(k) \\ \sin\phi_\delta(k) & \cos\phi_\delta(k) \end{bmatrix} \quad H[\phi_\delta(k)] = \begin{bmatrix} \cos\phi_\delta(k) & \sin\phi_\delta(k) \\ -\sin\phi_\delta(k) & \cos\phi_\delta(k) \end{bmatrix} \]

In (26b), the matrix \( H(k;m) \) is a function only of the signal. The vector, \( \varphi(k) \), is a function only of the phase-tracking error process, \( \varepsilon(k) \).

Detection of \( m \) in the presence of \( \varphi(k) \) is a multiplicative noise detection problem. The presence of the additive colored and white noise processes, \( \chi(k) \) and \( n(k) \), respectively, gives a compound detection problem, having multiplicative and additive colored noise.

The compound detection problem for multiplicative and additive colored Gaussian noise was solved in [4]. There it was found that the detector was one which tracked both the multiplicative and additive colored noises and attempted to remove them from the data, \( z(k) \). Although, in the present case, the various multiplicative and additive noises are not strictly Gaussian, the tracking detector may still be used. Note that when \( \varepsilon(k) \) is small then \( \varphi(k) \) is approximately

\[ \varphi(k) = A[\varepsilon(k)] ; \quad |\varepsilon(k)| << 1 \quad (28) \]

In this case, \( \varepsilon(k) \), the phase tracking error, is Gaussian and \( \varphi(k) \) is approximately Gaussian.

The final data generator diagram, corresponding to equations (26) is shown in Figure 5.
4. DETECTOR STRUCTURE AND ALGORITHMS

With the data generator given as in Figure 5, the tracking detector, with phase estimator, takes the form of Figure 6. In the detector, there are two decision-directed tracking filters, one implemented for the signal waveform corresponding to $m=0$, and the other for $m=1$. Each tracking filter is matched, in the Wiener sense, to both $p(k)$, the multiplicative noise, and $v(k)$, the additive noise. Thus, the detectors are implemented for the data, $z(k)$, in the form of equation (26c). The tracking error waveforms, $\xi(k;m)$, drive the decision circuitry which produces the decision on the received symbol as $\hat{m}$.

It was shown above that generally the phase estimator is decision-directed. However, a non-decision-directed phase estimator may be implemented if the transmitted signal possesses a residual unmodulated carrier component. This is shown as follows for a phase-shift-keyed signal.
Figure 6. Compound Detector and Phase Estimation
Suppose that the signal phase term is
\[
\phi(k;m) = \Delta \phi \cdot c(k;m) \quad ; \quad c(k;m) = 1; \quad m=0
\]
\[
= -1; \quad m=1
\]
\[
0 < \Delta \phi < \pi/2 \tag{29}
\]
Then
\[
\cos \phi(k;m) = \cos(\Delta \phi)
\]
\[
\sin \phi(k;m) = c(k;m) \cdot \sin(\Delta \phi) \tag{30}
\]
It follows that
\[
H[k;m]_\rho(k) = A \cos(\Delta \phi) \begin{bmatrix} \cos \varepsilon(k) \\ -\sin \varepsilon(k) \end{bmatrix} + c(k;m) \cdot \sin(\Delta \phi) \begin{bmatrix} \sin \varepsilon(k) \\ \cos \varepsilon(k) \end{bmatrix} \tag{31}
\]
From (31) it is seen that there is present in the received data an additive term proportional to \(-\sin \varepsilon(k)\), which may be used to drive the phase estimator. Likewise, there is an additive term proportional to \(\cos \varepsilon(k)\) which may be used to estimate \(A\) (coherent automatic gain control). The PSK waveform, \(c(k;m)\), is present in both I-Q channels, due to the multiplicative process with components \(\sin \varepsilon(k)\) and \(\cos \varepsilon(k)\). Provided that the bandwidth of \(c(k;m)\) is sufficiently wide and the closed-loop tracking bandwidth of the phase estimator is sufficiently small, the estimator can track phase in the presence of \(c(k;m)\) without decision-direction.

Each decision-directed tracking filter in Figure 6 is of the form of Figure 7. In the figure, the inner loop, composed of elements \(G_\rho, \phi_\rho, \Lambda_\rho\), and \(H[k;m]\), track the multiplicative process, \(\rho(k)\). The elements \(\{G_\rho, \phi_\rho, \Lambda_\rho\}\) are the elements of a Wiener filter in Kalman canonical form, matched to \(\rho(k)\). \(H[k,m]\) contains the signal waveform elements, as in (27). The outer loop tracks the additive colored interference, \(y(k)\). The elements, \(\{G_y, \phi_y, \Lambda_y\}\), are those of a Wiener filter matched to \(y(k)\). The filter algorithms are
It is seen from Figure 7 and (32) that the \( \hat{y}(k) \) filter and \( \hat{\rho}(k) \) filter are uncoupled, except for that coupling inherent in the pseudo-innovations, \( \xi(k) \). Filter design consists of selecting the two sets of parameters \( \{G_j, \Phi_j, \Lambda_j\} \) and \( \{G, \Phi, \Lambda\} \). The selection is based on either real-time identification of \( y(k) \) and \( \rho(k) \), as per [1], or on an ad hoc worst case design. The ad hoc design, while not optimum, would, under conditions discussed in [1], produce acceptable results.
5. THE PHASE ESTIMATOR

From equations (26c) and (31) we may write an expression for the (continuous-time) data vector, as seen by the phase estimator, as

\[ z(t) = A' \begin{bmatrix} \cos \epsilon(t) \\ -\sin \epsilon(t) \end{bmatrix} + \eta(t) \]  

(33)

In (33), \( \eta(t) \) is the total noise process due to \( y(t) \), \( \eta(t) \), and \( c(t;m) \). For the bandwidth of \( y(t) \) and the band-rate of \( c(t;m) \) sufficiently great with respect to the closed loop bandwidth of the phase estimator, the noise process, \( \eta(t) \), will appear white to the phase estimator.

It is seen that the problem of deriving the phase reference, \( \phi(t) \), which is an accurate estimate of the residual carrier phase, \( \phi_\delta(t) \), is that of minimizing \( \epsilon(t) \) in the presence of the unknown amplitude, \( A' \), and noise, \( \eta(t) \). This is, essentially, a phase-locked loop problem. Under the assumption that \( \eta(t) \) is white and Gaussian, the solution is the classical phase-locked loop.

Note that the usual problem of unknown signal amplitude, \( A' \), is present. There are two classical solutions. One is to use the Q-channel only, for phase estimation, with an ideal pre-limiter to remove dependence on \( A' \). The other solution is to also use the I-channel to estimate \( A' \) and to then control the gain of the Q-channel. An extension of the second method is shown in Figure 8.

In Figure 8, the Q-channel waveform, \( z_q(t) \) is processed by a "Loop Filter" with low frequency gain, \( H(0) \), to produce an estimate of the term, \( -A' \sin \epsilon(t) \), weighted by \( H(0) \). The I-channel waveform, \( z_i(t) \), is processed by a low-pass filter with unit low frequency gain to produce an estimate of the term, \( A' \cos \epsilon(t) \). The two filter output terms are then divided point-wise in a digital divider to provide an estimate of \( -\tan \epsilon(t) \), weighted by \( H(0) \). The latter estimate then drives the Voltage-Controlled-Oscillator (VCO) to produce the reference phase, \( \phi_\delta(t) \). It can be seen from the defining equation (23) for \( \epsilon(t) \) that the mechanization of Figure 8 causes \( \phi_\delta(t) \) to track \( \phi_\delta(t) \).
The usual phase-locked loop generates a tracking error voltage proportional to (-sine(t)). The present implementation provides a tracking error proportional to (-tanE(t)), which will yield higher loop gain for a large tracking error, E(t). However, the main reason for using the "Tangent-Loop" mechanization is to obtain the automatic gain control feature in the cancellation of the unknown amplitude, A'.

The design of the loop parameters, notably the loop filter, is performed by assuming linear operation of the loop. That is, when E(t) is small, say less than 12° in magnitude, then the approximation holds

\[ \tan E(t) = \sin E(t) = e(t) \]  \hspace{1cm} (34)

Then, the overall system operates as a linear servo-mechanism for phase, or as a linear phase-locked loop.

In the usual implementation, the Loop Filter in the quadrature channel is implemented with one finite zero of transmission and one finite, non-zero, pole. The pole frequency, zero frequency, and low-frequency gain, H(0), are set to realize the desired closed-loop noise bandwidth, static phase error for VCO frequency offset, and second order dynamic response. The low pass filter in the I-channel is set for the same zero and pole frequencies as for the Q-channel Loop Filter, but with unit low-frequency gain.

Note that for the PLL to operate properly, the signal to noise ratio must be large in the closed-loop equivalent noise bandwidth of the loop, itself. The PLL bandwidth is to be maintained small enough to just
accomodate the dynamics of the received signal phase, $\phi(t)$, due to Doppler effects on the transmission link. For the case where the incident noise is dominated by colored interference, such as jamming, the loop performance will be affected by that portion of the colored interference falling within the (narrow) loop bandwidth.

6. THE LOOP FILTER MECHANIZATION

The continuous-time version of the Loop Filter is characterized by the transfer function

$$H(s) = K\left[\frac{s-z}{s-p}\right]$$

(35)

where $K$, $z$, and $p$ are real, with $z$ and $p$ being negative. Let $\mu(t)$ and $z(t)$ denote the filter input and output, respectively. A state variable representation is set up, using the single filter state, $x(t)$, as

$$\dot{x}(t) = px(t) + \mu(t)$$

$$z(t) = K(p-z)x(t) + K\mu(t)$$

(36)

The filter is converted to discrete time by driving it with an ideal sampler and zero-order hold circuit and observing the output only at sampling instants, $t = t_k$ for $k = 1, 2, 3, \ldots$. The differential equation of (36) is then solved between the $k$th and $(k+1)$st sampling times as

$$x((k+1)T) = \exp[p((k+1)T-kT)] \cdot x(kT)$$

$$+ \int_{kT}^{(k+1)T} \exp[p(k+1)T - \tau]W(\tau)d\tau$$

(37)

where

$$W(t) = \mu(kT); \quad kT \leq t < (k+1)T$$

(38)

and $T$ is the sampling interval. The differential equation solution then yields the governing difference equation (discrete-time) for the filter as
\[
x(k+1) = \phi x(k) + \gamma u(k)
\]

\[
z(k) = K(p-z)x(k) + K\mu(k)
\]

\[
\phi = \exp(pT) : \gamma = 1/p(\phi-1)
\]

(39)

The Loop Filter constants, \(K, z, p,\) are set according to specifications on the linearized closed-loop transfer function for phase. The VCO output phase, \(\phi(t)\) is given by

\[
\hat{\phi}_\delta(t) = \int [-[\phi_\delta(t) - \hat{\phi}_\delta(t)]h(t)]dt \quad (40)
\]
or

\[
\hat{\phi}_\delta(s) = -\frac{[\phi_\delta(s) - \hat{\phi}_\delta(s)]H(s)}{s} \quad (41)
\]

where \(H(s)\) is the Loop Filter transfer function given in (35).

The closed-loop transfer function for the PLL is then

\[
G(s) = \frac{\hat{\phi}_\delta(s)}{\phi_\delta(s)} = \frac{H(s)}{s + H(s)} \quad (42)
\]

Substituting for \(H(s)\) yields

\[
G(s) = \frac{K(s-z)}{s^2 + (K-p)s - Kz} = \frac{K(s-z)}{s^2 + 2\sigma_\eta s + \omega_n^2} \quad (43)
\]

where \(\sigma\) and \(\omega_n\) are the classical damping ratio and resonant frequency for a second-order servo system.

The Loop Filter low frequency gain, \(H(0)\), is given by

\[
H(0) = \lim_{s \to 0} \frac{K(s-z)}{s} = \frac{Kz}{p} \quad (44)
\]

For most PLL designs the following assumptions hold

\[-z << H(0)\]

\[-p << K\]  \quad (45)

Thus, by equating like terms in the denominator of (43)
\[ K = 2\delta \omega \]  
\[ -Kz = \frac{\omega}{\omega_n} \]  

(46)

Now, it may be shown that the one-sided closed-loop noise bandwidth, in Hz, for \( G(s) \) is [5]

\[ B_n = \frac{K}{4(K-p)} = \frac{K-z}{4} = \frac{\omega}{8\delta} \left( 1 + 4\delta^2 \right) \]  

(47)

Thus,

\[ K = \left[ \frac{16 \delta^2}{1 + 4 \delta^2} \right] \cdot B_n \]

(48)

For loop dynamic stability, the damping ratio is set as

\[ \delta = \frac{1}{\sqrt{2}} \]  

(49)

Then

\[ K = \frac{8}{3} B_n \]

\[ z = -\frac{4}{3} B_n = -\frac{K}{2} \]  

(50)

The Loop Filter pole frequency, \( p \), is generally set as small as possible in magnitude. This is because \( p \) affects the "static phase error" when tracking with a fixed Doppler offset in the received frequency. In order to hold the loop in lock when the input phase \( \phi(t) \) has a constant first derivative requires a constant driving voltage into the VCO and hence a constant phase error, \( \varepsilon(t) \). Thus,

\[ \frac{d}{dt} \hat{\phi}(t) \hat{\phi} \Delta \omega = -H(0) \tan \varepsilon_{sp} \]  

(51)

where \( \varepsilon_{sp} \) is the static phase error for a Doppler offset, \( \Delta \omega = 2\pi \Delta f \). The d.c. gain of the loop filter is
\[ H(0) = k \frac{z}{p} = \frac{32}{9} \frac{B_n^2}{|p|} \]  

(52)

For desired small values of static phase error

\[ 2\pi \Delta f = \frac{32}{9} \frac{B_n^2}{|2f_p|} \cdot \varepsilon_{sp} \]  

(53)

where \( f_p \) is the Hertz value of \(-p\). Thus,

\[ f_p = \frac{32}{9(2\pi)^2} \frac{B_n^2 \varepsilon_{sp}}{\Delta f} \]  

(54)

Equation (54) gives the relation between the various quantities and \( f_p \).

Thus, the design equations for the quadrature channel loop filter are

\[
K_q = \frac{8}{3} B_n
\]

\[
z = -\frac{4}{3} B_n \quad ; \text{quadrature filter} \]  

\[
p = -\frac{B_n^2 \varepsilon_{sp}}{18 \Delta f} \]

where \( \varepsilon_{sp} \) is static phase error in radians for a Doppler offset of \( \Delta f \) Hertz and a closed loop noise bandwidth of \( B_n \) Hz.

For the inphase filter, the same pole, \( p \), and zero, \( z \), are used, but the d.c. gain is reduced to unity to give a filter gain constant

\[
K_i = \frac{p}{z} \quad ; \text{inphase filter} \]  

(56)

The block diagram of the phase estimator is given in Figure 9. In the figure, the discrete time version of the VCO (phase integrator) is represented by

\[
\hat{\phi}_d(k+1) = \hat{\phi}_d(k) + T/2[v(k+1) + v(k)] \]  

(57)

where \( v(k) \) is the VCO input.
Figure 9. Discrete-Time Phase Estimator
SECTION III
ON THE EXISTANCE OF NON-COHERENT TRACKING DETECTORS

It is desired now to determine if a non-coherent version of the tracking detector exists. In [1] the non-coherent version of the standard FSK detector for white noise was derived. The approach for the tracking detector will be similar. An unknown constant phase term will be introduced into the formulation of the detection problem. Then, the detection statistic will be averaged with respect to the unknown phase. Up to this point, the procedure is the same as was followed in II.2. That is, the problem is that of composite detection for unknown phase. In II.2 there existed a solution of the composite detection problem which produced a phase estimator as part of the detector. In the present formulation, the phase estimator solution is purposely rejected and no attempt is made to take advantage of possible phase information. Rather the unknown phase is defined to be uniformly distributed over the interval, \([0, 2\pi]\), and to be a constant random variable over the time interval of the signal symbol. Then it is to be determined whether averaging the decision statistic over phase produces a sufficient statistic for detection.

The unknown phase enters the problem as per Figure 2, where now \(\phi_0(t)\) is defined to be constant over the symbol interval, which is also the processing time. Also \(\phi_0(t)\) is uniformly distributed as

\[
\begin{align*}
\phi_0(t) &= \phi & 0 \leq t \leq T \\
p(\phi) &= \frac{1}{2\pi} & 0 \leq \phi \leq 2\pi \\
&= 0 & \text{otherwise}
\end{align*}
\]

(58)

The discrete time data model, \(z(k)\), is essentially that of (26a) where \(\hat{\phi}_0(k) = \phi_0\) and \(\phi_0(k) = 0\). Thus,

\[
z(k) = H(\phi)[\hat{\phi}(k) + \chi(k) + n(k)]
\]

(59)

where \(\hat{\phi}(k)\) is the transmitted signal, \(\chi(k)\) is the colored interference, and \(n(k)\) is the white noise.

The detection statistic, \(S(K)\), is formed recursively from the \(z(k)\), and is the Maximum A Posteriori Probability function, \(p(m|z(K))\), where \(Z(K)\) is the \(2K\) partitioned vector,

\[
\text{24}
\]
\[ Z(K) = [z(K), z(K-1), \ldots, z(1)]^T \]  

(60)

The quantity, \( m \), is the signal digit, which for the binary case is either 0 or 1. Under the assumption that the transmitted digits, \( m \), are equally distributed \( (p(m) = 1/2; m=0,1) \), the MAP statistic is equivalent to the Maximum-Likelihood (ML) statistic, \( p(Z(K)|m) \). Thus, \( S(K) \) is obtained by averaging the joint density on \( Z(K) \) and \( \phi \), given \( m \).

\[ S(K) = p(Z(K)|m) = \int_0^{2\pi} p(Z(K), \phi|m) d\phi \]

(61)

The conditional density, \( p(Z(K)|m, \phi) \) is

\[ p(Z(K)|m, \phi) = \prod_{k=1}^{K} p(z(k)|Z(k-1), m, \phi) \]

(62)

Now, \( p(z(k)|Z(k-1), m, \phi) \) is Gaussian, under the definition that \( y(k) \) and \( n(k) \) are Gaussian, and is given by

\[ p(z(k)|Z(k-1), m, \phi) = \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left[ -\frac{1}{2\sigma^2} (z(k) - \hat{z}(k|k-1, m, \phi))^T (z(k) - \hat{z}(k|k-1, m, \phi)) \right] \]

(63)

In (63), \( \sigma^2 \) is the steady-state Innovations variance and \( \hat{z}(k|k-1, m, \phi) \) is the recursive estimate of the \( k \)th data sample, given all the data up through the \( (k-1) \)st sample. This one-sample predictive estimate is obtained from the Kalman-form filter of Figure 10. In the figure, the quantities, \( \{\phi, A, G\} \), are the appropriate Kalman (Wiener) filter parameters for tracking \( y(k) \), the colored interference, in the presence of \( n(k) \), the white noise.
Figure 10. Kalman Filter

The filtering algorithms are

$$
\hat{z}(k|k-1) = H(\phi)[\hat{z}(k) + \hat{y}(k|k-1)] = H(\phi)\hat{z}(k) + H(\phi)\Lambda\hat{x}(k-1)
$$

$$
\hat{x}(k) = \Psi(\phi)\hat{x}(k-1) + \mu(k,\phi); \Psi(\phi) = [I - GH(\phi)A]\phi
$$

$$
\mu(k,\phi) = G[z(k) - H(\phi)\delta(k)]
$$

(64)

The solution to (64) at the kth sample is given by

$$
\hat{z}(k|k-1, m, \phi) = H(\phi)[\hat{z}(k;m) + \Lambda\Phi[\Psi^{k-1}(\phi)\hat{x}(0) + \sum_{i=1}^{k-1} \Psi^{i-1}(\phi)\mu(k-i,\phi)]]
$$

(65)

It is seen at this point that any hope of averaging $p(\hat{Z}(K)|m, \phi)$ over $\phi$ is futile due to the internal dependency of $\hat{z}(k|k-1, m, \phi)$ on $\phi$. That is, it is the feedback dependency of the estimate $\hat{y}(k|k-1, m, \phi)$ upon $\phi$ which defeats the prospect of averaging over $\phi$. 

26
There is a second possibility for a noncoherent implementation. The term, \( H(\phi)y(k) \), in the data model of (59) is not strictly Gaussian, but does have the same first and second moments as \( y(k) \), since \( H(\phi) \) is unitary. Also, since \( \phi \) is constant over a symbol period, \( H(\phi)y(k) \) has the same short-term spectral properties as \( y(k) \). Thus, the data form may be re-defined as
\[
\tilde{z}(k) = H(\phi)\Delta(k;m) + y(k) + n(k) ; \quad 1 \leq k \leq K
\] (66)

where \( y(k) \) and \( n(k) \) have replaced \( H(\phi)y(k) \) and \( H(\phi)n(k) \), respectively. In (66), \( y(k) \) and \( n(k) \) are taken as Gaussian.

The resulting Kalman estimator for \( \hat{z}(k|k-1, m, \phi) \), corresponding to the data model of (66) is as in Figure 11.

The filtering algorithms now are
\[
\hat{z}(k|k-1, m, \phi) = H(\phi)\Delta(k;m) + \hat{y}(k|k-1, m, \phi)
\]
\[
= H(\phi)\Delta(k;m) + \Delta\hat{x}(k-1)
\]
\[
\hat{x}(k) = \Psi\hat{x}(k-1) + \mu(k;\phi) ; \quad \Psi = (I-\Delta\Lambda)\Phi
\]
\[
\mu(k;\phi) = G[z(k) - H(\phi)\Delta(k;m)]
\] (67)
The solution to (67) is

$$\hat{z}(k|k-1, m, \phi) = H(\phi)\hat{\delta}(k;m) + \Lambda\Phi[\psi^{k-1}\hat{x}(0) + \sum_{j=1}^{k-1} \psi^{j-1} \mu(k-j, \phi)]$$

(68)

Now, (68) is somewhat of an improvement over (65) in that $\Psi$ is no longer a function of $\phi$. Unfortunately, $\mu(\cdot)$ is still dependent on $\phi$ and this causes the dependency of $\hat{z}(k|k-1, m, \phi)$ on $\phi$ to be internal because of the feedback structure of the filter. Thus, averaging $p(Z(K)|m, \phi)$ over $\phi$ is still not feasible.

The argument of the exponent of $p(z(k)|Z(K-1), m, \phi)$ in (63) is

$$\text{Arg} = (z(k) - \hat{\gamma}(k|k-1, m, \phi))^T(\cdot) + \hat{\delta}(k;m)\hat{\delta}(k;m)$$

$$- 2\hat{\delta}(k;m)H(\phi)[z(k) - \hat{\gamma}(k|k-1, m, \phi)]$$

(69)

This argument is of the same form as is encountered in the standard non-coherent FSK detector problem [1], except that $(z(k) - \hat{\gamma}(k|k-1, m, \phi))$ has replaced $z(k)$. Were it not for the fact that $\hat{\gamma}(k|k-1, m, \phi)$ is an explicit function of $\phi$, as in (67), then the averaging over $\phi$ would be exactly the same as in the FSK problem. Unfortunately, there seems to be no further recourse to the problem at this point.
SECTION IV
SIMULATION RESULTS

A Monte-Carlo simulation program was written to obtain error-rate results for coherent detection with phase estimation. The detection algorithm which was implemented was that detailed in Section II.4. The program realized the compound detector and phase estimator of Figure 6 where the tracking filters were of the form given in Figure 7. The phase estimator was the Tangent Phase-locked loop shown in Figure 9.

In order to reduce simulation run times, the Monte Carlo program, documented in [4], was not modified for present use. Rather, an entirely new program was written. In the new program, the data generator, shown in Figure 5, was reduced from three states, as in [4], to one state. This resulted in the Kalman filters also having one state in each branch shown in Figure 7. Since computation load increases exponentially with state size, a considerable savings was made. All that was lost was some flexibility in modeling the additive colored noise process. For the purposes of the present work, the one-state model was sufficient.

It was desired to test the compound detector and phase estimator in a realistic but stressful environment. Thus, a phase-locked loop noise bandwidth of 2.5 Hz was chosen as being as small as could likely be realized in a reasonable implementation. It was desired to run the phase-locked loop at 0.3 radians r.m.s., phase error, or less. Thus, it was necessary to relate the various simulation parameters, such as $E/N_0$, colored noise bandwidth, etc., to the phase-locked loop signal to noise ratio.

Letting $J$ denote the power of the colored process, $y(k)$, (in bandpass form) and $B_J$ the one-sided equivalent noise bandwidth of the low-pass I-Q process, an equivalent white bandpass spectral density, $N_J$, for the colored process is defined by

$$ J = N_J \cdot 2 B_J $$

Then, the total equivalent white noise spectral density is

$$ N_T = N_0 + N_J $$

29
where $N_0$ is the density of the incident additive white receiver noise.

The symbol energy, $E$, in the received signal is related to total signal power, $S$, and symbol period, $T$, by

$$E = S \cdot T \cdot L_M(\Delta \phi)$$

(72)

where $L_M(\Delta \phi)$ is the "modulation loss" factor given by

$$L_M(\Delta \phi) = \sin^2(\Delta \phi)$$

(73)

where $\Delta \phi$ is phase deviation in radians for the phase-shift keyed signal.

Thus,

$$S = \frac{E}{L_M(\Delta \phi) \cdot T}$$

(74)

From (70) and (74) results

$$\frac{S}{J} = \frac{E}{L_M(\Delta \phi) \cdot T \cdot N_J \cdot 2B_J}$$

(75)

Now,

$$\left(\frac{E}{N_0}\right)N_O = E = \left(\frac{S}{J}\right) \cdot L_M(\Delta \phi) \cdot (2\frac{B_J}{R}) \cdot N_J$$

(76)

where $R = 1/T$ is symbol rate. Thus,

$$N_J = \frac{(E/N_O)}{L_M(\Delta \phi) \cdot (\frac{S}{J})} \cdot (2\frac{B_J}{R}) \cdot N_O$$

(77)

and

$$N_T = [1 + \frac{(E/N_O)}{L_M(\Delta \phi) \cdot (\frac{S}{J})} \cdot (\frac{R}{2B_J})]N_O$$

(78)

It is desired to compute the ratio of residual carrier power to total white noise power in the Loop-noise bandwidth (one-sided), $B_N$. The residual carrier power, $S_C$ is

$$S_C = L_C(\Delta \phi)S = \frac{L_C(\Delta \phi)}{L_M(\Delta \phi)} \cdot \frac{E}{T} = \frac{RE}{\tan^2(\Delta \phi)}$$

(79)
where $L_C(\Delta \phi)$ is "carrier loss" given by

$$L_C(\Delta \phi) = \cos^2(\Delta \phi)$$  \hspace{1cm} (80)

The desired signal to noise ratio is

$$\frac{S_C}{N}B_N = \frac{S_C}{N}T_NB_N = \frac{(R/B_N) \cdot (E/N_0)}{(E/N_0) \cdot \tan^2(\Delta \phi) \cdot \left[ 1 + \frac{R}{2B_N} \right]}$$  \hspace{1cm} (81)

Note that when the equivalent white spectral density of the colored interfering process is much larger than the receiver white noise spectral density, then (81) reduces to

$$\frac{S_C}{N}B_N = 2 \cos^2(\Delta \phi) \cdot \left( \frac{B_j}{B_N} \right) \cdot \left( \frac{S}{J} \right)$$  \hspace{1cm} (82)

The loop phase error variance, under the assumption that the loop is operating linearly for phase (large loop signal to noise ratio), is given by

$$\sigma^2_\phi = \frac{1}{S_C \cdot \frac{N}{B_N}} \text{ radians}^2$$  \hspace{1cm} (83)

and, from (83) and (81),

$$\sigma^2_\phi = \tan^2(\Delta \phi) \left[ \frac{1}{(R/B_N) \cdot (E/N_0)} + \frac{1}{2} \cdot \left( \frac{B_j}{B_N} \right) \cdot \left( \frac{S}{J} \right) \right]$$  \hspace{1cm} (84)

Figure 12 shows simulation results for the case of narrow-band interference for binary phase-shift-keying (PSK). The equivalent square bandwidth of the colored interference process is 275 HZ. The signal symbol rate is 2500 baud. Thus the "bandwidth to bit-rate ratio" is $BW/BR = 0.109$. This is the same case for which extensive previous results were reported.
SIMULATED ERROR RATE - COLORED INTERFERENCE

2500 Symbols/Second  10 Samples/Symbol
Binary PSK, Mod. Index = 0.785
Interference Bandwidth = 275 Hz.: BW/BR = 0.109
Multiplicative Noise Tracking Bandwidth = 2.5 Hz.
Phase-Locked Loop Bandwidth = 2.5 Hz.
Phase Jitter = 5 degrees, r.m.s.
Perfect Identification

SYMBOL ENERGY TO NOISE SPECTRAL DENSITY RATIO, E/N₀, dB.

Figure 12. Simulation Results
For the case of Figure 12, the ratio of signal power to additive colored noise is unity, or zero dB. A phase-locked loop is implemented, as described in Sections II.5. and II.6. The loop noise bandwidth is 2.5 Hz, being the smallest assumed to be practical for this case. The detector employs both additive noise tracking and multiplicative noise tracking with the latter matched to a multiplicative noise bandwidth of 2.5 Hz. Perfect identification is assumed for the colored additive noise.

From (84), the predicted value of loop phase jitter is determined to be 5.4° r.m.s. The actual r.m.s. values recorded in the simulation were between 1.7° and 9.6° for runs up to 1500 symbols in length. The loop was observed to always be in lock, slipping no cycles during any run.

The results plotted in Figure 12 include the reference graphs of coherent PSK detection for white noise only, and IDEI detection with perfect phase reference. Also, is shown the behavior of the standard discrete-time matched filter detector. The matched filter is seen to saturate at an error rate of 0.14, as usual [1]. The IDEI detector is seen to yield a convex error rate curve for $-10 \text{ dB} \leq E/N_0 \leq 20 \text{ dB}$. However, for $20 \text{ dB} < E/N_0$, the slope of the error rate curve becomes much less steep. Although the error-rate continues to decrease for increasing $E/N_0$, the rate of decrease is not as good as was obtained for "pure" multiplicative noise in [4].

The implications (or "cause") of the change in slope of the error rate curve for $20 \text{ dB} < E/N_0$ are, at present, unknown. Clearly, there is a transition at $E/N_0 = 20 \text{ dB}$ for the case shown. It has been observed in the past that such transitions may be due to the breakdown of basic modeling assumptions on which the "optimum" detector is founded. One such questionable assumption which is suspect here is that the multiplicative noise process, due to carrier-tracking phase error, is Gaussian. Also, it may be that the phase-tracking detection algorithm is subject to an irreducible error-rate, as detailed in [8]. It is noted that the IDEI detector for multiplicative noise has not previously shown such an irreducible error.

In conclusion, this simulation for the SJR = 0 dB case shows that much of the performance measured previously for perfect phase is lost, when a standard phase-locked loop is used in parallel with the IDEI detector. It
is recalled that this implementation is not the true optimum, for two reasons. One is the Gaussian multiplicative noise approximation. The second is that the phase-tracking loop is external to the detector and, thus, does not take advantage of the colored noise tracking capability of the detector itself. It may well be that a more optimum implementation will result by imbedding the phase-estimation algorithm within the detector itself.
SECTION V
COMPLETE RECEIVER ALGORITHMS

1. A PROPOSED BIT SYNCHRONIZATION ALGORITHM

So far in the investigation of IDEI detection, it has been assumed that bit timing information is available. This is important for the detector in terms of setting the start and stop times of the computation which produces the decision statistic, $S(K)$. However, now the synchronization problem is finally examined.

Many practical bit synchronizers are based on the "Delay-lock Loop," [6, 7]. This technique applies to any coherent signalling scheme, but is generally used for phase-shift-keying (PSK). Generally, the implementation uses two signal cross-correlators driven with time-staggered signal reference waveforms. The correlator outputs are time-staggered versions of the noisy signal autocorrelation function. By subtracting the staggered autocorrelation functions, a tracking error function is produced which drives the reference generator into bit synchronism with the receiver signal.

The key to the functioning of the delay-lock bit synchronizer is the production of a signal (from the correlator output) which is a positive, even function whose maximum occurs when the reference generator is in synchronization with the received bit. Those positive even functions (autocorrelation functions) also happen to be the sufficient statistics for detection for the standard detectors which use delay-lock bit synchronization.

In the IDEI detector, the sufficient statistic for detection is the pseudo-innovations process, or noise tracking error. It was seen in [1] that there was associated with the statistic a function which was positive, with minimum value occurring for perfect identification of the required noise statistics. With "positive" or "negative" identification errors (in the sense of Figures 36 and 43 of [1]), the function value increased. The function was the variance of the noise tracking error.

Now, it is conjectured that the IDEI tracking error variance, which is necessarily positive, will be minimum for the reference signal, $\delta(k;n)$, exactly synchronized with the received bit. It is also conjectured that the variance will increase as the reference, $\delta(k;n)$, becomes unsynchr-
nized, regardless of whether \( \delta(k; n) \) leads or lags the received bit. If this conjecture proves true, then it is a simple matter to use the reciprocal of the tracking error variance in the same fashion that the Delay-Lock Loop uses the autocorrelation function, to form a synchronization tracking error function.

2. THE COMPLETE ALGORITHM

The complete IDEI algorithm (excluding identification) can be postulated as follows, for binary signalling. See Figure 13. Two IDEI detectors, with imbedded phase estimators are implemented, one with early waveform reference signals and one with late. Each detector contains two tracking filters of the form of Figure 7. In each detector are produced the detection statistics, \( S_0 \) and \( S_1 \), which are the tracking error variances, conditioned on the two different received symbols, \( m=0 \) and \( m=1 \), respectively. In each detector, symbol decision is made as usual. Based on the symbol decision, the assumed correct tracking error variances, \( \hat{S}_e \) and \( \hat{S}_l \), are produced by the early and late detectors, respectively. The reciprocal of each variance is taken and the results subtracted to form a "Synch Control" driving signal, which is filtered with suitable gain and time constant. A modulo-2 adder is implemented to determine if the decisions in the early and late detectors do not result in the same detected symbol. If not, the synch. control signal is inhibited, and synch is maintained as previously. Decision-directed reinitialization of the filters is carried out in the usual manner, independently in the early and late detectors.
SECTION VI
CONCLUSIONS

The research documented in this report has yielded several interesting results. These are summarized below in the order of the governing tasks in the Contract Statement of Work.

Task 4.
The IDEI (interference-tracking) detection algorithms were extended to include provision of the required carrier phase reference through phase tracking. A separate phase-locked Loop was implemented, processing the received data in parallel with the detection algorithm itself. The detection algorithm was augmented to track the multiplicative noise resulting from the phase reference variations, as well as tracking the colored additive noise.

Task 5.
It was shown analytically that a non-coherent version of the IDEI detection algorithm does not exist. This result is due to the feedback structure inherent in the IDEI tracking filter. The internal dependency of the detection statistic on the unknown phase makes it impractical to carry out the phase averaging necessary to obtain a non-coherent type algorithm.

Task 6.
Based on the result of Task 5, a non-coherent IDEI detector for Differential Phase Shift Keying is also impractical of derivation.

Task 7.
A bit synchronization technique was proposed, based on the Early-Late method. This bit synchronization scheme then led to the postulation of a complete receiver algorithm including interference-tracking, phase estimation, and bit synchronization. A block diagram of the algorithm was given.

Task 8.
The Monte Carlo simulation routine used and reported previously [1, 4] was restructured and re-written. The routine was simplified con-
considerably and was augmented to accommodate the new detection and phase-tracking algorithms. The chief reason for this effort was to achieve shorter run times in line with restrictions imposed by the ASD Computer Facility (CDC-6600).

The performance of the IDEI detector with phase tracking was evaluated. It was found that the performance was considerably degraded over previous results for perfect phase references. Two possible causes for the degradation were discussed.

In summary, further research on the IDEI algorithms is recommended in the following areas. Most importantly, a method of phase estimation should be sought wherein the phase estimator is imbedded in the interference tracking filter. The purpose is to reduce the effects of the large additive colored noise upon the phase estimator. Rather than tracking phase in parallel with the colored noise tracking filters, phase should be tracked after the colored noise has been removed from the data. Secondly, further effort should be devoted to optimizing the multiplicative noise tracking filter for the non-Gaussian perturbations produced by the phase variations. Finally, the proposed bit synchronization algorithm should be studied and evaluated.
APPENDIX A

THE CLOSED-FORM ERROR-RATE PROGRAM

(This appendix contains listings of the newly written simulation program and the closed-form error-rate evaluation program reported previously.)
C THIS IS MAIN PROGRAM FOR THE CLOSED-FORM ERROR RATE FOR
C IMPERFECT IDENTIFICATION WHICH IS A EXTENSION OF PROGRAM YOONM5
C ** REQUIRED SUBROUTINE **
C (1) RES3 ; MAIN, DATA, INPUT1, INPUT2, PARALL, PREPAR
C (2) CFERAT ; CFERAT, WKFLT, ERF
C (3) VTT ; VTT, CAYLEY, GAUS
C (4) EIGEN
C (5) COMAT
C REMARK
C (1) CHECKING THE CLOSED-FORM ERROR RATE FOR PERFECT
C IDENTIFICATION, SET IMODE 1 AVOIDING THE SAME EIGEN-VALUE
C IN SUB. CAYLEY
C (2) TO GET THE STEADY-STATE KALMAN GAIN, SET KSMAX 50-100
C IN GENERAL
C (3) ESTIMATED TRANSITION MATRIX PHEER AND DPHEE ARE VARIED
C IN SUBROUTINE INPUT1 AND ESTIMATED KALMAN GAIN GSTAR IS
C VARIED IN SUBROUTINE INPUT2 EACH TIME.
C PROGRAMMER
C CHANG-JUNE YOON
C ELECTRICAL ENGINEERING DEPT.
C TEXAS A & M UNIVERSITY
C COMMON/ORDER/N, N2
COMMON/SAMPLE/NSPB, TB, TBR
COMMON/OPTION/NOS, AEST
COMMON/RATIO/ENODB, ENODBR, SJRDB, SJRDBR
COMMON/GDB/ZN, GNR, GJ, GJR
COMMON/HORNOM/IMODE, KSMAX, IOJ
COMMON/FREQ/FZ, FP(3)
COMMON/PARAM/GAMMA(6, 2), PHEER(6, 6), H(2, 6), Q(2, 2), R(2, 2)
COMMON/PARAMR/PHEER(6, 6), DPHEE(6, 6), GSTAR(6, 2), BSTAR(2, 2)
CALL ASSIGN(S, 'SY: RES3. DAT', 11, 'RDO', 'NC', 1)
CALL DATA
C NOPTN1
1. NO CHANGE
2. CHANGE ENODB
3. CHANGE SJRDBR
4. CHANGE NSPB, GK
C NOPTN2
1. NO CHANGE
2. CHANGE ENODB
3. CHANGE SJRDBR
4. CHANGE GK
5. CHANGE SJRDB, SJRDBR
IF NOPTN1, NOPTN2 IS 1, THEN NCASE1, NCASE2 IS 1 RESPECTIVELY
NCASE1 ; NUMBER OF CASE FOR NOPTN1
NCASE2 ; NUMBER OF CASE FOR NOPTN2
IPARAM
0. NO PRINT-OUT PARAMETERS AND STATISTICS IN INPUT1 AND INPUT2
1. PRINT-OUT
IGV
0. NO CALCULATION KALMAN GAIN FOR A CORRECT PARAMETERS INPUT1.
1. CALCULATION.
READ(5, 701) NOPTN1, NOPTN2, NCASE1, NCASE2, IPARAM, IGV
701 FORMAT(615)
READ(5, 702) GK
702 FORMAT(E15.6)
DO 2000 II=1, NCASE1
99 TO (1, 2, 3, 4), NOPTN1
1 GO TO 50
2 READ(5, 705) ENODB
ENODBR=ENODB
GO TO 50
3 READ(5,705) SJRDBR
GO TO 50
4 READ(5,707) NSPB, QK
50 CONTINUE
705 FORMAT(E15.6)
706 FORMAT(2E15.6)
707 FORMAT(I5,E15.6)
C
DO 1000 III=1,NCASE2
GO TO (11,12,13,14,15), NOPTN2
11 GO TO 60
12 READ(5,705) ENODB
ENODBR=ENODB
GO TO 60
13 READ(5,705) SJRDBR
GO TO 60
14 READ(5,705) QK
GO TO 60
15 READ(5,706) SJRDB, SJRDBR
60 CONTINUE
C
WRITE(6,650) NSPB, TB, ENODB, SJRDB, SJRDBR, QK, AEST
650 FORMAT(2X,5NSPB=,I5,2X,3HTB=,E13.6,2X,6HENODB=,E13.6,2X,
16HSJRDB=,E13.6,2X,7HSJRDBR=,E13.6,2X,3HOK=,E13.6,2X,5HAEST=,E13.6)
CALL INPUT1(IPARAM, IGV)
CALL INPUT2(IPARAM, QK)
C
CALL CFERAT(ERATCL)
C
WRITE(6,651) ERATCL
651 FORMAT(2X,26HCLOSED-FORM ERROR RATE IS ,30X,10H**********=,E13.6)
1000 CONTINUE
WRITE(6,751)
751 FORMAT(5X,1IHEND OF CASE,)//
2000 CONTINUE
STOP
END
C
SUBROUTINE DATA
COMMON/ORDER/N, N2
COMMON/SAMPLE/NSPB, TB, TBR
COMMON/OPTION/NOS, AEST
COMMON/RATIO/ENODB, ENODBR, SJRDB, SJRDBR
COMMON/WORNOW/IMODE, KSMAX, IOJ
COMMON/FREQ/FZ, FP(3)
C
N, N2 : SYSTEM ORDER
NOS : (1) PSK, (2) FSK
AEST : SIGNAL MAGNITUDE IN SUB. REFGEN
IMODE: (1) DIAGONAL PHEE MATRIX AND PERFECT IDENTIFICATION
(2) DIAGONAL PHEE MATRIX
(3) GENERAL IMPERFECT IDENTIFICATION
KSMAX: MAXIMUM NUMBER OF ITERATION FOR STEADY-STATE KALMAN GAIN
IGV : (0) NO CALCULATION FOR CORRECT KALMAN GAIN AND VINOV IN INPUT1
(1) CALCULATION FOR CORRECT KALMAN GAIN AND VINOV IN INPUT1
FZ, FP: ZERO, POLE FREQUENCY FOR LOW-PASS FILTER
READ(5,600) N, N2
READ(5,601) NSPB, TB, TBR
READ(5,602) NOS, AEST
READ(5,603) ENODB, ENODBR, SJRDB, SJRDBR
READ(5,604) IMODE, KSMAX, IOJ
READ(5,603) FZ, (FP(I), I=1,3)
600 FORMAT(2I5)
601 FORMAT(I5,2E15.6)
602 FORMAT(I5,E15.6)
SUBROUTINE INPUT1(IPARAM, IQV)
C TO GET THE REAL PARAMETERS AND STATISTICS GIVEN VALUES.
C ALL WE NEED IN HERE ARE GAMMA, PHEE, H, R
C GAIN AND VINOV ARE FOR REFERENCE
C IF IQV : 0 - NO CALCULATION FOR CORRECT KALMAN GAIN AND VINOV
C : 1 - CALCULATION FOR CORRECT KALMAN GAIN AND VINOV
C THEREFORE GK ALWAYS SET 1..FOR PERFECT IDENTIFICATION.
COMMON/ORDER/N, N2
COMMON/SAMPLE/NSPB, TB, TBR
COMMON/OPTION/NOS, AEST
COMMON/RATIO/ENODB, ENODBR, SJRDB, SJRDBR
COMMON/QDB/QN, GN, GJ, GJR
COMMON/WORNOW/IMODE, KSMAX, IDJ
COMMON/FREQ/FZ, FP(3)
COMMON/PARAM/GAMMA(6, 2), PHEE(6, 6), H(2, 6), Q(2, 2), R(2, 2)
DIMENSION VINOV(2, 2), GAIN(6, 2)
CALL PARALL(1., BN, GAMMA, PHEE, H, ENODB, SJRDB, GN, GJ, R, IQV
1, GAIN, VINOV)
IF(IPARAM.EQ.0) GO TO 40
WRITE(6, 610)
610 FORMAT(2X, 32HREAL PARAMETERS AND STATISTICS*, /)
DO 20 I=1, N
WRITE(6, 611) I, GAMMA(I, 1), I, PHEE(I, I), I, H(1, 1)
611 FORMAT(2X, 5HGAMMD(, I1, 2H)=, E13. 6, 2X, 5HPHD(, I1, 2H)=, E13. 6,
12X, 3HHT(, I1, 2H)= E13. 6)
20 CONTINUE
IF(IQV.EQ.0) GO TO 40
WRITE(6, 615) GN
615 FORMAT(/, 2X, 3HGN=, E13. 6)
WRITE(6, 612)
612 FORMAT(/, 2X, 5HGAIN=, 26X, 6HVINOV=)
DO 25 I=1, N2
25 CONTINUE
IF(IQV.EQ.0) GO TO 40
WRITE(6, 614) (GAIN(IJ), I=1, N2)
WRITE(6, 613) (VINOV(IJ), I=1, N2)
613 FORMAT(2X, 2E13. 6, 5X, 2E13. 6)
614 FORMAT(2X, 2E13. 6)
25 CONTINUE
40 CONTINUE
WRITE(6, 617) BN
617 FORMAT(/, 2X, 21HEQUIVALENT BANDWIDTH=, E13. 6, /)
RETURN
END
SUBROUTINE INPUT2(IPARAM, IQV)
C THIS SUBROUTINE QSTAR AND DPHEE FOR DIFFERENT FILTER BANDWIDTH
C THESE QSTAR AND DPHEE WITH GAMMA, PHEE, H, R ARE USED TO CALCULATE
C RESIDUAL VARIANCE IN SUBROUTINE CFERAT AND VTT.
C THEREFORE IQV ALWAYS SET 1 HERE.
COMMON/ORDER/N, N2
COMMON/RATIO/ENODB, ENODBR, SJRDB, SJRDBR
COMMON/QDB/QN, GN, GJ, GJR
COMMON/WORNOW/IMODE, KSMAX, IDJ
COMMON/PARAM/GAMMA(6, 2), PHEE(6, 6), H(2, 6), Q(2, 2), R(2, 2)
DIMENSION GAMMAR(6, 2), RR(2, 2), VINOVR(2, 2)
CALL PARALL(QK, BN, QASTER, PHEE, H, ENODBR, SJRDBR, GN, GJ, RR
1, 1, QSTAR, VINOV)
DO 10 I=1, N2
10 CONTINUE
DO 10 J=1, N2
10 DPHEE(I,J)=PHEER(I,J)-PHEE(I,J)
   IF(IPARAM.EQ.0) RETURN
   WRITE(6,600)
600 FORMAT(/,2X,35HESTIMATED PARAMETERS AND STATISTICS,/)  
   DO 20 I=1,N
      WRITE(6,601) I, GAMMAR(I,1), I, PHEER(I,I), I, H(I,I), I, DPHEE(I,I)
   1, I, QSTAR(I,1)
601 FORMAT(2X,'GAMDR(',I1,')=',E13.6,2X,'PHIDR(',I1,')=',E13.6
      1,2X,'HTR(',I1,')=',E13.6,5X,'DPHEE(',I1,')=',E13.6,2X,'QSTAR('
      2, I1,')=',E13.6)
20 CONTINUE
   WRITE(6,602) BNR
602 FORMAT(/,2X,21HEQUIVALENT BANDWIDTH=,E13.6)
   WRITE(6,603) VINOV(1,1)
603 FORMAT(/,2X,12HVINOVR(1,1)=,E13.6)
   RETURN
   END
C
SUBROUTINE PARALL(GK, DNGAMMA, PHEE, H, ENODB, SJRDB, QN, QJ, R, IGV
1, GAIN, VINOVR)
COMMON/ORDER/N, N2
COMMON/OPTION/NOS, AEST
COMMON/SAMPLE/NSPB, TB, TBR
COMMON/WORK/IMODE, KSMAX, IOJ
COMMON/FREG/FZ, FP(3)
DIMENSION FPR(3)
DIMENSION QAMD(3), PHID(3), HT(3)
DIMENSION GAMMA(6,2), PHEE(6,6), H(2,6), Q(2,2), R(2,2)
DIMENSION PVPT(6,6), GTC6,6), VEST(6,6), VPRED(6,6), HVHT(2,2)
1, VINOV(2,2), VINV(2,2), VPHT(6,2), QAIN(6,2), QH(6,6)
REAL IMGHC6,6)
PI=4.*ATAN(1.)
DELPHI=.785
DELMEG=DELPHI*2.*PI/TB
IF(NOS.EQ.1) GO TO 1
SUMF=0.0
   DO 2 K=1,NSPB
2 SUMF=SUMF+(SIN((K-.5)*TB*DELMEG/NSPB))**2
   CONSTF=SQRT(SUMF)
   QN=CONSTF*10.**(-ENODB/20.)
   GO TO 3
1 CONSTP=SQRT(NSPB/2.)*ABS(SIN(DELPHI))
   QN=CONSTP*10.**(-ENODB/20.)
3 QJ=10.**(-SJRDB/20.)/SQRT(2.)
   R(1,1)=QN**2
   R(1,2)=0.0
   R(2,1)=0.0
   R(2,2)=QN**2
   FZR=QK*FZ
   DO 5 I=1,N
5 FPR(I)=QK*FP(I)
   T=TB/NSPB
   CALL PREPAR(T,FZR,FPR, QAMD, PHID, HT, BN, IDJ)
   DO 10 I=1,N2  
   DO 10 J=1,2
10 GAMMA(I,J)=0.0
   DO 11 I=1,N
11 GAMMA(I,1)=QAMD(I)
11 GAMMA(I+N,2)=QAMD(I)
C NEW WEIGHTED GAMMA MATRIX
   DO 12 I=1,N2
12 GAMMA(I,J)=QJ*GAMMA(I,J)
   DO 15 I=1,N2
15 PHEE(I,J)=0.0
44
DO 16 I=1,N
   PHEE(I, I)=PHID(I)
16 PHEE(I+N, I+N)=PHID(I)
DO 20 I=1,2
DO 20 J=1,N2
20 H(I, J)=0.0
DO 21 I=1, N
   H(I, I)=HT(I)
21 H(I, I+N)=HT(I)
C
IF (IGV.EQ.0) RETURN
C
CALCULATE THE STEADY-STATE KALMAN-GAIN
DO 32 I=1, N2
   H(IJ)=0.0
32 CONTINUE
DO 35 KS=1, KSMAX
   CALL MABCT(PHEE, N2, N2, VEST, N2, PHEE, N2, PVPT, 6, 6, 6, 6, 6, 6)
   CALL MATMUL(2, GAMMA, N2, 2, GAMMA, N2, GTG, 6, 2, 6, 2, 6, 6)
   CALL MATAS(1, PVPT, N2, N2, GTG, VPREDD, 6, 6)
   CALL MABCT(H, 2, N2, VPRED, N2, H, 2, HVHT, 2, 6, 6, 6, 6, 2, 2)
   CALL MATAS(1, R, 2, HVHT, VINV, 2, 2)
   CALL MATMUL(2, VPRED, N2, N2, H, 2, VPHT, 6, 6, 6, 6, 2, 6, 2)
   DET=VINV(1, 1)*VINV(2, 2)-VINV(1, 2)*VINV(2, 1)
   VINV(1, 1)=VINV(2, 2)/DET
   VINV(1, 2)=-VINV(1, 2)/DET
   VINV(2, 1)=-VINV(2, 1)/DET
   VINV(2, 2)=VINV(1, 1)/DET
   CALL MATMUL(1, VPHT, N2, 2, VINV, 2, GAIN, 6, 2, 2, 2, 6, 2)
   CALL MATMUL(1, GAIN, N2, 2, H, N2, GH, 6, 2, 2, 6, 6)
   DO 36 I=1, N2
      GH(I, J)=-GH(I, J)
   36 CONTINUE
   CALL MATMUL(1, IMGH, N2, N2, VPRED, N2, VEST, 6, 6, 6, 6, 6)
35 CONTINUE
RETURN
END

SUBROUTINE PREPAR(T, FZ, FP, GAMD, PHID, HT, BN, INOPT)
C
** I/O PARAMETERS *
C
* INPUT *
C
T: SAMPLING TIME
C
FZ: ZERO FREQUENCY
C
FP: POLE FREQUENCIES (3)
C
INOPT: 1-DIGIT CODE FOR SELECTION OF REAL/COMPLEX ZERO
   AND UNITY GAIN/VARIANCE FOR FILTER PARAMETER CALCULATIONS
   =1, REAL ZERO, UNIT GAIN
   =2, REAL ZERO, UNIT VARIANCE
   =3, COMPLEX ZERO, UNIT GAIN
   =4, COMPLEX ZERO, UNIT VARIANCE
C
* OUTPUT *
C
PHID: FILTER TRANSITION WEIGHTS(3)
C
GAMD: FILTER INPUT WEIGHTS(3)
C
HT: FILTER OUTPUT WEIGHTS(3)
C
BN: EQUIVALENCE NOISE BANDWIDTH
C
* INTERNAL FILTER PARAMETERS *
C
Z: ZERO FREQUENCY, IN RADIANS
C
P: POLE FREQUENCIES (3), IN RADIANS
C
R: RESIDUES(3)
C
RE: RESIDUES(3)
C
GAINK: GAIN CONSTANT
C
** SUBROUTINE ADDED TO SUBROUTINE INPUT **
C
TO PERFORM PRE-CALCULATIONS OF FILTER PARAMETERS
C
** **
PI=4.*ATAN(1.)
IF(INOPT.GT.2) GO TO 100
C FREQUENCY CALCULATIONS
Z=(-2.)*PI*FZ
DO 1 I=1,3
1 P(I)=(-2.)*PI*FP(I)
C RESIDUE CALCULATIONS
DO 5 I=1,3
D=1.
DO 10 J=1,3
IF(I.EQ.J) GO TO 10
D=D*(P(I)-P(J))
10 CONTINUE
R(I)=(P(I)-Z)/D
5 CONTINUE
C TRANSITION WEIGHTS
DO 20 I=1,3
20 PHID(I)=EXP(P(I)*T)
C INPUT WEIGHTS
DO 25 I=1,3
25 GAMD(I)=(1.-PHID(I))*R(I)/(-P(I))
C UNITY GAIN
IF(INOPT.NE.1) GO TO 30
GAINK=P(I)*P(2)*P(3)/Z
GO TO 35
30 CONTINUE
C UNITY VARIANCE
SUM=0.0
DO 40 I=1,3
DO 40 J=1,3
40 SUM=SUM+GAMD(I)*GAMD(J)/(1.0-PHID(I)*PHID(J))
GAINK=1./SQRT(SUM)
35 CONTINUE
C NEW WEIGHTED INPUT MATRIX
DO 36 I=1,3
36 QAMD(I)=GAINK*GAMD(I)
C OUTPUT WEIGHTS
DO 45 I=1,3
45 HT(I)=1.
C EQUIVALENT NOISE BANDWIDTH
DO 50 I=1,3
DO 50 J=1,3
IF(I.EQ.J) GO TO 55
D=D*(P(I)**2-P(J)**2)
55 CONTINUE
RE(I)=GAINK*2*(P(I)**2-Z**2)/(2.*P(I)*D)
50 CONTINUE
GO=GAINK**2/(P(1)**2*P(2)**2+P(3)**2)
BN=(RE(1)+RE(2)+RE(3))/(2.*GO**2)
RETURN
100 CONTINUE
C MODIFIED TRANSFER FUNCTION HAVING COMPLEX ZERO.
C FREQUENCY CALCULATION.
C Z**2=P(2)**2-2*P(1)**2, TO HAVE A JW-AXIS ZERO, Z SHOULD BE POSITIVE
Z=-2.*PI*FZ
P(1)=I
P(2)=SQRT(3.)*P(1)
P(3)=-2.*PI*FP(3)
FP(1)=P(1)/(-2.*PI)
FP(2)=P(2)/(-2.*PI)
FP(3)=P(3)/(-2.*PI)
C RESIDUE CALCULATIONS
DO 110 I=1,3
D=1.
DO 120 J=1,3
110 CONTINUE
IF(I.EQ.J) GO TO 120
   D=D*(P(I)-P(J))
120 CONTINUE
   R(I)=(P(I)**2+Z**2)/D
110 CONTINUE
C    TRANSITION WEIGHTS
   DO 125 I=1,3
125 PHID(I)=EXP(P(I)*T)
C    INPUT WEIGHTS
   DO 130 I=1,3
130 GAMD(I)=(1.-PHID(I))*R(I)/(1.0-PHID(I)*PHID(J))
C    UNITY GAIN
   IF(INOPT. NE. 3) GO TO 135
   GAINK=2.*PI*FP(1)*FP(2)*FP(3)/FZ**2
   GO TO 140
135 CONTINUE
C    UNITY VARIANCE
   SUM=0.0
   DO 150 I=1,3
   DO 150 J=1,3
150 SUM=SUM+GAMD(I)*GAMD(J)/(1.0-PHID(I)*PHID(J))
   GAINK=1./SGRT(SUM)
140 CONTINUE
C    NEW WEIGHTED INPUT MATRIX
   DO 141 I=1,3
141 GAMD(I)=GAINK*GAMD(I)
C    OUTPUT WEIGHT
   DO 155 I=1,3
155 HT(I)=1.
C    EQUIVALENT NOISE BANDWIDTH
   DO 160 I=1,3
   D=1.
   DO 170 J=1,3
   IF(I.EQ.J) GO TO 170
   D=D*(P(I)**2-P(J)**2)
170 CONTINUE
   RE(I)=(-1.)*GAINK**2*(P(I)**2+Z**2)**2/(2.*P(I)*D)
160 CONTINUE
   GO=(-1.)*GAINK*Z**2/(P(1)*P(2)*P(3))
   IN=RE(1)+RE(2)+RE(3))/(2.*GO**2)
   RETURN
END
SUBROUTINE CFERAT(CERATCL)
EXTERNAL ERF
COMMON/ORDER/N, N2
COMMON/SAMPLE/NSPB, TB, TBR
COMMON/WORKNOW/IMODE, KSMAX, IODJ
COMMON/PARAM/GAMMA(6, 2), PHEE(6, 6), H(2, 6), Q(2, 2), R(2, 2)
COMMON/PARAMR/PHEER(6, 6), DPHEE(6, 6), QSTAR(6, 2), BSTAR(2, 2)
DIMENSION XEST(6), XEST2(6), XPRED(6), XPRED2(6)
DIMENSION SIGI(2), SIG2(2), ES1(2), ES2(2)
DIMENSION BI(2, 300), YTTJ(2, 2), VTILDA(300)
DIMENSION VXX(6, 6), TEMP(6, 6), QTO(6, 6), F(6, 6), VXXT(6, 6), VXXT1(6, 6)
1, VXXT2(6, 6), VXXT3(6, 6), VXXT4(6, 6), VXXT5(6, 6), QRG(6, 6), TEMP(6, 6)
REAL IMGH(6, 6)
C
CALL MATMUL(2, GAMMA, N2, 2, GAMMA, N2, GTG, 6, 2, 6, 2, 6, 6)
CALL MATMUL(1, QSTAR, N2, 2, H, N2, TEMP, 6, 2, 6, 6, 6, 6)
DO 5 I=1, N2
DO 5 J=1, N2
IMGH(I, J)=TEMP(I, J)
5 CONTINUE
CALL MATMUL(1, PHEER, N2, N2, IMGH, N2, F, 6, 6, 6, 6, 6, 6)
C
INITIALIZE VXX, VXXT AND VXXXT
DO 6 I=1, N2
DO 6 J=1, N2
VXX(I, J)=0.0
VXXT(I, J)=0.0
6 VXXT(I, J)=0.0
C
DO 1 KS=1, KSMAX
VXX(K)=PHEE*VXX(K-1)*PHEE' + GAMMA*Q*GAMMA'
CALL MABCT(PHEE, N2, N2, VXX, N2, PHEE, N2, TEMP, 6, 6, 6, 6, 6, 6)
CALL MATAS(1, TEMP, N2, N2, QTO, VXX, 6, 6)
VXXT(K-1)=PHEE*VXXT(K-1-1)*F' + PHEE*VXX(K)*DPHEE'
+ GAMMA*Q*GAMMA'
CALL MABCT(PHEE, N2, N2, VXXT, N2, F, N2, VXXT1, 6, 6, 6, 6, 6, 6)
CALL MABCT(PHEE, N2, N2, VXXT, N2, N2, DPHEE, N2, VXXT2, 6, 6, 6, 6, 6, 6)
CALL MATAS(1, VXXT1, N2, N2, VXXT2, TEMP, 6, 6)
CALL MATAS(1, TEMP, N2, N2, QTO, VXXT, 6, 6)
VXXTK(K+1-K)=F*VXXT(K+1-K)*F' + 2.*DPHEE*VXXT(K+1-K)*F'
+ DPHEE*VXX(K)*DPHEE' + GAMMA*Q*GAMMA'
+ PHEER*QSTAR*R*QSTAR' + PHEER'
CALL MABCT(F, N2, N2, VXXT, N2, F, N2, VXXT1, 6, 6, 6, 6, 6, 6)
CALL MABCT(DPHEE, N2, N2, VXXT, N2, F, N2, VXXT2, 6, 6, 6, 6, 6, 6)
DO 15 I=1, N2
15 VXXT3(I, J)=VXXT2(J, I)
CALL MABCT(DPHEE, N2, N2, VXXT, N2, DPHEE, N2, VXXT4, 6, 6, 6, 6, 6, 6)
CALL MABCT(QSTAR, N2, 2, R, 2, QSTAR, N2, QRG, 6, 2, 2, 6, 2, 6, 6)
CALL MABCT(PHEER, N2, N2, QRG, N2, PHEER, N2, VXXT5, 6, 6, 6, 6, 6, 6)
CALL MATAS(1, VXXT1, N2, N2, VXXT2, TEMP, 6, 6)
CALL MATAS(1, TEMP, N2, N2, VXXT3, TEMP, 6, 6)
CALL MATAS(1, TEMP, N2, N2, VXXT4, TEMP, 6, 6)
CALL MATAS(1, TEMP, N2, N2, GTG, TEMP, 6, 6)
CALL MATAS(1, TEMP, N2, N2, VXXT5, VXXT, 6, 6)
1 CONTINUE
C
DO 25 I=1, N2
XEST1(I)=0.0
XEST2(I)=0.0
25 CONTINUE
C SIGI : ES(M=0, N=0)
DO 20 K = 1, NSPB
CALL REFGEN(K, 0, FRRO, GRR0, ORR0)
CALL REFGEN(K, 1, FR1, GTR1, FRR1, ORR1)
SIG1(1) = FRRO - FR1
SIG1(2) = GRR0 - GTR1
SIG2(1) = FRRO - FRR1
SIG2(2) = GRR0 - ORR1
CALL WKFLT(K, XEST1, XPRE1, SIG1, ES1)
CALL WKFLT(K, XEST2, XPRE2, SIG2, ES2)
A = A + (ES1(1)**2 + ES2(2)**2 - ES1(1)**2 - ES2(1)**2)
B1(1, K) = ES1(1) - ES2(1)
B1(2, K) = ES1(2) - ES2(2)
20 CONTINUE

C THE VII, INNOVATION VARIANCE, IS DECOUPLED, SO IS VTTJ SINCE
C THE TEST SYSTEM IS DECOUPLED.
C IF THE SYSTEM IS COUPLED, THEN THE EVALUATION OF MEAN AND VARIANCE
C MUST BE MODIFIED.
DO 30 J = 1, NSPB
L = J - 1
CALL VTT(L, F, VXTXT, VXXT, VTTJ)
VTILDA(J) = VTTJ(1, 1)
30 CONTINUE
B = 0.0
DO 35 J = 1, NSPB
DO 35 K = 1, NSPB
L = IABS(J - K) + 1
B = B + (B1(1, J) * B1(1, K) + B1(2, J) * B1(2, K)) * VTILDA(L)
35 CONTINUE
IF (B .LE. 0.0) GO TO 50
SIGMAB = SQRT(B)
X = A / SIGMAB
SUFX = X / (2.0 * SQRT(2.))
ERATCL = 0.5 * (1.0 - ERF(SUFX))
WRITE(6, 600) A, SIGMAB, VTILDA(1), X

600 FORMAT(2X, 'MU=', E13.6, 2X, 'SIGMA=', E13.6, 2X, 'VTT(0)=', E13.6, 12X, 'MU/SIGMA=', E13.6)
RETURN

50 WRITE(6, 601)

601 FORMAT(2X, 'HVARINANCE IS NEGATIVE')
DO 60 I = 1, NSPB
60 WRITE(6, 602) I, VTILDA(I), B1(1, I), B1(2, I)

602 FORMAT(2X, 'HTVILDA(i, 3.2H)=', E13.6, 2X, '15HTRACTING ERROR=', E13.6)
RETURN

END

SUBROUTINE WKFLT(KS, XEST, XPRED, SIG, V)
COMMON/ORDER/N, N2
COMMON/PARAM/GAMMA(6, 2), PHEE(6, 6), H(2, 6), Q(2, 2), R(2, 2)
COMMON/PARAM/PHEER(6, 6), DPHEE(6, 6), QSTAR(6, 2), BSTAR(2, 2)
DIMENSION XEST(6), XPRED(6), SIG(2), V(2), ZHAT(2), GV(6)
CALL MATVEC(PHEER, N2, N2, XEST, XPRED, 6, 6)
CALL MATVEC(H, 2, N2, XPRED, ZHAT, 2, 6)
CALL VECAS(2, SIG, ZHAT, V, 2)
CALL MATVEC(QSTAR, N2, 2, V, GV, 6, 2)
CALL VECAS(1, XPRED, GV, XEST, 6)
RETURN

END

SUBROUTINE REFGEN(KS, M, FR, GTR, FRR, GRR)
COMMON/SAMPLE/NSPB, TB, TBR
COMMON/OPTION/NOS, AEST
TK = (KS - 0.5) / NSPB
TKRMD = (TK - IFIX(TK)) * TBR
DELPHI = 785
DELMEG = DELPHI * 6.0 * ATAN(1.0) / TB
IF (NOS. NE. 1) GO TO 1
IF(M. EQ. 0) PHEE1 = DELPHI
IF(M. EQ. 1) PHEE1R = DELPHI
GO TO 2
1 IF(M. EQ. 0) PHEE1R = DELMEG*TKRMD
IF(M. EQ. 1) PHEE1R = DELMEG*TKRMD
2 FTR = COS(PHEE1R)
QTR = SIN(PHEE1R)
FRR = AEST*COS(PHEE1R)
QRR = AEST*SIN(PHEE1R)
RETURN
END

FUNCTION ERF(X)
C THIS IS AN APPROXIMATION OF ERROR FUNCTION HAVING
C LESS THAN 1.5E-7 ERROR AND ASSUMED X IS POSITIVE
C ERROR FUNCTION IS SYMMETRIC
P=0.3275911
A1=0.254829592
A2=0.284496736
A3=1.421413741
A4=-1.453152027
A5=1.061405429
XX=ABS(X)
T=1./(1.+XX)
ERF= 1.- (A1*T+A2*T**2+A3*T**3+A4*T**4+A5*T**5)*EXP(-XX**2)
IF(X.GE.0.) ERF=ERF
IF(X.LT.0.) ERF=-ERF
RETURN
END
SUBROUTINE VTT(JP, F, VXTXT, VXXT, VTTJ)
COMMON/ORDER/N, N2
COMMON/WORNOW/IMODE, KS MAX, IDJ
COMMON/PARAM/GAMMA (6, 2), PHEE (6, 6), H(2, 6), Q(2, 2), R(2, 2)
COMMON/PARAM/PHEE(6, 6), DPHEE(6, 6), GSTAR(6, 2), BSTAR(2, 2)
DIMENSION F(6, 6), VXTXT(6, 6), VXXT(6, 6), VTTJ(2, 2)
DIMENSION VHT(6, 2), HVHT(2, 2), B1(6, 2), B2(6, 2), B3(6, 2)
B4(6, 2), B5(2, 2), TEMP(6, 6), FL(6, 6)
DIMENSION PHEEJ(6, 6), V(6, 6), V1(6, 6), V2(6, 6), V3(6, 6), V4(2, 2)
IF(JP) 1, 2, 3
1 WRITE(6, 11)
 11 FORMAT(2X, 28HNEGATIVE J POWER IN VTT SUB.)
  RETURN
2 CALL MABCT(H, 2, N2, VXTXT, N2, H, 2, HVHT, 2, 6, 6, 6, 6, 2, 2)
  CALL MATAS(1, HVHT, 2, 2, R, BSTAR, 2, 2)
  DO 4 1 = 1, 2
  DO 4 J = 1, 2
4 VTTJ(I, J) = BSTAR(I, J)
  RETURN
3 CONTINUE
C [ V * H' - GSTAR * ( H * V * H' + R ) ]
  CALL MATMUL(2, VXTXT, N2, N2, H, 2, VHT, 6, 6, 6, 6, 6, 2, 2)
  CALL MATAS(1, GSTAR, N2, 2, BSTAR, 2, B1, 6, 2, 2, 6, 2)
  CALL MATAS(2, VHT, N2, 2, B1, 6, 2, 2)
C PHEE [ V * H' - GSTAR * ( H * V * H' + R ) ]
  CALL MATMUL(1, VHEE, N2, N2, B2, 6, 6, 6, 6, 6, 2, 2)
C F = [ PHEE * ( I - GSTAR * H ) ]**(J-1)
  CALL CAYLEY(IMODE, F, JP-1, FL)
C H * F**(J-1) * PHEE * [ V * H' - B ]
  CALL MATMUL(1, FL, N2, N2, B3, 6, 6, 6, 6, 6, 2, 2)
  CALL MATMUL(1, H, 2, N2, B4, 2, B5, 6, 6, 6, 6, 2, 2)
  IF(IMODE.EQ.1) GO TO 100
C
C SUM( F**(I-1) * DPHEE * PHEE**(J-I) )
  DO 5 II = 1, N2
  DO 5 JJ = 1, N2
5 TEMP(II, JJ) = 0.0
  DO 10 I = 1, JP
    L1 = I - 1
    L2 = JP - I
    CALL CAYLEY(IMODE, F, L1, FL)
    CALL CAYLEY(IMODE, PHEE, L2, PHEEJ)
    CALL MATMUL(1, FL, N2, N2, DPHEE, N2, V1, 6, 6, 6, 6, 6)
    CALL MATMUL(1, V1, N2, PHEEJ, N2, V2, 6, 6, 6, 6, 6)
    CALL MATAS(1, TEMP, N2, N2, V2, V, V, V, V)
  DD 10 II = 1, N2
  DD 10 JJ = 1, N2
  TEMP(II, JJ) = TEMP(II, JJ) + V(II, JJ)
  15 CONTINUE
C SUM( F**(I-1) * DPHEE * PHEE**(J-I) ) * VXXT
  CALL MATMUL(1, V, N2, N2, VXXT, N2, V3, 6, 6, 6, 6, 6)
C H * SUM( F**(I-1) * DPHEE * PHEE**(J-I) ) * VXXT * H'
  CALL MABCT(H, 2, N2, V3, N2, H, 2, V4, 6, 6, 6, 6, 6, 2, 2)
  CALL MATAS(1, B5, 2, V4, VTTJ, 2, 2)
  RETURN
100 DD 110 I = 1, 2
110 VTTJ(I, J) = B5(I, J)
110 CONTINUE
RETURN
END
SUBROUTINE CAYLEY(IMODE, F, L, FL)
C THIS SUBROUTINE PRODUCE THE MATRIX HIGH POWERED USING
CAYLEY-HAMILTON'S THEOREM TO REDUCE THE ERROR.

IMODE

(1) AND (2): F IS DIAGONAL MATRIX SO THAT IT HAS SAME EIGEN-VALUE.

THIS REQUIRE SPECIAL GAUSS SUBROUTINE TO SOLVE THE LINEAR EQUATIONS.

(3): F IS GENERAL MATRIX AND HAS THE DISTINGUISHED EIGEN-VALUE.

F: INPUT MATRIX TO BE MULTIPLIED BY HIGH POWER

RESULTANT MATRIX

COMMON/ORDER/N,N2

COMPLEX EV(6),AI(6,6),BI(6),ALFA(6),FL1(6,6),CMPLX

DIMENSION F(6,6),A(12,12),B(12),X(12),FL(6,6),FP(6,6,6)

DIMENSION SF(6,6)

IF(L) 1,2,3

WRITE(6,4)

4 FORMAT('NEGATIVE L IN SUB. CAYLEY')

RETURN

2 DO 5 I=1,N2

DO 5 J=1,N2

FL(I,J)=0.0

IF(I.EQ.J) FL(I,J)=1.

5 CONTINUE

RETURN

3 CONTINUE

IF(L.NE.1) GO TO 7

DO 6 I=1,N2

DO 6 J=1,N2

FL(I,J)=F(I,J)

6 CONTINUE

RETURN

7 CONTINUE

IF(IMODE.NE.3) GO TO 150

N4=N*4

CALL EIGEN(F,N2,EV)

C USING GAUSS ELIMINATION METHOD, COMPLEX MATRIX CONSISTED WITH EIGENVALUES IS PARTITIONED.

DO 20 I=1,N2

DO 10 J=1,N2

10 A1(I,J)=EV(I)**(J-1)

20 B1(I)=EV(I)**L

DO 40 I=1,N2

DO 30 J=1,N2

A(I,J)=REAL(A1(I,J))

A(I,J+N2)=-AIMAG(A1(I,J))

A(I+N2,J)=-AIMAG(A1(I,J))

A(I+N2,J+N2)=REAL(A1(I,J))

30 CONTINUE

B(I)=REAL(B1(I))

B(I+N2)=AIMAG(B1(I))

40 CONTINUE

CALL GAUS(A,B,X,N4,IERROR)

C GENERATE THE COEFFICIENTS OF CHARACTERISTIC FUNCTION

DO 50 I=1,N2

ALFA(I)=CMPLX(X(I),X(I+N2))

50 CONTINUE

C CAYLEY-HAMILTON'S THEOREM

DO 70 I=1,N2

DO 70 J=1,N2

FP(I,J,1)=0.0

IF(I.EQ.J) FP(I,J,1)=1.

70 CONTINUE

NM2=N-2

IF(NM1) 90,90,95
DO 80 NP=3,N
DO 85 I=1,N2
DO 85 J=1,N2
FP(I,J,NP)=0.0
DO 85 M=1,N2
FP(I,J,NP)=FP(I,J,NP)\*FP(I,M,NP-1)*F(M,J)
85 CONTINUE
80 CONTINUE
90 CONTINUE
DO 100 I=1,N2
DO 100 J=1,N2
FL1(I,J)=CMPLX(0,0,0,0)
100 CONTINUE
90 CONTINUE
DO 100 I=1,N2
DO 100 J=1,N2
FL1(I,J)=FL1(I,J)+ALFA(NP)\*FP(I,J,NP)
100 CONTINUE
110 CONTINUE
DO 130 I=1,N2
DO 130 J=1,N2
FL(I,J)=REAL(FL1(I,J))
130 CONTINUE
RETURN
150 CONTINUE
DO 152 I=1,N
DO 152 J=1,N
SF(I,J)=F(I,J)
CALL EIGEN(SF,N,EV)
DO 220 I=1,N
DO 220 J=1,N
AI(I,J)=EV(I)**(J-1)
220 CONTINUE
210 AI(I,J)=EV(I)**(J-1)
220 CONTINUE
B(I)=EV(I)**(J-1)
DO 240 I=1,N
DO 240 J=1,N
A(I,J)=REAL(A1(I,J))
A(I,J+N)=-AIMAG(A1(I,J))
A(I+N,J)=AIMAG(A1(I,J))
A(I+N,J+N)=REAL(A1(I,J))
230 CONTINUE
240 CONTINUE
CALL GAUS(A,B,X,N2,IERROR)
DO 250 I=1,N
ALFA(I)=CMPLX(X(I),X(I+N))
250 CONTINUE
DO 270 I=1,N
DO 270 J=1,N
FP(I,J,1)=0.0
IF(I.EQ.J) FP(I,J,1)=1.
270 CONTINUE
DO 275 I=1,N
DO 275 J=1,N
FP(I,J,2)=SF(I,J)
275 CONTINUE
NM2=N-2
IF(NM2) 290,290,295
295 DO 280 NP=3,N
DO 285 I=1,N
DO 285 J=1,N
FP(I,J,NP)=0.0
DO 285 M=1,N
FP(I,J,NP)=FP(I,J,NP)\*FP(I,M,NP-1)*SF(M,J)
285 CONTINUE
280 CONTINUE
SUBROUTINE GAUS(A, B, X, N, IERROR)
DIMENSION A(12, 12), B(12), X(12)

C THIS SUBROUTINE IS IN 'NUMERICAL ANALYSIS' BY L.W. JOHNSON AND R.D.
C RIESS, 1977 BY ADDISON-WESLEY PUB.CO.
C SUBROUTINE GAUS USES GAUSS ELIMINATION (WITHOUT PIVOTING) TO SOLVE
C VECTOR B AND AN INTEGER N (WHERE A IS (NXN). ARRAYS A AND B ARE
C DESTROYED IN GAUS. THE SOLUTION IS RETURNED IN X AND A FLAG, IERROR,
C IS SET TO 1 IF A IS NON-SINGULAR AND IS SET TO 2 IF A IS SINGULAR.
C TO GET MORE ACCURATE SOLUTION, CALL SUBROUTINE RESCOR AFTER GAUS.

C NM1=N-1
DO 5 I=1, NM1
C SEARCH FOR NON-ZERO PIVOT ELEMENT AND INTERCHANGE ROWS IF NECESSARY.
C IF NO NON-ZERO PIVOT ELEMENT IS FOUND, SET IERROR=2 AND RETURN

DO 3 J=I, N
IF(A(J, I).EQ.0.) GO TO 3
DO 2 K=I, N
TEMP=A(I, K)
A(I, K)=A(J, K)
2 A(J, K)=TEMP
TEMP=B(I)
B(I)=B(J)
B(J)=TEMP
GO TO 4
3 CONTINUE
GO TO 8

C ELIMINATE THE COEFFICIENTS OF X(I) IN ROWS I+1,...,N

4 IP1=I+1
DO 5 K=IP1, N
G=-A(K, I)/A(I, I)
A(K, I)=0.0
B(K)=G*B(I)+B(K)
5 DO 8 J=IP1, N
6 A(K, J)=G*A(I, J)+A(K, J)
IF(A(N, N).EQ.0.) GO TO 8

C BACKSOLVE THR EQUIVALENT TRIANGULARIZED SYSTEM, SET IERROR=1.
C AND RETURN

X(N)=B(N)/A(N, N)
RETURN
END
SUBROUTINE EIGEN(A, N, EVALUE)
DIMENSION A(6,6), RR(6), RI(6), IANA(36), AT(36)
COMPLEX CMPLX, EVALUE(6)
DO 6 I=1,N
   DO 7 J=I,N
      K=N*(I-1)
      AT(J+K)=A(I,J)
   CONTINUE
CALL HSBG(N, AT, N)
CALL ATEIG(N, AT, RR, RI, IANA, N)
DO 5 I=I,N
   EVALUE(I) = CMPLX(RR(I), RI(I))
C WRITE(6,500)
C 500 FORMAT(5X, 'THE EIGENVALUE IS')
C WRITE(6,600)
C 600 FORMAT(10X, 'REAL ROOT', 15X, 'IMAG ROOT')
C WRITE(6,700) RR(I), RI(I)
C 700 FORMAT(5X,E15.6,14XE15.6)
5 CONTINUE
RETURN
END
C SUBROUTINE HSBG
C PURPOSE HSBG 40
C TO REDUCE A REAL MATRIX INTO UPPER ALMOST TRIANGULAR FORM HSBG 70
C USAGE HSBG 90
C CALL HSBG(N, A, IA) HSBG 100
C DESCRIPTION OF THE PARAMETERS HSBG 120
C N ORDER OF THE MATRIX HSBG 130
C A THE INPUT MATRIX, N BY N HSBG 140
C IA SIZE OF THE FIRST DIMENSION ASSIGNED TO THE ARRAY HSBG 150
C A IN THE CALLING PROGRAM WHEN THE MATRIX IS IN HSBG 160
C DOUBLE SUBSCRIPTED DATA STORAGE MODE. IA=N WHEN HSBG 170
C THE MATRIX IS IN SSP VECTOR STORAGE MODE. HSBG 180
C REMARKS HSBG 200
C THE HESSENBERG FORM REPLACES THE ORIGINAL MATRIX IN THE HSBG 210
C ARRAY A. HSBG 220
C SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED HSBG 230
C NONE HSBG 240
C METHOD HSBG 250
C SIMILARITY TRANSFORMATIONS USING ELEMENTARY ELIMINATION HSBG 260
C MATRICES, WITH PARTIAL PIVOTING. HSBG 270
C REFERENCES HSBG 280
C J.H. WILKINSON - THE ALGEBRAIC EIGENVALUE PROBLEM - HSBG 290
C CLARENDON PRESS, OXFORD, 1965. HSBG 300
C SUBROUTINE HSBG(N, A, IA) HSBG 310
DIMENSION A(36)
L=N
HSBG 400
NIA=L#IA
LIA=NIA-IA

L IS THE ROW INDEX OF THE ELIMINATION

20 IF(L-3) 360, 40, 40
40 LIA=LIA-IA
L1=L-1
L2=L1-1

SEARCH FOR THE PIVOTAL ELEMENT IN THE LTH ROW

ISUB=LIA+L
IPIV=ISUB-IA
PIV=ABS(A(IPIV))
IF(L-3) 90, 90.50
50 M=IPIV-IA
DO 80 I=L, M, IA
T=ABS(A(I))
IF(T-PIV) 80, 80, 60
60 IPIV=I
PIV=T
80 CONTINUE
90 IF(PIV) 100, 320, 100
100 IF(PIV-ABS(A(ISUB))) 180, 180, 120

INTERCHANGE THE COLUMNS

120 M=IPIV-L
DO 140 I=1, L
J=M+I
T=A(J)
K=LIA+I
A(J)=A(K)
140 A(K)=T

INTERCHANGE THE ROWS

M=L2-M/IA
DO 160 I=L1, NIA, IA
T=A(I)
J=I-M
A(I)=A(J)
160 A(J)=T

TERMS OF THE ELEMENTARY TRANSFORMATION

180 DO 200 I=L, LIA, IA
200 A(I)=A(I)/A(ISUB)

RIGHT TRANSFORMATION

J=-IA
DO 240 I=1, L2
J=J+IA
LJ=L-J
DO 220 K=1, L1
KJ=K+J
KL=K+LIA
220 A(KJ)=A(KJ)-A(LJ)*A(KL)
240 CONTINUE

LEFT TRANSFORMATION

K=-IA
DO 300 I=1, N
SUBROUTINE ATEIG

PURPOSE
COMPUTE THE EIGENVALUES OF A REAL ALMOST TRIANGULAR MATRIX

USAGE
CALL ATEIG(M, A, RR, RI, IANA, IA)

DESCRIPTION OF THE PARAMETERS
M ORDER OF THE MATRIX
A THE INPUT MATRIX, M BY M
RR VECTOR CONTAINING THE REAL PARTS OF THE EIGENVALUES ON RETURN
RI VECTOR CONTAINING THE IMAGINARY PARTS OF THE EIGENVALUES ON RETURN
IANA VECTOR WHOSE DIMENSION MUST BE GREATER THAN OR EQUAL TO M, CONTAINING ON RETURN INDICATIONS ABOUT THE WAY THE EIGENVALUES APPEARED (SEE MATH. DESCRIPTION)
IA SIZE OF THE FIRST DIMENSION ASSIGNED TO THE ARRAY A IN THE CALLING PROGRAM WHEN THE MATRIX IS IN DOUBLE SUBSCRIPTED DATA STORAGE MODE. IA=M WHEN THE MATRIX IS IN SSP VECTOR STORAGE MODE.

REMARKS
THE ORIGINAL MATRIX IS DESTROYED
THE DIMENSION OF RR AND RI MUST BE GREATER OR EQUAL TO M

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
NONE

METHOD
QR DOUBLE ITERATION

REFERENCES

SUBROUTINE ATEIG(M, A, RR, RI, IANA, IA)
DIMENSION A(36), RR(6), RI(6), PRR(2), PRI(2), IANA(36)
INTEGER P, PI, Q

C
E7=1.0E-8
E6=1.0E-6
E10=1.0E-10
DELTA=0.5
MAXIT=30

INITIALIZATION
N=N
20 N1=N-1
IN=N1+1A
NN=IN+N
IF(N1) 30, 1300, 30
30 NP=N+1

ITERATION COUNTER
IT=0

ROOTS OF THE 2ND ORDER MAIN SUBMATRIX AT THE PREVIOUS ITERATION
DO 40 I=1,2
PRR(I)=0.0
40 PRI(I)=0.0

LAST TWO SUBDIAGONAL ELEMENTS AT THE PREVIOUS ITERATION
PAN=0.0
PAN1=0.0

ORIGIN SHIFT
R=0.0
S=0.0

ROOTS OF THE LOWER MAIN 2 BY 2 SUBMATRIX
N2=N1-1
IN1=IN-IA
NN1=IN1+N
N1=IN+N1
N1IN1=IN1+11
60 T=A(N1N1)-A(NN)
   U=T*T
   V=4.0*A(NIN1)*A(NN)
   IF(ABS(V)-U)*E7) 100, 100, 65
   65 T=U+V
   IF(ABS(T)-AMAX1(U,ABS(V))*E6) 67, 67, 68
   67 T=0.0
   68 U=(A(N1N1)+A(NN))/2.0
   V=SGRT(ABS(T))/2.0
   IF(T)140.70.70
   70 IF(U) 80, 75.75
   75 RR(N1)=U-V
   RR(N)=U-V
   GO TO 130
   80 RR(N1)=U-V
   RR(N)=U-V
   GO TO 130
   100 IF(T)120, 110, 110
   110 RR(N1)=A(N1N1)
   RR(N)=A(NN)
   GO TO 130
   120 RR(N1)=A(NN)
   RR(N)=A(N1N1)

58
130 RTT=0.0
   RTT=0.0
   GO TO 160
140 RR(N1)=U
   RR(N)=U
   RI(N1)=V
   RI(N)=V
160 IF(N2)>1280, 1280, 180
   TESTS OF CONVERGENCE
   180 N1N2=N1N1-IA
      RMOD=RR(N1)*RR(N1)+RI(N1)*RI(N1)
      EPS=E10*SGRT(RMOD)
      IF(ABS(A(N1N2))-EPS)<1280, 1280, 240
      240 IF(ABS(A(NN1))<E10*ABS(A(NN1)))<1300, 1300, 250
      250 IF(ABS(PAN1-A(N1N2))-ABS(A(N1N2))*E6)<1240, 1240, 260
      260 IF(ABS(PAN-A(N1N1))-ABS(A(N1N1))*E6)<1240, 1240, 300
      300 IF(IT-MAXIT)<320, 320, 1240
      COMPUTE THE SHIFT
      320 J=1
      DO 360 I=1,2
         K=NP-I
         IF(ABS(RR(K)-PRI(I))+ABS(RI(K))-PRI(I))<DELT*(ABS(RR(K))
            1 +ABS(RI(K)))<340, 360, 360
      340 J=J+1
      360 CONTINUE
      GO TO (440, 460, 460, 480), J
      EPS=RTT
      S=0.0
      GO TO 500
      J=N+2-J
      R=RR(J)*RR(J)
      S=R+R
      GO TO 500
      R=RR(N)*RR(N1)-RI(N)*RI(N1)
      S=RR(N)+RR(N1)
      SAVE THE LAST TWO SUBDIAGONAL TERMS AND THE ROOTS OF
      THE SUBMATRIX BEFORE ITERATION
      500 PAN=A(NN1)
      PANI=A(N1N2)
      DO 520 I=1,2
      K=NP-I
      PRR(I)=RR(K)
      PRI(I)=RI(K)
      520 PRI(I)=RI(K)
      SEARCH FOR A PARTITION OF THE Matrix, DEFINED BY P AND Q
      P=N2
      IF (N-3)<600, 600, 525
      525 P=N1N2
      DO 580 J=2, N2
         IF(APIP)<1
         IF(APIP)<EPS)<600, 600, 530
      530 IF(APIP)<1
         IF(APIP)<EPS)<600, 600, 530
      530 IF(APIP)<1
         IF(APIP)<EPS)<600, 600, 530
      540 IF(ABS(AAPIP)+ABS(AAPIP1))<ABS(AAPIP)+ABS(AAPIP2)<1
         IF(D)<540, 540, 540
      540 IF(ABS(AAPIP)+ABS(AAPIP1))<ABS(AAPIP)+ABS(AAPIP2)<1
         IF(D)<540, 540, 540
      550 IF(ABS(AAPIP)+ABS(AAPIP1))<ABS(AAPIP)+ABS(AAPIP2)<1
         IF(D)<540, 540, 540
      560 P=N1
580 CONTINUE
600 G=P
   GO TO 680
620 P1=P-1
   G=P1
   IF (P1-1) 680, 680, 650
630 DO 660 I=2,P1
   IP1=IP1-IA-1
   IF(ABS(A(IP1))-EPS)680,650,660
   G=G-1
   GO TO 680
660 C   QR DOUBLE ITERATION
C
680 II=(P-1)*IA+P
   DO 1220 I=P,N1
   II=II-IA
   IIP=II+IA
   IF(I-P)720,700,720
   IPI=II+1
   IPIP=IIP+1
   GO TO 780
700   C   INITIALIZATION OF THE TRANSFORMATION
C
720   G1=A(II)*(A(II)-S)+A(IIP)*A(IPI)+R
   G2=A(IP1)*(A(IP1)+A(II)-S)
   G3=A(IP1)*A(IP1+1)
   A(IP1+1)=0.0
   GO TO 780
760   G3=0.0
780   CAP=SQRT(G1*G1+G2*G2+G3*G3)
   IF(CAP)800,860,800
800   IF(G1)820,840,840
820   CAP=-CAP
840   T=G1+CAP
   PSII=G2/T
   PSI2=G3/T
   ALPHA=2.0/(1.0+PSI1*PSI1+PSI2*PSI2)
   GO TO 880
860   ALPHA=2.0
   PSII=0.0
   PSI2=0.0
880   IF(I-N2)900,960,960
900   IF(I-P)920,940,920
920   A(II1)=-CAP
   GO TO 960
940   A(II1)=-A(II1)
   C   ROW OPERATION
C
960   IJ=II
   DO 1040 J=I,N
   T=PSII*A(IJ+1)
   IF(I-N1)980,1000,1000
980   IP2J=IJ+2
   T=T+PSI2*A(IP2J)
1000  ETA=ALPHA*(T+A(IJ))
   A(IJ)=A(IJ)-ETA
   A(IJ+1)=A(IJ+1)-PSI1*ETA
   IF(I-N1)1020,1040,1040
1020  A(IP2J)=A(IP2J)-PSI2*ETA
1040  IJ=IJ+1
   GO TO 960
C
C COLUMN OPERATION

IF(I-N1)1080, 1060, 1060
1060 K=N
GO TO 1100
1080 K=I+2
1100 IP=IIP-I
DO 1180 J=0, K
JIP=IP+J
JI=JIP-IA
T=PSI1*A(JIP)
IF(I-N1)1120, 1140, 1140
1120 JIP2=JIP+IA
T=T+PSI2*A(JIP2)
1140 ETA=ALPHA*(T+A(JI))
A(JI)=A(JI)-ETA
A(JIP)=A(JIP)-ETA*PSI1
IF(I-N1)1160, 1180, 1180
1160 A(JIP2)=A(JIP2)-ETA*PSI2
1180 CONTINUE
IF(I-N2)1200, 1220, 1220
1200 JI=II+3
JIP=JI+IA
JIP2=JIP+IA
ETA=ALPHA*PSI2*A(JIP2)
A(JI)=-ETA
A(JIP)=-ETA*PSI1
A(JIP2)=A(JIP2)-ETA*PSI2
1220 II=IIP+1
IT=IT+1
GO TO 60
C END OF ITERATION
1240 IF(ABS(A(NN))<ABS(A(N1N2))) 1300, 1280, 1280
C TWO EIGENVALUES HAVE BEEN FOUND
C
1280 IANA(N)=0
IANA(N1)=2
N=N2
IF(N2)1400, 1400, 20
C ONE EIGENVALUE HAS BEEN FOUND
C
1300 RR(N)=A(NN)
RI(N)=0.0
IANA(N)=1
IF(N)1400, 1400, 1320
1320 N=N1
GO TO 20
1400 RETURN
END

C END OF ITERATION
1240 IF(ABS(A(NN))<ABS(A(N1N2))) 1300, 1280, 1280
C TWO EIGENVALUES HAVE BEEN FOUND
C
1280 IANA(N)=0
IANA(N1)=2
N=N2
IF(N2)1400, 1400, 20
C ONE EIGENVALUE HAS BEEN FOUND
C
1300 RR(N)=A(NN)
RI(N)=0.0
IANA(N)=1
IF(N)1400, 1400, 1320
1320 N=N1
GO TO 20
1400 RETURN
END
SUBROUTINE MATMUL(IMOT, A, N, M, B, L, C, NA, MA, NB, MB, NC, MC)
DIMENSION A(NA, MA), B(NB, MB), C(NC, MC)
C A, B, C ARE GENERAL MATRIX
C IF A X B = C, THEN IMOT IS 1
C IF A X B' = C, THEN IMOT IS 2
DO 1 I=1, N
DO 1 J=1, L
C(I, J)=0.0
DO 1 K=1, M
GO TO (2, 3), IMOT
2 B1=B(K, J)
GO TO 1
3 B1=B(J, K)
1 C(I, J)=C(I, J)+A(I, K)*B1
RETURN
END
SUBROUTINE MATASCIADS A, N, M, B, C, NA, MA)
DIMENSION A(NA, MA), B(NA, MA), C(NA, MA)
C IF A + B = C, THEN IAOS IS 1
C IF A - B = C, THEN IAOS IS 2
IF(IAOS .NE. 1) GO TO 10
DO 1 I=1, N
DO 1 J=1, M
1 C(I, J)=A(I, J)+B(I, J)
RETURN
10 DO 2 I=1, N
DO 2 J=1, M
2 C(I, J)=A(I, J)-B(I, J)
RETURN
END
SUBROUTINE MATVEC(A, N, M, B, C, NA, MA)
DIMENSION A(NA, MA), B(MA), C(NA)
DO 1 I=1, N
C(I)=0.0
DO 1 J=1, M
1 C(I)=C(I)+A(I, J)*B(J)
RETURN
END
SUBROUTINE VECAS(IAOS, A, B, C, N)
DIMENSION A(N), B(N), C(N)
C A, B, C ARE VECTORS
C IF A + B = C, THEN IAOS IS 1
C IF A - B = C, THEN IAOS IS 2
IF(IAOS .NE. 1) GO TO 10
DO 1 I=1, N
1 C(I)=A(I)+B(I)
RETURN
10 DO 2 I=1, N
2 C(I)=A(I)-B(I)
RETURN
END
SUBROUTINE MABCT(A, N, M, L, C, NA, MA, NB, MB, NC, MC, ND, MD)
DIMENSION A(NA, MA), B(NB, MB), C(NC, MC), D(ND, MD), AB(6, 6)
DO 10 I=1, N
DO 10 J=1, L
AB(I, J)=0.0
DO 10 K=1, M
AB(I, J)=AB(I, J)+A(I, K)*B(K, J)
10 CONTINUE
DO 20 I=1, N
DO 20 J=1, L
D(I, J)=0.0
DO 20 K=1, L
D(I, J)=D(I, J)+AB(I, K)*C(J, K)
20 CONTINUE
RETURN
END
APPENDIX B

THE MONTE-CARLO SIMULATION PROGRAM
THIS PROGRAMMING IS CALLED PHASET. ITS MAIN PURPOSE IS TO
ESTIMATE THE UNKNOWN PHASE USING THE PHASE LOCKED LOOP HAVING
A VERY NARROW BANDWIDTH.

PROGRAMMER
CHAN JUNE YOON
TEXAS A & M UNIVERSITY
START JUNE, 1978

COMMON/SAMPLE/NSPB, TB
COMMON/PHASE/PHEEG, PHEED
COMMON/ODDB/ENODDB, SJRDB
DIMENSION HMO(2,2), HM1(2,2), VESTO(4,4), VEST1(4,4), XESTO(4)
1, XEST1(4), VARINO(2,2), VARIN1(2,2), VO(2), V1(2)
DIMENSION GAINO(4,2), GAIN1(4,2)
REAL MEAN
LOGICAL*1 STRNG(8)
INTEGER*4 JTIME
CALL ASSIQN(5, 'SY:PHASET.DAT', 13, 'RDO', 'NC', 1)
CALL INPUT
READ(5, 1) NOCASE, NPRNT
1 FORMAT(2I5)
DO 2000 NCASE=1, NOCASE
READ(5, 2) NOSYM, ENODB
2 FORMAT(5E56)
KSMAX=NOSYM*NSPB
CALL INIT(XJI, XJQ, XESTO, XEST1, VESTO, VEST1, XPL, XPQ, VCD)
1, ERROR, ERRORS, MEAN, VARANS)
CALL GTIM(JTIME)
CALL TIMASC(JTIME, STRNG)
WRITE(6,7272) (STRNG(II),II=1,8)
7272 FORMAT(IX, 'START TIME IS ',XA1)
WRITE(6,50)
50 FORMAT(6X,2X,1I5,5X,5HERROR, 14X, 6HERRORATE, 11X, 6HERRORS, 12X)
1, 6HERRATS, 12X, 16HPEEED IN DEGREES, 5X, 17HMEAN AND VARIANCE)
DO 1000 KS=1, KSMAX
CALL SIGNAL(KS, BB, SI, SQ)
CALL RFI(KS, XJI, XJQ, YI, YQ)
CALL DATA(SI, SQ, YI, YQ, ZI, ZQ)
CALL VCOUT(KS, ZI, ZQ, XPI, XPQ, MEAN, VARANS)
CALL REFOEN(KS, 0, FTO, GTO, HMO)
CALL REFOEN(KS, 1, FTR1, QTR1, HM1)
CALL KALMAN(KS, ZI, ZQ, HMO, VESTO, XESTO, QAINO, VARINO, DETO, VO)
CALL KALMAN(KS, ZI, ZQ, HM1, VEST1, XEST1, QAIN1, VARIN1, DET1, V1)
CALL COST(KS, VO, VARINO, DETO, SUMO)
CALL COST(KS, V1, VARIN1, DET1, SUM1)
CALL STAND(KS, ZI, ZQ, SUMB, FTO, GTO, FTR1, QTR1)
1, AFSKO, AFSKI, BFSKO, BFSKI, SFSKO, SFSKI)
IB=1+IFIX((KS-5)/NSPB)
IF(MOD(KS, NSPB).NE.0) GO TO 1000
CALL DDCOM(KS, SUMO, SUMB, FTO, GTO, BB, ERROR, ERRATE)
CALL STDCOM(KS, SUMO, SUMB, FTO, GTO, BB, ERRORS, ERRATS)
IF(MOD(IB, NPRNT).EQ.0) WRITE(6, 100) IB, ERROR, ERRATE, ERRORS, ERRATS
1, PHEEED, MEAN, VARANS
100 FORMAT(2X, I5, 5E18.6, 2E13.6)
1000 CONTINUE
CALL GTIM(JTIME)
CALL TIMASC(JTIME, STRNG)
WRITE(6,7273) (STRNG(II),II=1,8)
7273 FORMAT(IX, 'TIME IS ',XA1)
REWIND 6
2000 CONTINUE
STOP
C BLOCK DATA
COMMON/SEED/IXS, JXS, IXJ1, JXJ1, IXJ2, JXJ2, IXN1, JXN1, IXN2, JXN2
COMMON/SAMPLE/NSPB, TB
COMMON/OPTION/NOS
COMMON/Delay/Delphi, Delmeg
COMMON/Sigma/Sigmaj, Sigma
COMMON/Phase/Phees, Pheed
COMMON/Color/PHIDJ, PHD1J, GAMDJ, GAMDJ
COMMON/ODD/ENODB, SJRDB
COMMON/PLLFLT/BNP, ESP, Delf
COMMON/FREGJ/FJ
COMMON/PHASIN/HO, P, Z, K1, KG, PHASP, PHASQ
REAL KI, KG
COMMON/TRACK/GAMMA(4, 4), PHEE(4, 4)
INTEGER*2 IXI(2), JX1(2), IX2(2), JX2(2), IX3(2), JX3(2), IX4(2), JX4(2)
1, IX5(2), JX5(2)
INTEGER*4 IXS, JXS, IXJ1, JXJ1, IXJ2, JXJ2, IXN1, JXN1, IXN2, JXN2
EQUIVALENCE (IXS, IX1), (JXS, JX1), (IXJ1, IX2), (JXJ1, JX2), (IXJ2, IX3)
1, (JXJ2, JX3), (IXN1, IX4), (JXN1, JX4), (IXN2, IX3), (JXN2, JX5)
DATA IX1, JX1/"136303, "053354, "041256, "141560/
DATA IX2, JX2/"176303, "037702, "141238, "056407, "125537, "103453, "055052, "032461/
DATA IX3, JX3/"176303, "037702, "141238, "056407, "125537, "103453, "055052, "032461/
DATA IX4, JX4, IX5, JX5/"034313, "103400, "021165, "104262, "072063, "122076, "016415, "041540/
END
C SUBROUTINE INPUT
COMMON/SAMPLE/NSPB, TB
COMMON/OPTION/NOS
COMMON/Delay/Delphi, Delmeg
COMMON/Phase/Phees, Pheed
COMMON/FREGJ/FJ
COMMON/ODD/ENODB, SJRDB
COMMON/PLLFLT/BNP, ESP, Delf
COMMON/PHASIN/HO, P, Z, K1, KG, PHASP, PHASQ
READ(5, 1) NSPB, TB
READ(5, 1) NOS, DELPHI
READ(5, 2) ENODB, SJRDB
PHEE=0.
C INITIALIZE PHEES AS PHEED
PHEED=0.
READ(5, 2) FJ, HO
READ(5, 3) BNP
1 FORMAT(I5, E15.6)
2 FORMAT(2E15.6)
3 FORMAT(E15.6)
RETURN
END
C SUBROUTINE INIT(XJJ, XJQ, XESTO, XEST1, VESTO, VEST1, XPI, XPG, VCO
1, ERROR, ERRORS, MEAN, VARANS)
COMMON/SAMPLE/NSPB, TB
COMMON/OPTION/NOS
COMMON/Delay/Delphi, Delmeg
COMMON/Sigma/Sigmaj, Sigma
COMMON/ODD/ENODB, SJRDB
COMMON/PLLFLT/BNP, ESP, Delf
COMMON/FREGJ/FJ
COMMON/Color/PHIDJ, PHD1J, GAMDJ, GAMDJ
COMMON/Phase/Phees, Pheed
COMMON/PHASIN/HO, P, Z, K1, KG, PHASP, PHASQ
REAL KI, KG, MEAN
COMMON/TRACK/GAMMA(4, 4), PHEE(4, 4)
DIMENSION XESTO(4), XEST1(4), VESTO(4, 4), VEST1(4, 4)
PI=4.*ATAN(1.)
DELMEG=DELPHI*2.*PI/TB
IF(ABS.EQ.1) GO TO 10
SUMF=0.
DO 15 K=1,NSPB
  15 SUMF=SUMF+(SIN((K-.5)*TB*DELMEG/NSPB))**2
SIGMAN=SQRT(SUMF)*10.**(2.*ENDDB/20.)
  GO TO 20
10 CONSTP=SQRT(NSPB/2.)*ABS(SIN(DDELPHI))
SIGMAN=CONSTP*10.**(2.*ENDDB/20.)
  20 SIGMAJ=10.**(2.*SQRT(2.))
C GENERATE THE COLOURED NOISE PARAMETERS AND ITS BANDWIDTH
C " THE RHO-FILTER AND ITS BANDWIDTH
T=TB/NSPB
POLEJ=-2.*PI*FJ
PHIDJ=EXP(POLEJ*T)
GAM=(PHIDJ-1.)/POLEJ
GAINK=1./SQRT(GAM**2/(1.-PHIDJ**2))
GAMDJ=GAINK*GAM
PHIOJ=0.
GAMOJ=0.
BNJ=-POLEJ/4.
BNR=BNP
POLER=-4.*BNR
PHIDR=EXP(POLER*T)
GAM=(PHIDR-1.)/POLER
GAIND=1./SQRT(GAM**2/(1.-PHIDR**2))
GAMDR=GAIND*GAM
PHIOR=0.
GAMOR=0.
DO 50 I=1,4
  DO 50 J=1,4
  GAMMA(I,J)=0.
50 PHEE(I,J)=0.
  GAMMA(1,1)=GAMDR
  GAMMA(2,2)=GAMDR
  GAMMA(3,3)=GAMDJ
  GAMMA(4,4)=GAMDJ
  PHEE(1,1)=PHIDR
  PHEE(2,2)=PHIDR
  PHEE(3,3)=PHIDJ
  PHEE(4,4)=PHIDJ
C GENERATE THE PHASE ESTIMATOR PARAMETERS
C A=2.*PI*ESP/360.
C TANHO=SIN(A)/COS(A)
C HO=(2.*PI*DELF)/TANHO
KQ=(B./3.)*BNP
Z=(4./3.)*BNP
P=KQ*Z/HO
KI=P/Z
PHASP=EXP(P*T)
PHASG=(PHASP-1.)/P
C C INITIALIZATION
C XJI=0.
  XJK=0.
  XPG=0.
  XPI=1./KI*(P-Z))
  VCO=0.
  DO 60 I=1,4
    XESTO(I)=0.
  60 XESTO(I)=0.
  DO 65 I=1,4
    DO 65 J=1,4
65 CONTINUE
  60 CONTINUE
VEST0(I, J)=0.
    IF(I .EQ. J) VESTO(I, J)=1.
65 CONTINUE
    DO 70 I=1, 4
    DO 70 J=1, 4
70 VEST1(I, J)=VESTO(I, J)
    ERROR=0.
    ERRORS=0.
    PHEED=0.
    MEAN=0.
    VARAN=0.

C
    WRITE(6, 99) ENODB, SJRDB
99 FORMAT(2X, 6HFORMAT=, E13. 6, 5X, 6HSGRDB=, E13. 6, /)
    WRITE(6, 100) NOS, NSPB, TB, DELPHI, PHEED
100 FORMAT(2X, 4HNSPB=, I2, 5X, 5HNSPB=, I5, 5X, 3HTB=, E13. 6, 5X
     1,7HDELPHI=, E13. 6, 5X, 6HPEES=, E13. 6, /)
    WRITE(6, 101) GAMDJ, PHIDJ, BNR
101 FORMAT(5X, 6HGAMDJ=, E13. 6, 5X, 6HPHIDJ=, E13. 6, 5X, 4HBNR=, E13. 6)
    WRITE(6, 102) GAMDR, PHIDR, BNP
102 FORMAT(5X, 6HGAMDR=, E13. 6, 5X, 6HPHIDR=, E13. 6, 5X, 4HBNR=, E13. 6)
    WRITE(6, 103) PHASP, PHASP, BNP
103 FORMAT(5X, 6HPHASP=, E13. 6, 5X, 6HPHASP=, E13. 6, 5X, 4HBNP=, E13. 6)
    WRITE(6, 105) HO, P, Z, KI, KG
105 FORMAT(2X, 18HPARAMETERS IN VCO=, /, 5X, 5HNSPB=, E13. 6, 5X, 2HP=, E13. 6
     1,5X, 2HZ=, E13. 6, 5X, 3HKI=, E13. 6, 5X, 3HKG=, E13. 6, //)
    REWIND 6
    RETURN
END

C
SUBROUTINE SIGNAL(KS, BB, SI, SQ)
COMMON/SEED/IXS, JXS, IXJ1, JXJ1, IXJ2, JXJ2, IXN1, JXN1, IXN2, JXN2
INTEGER*4 IXS, JXS, IXJ1, JXJ1, IXJ2, JXJ2, IXN1, JXN1, IXN2, JXN2
COMMON/SAMPLE/NSPB, TB
COMMON/OPTION/NOS
COMMON/PHASE/PHEED, PHEEO
COMMON/DELAY/DELPHI, DELMEG
IF(MOD(KS-1, NSPB).NE.0) GO TO 10
    CALL RANC(IXS, JXS, GB)
    BB=AINT(GB+. 5)
10 C=1.-2*BB
    TK=(KS-. 5)/NSPB
    TKMOD=(TK-IFIX(TK))*TB
    A=1.
    GO TO (1,2), NOS
1 PHEEM=DELPHI*C
    GO TO 20
2 PHEEM=DELMEG*C*TKMOD
20 SI=A*COS(PHEEM+PHEED)
    SQ=A*SIN(PHEEM+PHEED)
    RETURN
END

C
SUBROUTINE RFU(KS, XJ1, XJQ, YI, YQ)
COMMON/SEED/IXS, JXS, IXJ1, JXJ1, IXJ2, JXJ2, IXN1, JXN1, IXN2, JXN2
INTEGER*4 IXS, JXS, IXJ1, JXJ1, IXJ2, JXJ2, IXN1, JXN1, IXN2, JXN2
COMMON/COLOR/PHIDJ, PHIDJ, QAMDJ, QAMDJ
COMMON/SIGMA/SIGMAJ, SIGMAN
REAL NI, NQ
    CALL MARBA(IXJ1, JXJ1, WI)
    CALL MARBA(IXJ2, JXJ2, WQ)
    CALL MARBA(IXN1, JXN1, NI)
    CALL MARBA(IXN2, JXN2, NQ)
    XJ1=PHIDJ*XJ1+PHIDJ*YQ-QAMDJ*WQ+QAMDJ*WI
    XJQ=PHIDJ*XJ1+PHIDJ*YQ-QAMDJ*WQ+QAMDJ*WI
    YI=SIGMAJ*XJ1+SIGMAN*NI
SUBROUTINE DATA(SI, SQ, YI, YQ, ZI, ZQ)
    COMMON/PHASE/PHEE, PHEED
    ZI=SI+YI
    ZG=SG+YG
    ZI=ZI*COB(PHEED)+ZG*SIN(PHEED)
    ZG=-ZI*SIN(PHEED)+ZG*COS(PHEED)
    RETURN
END

SUBROUTINE VCOUT(KS, ZI, ZQ, XPI, XPQ, VCO, MEAN, VARANS)
    COMMON/SAMPLE/NSPB, TB
    COMMON/PHASE/PHEE, PHEED
    COMMON/PHA8E/HO, P, Z, KI, KQ, PHASP, PHASQ
    REAL KI, KG, MEAN
    XPI=PHEASP*XPI+PHASP*ZQ
    ZQ1=KQ*((P-Z)*XPQ+ZG)
    XPQ=XPQ1
    XPI=PHEASP*XPI+PHASP*ZI
    ZI1=KI*((P-Z)*XPI+ZI)
    XPQ1=ZQ1/ZI1
    T=TB/NSPB
    PHEED=PHEED+(VCDP1+VCO)*T/2.
    VCO=VCDP1
    ESTIMATE THE MEAN AND VARIANCE OF THE PHASE ERROR, RECURSIVELY
    MEAN=((KS-1. )*MEAN+PHEEOD)/KS
    VARANS=((KS-1. )*VARANS+EVAR)/KS
    RETURN
END

SUBROUTINE REFGEN(KS, M, FTR, GTR, HM)
    COMMON/SAMPLE/NSPB, TB
    COMMON/DELAY/DELPHI, DELMEG
    COMMON/OPTION/NGS
    DIMENSION HM(2,2)
    TK=(KS-.5)/NMB
    TKMOD=(TK-IFIX(TK))*TD
    AR=I.
    IF(NOS.NE.1) GO TO 1
    IF(M. EQ. 0) PHEEMR=DELPHI
    IF(M. EQ. 1) PHEEMR=-DELPHI
    GO TO 2
1     IF(M. EQ. 0) PHEEMR=DELMEG*TKMOD
     IF(M. EQ. 1) PHEEMR=-DELMEG*TKMOD
2     FTR=AR*COS(PHEEMR)
     GTR=AR*SIN(PHEEMR)
     HM(1,1)=COS(PHEEMR)
     HM(1,2)=SIN(PHEEMR)
     HM(2,1)=-SIN(PHEEMR)
     HM(2,2)=COS(PHEEMR)
     RETURN
END

SUBROUTINE KALMAN(KS, ZI, ZQ, HM, VEST, XEST, QAIN, VARINV, DET, V)
    COMMON/TRACK/GAMMA(4,4), PHEE(4,4)
    COMMON/SIGMA/SIGMAJ, SIGMAN
    DIMENSION VEST(4,4), PVP(4,4), QTP(4,4), VPR(4,4), VHT(4,2)
    1, HVHT(2,2), VAR(2,2), VARINV(2,2), QAIN(4,2), QH(4,4), HM(2,2)
DIMENSION VNN(2,2)
REAL IMGH(4,4)
DIMENSION XEST(4), XPRED(4), HXPRED(2), V(2), GV(4), HX(2,4)
DO 1 I=1,2
DO 1 J=1,2
1 HX(I,J)=H(I,J)
HX(1,3)=SIGMAJ
HX(1,4)=0.
HX(2,3)=0.
HX(2,4)=SIGMAJ
VNN(1,1)=SIGMAN**2
VNN(1,2)=0.
VNN(2,1)=0.
VNN(2,2)=SIGMAN**2
CALCULATE THE STEADY-STATE KALMAN GAIN
CALL MBCT(PHEE, 4, 4, VEST, 4, PHEE, 4, PVP, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4)
CALL MATMUL(2, GAMMA, 4, 4, GAMMA, 4, QTG, 4, 4, 4, 4, 4, 4)
CALL MATAS(1, PVP, 4, 4, QTG, VPRED, 4, 4)
CALL MBCT(HX, 2, 4, VPRED, 4, HX, 2, HVHT, 2, 4, 4, 4, 4, 4, 2, 4, 2, 4, 2)
CALL MATAS(1, HVHT, 2, 2, VNN, VAR, 2, 2)
DET=VAR(1, 1)*VAR(2, 2)-VAR(1, 2)*VAR(2, 1)
VARINV(1, 1)=VAR(2, 2)/DET
VARINV(1, 2)=-VAR(1, 2)/DET
VARINV(2, 1)=-VAR(2, 1)/DET
VARINV(2, 2)=VAR(1, 1)/DET
CALL MATMUL(2, VPRED, 4, 4, HX, 2, VHT, 4, 2, 4, 4, 4, 2)
CALL MATMUL(1, VHT, 4, 2, VARINV, 2, QAIN, 4, 2, 2, 4, 4, 4)
CALL MATMUL(1, QAIN, 4, 2, HX, 4, QH, 4, 2, 2, 4, 4, 4)
DO 10 I=1,4
DO 10 J=1,4
IMGH(I,J)=-GH(I,J)
IF(I.EQ.J) IMGH(I,J)=1.-QH(I,J)
10 CONTINUE
CALL MATMUL(1, IMGH, 4, 4, VPRED, 4, VEST, 4, 4, 4, 4, 4, 4)
CALL MATVEC(PHEE, 4, 4, XEST, XPRED, 4, 4)
CALL MATVEC(HX, 2, 4, XPRED, HXPRED, 2, 4)
V(1)=ZI-HXPRED(1)
V(2)=ZQ-HXPRED(2)
CALL MATVEC(QAIN, 4, 2, V, GV, 4, 2)
CALL VECAS(1, XPRED, GV, XEST, 4)
RETURN
END

SUBROUTINE COST(KS, V, VARINV, DET, SUM)
COMMON/SAMPLE/NSPB, TB
DIMENSION V(2), VARINV(2, 2)
IF(MOD(KS-1, NSPB).EQ.0) SUM=0.
ARG=-ALOG(DET)-(V(1)**2*VARINV(1, 1)+V(2)**2*VARINV(2,2)+V(1)*V(2)*(VARINV(1,2)+VARINV(2, 1)))
SUM=SUM+ARG
RETURN
END

SUBROUTINE STAND(KS, ZI, ZQ, SUM, FTRO, QTRO, FTR1, OTR1, AFSK0, AFSK1, BFSK0, BFSK1, BFSKO, BFSK1)
COMMON/SAMPLE/NSPB, TB
COMMON/OPTION/NOS
GO TO (1,2), NOS
1 IF(MOD(KS-1, NSPB).EQ.0) SUM=0.
SUM=SUM+ZQ
RETURN
2 IF(MOD(KS-1, NSPB).NE.0) GO TO 20
AFSK0=0.
AFSK1=0.
BFSKO=0.
SUBROUTINE DDCOM(KS, SUM0, SUM1, XEST0, XEST1, BB, ERROR, ERRATE)
COMMON/SAMPLE/NSPB, TB
DIMENSION XEST0(4), XEST1(4)
IF(SUM0.GT.SUM1) GO TO 10
BBHAT=1.
DO 1 I=1,4
1 XEST0(I)=XEST1(I)
GO TO 20
10 BBHAT=0.
DO 2 I=1,4
2 XEST1(I)=XEST0(I)
20 IF(BB.EQ.BBHAT) ERR0.
IF(BB.NE.BBHAT) ERR=1.
ERROR=ERROR+ERR
IB=1+IFIX((KS-.5)/NSPB)
ERRATE=ERROR/IB
RETURN
END

SUBROUTINE STDCOM(KS, SUM, SFSKO, SFSKI, BB, ERROR, ERRATE)
COMMON/SAMPLE/NSPB, TB
COMMON/OPTION/NOS
GO TO (1,2), NOS
1 IF(SUM.GE.0.) BBHAT=0.
IF(SUM.LT.0.) BBHAT=1.
GO TO 10
2 IF(SFSKO.GT.SFSKI) BBHAT=0.
IF(SFSKI.GT.SFSKO) BBHAT=1.
10 IF(BB.EQ.BBHAT) ERR=0.
IF(BB.NE.BBHAT) ERR=1.
IB=1+IFIX((KS-.5)/NSPB)
ERROR=ERROR+ERR
ERRATE=ERROR/IB
RETURN
END

SUBROUTINE MARSA(IXA, JXA, V)
INTEGER*4 IXA, JXA
CALL RANC(IXA, JXA, X1)
CALL RANC(IXA, JXA, X2)
X1=(X1-.5)*2.
X2=(X2-.5)*2.
5 W=X1**2+X2**2
IF(W.LE.1.) GO TO 10
CALL RANC(IXA, JXA, X1)
CALL RANC(IXA, JXA, X2)
X1=(X1-.5)*2.
X2=(X2-.5)*2.
GO TO 5
10 XX=X1*SQRT(-2.*ALOG(W)/W)
V=X2*XX/X1
RETURN
END
REFERENCES


