AESOP: An Interactive Computer Program for the Design of Linear Quadratic Regulators and Kalman Filters

Bruce Lehtinen
and Lucille C. Geyser
AESOP: An Interactive Computer Program for the Design of Linear Quadratic Regulators and Kalman Filters

Bruce Lehtinen
and Lucille C. Geyser

Lewis Research Center
Cleveland, Ohio
# Contents

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
</tr>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Theoretical Background and Problem Formulation</td>
</tr>
<tr>
<td>Open-Loop System Description</td>
</tr>
<tr>
<td>Eigenvalues and Eigenvectors</td>
</tr>
<tr>
<td>Controllability, Observability, and Mode Shapes</td>
</tr>
<tr>
<td>Residues</td>
</tr>
<tr>
<td>Steady-State Linear Quadratic Regulator (LQR) Design</td>
</tr>
<tr>
<td>Steady-State Kalman Filter Design</td>
</tr>
<tr>
<td>Normalization</td>
</tr>
<tr>
<td>Stochastic Linear Quadratic Regulator Design</td>
</tr>
<tr>
<td>System Response to Noise Inputs</td>
</tr>
<tr>
<td>Transfer Functions and Frequency Response Calculations</td>
</tr>
<tr>
<td>Transient Response Calculation</td>
</tr>
<tr>
<td>AESOP Program Operation</td>
</tr>
<tr>
<td>Program Structure</td>
</tr>
<tr>
<td>Operating Procedure</td>
</tr>
<tr>
<td>Data Input and Output</td>
</tr>
<tr>
<td>Description of AESOP Functions</td>
</tr>
<tr>
<td>Series 100 – Program Control</td>
</tr>
<tr>
<td>Series 200 – Data Input and Revision</td>
</tr>
<tr>
<td>Series 300 – Matrix Formation</td>
</tr>
<tr>
<td>Series 400 – Open-Loop System Analysis</td>
</tr>
<tr>
<td>Series 500 – Frequency Responses and Bode Plots; and Series 700 – Transfer Functions</td>
</tr>
<tr>
<td>Series 600 – Transient Responses</td>
</tr>
<tr>
<td>Series 800 – LQR and Filter Design</td>
</tr>
<tr>
<td>Series 900 – User-Supplied Subroutines</td>
</tr>
<tr>
<td>Concluding Remarks</td>
</tr>
<tr>
<td>Appendixes</td>
</tr>
<tr>
<td>A – Symbols</td>
</tr>
<tr>
<td>B – Subroutine Descriptions</td>
</tr>
<tr>
<td>C – Test Cases</td>
</tr>
<tr>
<td>Test Case I – Third-Order Problem to Demonstrate Full AESOP Program Capabilities</td>
</tr>
<tr>
<td>Terminal Printout for Test Case I</td>
</tr>
<tr>
<td>Test Case II – Interactive Design of a Nonzero-Set-Point Regulator</td>
</tr>
<tr>
<td>D – Terminal Output Options and Main PROCDEF</td>
</tr>
<tr>
<td>Standard Terminal Output</td>
</tr>
<tr>
<td>Extended Terminal Output</td>
</tr>
<tr>
<td>PROCDEF AESRUN</td>
</tr>
<tr>
<td>E – Flow Chart</td>
</tr>
<tr>
<td>F – Prerequisite Table</td>
</tr>
<tr>
<td>References</td>
</tr>
</tbody>
</table>
Summary

AESOP is a computer program for use in designing feedback controls and state estimators for linear multivariable systems. AESOP is meant to be used in an interactive manner. Each design task that the program performs is assigned a “function” number. The user accesses these functions either (1) by inputting a list of desired function numbers or (2) by inputting a single function number. In the latter case the choice of the function will in general depend on the results obtained by the previously executed function.

The most important of the AESOP functions are those that design linear quadratic regulators and Kalman filters. The user interacts with the program when using these design functions by inputting design weighting parameters and by viewing graphic displays of designed system responses. Supporting functions are provided that obtain system transient and frequency responses, transfer functions, and covariance matrices. The program can also compute open-loop system information such as stability (eigenvalues), eigenvectors, controllability, and observability.

The program is written in ANSI-66 Fortran for use on an IBM 3033 using TSS 370. Descriptions of all subroutines and results of two test cases are included in the appendixes.

Introduction

The computer program called AESOP (Algorithms for EStimator and OPtimal regulator design) was written to solve a number of problems associated with the design of controls and state estimators for linear time-invariant systems. The systems considered are modeled in state-variable form by a set of linear differential and algebraic equations with constant coefficients. Two key problems solved by AESOP are the linear quadratic regulator (LQR) design problem and the steady-state Kalman filter design problem. The remainder of AESOP is devoted to calculations in support of these two problems, mainly for analyzing the open-loop system and evaluating the resulting control or estimator designs. Thus the overall program can be subdivided as follows:

1. Open-loop system analyses
2. Control and filter design
3. System response calculations

The AESOP program was developed at Lewis for use in conducting design studies in propulsion system control (refs. 1 and 2). AESOP was an outgrowth of a previously developed control design program called LSOCE (ref. 3), which had been used in supersonic inlet controls development (refs. 4 and 5). AESOP differs from LSOCE mainly in that it was designed to be operated in an interactive manner, whereas LSOCE was strictly a batch type of program. In addition, AESOP contains system response and evaluation features that are not present in LSOCE. These additions tend to enhance AESOP's use as an interactive design tool.

Other control design computer programs appearing in the literature perform computations similar to those of AESOP. Notable among the original LQR design packages are ASP by Kalman and Englars (ref. 6) and its Fortran version VASP (ref. 7). Subsequent LQR design packages were the OPTSYS program of Bryson and Hall (ref. 8), the ORACL program of Armstrong (ref. 9), and Honeywell's DIGIKON (ref. 10). Computer-aided control system design program development has accelerated in recent years. A good summary of this development is contained in reference 11. Here, over 20 control design programs and packages, including AESOP, are discussed in varying degrees of detail. They represent a variety of design methodologies, ranging from classical single-loop approaches to multivariable LQR and multivariable frequency domain approaches, for both continuous and discrete formulations. Most are written in Fortran and have some sort of interactive capability, but except for a few commercially available packages, most are neither completely documented nor generally available. AESOP, at the present time, is the only interactive LQR type of control design program that is in the public domain. (Contact COSMIC, The University of Georgia, Athens, Ga. 30602, concerning the availability of this program.)

The AESOP program is structured around a list of predefined and numbered functions. Each function performs, basically, a single computation associated with control, estimation, or system response determination. For example, one AESOP function computes the eigenvalues of the open-loop system matrix A, another function reads in the A matrix, etc. These functions are described fully in the section Description of AESOP Functions. The use of these functions and the parts they play in AESOP can be described in general terms with the aid of figure 1. The figure illustrates what the user of the program does (left side of fig. 1), what the program does (right side of fig. 1), and the interaction between the user and the program.

The user begins by defining the problem to be solved (e.g., by defining the matrices that define the state-variable model of the open-loop system). The user then provides this information to AESOP as input data, generally storing it in a data file. Next the program "prompts" the user to enter a list of function numbers that are to be performed, in sequence, by AESOP to solve the user's problem. Usually this list of numbers is entered at a terminal, but it can also be entered from a pre-stored data file. The AESOP program then executes the desired functions in proper sequence, storing away all results on an output file (fig. 1) as "off-line output" but
also displaying selected portions of the results back at the user’s terminal (fig. 1) as "on-line display." The user then decides whether to terminate the program or to request that more functions be performed. The program again prompts the user to enter numbers that define the new functions, which can be entered singly or as a number string. Typically, one of the functions a user would enter, at this time, would be one that allows the user to vary some problem parameter. In this way the user can effectively interact with the program in an on-line manner to achieve the desired design results. At the conclusion of the terminal session the user commands the output data file to be printed and hard copies to be made of any graphic output generated that was not previously displayed on-line.

This concludes the overview of the basic operation of the AESOP program. The next section describes the various design problems that can be solved by using AESOP, indicating what function numbers the user would request in order to perform the computations. Following that section is the AESOP Program Operation. Here, examples are presented of a typical dialog between the user and the program. Following that is the section Description of AESOP Functions, which describes each of the 78 functions, what input each requires, and what calculations each performs. These latter two sections serve as a guide to which the user can refer as necessary while running the program. Appendixes include information such as a symbol table, brief subroutine descriptions, and two test cases that are useful both for program checkout and for gaining an understanding of how the program operates.

Theoretical Background and Problem Formulation

The computations performed by the AESOP program can be grouped into five basic categories. These categories are illustrated in figure 2. This section presents the equations that define the various problems to be solved and indicates the solution methods used by AESOP. After reading this and the next section, the reader should be able to use AESOP to solve a number of "standard" problems by using "standard" sequences of function numbers. The reader will then be able to devise other function number sequences that would allow other more specialized problems to be solved.

Open-Loop System Description

Before beginning any control or filter design on a linear dynamic system, it is important to thoroughly analyze the open-loop system under consideration. The linear open-loop system defined for use throughout this
Transfer functions and frequency response

LQR and Kalman filter design

System transient responses

System response to noise inputs

Figure 2. – Control system design computations performed by AESOP program.

The report is given in the following state-variable form and shown schematically in Figure 3. The state equation is

\[ \dot{x} = Ax + Bu + Dw \]  

where

- \( x \) : \( \text{N}^{\text{th}} \)-order state vector
- \( u \) : \( \text{NC}^{\text{th}} \)-order control vector
- \( w \) : \( \text{ND}^{\text{th}} \)-order white-noise disturbance vector

and \( A, B, \) and \( D \) are matrices of appropriate dimensions. Fortran symbols for matrices used in the AESOP program coding are used herein whenever possible. A measurement equation, defining the system's measurable output vector is

\[ z = z_1 + v \]  
\[ z_1 = Hx \]  

Figure 3. – Block diagram of open-loop system.

where

\[ z \] : \( \text{NM}^{\text{th}} \)-order measurement vector
\[ v \] : \( \text{NM}^{\text{th}} \)-order white-noise measurement vector
\[ H \] : \( \text{NM} \)-by-\( \text{N} \) matrix

In addition to the measurement vector an output vector, which represents unmeasurable outputs, is defined as

\[ y = Cx + D_{\text{OUT}}u \]  

where \( y \) is an \( \text{NO}^{\text{th}} \)-order output vector. Finally, a set of noise-free measurements called a set-point vector are defined. These represent outputs (NC in number) that are to be regulated to desired constant set-point values. This vector is given as

\[ y_{\text{SP}} = CSPx \]  

where \( y_{\text{SP}} \) is an \( \text{NC}^{\text{th}} \)-order set-point vector.
Eigenvalues and Eigenvectors

Of prime importance in designing a control system is knowledge of the open-loop system structure and stability. This knowledge affects the designer’s choice of performance index weighting matrices, sensed variables to use for control or estimation, etc.

Open-loop stability is determined by the eigenvalues of the system A matrix. The system is stable if and only if these eigenvalues \( \lambda_i \) (\( i = 1, 2, \ldots, N \)) all have negative real parts. Consider the unforced version of equation (1),

\[
\dot{x} = Ax
\]

Define a new state vector \( \tilde{x} \), relating to \( x \) through the transformation matrix \( T \) as

\[
T \tilde{x} = x
\]

Substitute for \( x \) in equation (5) to obtain

\[
\dot{\tilde{x}} = T^{-1} A T \tilde{x}
\]

If we let \( T^{-1} A T \) be equal to a diagonal matrix \( \Lambda \), equation (7) can be rewritten as

\[
\dot{\tilde{x}} = \Lambda \tilde{x}
\]

The diagonal elements of \( \Lambda \) are the eigenvalues of \( A \), \( T \) is the eigenvector matrix (a matrix whose columns are the eigenvectors of \( A \)), and \( \tilde{x} \) is defined as the modal state vector. The value of \( \Lambda \) is computed by using AESOP function 501, and the eigenvectors are obtained by using function 402. To avoid complex arithmetic, a block diagonal form is used for the matrix \( \Lambda \) such that a complex eigenvalue pair \( (\lambda_i, \lambda_{i+1}) = (\alpha + j\beta, \alpha - j\beta) \) appears in the 2-by-2 diagonal block of \( \Lambda \),

\[
\begin{align*}
\alpha & \quad -\beta \\
\beta & \quad \alpha
\end{align*}
\]

where the \( \alpha \)'s are along the diagonal. Let the complex eigenvector pair \( (\eta_i, \eta_{i+1}) = (\gamma + j\delta, \gamma - j\delta) \) correspond to the complex eigenvalue pair \( \alpha \pm j\beta \). Then in the columns corresponding to the defined diagonal block of \( \Lambda \) there appear two real vectors \( t_i \) and \( t_{i+1} \) defined as

\[
\begin{align*}
t_i &= \gamma + \delta \\
t_{i+1} &= \gamma - \delta
\end{align*}
\]

Hence, when \( A \) is block diagonal, \( T \) is called a modified eigenvector matrix. AESOP function 402 also calculates the so-called mode shapes. For a real eigenvector the mode-shape vector is the same as the eigenvector. However, for a complex eigenvector pair the corresponding mode-shape pair contains, in successive columns, the magnitude of \( \eta \) and the phase of \( \eta \). Each mode-shape vector is normalized by dividing all of the elements by the magnitude of the largest element. The phase of the largest element is set to zero, and the phases of all other components of the vector are adjusted accordingly.

Controllability, Observability, and Mode Shapes

Once the eigenvalues and eigenvectors (mode-shape vectors) are calculated, it is an easy task to determine controllability and observability. For this purpose, system equations (1) and (2) can be rewritten in terms of the modal state vector as

\[
\dot{x} = \Lambda x + T^{-1} B u
\]

\[
z = H T \tilde{x} + v
\]

System controllability is determined by examining the elements of \( T^{-1} B \). The system is uncontrollable if elements in a row of \( T^{-1} B \) are zero, meaning that it is impossible to excite a component of the modal state vector \( \tilde{x} \) with the control vector \( u \). Also, using this modal formulation, one can think of the matrix \( T^{-1} B \) as being the control effectiveness matrix. That is, the relative magnitudes of the row elements of \( T^{-1} B \) define the relative influence each control input has on a modal state variable (mode). For a meaningful comparison, however, the control inputs must be normalized (nondimensionalized). Normalization can be done by using AESOP function number 404. Normalization (and unnormalization) is discussed in detail later in this section. The control effectiveness matrix is calculated by AESOP function 403.

System observability can be determined similarly by using the modal state form. From equation (10) it can be seen that, if all elements of column \( k \) of the \( H T \) matrix are zero, modal state \( k \) will be unobservable through measurement \( z \). Also, the relative magnitudes of the elements of row \( k \) of \( H T \) define the relative contribution each mode makes to measurement \( z_k \). This information is useful, for example, if one wishes to minimize the number of measurements (sensors) required when designing a control system that is to shift certain system poles (modes, modal states). System observability (the \( H T \) matrix) is computed in AESOP by using function number 403.
In AESOP both the controllability matrix \((T^{-1} B)\) and the observability matrix \((H^T)\) are printed out in mode-shape format. This means that, for \(T^{-1} B\), when two successive rows \(k\) and \(k+1\) relate to a complex modal state pair \((x_k, x_{k+1})\), the \(k\)th elements in the columns of \(T^{-1} B\) are magnitudes and the \((k+1)\)th elements are phase angles. Similarly, for the \(H^T\) matrix, for a complex modal state pair, elements in the \(k\)th column of \(H^T\) are magnitudes and those in the \((k+1)\)th column are phase angles.

Residues

The availability of matrices \(H^T\) and \(T^{-1} B\) makes it very easy to compute the system residues. Consider the system in modal state vector form given by equations (9) and (10). Let \(B = T^{-1} B\) and \(H = H^T\). Thus equations (9) and (10) can be written as

\[ \dot{x} = A \bar{x} + \bar{B} u \]  
\[ z = H\bar{x} + v \]  
(10a)

For a single-input–single-output linear system a transfer function \(g(s)\) can be written in so-called residue form as

\[ g(s) = \sum_{j=1}^{N} \frac{r_j}{s - \lambda_j} \]  
(10c)

where each of the \(N\) constants \(r_j\) is defined as a residue at the transfer function pole \(\lambda_j\). The residues define the relative magnitude with which the system input affects the system output through each system pole. This single input/output concept generalizes directly to the multiple input/output case. Here the transfer function matrix \(G(s)\) for the system of equations (10a) and (10b) can be written as

\[ G(s) = H(sI - A)^{-1} \bar{B} \]  
(10d)

or in residue form

\[ G(s) = \sum_{j=1}^{N} \frac{R_j}{s - \lambda_j} = H(sI - A)^{-1} \bar{B} \]  
(10e)

where now the \(N\) elements \(R_j\) are residue matrices. Since \(A\) is a diagonal matrix, we can rewrite the matrix \((sI - A)^{-1}\) as

\[ (sI - A)^{-1} = \text{diag}\left(\frac{1}{s - \lambda_j}\right) = \sum_{j=1}^{N} \frac{E_j}{s - \lambda_j} \]  
(10f)

Substituting from equation (10f) into equation (10e), we obtain an equation that defines the residue matrices, namely,

\[ \sum_{j=1}^{N} \frac{R_j}{s - \lambda_j} = \sum_{j=1}^{N} \frac{H E_j \bar{B}}{s - \lambda_j} \]  
(10g)

Thus the \(j\)th residue matrix is simply

\[ R_j = H E_j \bar{B} \]  
(10h)

For a real eigenvalue \(\lambda_j\) the elements of the corresponding residue matrix \(R_j\) are real, being computed simply as the (outer) product of the \(j\)th column of \(H\) and the \(j\)th row of \(B\).

For a complex eigenvalue pair \((\lambda_j, \lambda_{j+1})\) AESOP makes use of the modified eigenvector matrix form for \(T\), which means that \(H\) and \(B\) are also used in that form. Thus real arithmetic can be used in computing the real and imaginary parts of the residue matrix. AESOP prints out that matrix in polar form (one matrix of residue magnitudes followed by one matrix of residue phase angles). The residues are computed along with open-loop controllability and observability checks in function 403.

Steady-State Linear Quadratic Regulator (LQR) Design

One of the primary functions of the AESOP program is to compute solutions to the steady-state linear quadratic regulator problem. Because this problem has been well documented (ref. 12, e.g.), the results are only briefly summarized herein. The system to be controlled is described by

\[ \dot{x} = A x + B u \]  
(11)

where the state \(x\) is assumed to be measurable and no plant disturbances are present.

A control that minimizes the quadratic performance index
\[ J = \int_0^\infty \{ x^T Q C x + 2x^T N N u + u^T (P C I N V)^{-1} u \} dt \]  

is given by

\[ u = -K C x \]  

For the optimal solution to exist, weighting matrices \( Q C, N N, \) and \( P C I N V \) must be as follows:

1. \( P C I N V \) is positive-definite
2. \( Q C \) can be written as \( Q C = M \bar{Q} M^T \) where the pair \( (M, A) \) is observable and \( \bar{Q} \) is symmetric and positive-definite
3. \( Q C - N N \cdot P C I N V \cdot N N^T \) is nonnegative-definite.

Feedback gain matrix \( K C \) is found by solving the following matrix Riccati equation for matrix \( S S \):

\[
S S (A - B \cdot P C I N V \cdot N N^T) + (A - B \cdot P C I N V \cdot N N^T)^T S S \\
- S S (B \cdot P C I N V \cdot B T) S S + (Q C - N N \cdot P C I N V \cdot N N^T) = 0
\]  

Then \( K C \) is given as

\[ K C = P C I N V (B T \cdot S S + N N^T) \]  

The Riccati solution matrix \( S S \) theoretically is positive-definite and symmetric. Three error checks are provided in AESOP (functions 805, 806, and 807) to determine the accuracy of the computed \( S S \). The eigenvalues of \( S S \) are computed and should be positive and real. The differences of the off-diagonals are displayed as a symmetry check. Finally, the computed \( S S \) is substituted back into equation (14) and a residual matrix is computed.

The standard steady-state linear quadratic regulator problem just outlined assumes that no command inputs are present. This problem can be modified to include set-point inputs by introducing a set of NC set-point outputs defined by

\[ y_{sp} = C S P x \]  

These outputs are to be made equal, in steady state, to NC corresponding desired set points \( y_{sp} \). This is the so-called nonzero-set-point regulator problem of Kwakernaak (ref. 12). The solution is to allow a feedforward term in the control such that the control law of equation (13) is modified to the form

\[ u = -K C x + K F F y_{spd} \]  

Figure 5 shows the configuration of the nonzero-set-point regulator. By stipulating that in steady state \( y_{sp} = y_{spd} \), matrix \( K F F \) can be computed as

\[ K F F = [-C S P \cdot (A - B \cdot K C)^{-1} \cdot B]^{-1} \]  

Thus, with NC degrees of control freedom available, NC outputs \( (y_{sp}) \) can be positioned in steady state by using a feedforward matrix. The matrix \( K F F \) is simply the inverse of the closed-loop LQR system transfer function matrix evaluated at \( s = 0 \).

Figure 4. – Block diagram of linear quadratic regulator.
Steady-State Kalman Filter Design

The second major computation performed by the AESOP program is the design of the steady-state Kalman filter for a linear time-invariant system described by equations (1) and (2) and shown schematically in figure 3. Data required to define the problem consist of plant matrices A, B, and H and power spectral density matrices for disturbance w and measurement noise v. These white zero-mean Gaussian noise signals are described by their covariance matrices, namely

\[ E\{w(t)w^T(t + \tau)\} = Q \delta(\tau) \]  
(20)

and

\[ E\{v(t)v^T(t + \tau)\} = (RRINV)^{-1} \delta(\tau) \]  
(21)

where matrices Q and (RRINV)\(^{-1}\) are power spectral density matrices. For the AESOP program the disturbance power spectral density matrix is entered as matrix QQ, where QQ is defined as

\[ QQ = DQDT \]  
(22)

Figure 6 is a block diagram of a linear system in "standard" form with its associated Kalman filter. The state equation defining the Kalman filter is

\[ \dot{x} = (A - KE\cdot H) \dot{x} + B u + KE z \]  
(23)

The constant gain matrix KE characterizes the filter and is obtained in AESOP by using function 809. In obtaining KE, AESOP solves the following Riccati equation:

\[ A\cdot PP + PP\cdot A^T - PP\cdot HT\cdot RRINV\cdot H\cdot PP + QQ = 0 \]  
(24)

Matrix PP is the covariance of estimation error e, where e = x - \(\dot{x}\). The Kalman gain matrix is computed by using PP as

\[ KE = PP\cdot HT\cdot RRINV \]  
(25)

As is the case with the LQR gain solution, AESOP uses the eigenvector decomposition method to solve the Riccati equation. Thus as a byproduct the eigenvalues and eigenvectors of the Kalman filter (of matrix A - KE\cdot H) are printed out.

As mentioned in the case of the LQR the Riccati solution matrix (PP in this case) should be positive-definite and symmetric. Three error checks are provided in AESOP to check on the accuracy of the estimation error covariance matrix PP: 813, to check for positive-definiteness; 814, to check symmetry; and 815, to perform a residual error check.

Normalization

One useful operation that is often performed on the matrices appearing in equations (1) to (4), which define the open-loop system, is that of normalization. Normalization alleviates possible numerical problems; allows meaningful comparison between control, state, or output variables having different units; and is generally recommended for all control and estimator design work. Generally one defines a normalization factor (usually a full-scale or operating point value) for each component of each vector. In AESOP a set of diagonal normalization matrices are defined for the system variables as follows:
The normalization (scale) factors SCX, etc., can be considered to be diagonal matrices but are stored in AESOP as single-dimensioned arrays. Function 404 is provided in AESOP to normalize all of the matrices that define the system and the control and estimation problems, namely: A, B, C, D, DOUT, CSP, QQ, and RRINV. As an example of the calculations performed, consider the normalization of the CSP matrix. We have that

\[ y_{sp} = \text{CSP} \cdot x \]  \hspace{1cm} (26)

Explicit definition of the normalization factors for \( x \) and \( y_{sp} \) are given by

\[ y_{sp} \triangleq \text{SCYSP} \cdot \bar{y}_{sp} \]  \hspace{1cm} (27)
\[ x \triangleq \text{SCX} \cdot \bar{x} \]  \hspace{1cm} (28)

where the overbar indicates the normalized vector. Thus the normalized CSP matrix (call it \( \overline{\text{CSP}} \)) can be obtained as

\[ \overline{y}_{sp} = \overline{\text{CSP}} \cdot \bar{x} \]  \hspace{1cm} (29)

where

\[ \overline{\text{CSP}} = \text{(SCYSP)}^{-1} \cdot \text{CSP} \cdot \text{SCX} \]  \hspace{1cm} (30)

The other matrices are normalized in a similar manner. Note that performance index weighting matrices QC, NN, and PCINV are not normalized in AESOP because they are considered to be "free" parameters to be manipulated by the designer. For example, "Bryson's rule," the often-used rule of thumb for choosing starting values of QC and \((\text{PCINV})^{-1}\) states that the matrices should be diagonal, where each diagonal term is simply 1 divided by the square of the maximum (or operating point) value of the corresponding state or control variable. If the system is normalized, the same result can be obtained by simply making QC and PCINV identity matrices.

If normalization is used before conducting LQR or Kalman filter designs, it may be desirable to have normalized gain matrices KC, KE, and KFF put back in dimensional form (unnormalized). Function 405 is pro-
vided in AESOP for this purpose. In addition, this function unnormalizes the error covariance matrix PP.

**Stochastic Linear Quadratic Regulator Design**

The solution of the linear quadratic regulator problem requires that the state vector \( x \) be completely measurable. In general, this will not be possible. Usually, only a vector of \( NM \) noisy measurements \( z \), which are linearly related to the state \( x \), will be present for use by the control. In line with the separation principle (ref. 12), the optimal control for this situation is constructed by feeding back an optimal state estimate (generated by a Kalman filter) through the optimal regulator gains \( KC \). This system is optimal with respect to minimizing the stochastic equivalent of the quadratic performance index given by equation (12). That equivalent index is given by

\[
J = E\left[ x^TQCx + 2x^TNNu + u^T(PCINV)^{-1}u \right]
\]

AESOP provides the means for solving this optimal control problem by using the previously mentioned two functions for computing gain matrices \( KC \) and \( KE \). The structure of the complete stochastic LQR problem is shown in figure 7. In addition to gain computations it is also of interest to compute various system responses to characterize the complete closed-loop system. Of particular interest in the case of the stochastic LQR system are the values of the covariance matrices for the system state, control, and output vectors. This is discussed in the following section.

**System Response to Noise Inputs**

The primary way to evaluate the overall performance of a system controlled by a stochastic linear quadratic regulator (such as is shown in fig. 7) is to examine the mean square or rms values of the various system variables. More generally, the quantities one wishes to compute are the covariance matrices for system vectors \( x, \hat{x}, u, z \), and \( y \). In particular, the mean square values are the diagonals of the covariance matrices. The two covariance matrices are \( XX \), the covariance of the state vector \( x \), and \( PP \), the covariance of the Kalman filter estimation error \( e \). As was mentioned previously, matrix \( PP \) is computed by AESOP function 809, which conducts the Kalman filter design. The second covariance matrix, \( XX \), is obtained by solving the following Lyapunov matrix equation (ref. 12):

\[
(A - BKC)XX + XX(A - BKC)^T + BKC\cdot PP + PP\cdot KCT - BT + QQ = 0
\]

AESOP uses an iterative method developed in references 13 and 14 to solve this equation. In comparison

Figure 7. — Stochastic LQR block diagram showing plant, Kalman filter, LQR gain matrix, and feedforward gain matrix.
with other techniques this method has been found to be especially effective for cases where system order \( N \) is large (50 to 100).

The Lyapunov solution is computed in AESOP function 817. Also, this function computes three other system covariance matrices, all simple functions of \( XX \) and \( PP \). They are

1. The covariance matrix of control \( u \),

\[
UU = KC \cdot (XX - PP) \cdot KCT
\]

(33)

2. The covariance matrix of measurement component \( z_1 \),

\[
ZZ = H \cdot XX \cdot HT
\]

(34)

3. The covariance matrix of output \( y \),

\[
YY = C \cdot PP \cdot CT + (C - DOUT \cdot KC) \cdot (XX - PP) \cdot (C - DOUT \cdot KC)^T
\]

(35)

Note that covariances \( XX, UU, ZZ, \) and \( YY \) can be computed for cases where either (1) no control is used (open-loop response) or (2) no Kalman filter is used (state feedback only). The open-loop case can be computed by simply calling function 817 without first computing either \( PP \) or \( KC \) (these two matrices will thus be all zeros). The state feedback case can be computed by first calling function 801 to obtain \( KC \) and then calling function 817.

In addition to solving Lyapunov equation (32) for the state covariance matrix, AESOP has an error check function (number 818) that gives information on the accuracy of the solution. Consider a general Lyapunov equation

\[
A \cdot X + X \cdot A^T + W = 0
\]

(36)

Let the actual computed solution to equation (36) be \( \bar{X} \). Substituting \( \bar{X} \) into equation (36) for \( X \) we obtain

\[
A \cdot \bar{X} + \bar{X} \cdot A^T + W = R
\]

(37)

where \( R \) is a residual matrix. Define an error matrix

\[
E = \bar{X} - X
\]

Subtracting equation (36) from equation (37), we obtain another Lyapunov equation

\[
A \cdot E + E \cdot A^T - R = 0
\]

(38)

AESOP function 818 uses matrix \( XX \) obtained from the Lyapunov solution of function 817 and (1) solves for the residual matrix, (2) solves a Lyapunov equation to obtain the error matrix, and (3) computes an average error value as \( \text{Trace} \left( \frac{E}{X} \right) \).

Transfer Functions and Frequency Response Calculations

It is often quite useful to examine the characteristics of a state-variable system, either open or closed loop, in the frequency domain. For instance, one may wish to analyze the pole-zero structure of the system transfer function matrix, given a state-variable system description. For this purpose, one needs to be able to compute transfer function poles, zeros, and gain given the system matrices. As another example, one may examine the transfer function matrix of an optimal feedback controller to see if any simplifying pole-zero cancellations exist such that a lower order approximation can be made. Here too it is desirable to compute poles and zeros from a state-space description. Frequency response plots (for example, Bode plots) may be desirable so that one can evaluate, using classical frequency domain criteria, the response of a control system that was designed by using LQR methods. Or, given a state-space, open-loop system description, one may wish to compute a matrix of system transfer functions or a matrix of frequency responses that can subsequently be used by a frequency domain control design program. For these reasons, it was decided to include in AESOP the capability to compute transfer functions and frequency responses for various systems and subsystems defined in state-space terms.

Consider the generalized nth-order system described by state equations

\[
\dot{x} = A \cdot x + B \cdot u
\]

(39)

\[
y = C \cdot x + D \cdot u
\]

(40)

and having “nc” inputs \( u \) and “no” outputs \( y \). These equations could represent any linear system (open-loop plant, Kalman filter, closed-loop regulator, etc.) by appropriate choice of vectors \( \bar{x}, \bar{y}, \) and \( \bar{u} \) and matrices \( \bar{A}, \bar{B}, \bar{C}, \) and \( \bar{D} \). The transfer function matrix \( \bar{G}(s) \) relating output vector \( \bar{y} \) to input vector \( \bar{u} \) can be written as

\[
\bar{y}(s) = [\bar{C} \cdot (sI - \bar{A})^{-1} \bar{B} + \bar{D}] \cdot (sI - \bar{A})^{-1} \bar{C} \cdot \bar{x} + \bar{D} \cdot \bar{u}(s)
\]

(41)

AESOP allows the user to obtain solutions to equation (41) for a variety of system configurations. It computes a transfer function \( \bar{G}_{ij}(s) \) relating a component \( \bar{u}_i(s) \) to a component \( \bar{y}_j(s) \) in two forms.

The first transfer function form computed is where each \( \bar{G}_{ij}(s) \) is a ratio of polynomials. In this case the general expression for \( \bar{G}_{ij}(s) \) is
Note that since a feedforward matrix $\mathbf{D}$ is assumed, $\bar{G}_{ij}(s)$ may have equal-order $(n)$ numerator and denominator. AESOP uses the technique of reference 15, modified slightly to include a $\mathbf{D}$ matrix, to compute coefficients $a_k$ and $b_k$. Constants $b_k$ are the coefficients of the characteristic equation of $\bar{A}$ and are thus common to all $\bar{G}_{ij}(s)$.

The second transfer function form computed by AESOP is the so-called factored form, where the general expression is given as

$$\bar{G}_{ij}(s) = \frac{K_{ij} \prod_{k=1}^{m} (s + z_k)}{\prod_{k=1}^{n} (s + p_k)}$$

(43)

where $z_k$ and $p_k$ are the transfer function zeros and poles, respectively. In addition to poles and zeros, AESOP computes gains $K_{ij}$ and the number of numerator zeros $m$ ($m \leq n$). The poles $p_k$ are obtained in AESOP simply as the eigenvalues of $\bar{A}$. The method for obtaining $m$ and $z_k$ depends on the value of $\mathbf{D}_{ij}$.

For cases where $\mathbf{D}_{ij}$ is equal to zero the values of $m$ and $z_k$ are obtained by using a method developed by Davison (ref. 16). The method is essentially based on a concept from root locus theory. That is, if a proportional loop is closed between output $y_i$ and input $u_j$ and the loop gain is allowed to increase to infinity, $m$ of the root loci (poles of the closed-loop transfer function) will go to the $m$ open-loop transfer function zeros, and the remainder $(n-m)$ will go off to infinity. Davison’s method successively computes the eigenvalues of such a system while increasing the loop gain. It stops when the $n-m$ “extraneous” eigenvalues all exceed a “large” value.

For cases where the value of $\mathbf{D}_{ij}$ is nonzero, the number of zeros and poles of the transfer function $\bar{G}_{ij}(s)$ are both equal to $n$. In this case the zeroes are simply the eigenvalues of the matrix

$$\bar{A}^* = \left[ A - \bar{b}_j^T \bar{e}^T \bar{u}_i \right]$$

(44)

where

$$\bar{B} \triangleq [\bar{b}_1 \quad \bar{b}_2 \quad \cdots \quad \bar{b}_m]$$

This fact can be seen by applying a feedback control

$$\bar{u}_j = -k^* y_i$$

(45)

to the system of equations (39) and (40) and allowing gain $k^*$ to go to infinity. In the limit the eigenvalues of $\bar{A}^*$ become the zeros of the transfer function $\bar{G}_{ij}(s)$.

The remaining transfer function term in equation (43) to be computed is gain $K_{ij}$. AESOP uses the technique described by Brockett (ref. 17) to compute this gain as

$$K_{ij} = \begin{cases} \mathbf{d}_{ij}, \quad \bar{d}_{ij} \neq 0 \\ \bar{e}^T A^{n-m-1} \bar{b}_j, \quad \bar{d}_{ij} = 0 \end{cases}$$

(46)

Computations for two types of transfer functions, polynomial form and factored form, are performed in AESOP series 500 and 700, respectively. System configurations for which calculations are done are (1) an open-loop system, (2) a system with state feedback, (3) a system with a Kalman filter in the feedback loop, and (4) an optimal controller. Polynomial-form transfer function coefficients are computed for the purpose of plotting frequency responses. The AESOP program uses the same subroutines to compute frequency responses and transfer functions for each configuration, first forming the appropriate $\bar{A}$, $\bar{B}$, $\bar{C}$, and $\bar{D}$ matrices as functions of $A$, $B$, $K_E$, $K_C$, etc. Factored-form transfer function information (poles, zeros, and gain) is mainly of interest in one of two instances: (1) when investigating the structure of the open-loop system, and (2) when examining the pole-zero structure of an optimal controller. Therefore data are obtained by AESOP only for these two configurations. Frequency responses, however, are obtained for all four configurations mentioned previously. Table VII in the next section outlines in detail which calculations AESOP does perform.

Figure 8 shows the open-loop configuration for which transfer functions and frequency responses can be calculated. Corresponding to the form of equation (41), the four transfer functions that AESOP computes here are
Control to output:

\[ y(s) = [(sI - A)^{-1} B + DOUT]u(s) \]  

(47)

Disturbance to output:

\[ y(s) = C(sI - A)^{-1} D w(s) \]  

(48)

Control to measurement:

\[ z(s) = H(sI - A)^{-1} B u(s) \]  

(49)

Disturbance to measurement:

\[ z(s) = H(sI - A)^{-1} D w(s) \]  

(50)

Figure 9 shows the second configuration—a system with state-variable feedback. Here AESOP computes the following frequency responses by using disturbance \( w \) as the input:

Disturbance to output:

\[ y(s) = (C - DOUT \cdot KC) [(sI - (A - B \cdot KC))^{-1} D w(s) \]  

(51)

Disturbance to control:

\[ u(s) = -KC(sI - (A - B \cdot KC))^{-1} D w(s) \]  

(52)

Disturbance \( w \) is not explicitly used in the design of the (noise free) linear quadratic regulator (e.g., eq. (14)). However, it is instructive to examine its disturbance response in the frequency domain.

Figure 10 depicts the configuration where disturbance \( w \) is explicitly considered in the control system design, namely the stochastic linear quadratic regulator. The overall system order in this configuration is \( 2N \), there being \( N \) states \( x \) and \( N \) state estimates \( \hat{x} \). In obtaining frequency responses AESOP first forms partitioned system matrices before calling the subroutine that computes the responses. The following responses are computed:

Disturbance to output:
\begin{align*}
y(s) &= CTOT(sI - ATOT)^{-1} DTOT w(s) \quad (53) \\
Disturbance to measurement: \\
z(s) &= HTOT(sI - ATOT)^{-1} DTOT w(s) \quad (54) \\
Disturbance to control: \\
u(s) &= KCTOT(sI - ATOT)^{-1} DTOT w(s) \quad (55) \\
\end{align*}

where

\[
ATOT = \begin{bmatrix}
A & -B \cdot KC \\
KE \cdot H & A - B \cdot KC - KE \cdot H
\end{bmatrix}, \quad 2N \times 2N
\]

\[
DTOT = \begin{bmatrix}
D \\
0
\end{bmatrix}, \quad 2N \times ND
\]

\[
CTOT = [C \mid -DOUT \cdot KC], \quad NO \times 2N
\]

\[
HTOT = [H \mid 0], \quad NM \times 2N
\]

\[
KCTOT = [0 \mid -KC], \quad NC \times 2N
\]

Note that measurement noise, although a consideration in the Kalman filter design, is not considered here as an input.

The last configuration for which frequency responses as well as transfer function zeros and gain is computed is the optimal controller, shown in figure 11. This is simply the feedback portion of the control loop of figure 10, the stochastic regulator. The desired response is for measurement \( z \) as an input and control \( u \) as an output. This transfer function can be expressed as

\[
u(s) = -KC[sI - (A - B \cdot KC - KE \cdot H)]^{-1} KE z(s) \quad (56)
\]

It has been noted in the literature that this optimal controller transfer function can have some interesting properties; for example it can sometimes be unstable (ref. 18) or can have right-half-plane zeros (ref. 1).

**Transient Response Calculation**

The AESOP program computes and plots transient responses for several important system configurations. Consider again the general time-invariant system, which is described in state-variable form as

\[
\dot{x}(t) = A \dot{x}(t) + B \ddot{u}(t) 
\]  

\[
\ddot{y}(t) = C \dot{x}(t) + D \ddot{u}(t) 
\]  

Figure 10. – Block diagram of stochastic LQR configuration used for obtaining frequency responses. Input is \( w \); outputs are \( y, z, \) and \( u \).
In AESOP it was desired to solve these general equations (57) and (58) for two cases: (1) an initial condition \( \mathbf{x}(0) \), and (2) a step change in input \( \mathbf{u}(t) \). The normal approach, and that taken in AESOP, is to discretize equation (57) so as to obtain the exact solution at time points \( t_0, t_1, \ldots, t_k \) that are spaced \( DT \) seconds apart. The resulting discrete difference equations are thus

\[
\begin{align*}
\mathbf{x}_{n+1} &= \mathbf{A}\mathbf{x}_n + \mathbf{B}\mathbf{u}_n \\
\mathbf{y}_n &= \mathbf{C}\mathbf{x}_n + \mathbf{D}\mathbf{u}_n
\end{align*}
\]

where

\[
\begin{align*}
\mathbf{x}_{n+1} &= \mathbf{x}(t_{n+1}) \\
\mathbf{x}_0 &= \mathbf{x}(t_0) = \mathbf{x}(0)
\end{align*}
\]

and matrices \( \mathbf{A} \) and \( \mathbf{B} \) are given by the series representations

\[
\Phi(DT) = e^{\mathbf{A}DT} = I + \mathbf{A}DT + \frac{\mathbf{A}^2DT^2}{2!} + \ldots
\]

and

\[
\Gamma = \left[\int_0^{DT} \Phi(s)ds\right] \mathbf{B}
\]

\[
= \left(I\cdot DT + \frac{\mathbf{A}DT^2}{2!} + \frac{\mathbf{A}^2DT^3}{3!} + \ldots\right) \mathbf{B}
\]

Matrix \( \Gamma \) is computed by assuming that input \( \mathbf{u}(t) \) is constant over \( t_n \leq t < t_{n+1} \). The series in equations (63) and (64) are carried out until the desired accuracy in \( \Phi \) and \( \Gamma \) is achieved.

The general procedure just described is used to compute transients for the following three situations:

1. **Open-loop system** – The equations for this system, shown in figure 3, are

\[
\begin{align*}
\dot{x} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\
y &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}
\end{align*}
\]

AESOP calculates and plots \( \mathbf{x} \) and \( \mathbf{y} \) for ITRMX time points for either

- (a) A step input applied to component \( i \) of input \( \mathbf{u}(t) \)
- (b) An initial condition \( \mathbf{x}(0) \) on the \( i \)th component of the state vector \( \mathbf{x}(t) \).

These computations are performed by AESOP functions 601 and 602, respectively.

2. **Closed-loop linear quadratic regulator** – Figure 4 shows the configuration of this system. The describing equations are

\[
\begin{align*}
\dot{x} &= (\mathbf{A} - \mathbf{B}\cdot\mathbf{K}\mathbf{C})\mathbf{x} \\
y &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{O}\mathbf{U}\mathbf{u}
\end{align*}
\]

Matrices \( \mathbf{A}, \mathbf{B}, \mathbf{C}, \) and \( \mathbf{D} \) are appropriately formed by AESOP and variables \( \mathbf{x}, \mathbf{y}, \) and \( \mathbf{u} = -\mathbf{K}\mathbf{C}\mathbf{x} \) are calculated and plotted. This is done for an initial condition \( \mathbf{x}(0) \) on a selected component of \( \mathbf{x}(0) \). Function 603 accomplishes these calculations. The responses for \( \mathbf{x}, \mathbf{y}, \) and \( \mathbf{u} \) can be compared by the user with the open-loop responses in order to help evaluate a particular LQR design.

3. **Nonzero-set-point linear quadratic regulator** – This system, shown in figure 5, is described by one state and three "output" equations:

\[
\begin{align*}
\dot{x} &= (\mathbf{A} - \mathbf{B}\cdot\mathbf{K}\mathbf{C})\mathbf{x} + \mathbf{B}\cdot\mathbf{K}\mathbf{F}\mathbf{F}\mathbf{Y}_{spd} \\
y &= (\mathbf{C} - \mathbf{D}\mathbf{O}\mathbf{U}\cdot\mathbf{K}\mathbf{C})\mathbf{x} + \mathbf{D}\mathbf{O}\mathbf{U}\mathbf{F}\mathbf{F}\mathbf{Y}_{spd} \\
y_{sp} &= \mathbf{CSP}\mathbf{x} \\
\mathbf{u} &= -\mathbf{K}\mathbf{C}\mathbf{x} + \mathbf{K}\mathbf{F}\mathbf{F}\mathbf{Y}_{spd}
\end{align*}
\]
Here again, AESOP forms the appropriate matrices before computing the responses. The response of interest here is to a step change in a selected component of the set-point vector $y_{spd}$. Of interest to the user in this configuration are such things as rise time and overshoot of the $k^{th}$ component of $y_p$ in response to a step change in the $k^{th}$ component of $y_{spd}$; interaction – the response of the $k^{th}$ component of $y_{sp}$ to a change in the $i^{th}$ component of $y_{spd}$; and the magnitudes of control variable excursions. The computation and response plotting for the non-zero-set-point regulator are performed by AESOP function 604.

AESOP Program Operation

This section provides an overview of how to operate the AESOP program. In particular, it discusses how the user interfaces with the program within the IBM 370 TSS PCS (Program Control System) environment, how to enter data, how to enter a string of control numbers that dictate the series of AESOP functions to be performed, and in general, how the user interfaces with the program at the “terminal,” not Fortran, level. (It is assumed that the reader is familiar with PCS commands and the operation of the 370 TSS system.)

Program Structure

The general structure and operation of AESOP was described in the Introduction and shown schematically in figure 1. A more detailed diagram of the program showing the main AESOP program and nine main subroutines is given in figure 12. Each main subroutine contains a number of related AESOP functions. The numbers above the main subroutine boxes in figure 12 indicate the function series that each main subroutine contains (800 contains functions 801 to 899, etc.). The details as to what each function does will be discussed later in this section.

All input and output performed by the main AESOP program relate to program control and required interfacing with the user. The main program (1) initializes default or reference values, (2) accepts a string of function numbers that the user wishes to have performed, (3) performs checks to see whether the user has requested functions to be done in a reasonable order, and (4) calls the appropriate main subroutines in which the desired functions reside.

To run the AESOP program, the user first calls the PROCODE (for PROcedure DEFinition) AERSRUN, defined in appendix D. Through use of TSS/370 datadef (DDEF) statements, AERSRUN

(1) Defines (“datadefs”) the library dataset (file) that contains the compiled AESOP program and all subroutines and then loads the program
(2) Defines the link to the graphics package that contains graphics subroutines called by AESOP
(3) Links (“datadefs”) Fortran unit numbers with specific dataset (file) names so that these files can be written to or read from during the subsequent AESOP run

AERSRUN has a single parameter that is used to identify all output datasets to be generated during the AESOP run. After calling AERSRUN, the user types “AESOP” to run the program.

Operating Procedure

The key element in the use of AESOP is the single-dimensioned function number array (Fortran symbol IFN). The user loads, in sequence, this vector with the numbers of the AESOP functions that are to be performed. The use of the IFN vector and the general operation of AESOP will be explained with the aid of the flow chart of figure 13. The user many also refer to the terminal listing included in appendix C for test case I for an actual example of how AESOP is run. First, the user types “AESOP” followed by its parameter. The
computer responds with information that the libraries have been set up and all datasets have been datadeffed. The user then types "AESOP" to call the main program. (Note that the heavy blocks in figure 13 indicate either user input commands or messages printed out at the user's terminal.) The program then responds by prompting the user with the message:

**DO YOU WISH TO MAKE PLOTS, Y OR N?**

If the response is "Y," the user is then asked to

**ENTER THE PLOT NAME – 8 ALPHANUMERIC CHARACTERS**

This name will be used to name the plot dataset in which any plots generated will be stored. Next, the user is asked,

**DO YOU WISH TO MAKE ON-LINE PLOTS, Y or N?**

If the reply is "Y," the program sets an internal flag that will cause the program to PAUSE after each on-line plot is displayed to allow the user to view it. Off-line plotting requires no such action to be taken. Finally, the user is asked for two pieces of information that will appear on all plots for identification purposes:

**ENTER TODAY’S DATE (LESS THAN OR EQUAL TO 20 CHARACTERS)**
Figure 13. – Concluded.

and

ENTER THE PARAMETER YOU USED FOR AESRUN

In view of the fact that graphics subroutines are rather specific to the user's computer system, they are not provided as part of AESOP. Thus the preceding messages and prompts would be tailored by each user to reflect the specific graphics package being used.

After the graphics-oriented prompts are handled, the program displays the message

EXTENDED TERMINAL OUTPUT?
(This question and all subsequent ones are to be answered “Y” (yes) or “N” (no). All output produced by the program is stored on the output dataset, which the PROCDEF AESRUN datadefed to unit 06. Two subsets of this output are available for display on-line at the user’s terminal. The EXTENDED TERMINAL OUTPUT option (appendix D) allows the user to view a more extensive subset of data if desired.

Next, the user is prompted with the message

**READ IN N1 FROM STORAGE?**

The N1 referred to here is the name of a Fortran NAMELIST that contains the vector IFN. For problems that are solved over and over again, using the same AESOP functions but different data, it is convenient to store the NAMELIST N1 in a dataset to avoid having to type it each time at the terminal. If this is the case, the user would respond with “Y” and the program would print out

**DDEF 21 TO THE N1 DATASET**

At this point the user would then type the required datadef statement, for example,

**DDEF FT21F001,VS,MYN1DS**

where dataset MYN1DS would contain typical NAMELIST data such as

&&1 IFN = 201,701,999 &&END

However, to enter a new set of numbers into the IFN vector, the user would enter “N” and the program would respond with

**ENTER NAMELIST DATA AS '&&1 IFN = , , , &&END**

At this point the user would type a NAMELIST N1, such as was given in the preceding example.

The IFN string can be terminated in two ways. If the user ends it with a number greater than or equal to 999 (as was done in the preceding example), the AESOP program will execute all of the requested functions and then terminate the run. (The number 999 is a request for termination.) If the user ends the string with the number of an executable function, the program will allow more function numbers to be entered when the present string of functions has been executed. In either case, as soon as the NAMELIST has been read by the program, the program displays all of the elements of the IFN vector at the terminal and starts to execute the requested series of functions.

**Figure 13 shows that the main program indexes integer MZ each time just before it begins to execute a function. This integer denotes the element of the IFN vector that contains the number of the function about to be executed. The program then performs four successive checks on IFN (MZ), the MZth element of IFN. These checks are**

1. **Is the IFN(MZ) ≥ 999?** If so, terminate the program; if not, continue checking.
2. **Is IFN(MZ) ≤ 100 but not equal to zero?** If so, you have requested a nonexistent function and will be allowed to change the requested number.
3. **Is IFN(MZ) = 0?** If so, you have reached the end of the function number string requested and will now be able to enter additional function numbers if desired.
4. **Is IFN(MZ) the number of an existing executable function in series 100 to 900?** If so, the function will be executed; if not, you will be allowed (as in check (2)) to change the requested number.

As an example, first, assume that the present IFN(MZ) encountered is a nonexistent function number (i.e., checks (2) or (4) above were failed). The program displays the messages

**THESE ARE THE FUNCTIONS YOU HAVE RUN THUS FAR

nnn
THE PRESENT FUNCTION IS NOT A LEGITIMATE ONE, YOU WILL NOW HAVE A CHANCE TO REPLACE IT WITH ONE GOOD ONE OR TO TERMINATE OR TO ENTER A LIST OF FUNCTIONS. TYPE Y TO ENTER ONE REPLACEMENT FUNCTION; TYPE N TO ENTER A LIST OF FUNCTIONS.**

If the user types “Y,” the program responds with

**TYPE IN AN I3 NUMBER TO REPLACE THE CURRENT FUNCTION; IF THE NUMBER IS 999, THE PROGRAM WILL TERMINATE**

The user would then enter a new function number and the program would repeat the four checks. If the new number is that of an executable function, the program proceeds to execute the function.

If the user types “N” in response to the previous prompt, the program will display the message

**TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS. (I3), ONE PER LINE; TO TERMINATE ENTER 999 (LAST ENTRY MUST BE A RETURN).**

The user may then enter a series of function numbers, hitting “RETURN” after each three-digit number and
hitting “RETURN” twice after entering the last number in the series. The program also responds with the preceding prompt whenever it encounters IFN(MZ) = 0, which is the indication that the end of the previously requested function string has been reached.

The previous paragraphs have dealt with situations where the program detects nonexistent function numbers in the input string. In the case where the function number is correct, the main AESOP program then calls the appropriate main AESOP subroutine in which the function resides. At that point a prerequisite check is begun. As a user aid the AESOP program contains a table of prerequisite functions (appendix F). The program checks the prerequisite table immediately before executing each requested function to insure that the specified prerequisite functions have already been performed. For example, suppose the user enters function number 401, which requests that open-loop eigenvalues (of the A matrix) be computed, without preceding this number with either number 201 or 202, the functions that form or read in data that define the system. Just before calling function 401 the program checks the prerequisite table and detects that the proper prerequisites (either function 201 or 202) have not been performed prior to this function. The program then displays the message

YOU HAVE NOT EXECUTED THE FOLLOWING PREREQUISITE FUNCTION(S) FOR FUNCTION 401

and then prints out pertinent function numbers. It then displays the message

IF YOU THINK YOU KNOW WHAT YOU ARE DOING AND WISH TO IGNORE THE PREREQS AND CONTINUE ON TO DO THIS FUNCTION, TYPE Y AND RETURN; OTHERWISE, JUST RETURN

The prerequisites in the table for each AESOP function were selected so as to catch most, if not all, errors a user might make in selecting a series of interdependent functions that could lead to calculations that would produce major errors. These would be errors such as zero divides during inversion of a singular matrix, etc. It was felt that protecting against these nuisance errors would make it less likely that a user would have to restart the program in midcourse because of a major nonrecoverable error. However, it is still possible to select a series of functions that produce nonsensical results even though prerequisite checks show no error.

To terminate the program, the user types “999” when the program asks for more function numbers. The program will then display the message

STORE THE N1 FOR THIS RUN?

This allows the user to store the IFN vector, which was just used in the present run, on a dataset for possible future use. If this is desired, the user replies with “Y” and will receive the message

DDEF NAME OF N1 DATASET TO 22

The user would execute the required datadef, using whatever dataset name was desired, and then type “GO” to cause the program to terminate. If the IFN array is not to be stored, a previous response of “N” would terminate the run immediately.

Data Input and Output

A key aspect of the AESOP program is the handling of data input and output. The primary input and output device is the user’s terminal. Most of the data sent or received through the terminal pertain to program control, as was demonstrated in the examples in the previous section. That section also showed two examples of how the program accesses data that are stored in a dataset (IFN, read from unit 21) and how the program writes data out to a dataset (writing IFN onto unit 22). In the program, however, 14 unit numbers are used for input and output of data (in addition to unit 02, reserved for the terminal, and unit 06, reserved for the high-speed printer output). All units are listed in table I together with the names, contents, and format of their associated datasets. Dataset names are created by the PROCDEF AERSRUN, which appends the characters in parameter $1 to an identifying prefix. For example, referring to table I, if the parameter $1 were entered as XYZ, the PROCDEF would datadef unit 08 to dataset CGXYZ. This dataset is the one in which the control gain matrix is stored. Other datasets that are identified by using the parameter $1 are those datadefed to unit 09 (containing the Kalman filter gain matrix), units 10 to 14 (containing frequency response magnitudes and phase angles), unit 15 (estimation error covariance matrix), unit 16 (control Riccati equation solution matrix), and unit: 17 (feedforward gain matrix). The remaining units, 33 and 34, are for datasets that store the data that define the design problem to be solved by AESOP, namely the open-loop plant data (unit 33) and normalizing factors (unit 34). The user must datadef these units and specify the names of their associated datasets.

Much of the input data for AESOP is in NAMELIST form. Table II lists all NAMELIST’s used by the program and the names of the Fortran variables contained in each. A key NAMELIST is MATDAT, which contains all matrix coefficients and dimensions needed to define the problem to be solved. NAMELIST’s CONPAR and ESTPAR, which are useful when modifying basic problem data, contain subsets of the variables contained in MATDAT. CONPAR is used for
TABLE I - DATASETS AND UNITS USED FOR PROGRAM INPUT AND OUTPUT

<table>
<thead>
<tr>
<th>Unit</th>
<th>Dataset name</th>
<th>Contents of dataset</th>
<th>Format</th>
<th>Function where read/written</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>OUT$^b$</td>
<td>All terminal input and output</td>
<td>Various</td>
<td>Many</td>
<td>Terminal input and output</td>
</tr>
<tr>
<td>06</td>
<td>CG$^b$</td>
<td>High-speed printer output</td>
<td>Various</td>
<td>Many</td>
<td>Terminal input and output</td>
</tr>
<tr>
<td>08</td>
<td>EG$^b$</td>
<td>KC, control gain matrix</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>09</td>
<td>EG$^b$</td>
<td>KE, Kalman filter gain matrix</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>10</td>
<td>PFRUZ$^b$</td>
<td>Frequency response, $z(\text{IMEAS})/u(\text{JINC})$</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>11</td>
<td>PFRUZ$^b$</td>
<td>Frequency response, $y(\text{IOUT})/u(\text{JINC})$</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>12</td>
<td>PFRUZ$^b$</td>
<td>Frequency response, $u(\text{JINC})/z(\text{IMEAS})$</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>13</td>
<td>PFRUZ$^b$</td>
<td>Frequency response, $z(\text{IMEAS})/w(\text{JIND})$</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>14</td>
<td>PFRUZ$^b$</td>
<td>Frequency response, $y(\text{IOUT})/w(\text{JIND})$</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>15</td>
<td>PFRUZ$^b$</td>
<td>PP, estimation error covariance matrix</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>16</td>
<td>SS$^b$</td>
<td>SS, control Riccati solution matrix</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>17</td>
<td>FF$^b$</td>
<td>KF, feedforward gain matrix</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>See table VI for details on frequency responses</td>
</tr>
<tr>
<td>21</td>
<td>(b) IFN vector, contained in NAMELIST N1</td>
<td>IFN vector, contained in NAMELIST N1</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>For input of IFN</td>
</tr>
<tr>
<td>22</td>
<td>(b) IFN vector, contained in NAMELIST N1</td>
<td>IFN vector, contained in NAMELIST N1</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>For output of IFN</td>
</tr>
<tr>
<td>33</td>
<td>(b) Open-loop plant data, NAMELIST's MATDAT and REFS</td>
<td>Open-loop plant data, NAMELIST's MATDAT and REFS</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>Usual source of problem data</td>
</tr>
<tr>
<td>34</td>
<td>(b) Normalizing factors; NAMELIST NRMS</td>
<td>Normalizing factors; NAMELIST NRMS</td>
<td>Unformatted</td>
<td>205/802, 206/810</td>
<td>Usual source of problem data</td>
</tr>
</tbody>
</table>

$^b$ is the parameter for PROCDEF AESRUN; it serves as an arbitrary identifying tag (fewer than four characters) for the naming of datasets. User specified.

Making changes in performance index weights, and ESTPAR is used for varying noise matrix elements.

NAMELIST NRMS contains normalizing factors whose use is described by equations (26) to (30). The remaining NAMELIST, REFS, is used for setting various parameter values that specify such things as time steps, frequency point spacing and numbers, perturbation amplitudes, and input and output selection indices. Table III specifies each parameter in REFS and its default value and indicates in which AESOP function each parameter is used. For a more detailed explanation of each parameter, refer to the appropriate function description given in the next section.

Description of AESOP Functions

Each AESOP function will now be described in sufficient detail so that this section can serve as a primary reference when using the program. All functions, as indicated previously, are grouped into nine series (series 100 to 900). When possible, a general description will be provided for each series, with an accompanying table included to show similarities and differences among functions within the series. Also, whenever possible, the reader will be referred to appendix C, test case I, for an example of the use of each function.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Dimension</th>
<th>Definition</th>
<th>Default value</th>
<th>Functions where used</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>Scalar</td>
<td>Time step</td>
<td>0.05 sec</td>
<td>Series 600 (transient responses); functions 601 to 604</td>
</tr>
<tr>
<td>ITRMX</td>
<td>Scalar</td>
<td>Desired maximum number of time steps</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>AMPSP</td>
<td>NMAX</td>
<td>Open-loop system step input amplitude vector</td>
<td>All elements are set to 1.0</td>
<td></td>
</tr>
<tr>
<td>AMPICX</td>
<td>NMAX</td>
<td>Closed-loop system step input amplitude vector</td>
<td>All elements are set to 1.0</td>
<td></td>
</tr>
<tr>
<td>MSROLYa</td>
<td>NMAXb, NOMAXc</td>
<td>Initial-condition amplitude vector</td>
<td>All elements are set to 1.0</td>
<td></td>
</tr>
<tr>
<td>MSROLXa</td>
<td>NMAXb, NOMAXc</td>
<td>Input/output selection matrix</td>
<td>All elements are set to 1</td>
<td></td>
</tr>
<tr>
<td>MICOLYa</td>
<td>NMAXb, NOMAXc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MICOLXa</td>
<td>NMAXb, NOMAXc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MICCLYa</td>
<td>NMAXb, NOMAXc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MICCLXa</td>
<td>NMAXb, NOMAXc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSPYSpa</td>
<td>NMAXb, NOMAXc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSPYa</td>
<td>NMAXb, NOMAXc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSPUa</td>
<td>NMAXb, NOMAXc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FI</td>
<td>Scalar</td>
<td>Starting frequency for frequency responses</td>
<td>0.01 Hz</td>
<td>Series 500, frequency responses</td>
</tr>
<tr>
<td>DELF</td>
<td>Scalar</td>
<td>Frequency-point spacing</td>
<td>0.02 Hz</td>
<td></td>
</tr>
<tr>
<td>NCURV</td>
<td></td>
<td>= 2, cross plot open- and closed-loop responses; = 1, closed loop only</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LINLOG</td>
<td></td>
<td>= 1, linear Bode plots; = 2, log Bode plots; = 3, both linear and log plots</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td></td>
<td>Desired number of frequency response points</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>ISPACE</td>
<td></td>
<td>Every ISPACE frequency response point is listed</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TSFTR</td>
<td></td>
<td>Time scale factor, used for increased precision</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>IOUT</td>
<td>Scalar</td>
<td>Index for selecting component of ( y ) vector</td>
<td>1</td>
<td>Series 500 (frequency responses) and series 700 (transfer functions)</td>
</tr>
<tr>
<td>IMEAS</td>
<td></td>
<td>Index for selecting component of ( z ) vector</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JINC</td>
<td></td>
<td>Index for selecting component of ( u ) vector</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JIND</td>
<td></td>
<td>Index for selecting component of ( w ) vector</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ZERMAX</td>
<td>Scalar</td>
<td>10 times value of largest expected transfer function zero</td>
<td>100 rad/sec</td>
<td></td>
</tr>
</tbody>
</table>

\(^a\) See table VII for detailed description.  
\(^b\) Row size.  
\(^c\) Column size.
When running the AESOP program, it has been found helpful to have available a brief list of all possible functions. This list is provided in table IV. Once the user has read this report and is generally familiar with what the program can do, table IV can be used as a ready reference guide while running AESOP. More specific details on the various functions are given here.

**Series 100 – Program Control**

Series 100 contains only two functions: one for changing reference parameter values, and the other to allow the user to change datadefs in the middle of a run.

101 – **Change reference values by using NAMELIST REFS.** – Table III describes various parameters in NAMELIST REFS that are used in the AESOP program to determine time and frequency steps, input and output indices, etc. The table also gives the default values of these parameter values that are initialized in the main AESOP program. Function 101 allows the user to change any or all of these parameters. It prompts the user with the message

```
ENTER CHANGES TO NAMELIST REFS (TSFTR, DT, FI, DELF, ZERMAX, AMPSP, AMPSR, AMPICX, IF, ISPACE, IOUT, IMEAS, JINC, JIND, ITRMX, NCURV, LINLOG, MSPY, MSPYSP, MSPU, MSROLY, MSROLX, MICCLY, MICCLX, MICCLU, MICOLY, MICOLX)
```

after which the user can enter the desired parameter changes. An example of the use of function 101 appears in appendix C, test case I, page 65.

102 – **PAUSE to allow user to change datadefs.** – If the user requests this function, the program will display the message

```
YOU MAY NOW CHANGE YOUR DDEFS IF YOU WISH. DON'T FORGET TO CLOSE AND RELEASE THE OLD ONES FIRST
```

The program then effects a Fortran PAUSE and the keyboard unlocks to allow the user to make the appropriate changes. This function is useful, for example, when the user wishes to read in NAMELIST’s MATDAT and REFS from dataset BBB, where previously similar data had been read from dataset AAA. At the requested PAUSE, the user can then enter

```
CLOSE AAA
RELEASE FT33F001
and then
DDEF FT33F001,VS,BBB
```

and then type “GO” to continue execution. Subsequently, the user can request function 202, which will read in the desired data from dataset BBB.

**Series 200 – Data Input and Revision**

This series of nine functions is used for inputting basic problem-defining data from datasets or for inputting changes to that data. The changes are typed in from the user’s terminal.

201 – **Forming matrices for test case I.** – Function 201 is of use mainly when executing test case I, which is presented in appendix C. Function 201 simply forms all matrices and defines all dimensions, all of which are included in NAMELIST MATDAT. The specific matrices that this function forms for the test case are shown in table V. The dimensions for this problem are N = 3, NC = 2, NM = 1, NO = 2, and ND = 2.

202 – **Primary function for problem-defining data input.** – Function 202 is the one usually used for reading in (from a VS dataset) the data that define the user’s problem. The data must reside in the dataset in the following NAMELIST form:

```
&MATDAT (data for variables in NAMELIST MATDAT – see table II)
&END

&REFS (data for variables in NAMELIST REFS – see table II)
&END
```

Note that data for the two NAMELIST’s must be in the order shown. It is not necessary to have any of the REFS parameters entered if the user wishes to have AESOP use the default values shown in table III. However, the entry in the aforementioned dataset must then appear as

```
&REFS &END
```

203, 204, and 210 – **Functions used for revising data that define the user’s problem.** – Functions 203, 204, and 210 are useful when conducting design iterations, for filter design, regulator design, or plant sensitivity studies. Each function allows various parameter changes to be made via NAMELIST’s from the user’s terminal. (For NAMELIST definitions, see table II.)

Function 203 is used to revise elements in the control design performance index weighting matrices QC, NN, and PCINV. NAMELIST CONPAR is used for this purpose. Function 204 is used to revise elements in noise power spectral density matrices QQ and RRINV through NAMELIST ESTPAR. These noise parameters, of course, strongly influence the characteristics of the associated Kalman filter. Function 210 is useful when changes are to be made in any of the variables in
## Table IV. Summary of AESOP Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Program control</td>
</tr>
<tr>
<td>102</td>
<td>Change reference values</td>
</tr>
<tr>
<td>103</td>
<td>Change program pause</td>
</tr>
<tr>
<td>201</td>
<td>Inputting or revising matrices</td>
</tr>
<tr>
<td>202</td>
<td>Basic matrix data input</td>
</tr>
<tr>
<td>203</td>
<td>Matrix input:</td>
</tr>
<tr>
<td>204</td>
<td>Matrices for test case</td>
</tr>
<tr>
<td>205</td>
<td>Revising matrices:</td>
</tr>
<tr>
<td>206</td>
<td>QC, NN, and PCINV (for LQR)</td>
</tr>
<tr>
<td>207</td>
<td>QQ and RRINV (for filter)</td>
</tr>
<tr>
<td>208</td>
<td>A, B, C, DOUT, H, CSP, QC, NN, PCINV, QQ, and RRINV</td>
</tr>
<tr>
<td>209</td>
<td>Reading stored matrices:</td>
</tr>
<tr>
<td>210</td>
<td>KC (from LQR)</td>
</tr>
<tr>
<td>211</td>
<td>KE (from filter)</td>
</tr>
<tr>
<td>212</td>
<td>SS (from LQR)</td>
</tr>
<tr>
<td>213</td>
<td>PP (from filter)</td>
</tr>
<tr>
<td>214</td>
<td>KFF</td>
</tr>
<tr>
<td>215</td>
<td>Matrix formation</td>
</tr>
<tr>
<td>301</td>
<td>A - B*KC</td>
</tr>
<tr>
<td>302</td>
<td>A - B<em>KC - KE</em>H</td>
</tr>
<tr>
<td>303</td>
<td>&quot;Total&quot; system matrices</td>
</tr>
</tbody>
</table>

### Open-loop system analysis

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>Compute open-loop system</td>
</tr>
<tr>
<td>402</td>
<td>Plot open-loop system</td>
</tr>
<tr>
<td>403</td>
<td>Store open-loop system</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>404</td>
<td>Eigenvalues and mode shapes</td>
</tr>
<tr>
<td>405</td>
<td>Eigenvectors and observability, and residues</td>
</tr>
<tr>
<td>406</td>
<td>Normalization</td>
</tr>
<tr>
<td>407</td>
<td>Unnormalization</td>
</tr>
</tbody>
</table>

### Frequency responses

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>501</td>
<td>Open-loop step</td>
</tr>
<tr>
<td>502</td>
<td>Open-loop initial condition</td>
</tr>
<tr>
<td>503</td>
<td>Closed-loop LQR initial condition</td>
</tr>
<tr>
<td>504</td>
<td>Nonzero-set-point LQR step</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>505</td>
<td>Optimal controller</td>
</tr>
</tbody>
</table>

### LQR, Kalman filter, and covariances

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>801</td>
<td>Solve Riccati equation</td>
</tr>
<tr>
<td>802</td>
<td>Store solution matrix SS/PP</td>
</tr>
<tr>
<td>803</td>
<td>Store gain KC/KE</td>
</tr>
<tr>
<td>804</td>
<td>Check positive-definiteness of SS/PP</td>
</tr>
<tr>
<td>805</td>
<td>Check symmetry of SS/PP</td>
</tr>
<tr>
<td>806</td>
<td>Compute residual error</td>
</tr>
</tbody>
</table>

### Eigenvalues and eigenvectors

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>807</td>
<td>E-values of A - B*KC</td>
</tr>
<tr>
<td>808</td>
<td>E-values of A - B<em>KC - KE</em>H</td>
</tr>
<tr>
<td>809</td>
<td>E-values of ATOUT</td>
</tr>
<tr>
<td>810</td>
<td>E-vectors of A - B*KC</td>
</tr>
</tbody>
</table>

### Covariances, etc.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>811</td>
<td>Covariances of LQR system</td>
</tr>
<tr>
<td>812</td>
<td>Covariance error check</td>
</tr>
<tr>
<td>813</td>
<td>Feedforward gain calculation (KFF)</td>
</tr>
<tr>
<td>814</td>
<td>Store KFF</td>
</tr>
</tbody>
</table>

### User-supplied subroutines

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>UZR901</td>
</tr>
<tr>
<td>902</td>
<td>UZR902</td>
</tr>
<tr>
<td>903</td>
<td>UZR903</td>
</tr>
<tr>
<td>904</td>
<td>UZR904</td>
</tr>
</tbody>
</table>
NAMELIST MATDAT, in particular, problem dimensions and matrices A, B, C, D, DOUT, H, and CSP. However, the noise matrices and the weighting matrices can be modified here also if so desired.

205, 206, 207, 208, and 209—Functions for reading gain and Riccati solution matrices. AESOP contains three functions (801, 809, and 819) that solve Riccati equations and associated gain matrices and five functions (802, 808, 810, 816, and 820) that store results in datasets. Functions 205 to 209 are used for reading in datasets that contain gain or Riccati solution matrices previously computed by function 801, 809, or 819. Function 205 reads control gain matrix K from dataset CGS1 (S1 is the parameter in PROCDEF AESRUN). Function 206 reads Kalman filter gain matrix KF from dataset EG$1. Function 207 reads control Riccati solution matrix SS from dataset SS$1. Function 208 reads Kalman filter Riccati solution matrix PP from dataset PP$1. Function 209 reads feedforward gain matrix KFF from dataset FFG$1. Note that by using PAUSE function 102, the user can appropriately re-datadef any of the preceding datasets so as to read in the data from the particular dataset desired. Refer to table I for definition of the unit numbers associated with these five datasets.

Series 300—Matrix Formation

The three functions in series 300 all take gain and open-loop plant matrices and form various matrices related to closed-loop control system configurations. Separate functions are assigned to forming these matrices because a number of AESOP functions require these same matrices as input. Table VI summarizes the three functions and indicates for which other AESOP function these three functions are prerequisites. Examples of using these functions are given in appendix C; for 301, pages 69 and 79; for 302 and 303, page 70.

Series 400—Open-Loop System Analysis

This series of five functions is used in conducting pre-design analyses for the open-loop plant or for plant data normalization.

401—Open-loop system eigenvalues. Function 401 simply computes the eigenvalues of the A matrix. The eigenvalues are displayed in both Cartesian (α ± jβ) form and polar (frequency and ζ) where the frequency is in hertz and ζ is the damping ratio, \( ζ = \cos(\arctan β/α) \). A negative ζ indicates a right-half-plane eigenvalue (or pair). An example appears on page 68 of appendix C.

402—Open-loop eigenvectors and mode shapes. Function 402 computes and prints out the modified eigenvector matrix for the A matrix and also these vectors in mode-shape form. Refer to the section Theoretical Background for a description of the modified eigenvector and mode-shape format. Note that one must call function 401 before calling function 402. Use of function 402 is also shown on page 68 of appendix C.
TABLE VI. - AESOP FUNCTIONS USED FOR MATRIX FORMATION

<table>
<thead>
<tr>
<th>Function</th>
<th>Matrix formed</th>
<th>Dimension</th>
<th>Equation</th>
<th>Function where used</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>AMBKC</td>
<td>NxN</td>
<td>AMBKC = A - B + KC</td>
<td>513, 515, 603, 604, 803, 819</td>
</tr>
<tr>
<td>302</td>
<td>ABKCEH</td>
<td>NxN</td>
<td>ABKCEH = A - (B + KC - KE + H)</td>
<td>523, 705, 811</td>
</tr>
<tr>
<td>303</td>
<td>ATOT, DTOT, CTOT, KCTOT, HTOT</td>
<td>(a)</td>
<td>(a)</td>
<td>517, 519, 521, 812</td>
</tr>
</tbody>
</table>

\[ ^{\text{a}} \text{ATOT} = \begin{bmatrix} \frac{A}{\text{KE}+H} & -\frac{B \cdot \text{KC}}{\text{KE}+H} \\ \frac{A}{\text{KE}+H} & -\frac{B \cdot \text{KC}}{\text{KE}+H} \end{bmatrix}, \text{2Nx2N} \]

\[ \text{DTOT} = \begin{bmatrix} -D \\ 0 \end{bmatrix}, \text{2NxND} \]

\[ \text{CTOT} = [C - \text{DOUT} \cdot \text{KC}], \text{N} \times 2N \]

\[ \text{HTOT} = [H \cdot 0], \text{NM} \times 2N \]

\[ \text{KCTOT} = [0 - \text{KC}], \text{NC} \times 2N \]

403 - Controlability, observability, and residues. - Prerequisites for function 403 are functions 401 and 402. Given the open-loop system described by A, B, C, and H, function 403 computes the controls effectiveness matrix \((T^{-1} B)\) and two system observability matrices \((H^T \text{T} \text{C})\) and \((H^T \text{C})\). Also, this function computes the residues for systems \((A, B, \text{C})\) and \((A, B, H)\). All outputs are generated in mode-shape format. See page 68 of appendix C for an example of the use of function 403.

404 - Normalization of system matrices. - Function 404 allows the user to normalize all system matrices by using scaling factors that have been prestored in a dataset. The program prompts the user to datadef to unit 34 the dataset containing these (normalizing) factors. The factors are defined as

<table>
<thead>
<tr>
<th>System variable (vector)</th>
<th>Vector of normalization factor</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>SCX</td>
<td>50</td>
</tr>
<tr>
<td>u</td>
<td>SCU</td>
<td>5</td>
</tr>
<tr>
<td>z</td>
<td>SCZ</td>
<td>5</td>
</tr>
<tr>
<td>y</td>
<td>SCY</td>
<td>50</td>
</tr>
<tr>
<td>( \gamma_p )</td>
<td>SCYSP</td>
<td>5</td>
</tr>
</tbody>
</table>

The program initializes all normalizing factors to unity before reading in the dataset. The (VS) dataset must contain the data in NAMELIST form, where the NAMELIST name is NRMS. An example of a dataset’s contents might be

\&NRMS SCX(1) = 2., SCU(1) = 42., SCU(3) = .22
\&END

Here only the normalizing factors for the first and second state variables and the first and third control variables are to be nonunity. Matrices affected by normalization are A, B, C, H, QQ, RRINV, D, DOUT, and CSP. Examples of using function 404 and companion function 405 appear on pages 77 and 79 of appendix C.

405 - Unnormalization of matrices. - Function 405 is a companion to function 404 and allows resultant matrices KC, KE, KFF, and PP to be transformed back to dimensional form if desired. If prior normalization of input matrices is performed (via 404) in the same run of AESOP, no new scaling factors need be read in. However, function 405 automatically prompts the user (as is done in function 404) for the dataset containing the normalizing factors if not previously read in.

Series 500 - Frequency Responses and Bode Plots; and Series 700 - Transfer Functions

Functions in these two series (500 and 700) are closely related in that all allow the user to perform frequency domain input/output calculations for both open- and closed-loop system configurations. A general description of these calculations is provided in the section Theoretical Background under the heading Transfer Functions and Frequency Response Calculations. All functions in series 500 and 700 are summarized in table VII. Four system configurations are considered here:

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open loop</td>
<td>8</td>
</tr>
<tr>
<td>State-variable feedback</td>
<td>9</td>
</tr>
<tr>
<td>Kalman filter feedback</td>
<td>10</td>
</tr>
<tr>
<td>Optimal controller only</td>
<td>11</td>
</tr>
</tbody>
</table>

The figures describing the configurations appear in the section Theoretical Background. As noted in table VII, the user specifies the component of the input/output pair of interest by selecting values for indices JINC, JIND, IMEAS, or IOUT, which are all members of NAMELIST.
### Table VII. AESOP Functions for Computing Transfer Functions and Frequency Responses

<table>
<thead>
<tr>
<th>System configuration</th>
<th>Input variable</th>
<th>Output variable</th>
<th>Transfer functions</th>
<th>AESOP program function numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td>Index</td>
<td>Name</td>
<td>Index</td>
</tr>
<tr>
<td>Open loop</td>
<td>u</td>
<td>JINC</td>
<td>z</td>
<td>IMEAS</td>
</tr>
<tr>
<td></td>
<td>w</td>
<td>JIND</td>
<td>y</td>
<td>IOUT</td>
</tr>
<tr>
<td>State-variable feedback</td>
<td>y</td>
<td>IOUT</td>
<td>---</td>
<td>803</td>
</tr>
<tr>
<td>State-variable feedback</td>
<td>u</td>
<td>JINC</td>
<td>---</td>
<td>803</td>
</tr>
<tr>
<td>Kalman filter feedback</td>
<td>z</td>
<td>IMEAS</td>
<td>---</td>
<td>812</td>
</tr>
<tr>
<td>Kalman filter feedback</td>
<td>y</td>
<td>IOUT</td>
<td>---</td>
<td>812</td>
</tr>
<tr>
<td>Optimal controller only</td>
<td>z</td>
<td>IMEAS</td>
<td>u</td>
<td>JINC</td>
</tr>
</tbody>
</table>

aIndices are included in NAMELIST REFS (table III). Index values determine vector input/output pair for which the transfer function or frequency response will be computed.
bComputes (1) coefficients of the transfer function numerator and denominator polynomials and (2) frequency response magnitudes and phase angles for initial frequency $f_1$(Hz), frequency spacing $\Delta f$(Hz), number of points $IF$($\leq$1000), and a time scale factor $TSF$.
cIf $NCURV = 2$, this closed-loop response will be crossplotted with its corresponding open-loop response. Note that the open-loop response used is the last one calculated.

MPS - It can be seen that transfer function numerator and denominator polynomial coefficients (eq. (42)), frequency response calculations, storing of frequency responses, and Bode plots are available for all four system configurations. Specific functions allow the user to choose between control $u$ or disturbance $w$ inputs and between noisy $z$ or noise-free $y$ outputs. One exception is for the optimal-controller-only configuration, where the only input is measurement vector $z$ and the output is the control vector $u$. Examples of the use of 500 series functions appear on pages 72 to 74 of appendix C, including plotted output. Transfer function zeroes and gain information can be obtained only for the open-loop plant and for optimal controller configurations. However, transfer function poles can be computed for all configurations. Note that these are all eigenvalue computations and are included in series 400 or series 800. Examples of the use of functions 701 to 705 are shown on pages 76 and 77 of appendix C.

Frequency response data (frequency, amplitude, and phase angle), which are stored on datasets by functions 503, 506, 509, 512, and 525, are all written by using a Fortran WRITE statement of the form

```
WRITE (i, b) (FREQ(I),AMP(I), PHASE(I),I=1,IF)
```

Here, $i$ is an appropriate unit number, IF is the number of response data points, and FORMAT statement b is FORMAT $(10G12.5)$. An appropriately formatted READ statement would be used if one wished to use any of these computed frequency responses in a separate program.

#### Series 600 - Transient Responses

Transient responses— for either step or initial condition inputs—are computed by the four AESOP functions in this series. Pertinent information required to use these functions is shown in table VIII. Functions 601 and 602 are for the open-loop system configuration and 603 and 604, for linear quadratic regulators. Note that to use a function, the user should specify:

1. Input amplitude (or amplitudes), AMP...
2. Desired time step, DT
3. The number of time points to be computed, ITRMX $\leq$ 1000
4. Desired input/output response pairs for which responses are to be calculated, MS or MIC...

All of these parameters are in NAMELIST REFS and are set to default values in the main AESOP program (see table III for default values). The transient response plots that are generated by these functions will be generated on the particular graphics device that the user has previously defined. Examples of these plots and the use of functions 601 to 604 appear in test case I in appendix C.

#### Series 800 - LQR and Filter Design

Functions in series 800 are for (1) solving the Riccati equations associated with the LQR or Kalman filter design problems or (2) related gain, covariance, or eigenvalue/eigenvector calculations.

**Functions for solving Riccati equations**— Table IX lists the numbers of the functions in the 800 series that deal directly with LQR or Kalman filter design. Three
types of functions are provided: (1) those for solving Riccati equations, (2) those for storing Riccati equation results in datasets for future use, and (3) those for checking the accuracy of Riccati solutions. Companion functions are provided in series 200 for subsequently reading in the matrices computed and stored by functions in the 800 series. Examples of the use of functions involved with the LQR problem (801 to 808) are given in appendix C, pages 68 to 70. Examples of Kalman filter design calculations (functions 809 to 816) appear on pages 70 to 72 of appendix C. To read Riccati or gain matrices from datasets into another program (e.g., into a program that implements an LQR control law), the user must know the correct Fortran READ statements to use. The appropriate (unformatted) READ statements are as follows:

(1) For reading the control Riccati solution (stored in dataset SS$1$):

```
READ (i)((SS(I,J),I=1,N),J=1,N)
```

(2) For reading the Kalman filter Riccati solution (stored in dataset PP$1$):

```
READ (i)((PP(I,J),I=1,N),J=1,N)
```

(3) For reading the LQR gains (stored in dataset CG$1$):

```
READ (i)((KC(I,J),I=1,NC),J=1,N)
```

(4) For reading the Kalman filter gains (stored in dataset EG$1$):

```
READ (i)((KE(I,J),I=1,NC),J=1,N)
```
READ (i)((KE(I,J),I=1,N),J=1,NM)

(5) For reading the feedforward gains (stored in dataset FFG$)j$):
READ (i)((KFF(I,J),I=1,NC),J=1,NC)

It is assumed that the user datadefs unit "i" to the appropriate dataset.

Functions for eigenvalue/eigenvector computation. – Table X lists the functions that compute eigenvalues and eigenvectors associated with either regulator or filter designs. The eigenvalues and eigenvectors of the matrix $A - B\cdot KC$ are those of the linear regulator. The eigenvalues of matrix $A - B\cdot KC - KE\cdot H$ are those of the optimal controller (depicted in fig. 11). The eigenvalues of matrix ATOT are those of $A - B\cdot KC$ plus those of $A - KE\cdot H$, the latter being the Kalman filter eigenvalues.

817–Covariance matrices for system controlled by linear stochastic regulator. – Function 817 implements the solution to the state covariance matrix (Lyapunov) equation given by equation (32) plus associated matrix equations (33), (34), and (35). The latter equations compute control, measurement, and output covariance matrices. Normally, function 817 is called after computing LQR gain matrix $KC$ and Kalman filter gain matrix $KE$. However, one or both of these two matrices may be zero when function 817 is called. The resultant state covariance matrix $XX$ will then correspond to the meaningful system configurations as shown in the following table. An example of the use of function 817 appears on page 72 of appendix C.

818–Covariance matrix error check. – Function 818 may be called after function number 817 to compute an estimate of the error incurred in computing the state covariance matrix. An example of its use appears on page 72 of appendix C.

819–Feedforward gain matrix for nonzero-set-point regulator. – Function 819 computes feedforward gain matrix $KFF$, which is part of the nonzero-set-point regulator. Matrix $KFF$ is given by

$$KFF = \left(-CSP(A - B\cdot KC)^{-1}B\right)^{-1}$$

The matrix $KFF$ is needed when computing the closed-loop system step responses with function 604. Functions 819 and 820 are demonstrated in test case I, appendix C, on page 72.

820–Store feedforward gain matrix. – Function 820 stores $KFF$ in a dataset by using the following unformatted WRITE statement:

WRITE(iii)((KFF(I,J),I=1,NC),J=1,NC)

The user can read these data into another program by using a matching Fortran READ statement.

Series 900–User-Supplied Subroutines

A series of four function numbers have been set aside for use as user-defined subroutines. These subroutines are called UZR901, UZR902, UZR903, and UZR904. Thus users may write their own special-purpose subroutines by using any of the above as subroutine names. Linkage between the subroutine and the main AESOP program would be achieved by using any of the COMMON's used in the AESOP program. Users need not recompile the AESOP subroutine containing CALLS to the four user-supplied subroutines, but need only compile their subroutines into a library that has higher priority in the JOBLIB chain than the "standard" AESOP program library.

Concluding Remarks

The program described in this report was not meant to be a static entity but rather was envisioned as a basic structure to which can be added new and useful system design functions as the need arises. Some functions presently contemplated for future inclusion are discrete LQR and Kalman filter gain calculations, time responses for Kalman filters to noise signal inputs, linear model-order reduction procedures, LQR or Kalman filter eigenvalue sensitivity calculations, transient responses of
discrete LQR or Kalman filters, and multivariable frequency domain control design algorithms.

Modifications are also envisioned that could improve program-user interfacing. One would be to replace the present function-number-input system with a set of mnemonics (i.e., the user would enter LQR instead of 801 when requesting an LQR problem solution, etc.). Another addition would be to allow the user to use a light pen to select AESOP functions from a menu displayed on a CRT terminal screen. The present program structure has sufficient flexibility to accommodate these types of additions without compromising present interactive capability.

Lewis Research Center
National Aeronautics and Space Administration
Cleveland, Ohio, June 14, 1983
## Appendix A

### Symbols

Dimensions are given for all vectors, matrices, and higher dimensional arrays. Where dimensions are not given, the variable is a scalar quantity.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$N \times N$</td>
<td>system matrix</td>
</tr>
<tr>
<td>$\hat{A}$</td>
<td>$n \times n$</td>
<td>matrix whose eigenvalues are transfer function zeroes</td>
</tr>
<tr>
<td>$\dot{A}$</td>
<td>$n \times n$</td>
<td>general matrix</td>
</tr>
<tr>
<td>$ABKCEH$</td>
<td>$N \times N$</td>
<td>matrix $A - B(KC - KE-H)$</td>
</tr>
<tr>
<td>$AMBKC$</td>
<td>$N \times N$</td>
<td>matrix $A - B KC$</td>
</tr>
<tr>
<td>$AMPICX$</td>
<td>$N$</td>
<td>vector of initial-condition amplitudes</td>
</tr>
<tr>
<td>$AMPSP$</td>
<td>$NC$</td>
<td>vector of input step amplitudes for closed-loop system</td>
</tr>
<tr>
<td>$AMPSR$</td>
<td>$NC$</td>
<td>vector of input step amplitudes for open-loop system</td>
</tr>
<tr>
<td>$ATOT$</td>
<td>$NTOT \times NTOT$</td>
<td>system matrix for combined regulator-Kalman filter system</td>
</tr>
<tr>
<td>$a_k$</td>
<td></td>
<td>coefficient of transfer function numerator polynomial</td>
</tr>
<tr>
<td>$B$</td>
<td>$N \times NC$</td>
<td>control input matrix</td>
</tr>
<tr>
<td>$\check{B}$</td>
<td>$N \times NC$</td>
<td>controllability matrix</td>
</tr>
<tr>
<td>$b_k$</td>
<td>$n \times nc$</td>
<td>general matrix</td>
</tr>
<tr>
<td>$C$</td>
<td>$NO \times N$</td>
<td>output matrix</td>
</tr>
<tr>
<td>$\check{C}$</td>
<td>$no \times n$</td>
<td>general matrix</td>
</tr>
<tr>
<td>$CS$</td>
<td>$NC \times N$</td>
<td>set-point output matrix</td>
</tr>
<tr>
<td>$CS$</td>
<td>$NC \times N$</td>
<td>normalized $CS$ matrix</td>
</tr>
<tr>
<td>$CTOT$</td>
<td>$NO \times NTOT$</td>
<td>output matrix for combined regulator-Kalman filter system</td>
</tr>
<tr>
<td>$\check{D}$</td>
<td>$no \times nc$</td>
<td>general matrix</td>
</tr>
<tr>
<td>$D$</td>
<td>$N \times ND$</td>
<td>disturbance input matrix</td>
</tr>
<tr>
<td>DELF</td>
<td></td>
<td>spacing between frequency points</td>
</tr>
<tr>
<td>DOUT</td>
<td>$NO \times NC$</td>
<td>feedforward matrix</td>
</tr>
<tr>
<td>DT</td>
<td></td>
<td>time step</td>
</tr>
<tr>
<td>DTOT</td>
<td>$NTOT \times ND$</td>
<td>disturbance input matrix for combined regulator-Kalman filter system</td>
</tr>
<tr>
<td>$e$</td>
<td>$N$</td>
<td>Kalman filter estimation error vector</td>
</tr>
<tr>
<td>$E$</td>
<td>$N \times N$</td>
<td>error matrix ($\hat{X} - X$)</td>
</tr>
<tr>
<td>$E_j$</td>
<td>$NM \times NC$</td>
<td>selection matrix</td>
</tr>
<tr>
<td>$Fi$</td>
<td></td>
<td>initial frequency</td>
</tr>
<tr>
<td>$G(s)$</td>
<td>$NM \times NC$</td>
<td>transfer function matrix</td>
</tr>
<tr>
<td>$g(s)$</td>
<td></td>
<td>transfer function</td>
</tr>
<tr>
<td>$H$</td>
<td>$NM \times N$</td>
<td>measurement matrix</td>
</tr>
<tr>
<td>$\dot{H}$</td>
<td>$NM \times N$</td>
<td>observability matrix</td>
</tr>
<tr>
<td>HTOT</td>
<td>$NM \times NTOT$</td>
<td>measurement matrix for combined regulator-Kalman filter system</td>
</tr>
<tr>
<td>$I$</td>
<td>$N \times N$</td>
<td>identity matrix</td>
</tr>
<tr>
<td>IF</td>
<td></td>
<td>integer, number of desired frequency response points</td>
</tr>
<tr>
<td>IMEAS</td>
<td></td>
<td>integer, index of measurement</td>
</tr>
<tr>
<td>IOUT</td>
<td></td>
<td>integer, index of output</td>
</tr>
<tr>
<td>ISPACE</td>
<td></td>
<td>integer, controls frequency of frequency response printouts</td>
</tr>
<tr>
<td>ITRMX</td>
<td></td>
<td>integer, number of desired time response points</td>
</tr>
<tr>
<td>$J$</td>
<td></td>
<td>performance index</td>
</tr>
<tr>
<td>JINC</td>
<td></td>
<td>integer, index of control</td>
</tr>
<tr>
<td>JIND</td>
<td></td>
<td>integer, index of disturbance</td>
</tr>
<tr>
<td>$K_{ij}$</td>
<td></td>
<td>transfer function gain</td>
</tr>
<tr>
<td>$KC$</td>
<td>$NC \times N$</td>
<td>control gain matrix</td>
</tr>
<tr>
<td>KCTOT</td>
<td>$NC \times NTOT$</td>
<td>control gain matrix for combined regulator-Kalman filter system</td>
</tr>
<tr>
<td>$KE$</td>
<td>$N \times NM$</td>
<td>Kalman filter gain matrix</td>
</tr>
<tr>
<td>KFF</td>
<td>$NC \times NC$</td>
<td>feedforward gain matrix for nonzero-set-point regulator</td>
</tr>
<tr>
<td>$k^*$</td>
<td></td>
<td>feedback gain term</td>
</tr>
<tr>
<td>LINLOG</td>
<td></td>
<td>integer, indicates whether frequency response plots are to be linear, log, or both</td>
</tr>
<tr>
<td>$\ell$</td>
<td></td>
<td>matrix dimension</td>
</tr>
</tbody>
</table>
MCCLU \( N \times NC \) integer, input/output, closed-loop, initial-condition response selection matrix for controls; state feedback

MICLX \( N \times N \) integer, input/output, closed-loop, initial-condition response selection matrix for states; state feedback

MICLY \( N \times NO \) integer, input/output, closed-loop, initial-condition response selection matrix for outputs; state feedback

MICOLX \( N \times N \) integer, input/output, open-loop, initial-condition response selection matrix for states

MICOLY \( N \times NO \) integer, input/output, open-loop, initial-condition response selection matrix for outputs

MSPU \( NC \times NC \) integer, input/output, closed-loop, step response selection matrix for controls; state feedback

MSPY \( NC \times NO \) integer, input/output, closed-loop, step response selection matrix for outputs; state feedback

MSPYSP \( NC \times NC \) integer, input/output, closed-loop, step response selection matrix for set-point outputs; state feedback

MSROLX \( NC \times N \) integer, input/output, open-loop, step response selection matrix for states

MSROLY \( NC \times NO \) integer, input/output, open-loop, step response selection matrix for outputs

M \( N \times \ell \) matrix, general factor of QC matrix

ND integer, number of disturbances

NM integer, number of measurements

NMAX integer, maximum number of states

NN \( N \times NC \) state-control weighting matrix

NO integer, number of outputs

NTOT integer, \( 2 \times N \)

\( n \) number of states for general dynamic system

nc number of inputs for general dynamic system

no number of outputs for general dynamic system

PCINV \( NC \times NC \) inverse of control weighting matrix

PP \( N \times N \) Kalman filter error covariance matrix

\( P_k \) transfer function pole

\( Q \) symmetric, positive-definite matrix

QC \( N \times N \) state weighting matrix

QQ \( N \times N \) power spectral density matrix of plant disturbance

R \( R \) residual matrix

R \( R_{ij} \) residue matrix of transfer function matrix \( G(s) \)

RRINV \( NM \times NM \) inverse of power spectral density matrix of measurement noise

\( r_j \) jth residue of transfer function \( g(s) \)

SCU \( NC \) vector of normalization factors for controls

SCX \( N \) vector of normalization factors for states

SCY \( NO \) vector of normalization factors for outputs

SCYSP \( NC \) vector of normalization factors for set-point outputs

SCZ \( NM \) vector of normalization factors for measurements

SS \( N \times N \) Riccati solution matrix for linear quadratic regulator

\( S \) Laplace variable
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>N x N</td>
</tr>
<tr>
<td>TSFTR</td>
<td>time scale factor</td>
</tr>
<tr>
<td>l</td>
<td>time</td>
</tr>
<tr>
<td>$t_i, t_{i+1}$</td>
<td>modified eigenvectors</td>
</tr>
<tr>
<td>UU</td>
<td>NC x NC</td>
</tr>
<tr>
<td>u</td>
<td>NC</td>
</tr>
<tr>
<td>v</td>
<td>NM</td>
</tr>
<tr>
<td>W</td>
<td>N x N</td>
</tr>
<tr>
<td>w</td>
<td>ND</td>
</tr>
<tr>
<td>X</td>
<td>N x N</td>
</tr>
<tr>
<td>$\bar{X}$</td>
<td>computed value of matrix X</td>
</tr>
<tr>
<td>XX</td>
<td>N x N</td>
</tr>
<tr>
<td>x</td>
<td>N</td>
</tr>
<tr>
<td>$\dot{x}$</td>
<td>state vector</td>
</tr>
<tr>
<td>$\dot{x}$</td>
<td>modal state vector</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>N x N</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>output covariance matrix</td>
</tr>
<tr>
<td>y</td>
<td>NO</td>
</tr>
<tr>
<td>$\gamma_p$</td>
<td>set-point vector</td>
</tr>
<tr>
<td>$\gamma_{spd}$</td>
<td>desired set-point vector</td>
</tr>
<tr>
<td>ZERMAX</td>
<td>maximum expected value of transfer function zeroes</td>
</tr>
<tr>
<td>ZZ</td>
<td>NM x NM</td>
</tr>
<tr>
<td>z</td>
<td>NM</td>
</tr>
<tr>
<td>$z_1$</td>
<td>NM</td>
</tr>
<tr>
<td>$\hat{z}$</td>
<td>NM</td>
</tr>
<tr>
<td>$z_k$</td>
<td>NM</td>
</tr>
<tr>
<td>$\hat{z}$</td>
<td>estimated value of $z$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>numerator zero</td>
</tr>
<tr>
<td>$\beta$</td>
<td>imaginary part of eigenvalue</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>discrete system</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>real part of eigenvector</td>
</tr>
<tr>
<td>$\delta$</td>
<td>imaginary part of eigenvector</td>
</tr>
<tr>
<td>$\delta(t)$</td>
<td>unit impulse function</td>
</tr>
<tr>
<td>$\xi$</td>
<td>N</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>N</td>
</tr>
<tr>
<td>$\lambda_i, \lambda_{i+1}$</td>
<td>complex eigenvalue pair</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>N x N</td>
</tr>
<tr>
<td>$\mathbf{E}{\cdot}$</td>
<td>expected value of</td>
</tr>
</tbody>
</table>

Superscripts:
- T
- -1

Operators:
- $\mathbf{E}\{\cdot\}$
Appendix B
Subroutine Descriptions

This appendix describes the AESOP main program and all associated subroutines. A short description of each subroutine is given, including what subroutines may call it, and what subroutines it calls. Where a subroutine has a parameter list, all variables in that list are defined.

A tape of the AESOP program documented in this report is available from COSMIC. That version of AESOP has been sized to accommodate systems having the following dimensions:

- N ≤ 50
- NM ≤ 5
- NC ≤ 5
- ND ≤ 15
- NO ≤ 50

If these dimensions are greater than or equal to the corresponding dimensions of the user's problem, the user need make no changes to the main AESOP program (assuming that the program will fit on the user's computer). However, the user may wish to resize the program, either to accommodate problems with larger dimensions or to reduce storage requirements in order to allow the program to fit on a smaller computer. As currently dimensioned, AESOP requires approximately 300 000 bytes for all Fortran source codes and about 1 700 000 bytes for variable storage (the variables in COMMON's appearing in the main program). To change program dimensions, array dimensions appearing in 11 labeled COMMON's must be changed. These COMMON's appear only in the main program and the nine main subroutines (AES100 to AES900).

AESOP

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

AESOP IS THE MAIN ROUTINE. AESOP CALLS SUBROUTINES AES100, AES200, AES300, AES400, AES500, AES600, AES700, AES800, AES900, AND THE PLOTTING SUBROUTINES.

AESOP IS NOT CALLED BY ANY SUBROUTINES.

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

SUBROUTINE AES100 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)

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

SUBROUTINE AES100 CONTAINS THOSE FUNCTIONS WHICH PERFORM PROGRAM CONTROL. AES100 CALLS SUBROUTINE PREREQ.

AES100 IS CALLED BY MAIN PROGRAM AESOP.

INPUTS:

IFN VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)
IFUNC FUNCTION NUMBER
MZ WHICH FUNCTION IS TO BE DONE
IPRT PRINT OPTION: 1, STANDARD PRINT;
WHEN LOGICAL MATRIX OF PREREQUISITES (450,50)

OUTPUTS:

IAND DECISION VARIABLE:
0, PREREQUISITES HAVE BEEN DONE;
1, PREREQUISITES HAVE NOT BEEN DONE

******************************************************************************
SUBROUTINE AES200 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)

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

SUBROUTINE AES200 CONTAINS THOSE FUNCTIONS WHICH INPUT DATA. AES200 CALLS SUBROUTINES MATCHG, MATIN, MATPKI, MATRD, AND PREREQ. AES200 IS CALLED BY MAIN PROGRAM AESOP.

INPUTS:

   IFN   VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)
   IFUNC FUNCTION NUMBER
   MZ WHICH FUNCTION IS TO BE DONE
   IPRT PRINT OPTION: 1, STANDARD PRINT;
                     2, EXTENDED PRINT
   WHEN LOGICAL MATRIX OF PREREQUISITES (450,50)

OUTPUTS:

   IAND DECISION VARIABLE:
          0, PREREQUISITES HAVE BEEN DONE;
          1, PREREQUISITES HAVE NOT BEEN DONE

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

SUBROUTINE AES300 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)

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

SUBROUTINE AES300 CONTAINS THOSE FUNCTIONS WHICH FORM MATRICES. AES300 CALLS SUBROUTINES MATPRT AND PREREQ. AES300 IS CALLED BY MAIN PROGRAM AESOP.

INPUTS:

   IFN   VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)
   IFUNC FUNCTION NUMBER
   MZ WHICH FUNCTION IS TO BE DONE
   IPRT PRINT OPTION: 1, STANDARD PRINT;
                     2, EXTENDED PRINT
   WHEN LOGICAL MATRIX OF PREREQUISITES (450,50)

OUTPUTS:

   IAND DECISION VARIABLE:
          0, PREREQUISITES HAVE BEEN DONE;
          1, PREREQUISITES HAVE NOT BEEN DONE

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

SUBROUTINE AES400 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)

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

SUBROUTINE AES400 CONTAINS THOSE FUNCTIONS WHICH CALCULATE CONTROLLABILITY, OBSERVABILITY, EIGENVALUES, EIGENVECTORS, AND RESIDUES, AND WHICH PERFORM NORMALIZATION AND UN-NORMALIZATION. AES400 CALLS SUBROUTINES CTBL, EGVCRT, EIGEN, MATPRT, MODSHP, NRML, OBSBL, PREREQ, RESI, AND UNRML. AES400 IS CALLED BY MAIN PROGRAM AESOP.

INPUTS:

   IFN   VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)
   IFUNC FUNCTION NUMBER
   MZ WHICH FUNCTION IS TO BE DONE
IPRT  PRINT OPTION: 1, STANDARD PRINT;  
2, EXTENDED PRINT  

WHEN  LOGICAL MATRIX OF PREREQUISITES (450,50)  

OUTPUTS:  

IAND  DECISION VARIABLE:  
0, PREREQUISITES HAVE BEEN DONE;  
1, PREREQUISITES HAVE NOT BEEN DONE  

SUBROUTINE AES500 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)  

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

SUBROUTINE AES500 CONTAINS THOSE FUNCTIONS WHICH CALCULATE FREQUENCY RESPONSE AND BODE PLOTS. AES500 CALLS SUBROUTINES BODE, FRSPNS, AND PREREQ. AES500 IS CALLED BY MAIN PROGRAM AESOP.  

INPUTS:  

IFN  VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)  
IFUNC  FUNCTION NUMBER  
MZ  WHICH FUNCTION IS TO BE DONE  
IPRT  PRINT OPTION: 1, STANDARD PRINT;  
2, EXTENDED PRINT  
WHEN  LOGICAL MATRIX OF PREREQUISITES (450,50)  

OUTPUTS:  

IAND  DECISION VARIABLE:  
0, PREREQUISITES HAVE BEEN DONE;  
1, PREREQUISITES HAVE NOT BEEN DONE  

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

SUBROUTINE AES600 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)  

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

SUBROUTINE AES600 CONTAINS THOSE FUNCTIONS WHICH CALCULATE TIME RESPONSES AND THE ASSOCIATED PLOTS. AES600 CALLS SUBROUTINES DSCRT, ICRSP, MATPRT, PREREQ, AND STP. AES600 IS CALLED BY MAIN PROGRAM AESOP.  

INPUTS:  

IFN  VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)  
IFUNC  FUNCTION NUMBER  
MZ  WHICH FUNCTION IS TO BE DONE  
IPRT  PRINT OPTION: 1, STANDARD PRINT;  
2, EXTENDED PRINT  
WHEN  LOGICAL MATRIX OF PREREQUISITES (450,50)  

OUTPUTS:  

IAND  DECISION VARIABLE:  
0, PREREQUISITES HAVE BEEN DONE;  
1, PREREQUISITES HAVE NOT BEEN DONE  

******************************************************************
SUBROUTINE AES700 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)

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

SUBROUTINE AES700 CONTAINS THOSE FUNCTIONS WHICH CALCULATE ZEROES AND GAINS.
AES700 CALLS SUBROUTINES GAIN, MATPRT, PREREQ, AND ZEROES.
AES700 IS CALLED BY MAIN PROGRAM AESOP.

INPUTS:
IFN VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)
IFUNC FUNCTION NUMBER
MZ WHICH FUNCTION IS TO BE DONE
IPRT PRINT OPTION: 1, STANDARD PRINT;
2, EXTENDED PRINT
WHEN LOGICAL MATRIX OF PREREQUISITES (450,50)

OUTPUTS:
IAND DECISION VARIABLE:
0, PREREQUISITES HAVE BEEN DONE;
1, PREREQUISITES HAVE NOT BEEN DONE

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

SUBROUTINE AES800 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)

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

SUBROUTINE AES800 CONTAINS THOSE FUNCTIONS WHICH CALCULATE EIGENVALUES AND EIGENVECTORS OF CONTROLS OR FILTERS. AES800 CALLS SUBROUTINES CONTRL, COVAR, EGVCTR, EIGEN, ESTMAT, LYPCK, MATPRT, MODSHIP, MAINV, PREREQ, AND RICCHK.
AES800 IS CALLED BY MAIN PROGRAM AESOP.

INPUTS:
IFN VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)
IFUNC FUNCTION NUMBER
MZ WHICH FUNCTION IS TO BE DONE
IPRT PRINT OPTION: 1, STANDARD PRINT;
2, EXTENDED PRINT
WHEN LOGICAL MATRIX OF PREREQUISITES (450,50)

OUTPUTS:
IAND DECISION VARIABLE:
0, PREREQUISITES HAVE BEEN DONE;
1, PREREQUISITES HAVE NOT BEEN DONE

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

SUBROUTINE AES900 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)

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

SUBROUTINE AES900 CONTAINS THOSE FUNCTIONS WHICH ARE SUPPLIED BY THE USER. AES900 DOES NOT CALL ANY SUBROUTINES.
AES900 IS CALLED BY MAIN PROGRAM AESOP.

INPUTS:
IFN VECTOR OF FUNCTION NUMBERS TO BE DONE (1000)
IFUNC FUNCTION NUMBER
MZ WHICH FUNCTION IS TO BE DONE
IPRT PRINT OPTION: 1, STANDARD PRINT;
2, EXTENDED PRINT
WHEN LOGICAL MATRIX OF PREREQUISITES (450,50)

OUTPUTS:
IAND DECISION VARIABLE:
0, PREREQUISITES HAVE BEEN DONE;
1, PREREQUISITES HAVE NOT BEEN DONE

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

SUBROUTINE ARRAY (IOPT, I, J, NROW, A)
******************************************************************************************

SUBROUTINE ARRAY CONVERTS AN ARRAY FROM VECTOR TO MATRIX OR THE REVERSE. ARRAY DOES NOT CALL ANY SUBROUTINES. ARRAY IS CALLED BY SUBROUTINES COVAR, EGVCTR, EIGEN, LYPCK, RICSS, AND ZEROES.

INPUTS:
IOPT OPTION INDICATING TYPE OF CONVERSION
1 - FROM VECTOR TO MATRIX
2 - FROM MATRIX TO VECTOR
I NUMBER OF ROWS IN ACTUAL MATRIX
J NUMBER OF COLUMNS IN ACTUAL MATRIX
NROW NUMBER OF ROWS SPECIFIED FOR THE MATRIX A IN DIMENSION STATEMENT
A IF MODE = 1, CONTAINS A VECTOR OF I*J LENGTH.
   IF MODE = 2, CONTAINS A MATRIX OF N BY J SIZE.

OUTPUTS:
A IF MODE = 1, CONTAINS A MATRIX OF N BY J SIZE.
   IF MODE = 2, CONTAINS A VECTOR OF I*J LENGTH.

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

BLOCK DATA
******************************************************************************************

THE BLOCK DATA SUBROUTINE CONTAINS ONLY INFORMATION FOR PLOT TITLES AND LABELS
******************************************************************************************

SUBROUTINE BODE (FRQ, A1, A2, PHI1, PHI2, TTITL, TB, NPTS, KI, 1 AMP, PHA, SETAP, KTYPE1, KTYPE2, IP, NAME, IONPLT)
******************************************************************************************

SUBROUTINE BODE MAKES PLOTS OF FREQUENCY RESPONSES. AMP, PHA AND SETAP MUST BE SINGLE PRECISION BECAUSE OF THE PLOT SUBROUTINES. BODE CALLS PLOTTING SUBROUTINES ONLY. BODE IS CALLED BY SUBROUTINE AES500.

INPUTS:
FRQ VECTOR OF FREQUENCY
   (DIMENSION IS LESS THAN OR EQUAL TO 500)
A1 VECTOR OF AMPLITUDE FOR 1ST CURVE
   (DIMENSION IS LESS THAN OR EQUAL TO 500)
A2 VECTOR OF AMPLITUDE FOR 2ND CURVE, IF DESIRED  
(DIMENSION IS LESS THAN OR EQUAL TO 500)

PHI1 VECTOR OF PHASE FOR 1ST CURVE 
(DIMENSION IS LESS THAN OR EQUAL TO 500)

PHI2 VECTOR OF PHASE FOR 2ND CURVE, IF DESIRED  
(DIMENSION IS LESS THAN OR EQUAL TO 500)

TTITL TITLE OF PLOT  
(DIMENSION IS GREATER THAN OR EQUAL TO 15)

TB TITLE OF PLOT 
(DIMENSION IS GREATER THAN OR EQUAL TO 15)

NPTS NUMBER OF POINTS PER CURVE  
(LESS THAN OR EQUAL TO 500)

KI 1, IF ONE CURVE PER PLOT  
2, IF TWO CURVES PER PLOT

KTYPE1 THESE TWO VARIABLES DEFINE WHETHER  
) THE PLOT IS TO BE LINEAR,

KTYPE2 LOG, OR SEMI-LOG

IP PLOT ENTITY INDEX (USED BY PLOTSUBS ONLY) 
INCREASES BY ONE FOR EACH FRAME

NAME NAME OF PLOT DATASET (9) (USED BY PLOTSUBS ONLY)  
(PARTITIONED DATASET THAT HOLDS PLOT ENTITIES)

IONPLT 0, IF OFFLINE PLOTS  
1, IF ONLINE PLOTS

OUTPUTS:
AMP STORAGE VECTOR OF AMPLITUDES FOR 1 OR 2 CURVES  
(DIMENSION IS LESS THAN OR EQUAL TO 1000)

PHA STORAGE VECTOR OF PHASES FOR 1 OR 2 CURVES  
(DIMENSION IS LESS THAN OR EQUAL TO 1000)

IP PLOT ENTITY INDEX (USED BY PLOTSUBS ONLY) 
INCREASES BY ONE FOR EACH FRAME

TEMPORARY STORAGE:
SETAP VECTOR 
(DIMENSION IS LESS THAN OR EQUAL TO 500)

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

SUBROUTINE BOLLIN (A, AS, B, C, X, Y, Z1, Z2, Z3, N, NMAX)
***************************************************************************

SUBROUTINE BOLLIN CONVERTS X(DOT)=AX+BU,Y=C(TRANSPOSE)X  
TO TRANSFER FUNCTION Y/U=Z2/Z1 RATIO OF POLYNOMIALS.  
BOLLIN CALLS SUBROUTINES DAVIS0 AND DANSKY. BOLLIN IS CALLED BY  
SUBROUTINE FRSPNS.

INPUTS:
A SYSTEM MATRIX (N,N)  
B SYSTEM VECTOR (N)  
C SYSTEM VECTOR (N)  
N ACTUAL SIZE OF MATRIX A  
NMAX MAXIMUM SIZE OF N

OUTPUTS:
Z1 DENOMINATOR COEFFICIENT VECTOR (N)  
Z2 NUMERATOR COEFFICIENT VECTOR (N)  
Z3 NUMERATOR COEFFICIENT VECTOR (N)

TEMPORARY STORAGE:
AS MATRIX (N,N)  
X VECTOR (N)  
Y VECTOR (N)

***************************************************************************
SUBROUTINE CONDI (VARO, SS, S, IN, JBL, IOR, NBL, IBL, IC, D,  
1 IOP1, N, NMAX)

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

SUBROUTINE CONDI CHANGES CONDITION OF A MATRIX BY PUTTING IT IN 
BLOCK DIAGONAL FORM (IF REDUCIBLE) AND THEN SCALING. 
CONDI CALLS SUBROUTINES REDU AND SCALEA. CONDI IS CALLED BY 
SUBROUTINES EIGEN AND ZEROES.

INPUTS:

VARO  MATRIX TO BE CONDITIONED (N,N)  
IOP1  PRINT OPTION: 0 NO PRINT, 1 PRINT  
N    ACTUAL SIZE OF MATRIX VARO  
NMAX  MAXIMUM SIZE OF N

OUTPUTS:

S    CONDITIONED MATRIX (N,N)  
IOR  BLOCK-DIAGONALIZING PERMUTATION INTEGER 
     VECTOR (N) 
NBL  INTEGER VECTOR OF SIZES OF EACH IRREDUCIBLE 
     BLOCK (N) 
D    VECTOR OF DIAGONAL ELEMENTS OF DIAGONAL SCALING 
     MATRIX (N)

TEMPORARY STORAGE:

SS  MATRIX (N,N)  
IN  INTEGER VECTOR (N)  
JBL  INTEGER VECTOR (N)  
IBL  INTEGER VECTOR (N)  
IC  INTEGER VECTOR (N)

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

SUBROUTINE CONTRL (AA, BB, QC, NN, PCINV, KC, SS, CR, CI, X, TS,  
1 XR, TT, AAA, EXT, AR, AI, IPER, IPEPN, ADBLE, N, NC, N2, IOP1,  
2 IOP2, NMAX, NCMAX, N2MAX)

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

SUBROUTINE CONTRL SOLVES THE OPTIMAL LINEAR REGULATOR PROBLEM. IT 
SETS UP AN N2 BY N2 MATRIX AAA, USING MATRICES AA, BB, QC, NN, AND 
PCINV. CONTRL OBTAINS THE SOLUTION TO THE RICCATI EQUATION, SS, 
AND THEN COMPUTES THE CONTROL GAINS, KC. CONTRL CALLS SUBROUTINES 
MATPRT AND RICSS. CONTRL IS CALLED BY SUBROUTINE AES800.

INPUTS:

AA    SYSTEM MATRIX (N,N)  
BB    CONTROL INPUT MATRIX (N,NC)  
QC    STATE WEIGHTING MATRIX (N,N)  
NN    STATE-CONTROL PRODUCT WEIGHTING MATRIX (N,NC)  
PCINV INVulnerable OF CONTROL WEIGHTING MATRIX (NC,NC)  
IOP1  SCALING PRINT OPTION: 0, NO PRINT; 1, PRINT  
IOP2  EIGENVECTOR PRINT OPTION: 0, NO PRINT; 1, PRINT  
N    NUMBER OF STATE VARIABLES  
NC  NUMBER OF CONTROL INPUTS  
N2  DIMENSION OF HAMILTONIAN MATRIX, 2 X N  
NMAX  MAXIMUM SIZE OF N  
NCMAX  MAXIMUM SIZE OF NC  
N2MAX  MAXIMUM SIZE OF N2
OUTPUTS:

- KC: CONTROL GAIN MATRIX (NC,N)
- SS: LQR RICCATI SOLUTION MATRIX (N,N)
- CR: VECTOR OF REAL PARTS OF EIGENVALUES OF AAA (N2)
- CI: VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N2)
- X: MODIFIED EIGENVECTOR MATRIX OF AAA (N2,N2)
- TS: SCALING TRANSFORMATION VECTOR OF AAA (N2)
- AAA: HAMILTONIAN MATRIX FOR LQR RICCATI EQUATION (N2,N2)

TEMPORARY STORAGE:

- XR: MATRIX (N2,N2)
- TT: MATRIX (N2,N2)
- EXT: MATRIX (N2,N2)
- AR: VECTOR (N2)
- AI: VECTOR (N2)
- IPER: INTEGER VECTOR (N2)
- IPERN: INTEGER VECTOR (N2)
- ADBLE: VECTOR (N X N)

SUBROUTINE COVAR (AA, BB, HH, CC, DOUT, QQ, PP, KC, N, NM, NC, NO, XX, YY, ZZ, UU, A, Q, WORK, NMAX, NMMAX, NCMAX, NOMAX)

SUBROUTINE COVAR SETS UP MATRICES FOR SUBROUTINE LAPNV (LYAPUNOV EQUATION) WHICH IS THEN CALLED TO OBTAIN STATE COVARIANCE MATRIX, XX, YY, KALMAN FILTER ERROR COVARIANCE PP, AND CONTROL GAINS KC ARE USED TO OBTAIN CONTROL COVARIANCE - UU, OUTPUT COVARIANCE - YY, AND MEASUREMENT COVARIANCE - ZZ. COVAR CALLS SUBROUTINES ARRAY, LAPNV, AND MATPRT. COVAR IS CALLED BY SUBROUTINE AES800.

INPUTS:

- AA: SYSTEM MATRIX (N,N)
- BB: CONTROL INPUT MATRIX (N,NC)
- HH: MEASUREMENT MATRIX (NM,N)
- CC: OUTPUT MATRIX (NO,N)
- DOUT: FEED FORWARD MATRIX (NO,NC)
- QQ: POWER SPECTRAL DENSITY MATRIX (N,N)
- PP: KALMAN FILTER ERROR COVARIANCE MATRIX (N,N)
- KC: CONTROL GAIN MATRIX (NC,N)
- N: NUMBER OF STATE VARIABLES
- NM: NUMBER OF MEASUREMENTS
- NC: NUMBER OF CONTROL INPUTS
- NO: NUMBER OF OUTPUTS
- NMAX: MAXIMUM SIZE OF N
- NMMAX: MAXIMUM SIZE OF NM
- NCMAX: MAXIMUM SIZE OF NC
- NOMAX: MAXIMUM SIZE OF NO

OUTPUTS:

- XX: STATE COVARIANCE MATRIX (N,N)
- YY: OUTPUT COVARIANCE MATRIX (NO,NO)
- ZZ: MEASUREMENT COVARIANCE MATRIX (NM,NM)
- UU: CONTROL COVARIANCE MATRIX (NC,NC)

TEMPORARY STORAGE:

- A: MATRIX (N,N)
- Q: MATRIX (N,N)
- WORK: VECTOR (N)

**************************************************************************
SUBROUTINE CTBL (B, CI, T, TINV, TINVB, EX1, ADBLE, LEX, MEX, N, NC, NMAX)

**************************************************************
SUBROUTINE CTBL COMPUTES THE (RELATIVE) CONTROLLABILITY OF A LINEAR SYSTEM DESCRIBED BY XDOT=A*X + B*U.
NOTE: FOR A COMPLEX EIGENVALUE PAIR, THE CORRESPONDING TWO COLUMN ELEMENTS IN TINVB ARE STORED AS MAGNITUDE AND ANGLE (IN DEGREES) RESPECTIVELY.
CTBL CALLS SUBROUTINES MATPRT AND MXINV. CTBL IS CALLED BY SUBROUTINE AES400.

INPUTS:

B SYSTEM INPUT MATRIX B (N,NC)
CI VECTOR OF IMAG PARTS OF THE EIGENVALUES (N)
(T OF MATRIX A)
T MODIFIED EIGENVECTOR MATRIX OF MATRIX A (N,N)
N NUMBER OF STATES
NC NUMBER OF INPUTS
NMAX MAXIMUM SIZE OF N

OUTPUTS:

TINV INVERSE OF MATRIX T (N,N)
TINVB CONTROL EFFECTIVENESS MATRIX (N,NC)
(EX1 TINV*B WHERE TINV IS IN MODIFIED FORM (N,NC)

TEMPORARY STORAGE:

ADBLE VECTOR OF LENGTH N X N
LEX INTEGER VECTOR (N)
MEX INTEGER VECTOR (N)

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

SUBROUTINE DANSKY (A, X, Y, Z, N, NMAX)

**************************************************************
SUBROUTINE DANSKY COMPUTES THE COEFFICIENTS OF THE CHARACTERISTIC EQUATION. DANSKY CALLS SUBROUTINE POLMPY. DANSKY IS CALLED BY SUBROUTINE BOLLIN.

INPUTS:

AS CHARACTERISTIC EQUATION MATRIX (N,N)
N ACTUAL SIZE OF MATRIX AS
NMAX MAXIMUM SIZE OF N

OUTPUTS:

Z CHARACTERISTIC EQUATION COEFFICIENT VECTOR (N)

TEMPORARY STORAGE:

X VECTOR (N)
Y VECTOR (N)

**************************************************************
SUBROUTINE DAVISO (A, B, C, N, NMAX, MC)

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

SUBROUTINE DAVISO TRANSFORMS X(DOT)=AX+BU,Y=C(TRANSPOSE)X USING
Z=TX SUCH THAT Y IS A STATE VARIABLE OF Z(DOT)=TAI(INVERSE)+TBu.
DAVISO DOES NOT CALL ANY SUBROUTINES. DAVISO IS CALLED BY
SUBROUTINE BOLLIN.

INPUTS:
A  SYSTEM MATRIX (N,N)
B  SYSTEM VECTOR (N)
C  SYSTEM VECTOR (N)
N  ACTUAL SIZE OF MATRIX A
NMAX MAXIMUM SIZE OF N

OUTPUTS:
A  TRANSFORMED MATRIX A (N,N)
B  TRANSFORMED VECTOR B (N)

TEMPORARY STORAGE:
MC INTEGER SCALAR

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

SUBROUTINE DSCA (A, R, CC, N, MS)

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

SUBROUTINE DSCA FORMS R = A + CC * I, FOR EITHER VECTOR OR MATRIX
IN VECTOR STORAGE MODE. DSCA DOES NOT CALL ANY SUBROUTINES.
DSCA IS CALLED BY SUBROUTINE LAPNV.

INPUTS:
A  INPUT MATRIX (N,N), OR INPUT VECTOR (N)
CC  CONSTANT
N  ACTUAL SIZE OF SUBSCRIPT(S) OF MATRIX (VECTOR) A
MS  DECISION VARIABLE, 2 = MATRIX, 0 OR 1 = VECTOR

OUTPUTS:
R  OUTPUT MATRIX (N,N), OR OUTPUT VECTOR (N)

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

SUBROUTINE DSCRT (DT, A, N, NMAX, ITIMES, EADT, INTGRL, C)

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

SUBROUTINE DSCRT CALCULATES EXP(A*DT) AND THE INTEGRAL
FROM 0 TO DT OF EXP(A*T). AFTER EACH TERM OF THE SERIES IS
MADE ON THE PERCENT CHANGE OCCURRING IN EACH TERM OF INTGRL.
WHEN ALL CHANGES ARE LESS THAN .00001, COMPUTATION IS STOPPED.
IF ITIMES=50 BEFORE CONVERGENCE, DSCRT PROMPTS THE USER AS TO
WHETHER TO COMPUTE MORE TERMS IN ORDER TO OBTAIN CONVERGENCE.
DSCRT DOES NOT CALL ANY SUBROUTINES. DSCRT IS CALLED BY
SUBROUTINE AES600.

42
INPUTS:

DT  TIME STEP
A INPUT MATRIX (N,N)
N ACTUAL SIZE OF MATRIX A
NMAX MAXIMUM SIZE OF N

OUTPUTS:

ITIMES NO. OF TERMS IN SERIES EXPANSION
EADT  EXP(A*DT) (N,N)
INTGRL INTEGRAL OF EXP(A*T) FROM T=0 TO T=DT (N,N)

TEMPORARY STORAGE:

C VECTOR (N)

SUBROUTINE EGCK (AAA, X, CPR, CPI, EX1, EX2, PLAM, N, NMAX)

SUBROUTINE EGCK PERFORMS THE EIGENVALUE AND EIGENVECTOR CHECK. IT
FORMS (AAA * X) AND (X * LAMBDA). EGCK DOES NOT CALL ANY
SUBROUTINE. EGCK IS CALLED BY SUBROUTINE RICSS.

INPUTS:

AAA ORIGINAL MATRIX FOR WHICH EIGENVALUES
AND EIGENVECTORS WERE FOUND (N,N)
X MODIFIED EIGENVECTOR MATRIX (N,N)
CPR VECTOR OF REAL EIGENVALUES (N)
CPI VECTOR OF IMAGINARY EIGENVALUES (N)
N ACTUAL SIZE OF MATRIX AAA
NMAX MAXIMUM SIZE OF N

OUTPUTS:

EX1 AAA * X MATRIX (N,N)
EX2 X * LAMBDA MATRIX (N,N)

TEMPORARY STORAGE:

PLAM MATRIX (N,N)

SUBROUTINE EGVCTR (MA, CPR, CPI, X, N2, TT, EXT, AR, AI, IPERN,
1 IPER, IOP2, N2MAX, ISEL, IHALF)

SUBROUTINE EGVCTR OBTAINS THE N2 BY N2 MODIFIED EIGENVECTOR MATRIX
X OF MATRIX AAA USING THE INVERSE ITERATION ALGORITHM. (THE
EIGENVALUES OF AAA SHOULD HAVE BEEN PREVIOUSLY GENERATED USING
SUBROUTINE EIGQR AND STORED IN CPR AND CPI.) IEND SPECIFIES THE
NUMBER OF PASSES THRU THE INVERSE ITERATION ALGORITHM.
IF ISEL=0, ALL EIGENVECTORS ARE OBTAINED.
IF ISEL[0], ONLY THE ISEL (AND NEXT ONE IF A COMPLEX PAIR) IS
OBTAINED.
IF ISEL 0, THE ISELTH VECTOR IS PRINTED OUT AFTER EACH ITER.
IF IHALF=1, THE FIRST N2/2 VECTORS ARE OBTAINED.
IF IHALF .NE. 1, ALL VECTORS ARE OBTAINED.
EGVCTR CALLS SUBROUTINES ARRAY, FACTR, MATPRT, AND PRMUTE. EGVCTR
IS CALLED BY SUBROUTINES AES400, AES800, AND RICSS.
SUBROUTINE EIGEN (A, EIGR, EIGI, EX1, SSS, S, IA, IB, LEX, MEX, IBL, IC, EX4, N, NMAX)

SUBROUTINE EIGEN OBTAINS THE EIGENVALUES (EIGR AND EIGI) OF N X N MATRIX A BY FIRST REDUCING (IF A IS REDUCIBLE), THEN SCALING, HESSENBURG TRANSFORMING AND FINALLY APPLYING THE 'QR' ALGORITHM. EIGEN CALLS SUBROUTINES SCALEA, CONDI, ARRAY, HSBG, AND EIGQR. EIGEN IS CALLED BY SUBROUTINES AES400 AND AES800.

INPUTS:
A   MATRIX TO GET EIGENVALUES FOR (N,N)
N   ACTUAL SIZE OF MATRIX A
NMAX MAXIMUM SIZE OF N

OUTPUTS:
EIGR VECTOR OF REAL PARTS OF EIGENVALUES (N)
EIGI VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N)
S   MATRIX A IN REDUCED AND SCALED FORM (N,N)
LEX BLOCK-DIAGONALIZING PERMUTATION INTEGER VECTOR (N)
MEX INTEGER VECTOR OF SIZES OF EACH IRREDUCIBLE BLOCK (N)
EX4 VECTOR OF DIAGONAL ELEMENTS OF DIAGONAL SCALING MATRIX (N)

TEMPORARY STORAGE:
EX1 MATRIX (N,N)
SSS MATRIX (N,N)
IA INTEGER VECTOR (N)
IB INTEGER VECTOR (N)
IBL INTEGER VECTOR (N)
IC INTEGER VECTOR (N)
SUBROUTINE EIGQR (XR, N2, CR, CI, IOP, N2MAX)

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

SUBROUTINE EIGQR COMPUTES THE EIGENVALUES OF MATRIX XR USING THE QR ALGORITHM. THIS MATRIX MUST BE IN UPPER HESSENBerg FORM. THE MAXIMUM NUMBER OF QR ITERATIONS USED IN FINDING ANY ONE EIGQR DOES NOT CALL ANY SUBROUTINES. EIGQR IS CALLED BY SUBROUTINES EIGEN, RICSS, AND ZEROES.

INPUTS:

XR   MATRIX (IN UPPER HESSENBerg FORM) FOR WHICH EIGENVALUES ARE TO BE FOUND (N2,N2)
N2   ACTUAL SIZE OF MATRIX XR
IOP  PRINT OPTION
     IOP=0, THE EIGENVALUES ARE WRITTEN ON UNIT 06
     IOP=0, THE EIGENVALUES ARE WRITTEN ON UNIT 06 AND ON UNIT 02 (TERMINAL)
N2MAX MAXIMUM SIZE OF N2

OUTPUTS:

CR   VECTOR OF REAL PARTS OF EIGENVALUES (N2)
CI   VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N2)

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

SUBROUTINE ESTMAT (AA, HH, QQ, RRINV, KE, PP, CR, CI, X, TS, XR, 1 TT, AAA, EXT, AR, AI, IPER, IPERN, AOBLE, N, NM, N2, IOP1, IOP2, 2 NMAX, NMMAX, N2MAX)

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

SUBROUTINE ESTMAT SOLVES THE OPTIMAL LINEAR STATE ESTIMATION PROBLEM. IT SETS UP AN N2 BY N2 MATRIX AAA. USING MATRICES AA, HH, QQ, AND RRINV. ESTMAT OBTAINS THE KALMAN FILTER ERROR COVARIANCE, PP, AND THEN COMPUTES THE KALMAN FILTER GAINS, KE. ESTMAT CALLS SUBROUTINES MATPRT AND RICSS. ESTMAT IS CALLED BY SUBROUTINE AESBOO.

INPUTS:

AA   SYSTEM MATRIX (N,N)
HH   MEASUREMENT MATRIX (NM,N)
QQ   POWER SPECTRAL DENSITY MATRIX (N,N)
     (OF PLANT DISTURBANCE)
RRINV INVERSE OF POWER SPECTRAL DENSITY MATRIX (NM,NM)
     (OF MEASUREMENT NOISE)
IOP1  SCALING PRINT OPTION: 0, NO PRINT; 1, PRINT
IOP2  EIGENVECTOR PRINT OPTION: 0, NO PRINT; 1, PRINT
N   NUMBER OF STATE VARIABLES
NM   NUMBER OF MEASUREMENTS
N2   DIMENSION OF HAMILTONIAN MATRIX, 2 X N
NMAX MAXIMUM SIZE OF N
NMMAX MAXIMUM SIZE OF NM
N2MAX MAXIMUM SIZE OF N2

OUTPUTS:

KE   KALMAN FILTER GAIN MATRIX (N,NM)
PP   KALMAN FILTER ERROR COVARIANCE MATRIX (N,N)
CR VECTOR OF REAL PARTS OF EIGENVALUES (N2)
    (OF AAA)
CI VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N2)
    (OF AAA)
X MODIFIED EIGENVECTOR MATRIX OF AAA (N2,N2)
TS SCALING TRANSFORMATION VECTOR OF AAA (N2)
AAA HAMILTONIAN MATRIX FOR KALMAN FILTER RICCATI
    EQUATION (N2,N2)
TEMPORARY STORAGE:
XR MATRIX (N2,N2)
TT MATRIX (N2,N2)
EXT MATRIX (N2,N2)
AR VECTOR (N2)
AI VECTOR (N2)
IPER INTEGER VECTOR (N2)
IPERN INTEGER VECTOR (N2)
ADBLE VECTOR (N X N)

***********************************************************************
SUBROUTINE FACTR (A, PER, N, IA, IER)
***********************************************************************
SUBROUTINE FACTR FORMS THE LOWER AND UPPER TRIANGULAR MATRICES OF
INPUT MATRIX A, SUCH THAT UPPER * LOWER = A.
FACTR DOES NOT CALL ANY SUBROUTINES. FACTR IS CALLED BY
SUBROUTINE EGVCTR.
INPUTS:
A INPUT MATRIX (N,N)
N ACTUAL SIZE OF MATRIX A
IA SAME AS N
OUTPUTS:
A INPUT MATRIX IN UPPER AND LOWER TRIANGULAR FORM (N,N)
PER TRANSPOSITION VECTOR FOR MATRIX A (N)
IER ERROR OPTION.
   IF IER .NE. 0, FACTR IS WRONG
   IF IER .EQ. 0, FACTR HAS WORKED CORRECTLY
***********************************************************************
SUBROUTINE FRPOLY (Zl, Z2, DD, HZ, AMP, PHA, N)
***********************************************************************
SUBROUTINE FRPOLY EVALUATES TRANSFER FUNCTION Z2(S) / Z1(S) FOR
S = 6.28 * HZ * J. FRPOLY DOES NOT CALL ANY SUBROUTINES. FRPOLY
IS CALLED BY SUBROUTINE FRQP.
INPUTS:
Z1 DENOMINATOR COEFFICIENT VECTOR
Z2 NUMERATOR COEFFICIENT VECTOR
DD DOUT OR 0.0
HZ FREQUENCY
N SIZE OF COEFFICIENT VECTORS
OUTPUTS:
G COMPLEX TRANSFER FUNCTION
SUBROUTINE FRQP (Z1, Z2, DD, N, FI, DELF, IF, FREQ, AMP, PHASE,
1 ISPACE, TSFTR)

SUBROUTINE FRQP GENERATES FREQUENCY RESPONSE AMP. AND PHASE, GIVEN
TRANSFER FUNCTION NUMERATOR AND DENOMINATOR POLYNOMIAL
COEFFICIENTS (GENERATED BY SUBROUTINE BOLLIN).
FRQP CALLS SUBROUTINE FRPOLY, WHICH COMPUTES AMPLITUDE AND PHASE.
FRQP IS CALLED BY SUBROUTINE FRSPNS.

INPUTS:
Z1  DENOMINATOR POLYNOMIAL COEFFICIENT VECTOR
Z2  NUMERATOR POLYNOMIAL COEFFICIENT VECTOR
DD  DOUT OR 0.0
N   SIZE OF COEFFICIENT VECTORS
TSFTR TIME SCALE FACTOR
FI  INITIAL FREQUENCY
DELF SPACING BETWEEN FREQUENCY POINTS
IF  NUMBER OF DESIRED POINTS TO BE GENERATED
ISPACE CONTROLS FREQUENCY OF PRINTOUT FOR FREQ,
AMP AND PHASE

OUTPUTS:
FREQ  FREQUENCY VECTOR
AMP  AMPLITUDE VECTOR
PHASE PHASE VECTOR

SUBROUTINE FRSPNS (A, B, C, DD, IOUT, JIN, N, TSFTR, DXM1, DXV1,
1 DXV2, DXV3, EXM1, EXV1, EXV2, NMAX, NOMAX, DDCOF, DNCOF, FI,
2 DELF, IF, FREQ, AMP, PHASE, ISPACE, IPRNT)

SUBROUTINE FRSPNS COMPUTES THE FREQUENCY RESPONSE OF THE IOUT
OUTPUT TO THE JIN INPUT OF THE SYSTEM XDOT = A*X + B*U - Y = C*X
FRSPNS CALLS SUBROUTINES BOLLIN AND FRQP. FRSPNS IS CALLED BY
SUBROUTINE AES500.

INPUTS:
A   SYSTEM MATRIX (N,N)
B   SYSTEM MATRIX (N,NC)
C   SYSTEM MATRIX (NO,N)
DD  DOUT OR 0.0
IOUT INDEX OF OUTPUT
JIN INDEX OF INPUT
N   ACTUAL SIZE OF MATRIX A
TSFTR TIME SCALE FACTOR
NMAX MAXIMUM SIZE OF N
NOMAX MAXIMUM SIZE OF NO
FI  INITIAL FREQUENCY
DELF SPACING BETWEEN FREQUENCY POINTS
IF  NUMBER OF DESIRED POINTS TO BE GENERATED
ISPACE CONTROLS FREQUENCY OF PRINTOUT FOR FREQ,
AMP AND PHASE
IPRNT PRINT OPTION, 0 IF STANDARD, 1 IF EXTENDED
OUTPUTS:

DXV3  NUMERATOR COEFFICIENTS (N)
EXM1  A * TSFTR (N,N)
EXV1  JINTH ROW OF B * TSFTR (N)
EXV2  IOUTh COLUMN OF C (N)
DDCF  DENOMINATOR COEFFICIENTS (N)
DNCF  NUMERATOR COEFFICIENTS (N)
FREQ  FREQUENCY VECTOR (500)
AMP   AMPLITUDE VECTOR (500)
PHASE PHASE VECTOR (500)

TEMPORARY STORAGE:

DXM1  MATRIX (N,N)
DXV1  VECTOR (N)
DXV2  VECTOR (N)

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

SUBROUTINE GAIN (AA, BB, CC, DD, JJ, N, NZ, GAYN, EX1, EX4, -
1 NMAX, NMMAX)

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

SUBROUTINE GAIN IS A COMPANION TO SUBROUTINE ZEROES. GAIN
COMPUTES THE GAIN OF THE TRANSFER FUNCTION RELATING INPUT JJ AND
OUTPUT I1 OF THE FOLLOWING NTH ORDER SYSTEM.
(IN STATE VARIABLE FORM):
XDOT = AA * X + BB * U
Y = CC * X + DD * U
GAIN DOES NOT CALL ANY SUBROUTINES. GAIN IS CALLED BY SUBROUTINE
AES700.

INPUTS:

AA  SYSTEM MATRIX (N,N)
BB  CONTROL INPUT MATRIX (N,NUMBER OF POSSIBLE
    INPUTS)
CC  OUTPUT MATRIX (NUMBER OF POSSIBLE OUTPUTS,N)
DD  SCALAR RELATING U(JJ) TO Y(I1)
II  INDEX OF OUTPUT Y
JJ  INDEX OF INPUT U
N  ACTUAL NUMBER OF STATES
NZ  NUMBER OF NUMERATOR ZEROES IN TRANSFER FUNCTION
    (OBTAINED USING SUBROUTINE ZEROES)
NMAX  MAXIMUM SIZE OF N
NMMAX  MAXIMUM NUMBER OF OUTPUTS

OUTPUTS:

GAYN  TRANSFER FUNCTION GAIN

TEMPORARY STORAGE:

EX1  MATRIX (N,N)
EX4  VECTOR (N)

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

SUBROUTINE HSBG (N, A, IN)

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

SUBROUTINE HSBG REDUCES A MATRIX INTO UPPER ALMOST TRIANGULAR
FORM. HSBG DOES NOT CALL ANY SUBROUTINES. HSBG IS CALLED BY
SUBROUTINES EIGEN, RICSS, AND ZEROES.
INPUTS:

- \( N \) - Actual size of matrix \( A \)
- \( A \) - Input matrix \((N,N)\)
- \( I \) - Maximum size of matrix \( A \) in the calling program;
  \( I = N \), when matrix \( A \) is in vector storage mode.

OUTPUTS:

- \( A \) - Output matrix \((N,N)\)

Subroutine ICRSP computes multiple initial condition responses of the system: \( \dot{x} = Ax \) and \( y = Cx \) by solving the difference equation: \( x_{new} = E1 x_{old} \).

This subroutine requires that the state transition matrix, \( \exp(A*DT) \), be supplied as input matrix "E1". Desired initial condition magnitudes are supplied as vector 'AMPIN' and the selected by appropriately defining elements of the matrix 'ICMTX'. ICRSP calls plotting subroutines only. ICRSP is called by subroutine AES600.

INPUTS:

- \( E1 \) - State transition, \( \exp(A*DT) \), matrix \((N,N)\)
- \( C \) - System output matrix \((N_{out}, N)\)
- \( ICMTX \) - Matrix of zeroes and ones \((N, N_{out})\).
  Ones are placed in selected matrix positions to indicate the initial condition responses desired.
  The first index is 'state', and the second is 'output'. Thus subroutine ICRSP may calculate as many as \( N * N_{out} \) initial condition responses.
- \( AMPIN \) - Vector of input initial condition amplitudes \((N)\)
- \( DT \) - Time step
- \( TTI \) - Plot title \((12)\)
- \( TTOP \) - Plot title \((12)\)
- \( TYTIT \) - Y axis title \((4)\)
- \( N \) - Actual size of state transition matrix
- \( N_{out} \) - Actual number of possible outputs
- \( N_{max} \) - Maximum size of \( N \)
- \( N_{out_{max}} \) - Maximum size of \( N_{out} \)
- \( ITRMX \) - Number of desired time response points
- \( IP \) - Plot entity index (used by PLOTTSUBS only)
  Increases by one for each frame
- \( NAME \) - Name of plot dataset \((9)\) (used by PLOTTSUBS only)
  (partitioned dataset that holds plot entities)
- \( IONPLT \) - 0, if offline plots
  1, if online plots

OUTPUTS:

- \( TIME \) - Vector of time points \((ITRMX)\)
  (single precision)
- \( TYOUT \) - Matrix of output transient responses for any one specific initial condition. \((ITRMX, N_{out})\)
  (single precision)
- \( IP \) - Plot entity index (used by PLOTTSUBS only)
  Increases by one for each frame

Temporary storage:

- \( XNEW \) - Vector \((N)\)
- \( XOLD \) - Vector \((N)\)
- \( ITEXT \) - Integer vector \((N)\)

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

49
SUBROUTINE LAPNV (A, X, B, QIN, NIN, WORK)

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

SUBROUTINE LAPNV SOLVES THE LYAPUNOV EQUATION,
X*A' + A*X*B = 0
WHERE A' IS A TRANSPOSE,
A, B, AND X ARE ALL NXN MATRICES IN VECTOR STORAGE MODE, B IS
SYMMETRIC ON INPUT, AND X IS SYMMETRIC ON OUTPUT.
STEP 1 CALCULATE X(0) = A**B*A
STEP 2 THE EXACT SOLUTION X IS THE LIMIT OF THE SEQUENCE X(M)
WHERE THE M REFERS TO THE M-TH TERM OF THE SEQUENCE.
COMPUTE EACH TERM X(M+1) RECURSIVELY, BASED ON X(M) AS FOLLOWS,
X(M+1) = X(M) + U(M)*X(M) * U'(M)
U(0) = (Q*I-A')**(1) * (Q*I+A')
U(M) = U(0)**(2*M)
LAPNV CALLS SUBROUTINES DSCA, MXINV, MXMT, MXTRA, AND MXADD.
LAPNV IS CALLED BY SUBROUTINES COVAR AND LYPCK.

INPUTS:
A  LYAPUNOV EQUATION MATRIX (NIN,NIN)
B  LYAPUNOV EQUATION MATRIX (NIN,NIN)
QIN  CONVERGENCE FACTOR (TYPICALLY .1)
NIN  ACTUAL SIZE OF MATRIX A
WORK(1)  CONVERGENCE CHECK CRITERION (TYPICALLY 1.E-6)

OUTPUTS:
X  OUTPUT MATRIX (NIN,NIN)

TEMPORARY STORAGE:
WORK  VECTOR (2 X NIN X NIN)

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

SUBROUTINE LOGSET (TMIN, TMAX, TNARR, INT)

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

SUBROUTINE LOGSET CALCULATES THE DECADES NECESSARY TO INCLUDE THE
MINIMUM AND MAXIMUM OF THE DATA TO BE PLOTTED.
LOGSET DOES NOT CALL ANY SUBROUTINES.

INPUTS:
TNARR  ACTUAL DATA MINIMUM AND MAXIMUM (2)
LOCATION 1 HOLDS THE MINIMUM
LOCATION 2 HOLDS THE MAXIMUM

OUTPUTS:
TMIN  MINIMUM DECADE FOR PLOTTING
TMAX  MAXIMUM DECADE FOR PLOTTING
INT  NUMBER OF INTERVALS FOR LOG AXIS

*****************************************************************************
SUBROUTINE LYPCK (A, Q, XHT, E, R, EX1, WORK, N, NMAX)

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

SUBROUTINE LYPCK COMPUTES THE RESIDUAL AND ERROR MATRICES ASSOCIATED WITH THE LYAPUNOV EQ.

\[ A^*X + X^*A^{**T} + Q = 0 \]

WHERE

- SOLUTION -- XHT
- RESIDUAL -- R = A^*XHT + XHT^*A^{**T} + Q
- ERROR -- E = XHT - X

WHERE

\[ A^*E + E^*A^{**T} - R = 0 \]

WORK VECTOR 'WORK' MUST BE DIMENSIONED \( | = 2 \times N^2 \) BEFORE COMPUTING E, R IS SYMMETRIZED.

THE TRACES OF XHT, R, & E ARE PRINTED OUT; ALSO THE NORMALIZED ERROR INDEX, TR(E)/TR(XHT), AND THE NORMALIZED DIAGONAL ELEMENTS OF THE ERROR MATRIX ARE PRINTED OUT.

LYPCK CALLS SUBROUTINES MATPRT, ARRAY, AND LAPNV. LYPCK IS CALLED BY SUBROUTINE AES800.

INPUTS:

- A: LYAPUNOV EQUATION MATRIX (N,N)
- Q: LYAPUNOV EQUATION MATRIX (N,N)
- XHT: LYAPUNOV SOLUTION MATRIX (N,N)
- N: ACTUAL SIZE OF MATRIX A
- NMAX: MAXIMUM SIZE OF N

OUTPUTS:

- E: ERROR MATRIX (N,N)
- R: RESIDUAL MATRIX (N,N)

TEMPORARY STORAGE:

- EX1: MATRIX (N,N)
- WORK: VECTOR (2 x N x N)

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

SUBROUTINE MATCHG

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

SUBROUTINE MATCHG IS USED FOR CHANGING MATRICES AND DIMENSIONS USING NAMELIST 'MATDAT'. THE CHANGES IN MATDAT ARE READ IN FROM THE TERMINAL. DATA IS TRANSFERRED TO PROGRAM AESOP VIA COMMONS 'ABETC' AND 'DIMS'...

MATCHG DOES NOT CALL ANY SUBROUTINES. MATCHG IS CALLED BY SUBROUTINE AES200.

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

SUBROUTINE MATIN

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

SUBROUTINE MATIN IS USED FOR INPUTTING MATRICES AND DIMENSIONS FOR A 'SMALL' LQR/KALMAN FILTER PROBLEM TESTCASE. IT ALSO PRINTS OUT THE REFS NAMELIST. DATA IS PROVIDED FOR A 3RD ORDER TEST CASE HAVING TWO CONTROLS AND TWO SET-POINT OUTPUTS, TWO DISTURBANCES AND ONE NOISY MEASUREMENT. MATIN CALLS SUBROUTINE MATPRT. MATIN IS CALLED BY SUBROUTINE AES200.

*******************************************************************************
SUBROUTINE  MATPRT (A, N, M, NMAX)

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

SUBROUTINE MATPRT PRINTS MATRIX A TEN COLUMNS PER PAGE. THE DEVICE ON WHICH THE PRINTING TAKES PLACE IS CONTROLLED BY 'IUNIT'. IUNIT=2---TERMINAL, IUNIT=6---LINEPRINTER. MATPRT DOES NOT CALL ANY SUBROUTINES. MATPRT IS CALLED BY SUBROUTINES AES200, AES300, AES400, AES600, AES700, AES800, COVAR, CTBL, EGCK, LYPCK, MATIN, MATRD, MODSHP, OBSBL, RESI, RICCHK, RICSS, AND UNRML.

INPUTS:

A         MATRIX TO BE PRINTED (N,M)
N         NUMBER OF ROWS IN MATRIX A
M         NUMBER OF COLUMNS IN MATRIX A
NMAX      MAXIMUM SIZE OF N

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

SUBROUTINE MATRD

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

SUBROUTINE MATRD IS USED FOR INPUTTING MATRICES, DIMENSIONS, AND REFS USING NAMELISTS 'MATDAT' AND 'REFS'. MATDAT AND REFS ARE READ IN FROM UNIT 33. DATA IS TRANSFERRED TO PROGRAM AESOP VIA COMMONS 'COM1', 'ABETC', 'DIMS', 'DIMS2', AND 'REFCOM'. MATRD CALLS SUBROUTINE MATPRT. MATRD IS CALLED BY SUBROUTINE AES200.

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

SUBROUTINE MODSHP (A, B, CI, N, NMAX)

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

SUBROUTINE MODSHP CALCULATES MODE SHAPES IN MAGNITUDE AND ANGLE (DEGREE) FORM. MODSHP CALLS SUBROUTINE MATPRT. MODSHP IS CALLED BY SUBROUTINES AES400, AES800, AND RICSS.

INPUTS:

A         MODIFIED EIGENVECTOR MATRIX (N,N)
CI         VECTOR OF IMAGINARY EIGENVALUES (N)
N         ACTUAL NUMBER OF EIGENVALUES
NMAX      MAXIMUM SIZE OF N

OUTPUTS:

B         MODESHAPE IN MAGNITUDE AND ANGLE FORM (N,N)

**************************************************************************************************
SUBROUTINE MXADD (A, B, R, N, M)

SUBROUTINE MXADD ADDS TWO IDENTICALLY SIZED MATRICES TO FORM A RESULTANT MATRIX. R CAN BE THE SAME AS A OR B IN THE CALLING PROGRAM. MXADD DOES NOT CALL ANY SUBROUTINES. MXADD IS CALLED BY SUBROUTINE LAPNV.

INPUTS:
A FIRST INPUT MATRIX (N,M)
B SECOND INPUT MATRIX (N,M)
N NUMBER OF ROWS IN MATRICES A, B, AND R
M NUMBER OF Columns IN MATRICES A, B, AND R

OUTPUTS:
R OUTPUT MATRIX (N,M)

SUBROUTINE MXINV (A, N, D, L, M)

SUBROUTINE MXINV INVERTS A DOUBLE PRECISION MATRIX IN VECTOR STORAGE MODE BY USING THE GAUSS-JORDAN METHOD. THE DETERMINANT IS CALCULATED BUT IS ZERO IF THE MATRIX BEING INVERTED IS SINGULAR. MXINV DOES NOT CALL ANY SUBROUTINES. MXINV IS CALLED BY SUBROUTINES AES300, AES800, CTBL, LAPNV, AND RICSS.

INPUTS:
A MATRIX TO BE INVERTED, VECTOR STORAGE MODE (N,N)
N ACTUAL SIZE OF MATRIX A

OUTPUTS:
A MATRIX INVERTED FORM, VECTOR STORAGE MODE (N,N)
D SCALAR DETERMINANT (ZERO IF MATRIX A IS SINGULAR)

TEMPORARY STORAGE:
L INTEGER VECTOR (N)
M INTEGER VECTOR (N)

SUBROUTINE MXMLT (A, B, R, N, L, M)

SUBROUTINE MXMLT MULTIPLIES TWO MATRICES IN VECTOR STORAGE MODE TO FORM A RESULTANT MATRIX IN VECTOR STORAGE MODE. MXMLT DOES NOT CALL ANY SUBROUTINES. MXMLT IS CALLED BY SUBROUTINE LAPNV.

INPUTS:
A FIRST INPUT MATRIX (N,L)
B FIRST INPUT MATRIX (L,M)
N NUMBER OF ROWS IN MATRIX A
L NUMBER OF Columns IN MATRIX A AND ROWS IN MATRIX B
M NUMBER OF Columns IN MATRIX B

OUTPUTS:
R OUTPUT MATRIX (N,M)
SUBROUTINE MXTRA (A, R, N, M)

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

SUBROUTINE MXTRA TRANSPOSES AN N BY M MATRIX A IN VECTOR STORAGE
MODE TO FORM AN M BY N MATRIX R IN VECTOR STORAGE MODE. MXTRA
DOES NOT CALL ANY SUBROUTINES. MXTRA IS CALLED BY SUBROUTINE
LAPNV.

INPUTS:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>MATRIX TO BE TRANSPOSED (N,M)</td>
</tr>
<tr>
<td>N</td>
<td>NUMBERS OF ROWS IN MATRIX A</td>
</tr>
<tr>
<td></td>
<td>AND COLUMNS IN MATRIX R</td>
</tr>
<tr>
<td>M</td>
<td>NUMBERS OF COLUMNS IN MATRIX A</td>
</tr>
<tr>
<td></td>
<td>AND ROWS IN MATRIX R</td>
</tr>
</tbody>
</table>

OUTPUTS:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>RESULTANT MATRIX (M,N)</td>
</tr>
</tbody>
</table>

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

SUBROUTINE NRML (A, B, C, H, Q, RINV, D, DOUT, CSP, N, NC, NO, NM, |
1 ND, NMAX, NCMAX, NOMAX, NMMAX, FL34)

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

SUBROUTINE NRML READS FOUR NORMALIZATION VECTORS FROM NAMELIST
NRMS AND NORMALIZES THE A, B, C, H, Q, RINV, D, DOUT, AND CSP
MATRICES. THE SYSTEM THUS REPRESENTED IS DEFINED BY NORMALIZED
STATE, CONTROL, OUTPUT, MEASUREMENT, AND SET POINT VECTORS.
THE NORMALIZATION VECTORS ARE TRANSFERRED TO THE MAIN PROGRAM
THROUGH COMMON 'NORMS'. NRML DOES NOT CALL ANY SUBROUTINES.
NRML IS CALLED BY SUBROUTINE AES400.

INPUTS:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>UN-NORMALIZED SYSTEM MATRIX (N,N)</td>
</tr>
<tr>
<td>B</td>
<td>UN-NORMALIZED CONTROL INPUT MATRIX (N,NC)</td>
</tr>
<tr>
<td>C</td>
<td>UN-NORMALIZED OUTPUT MATRIX (NO,N)</td>
</tr>
<tr>
<td>H</td>
<td>UN-NORMALIZED MEASUREMENT MATRIX (NM,N)</td>
</tr>
<tr>
<td>Q</td>
<td>UN-NORMALIZED POWER SPECTRAL DENSITY MATRIX OF</td>
</tr>
<tr>
<td></td>
<td>PLANT DISTURBANCE (N,N)</td>
</tr>
<tr>
<td>RINV</td>
<td>UN-NORMALIZED INVERSE OF POWER SPECTRAL DENSITY</td>
</tr>
<tr>
<td></td>
<td>MATRIX OF MEASUREMENT NOISE (NM,NM)</td>
</tr>
<tr>
<td>D</td>
<td>UN-NORMALIZED DISTURBANCE INPUT MATRIX (N,ND)</td>
</tr>
<tr>
<td>DOUT</td>
<td>UN-NORMALIZED FEED FORWARD MATRIX FOR</td>
</tr>
<tr>
<td></td>
<td>NON-ZERO SET POINT REGULATOR (NO,NC)</td>
</tr>
<tr>
<td>CSP</td>
<td>UN-NORMALIZED SET POINT OUTPUT MATRIX (NC,N)</td>
</tr>
<tr>
<td>N</td>
<td>ACTUAL NUMBER OF STATES</td>
</tr>
<tr>
<td>NC</td>
<td>ACTUAL NUMBER OF CONTROL INPUTS</td>
</tr>
<tr>
<td>NO</td>
<td>ACTUAL NUMBER OF OUTPUTS</td>
</tr>
<tr>
<td>NM</td>
<td>ACTUAL NUMBER OF MEASUREMENTS</td>
</tr>
<tr>
<td>ND</td>
<td>ACTUAL NUMBER OF DISTURBANCE INPUTS</td>
</tr>
<tr>
<td>NMAX</td>
<td>MAXIMUM SIZE OF N</td>
</tr>
<tr>
<td>NCMAX</td>
<td>MAXIMUM SIZE OF NC</td>
</tr>
<tr>
<td>NOMAX</td>
<td>MAXIMUM SIZE OF NO</td>
</tr>
<tr>
<td>NMMAX</td>
<td>MAXIMUM SIZE OF NM</td>
</tr>
<tr>
<td>FL34</td>
<td>LOGICAL VARIABLE, ON INPUT</td>
</tr>
<tr>
<td></td>
<td>TRUE, NORMALIZATION VECTOR INFORMATION</td>
</tr>
<tr>
<td></td>
<td>(NAMELIST NRMS) HAS ALREADY BEEN READ IN</td>
</tr>
<tr>
<td></td>
<td>FALSE, NORMALIZATION VECTOR INFORMATION</td>
</tr>
<tr>
<td></td>
<td>(NAMELIST NRMS) NEEDS TO BE READ IN</td>
</tr>
</tbody>
</table>
OUTPUTS:
A: NORMALIZED SYSTEM MATRIX (N,N)
B: NORMALIZED CONTROL INPUT MATRIX (N,NC)
C: NORMALIZED OUTPUT MATRIX (NO,N)
H: NORMALIZED MEASUREMENT MATRIX (NM,N)
Q: NORMALIZED POWER SPECTRAL DENSITY MATRIX OF
   PLANT DISTURBANCE (N,N)
R_INV: NORMALIZED INVERSE OF POWER SPECTRAL DENSITY
   MATRIX OF MEASUREMENT NOISE (NM,NM)
D: NORMALIZED DISTURBANCE INPUT MATRIX (N,ND)
DOUT: NORMALIZED FEED FORWARD MATRIX FOR
      NON-ZERO SET POINT REGULATOR (NO,NC)
CSP: NORMALIZED SET POINT OUTPUT MATRIX (NC,N)
FL34: LOGICAL VARIABLE, ON OUTPUT SET TO
     TRUE IF NAMELIST NRMS HAS BEEN READ IN

SUBROUTINE OBSBL (H, T, CI, HT, EX1, N, NR, NRMAX, NMAX)

SUBROUTINE OBSBL COMPUTES THE OBSERVABILITY MATRIX HT FOR THE
LINEAR SYSTEM DESCRIBED BY XDOT=A*X + B*U, AND Y=H*X.
NOTE: FOR A COMPLEX EIGENVALUE PAIR, THE CORRESPONDING TWO COLUMN
ELEMENTS IN HT ARE STORED AS MAGNITUDE AND ANGLE (IN DEGREES)
RESPECTIVELY.
OBSBL CALLS SUBROUTINE MATPRT. OBSBL IS CALLED BY SUBROUTINE
AES400.

INPUTS:
H: SYSTEM OUTPUT MATRIX (NR,N)
T: MODIFIED EIGENVECTOR MATRIX OF MATRIX A (N,N)
CI: VECTOR OF IMAG PARTS OF THE EIGENVALUES (N)
N: ACTUAL NUMBER OF COLUMNS IN MATRIX H
NR: ACTUAL NUMBER OF ROWS IN MATRIX H
NRMAX: MAXIMUM SIZE OF NR
NMAX: MAXIMUM SIZE OF N

OUTPUTS:
HT: OBSERVABILITY MATRIX (NR,N)
   (IN MAGNITUDE AND PHASE ANGLE FORM)
EX1: OBSERVABILITY MATRIX (NR,N)

SUBROUTINE ORDER (CR, CI, NE, EPS)

GIVEN A SET OF NE EIGENVALUES, SYMMETRICALLY LOCATED WITH RESPECT
TO THE IMAGINARY AXIS, SUBROUTINE ORDER PLACES ONES WITH POSITIVE
REAL PARTS IN FIRST NE/2 LOCATIONS. CORRESPONDING SYMMETRIC
EIGENVALUES WITH NEGATIVE REAL PARTS ARE PUT IN LOCATIONS
NE/2 + 1 THROUGH NE. EPS IS THE CONVERGENCE CRITERION USED IN
DETERMINING IF A PAIR OF EIGENVALUES ARE SYMMETRIC. ORDER DOES
NOT CALL ANY SUBROUTINES. ORDER IS CALLED BY SUBROUTINE RICSS.
INPUTS:

- CR VECTOR OF REAL PARTS OF EIGENVALUES UNORDERED (NE)
- CI VECTOR OF IMAGINARY PARTS OF EIGENVALUES UNORDERED (NE)
- NE NUMBER OF EIGENVALUES
- EPS CRITERION FOR SYMMETRY

OUTPUTS:

- CR VECTOR OF REAL PARTS OF EIGENVALUES ORDERED (NE)
- CI VECTOR OF IMAGINARY PARTS OF EIGENVALUES ORDERED (NE)

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

SUBROUTINE POLMPY (X, Y, Z, NX, NY, NZ)

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

SUBROUTINE POLMPY MULTIPLIES X*Y=Z (THE LEADING POLYNOMIAL COEFFICIENT IS ASSUMED TO BE UNITY). POLMPY DOES NOT CALL ANY SUBROUTINES. POLMPY IS CALLED BY SUBROUTINE DANSKY.

INPUTS:

- X POLYNOMIAL COEFFICIENT VECTOR (NX)
- Y POLYNOMIAL COEFFICIENT VECTOR (NY)
- NX ORDER OF POLYNOMIAL FOR WHICH VECTOR X IS THE LIST OF COEFFICIENTS (OTHER THAN THE FIRST)
- NY ORDER OF POLYNOMIAL FOR WHICH VECTOR Y IS THE LIST OF COEFFICIENTS (OTHER THAN THE FIRST)

OUTPUTS:

- Z X VECTOR * Y VECTOR (NZ)
- NZ ORDER OF POLYNOMIAL FOR WHICH VECTOR Z IS THE LIST OF COEFFICIENTS (OTHER THAN THE FIRST)

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

SUBROUTINE PREREQ (WHEN, ICA, IAND, MZ, I1)

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

SUBROUTINE PREREQ CHECKS TO SEE IF PREREQUISITE CASES HAVE BEEN DONE FOR THE CASE ABOUT TO BE RUN. IF NOT, IT PRINTS OUT WHAT THE PREREQUISITES ARE. PREREQ DOES NOT CALL ANY SUBROUTINES. PREREQ IS CALLED BY SUBROUTINES AES100, AES200, AES300, AES400, AES500, AES600, AES700, AND AES800.

INPUTS:

- WHEN LOGICAL MATRIX OF PREREQS (450,50)
- ICA VECTOR OF CASE NUMBERS TO BE DONE (1000)
- MZ WHICH CASE IS TO BE CHECKED FOR PREREQUISITES
- I1 WHICH ROW OF MATRIX 'WHEN' IS TO BE LOOKED AT

OUTPUTS:

- IAND DECISION VARIABLE, 0 IF PREREQUISITES HAVE BEEN DONE, 1 IF PREREQUISITES HAVE NOT BEEN DONE

*******************************************************************************
SUBROUTINE PRMUTE (X, ITRANS, IA, NE, LA, NEMAX)

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

SUBROUTINE PRMUTE PERMUTES ELEMENTS IN LA COLUMN OF NE BY NE MATRIX X AS
DICTATED BY TRANSPOSITION VECTOR ITRANS. ITRANS IS PRODUCED, IN THIS CASE, BY
SUBROUTINE FACTR. PRMUTE DOES NOT CALL ANY SUBROUTINES. PRMUTE IS CALLED BY
SUBROUTINE EGVCTR.

INPUTS:

X     MATRIX TO BE PERMUTED (NE,NE)
ITRANS TRANSPOSITION VECTOR (NE)
NE    ACTUAL SIZE OF MATRIX X
LA    SPECIFIC COLUMN OF MATRIX X TO BE PERMUTED BY
      THE TRANSPOSITION VECTOR ITRANS
NEMAX MAXIMUM SIZE OF NE

OUTPUTS:

X     INPUT MATRIX IN PERMUTED FORM (NE,NE)

TEMPORARY STORAGE:

IA    INTEGER VECTOR (NE)

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

SUBROUTINE REDU (VARO, SS, S, IN, JBL, INBL, IOR, NBL, IBL, IC, N,
1 NMAX)

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

SUBROUTINE REDU USES HARARY'S METHOD FOR REDUCTION OF A REDUCIBLE
MATRIX TO BLOCK DIAGONAL FORM. REDU DOES NOT CALL ANY SUBROUTINE.
REDU IS CALLED BY SUBROUTINE CONDI.

INPUTS:

VARO  MATRIX TO BE REDUCED (N,N)
N     ACTUAL SIZE OF MATRIX VARO
NMAX  MAXIMUM SIZE OF N

OUTPUTS:

S     BLOCK DIAGONAL MATRIX (N,N)
INBL  NUMBER OF REDUCIBLE BLOCKS
IOR   BLOCK-DIAGONALIZING PERMUTATION INTEGER
      VECTOR (N)
NBL   INTEGER VECTOR OF SIZES
      OF EACH IRREDUCIBLE
      BLOCK (N)

TEMPORARY STORAGE:

SS    MATRIX (N,N)
IN    INTEGER VECTOR (N)
JBL   INTEGER VECTOR (N)
IBL   INTEGER VECTOR (N)
IC    INTEGER VECTOR (N)

******************************************************************************
SUBROUTINE RESI (C, B, EIGR, EIGI, R, EXA, N, NC, NO, NMAX, NOMAX)

SUBROUTINE RESI computes the residue matrices for the linear system, \( \dot{x} = Ax + Bu \), and \( y = Cx \) where the system is assumed to be in block-diagonal form. Matrices \( C \) and \( B \) are input to the program. Matrix \( C \) is assumed to have been transformed to the form corresponding to block-diagonal \( A \) matrix using subroutine OBSBL. Matrix \( B \) is assumed to have been similarly transformed using subroutine CTBL.

Note: Two residue matrices are printed out for a complex eigenvalue; the first contains residue magnitudes and the second contains residue phase angles in degrees. RESI calls subroutine MATPRT. RESI is called by subroutine AES400.

**Inputs:**

- \( C \): output matrix \((N_0, N)\)
- \( B \): input matrix \((N, NC)\)
- \( EIGR \): vector of real parts of eigenvalues \((N)\)
- \( EIGI \): vector of imaginary parts of eigenvalues \((N)\)
- \( N \): number of states
- \( NC \): number of inputs
- \( NO \): number of outputs
- \( NMAX \): maximum size of \( N \)
- \( NOMAX \): maximum size of \( NO \)

**Outputs:**

- \( R \): residue array \((N, NO, NC)\)

**Temporary Storage:**

- \( EXA \): matrix \((N_0, NC)\) temporarily stores each residue matrix before being printed out.

SUBROUTINE RICCHK (AAA, S, R, N, NMAX, NZMAX)

SUBROUTINE RICCHK computes the residual error matrix for the RICCATI equation:

\[
S \cdot AAA(22) \cdot T + AAA(22) \cdot S - S \cdot AAA(12) \cdot S + AAA(21) = 0
\]

or,

\[
-S \cdot AAA(11) - AAA(11) \cdot T \cdot S - S \cdot AAA(12) \cdot S + AAA(21) = 0
\]

where

- \( S \): RICCATI solution matrix
- \( AAA \): the four \( N \) by \( N \) blocks of the Hamiltonian matrix
- \( R \): residual error matrix

R is given by

\[
R = S \cdot AAA(22) \cdot T + AAA(22) \cdot S - S \cdot AAA(12) \cdot S + AAA(21) = 0
\]

or,

\[
-S \cdot AAA(11) - AAA(11) \cdot T \cdot S - S \cdot AAA(12) \cdot S + AAA(21) = 0
\]

RICCHK calls subroutine MATPRT. RICCHK is called by subroutine AESBOO.

**Inputs:**

- \( AAA \): Hamiltonian matrix \((2 \times N, 2 \times N)\)
- \( S \): RICCATI solution matrix \((N, N)\)
- \( N \): actual size of matrix \( S \)
- \( NMAX \): maximum size of \( N \)
- \( NZMAX \): \( 2 \times NMAX \)
SUBROUTINE RICSS (AAA, X, OUTPUT, CR, CI, TS, XR, EXT, TT, IPER, IPERN, AR, AI, ADBLE, IOP1, IOP2, N, N2, NMAX, N2MAX)

SUBROUTINE RICSS COMPUTES THE OUTPUT SOLUTION TO THE STEADY STATE MATRIX RICCATI EQUATION. THE INPUT IS AN N2 BY N2 MATRIX, AAA, WHICH IS THE HAMILTONIAN MATRIX FOR KALMAN FILTER MATRIX RICCATI EQUATION. RICSS CALLS SUBROUTINES ARRAY, MXINV, EGCK, EGVCTR, EIGQR, HSBG, MATPRT, MODSHP, ORDER, AND SCALEA. RICSS IS CALLED BY SUBROUTINES CONTRL AND ESTMAT.

INPUTS:

AAA HAMILTONIAN MATRIX FOR KALMAN FILTER RICCATI EQUATION (N2,N2)
IOP1 SCALING PRINT OPTION: 0, NO PRINT; 1, PRINT
IOP2 EIGENVECTOR PRINT OPTION: 0, NO PRINT; 1, PRINT
N NUMBER OF STATE VARIABLES
N2 DIMENSION OF HAMILTONIAN MATRIX, 2 X N
NMAX MAXIMUM SIZE OF N
N2MAX MAXIMUM SIZE OF N2

OUTPUTS:

X MODIFIED EIGENVECTOR MATRIX OF AAA (N2,N2)
OUTPUT RICCATI SOLUTION MATRIX (N,N)
CR VECTOR OF REAL PARTS OF EIGENVALUES (N2)
CI VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N2)
TS SCALING TRANSFORMATION VECTOR OF AAA (N2)

TEMPORARY STORAGE:

XR MATRIX (N2,N2)
EXT MATRIX (N2,N2)
TT MATRIX (N2,N2)
IPER INTEGER VECTOR (N2)
IPERN INTEGER VECTOR (N2)
AR VECTOR (N2)
AI VECTOR (N2)
ADBLE VECTOR (N X N)

SUBROUTINE SCALEA (A, TS, N2, IOP1, N2MAX)

SUBROUTINE SCALEA TRANSFORMS N2 BY N2 MATRIX A USING DIAGONAL MATRIX TS SO THAT THE NORM OF A IS MINIMIZED. THE RESULTING SCALED MATRIX IS STORED IN A. IF SCALEA FINDS A TO BE REDUCIBLE, IER IS SET TO 1. SCALEA DOES NOT CALL ANY SUBROUTINES. SCALET IS CALLED BY SUBROUTINES CONDI, EIGEN, AND RICSS.

INPUTS:

A MATRIX TO BE SCALED (N2,N2)
N2 ACTUAL SIZE OF MATRIX A
IOP1 PRINT OPTION; 0 NO PRINT, 1 PRINT
N2MAX MAXIMUM SIZE OF N2
**Outputs:**

A  
INPUT MATRIX IN SCALED FORM (N2,N2)

TS  
VECTOR OF DIAGONAL ELEMENTS OF DIAGONAL SCALING MATRIX (N2)

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

**Subroutine STP**

**Inputs:**

- EX1: STATE TRANSITION, EXP(A*DT), MATRIX (N,N)
- EX2: INTEGRAL OF THE STATE TRANSITION MATRIX FROM TIME=0 TO TIME=DT (N,N)
- B: (CONTINUOUS) SYSTEM INPUT MATRIX (N,NIN)
- C: SYSTEM OUTPUT MATRIX (NOUT,N)
- DOUT: SYSTEM INPUT/OUTPUT FEEDTHRU MATRIX (NOUT,NIN)
- IOMTX: MATRIX OF ZEROS AND ONES (NIN,NOUT). ONES ARE PLACED IN SELECTED MATRIX POSITIONS TO INDICATE THE STEP RESPONSES DESIRED. THE FIRST INDEX IS 'INPUT', THE SECOND IS 'OUTPUT'. THUS SUBROUTINE STP MAY CALCULATE AS MANY AS NIN*NOUT STEP RESPONSES.
- AMPIN: VECTOR OF INPUT STEP AMPLITUDES (NIN)
- DT: TIME STEP
- TTIT: PLOT TITLE (12)
- TTOP: PLOT TITLE (12)
- TYTIT: Y AXIS TITLE (4)
- N: ACTUAL SIZE OF STATE TRANSITION MATRIX
- NIN: ACTUAL NUMBER OF POSSIBLE INPUTS
- NOUT: ACTUAL NUMBER OF POSSIBLE OUTPUTS
- NMAX: MAXIMUM SIZE OF N
- NINMAX: MAXIMUM SIZE OF NIN
- NOUTMX: MAXIMUM SIZE OF NOUT
- ITRMX: NUMBER OF DESIRED TIME RESPONSE POINTS
- IP: PLOT ENTITY INDEX (USED BY PLOTSUBS ONLY) INCREASES BY ONE FOR EACH FRAME
- NAME: NAME OF PLOT DATASET (9) (USED BY PLOTSUBS ONLY) (PARTITIONED DATASET THAT HOLDS PLOT ENTITIES)
- IONPLT: 0, IF OFFLINE PLOTS 1, IF ONLINE PLOTS

**Outputs:**

- TIME: VECTOR OF TIME POINTS (ITRMX) (SINGLE PRECISION)
- TYOUT: MATRIX OF OUTPUT TRANSIENT RESPONSES FOR ANY ONE SPECIFIC INPUT STEP (ITRMX,NOUT) (SINGLE PRECISION)
- IP: PLOT ENTITY INDEX (USED BY PLOTSUBS ONLY) INCREASES BY ONE FOR EACH FRAME
TEMPORARY STORAGE:

XNEW VECTOR (N)
XOLD VECTOR (N)
ANR VECTOR (N)
IEXT INTEGER VECTOR (N)

SUBROUTINE UNRML (KC, KE, KFF, PP, NC, NM, N, NCMAX, NMAX, FL34)

SUBROUTINE UNRML IS USED TO CONVERT NORMALIZED KC, KE, KFF, & PP MATRICES TO UN-NORMALIZED FORM. NORMALIZATION VECTOR INFORMATION IS FED IN THRU COMMON 'NORMS' OR IF NECESSARY, READ IN OFF UNIT 34 AS NAMELIST NRMS. IF FL34 IS .TRUE., IT MEANS THAT NRMS HAS ALREADY BEEN READ IN BY SUBROUTINE NRML & THUS IT SHOULDN'T BE READ IN HERE. UNRML CALLS SUBROUTINE MATPRT. UNRML IS CALLED BY SUBROUTINE AES400.

INPUTS:

KC NORMALIZED CONTROL GAIN MATRIX (NC,N)
KE NORMALIZED KALMAN FILTER GAIN MATRIX (N,NM)
KFF NORMALIZED FEED FORWARD GAIN MATRIX FOR NON-ZERO SET POINT REGULATOR (NC,NC)
PP NORMALIZED KALMAN FILTER ERROR COVARIANCE MATRIX (N,N)
NC ACTUAL NUMBER OF CONTROL INPUTS
NM ACTUAL NUMBER OF MEASUREMENTS
N ACTUAL NUMBER OF STATES
NCMAX MAXIMUM SIZE OF NC
NMAX MAXIMUM SIZE OF N
FL34 LOGICAL VARIABLE, ON INPUT
  TRUE, NORMALIZATION VECTOR INFORMATION (NAMELIST NRMS) HAS ALREADY BEEN READ IN
  FALSE, NORMALIZATION VECTOR INFORMATION (NAMELIST NRMS) NEEDS TO BE READ IN

OUTPUTS:

KC UN-NORMALIZED CONTROL GAIN MATRIX (NC,N)
KE UN-NORMALIZED KALMAN FILTER GAIN MATRIX (N,NM)
KFF UN-NORMALIZED FEED FORWARD GAIN MATRIX FOR NON-ZERO SET POINT REGULATOR (NC,NC)
PP UN-NORMALIZED KALMAN FILTER ERROR COVARIANCE MATRIX (N,N)
FL34 LOGICAL VARIABLE, ON OUTPUT SET TO
  TRUE IF NORMALIZATION VECTOR INFORMATION (NAMELIST NRMS) HAS BEEN READ IN

SUBROUTINE UZ8901

SUBROUTINE UZ8901 IS FOR A USER-WRITTEN ROUTINE

******************************************************************************
SUBROUTINE UZR902

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

THIS IS FOR A USER-WRITTEN ROUTINE
******************************************************************************

SUBROUTINE UZR903

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

THIS IS FOR A USER-WRITTEN ROUTINE
******************************************************************************

SUBROUTINE UZR904

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

THIS IS FOR A USER-WRITTEN ROUTINE
******************************************************************************

SUBROUTINE ZEROES (AA, BB, CC, DD, CONST, ANR, ANI, N, II, JJ, L,  
I AR, TS, BR, LWV, MWV, ZERMAX, IA, IB, IBL, IC, S, SS, NMAX,  
2 NOMAX)

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

SUBROUTINE ZEROES FINDS THE NUMERATOR ZEROES OF THE TRANSFER  
FUNCTION \( y(\text{II}) / u(\text{JJ}) = CC * ((S * I - A)^{-1}) * BB \)  
\text{II} DENOTES DESIRED COMPONENT OF OUTPUT VECTOR  
\text{JJ} DENOTES DESIRED COMPONENT OF INPUT VECTOR  
ZEROES CALLS SUBROUTINES CONDI, EIGQR, HSBG, AND ARRAY. ZEROES IS  
CALLED BY SUBROUTINE AES700.

INPUTS:

\begin{align*}  
\text{AA} & \quad \text{SYSTEM MATRIX (N,N)} \\
\text{BB} & \quad \text{INPUT MATRIX (N,NUMBER OF POSSIBLE INPUTS)} \\
\text{CC} & \quad \text{OUTPUT MATRIX (NUMBER OF POSSIBLE OUTPUTS,N)} \\
\text{DD} & \quad \text{CONSTANT} \\
\text{CONST} & \quad \text{ITERATION CONSTANT} \\
\text{N} & \quad \text{ACTUAL SIZE OF MATRIX AA} \\
\text{II} & \quad \text{OUTPUT COMPONENT} \\
\text{JJ} & \quad \text{INPUT COMPONENT} \\
\text{NMAX} & \quad \text{MAXIMUM SIZE OF N} \\
\text{NOMAX} & \quad \text{MAXIMUM NUMBER OF OUTPUTS} \\
\end{align*}

OUTPUTS:

\begin{align*}  
\text{ANR} & \quad \text{VECTOR OF REAL PARTS OF EIGENVALUES (N)} \\
\text{ANI} & \quad \text{VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N)} \\
\text{L} & \quad \text{NUMBER OF ZEROES} \\
\text{ZERMAX} & \quad \text{MAXIMUM EXPECTED VALUE OF TRANSFER FUNCTION ZEROES} \\
\end{align*}
TEMPORARY STORAGE:

AR MATRIX (N,N)
TS MATRIX (N,N)
BR VECTOR (N)
LWV INTEGER VECTOR (2 X N)
MWV INTEGER VECTOR (2 X N)
IA INTEGER VECTOR (2 X N)
IB INTEGER VECTOR (2 X N)
IBL INTEGER VECTOR (2 X N)
IC INTEGER VECTOR (2 X N)
S MATRIX (N,N)
SS MATRIX (N,N)

*******************************************************************************
Appendix C
Test Cases

This appendix presents the results of two test cases. Test case I exercises almost all of the 77 available AESOP functions. Test case II shows how AESOP can be used in an interactive manner to design a control system.

Test Case I – Third-Order Problem to Demonstrate Full AESOP Program Capabilities

A block diagram of the selected third-order, open-loop plant is shown in figure 14. The plant is characterized by three states, two controls, one noisy measurement, two plant-noise disturbances, two outputs, and two set-point variables. The plant is stable with one real pole at 0.1 rad/sec and a complex pole pair with a natural frequency of 1.0 rad/sec and a damping ratio of 0.001. The computations performed do not all relate to one specific design problem but are done so as to exercise all of the AESOP functions. For example, frequency responses are computed for a system with perfect state measurement \((x_1, x_2, x_3)\) and state feedback as well as for Kalman filter feedback using a single noisy measurement, \(z_1\).

The input matrices for test case I are generated by function 201 and are listed in table V. They include the plant matrices \(A, B, C, D, CSP, H,\) and \(DOUT\) as well as quadratic weighting matrices \(QC, NN,\) and \(PCINV\) and noise matrices \(QQ\) and \(RRINV\). The test case exercises the AESOP functions shown in table XI.

Pages 65 to 80 are the terminal printout sheets produced at the user's terminal. Notes have been added to indicate what results are being generated and displayed. In the pocket at the rear of this report are microfiche copies of the output dataset and plots generated for this test case. The output dataset contains the complete results generated by the AESOP run, including results that were printed out at the user's terminal. The CPU time required to run this test case on an IBM 370/3033 is approximately 30 to 40 seconds depending on which device was used to produce graphic output.

---

![Block diagram of third-order system used for test case I.](image-url)
TABLE XI. - TASKS PERFORMED IN TEST CASE I

<table>
<thead>
<tr>
<th>Design task performed</th>
<th>Page number, appendix C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-loop analyses</td>
<td>68</td>
</tr>
<tr>
<td>Optimal regulator gains and associated error checks</td>
<td>68</td>
</tr>
<tr>
<td>Kalman filter gains and associated error checks</td>
<td>70</td>
</tr>
<tr>
<td>State covariance matrices and feedforward matrix</td>
<td>72</td>
</tr>
<tr>
<td>Frequency responses (and Bode plots)</td>
<td>72</td>
</tr>
<tr>
<td>Transient responses (and plots)</td>
<td>74</td>
</tr>
<tr>
<td>Transfer function gain and zeros</td>
<td>76</td>
</tr>
<tr>
<td>System matrix normalization</td>
<td>77</td>
</tr>
<tr>
<td>Repeat of tasks 2 and 3 with normalized matrices</td>
<td>79</td>
</tr>
<tr>
<td>Unnormalization of resultant matrices</td>
<td>79</td>
</tr>
</tbody>
</table>

Terminal Printout for Test Case I

AESRUN 1
CANCELED: OUT1 UNKNOWN.
OUT1 IS DDEFD TO THE HI-SPEED PRINTER
CGI IS DDEFD TO IO UNIT 8
EGI IS DDEFD TO IO UNIT 9
PFUZI IS DDEFD TO IO UNIT 10
PFKUI IS DDEFD TO IO UNIT 11
PFUZI IS DDEFD TO IO UNIT 12
PFUZI IS DDEFD TO IO UNIT 13
CFRI IS DDEFD TO IO UNIT 14
FP1 IS DDEFD TO IO UNIT 15
S51 IS DDEFD TO IO UNIT 16
PFOI IS DDEFD TO IO UNIT 17
RUNI IS LIBRARY GRAPHICS
RUN2 IS LIBRARY AESLIB

AESOP
DO YOU WISH TO MAKE PLOTS, Y OR N?
Y
ENTER THE PLOT NAME - a ALPHANUMERIC CHARACTERS
LUCYTEST
DO YOU WISH TO MAKE ONLINE PLOTS, Y OR N?
M
ENTER TODAY'S DATE (LESS THAN OR EQUAL TO 20 CHARACTERS)
DEC 2, 1982
ENTER THE PARAMETER YOU USED FOR AESRUN 1
EXTENDED TERMINAL OUTPUT?
Y
READ IN N1 FROM STORAGE?
Y
DDEF 21 TO THE N1 DATASET
PROCEEDING: PAUSE . PROGRAM WILL PROCEED IF _RUN COMMAND ISSUED.
DDEF FT21FOOl,VS,DEC80

PROCEEDING(PCS): EXECUTION CONTINUES AT CHCRCW.(X'4CD4')

FUNCTION 101
CHANGES TO REFS
DISPLAY REFS BEFORE MAKING CHANGES?
Y
&REFS
TSFR= 1.0
DT= 0.500-01
FE= 0.100-01
D2L= 0.200-01
ZERMAX= 100.0
AMPSR= 1.0
INF= 101
IN= 201 401 402 403 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 0
&END

Function 101 requested to demonstrate procedure for changing a "reference" parameter (in NAMELIST 'REFS')
FUNCTION 201
PUT IN THE 3RD ORDER TEST CASE MATRICES BY USING SUBROUTINE MATIN

FUNCTION THAT FORMS ALL REQUIRED MATRICES FOR RUNNING THE TEST CASE

A=

1 2 3
1 -0.1000D 00 1.000 0.0000
2 0.0000 0.0000 1.000
3 0.0000 -1.000 -0.2000D-02

B=

1 2
1 0.0000 0.0000
2 0.0000 1.000
3 1.000 0.0000

D=

1 2
1 0.0000 0.0000
2 0.0000 1.000
3 1.000 0.0000

C=

1 2 3
1 1.000 0.0000 0.0000
2 0.0000 0.0000 1.000

H=

1 2 3
1 1.000 0.0000 0.0000
DOUT=

1  2
1  0.0000  0.0000
2  0.0000  1.000

CSP=

1  2  3
1  1.000  0.0000  0.0000
2  0.0000  0.0000  2.000

QC=

1  2  3
1  20.00  0.0000  0.0000
2  0.0000  1.000  0.0000
3  0.0000  0.0000  10.00

NM=

1  2
1  1.000  0.0000
2  0.0000  1.000
3  2.000  0.0000

PCINV=

1  2
1  1.000  0.0000
2  0.0000  0.10000

QQ=

1  2  3
1  0.0000  0.0000  0.0000
2  0.0000  2.000  0.0000
3  0.0000  0.0000  20.00

RRINV=

1

1  1.000

&REFS
TSFTR= 1.0
DF= 0.00D-01
FI= 0.10D-01
DELF= 0.00D-01
ZEMAX= 500.0
AMPSP= 5×1.0
AMPSR= 5×1.0
AMPICX= 5×1.0
IF= 49
ISPACE= 1
IGUT= 1
IMEAS= 1
JINC= 1
JINDE= 1
ITRMX= 100
NCURV= 2
LINLOG= 3
MSPY= 250×1
MSPYSR= 25×1
MSPU= 25×1
MSROIY= 250×1
MSROI= 250×1
MICCLY= 250×1
MICCLX= 250×1
MICOLY= 250×1
MICOLX= 250×1
NEND
The following three functions perform analyses on the open-loop system:

### FUNCTION 401
**OPEN LOOP SYSTEM EIGENVALUES**

Matrix is found to be reducible.

<table>
<thead>
<tr>
<th>NAT FREQ (HZ)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1594</td>
<td>0.1000D-02</td>
</tr>
<tr>
<td>0.1592D-01</td>
<td>1.000</td>
</tr>
</tbody>
</table>

### FUNCTION 402
**OPEN LOOP EIGENVECTORS AND MODE SHAPES**

**MODIFIED EIGENVECTOR MATRIX OF A**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.000</td>
<td>-0.9990</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>0.9990</td>
<td>0.9990</td>
<td>0.0000</td>
</tr>
<tr>
<td>3</td>
<td>1.000</td>
<td>1.000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

### FUNCTION 403
**OPEN LOOP SYSTEM ANALYSIS CALCULATIONS**

**SYSTEM CONTROLABILITY**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>2</td>
<td>0.5730D-01</td>
<td>0.0000</td>
</tr>
<tr>
<td>3</td>
<td>0.9903</td>
<td>-0.9705D-01</td>
</tr>
</tbody>
</table>

**SYSTEM OBSERVABILITY FOR (A AND H)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.9571</td>
<td>174.4</td>
<td>1.000</td>
</tr>
</tbody>
</table>

**SYSTEM OBSERVABILITY FOR (A AND C)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9951</td>
<td>174.4</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>1.000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

**RESIDUES FOR (A, B, H) SYSTEM**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19.24</td>
<td>10.62</td>
</tr>
<tr>
<td>2</td>
<td>18.42</td>
<td>9.28</td>
</tr>
<tr>
<td>3</td>
<td>1.301</td>
<td>1.819</td>
</tr>
</tbody>
</table>

**RESIDUES FOR (A, B, C) SYSTEM**

**BEGINNING OF RICCATI EQUATION SOLUTION FOR LQR PROBLEM**

### FUNCTION 801
**DESIGN A LINEAR QUADRATIC REGULATOR**

**EIGENVALUES OF REGULATOR OR FILTER IN FH/ZETA FORM**

<table>
<thead>
<tr>
<th>NAT FREQ (HZ)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2582</td>
<td>0.7116</td>
</tr>
<tr>
<td>0.4519</td>
<td>1.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19.24</td>
<td>10.62</td>
<td>1.301</td>
</tr>
<tr>
<td>2</td>
<td>18.42</td>
<td>9.28</td>
<td>1.819</td>
</tr>
<tr>
<td>3</td>
<td>1.301</td>
<td>1.819</td>
<td>1.647</td>
</tr>
</tbody>
</table>
KC =

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.301</td>
<td>1.819</td>
<td>3.647</td>
</tr>
<tr>
<td>2</td>
<td>1.042</td>
<td>1.220</td>
<td>0.1819</td>
</tr>
</tbody>
</table>

FUNCTION 802
STORE OPTIMAL CONTROL GAINS (KC) ON UNIT 08.

FUNCTION 301
FORM A-BKC MATRIX

FUNCTION 803
EIGENVALUES OF SYSTEM WITH STATE FEEDBACK

EIGENVALUES

<table>
<thead>
<tr>
<th>NAT FREQ (HZ)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2382</td>
<td>0.7116</td>
</tr>
<tr>
<td>0.4519</td>
<td>1.000</td>
</tr>
</tbody>
</table>

FUNCTION 804
EIGENVECTORS AND MODE SHAPES WITH STATE FEEDBACK

MODIFIED EIGENVECTOR MATRIX OF AMBKC

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.9899</td>
<td>0.4246D-01</td>
<td>0.1694</td>
</tr>
<tr>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>-0.4092</td>
</tr>
<tr>
<td>3</td>
<td>0.2144</td>
<td>-1.042</td>
<td>1.000</td>
</tr>
</tbody>
</table>

THE MATRIX OF MODE SHAPES IN MAG. AND ANGLE(DEG.) FORM

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7006</td>
<td>-132.5</td>
<td>0.1694</td>
</tr>
<tr>
<td>2</td>
<td>1.000</td>
<td>0.000</td>
<td>-0.4092</td>
</tr>
<tr>
<td>3</td>
<td>0.7519</td>
<td>123.4</td>
<td>1.000</td>
</tr>
</tbody>
</table>

--- Beginning of error checks for the Riccati solution matrix ---

FUNCTION 805
POSITIVE DEFINITENESS CHECK OF CONTROL RICCATI SOLUTION MATRIX, SS

EIGENVALUES

<table>
<thead>
<tr>
<th>NAT FREQ (HZ)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1642</td>
<td>-1.000</td>
</tr>
<tr>
<td>4.133</td>
<td>-1.000</td>
</tr>
</tbody>
</table>

Negative † indicates that eigenvalues all have positive real parts; therefore SS is positive-definite

FUNCTION 806
SYMNETRY CHECK OF CONTROL RICCATI SOLUTION MATRIX, SS

MAX. SYMMETRY ERROR IN SS = 3.2429E-15
AVG. ABSOLUTE SYMMETRY ERROR IN SS = 1.7184E-15

SYMNETRY ERROR IN SS =

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>-0.4475D-15</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>3</td>
<td>-0.3243D-14</td>
<td>-3.1465D-14</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

FUNCTION 807
RESIDUAL ERROR CHECK OF SS

MAX. RESIDUAL, R(1, 2) = 0.2842D-13
TRACE OF RESIDUAL = -0.2977D-13
RESIDUAL ERROR MATRIX FOR CONTROL RICCATI EQUATION

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & -0.1421D-13 & -0.2842D-13 & -0.1577D-13 \\
2 & 0.1754D-13 & -0.1622D-13 & -0.1438D-14 \\
3 & -0.8882D-15 & 0.4441D-15 & 0.6661D-15 \\
\end{bmatrix}
\]

The residual remaining upon substitution of solution matrix SS back into the original Riccati equation.

FUNCTION A08
STORE CONTROL RICCATI SOLUTION MATRIX (SS) ON UNIT 16

FUNCTION A09
DESIGN A KALMAN FILTER

EIGENVALUES OF REGULATOR OR FILTER IN FM/ZETA FORM

<table>
<thead>
<tr>
<th>NAT FREQ (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2262</td>
<td>1.000</td>
</tr>
<tr>
<td>0.2862</td>
<td>0.4491</td>
</tr>
</tbody>
</table>

PP =

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & 2.882 & 4.442 & 1.482 \\
2 & 4.442 & 11.76 & 8.665 \\
3 & 1.482 & 8.665 & 18.57 \\
\end{bmatrix}
\]

KE =

\[
\begin{bmatrix}
1 \\
2 & 2.882 \\
3 & 4.442 \\
\end{bmatrix}
\]

FUNCTION A10
STORE KALMAN FILTER GAINS (KE) ON UNIT 09

FUNCTION 302
FORM A-BKC-KEH MATRIX

FUNCTION A11
EIGENVALUES OF OPTIMAL CONTROLLER A-BKC-KEH

EIGENVALUES

<table>
<thead>
<tr>
<th>NAT FREQ (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6057</td>
<td>1.000</td>
</tr>
<tr>
<td>0.5366</td>
<td>0.6000</td>
</tr>
</tbody>
</table>

FUNCTION 303
FORM ATOT, CTOT, DTOT, KCTOT, AND HTOT MATRICES FOR OPTIMAL CONTROL SYSTEM WITH KALMAN FILTER IN FEEDBACK LOOP

ATOT =

\[
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & -0.1000D 00 & 1.000 & 0.0000 & 9.0000 & 0.0000 & 0.0000  \\
2 & 0.0000 & 0.0000 & 1.000 & -1.042 & -1.220 & -0.1819  \\
3 & 0.0000 & -1.000 & -0.2000D-02 & -3.701 & -1.819 & -3.647  \\
4 & 2.882 & 0.0000 & 0.0000 & -2.982 & 1.000 & 0.0000  \\
5 & 4.442 & 0.0000 & 0.0000 & -5.484 & -1.220 & 0.8101  \\
6 & 1.482 & 0.0000 & 0.0000 & -3.783 & -2.819 & -3.649  \\
\end{bmatrix}
\]

CTOT =

\[
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1.000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000  \\
2 & 0.0000 & 0.0000 & 1.000 & -1.042 & -1.220 & -0.1819  \\
\end{bmatrix}
\]

70
FUNCTION a12
EIGENVALUES OF CONTROL SYSTEM WITH A FILTER IN THE FEEDBACK LOOP

EIGENVALUES

<table>
<thead>
<tr>
<th>FREQUENCY (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2202</td>
<td>1.000</td>
</tr>
<tr>
<td>0.2382</td>
<td>0.7116</td>
</tr>
<tr>
<td>0.2862</td>
<td>0.4451</td>
</tr>
<tr>
<td>0.4519</td>
<td>1.000</td>
</tr>
</tbody>
</table>

These are the eigenvalues of $A - KEH$

These are the eigenvalues of $A - BKC$

FUNCTION a13
POSITIVE DEFINITENESS CHECK OF ERROR COVARIANCE MATRIX, PP

EIGENVALUES

<table>
<thead>
<tr>
<th>FREQUENCY (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1103</td>
<td>-1.000</td>
</tr>
<tr>
<td>1.138</td>
<td>-1.000</td>
</tr>
<tr>
<td>4.006</td>
<td>-1.000</td>
</tr>
</tbody>
</table>

Matrix PP is positive definite

FUNCTION a14
SYMMETRY CHECK OF ERROR COVARIANCE MATRIX, PP

MAX. SYMMETRY ERROR IN PP = -2.9962E-16
AVG. ABSOLUTE SYMMETRY ERROR IN PP = 1.6667E-16

SYMMETRY ERROR IN PP =

FUNCTION a15
RESIDUAL ERROR CHECK OF PP

MAX. RESIDUAL, R(2,2) = -0.1155D-13
TRACE OF RESIDUAL = -0.2265D-13

RESIDUAL ERROR MATRIX FOR ESTIMATION RICCATI EQUATION

<table>
<thead>
<tr>
<th>FREQUENCY (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>-0.2996E-15</td>
<td>0.2004E-15</td>
</tr>
</tbody>
</table>

RESIDUAL ERROR MATRIX FOR ESTIMATION RICCATI EQUATION

<table>
<thead>
<tr>
<th>FREQUENCY (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.4410D-15</td>
<td>-0.3775D-14</td>
</tr>
<tr>
<td>-0.4219D-14</td>
<td>-0.5958D-14</td>
</tr>
<tr>
<td>-0.2849D-14</td>
<td>-0.1087D-13</td>
</tr>
</tbody>
</table>
FUNCTION 816
STORE ERROR COVARIANCE MATRIX (PP) ON UNIT 15

FUNCTION 817
COMPUTE COVARIANCE MATRICES FOR LINEAR QUADRATIC REGULATOR
WITH KALMAN FILTER IN FEEDBACK LOOP

UU, CONTROL COVARIANCE MATRIX

1 2
1 41.42  20.81
2 20.81  25.13

XX, STATE COVARIANCE MATRIX

1 2 3
1 20.99  2.099  -6.849
2 2.099  21.56  8.428
3 -6.849  8.428  24.26

YY, OUTPUT COVARIANCE MATRIX

1 2
1 20.99  -21.35
2 -21.35  65.67

ZZ, MEASUREMENT COVARIANCE MATRIX

1
1 20.99

FUNCTION 818
LYAPUNOV ERROR CHECK FOR FUNCTION 817 — Computes error incurred in calculating state covariance matrix XX

THE TRACE OF THE RESIDUAL = 0.6516D12
NORMALIZED DIAGONAL ELEMENTS OF THE ERROR MATRIX =
-0.6497D-15 0.6305D-15 -0.4062D-14
THE TRACE OF THE ERROR =-0.9857D-13
THE TRACE OF THE COVARIANCE= 66.806
TRERROR/TCOV =-0.1475D-14

FUNCTION 819
FORM FEED FORWARD MATRIX FOR NON-ZERO SET POINT CONTROL

KFF =

1 2
1 2.583  1.826
2 1.164  -0.4090

FUNCTION 820
STORE FEED FORWARD MATRIX (KFF) ON UNIT 17

Beginning of frequency response and Bode plot computation

FUNCTION 581
OPEN LOOP FREQUENCY RESPONSE OF MEASUREMENT 1 TO CONTROL INPUT 1

NUMERATOR COEFFICIENTS
-0.12089D-16 1.0000  1.0000
-0.12089D-16 1.0000  1.0000
0.0000  1.0000  1.0000
1.0000  1.0000  1.0000

DENOMINATOR COEFFICIENTS
0.0000  1.0000  1.0000
-0.12089D-16 1.0000  1.0000
0.0000  1.0000  1.0000
1.0000  1.0000  1.0000
FUNCTION 502
PLOT OPEN LOOP FREQ. RESPONSE OF MEASUREMENT TO CONTROL INPUT

GRAPHICS DEVICE NOT DEFINED BY DDEF.
ENTER UNIT NAME. DEFAULT TO CANCEL.
ZETA12
PLOT IDENTIFICATION = PELUCY** 02 DEC 1982 08:06:18

FUNCTION 503
STORE OPEN LOOP FREQ. RESPONSE OF MEASUREMENT TO CONTROL INPUT ON UNIT 10

FUNCTION 504
OPEN LOOP FREQUENCY RESPONSE OF OUTPUT 1 TO CONTROL INPUT 1

NUMERATOR COEFFICIENTS
DENOMINATOR COEFFICIENTS
0.00000  *S** 3  1.0000  *S** 3
-0.12089D-16* S** 2  0.10200  *S** 2
0.00000  *S** 1  1.0000  *S** 1
1.0000  *S* 0  0.10000D 00* S** 0

FUNCTION 505
PLOT OPEN LOOP FREQ. RESPONSE OF OUTPUT TO CONTROL INPUT

FUNCTION 506
STORE OPEN LOOP FREQ. RESPONSE OF OUTPUT TO CONTROL INPUT ON UNIT 11

FUNCTION 507
OPEN LOOP FREQUENCY RESPONSE OF OUTPUT 1 TO DISTURBANCE INPUT 1

NUMERATOR COEFFICIENTS
DENOMINATOR COEFFICIENTS
0.00000  *S** 3  1.0000  *S** 3
-0.12089D-16* S** 2  0.10200  *S** 2
0.00000  *S** 1  1.0000  *S** 1
1.0000  *S* 0  0.10000D 00* S** 0

FUNCTION 508
PLOT OPEN LOOP FREQ. RESPONSE OF OUTPUT TO DISTURBANCE INPUT

FUNCTION 509
STORE OPEN LOOP FREQ. RESPONSE OF OUTPUT TO DISTURBANCE INPUT ON UNIT 12

FUNCTION 510
OPEN LOOP FREQUENCY RESPONSE OF OUTPUT 1 TO DISTURBANCE INPUT 1

NUMERATOR COEFFICIENTS
DENOMINATOR COEFFICIENTS
0.00000  *S** 3  1.0000  *S** 3
-0.12089D-16* S** 2  0.10200  *S** 2
0.00000  *S** 1  1.0000  *S** 1
1.0000  *S* 0  0.10000D 00* S** 0

FUNCTION 511
PLOT OPEN LOOP FREQ. RESPONSE OF OUTPUT TO DISTURBANCE INPUT

FUNCTION 512
STORE OPEN LOOP FREQ. RESPONSE OF OUTPUT TO DISTURBANCE INPUT ON UNIT 13

FUNCTION 513
CLOSED LOOP FREQUENCY RESPONSE OF OUTPUT 1 TO DISTURBANCE INPUT 1
FOR STATE FEEDBACK

NUMERATOR COEFFICIENTS
DENOMINATOR COEFFICIENTS
0.00000  *S** 3  1.0000  *S** 3
-0.43240D-19* S** 2  4.9693  *S** 2
-0.22204D-15* S** 1  8.2882  *S** 1
0.61010  *S* 0  6.3604  *S* 0

FUNCTION 514
PLOT CLOSED LOOP FREQ. RESPONSE OF OUTPUT TO DISTURBANCE INPUT
FOR STATE FEEDBACK
(PLUS THE CORRESPONDING OPEN LOOP RESPONSE, IF DESIRED)

FUNCTION 515
CLOSED LOOP FREQUENCY RESPONSE OF CONTROL 1 TO DISTURBANCE INPUT 1
FOR STATE FEEDBACK

NUMERATOR COEFFICIENTS
DENOMINATOR COEFFICIENTS
0.00000  *S** 3  1.0000  *S** 3
-3.6670  *S** 2  4.9693  *S** 2
-6.3034  *S** 1  8.2882  *S** 1
-6.2763  *S* 0  6.3604  *S* 0
FUNCTION 516
PLOT CLOSED LOOP FREQUENCY RESPONSE OF CONTROL TO DISTURBANCE INPUT
FOR STATE FEEDBACK

FUNCTION 517
CLOSED LOOP FREQUENCY RESPONSE OF MEASUREMENT 1 TO DISTURBANCE INPUT 1
FOR CONTROL SYSTEM WITH A FILTER IN THE FEEDBACK LOOP

NUMERATOR COEFFICIENTS

<table>
<thead>
<tr>
<th>Numerator Coefficients</th>
<th>Denominator Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000 M** 6</td>
<td>1.0000 M** 6</td>
</tr>
<tr>
<td>0.00000 M** 5</td>
<td>7.9536 M** 5</td>
</tr>
<tr>
<td>0.00000 M** 4</td>
<td>28.566 M** 4</td>
</tr>
<tr>
<td>1.0000 M** 3</td>
<td>62.639 M** 3</td>
</tr>
<tr>
<td>7.8516 M** 2</td>
<td>86.362 M** 2</td>
</tr>
<tr>
<td>26.764 M** 1</td>
<td>71.725 M** 1</td>
</tr>
<tr>
<td>43.263 M** 0</td>
<td>28.452 M** 0</td>
</tr>
</tbody>
</table>

FUNCTION 518
PLOT CLOSED LOOP FREQUENCY RESPONSE OF MEASUREMENT TO DISTURBANCE INPUT
FOR CONTROL SYSTEM WITH FILTER IN FEEDBACK LOOP
(PLUS THE CORRESPONDING OPEN LOOP RESPONSE, IF DESIRED)

FUNCTION 519
CLOSED LOOP FREQUENCY RESPONSE OF OUTPUT 1 TO DISTURBANCE INPUT 1
FOR CONTROL SYSTEM WITH A FILTER IN THE FEEDBACK LOOP

NUMERATOR COEFFICIENTS

<table>
<thead>
<tr>
<th>Numerator Coefficients</th>
<th>Denominator Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000 M** 6</td>
<td>1.0000 M** 6</td>
</tr>
<tr>
<td>0.00000 M** 5</td>
<td>7.9536 M** 5</td>
</tr>
<tr>
<td>0.00000 M** 4</td>
<td>28.566 M** 4</td>
</tr>
<tr>
<td>1.0000 M** 3</td>
<td>62.639 M** 3</td>
</tr>
<tr>
<td>7.8516 M** 2</td>
<td>86.362 M** 2</td>
</tr>
<tr>
<td>26.764 M** 1</td>
<td>71.725 M** 1</td>
</tr>
<tr>
<td>43.263 M** 0</td>
<td>28.452 M** 0</td>
</tr>
</tbody>
</table>

FUNCTION 520
PLOT CLOSED LOOP FREQUENCY RESPONSE OF OUTPUT TO DISTURBANCE INPUT
FOR CONTROL SYSTEM WITH FILTER IN FEEDBACK LOOP
(PLUS THE CORRESPONDING OPEN LOOP RESPONSE, IF DESIR ED)

FUNCTION 521
CLOSED LOOP FREQUENCY RESPONSE OF CONTROL 1 TO DISTURBANCE INPUT 1
FOR CONTROL SYSTEM WITH A FILTER IN THE FEEDBACK LOOP

NUMERATOR COEFFICIENTS

<table>
<thead>
<tr>
<th>Numerator Coefficients</th>
<th>Denominator Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000 M** 6</td>
<td>1.0000 M** 6</td>
</tr>
<tr>
<td>-0.44409D-15 M** 5</td>
<td>7.9536 M** 5</td>
</tr>
<tr>
<td>-0.13323D-14 M** 4</td>
<td>28.566 M** 4</td>
</tr>
<tr>
<td>-0.39968D-14 M** 3</td>
<td>62.639 M** 3</td>
</tr>
<tr>
<td>-0.20117 M** 2</td>
<td>86.362 M** 2</td>
</tr>
<tr>
<td>-4.8315 M** 1</td>
<td>71.725 M** 1</td>
</tr>
<tr>
<td>-24.088 M** 0</td>
<td>28.452 M** 0</td>
</tr>
</tbody>
</table>

FUNCTION 522
PLOT CLOSED LOOP FREQUENCY RESPONSE OF CONTROL TO DISTURBANCE INPUT
FOR CONTROL SYSTEM WITH FILTER IN FEEDBACK LOOP

FUNCTION 523
FREQUENCY RESPONSE OF CONTROL 1 TO MEASUREMENT 1
FOR OPTIMAL CONTROLLER

NUMERATOR COEFFICIENTS

<table>
<thead>
<tr>
<th>Numerator Coefficients</th>
<th>Denominator Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000 M** 3</td>
<td>1.0000 M** 3</td>
</tr>
<tr>
<td>-0.44409D-15 M** 5</td>
<td>7.9536 M** 5</td>
</tr>
<tr>
<td>-0.13323D-14 M** 4</td>
<td>28.566 M** 4</td>
</tr>
<tr>
<td>-0.39968D-14 M** 3</td>
<td>62.639 M** 3</td>
</tr>
<tr>
<td>-0.20117 M** 2</td>
<td>86.362 M** 2</td>
</tr>
<tr>
<td>-4.8315 M** 1</td>
<td>71.725 M** 1</td>
</tr>
<tr>
<td>-24.088 M** 0</td>
<td>28.452 M** 0</td>
</tr>
</tbody>
</table>

FUNCTION 524
PLOT FREQUENCY RESPONSE OF CONTROL TO MEASUREMENT
FOR OPTIMAL CONTROLLER

FUNCTION 525
STORE FREQUENCY RESPONSE OF CONTROL TO MEASUREMENT
FOR OPTIMAL CONTROLLER ON UNIT 14

FUNCTION 601
OBTAIN AND PLOT SELECTED OPEN LOOP STEP RESPONSES
STATE TRANSITION MATRIX FOR OPEN LOOP SYSTEM FOR TIME STEP =0.5000D-01

Beginning of transient response calculations and plots

74
1  2  3
1  0.9950  0.4985D-01  0.1248D-02
2  0.0000  0.9958  0.4998D-01
3  0.0000  -0.4998D-01  0.9987

FORCED RESPONSE MATRIX OF OPEN LOOP SYSTEM FOR TIME STEP 0.5000D-01

1  2  3
1  0.4985D-01  0.1248D-02  0.2088D-04
2  0.0000  0.4998D-01  0.1256D-02
3  0.0000  -0.1256D-02  0.4998D-01

FUNCTION 602
OBTAIN AND PLOT SELECTED INITIAL CONDITION RESPONSES FOR THE OPEN LOOP SYSTEM

STATE TRANSITION MATRIX FOR OPEN LOOP SYSTEM FOR TIME STEP 0.5000D-01

1  2  3
1  0.9950  0.4985D-01  0.1248D-02
2  0.0000  0.9958  0.4998D-01
3  0.0000  -0.4998D-01  0.9987

FORCED RESPONSE MATRIX OF OPEN LOOP SYSTEM FOR TIME STEP 0.5000D-01

1  2  3
1  0.4985D-01  0.1248D-02  0.2088D-04
2  0.0000  0.4998D-01  0.1256D-02
3  0.0000  -0.1256D-02  0.4998D-01

FUNCTION 603
OBTAIN AND PLOT SELECTED INITIAL CONDITION RESPONSES FOR THE CLOSED LOOP LINEAR REGULATOR

STATE TRANSITION MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.5000D-01

1  2  3
1  0.9937  0.4832D-01  0.9613D-01
2  -0.5251D-01  0.9369  0.3613D-01
3  -0.1014  -0.1273  0.8307

FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.5000D-01

1  2  3
1  0.4985D-01  0.1222D-02  0.1602D-04
2  -0.1310D-02  0.4844D-01  0.9429D-03
3  -0.2645D-02  -0.3294D-02  0.4566D-01

FUNCTION 604
OBTAIN AND PLOT SELECTED STEP RESPONSES FOR THE NON-ZERO SET POINT LINEAR REGULATOR

STATE TRANSITION MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.5000D-01

1  2  3
1  0.9937  0.4832D-01  0.9613D-01
2  -0.5251D-01  0.9369  0.3613D-01
3  -0.1014  -0.1273  0.8307

FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.5000D-01

1  2  3
1  0.4985D-01  0.1222D-02  0.1602D-04
2  -0.1310D-02  0.4844D-01  0.9429D-03
3  -0.2645D-02  -0.3294D-02  0.4566D-01
FUNCTION 701
GAIN AND ZEROS OF OPEN LOOP TRANSFER FUNCTION
RELATING MEASUREMENT 1 TO CONTROL INPUT 1
GAIN = 1.0000
NUMBER OF ZEROS = 0
REAL PARTS OF NUMERATOR ZEROS

1
1 0.0000

IMAGINARY PARTS OF NUMERATOR ZEROS

1
1 0.0000

FUNCTION 702:
GAIN AND ZEROS OF OPEN LOOP TRANSFER FUNCTION
RELATING OUTPUT 1 TO CONTROL INPUT 1
GAIN = 1.0000
NUMBER OF ZEROS = 0
REAL PARTS OF NUMERATOR ZEROS

1
1 0.0000

IMAGINARY PARTS OF NUMERATOR ZEROS

1
1 0.0000

FUNCTION 703
GAIN AND ZEROS OF OPEN LOOP TRANSFER FUNCTION
RELATING MEASUREMENT 1 TO DISTURBANCE INPUT 1
GAIN = 1.0000
NUMBER OF ZEROS = 0
REAL PARTS OF NUMERATOR ZEROS

1
1 0.0000

IMAGINARY PARTS OF NUMERATOR ZEROS

1
1 0.0000

FUNCTION 704
GAIN AND ZEROS OF OPEN LOOP TRANSFER FUNCTION
RELATING OUTPUT 1 TO DISTURBANCE INPUT 1
GAIN = 1.0000
NUMBER OF ZEROS = 0
REAL PARTS OF NUMERATOR ZEROS

1
1 0.0000

IMAGINARY PARTS OF NUMERATOR ZEROS

1
1 0.0000
FUNCTION 705
GAIN AND ZEROS OF OPTIMAL CONTROLLER TRANSFER FUNCTION
RELATING CONTROL 1 TO MEASUREMENT 1

GAIN =  28.1170
NUMBER OF ZEROS =  2
REAL PARTS OF NUMERATOR ZEROS

1  -0.1699
2  -0.1699

IMAGINARY PARTS OF NUMERATOR ZEROS

1  1.081
2  -1.081

Beginning of series of functions that demonstrate (1) normalizing of input matrices,
(2) normalizing of control weighting matrices, (3) recomputing of KC, KE, and KFF
matrices, and (4) unnormalizing KC, KE, KFF, and FF to show that results are
identical to prior calculations.

FUNCTION 210
CHANGE MATRICES A, C, D, H, DOUT, CSP, QC, NN, PCINV, QQ, OR RRINV AND ANY
OR ALL DIMENSIONS BY READING CHANGES IN AT TERMINAL USING NAMELIST 'MATDAT'
DISPLAY MATDAT BEFORE MAKING CHANGES!

Y

MATDAT
A= -0.10D0, 49W0.0, 1.0, 0.0. , -1.0, 48W0.0, 1.0, -0.20D-02, 2397W0.0
B= 2W0.0, 1.0, 48W0.0, 1.0, 198W0.0
C= 1.0, 100W0.0, 1.0, 2398W0.0
D= 2W0.0, 1.0, 48W0.0, 1.0, 698W0.0
H= 1.0, 2498W0.0
DOUT= 51W0.0, 1.0, 198W0.0
CSP= 2W0.0, 1.0, 2.0, 258W0.0
QC= 20.0, 50W0.0, 1.0, 50W0.0, 10.0, 2397W0.0
NN= 1.0, 0.0, 2.0, 48W0.0, 3.0, 198W0.0
PCINV= 1.0, 5W0.0, 0.10D0, 18W0.0
QQ= 51W0.0, 2.0, 50W0.0, 20.0, 2397W0.0
RRINV= 1.0, 24W0.0
N= 3
M= 1
NC= 2
ND= 2
NN= 2
END

ENTER MATDAT CHANGES AS &MATDAT A(1,1)= , ETC. &END

MATDAT
QC(1,1) = 500.000
QC(2,2) = 9.000
QC(3,3) = 40.000
PCINV(1,1) = 1.5625D-2
PCINV(2,2) = 1.2345679D-3
NN(1,1) = 40.000
NN(3,1) = 32.000
NN(2,2) = 01.000
END

MATDAT
A= -0.10D0, 49W0.0, 1.0, 0.0. , -1.0, 48W0.0, 1.0, -0.20D-02, 2397W0.0
B= 2W0.0, 1.0, 48W0.0, 1.0, 198W0.0
C= 1.0, 100W0.0, 1.0, 2398W0.0
D= 2W0.0, 1.0, 48W0.0, 1.0, 698W0.0
H= 1.0, 2498W0.0
DOUT= 51W0.0, 1.0, 198W0.0
CSP= 2W0.0, 1.0, 2.0, 258W0.0
QC= 20.0, 50W0.0, 1.0, 50W0.0, 10.0, 2397W0.0
NN= 1.0, 0.0, 2.0, 48W0.0, 3.0, 198W0.0
PCINV= 1.0, 5W0.0, 0.10D0, 18W0.0
QQ= 51W0.0, 2.0, 50W0.0, 20.0, 2397W0.0
RRINV= 1.0, 24W0.0
N= 3
M= 1
NC= 2
ND= 2
NN= 2
END

ARE THERE ANYMORE CHANGES, Y OR N?

FUNCTION 404
NORMALIZE SYSTEM MATRICES;
NORMALIZING FACTORS ARE READ IN FROM UNIT 34 IF NOT PREVIOUSLY SET
IF NOT ALREADY DONE, DDEF DATASET CONTAINING NAMELIST 'HRMS' TO UNIT 34
PROCEEDING: PAUSE . PROGRAM WILL PROCEED IF _RUN COMMAND ISSUED.

DDEF FT34F001.VS.ANRM
GO
PROCEEDING(PCS): EXECUTION CONTINUES AT CHCRWC.(X'A4CD4')

NORMALIZING FACTORS ARE

ANRMS
SCX= 5.0, 3.0, 2.0, 47W0.0
SCU= 8.0, 9.0, 34W0.0
NORMALIZED SYSTEM MATRICES

THE A MATRIX IS

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & -0.100000 & 0.0000 & 0.0000 \\
2 & 0.0000 & 0.0000 & -0.100000 \\
3 & 0.0000 & -1.5000 & -0.200000 \\
\end{bmatrix}
\]

THE B MATRIX IS

\[
\begin{bmatrix}
1 & 2 \\
1 & 0.0000 & 0.0000 \\
2 & 0.0000 & 3.0000 \\
3 & 4.0000 & 0.0000 \\
\end{bmatrix}
\]

THE C MATRIX IS

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & 1.2500 & 0.0000 & 0.0000 \\
2 & 0.0000 & 0.0000 & 0.2057 \\
\end{bmatrix}
\]

THE H MATRIX IS

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & 0.8333 & 0.0000 & 0.0000 \\
\end{bmatrix}
\]

THE QQ MATRIX IS

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & 0.0000 & 0.0000 & 0.0000 \\
2 & 0.0000 & 0.2222 & 0.0000 \\
3 & 0.0000 & 0.0000 & 5.0000 \\
\end{bmatrix}
\]

THE RRINV MATRIX IS

\[
\begin{bmatrix}
1 \\
1 & 36.00 \\
\end{bmatrix}
\]

THE D MATRIX IS

\[
\begin{bmatrix}
1 & 2 \\
1 & 0.0000 & 0.0000 \\
2 & 0.0000 & 0.3333 \\
3 & 0.5000 & 0.0000 \\
\end{bmatrix}
\]

THE DOUT MATRIX IS

\[
\begin{bmatrix}
1 & 2 \\
1 & 0.0000 & 0.0000 \\
2 & 0.0000 & 1.2860 \\
\end{bmatrix}
\]

THE CSP MATRIX IS

\[
\begin{bmatrix}
1 & 2 & 3 \\
1 & 0.4167 & 0.0000 & 0.0000 \\
2 & 0.0000 & 0.0000 & 0.3636 \\
\end{bmatrix}
\]

FUNCTION 801
DESIGN A LINEAR QUADRATIC REGULATOR
EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM

<table>
<thead>
<tr>
<th>NAT FREQ (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2382</td>
<td>0.7116</td>
</tr>
<tr>
<td>0.4519</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

\[55 = \]

\[1 \hspace{1cm} 2 \hspace{1cm} 3\]

1  681.1  156.3  13.01
2  156.3  82.83  10.91
3  13.01  10.91  6.568

\[KC = \]

\[1 \hspace{1cm} 2 \hspace{1cm} 3\]

1  1.438  0.6821  0.9117
2  0.5789  0.4088  0.40420-01

FUNCTION 301
FORM A-BKC MATRIX

FUNCTION 809
DESIGN A KALMAN FILTER

EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM

<table>
<thead>
<tr>
<th>NAT FREQ (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2202</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.2862</td>
<td>0.4451</td>
</tr>
</tbody>
</table>

\[PP = \]

\[1 \hspace{1cm} 2 \hspace{1cm} 3\]

1  0.1153  0.2961  0.1482
2  0.2961  1.297  1.477
3  0.1482  1.477  4.591

\[KE = \]

\[1 \]

1  3.459
2  8.884
3  4.446

FUNCTION 819
FORM FEED FORWARD MATRIX FOR NON-ZERO SET POINT CONTROL

\[KFF = \]

\[1 \hspace{1cm} 2\]

1  3.874  2.589
2  1.552  -0.4999

FUNCTION 405
UNNORMALIZE GAINS AND ERROR COVARIANCE MATRIX;
NORMALIZING FACTORS ARE READ IN FROM UNIT 34 IF NOT PREVIOUSLY SET

\[\text{NORMALIZING FACTORS} \]

\[\$\text{NEMS} \]

\[\text{SCX} = 5.0, \ 3.0, \ 2.0, \ 47\times0.0\]
\[\text{SCU} = 8.0, \ 9.0, \ 38.0\]
\[\text{SCV} = 4.0, \ 7.0, \ 10.0, \ 47\times0.0\]
\[\text{SCZ} = 6.0, \ 4\times0.0\]
\[\text{SC7SP} = 12.0, \ 11.0, \ 3\times0.0\]

\[\&\text{END} \]

Check shows that the following matrices are identical to those previously computed:

UNNORMALIZED SYSTEM MATRICES

THE KC MATRIX IS
Test Case II – Interactive Design of a Nonzero-Set-Point Regulator

Test case II demonstrates how AESOP can be used in an interactive manner to design a feedback control system. The same third-order state-variable system model used in test case I is considered, except that the outputs of primary interest are now the set-point outputs $y_{sp}$. They are selected to be $y_{sp1} = x_1$ and $y_{sp2} = x_3$ by proper definition of matrix $C_{SP}$. The following assumptions are made:

1. All three states are measurable.
2. Both disturbance $w$ and measurement noise $v$ are zero.

The design problem is to compute feedforward ($K_{FF}$) and feedback ($K_{C}$) gain matrices such that the resulting closed-loop system meets the following design criteria:

1. Each of the two set-point outputs follows a step in its corresponding set point with zero steady-state error. (This is insured by the nonzero-set-point regulator structure.)
2. Set-point output step responses are well damped (less than 10 percent overshoot) and settle out in less than 5 seconds.
3. For a unit step on either set point, excursions of the two control variables must be such that $|u_1| \leq 5$ and $|u_2| \leq 5$.

The resulting closed-loop system is shown in figure 15. Basically, the design procedure followed is to vary performance index weights in an interactive fashion, displaying the step responses of $y_{sp1}$ and $y_{sp2}$ for each candidate design and repeating the process until acceptable transient responses are observed. Once transient performance is considered acceptable (criteria 1 and 2 are satisfied), the control variable responses are displayed to check that criterion 3 is also met.

The input data that define the problem are stored in dataset TEST.CASE2, which is shown in figure 16. Note that the values of some of the reference parameters in NAMELIST REFS have been made different from the default values so as to obtain the desired output variable selection and time step for the step responses. Also, unlike test case I the initial values of weighting matrices $Q_{C}$ and $N_{N}$ are allowed to be zero, and $P_{CINV}$ is set equal to a $2 \times 2$ identity matrix.

To begin the design process the AESOP program is initiated as was done in test case I. When the program prompts the user with the message READ IN NI FROM STORAGE?, the user replies "N." The program then prompts the user to enter function numbers from the terminal.
**Figure 15.** Nonzero-set-point regulator.

**Figure 16.** Data for test case II, contained in dataset TEST.CASE2.

Enter namelist data as `&N1 IFN = , , &END` and the user responds with

```
&n1 ifn = 202,203,801 &end
```

The user then datadefs dataset TEST.CASE2 to unit 33.

**FUNCTION 202**

READ INPUT DATA -- MATRICES AND REFERENCE VALUES DEFINED
IN NAMELISTS 'MATDAT' AND 'REFS'

IF NOT ALREADY DONE, DDEF DATASET CONTAINING NAMELISTS
'MATDAT' AND 'REFS' TO UNIT 33
CHCRW410 PROCEEDING: PAUSE
ddef ft33f001,vs,test.case2
endo
CZAP8030 PROCEEDING(FCS): EXECUTION CONTINUES AT CHKRTC.(X'4CD4')
DISPLAY INPUT MATRICES?

The next requested function (203) allows the user to enter desired performance index weights. For this test case, weights on the two controls will be fixed at unity and the weight on state 1 will be varied. A low value (0.001) is selected initially, and the program proceeds to compute an LQR solution.

**FUNCTION 203**

DISPLAY COMPAR BEFORE MAKING CHANGES?

```
&compar
DC= 2500X0.0
NN= 250X0.0
```
The user then requests that step responses be plotted for a nonzero-set-point regulator. This requires, as prerequisites, forming of $A-B\cdot KC$ and calculation of feedforward gain matrix $KFF$. 

FUNCTION 301
FORM A-BKC MATRIX

FUNCTION 819
FORM FEED FORWARD MATRIX FOR NON-ZERO SET POINT CONTROL

$KFF =$

$1$ $2$

$1$ 0.1048 0.2801D-01
$2$ 0.2737D-02 0.9996
FUNCTION 604
OBTAIN AND PLOT SELECTED STEP RESPONSES
FOR THE NON-ZERO SET POINT LINEAR REGULATOR

STATE TRANSITION MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9763</td>
<td>0.2343</td>
<td>0.2361D-01</td>
</tr>
<tr>
<td>2</td>
<td>0.2002D-03</td>
<td>0.9664</td>
<td>0.2362</td>
</tr>
<tr>
<td>3</td>
<td>0.1108D-02</td>
<td>-0.2365</td>
<td>0.9648</td>
</tr>
</tbody>
</table>

FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2371</td>
<td>0.2839D-01</td>
<td>0.2276D-02</td>
</tr>
<tr>
<td>2</td>
<td>-0.2841D-04</td>
<td>0.2371</td>
<td>0.2854D-01</td>
</tr>
<tr>
<td>3</td>
<td>-0.1349D-03</td>
<td>-0.2857D-01</td>
<td>0.2369</td>
</tr>
</tbody>
</table>

Next, before plots can be obtained, the user must indicate on which device the plots are to be generated by entering the user's terminal number (LA001), indicating that plots are to be displayed not on an off-line device but at the user's terminal.

GRIN100  GRAPHICS DEVICE NOT DEFINED BY DDEF.
ENTER UNIT NAME. DEFAULT TO CANCEL.
la011

Two plots are then displayed: one for the response of $y_{sp1}$ to a step in $y_{spd1}$, and the other for $y_{sp2}$ to a step in $y_{spd2}$. 
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD ( 1 )
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (\( \frac{2}{2} \))
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 9
With this low weighting on $x_1$, the system responses are obviously not acceptable, a fact also discernible from an examination of the closed-loop eigenvalues displayed by function 801. Thus the user requests another design iteration.

FUNCTION 0
TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS.(13), ONE PER LINE
TO TERMINATE ENTER 999(last entry must be a return)
203
801
819
604

This time, the weight on $x_1$ is increased to 1.0, and the LQR and feedforward gains are recomputed.

FUNCTION 203
DISPLAY COMPARE BEFORE MAKING CHANGES?

FUNCTION 801
DESIGN A LINEAR QUADRATIC REGULATOR

EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM

<table>
<thead>
<tr>
<th>MAT FREQUENCY (Hz)</th>
<th>ZETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1392</td>
<td>1.000</td>
</tr>
<tr>
<td>0.3027</td>
<td>0.5564</td>
</tr>
</tbody>
</table>

SS =

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.206</td>
<td>2.783</td>
<td>0.9032</td>
</tr>
<tr>
<td>2.783</td>
<td>1.950</td>
<td>0.6620</td>
</tr>
<tr>
<td>0.9032</td>
<td>0.6620</td>
<td>0.9392</td>
</tr>
</tbody>
</table>

KC =

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9032</td>
<td>0.6620</td>
<td>0.9392</td>
</tr>
<tr>
<td>2.783</td>
<td>1.950</td>
<td>0.6620</td>
</tr>
</tbody>
</table>

FUNCTION 819
FORM FEED FORWARD MATRIX FOR NON-ZERO SET POINT CONTROL
KFF =

\[
\begin{array}{cc}
1 & 2 \\
1 & 1.069 & 0.9412 \\
2 & 2.978 & -0.3380 \\
\end{array}
\]

FUNCTION 604
OBTAIN AND PLOT SELECTED STEP RESPONSES
FOR THE NON-ZERO SET POINT LINEAR REGULATOR

STATE TRANSITION MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

\[
\begin{array}{ccc}
1 & 2 & 3 \\
1 & 0.9089 & 0.1832 & 0.7571D-02 \\
2 & -0.5167 & 0.5573 & 0.5555D-01 \\
3 & -0.8230D-01 & -0.2934 & 0.7855 \\
\end{array}
\]

FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

\[
\begin{array}{ccc}
1 & 2 & 3 \\
1 & 0.2315 & 0.2420D-01 & 0.6446D-03 \\
2 & -0.6794D-01 & 0.1856 & 0.7636D-02 \\
3 & -0.1475D-01 & -0.3928D-01 & 0.2139 \\
\end{array}
\]

Now, the damping ratio of the complex eigenvalue pair has increased to 0.388, which is still too small to give acceptably damped responses. This fact is confirmed by the set-point responses, which are displayed next.
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (1)
AMPLITUDE = 1.00
OEC 7, 1982
RUN NO. 8
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND \( Y_{SPD} \) \( (2) \)
AMPLITUDE = 1.00

DEC 7, 1982
RUN NO. 3
Actually, the 10-percent overshoot criterion is met, but $y_{sp1}$ does not quite reach steady state in less than 5 seconds. Thus the user begins a third design iteration, choosing the weighting on $x_1$ to be 10 this time.

FUNCTION 203
TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS. (13), ONE PER LINE
TO TERMINATE ENTER 999 (LAST ENTRY MUST BE A RETURN)
203
801
301
819
604

FUNCTION 203
DISPLAY COMPARE BEFORE MAKING CHANGES?

DISPLAY CONTROL WTS QC, NN AND/OR PCINV (NAMELIST COMPARE)

FUNCTION 801
DESIGN A LINEAR QUADRATIC REGULATOR

EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM
NAT FREQ (HZ) ZETA
0.95240 0.01 1.000
0.2063 0.3579

The preceding eigenvalues indicate that response time should now be acceptable as the slowest eigenvalue ($\lambda_1 = 0.1392$ Hz = 0.875 rad/sec) should give a settling time ($\approx 4/\lambda_1$) of about 4.5 seconds.

SS =

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.285</td>
<td>0.6844</td>
<td>0.5240</td>
</tr>
<tr>
<td>2</td>
<td>0.6844</td>
<td>0.7483</td>
<td>0.3449</td>
</tr>
<tr>
<td>3</td>
<td>0.5240</td>
<td>0.3449</td>
<td>0.7536</td>
</tr>
</tbody>
</table>

KC =

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5240</td>
<td>0.3449</td>
<td>0.7536</td>
</tr>
<tr>
<td>2</td>
<td>0.6844</td>
<td>0.7483</td>
<td>0.3449</td>
</tr>
</tbody>
</table>

FUNCTION 301
FORM A-BKC MATRIX

FUNCTION 819
FORM FEED FORWARD MATRIX FOR NON-ZERO SET POINT CONTROL
$K_{FF} =$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.6585</td>
<td>0.7556</td>
</tr>
<tr>
<td>2</td>
<td>0.7592</td>
<td>-0.6551</td>
</tr>
</tbody>
</table>

**FUNCTION 604**

**OBTAIN AND PLOT SELECTED STEP RESPONSES**

**FOR THE NON-ZERO SET POINT LINEAR REGULATOR**

**STATE TRANSITION MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9574</td>
<td>0.2138</td>
<td>0.1648E-01</td>
</tr>
<tr>
<td>2</td>
<td>-0.1549</td>
<td>0.7967</td>
<td>0.1292</td>
</tr>
<tr>
<td>3</td>
<td>-0.8076E-01</td>
<td>-0.2785</td>
<td>0.8125</td>
</tr>
</tbody>
</table>

**FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2354</td>
<td>0.2673E-01</td>
<td>0.1366E-02</td>
</tr>
<tr>
<td>2</td>
<td>-0.1901E-01</td>
<td>0.2164</td>
<td>0.1662E-01</td>
</tr>
<tr>
<td>3</td>
<td>-0.1208E-01</td>
<td>-0.3521E-01</td>
<td>0.2177</td>
</tr>
</tbody>
</table>

The step responses confirm that both set-point output transients are acceptable.
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (1)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3

TIME
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPB (2)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
As a final check, the control responses are examined for this design to see that required control excursions are within limits (±5.0). To do this, the user requests function 101, which allows changes to be made in reference values. In particular, the user changes elements of transient response selection matrices $\text{MSYSP}$ and $\text{MSPU}$ so as to request both the control responses and the output responses, thus displaying the interaction between output 1 and input 2 and vice-versa.

FUNCTION 0
TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS. (I3), ONE PER LINE
TO TERMINATE ENTER 999 (LAST ENTRY MUST BE A RETURN)
101
604

FUNCTION 101
CHANGES TO REFS
DISPLAY REFS BEFORE MAKING CHANGES?

n
ENTER CHANGES TO NAMELIST REFS (TSFTR, DT, FI, DELF, ZERMAX, AMPSF, AMPSR, AMPICX, IF, ISPACE, IOUT, IMESA, JINC, JIND, ITRMX, NCURV, LINLOG, MSPY, MSPSP, MSPU, MSROLY, MSROLX, MICYL, MICYLX, MICYLU, MICYLUX, MICYLUX)

The program then displays the reference values and plots the requested transients.

$\text{REFS}$
$\text{MSYSP}(1,1)=0.1, \text{MSYSP}(1,2)=1,0, \text{MSPU}(1,1)=1,1, \text{MSPU}(1,2)=1,1$ &end

$\text{REFS}$
$\text{TSFTR}=1.0$
$\text{DT}=0.240$
$\text{FI}=0.100 \times 01$
$\text{DELF}=0.100 \times 01$
$\text{ZERMAX}=100.0$
$\text{AMPSF}=5 \times 0$
$\text{AMPSR}=5 \times 1.0$
$\text{AMPICX}=5 \times 1.0$
$\text{IF}=90$
$\text{ISPACE}=1$
$\text{IOUT}=1$
$\text{JINC}=1$
$\text{JIND}=1$
$\text{ITRMX}=100$
$\text{NCURV}=2$
$\text{LINLOG}=2$
$\text{MSPY}=250 \times 0$
$\text{MSYSP}=0, 1, 3 \times 0, 1, 3 \times 0$
$\text{MSPU}=2, 1, 3 \times 0, 2, 0, 1, 18 \times 0$
$\text{MSROLY}=250 \times 0$
$\text{MSROLX}=2, 1, 3 \times 0, 2, 0, 1, 23 \times 0$
$\text{MICCLY}=1, 2499 \times 0$
$\text{MICCLX}=2500 \times 0$
$\text{MICCLU}=2500 \times 0$
$\text{MICOLY}=1, 2499 \times 0$
$\text{MICOLX}=1, 50 \times 0, 1, 2448 \times 0$

&end

ARE THERE ANYMORE CHANGES, Y OR N?

FUNCTION 604
OBTAIN AND PLOT SELECTED STEP RESPONSES
FOR THE NON-ZERO SET POINT LINEAR REGULATOR

STATE TRANSITION MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9089</td>
<td>0.1832</td>
<td>0.75719-02</td>
</tr>
<tr>
<td>2</td>
<td>-0.5167</td>
<td>0.5573</td>
<td>0.55550-01</td>
</tr>
<tr>
<td>3</td>
<td>-0.8230D-01</td>
<td>-0.2934</td>
<td>0.7855</td>
</tr>
</tbody>
</table>
FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2315</td>
<td>0.2420D-01</td>
<td>0.6469D-03</td>
</tr>
<tr>
<td>2</td>
<td>-0.6794D-01</td>
<td>0.1056</td>
<td>0.7636D-02</td>
</tr>
<tr>
<td>3</td>
<td>-0.1475D-01</td>
<td>-0.3928D-01</td>
<td>0.2138</td>
</tr>
</tbody>
</table>

STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (1)
AMPLITUDE = 1.00
DECEMBER 7, 1982
RUN NO. 3
Note that there is some interaction between input 1 and output 2 but little between input 2 and output 1.

Finally, the four control responses are displayed. These responses confirm that design criterion 3 has been met. That is, both control signal absolute magnitudes remain less than 5 during a step response. The maximum excursion for $u_1$ is $+1.07$ and that for $u_2$ is $+2.98$, both well below the design limits.
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (1)
AMPLITUDE = 1.00

DEC 7, 1982
RUN NO. 3
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD ( 1 )
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (2)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (2)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
After the responses are displayed, the user is again prompted for more requests, at which time the user terminates the program. The terminal session ends with the user requesting a printout of the output dataset (OUTxxx) that was generated during the present run for further off-line analysis if desired.

FUNCTION 0
TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS. (I3), ONE PER LINE
TO TERMINATE ENTER 999 (LAST ENTRY MUST BE A RETURN)
999
STORE THE N1 FOR THIS RUN?

CHCRW400 TERMI NATED: STOP RETURN
PRINT OUTAS, RRTSR=EDIT
CZAI8050 PRINT BSN=0498, 800 LINES
off
8007 LOGOFF AT 14:10 ON 07/07/81 - CPU TIME = 0.07 MINUTES.
LOGICAL DISCONNECT, LOGON OR HANG UP
Appendix D
Terminal Output Options and Main PROCDEF

This appendix includes (1) a list of those items included in the “standard” terminal output, (2) a list of those items included in the “extended” terminal output, and (3) the PROCDEF that is used to set up the program before it is run.

Standard Terminal Output

The following data will be displayed in the user’s terminal if the user has requested the functions that generate these data:

(1) NAMELIST N1
(2) NAMELIST REFS
(3) NAMELIST CONPAR
(4) NAMELIST ESTPAR
(5) NAMELIST MATDAT
(6) Open- and closed-loop eigenvalues
(7) Kalman filter eigenvalues
(8) Transfer function numerator and denominator polynomial coefficients
(9) Transfer function gains
(10) Maximum and average symmetry error for the SS and PP matrices
(11) Positive-definiteness checks for the SS and PP matrices
(12) Maximum element of and trace of the residual error matrix for the control and Kalman filter

Riccati equations

(13) Lyapunov error check data consisting of
   (a) Trace of residual
   (b) Normalized diagonal elements of error matrix
   (c) Trace of error
   (d) Trace of covariance
   (e) Ratio of trace of error to trace of covariance
(14) Normalizing factors
(15) Error messages

Extended Terminal Output

The extended terminal output consists of all of the standard terminal output plus

(1) Input matrices
(2) All eigenvectors and mode shapes
(3) The ATOT, CTOT, DTOT, KCTOT, and HTOT matrices
(4) The SS matrix
(5) The PP matrix
(6) The KC matrix
(7) The KE matrix
(8) The KFF matrix
(9) All covariance matrices
(10) Controllability check matrix
(11) Observability check matrix (through H)
(12) Observability check matrix (through C)
(13) Transfer function gains and zeroes
(14) All state-transition matrices
(15) All forced-response matrices
(16) Symmetry error matrix for the control Riccati equation
(17) Residual error matrix for the control Riccati equation
(18) Symmetry error matrix for the Kalman filter Riccati equation
Residual error matrix for the Kalman filter Riccati equation
Normalized system matrices: A, B, C, H, QQ, RRINV, D, DOUT, and CSP

PROCDEF AESRUN

The user invokes the following PROCDEF, AESRUN, before running the AESOP program. The purpose of the PROCDEF is to datadef all necessary libraries and required datasets. The PROCDEF requires one parameter (up to three characters), which is used to label all datasets that may be generated during the subsequent run of the AESOP program.

AESRUN 0000000 PROCDEF AESRUN
AESRUN 0000100 PARAM $1
AESRUN 0000200 ERASE OUT$1
AESRUN 0000300 DDEF FT06F001,VS,OUT$1,DCB = (RECFM = V,LRECL = 132),RET = T
AESRUN 0000400 DISPLAY ‘OUT$1 IS DDEFD TO THE HI-SPEED PRINTER’
AESRUN 0000500 DDEF FT08F001,VS,CG$1; DISPLAY ‘CG$1 IS DDEFD TO IO UNIT 8’
AESRUN 0000600 DDEF FT09F001,VS,EG$1; DISPLAY ‘EG$1 IS DDEFD TO IO UNIT 9’
AESRUN 0000700 DDEF FT10F001,VS,PFRUZ$1; DISPLAY ‘PFRUZ$1 IS DDEFD TO IO UNIT 10’
AESRUN 0000800 DDEF FT11F001,VS,PFRUY$1; DISPLAY ‘PFRUY$1 IS DDEFD TO IO UNIT 11’
AESRUN 0000900 DDEF FT12F001,VS,PFRWZ$1; DISPLAY ‘PFRWZ$1 IS DDEFD TO IO UNIT 12’
AESRUN 0001000 DDEF FT13F001,VS,PFRWY$1; DISPLAY ‘PFRWY$1 IS DDEFD TO IO UNIT 13’
AESRUN 0001100 DDEF FT14F001,VS,CFR$1; DISPLAY ‘CFR$1 IS DDEFD TO IO UNIT 14’
AESRUN 0001200 DDEF FT15F001,VS,PP$1; DISPLAY ‘PP$1 IS DDEFD TO IO UNIT 15’
AESRUN 0001300 DDEF FT16F001,VS,SS$1; DISPLAY ‘SS$1 IS DDEFD TO IO UNIT 16’
AESRUN 0001400 DDEF FT17F001,VS,FFG$1; DISPLAY ‘FFG$1 IS DDEFD TO IO UNIT 17’
AESRUN 0001500 DDEF RUN1,VP,GRAPHICS, OPTION = JOBLIB; DISPLAY ‘RUN1 IS LIBRARY GRAPHICS’
AESRUN 0001600 DDEF RUN2,VP,AESLIB,OPTION = JOBLIB; DISPLAY ‘RUN2 IS LIBRARY AESLIB’
AESRUN 0001700 LOAD BLOCKA9; LOAD AESOP

Eleven VS datasets that are to contain program input or output are datadeffed by this PROCDEF. Table I lists these datasets plus four others that might be datadeffed by the user during the course of running the program.
Appendix E
Flow Chart

This appendix contains a subroutine flow chart of the AESOP program in “tree” form. Where specific functions are performed in a subroutine, the function numbers are listed (in parentheses) below the name of the subroutine.
This appendix contains the information used by the AESOP program to make prerequisite checks. These checks are performed before each AESOP function is to be executed to see that the necessary function or combination of functions has been performed prior to execution of the present function. Table XII lists this prerequisite check information in the following form: The specific prerequisite function or combinations of functions are listed across the top, and each function whose prerequisites are to be checked is listed on the left. A checkmark appears in a row to indicate a prerequisite. Multiple checkmarks in a row indicate that the corresponding prerequisites are to be logically "ANDed." For example, the logical prerequisite statement for function 303 is (201 OR 202) AND (205 OR 801) AND (206 OR 809).
### TABLE XII. - PREREQUISITES FOR AESOP FUNCTIONS

<p>| Function | 201 or 202 | 205 or 206 | 207 or 208 | 209 or 301 | 302 | 303 | 205, 206, 207, 208, 301, or 209 | 401 | 402 | 404 | 501 | 504 | 507 | 510 | 513 | 515 | 517 | 519 | 521 | 523 | 801 | 803 | 809 | 817 | 819 or 209 |
|----------|------------|------------|------------|------------|-----|-----|-------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 101      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 102      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 201      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 202      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 203      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 204      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 205      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 206      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 207      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 208      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 209      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 210      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 301      | X          | X          |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 302      | X          | X          | X          |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 303      | X          | X          | X          |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 401      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 402      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 403      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 404      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 405      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 501      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 502      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 503      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 504      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 505      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 506      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 507      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 508      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 509      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 510      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 511      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 512      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 513      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 514      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 515      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 516      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 517      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 518      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 519      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 520      |            |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 521      | X          |            |            |            |     |     |                               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Prerequisite function</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td>522</td>
<td>x</td>
</tr>
<tr>
<td>523</td>
<td></td>
</tr>
<tr>
<td>524</td>
<td></td>
</tr>
<tr>
<td>525</td>
<td></td>
</tr>
<tr>
<td>601</td>
<td>x</td>
</tr>
<tr>
<td>602</td>
<td></td>
</tr>
<tr>
<td>603</td>
<td>x</td>
</tr>
<tr>
<td>604</td>
<td>x</td>
</tr>
<tr>
<td>701</td>
<td>x</td>
</tr>
<tr>
<td>702</td>
<td></td>
</tr>
<tr>
<td>703</td>
<td></td>
</tr>
<tr>
<td>704</td>
<td>x</td>
</tr>
<tr>
<td>705</td>
<td></td>
</tr>
<tr>
<td>801</td>
<td>x</td>
</tr>
<tr>
<td>802</td>
<td></td>
</tr>
<tr>
<td>803</td>
<td>x</td>
</tr>
<tr>
<td>804</td>
<td></td>
</tr>
<tr>
<td>805</td>
<td>x</td>
</tr>
<tr>
<td>806</td>
<td></td>
</tr>
<tr>
<td>807</td>
<td></td>
</tr>
<tr>
<td>808</td>
<td></td>
</tr>
<tr>
<td>809</td>
<td>x</td>
</tr>
<tr>
<td>810</td>
<td></td>
</tr>
<tr>
<td>811</td>
<td></td>
</tr>
<tr>
<td>812</td>
<td>x</td>
</tr>
<tr>
<td>813</td>
<td>x</td>
</tr>
<tr>
<td>814</td>
<td></td>
</tr>
<tr>
<td>815</td>
<td></td>
</tr>
<tr>
<td>816</td>
<td></td>
</tr>
<tr>
<td>817</td>
<td></td>
</tr>
<tr>
<td>818</td>
<td></td>
</tr>
<tr>
<td>819</td>
<td></td>
</tr>
<tr>
<td>820</td>
<td></td>
</tr>
<tr>
<td>901</td>
<td>x</td>
</tr>
<tr>
<td>902</td>
<td></td>
</tr>
<tr>
<td>903</td>
<td></td>
</tr>
<tr>
<td>904</td>
<td></td>
</tr>
</tbody>
</table>
References


AESOP is a computer program for use in designing feedback controls and state estimators for linear multivariable systems. AESOP is meant to be used in an interactive manner. Each design task that the program performs is assigned a "function" number. The user accesses these functions either (1) by inputting a list of desired function numbers or (2) by inputting a single function number. In the latter case the choice of the function will in general depend on the results obtained by the previously executed function. The most important of the AESOP functions are those that design linear quadratic regulators and Kalman filters. The user interacts with the program when using these design functions by inputting design weighting parameters and by viewing graphic displays of designed system responses. Supporting functions are provided that obtain system transient and frequency responses, transfer functions, and covariance matrices. The program can also compute open-loop system information such as stability (eigenvalues), eigenvectors, controllability, and observability. The program is written in ANSI-66 Fortran for use on an IBM 3033 using TSS 370. Descriptions of all subroutines and results of two test cases are included in the appendixes.
THIS ENVELOPE CONTAINS THE
MICROFICHE REFERENCE SUPPLEMENT
TO NASA TP-2221
OUTPUT OF TEST CASE I
(SUPPLEMENT TO APPENDIX C)