NASA
Technical
Paper
2221

January 1984

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

Bruce Lehtinen
and Lucille C. Geyser

NASA

NASA
Technical
Paper
2221

1984

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

National Aeronautics
and Space Administration

**Scientific and Technical
Information Office**

1984

Contents

Page

Summary 1

Introduction 1

Theoretical Background and Problem Formulation 2
- Open-Loop System Description 2
- Eigenvalues and Eigenvectors 4
- Controllability, Observability, and Mode Shapes 4
- Residues 5
- Steady-State Linear Quadratic Regulator (LQR) Design 5
- Steady-State Kalman Filter Design 7
- Normalization 7
- Stochastic Linear Quadratic Regulator Design 9
- System Response to Noise Inputs 9
- Transfer Functions and Frequency Response Calculations 10
- Transient Response Calculation 13

AESOP Program Operation 15
- Program Structure 15
- Operating Procedure 15
- Data Input and Output 19

Description of AESOP Functions 20
- Series 100 – Program Control 22
- Series 200 – Data Input and Revision 22
- Series 300 – Matrix Formation 24
- Series 400 – Open-Loop System Analysis 24
- Series 500 – Frequency Responses and Bode Plots; and Series 700 – Transfer Functions 25
- Series 600 – Transient Responses 26
- Series 800 – LQR and Filter Design 26
- Series 900 – User-Supplied Subroutines 28

Concluding Remarks 28

Appendixes
- A – Symbols 30
- B – Subroutine Descriptions 33
- C – Test Cases 64
  - Test Case I – Third-Order Problem to Demonstrate Full AESOP Program Capabilities 64
  - Terminal Printout for Test Case I 65
  - Test Case II – Interactive Design of a Nonzero-Set-Point Regulator 80
- D – Terminal Output Options and Main PROCDEF 102
  - Standard Terminal Output 102
  - Extended Terminal Output 102
  - PROCDEF AESRUN 103
- E – Flow Chart 104
- F – Prerequisite Table 108

References 111

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 programs are ASP by Kalman and Englar (ref. 6) and its Fortran version VASP (ref. 7). Subsequent LQR design packages were the OPTSYS program of Bryson and Hall (ref. 8), the ORACLS 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 part 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 prestored 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

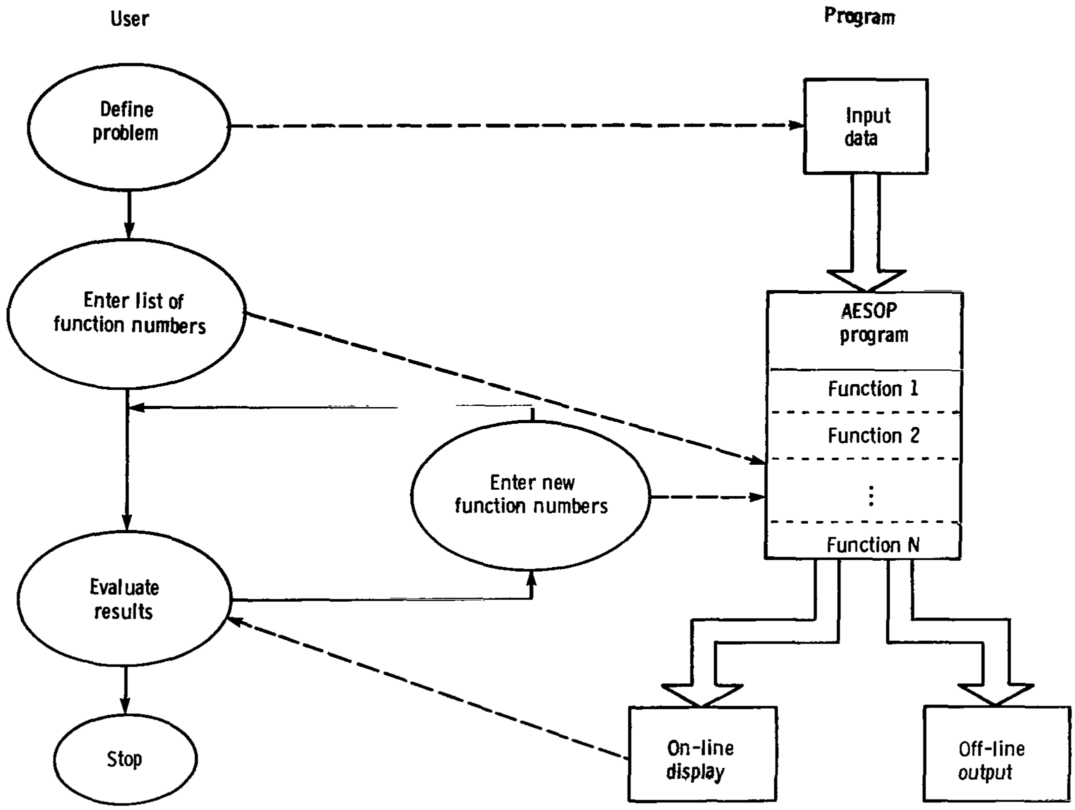


Figure 1. – Overview of AESOP program operation.

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 section 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

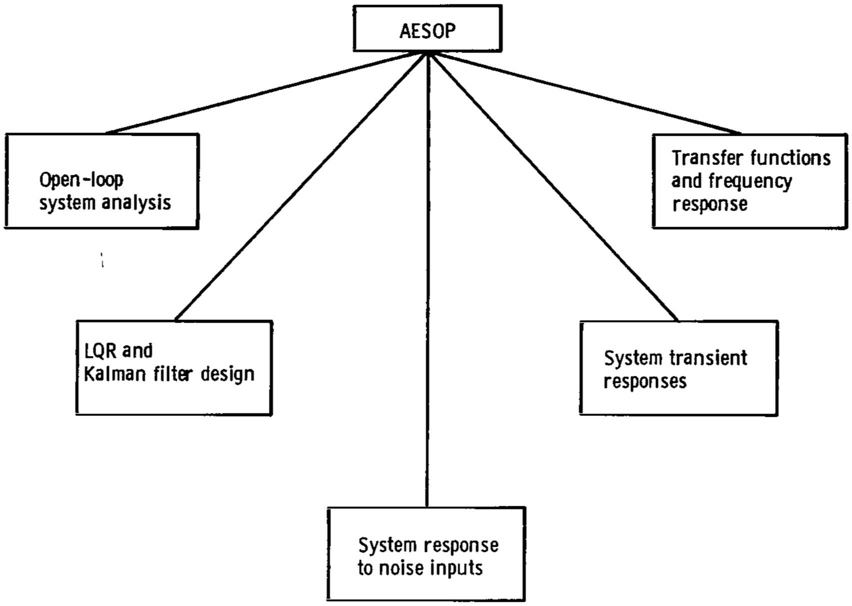


Figure 2. – Control system design computations performed by AESOP program.

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

$$\dot{\mathbf{x}} = \mathbf{A}\ \mathbf{x} + \mathbf{B}\ \mathbf{u} + \mathbf{D}\ \mathbf{w} \tag{1}$$

where

- **x** N^{th}-order state vector
- **u** NC^{th}-order control vector
- **w** ND^{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

$$\mathbf{z} = \mathbf{z}_1 + \mathbf{v} \tag{2a}$$

$$\mathbf{z}_1 = \mathbf{H}\ \mathbf{x} \tag{2b}$$

where

- **z** NM^{th}-order measurement vector
- **v** NM^{th}-order white-noise measurement vector
- **H** NM-by-N matrix

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

$$\mathbf{y} = \mathbf{C}\ \mathbf{x} + \mathbf{DOUT}\ \mathbf{u} \tag{3}$$

where **y** is an NO^{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

$$\mathbf{y}_{sp} = \mathbf{CSP}\ \mathbf{x} \tag{4}$$

where $\mathbf{y}_{sp}$ is an NC^{th}-order set-point vector.

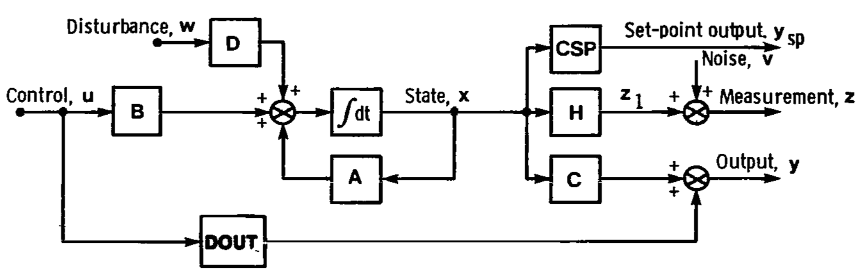


Figure 3. – Block diagram of open-loop system.

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 λ_i ($i=1, 2, \ldots, N$) all have negative real parts. Consider the unforced version of equation (1),

$$\dot{\mathbf{x}}=\mathbf{A}\,\mathbf{x} \tag{5}$$

Define a new state vector $\bar{\mathbf{x}}$, relating to **x** through the transformation matrix **T** as

$$\mathbf{T}\,\bar{\mathbf{x}}=\mathbf{x} \tag{6}$$

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

$$\dot{\bar{\mathbf{x}}}=\mathbf{T}^{-1}\,\mathbf{A}\,\mathbf{T}\,\bar{\mathbf{x}} \tag{7}$$

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

$$\dot{\bar{\mathbf{x}}}=\Lambda\,\bar{\mathbf{x}} \tag{8}$$

The diagonal elements of Λ are the eigenvalues of **A**, **T** is the eigenvector matrix (a matrix whose columns are the eigenvectors of **A**), and $\bar{\mathbf{x}}$ is defined as the *modal* state vector. The value of Λ 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 Λ 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 Λ,

$$\begin{array}{c|c} \alpha & -\beta \\ \hline \beta & \alpha \end{array}$$

where the α'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 Λ there appear two *real* vectors $\mathbf{t}_i$ and $\mathbf{t}_{i+1}$ defined as

$$\mathbf{t}_i=\gamma+\delta$$

$$\mathbf{t}_{i+1}=\gamma-\delta$$

Hence, when Λ 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 vector pair contains, in successive columns, the magnitude of η_i and the phase of η_i. 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 (2a) can be rewritten in terms of the modal state vector as

$$\dot{\bar{\mathbf{x}}}=\Lambda\,\bar{\mathbf{x}}+\mathbf{T}^{-1}\,\mathbf{B}\,\mathbf{u} \tag{9}$$

$$\mathbf{z}=\mathbf{H}\,\mathbf{T}\,\bar{\mathbf{x}}+\mathbf{v} \tag{10}$$

System controllability is determined by examining the elements of $\mathbf{T}^{-1}\,\mathbf{B}$. The system is uncontrollable if elements in a row of $\mathbf{T}^{-1}\,\mathbf{B}$ are zero, meaning that it is impossible to excite a component of the modal state vector $\bar{\mathbf{x}}$ with the control vector **u**. Also, using this modal formulation, one can think of the matrix $\mathbf{T}^{-1}\,\mathbf{B}$ as being the *control effectiveness* matrix. That is, the relative magnitudes of the row elements of $\mathbf{T}^{-1}\,\mathbf{B}$ 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 $\mathbf{z}_k$. 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 ($\mathbf{T}^{-1}\,\mathbf{B}$) and the observability matrix ($\mathbf{H}\ \mathbf{T}$) are printed out in mode-shape format. This means that, for $\mathbf{T}^{-1}\,\mathbf{B}$, when two successive $\underline{\text{rows}}$ k and $k+1$ relate to a complex modal state pair ($\bar{\mathbf{x}}_k$, $\bar{\mathbf{x}}_{k+1}$), the k^{th} elements in the columns of $\mathbf{T}^{-1}\,\mathbf{B}$ are magnitudes and the $(k+1)^{\text{th}}$ elements are phase angles. Similarly, for the $\mathbf{H}\ \mathbf{T}$ matrix, for a complex modal state pair, elements in the k^{th} column of $\mathbf{H}\ \mathbf{T}$ are magnitudes and those in the $(k+1)^{\text{th}}$ column are phase angles.

Residues

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

$$\dot{\bar{\mathbf{x}}}=\mathbf{\Lambda}\ \bar{\mathbf{x}}+\bar{\mathbf{B}}\ \mathbf{u} \tag{10a}$$

$$\mathbf{z}=\bar{\mathbf{H}}\ \bar{\mathbf{x}}+\mathbf{v} \tag{10b}$$

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

$$g(s)=\sum_{j=1}^{\mathrm{N}}\frac{r_j}{s-\lambda_j} \tag{10c}$$

where each of the N constants r_j is defined as a *residue* at the transfer function pole λ_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 $\mathbf{G}(s)$ for the system of equations (10a) and (10b) can be written as

$$\mathbf{G}(s)=\bar{\mathbf{H}}(s\mathbf{I}-\mathbf{\Lambda})^{-1}\,\bar{\mathbf{B}} \tag{10d}$$

or in residue form

$$\mathbf{G}(s)=\sum_{j=1}^{\mathrm{N}}\frac{\mathbf{R}_j}{s-\lambda_j}=\bar{\mathbf{H}}(s\mathbf{I}-\mathbf{\Lambda})^{-1}\,\bar{\mathbf{B}} \tag{10e}$$

where now the N elements $\mathbf{R}_j$ are *residue matrices*. Since $\mathbf{\Lambda}$ is a diagonal matrix, we can rewrite the matrix $(s\mathbf{I}-\mathbf{\Lambda})^{-1}$ as

$$(s\mathbf{I}-\mathbf{\Lambda})^{-1}=\operatorname{diag}\left(\frac{1}{s-\lambda_j}\right)=\sum_{j=1}^{N}\frac{\mathbf{E}_j}{s-\lambda_j} \tag{10f}$$

where

$$\mathbf{E}_j \triangleq \begin{array}{c} \qquad j \\ \qquad \downarrow \\ \begin{bmatrix} 0 & & & & & 0 \\ & 0 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & & 0 & \\ 0 & & & & & \ddots\;0 \end{bmatrix} \end{array} \leftarrow j$$

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

$$\sum_{j=1}^{\mathrm{N}}\frac{\mathbf{R}_j}{s-\lambda_j}=\sum_{j=1}^{\mathrm{N}}\frac{\bar{\mathbf{H}}\ \mathbf{E}_j\ \bar{\mathbf{B}}}{s-\lambda_j} \tag{10g}$$

Thus the j^{th} residue matrix is simply

$$\mathbf{R}_j=\bar{\mathbf{H}}\ \mathbf{E}_j\ \bar{\mathbf{B}} \tag{10h}$$

For a *real* eigenvalue λ_j the elements of the corresponding residue matrix $\mathbf{R}_j$ are real, being computed simply as the (outer) product of the j^{th} *column* of $\bar{\mathbf{H}}$ and the j^{th} *row* of $\bar{\mathbf{B}}$.

For a complex eigenvalue pair (λ_j, λ_{j+1}) AESOP makes use of the modified eigenvector matrix form for $\mathbf{T}$, which means that $\bar{\mathbf{H}}$ and $\bar{\mathbf{B}}$ 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

$$\dot{\mathbf{x}}=\mathbf{A}\ \mathbf{x}+\mathbf{B}\ \mathbf{u} \tag{11}$$

where the state $\mathbf{x}$ is assumed to be measurable and no plant disturbances are present.

A control that minimizes the quadratic performance index

$$J=\int_0^{\infty}\{\mathbf{x^T\,QC\,x}+2\mathbf{x^T\,NN\,u}+\mathbf{u^T(PCINV)^{-1}\,u}\}dt \quad (12)$$

is given by

$$\mathbf{u}=-\mathbf{KC\,x} \quad (13)$$

For the optimal solution to exist, weighting matrices **QC**, **NN**, and **PCINV** must be as follows:

(1) **PCINV** is positive-definite

(2) **QC** can be written as $\mathbf{QC}=\mathbf{M}\,\bar{\mathbf{Q}}\,\mathbf{M^T}$ where the pair **(M,A)** is observable and $\bar{\mathbf{Q}}$ is symmetric and positive-definite

(3) $\mathbf{QC}-\mathbf{NN{\cdot}PCINV{\cdot}NNT}$ is nonnegative-definite.

Feedback gain matrix **KC** is found by solving the following matrix Riccati equation for matrix **SS**:

$$\mathbf{SS(A-B{\cdot}PCINV{\cdot}NNT)+(A-B{\cdot}PCINV{\cdot}NNT)^TSS}$$
$$\mathbf{-SS(B{\cdot}PCINV{\cdot}BT)SS+(QC-NN{\cdot}PCINV{\cdot}NNT)}=0 \quad (14)$$

Then **KC** is given as

$$\mathbf{KC=PCINV(B^T{\cdot}SS+NNT)} \quad (15)$$

Figure 4 shows the structure of the LQR solution. The gain matrix **KC** and the Riccati equation solution matrix **SS** are computed in AESOP by function 801. The closed-loop state equation for the regulator system shown in figure 4 is given by

$$\dot{\mathbf{x}}=\mathbf{(A-B{\cdot}KC)\,x} \quad (16)$$

AESOP uses the eigenvector decomposition method (ref. 8) to solve the Riccati equation, and as a byproduct it prints out both the eigenvalues and eigenvectors of $\mathbf{A-B{\cdot}KC}$.

The Riccati solution matrix **SS** 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 **SS**. The eigenvalues of **SS** are computed and should be positive and real. The differences of the off-diagonals are displayed as a symmetry check. Finally, the computed **SS** 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

$$\mathbf{y_{sp}=CSP\,x} \quad (17)$$

These outputs are to be made equal, in steady state, to NC corresponding *desired* set points $\mathbf{y_{sp}}$. 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

$$\mathbf{u}=-\mathbf{KC\,x}+\mathbf{KFF\,y_{spd}} \quad (18)$$

Figure 5 shows the configuration of the nonzero-set-point regulator. By stipulating that in steady state $\mathbf{y_{sp}=y_{spd}}$, matrix **KFF** can be computed as

$$\mathbf{KFF}=[-\mathbf{CSP\,(A-B{\cdot}KC)^{-1}\,B}]^{-1} \quad (19)$$

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

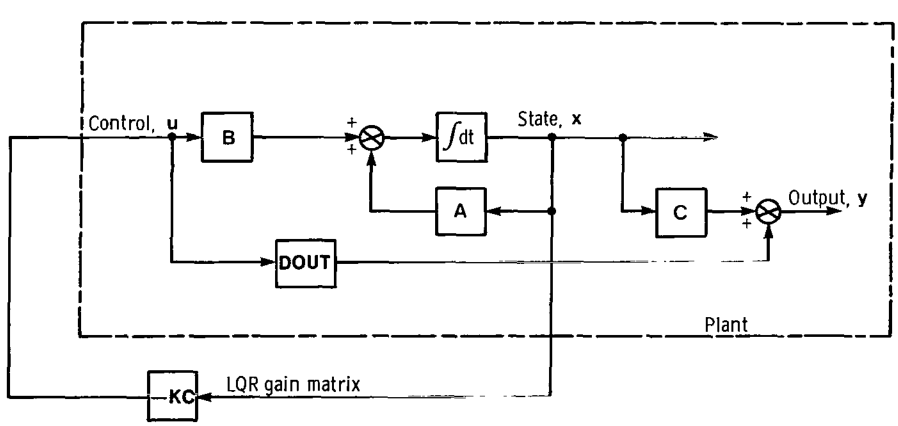


Figure 4. – Block diagram of linear quadratic regulator.

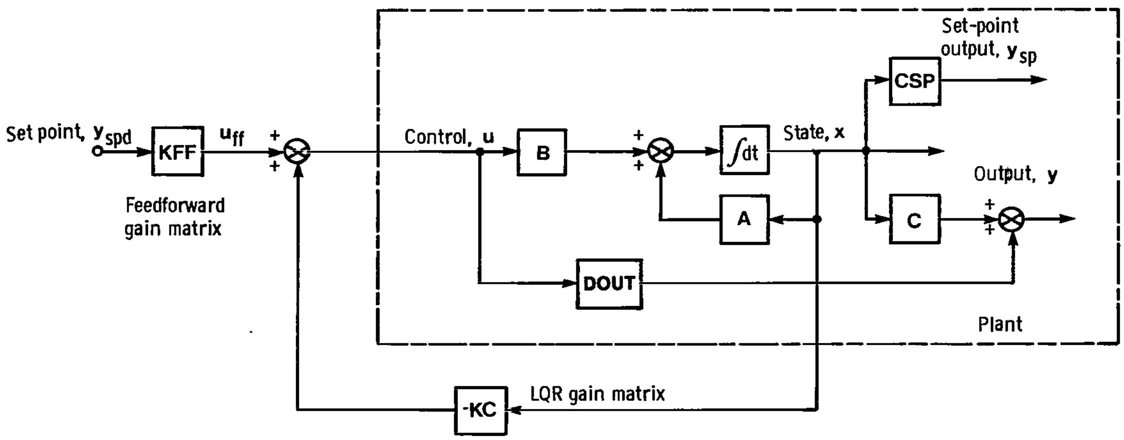


Figure 5. – Block diagram of nonzero-set-point linear regulator.

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

$$E\{\mathbf{w}(t)\mathbf{w}^{\mathrm{T}}\ (t+\tau)\} = \mathbf{Q}\ \delta(\tau) \tag{20}$$

and

$$E\{\mathbf{v}(t)\mathbf{v}^{\mathrm{T}}\ (t+\tau)\} = (\mathbf{RRINV})^{-1}\ \delta(\tau) \tag{21}$$

where matrices **Q** and $(\mathbf{RRINV})^{-1}$ 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

$$\mathbf{QQ} = \mathbf{D\ Q\ D}^{\mathrm{T}} \tag{22}$$

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

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{KE{\cdot}H})\ \ \hat{\mathbf{x}} + \mathbf{B\ u} + \mathbf{KE\ z} \tag{23}$$

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:

$$\mathbf{A{\cdot}PP} + \mathbf{PP{\cdot}A}^{\mathrm{T}} - \mathbf{PP{\cdot}H}^{\mathrm{T}}\mathbf{{\cdot}RRINV{\cdot}H{\cdot}PP} + \mathbf{QQ} = 0 \tag{24}$$

Matrix **PP** is the covariance of estimation error **e**, where $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$. The Kalman gain matrix is computed by using **PP** as

$$\mathbf{KE} = \mathbf{PP{\cdot}H}^{\mathrm{T}}\mathbf{{\cdot}RRINV} \tag{25}$$

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 $\mathbf{A} - \mathbf{KE{\cdot}H}$) 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:

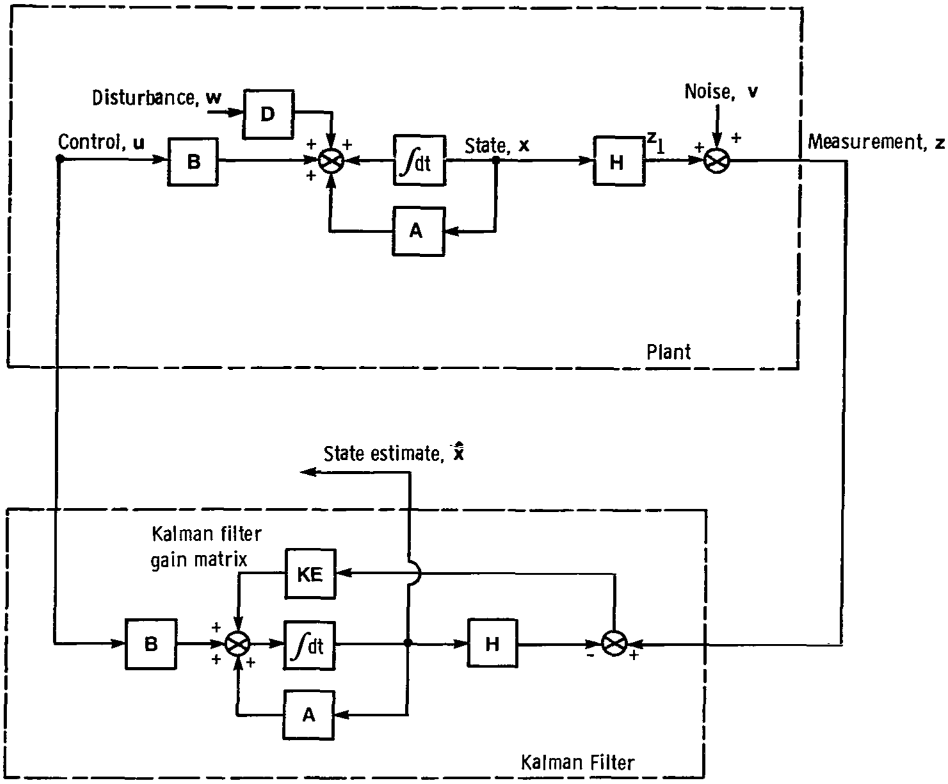


Figure 6. – Block diagram of open-loop system with Kalman filter.

System variable (vector)	Vector of normalization factor
x	**SCX**
u	**SCU**
z	**SCZ**
y	**SCY**
$\mathbf{y}_{sp}$	**SCYSP**

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

$$\mathbf{y}_{sp} = \mathbf{CSP}\ \mathbf{x} \tag{26}$$

Explicit definition of the normalization factors for **x** and $\mathbf{y}_{sp}$ are given by

$$\mathbf{y}_{sp} \triangleq \mathbf{SCYSP}\ \bar{\mathbf{y}}_{sp} \tag{27}$$

$$\mathbf{x} \triangleq \mathbf{SCX}\ \bar{\mathbf{x}} \tag{28}$$

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

$$\bar{\mathbf{y}}_{sp} = \overline{\mathbf{CSP}}\ \bar{\mathbf{x}} \tag{29}$$

where

$$\overline{\mathbf{CSP}} = (\mathbf{SCYSP})^{-1} \cdot \mathbf{CSP} \cdot \mathbf{SCX} \tag{30}$$

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 $(\mathbf{PCINV})^{-1}$ 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 provided 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

$$J = E\{\mathbf{x}^T\mathbf{QC}\,\mathbf{x} + 2\mathbf{x}^T\mathbf{NN}\,\mathbf{u} + \mathbf{u}^T(\mathbf{PCINV})^{-1}\,\mathbf{u}\} \qquad (31)$$

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**, $\hat{\mathbf{x}}$, **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):

$$(\mathbf{A} - \mathbf{B}\cdot\mathbf{KC})\mathbf{XX} + \mathbf{XX}(\mathbf{A} - \mathbf{B}\cdot\mathbf{KC})^T$$
$$+ \mathbf{B}\cdot\mathbf{KC}\cdot\mathbf{PP} + \mathbf{PP}\cdot\mathbf{KC}^T\cdot\mathbf{B}^T + \mathbf{QQ} = 0 \qquad (32)$$

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

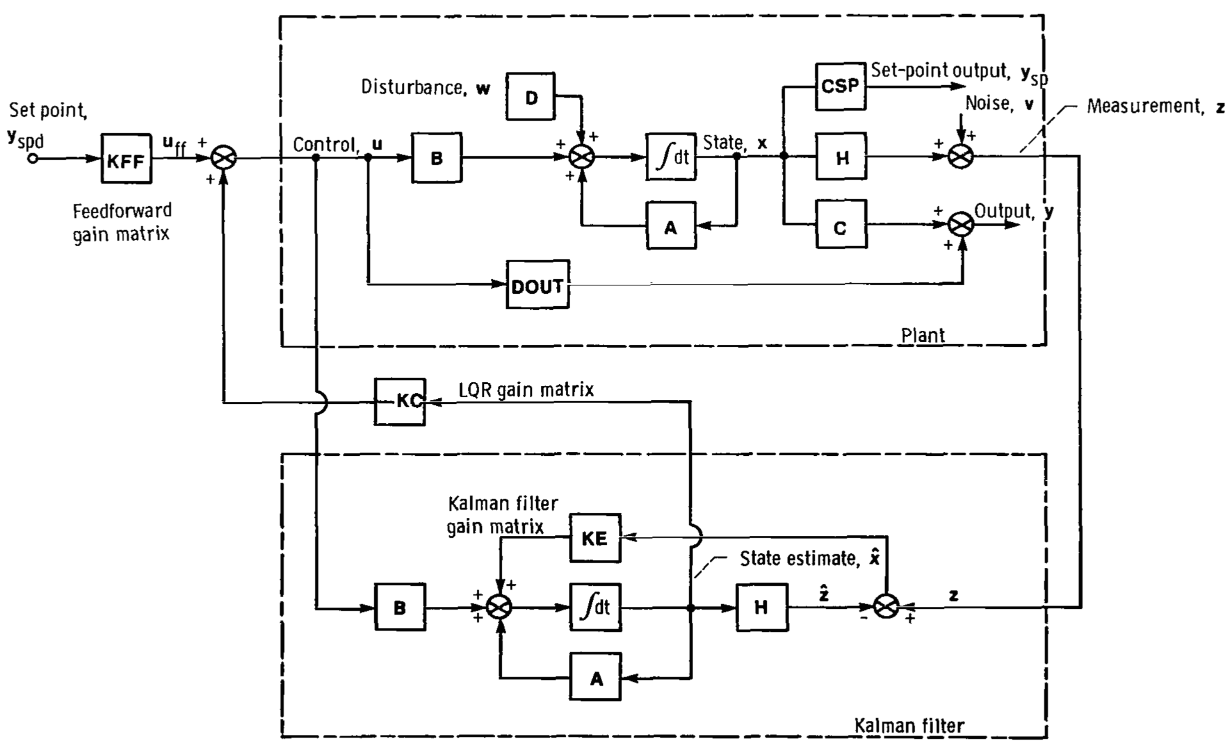


Figure 7. – Stochastic LQR block diagram showing plant, Kalman filter, LQR gain matrix, and feedforward gain matrix.

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**,

$$\mathbf{UU} = \mathbf{KC} \cdot (\mathbf{XX} - \mathbf{PP}) \cdot \mathbf{KC}^{\mathrm{T}} \tag{33}$$

(2) The covariance matrix of measurement component $\mathbf{z}_1$,

$$\mathbf{ZZ} = \mathbf{H} \cdot \mathbf{XX} \cdot \mathbf{H}^{\mathrm{T}} \tag{34}$$

(3) The covariance matrix of output **y**,

$$\mathbf{YY} = \mathbf{C} \cdot \mathbf{PP} \cdot \mathbf{C}^{\mathrm{T}} + (\mathbf{C} - \mathbf{DOUT} \cdot \mathbf{KC}) \cdot (\mathbf{XX} - \mathbf{PP}) \cdot (\mathbf{C} - \mathbf{DOUT} \cdot \mathbf{KC})^{\mathrm{T}} \tag{35}$$

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

$$\mathbf{A}\,\mathbf{X} + \mathbf{X}\,\mathbf{A}^{\mathrm{T}} + \mathbf{W} = 0 \tag{36}$$

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

$$\mathbf{A}\,\bar{\mathbf{X}} + \bar{\mathbf{X}}\,\mathbf{A}^{\mathrm{T}} + \mathbf{W} = \mathbf{R} \tag{37}$$

where **R** is a *residual* matrix. Define an *error* matrix $\mathbf{E} \triangleq \bar{\mathbf{X}} - \mathbf{X}$. Subtracting equation (36) from equation (37), we obtain another Lyapunov equation

$$\mathbf{A}\,\mathbf{E} + \mathbf{E}\,\mathbf{A}^{\mathrm{T}} - \mathbf{R} = 0 \tag{38}$$

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 Trace (**E**)/Trace ($\bar{\mathbf{X}}$).

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 n^{th}-order system described by state equations

$$\dot{\tilde{\mathbf{x}}} = \tilde{\mathbf{A}}\,\tilde{\mathbf{x}} + \tilde{\mathbf{B}}\,\tilde{\mathbf{u}} \tag{39}$$

$$\tilde{\mathbf{y}} = \tilde{\mathbf{C}}\,\tilde{\mathbf{x}} + \tilde{\mathbf{D}}\,\tilde{\mathbf{u}} \tag{40}$$

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 $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}$, and $\tilde{\mathbf{u}}$ and matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{C}}$, and $\tilde{\mathbf{D}}$. The transfer function matrix $\tilde{\mathbf{G}}(s)$ relating output vector $\tilde{\mathbf{y}}$ to input vector $\tilde{\mathbf{u}}$ can be written as

$$\tilde{\mathbf{y}}(s) = [\tilde{\mathbf{C}}(s\mathbf{I} - \tilde{\mathbf{A}})^{-1}\tilde{\mathbf{B}} + \tilde{\mathbf{D}}]\mathbf{u}(s) = \tilde{\mathbf{G}}(s)\tilde{\mathbf{u}}(s) \tag{41}$$

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

The first transfer function form computed is where each $\tilde{\mathbf{G}}_{ij}(s)$ is a ratio of polynomials. In this case the general expression for $\tilde{\mathbf{G}}_{ij}(s)$ is

$$\tilde{\mathbf{G}}_{ij}(s) = \frac{\sum_{k=1}^{n+1} a_{k-1} s^{k-1}}{s^n + \sum_{k=1}^{n} b_{k-1} s^{k-1}} \tag{42}$$

Note that since a feedforward matrix $\tilde{\mathbf{D}}$ is assumed, $\tilde{\mathbf{G}}_{ij}(s)$ may have equal-order (n) numerator and denominator. AESOP uses the technique of reference 15, modified slightly to include a $\tilde{\mathbf{D}}$ matrix, to compute coefficients a_k and b_k. Constants b_k are the coefficients of the characteristic equation of $\tilde{\mathbf{A}}$ and are thus common to all $\tilde{\mathbf{G}}_{ij}(s)$.

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

$$\tilde{\mathbf{G}}_{ij}(s) = \frac{K_{ij} \prod_{k=1}^{m} (s+z_k)}{\prod_{k=1}^{n} (s+p_k)} \tag{43}$$

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 $\tilde{\mathbf{A}}$. The method for obtaining m and z_k depends on the value of $\tilde{\mathbf{D}}_{ij}$.

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

$$\tilde{\mathbf{A}}^* = \left[\tilde{\mathbf{A}} - \frac{\tilde{\mathbf{b}}_j \tilde{\mathbf{c}}_i^{\mathbf{T}}}{\tilde{\mathbf{d}}_{ij}}\right] \tag{44}$$

where

$$\tilde{\mathbf{B}} \triangleq [\tilde{\mathbf{b}}_1 \,\vdots\, \tilde{\mathbf{b}}_2 \,\vdots\, -- \,\vdots\, \tilde{\mathbf{b}}_{nc}]$$

$$\tilde{\mathbf{C}} \triangleq \begin{bmatrix} \tilde{\mathbf{c}}_1^{\mathbf{T}} \\ -- \\ \tilde{\mathbf{c}}_2^{\mathbf{T}} \\ -- \\ \vdots \\ -- \\ \tilde{\mathbf{c}}_{no}^{\mathbf{T}} \end{bmatrix}$$

This fact can be seen by applying a feedback control

$$\tilde{\mathbf{u}}_j = -k^* \mathbf{y}_i \tag{45}$$

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

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

$$K_{ij} = \begin{cases} \tilde{\mathbf{d}}_{ij}, & \tilde{\mathbf{d}}_{ij} \neq 0 \\ \tilde{\mathbf{c}}_i^{\mathbf{T}} \tilde{\mathbf{A}}^{n-m-1} \tilde{\mathbf{b}}_j, & \tilde{\mathbf{d}}_{ij} = 0 \end{cases} \tag{46}$$

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 $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{C}}$, and $\tilde{\mathbf{D}}$ matrices as functions of **A**, **B**, **KE**, **KC**, 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

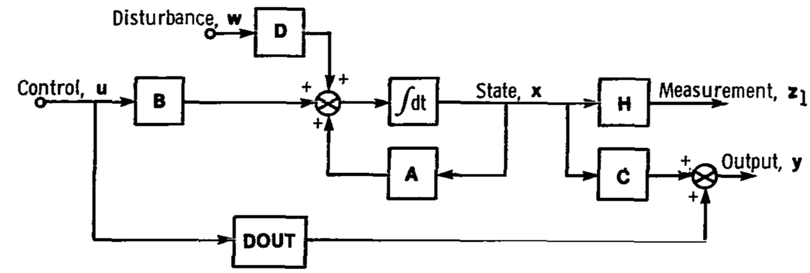


Figure 8. – Block diagram of open-loop system for which transfer functions and frequency responses are obtained. Inputs are u and w; outputs are z_1 and y.

Control to output:

$$\mathbf{y}(s) = [(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} + \mathbf{DOUT}]\mathbf{u}(s) \tag{47}$$

Disturbance to output:

$$\mathbf{y}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{D}\,\mathbf{w}(s) \tag{48}$$

Control to measurement:

$$\mathbf{z}(s) = \mathbf{H}(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B}\,\mathbf{u}(s) \tag{49}$$

Disturbance to measurement:

$$\mathbf{z}(s) = \mathbf{H}(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{D}\,\mathbf{w}(s) \tag{50}$$

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:

$$\mathbf{y}(s) = (\mathbf{C} - \mathbf{DOUT}{\cdot}\mathbf{KC})[(s\mathbf{I} - (\mathbf{A} - \mathbf{B}{\cdot}\mathbf{KC})]^{-1}\,\mathbf{D}\,\mathbf{w}(s) \tag{51}$$

Disturbance to control:

$$\mathbf{u}(s) = -\mathbf{KC}[s\mathbf{I} - (\mathbf{A} - \mathbf{B}{\cdot}\mathbf{KC})]^{-1}\,\mathbf{D}\,\mathbf{w}(s) \tag{52}$$

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 **x̂**. 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:

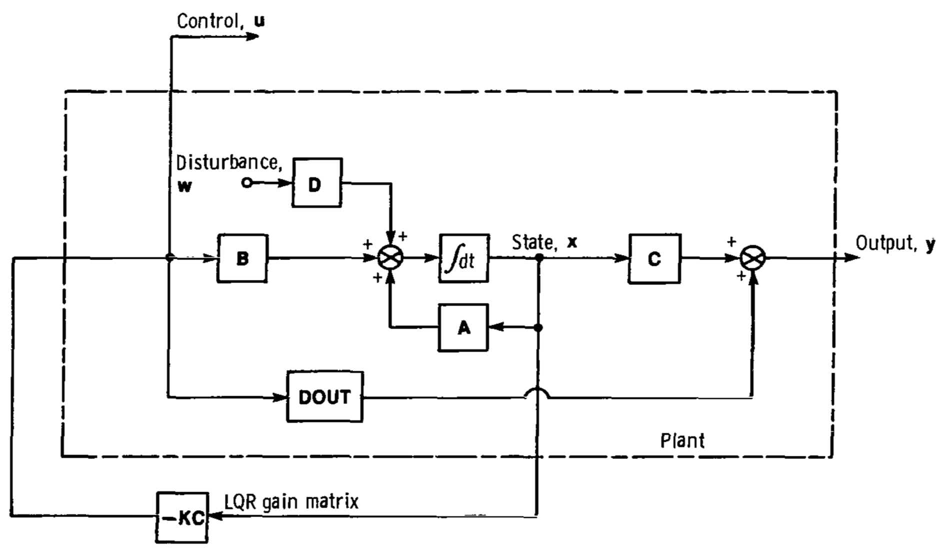


Figure 9. – Block diagram of LQR configuration used for obtaining frequency responses input is w; outputs are y and u.

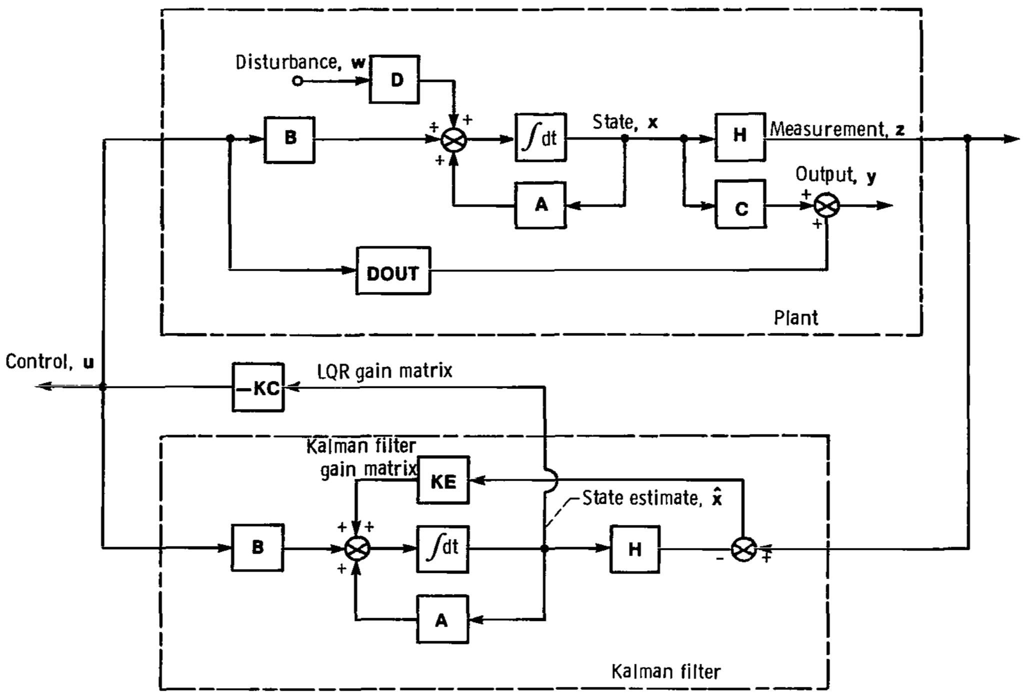


Figure 10. – Block diagram of stochastic LQR configuration used for obtaining frequency responses. Input is *w*; outputs are *y*, *z*, and *u*.

$$\mathbf{y}(s) = \mathbf{CTOT}(s\mathbf{I} - \mathbf{ATOT})^{-1}\,\mathbf{DTOT}\;\mathbf{w}(s) \tag{53}$$

Disturbance to measurement:

$$\mathbf{z}(s) = \mathbf{HTOT}(s\mathbf{I} - \mathbf{ATOT})^{-1}\,\mathbf{DTOT}\;\mathbf{w}(s) \tag{54}$$

Disturbance to control:

$$\mathbf{u}(s) = \mathbf{KCTOT}(s\mathbf{I} - \mathbf{ATOT})^{-1}\,\mathbf{DTOT}\;\mathbf{w}(s) \tag{55}$$

where

$$\mathbf{ATOT} = \left[\begin{array}{c|c} \mathbf{A} & -\mathbf{B}\cdot\mathbf{KC} \\ \hline \mathbf{KE}\cdot\mathbf{H} & \mathbf{A} - \mathbf{B}\cdot\mathbf{KC} - \mathbf{KE}\cdot\mathbf{H} \end{array}\right],\; 2\mathrm{N}\times 2\mathrm{N}$$

$$\mathbf{DTOT} = \left[\begin{array}{c} \mathbf{D} \\ \hline 0 \end{array}\right],\; 2\mathrm{N}\times \mathrm{ND}$$

$$\mathbf{CTOT} = [\mathbf{C} \mid -\mathbf{DOUT}\cdot\mathbf{KC}],\; \mathrm{NO}\times 2\mathrm{N}$$

$$\mathbf{HTOT} = [\mathbf{H} \mid 0],\; \mathrm{NM}\times 2\mathrm{N}$$

$$\mathbf{KCTOT} = [0 \mid -\mathbf{KC}],\; \mathrm{NC}\times 2\mathrm{N}$$

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

$$\mathbf{u}(s) = -\mathbf{KC}[s\mathbf{I} - (\mathbf{A} - \mathbf{B}\cdot\mathbf{KC} - \mathbf{KE}\cdot\mathbf{H})]^{-1}\,\mathbf{KE}\;\mathbf{z}(s) \tag{56}$$

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

$$\dot{\tilde{\mathbf{x}}}(t) = \tilde{\mathbf{A}}\,\tilde{\mathbf{x}}(t) + \tilde{\mathbf{B}}\,\tilde{\mathbf{u}}(t) \tag{57}$$

$$\tilde{\mathbf{y}}(t) = \tilde{\mathbf{C}}\,\tilde{\mathbf{x}}(t) + \tilde{\mathbf{D}}\,\tilde{\mathbf{u}}(t) \tag{58}$$

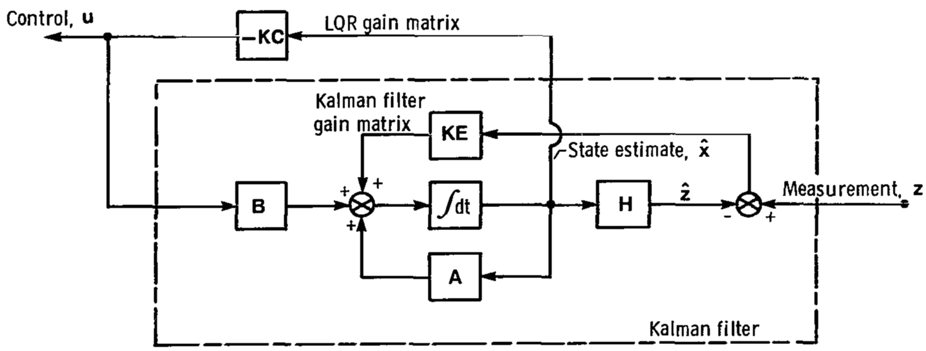


Figure 11. – Optimal controller, comprising Kalman filter and LQR gain matrix. Input is measurement z and output is control u.

In AESOP it was desired to solve these general equations (57) and (58) for two cases: (1) an initial condition $\tilde{\mathbf{x}}(0)$, and (2) a step change in input $\tilde{\mathbf{u}}(t)$. 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

$$\tilde{\mathbf{x}}_{n+1} = \mathbf{\Phi}\tilde{\mathbf{x}}_n + \mathbf{\Gamma}\tilde{\mathbf{u}}_n \tag{59}$$

$$\tilde{\mathbf{y}}_n = \tilde{\mathbf{C}}\,\tilde{\mathbf{x}}_n + \tilde{\mathbf{D}}\,\tilde{\mathbf{u}}_n \tag{60}$$

where

$$\tilde{\mathbf{x}}_{n+1} \triangleq \tilde{\mathbf{x}}(t_{n+1}) \tag{61}$$

$$\tilde{\mathbf{x}}_0 = \tilde{\mathbf{x}}(t_0) = \tilde{\mathbf{x}}(0) \tag{62}$$

and matrices $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ are given by the series representations

$$\mathbf{\Phi}(\mathrm{DT}) = e^{(\tilde{\mathbf{A}}\cdot\mathrm{DT})} = \mathbf{I} + \tilde{\mathbf{A}}\cdot\mathrm{DT} + \frac{\tilde{\mathbf{A}}^2\cdot\mathrm{DT}^2}{2!} + \ldots \tag{63}$$

and

$$\mathbf{\Gamma} = \left[\int_0^{\mathrm{DT}} \mathbf{\Phi}(\tau)d\tau\right]\tilde{\mathbf{B}}$$

$$= \left(\mathbf{I}\cdot\mathrm{DT} + \frac{\tilde{\mathbf{A}}\cdot\mathrm{DT}^2}{2!} + \frac{\tilde{\mathbf{A}}^2\cdot\mathrm{DT}^3}{3!} + \ldots\right)\tilde{\mathbf{B}} \tag{64}$$

Matrix $\mathbf{\Gamma}$ is computed by assuming that input $\tilde{\mathbf{u}}(t)$ 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 $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ 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

$$\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u} \tag{65}$$

$$\mathbf{y} = \mathbf{C}\,\mathbf{x} + \mathbf{DOUT}\,\mathbf{u} \tag{66}$$

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

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\cdot\mathbf{KC})\,\mathbf{x} \tag{67}$$

$$\mathbf{y} = \mathbf{C}\,\mathbf{x} + \mathbf{DOUT}\,\mathbf{u} \tag{68}$$

Matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{C}}$, and $\tilde{\mathbf{D}}$ are appropriately formed by AESOP and variables $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{u} = -\mathbf{KC}\,\mathbf{x}$ are calculated and plotted. This is done for an initial condition $\mathbf{x}_i(0)$ 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:

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\cdot\mathbf{KC})\,\mathbf{x} + \mathbf{B}\cdot\mathbf{KFF}\,\mathbf{y}_{\mathrm{spd}} \tag{69}$$

$$\mathbf{y} = (\mathbf{C} - \mathbf{DOUT}\cdot\mathbf{KC})\,\mathbf{x} + \mathbf{DOUT}\cdot\mathbf{KFF}\,\mathbf{y}_{\mathrm{spd}} \tag{70}$$

$$\mathbf{y}_{\mathrm{sp}} = \mathbf{CSP}\,\mathbf{x} \tag{71}$$

$$\mathbf{u} = -\mathbf{KC}\,\mathbf{x} + \mathbf{KFF}\,\mathbf{y}_{\mathrm{spd}} \tag{72}$$

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 $\mathbf{y}_{spd}$. Of interest to the user in this configuration are such things as rise time and overshoot of the k^{th} component of $\mathbf{y}_{sp}$ in response to a step change in the k^{th} component of $\mathbf{y}_{spd}$; interaction – the response of the k^{th} component of $\mathbf{y}_{sp}$ to a change in the i^{th} component of $\mathbf{y}_{spd}$; and the magnitudes of control variable excursions. The computation and response plotting for the nonzero-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 PROCDEF (for PROcedure DEFinition) AESRUN, defined in appendix D. Through use of TSS/370 datadef (DDEF) statements, AESRUN

(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

AESRUN has a single parameter that is used to identify all output datasets to be generated during the AESOP run. After calling AESRUN, 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 "AESRUN" followed by its parameter. The

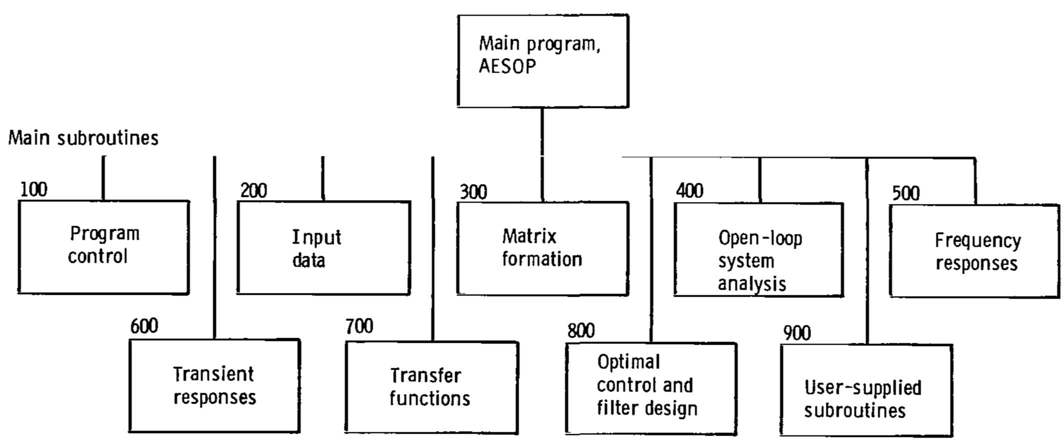


Figure 12. – AESOP program structure.

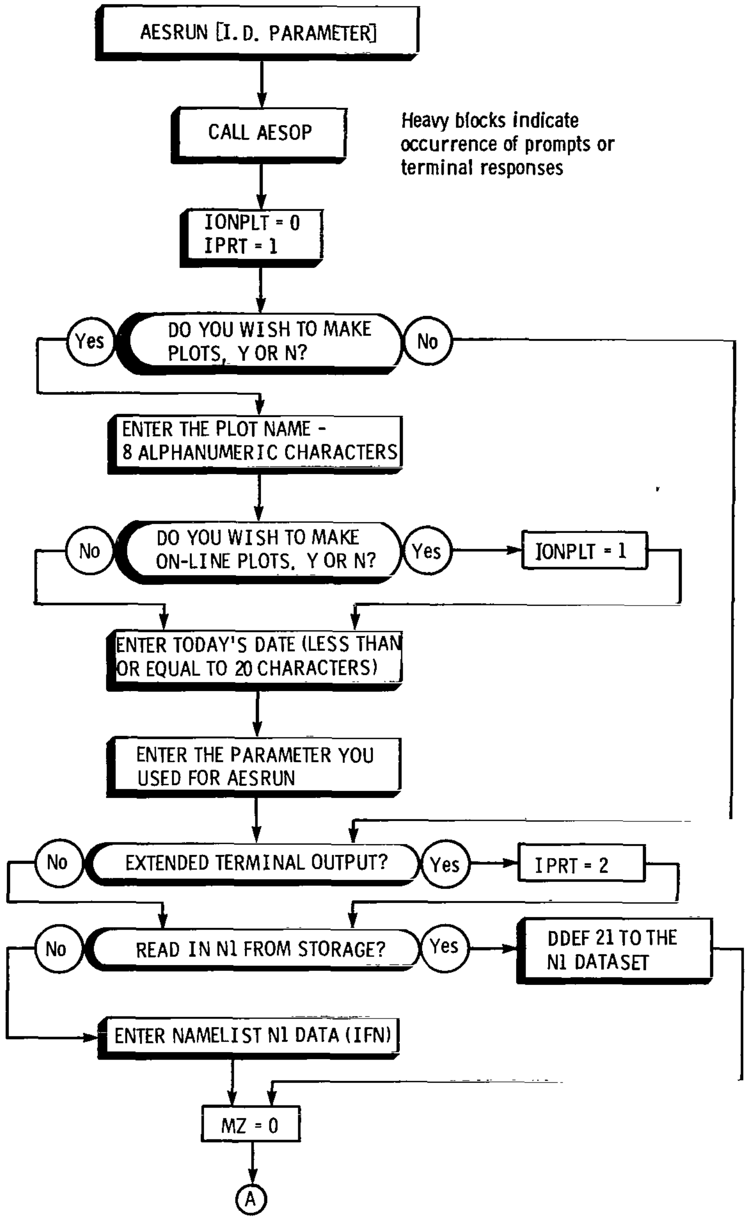


Figure 13. – Flow chart for AESOP program.

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)

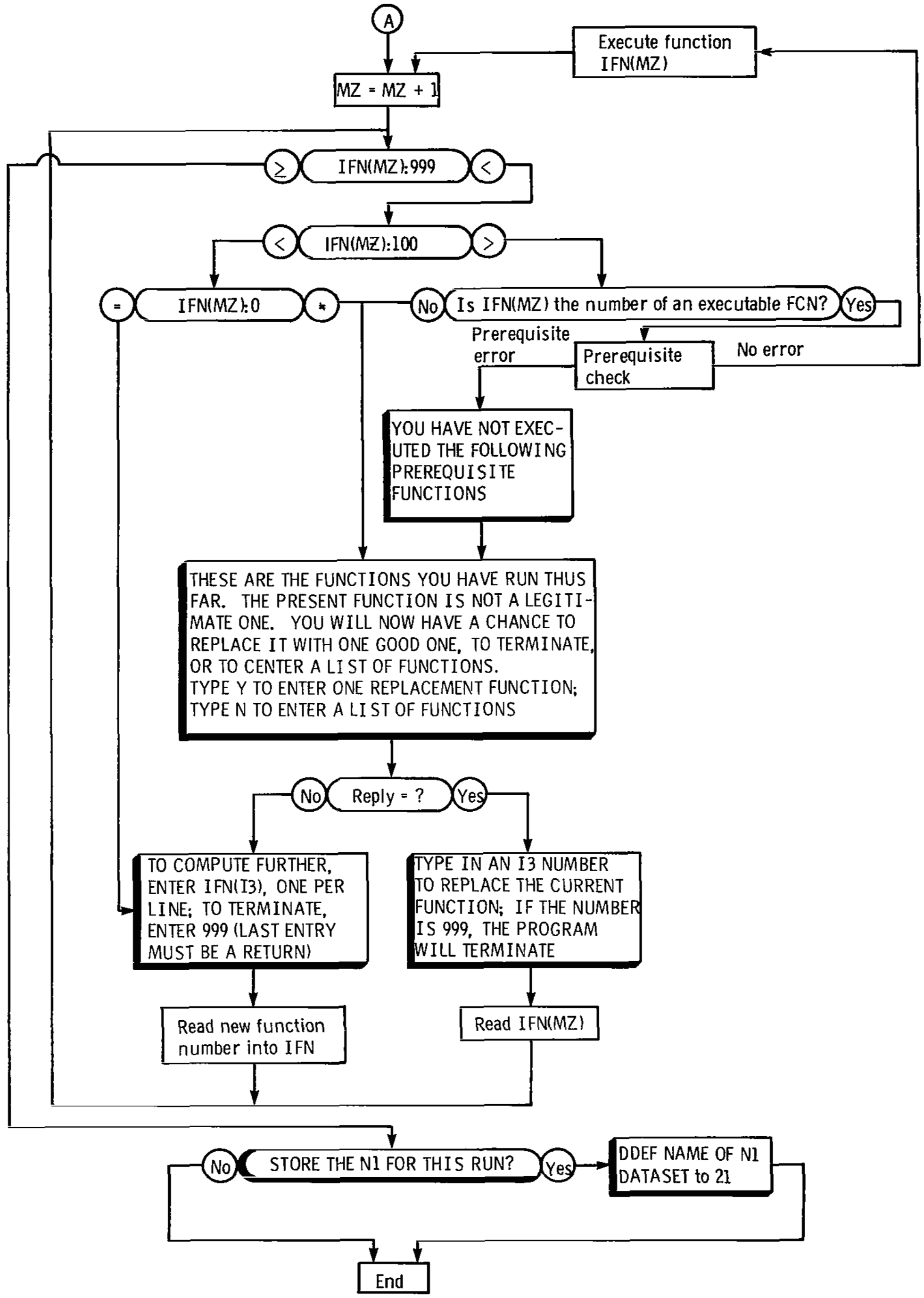


Figure 13. – Concluded.

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 datadeffed 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

&N1 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 '&N1 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)$\geq$999? If so, terminate the program; if not, continue checking.

(2) Is IFN(MZ)$\leq$100 but not equal to zero? If so, you have requested a nonexisting 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 nonexisting 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 AESRUN, 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 **KC** is stored. Other datasets that are identified by using the parameter $1 are those datadeffed 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

Unit	Dataset name	Contents of dataset	Format	Function where read/written	Comments
02	-------	All terminal input and output	Various	Many	Terminal input and output
06	OUT$1[a]	High-speed-printer output	Various	Many	-------------------------
08	CG$1	**KC**, control gain matrix	Unformatted	205/802	-------------------------
09	EG$1	**KE**, Kalman filter gain matrix	↓	206/810	-------------------------
10	PFRUZ$1	Frequency response, **z**(IMEAS)/**u**(JINC)		------------	See table VI for details on frequency responses
11	PFRUY$1	Frequency response, **y**(IOUT)/**u**(JINC)		------------	
12	PFRWZ$1	Frequency response, **z**(IMEAS)/**w**(JIND)		------------	
13	PFRWY$1	Frequency response, **y**(IOUT)/**w**(JIND)		------------	
14	CFR$1	Frequency response, **u**(JINC)/**z**(IMEAS)		------------	
15	PP$1	**PP**, estimation error covariance matrix		208/816	-------------------------
16	SS$1	**SS**, control Riccati solution matrix		207/808	-------------------------
17	FFG$1	**KFF**, feedforward gain matrix	↓	209/820	-------------------------
21	(b)	IFN vector, contained in NAMELIST N1	NAMELIST	Main program	For input of IFN
22	(b)	IFN vector, contained in NAMELIST N1		Main program	For output of IFN
33	(b)	Open-loop plant data, NAMELIST's MATDAT and REFS		202	Usual source of problem data
34	(b)	Normalizing factors; NAMELIST NRMS	↓	404 and 405	-------------------------

[a]$1 is the parameter for PROCDEF AESRUN; it serves as an arbitrary identifying tag (fewer than four characters) for the naming of datasets).
[b]User specified.

TABLE II. - NAMELISTS USED IN AESOP PROGRAM

NAMELIST	Variables in NAMELIST	Input source	Comments
CONPAR	**QC**, **NN**, **PCINV**	Terminal	For revising weighting matrices; read in function 203
ESTPAR	**QQ**, **RRINV**	Terminal	For revising noise power spectral densities; read in function 204
MATDAT	**A**, **B**, **C**, **D**, **H**, **DOUT**, **CSP**, **QC**, **NN**, **PCINV**, **QQ**, **RRINV** N, NM, NC, ND, NO	Terminal or dataset	Primary means of entering system data; read in function 202; changed in function 210
NRMS	SCX, SCU, SCY, SCZ, SCYSP	Dataset	Contains normalizing factors; read in function 404 or 405
N1	IFN	Terminal	Program control; read or written in main program
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	Terminal or dataset	See table III for definition of all variables and default values; set in main program; change in function 101; read in function 202

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 III. - DEFINITION OF PARAMETERS CONTAINED IN NAMELIST REFS

Variable	Dimension	Definition	Default value	Functions where used
DT	Scalar	Time step	0.05 sec	Series 600 (transient responses); functions 601 to 604
ITRMX	Scalar	Desired maximum number of time steps	100	
AMPSR	NCMAX	Open-loop system step input amplitude vector	All elements are set to 1.0	
AMPSP	NCMAX	Closed-loop system step input amplitude vector	All elements are set to 1.0	
AMPICX	NMAX	Initial-condition amplitude vector	All elements are set to 1.0	
MSROLY[a]	NCMAX[b], NOMAX[c]	Input/output selection matrix	All elements are set to 1	
MSROLX[a]	NCMAX[b], NMAX[c]			
MICOLY[a]	NMAX[b], NOMAX[c]			
MICOLX[a]	NMAX[b], NMAX[c]			
MICCLY[a]	NMAX[b], NOMAX[c]			
MICCLX[a]	NMAX[b], NMAX[c]			
MICCLU[a]	NMAX[b], NCMAX[c]			
MSPYSP[a]	NCMAX[b], NCMAX[c]			
MSPY[a]	NCMAX[b], NOMAX[c]			
MSPU[a]	NCMAX[b], NCMAX[c]			
FI	Scalar	Starting frequency for frequency responses	0.01 Hz	Series 500, frequency responses
DELF		Frequency-point spacing	0.02 Hz	
NCURV		= 2, cross plot open- and closed-loop responses; = 1, closed loop only	2	
LINLOG		= 1, linear Bode plots; = 2, log Bode plots; = 3, both linear and log plots	3	
IF		Desired number of frequency response points	49	
ISPACE		Every ISPACE frequency response point is listed	1	
TSFTR		Time scale factor, used for increased precision	1.0	
IOUT	Scalar	Index for selecting component of **y** vector	1	Series 500 (frequency responses) and series 700 (transfer functions)
IMEAS		Index for selecting component of **z** vector	1	
JINC		Index for selecting component of **u** vector	1	
JIND		Index for selecting component of **w** vector	1	
ZERMAX	Scalar	10 times value of largest expected transfer function zero	100 rad/sec	Series 700

[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

Function	Description
Program control	
101	Change reference values
102	Program pause
Inputting or revising matrices	
	Matrix input:
201	Matrices for test case
202	Basic matrix data input
	Revising matrices:
203	**QC**, **NN**, and **PCINV** (for LQR)
204	**QQ** and **RRINV** (for filter)
210	**A**, **B**, **C**, **DOUT**, **H**, **CSP**, **QC**, **NN**, **PCINV**, **QQ**, and **RRINV**
	Reading stored matrices:
205	**KC** (from LQR)
206	**KE** (from filter)
207	**SS** (from LQR)
208	**PP** (from filter)
209	**KFF**
Matrix formation	
301	**A** - **B**•**KC**
302	**A** - **B**•**KC** - **KE**•**H**
303	"Total" system matrices

Function: Compute	Function: Plot	Function: Store	Description
Open-loop system analysis			
401			Eigenvalues
402			Eigenvectors and mode shapes
403			Controllability, observability, and residues
404			Normalization
405			Unnormalization
Frequency responses			
			Open loop:
501	502	503	**u** to **z**
504	505	506	**u** to **y**
507	508	509	**w** to **z**
510	511	512	**w** to **y**
			With state feedback:
513	514		**w** to **y**
515	516		**w** to **u**
			With Kalman filter feedback:
517	518		**w** to **z**
519	520		**w** to **y**
521	522		**w** to **u**
523	524	525	Optimal controller
Transient response			
601			Open-loop step
602			Open-loop initial condition
603			Closed-loop LQR initial condition
604			Nonzero-set-point LQR step

Function	Description
Transient function zeros	
	Open-loop system:
701	**u** to **z**
702	**u** to **y**
703	**w** to **z**
704	**w** to **y**
705	Optimal controller

Function: LQR	Function: Kalman filter	Description
LQR, Kalman filter, and covariances		
801	809	Solve Riccati equation
808	816	Store solution matrix **SS**/**PP**
802	810	Store gain **KC**/**KE**
805	813	Check positive-definiteness of **SS**/**PP**
806	814	Check symmetry of **SS**/**PP**
807	815	Compute residual error

Function	Description
Eigenvalues and eigenvectors	
803	E-values of **A** - **B**•**KC**
811	E-values of **A** - **B**•**KC** - **KE**•**H**
812	E-values of **ATOT**
804	E-vectors of **A** - **B**•**KC**
Covariances, etc.	
817	Covariances of LQR system
818	Covariance error check
819	Feedforward gain calculation (**KFF**)
820	Store **KFF**
User-supplied subroutines	
901	UZR901
902	UZR902
903	UZR903
904	UZR904

TABLE V. - INPUT MATRICES FOR THIRD-ORDER TEST CASE

A =	Input matrix			CSP =	Input matrix		
	1	2	3		1	2	3
1	-0.1000D-00	1.000	0.0000	1	1.000	0.0000	0.0000
2	0.0000	0.0000	1.000	2	0.0000	0.0000	2.000
3	0.0000	-1.000	-0.2000D-07				
B =	1	2		QC =	1	2	3
1	0.0000	0.0000		1	500.0	0.0000	0.0000
2	0.0000	1.000		2	0.0000	9.000	0.0000
3	1.000	0.0000		3	0.0000	0.0000	0.4000D-07
D =	1	2		NN =	1	2	
1	0.0000	0.0000		1	40.00	0.0000	
2	0.0000	1.000		2	0.0000	81.00	
3	1.000	0.0000		3	32.00	0.0000	
C =	1	2	3	PCINV =	1	2	
1	1.000	0.0000	0.0000	1	0.1563D-01	0.0000	
2	0.0000	0.0000	1.000	2	0.0000	0.1235D-02	
H =	1	2	3	QQ =	1	2	3
1	1.000	0.0000	0.0000	1	0.0000	0.0000	0.0000
				2	0.0000	2.000	0.0000
				3	0.0000	0.0000	20.00
DOUT =	1	2		RRINV =	1		
1	0.0000	0.0000		1	1.000		
2	0.0000	1.000					

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 **KC** from dataset CG$1 ($1 is the parameter in PROCDEF AESRUN). Function 206 reads Kalman filter gain matrix **KE** 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 ($\alpha \pm j\beta$) form and polar (frequency and ζ) where the frequency is in hertz and ζ is the damping ratio, $\zeta \triangleq -\cos(\arctan |\beta|/\alpha)$. 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

Function	Matrix formed	Dimension	Equation	Function where used
301	AMBKC	NxN	AMBKC = A - B•KC	513,515,603, 604,803,819
302	ABKCEH	NxN	ABKCEH = A - B•KC - KE•H	523,705,811
303	ATOT,DTOT,CTOT, KCTOT,HTOT	(a)	(a)	517,519,521,812

$$^{a}\text{ATOT} = \left[\begin{array}{c|c} \text{A} & -\text{B}\bullet\text{KC} \\ \hline \text{KE}\bullet\text{H} & \text{A} - \text{B}\bullet\text{KC} - \text{KE}\bullet\text{H} \end{array}\right], \quad 2\text{N}\times 2\text{N}$$

$$\text{DTOT} = \left[\begin{array}{c} \text{D} \\ \hline 0 \end{array}\right], \quad 2\text{N}\times\text{ND}$$

$$\text{CTOT} = [\text{C} \mid -\text{DOUT}\bullet\text{KC}], \quad \text{NO}\times 2\text{N}$$

$$\text{HTOT} = [\text{H} \mid 0], \quad \text{NM}\times 2\text{N}$$

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

403 – Controllability, 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 ($\mathbf{T}^{-1}\mathbf{B}$) and two system observability matrices (**H T** and **H C**). Also, this function computes the residues for systems (**A**, **B**, **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

System variable (vector)	Vector of normalization factor	Size
x	**SCX**	50
u	**SCU**	5
z	**SCZ**	5
y	**SCY**	50
$\mathbf{y}_{sp}$	**SCYSP**	5

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., 3.,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:

Configuration	Figure
Open loop	8
State-variable feedback	9
Kalman filter feedback	10
Optimal controller only	11

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

System configuration	Input variable		Output variable		AESOP program function numbers				
					Transfer functions		Frequency responses		
	Name	Index[a]	Name	Index[a]	Numerator zeros and gain	Poles	Magnitude, phase, and coefficients[b]	Response plots	Store magnitude and phase
Open loop	u	JINC	z	IMEAS	701	401	501	502	503
	u	JINC	y	IOUT	702		504	505	506
	w	JIND	z	IMEAS	703		507	508	509
			y	IOUT	704		510	511	512
State-variable feedback			y	IOUT	---	803	513	[c]514	---
State-variable feedback			u	JINC	---	803	515	516	---
Kalman filter feedback			z	IMEAS	---	812	517	[c]518	---
Kalman filter feedback			y	IOUT	---	812	519	[c]520	---
Kalman filter feedback			u	JINC	---	812	521	522	---
Optimal controller only	z	IMEAS	u	JINC	705	811	523	524	525

[a]Indices 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.
[b]Computes (1) coefficients of the transfer function numerator and denominator polynomials and (2) frequency response magnitudes and phase angles for initial frequency FI(Hz), frequency spacing DELF(Hz), number of points IF($\leq$1000), and a time scale factor TSFTR.
[c]If 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.

REFS. It can be seen that transfer function numerator and denominator polynomial coefficients (eq. (42)), frequency response calculations, storing of frequency responses, and Bode plotting are available for all four system configurations. Specific function numbers 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 = I,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_ _ _ vectors

(2) Desired time step, DT

(3) The number of time points to be computed, ITRMX ≤ 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

TABLE VIII. - AESOP FUNCTIONS FOR COMPUTING TRANSIENT RESPONSES

Function	Type of transient	Input variable	Vector containing input amplitudes	Output variables	Input/output select matrix[a]
601	Step response of open-loop system	**u**	AMPSR	**y** **x**	MSROLY MSROLX
602	Initial-condition response of open-loop system	**x**(0)	AMPICX	**y** **x**	MICOLY MICOLX
603	Initial-condition response of linear quadratic regulator	**x**(0)	AMPICX	**y** **x** **u**	MICCLY MICCLX MICCLU
604	Step response of nonzero-set-point regulator	y_{spd}	AMPSP	y_{sp} **y** **u**	MSPYSP MSPY MSPU

[a]Use of input/output select matrices:

M XXXX (input index, output index) = 1, if response is to be calculated and plotted
0, otherwise

That is, for function 601, if response of **y**(3) to **u**(2) is desired, set MSROLY(2,3) = 1 or, for function 602, if response of **y**(1) to an initial condition on **x**(2) is desired, set MICOLY(2,1) = 1. All amplitude vectors and input/output select matrices are defaulted to all ones. They can be changed via NAMELIST REFS. All responses are for only one input or initial-condition component applied. Default for time step DT is 0.01 and that for time points ITRMX is 100. They can be changed via NAMELIST REFS.

TABLE IX. - RICCATI EQUATION SOLUTION

Task	LQR design problem	Kalman filter design problem
	Function	
Solve matrix Riccati equation:		
Number of function	801	809
Name of solution matrix	SS	PP
Name of gain matrix	KC	KE
Store Riccati results in dataset:		
Function for storing solution matrix	808	816
Function for storing gain matrix	802	810
Riccati equation accuracy checks performed on solution matrix:		
Positive-definiteness	805	813
Symmetry	806	814
Residual error matrix	807	815

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,N),J = 1,NM)

(5) For reading the feedforward gains (stored in dataset FFG$1):

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**·**KC** are those of the linear regulator. The eigenvalues of matrix **A** – **B**·**KC** – **KE**·**H** are those of the optimal controller (depicted in fig. 11). The eigenvalues of matrix **ATOT** are those of **A** – **B**·**KC** plus those of **A** – **KE**·**H**, the latter being the Kalman filter eigenvalues.

TABLE X. – EIGENVALUE AND EIGENVECTOR COMPUTATIONS IN SERIES 800

Matrix	Eigenvalue	Eigenvector
	Function	
A - B•KC	803	804
A - B•KC - KE•H	811	---
ATOT [a]	812	---

[a]Matrix defined in table VI.

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.

LQR gain matrix, **KC**	Kalman gain matrix, **KE**	System configuration for which **XX** is covariance matrix	Figure
≠0	≠0	Linear stochastic regulator	7
=0	=0	Open-loop system with noise input	3
≠0	=0	State feedback system with noise input	9
=0	≠0	Open-loop system with Kalman filter	6

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

$$\mathbf{KFF} = [-\mathbf{CSP}(\mathbf{A} - \mathbf{B}\cdot\mathbf{KC})^{-1}\,\mathbf{B}]^{-1}$$

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.

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

MICCLU	N×NC	integer, input/output, closed-loop, initial-condition response selection matrix for controls; state feedback
MICCLX	N×N	integer, input/output, closed-loop, initial-condition response selection matrix for states; state feedback
MICCLY	N×NO	integer, input/output, closed-loop, initial-condition response selection matrix for outputs; state feedback
MICOLX	N×N	integer, input/output, open-loop, initial-condition response selection matrix for states
MICOLY	N×NO	integer, input/output, open-loop, initial-condition response selection matrix for outputs
MSPU	NC×NC	integer, input/output, closed-loop, step response selection matrix for controls; state feedback
MSPY	NC×NO	integer, input/output, closed-loop, step response selection matrix for outputs; state feedback
MSPYSP	NC×NC	integer, input/output, closed-loop, step response selection matrix for set-point outputs; state feedback
MSROLX	NC×N	integer, input/output, open-loop, step response selection matrix for states
MSROLY	NC×NO	integer, input/output, open-loop, step response selection matrix for outputs
M	N×ℓ	matrix, general factor of **QC** matrix
N		integer, number of states
NC		integer, number of controls
NCURV		integer, controls cross plotting of frequency responses
ND		integer, number of disturbances
NM		integer, number of measurements
NMAX		integer, maximum number of states
NN	N×NC	state-control weighting matrix
NO		integer, number of outputs
NTOT		integer, 2×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×NC	inverse of control weighting matrix
PP	N×N	Kalman filter error covariance matrix
p_k		transfer function pole
$\bar{\mathbf{Q}}$	$\ell\times\ell$	symmetric, positive-definite matrix
QC	N×N	state weighting matrix
QQ	N×N	power spectral density matrix of plant disturbance
$\mathbf{R}_{ij}$	NM×NC	residue matrix of transfer function matrix $\mathbf{G}(s)$
R	N×N	residual matrix
RRINV	NM×NM	inverse of power spectral density matrix of measurement noise
r_j		j^{th} 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×N	Riccati solution matrix for linear quadratic regulator
s		Laplace variable

$\mathbf{T}$	N×N	modified eigenvector matrix
TSFTR		time scale factor
t		time
t_i, t_{i+1}	N	modified eigenvectors
UU	NC×NC	control covariance matrix
$\mathbf{u}$	NC	control vector
$\mathbf{v}$	NM	measurement noise vector
$\mathbf{W}$	N×N	general matrix
$\mathbf{w}$	ND	disturbance vector
$\mathbf{X}$	N×N	general matrix
$\bar{\mathbf{X}}$	N×N	computed value of matrix $\mathbf{X}$
XX	N×N	state covariance matrix
$\mathbf{x}$	N	state vector
$\bar{\bar{\mathbf{x}}}$	N	modal state vector
$\hat{\mathbf{x}}$	N	estimated state vector
YY	NO×NO	output covariance matrix
$\mathbf{y}$	NO	output vector
$\mathbf{y}_{sp}$	NC	set-point vector
$\mathbf{y}_{spd}$	NC	desired set-point vector
ZERMAX		maximum expected value of transfer function zeroes
ZZ	NM×NM	measurement covariance matrix
$\mathbf{z}$	NM	measurement vector
$\mathbf{z}_1$	NM	noise-free component of vector $\mathbf{z}$
$\hat{\mathbf{z}}$	NM	estimated value of $\mathbf{z}$
z_k		transfer function numerator zero
α		real part of eigenvalue
β		imaginary part of eigenvalue
Γ	N×NC	forced-response matrix for discrete system
γ	N	real part of eigenvector
δ	N	imaginary part of eigenvector
$\delta(\tau)$		unit impulse function
ζ		damping ratio
η	N	eigenvector
$\mathbf{\Lambda}$	N×N	block diagonal form of $\mathbf{\Lambda}$ matrix
λ_i, λ_{i+1}		complex eigenvalue pair
τ		time
$\mathbf{\Phi}$	N×N	state-transition matrix

Superscripts:

$\mathbf{T}$		matrix transpose
-1		matrix inverse

Operators:

$E\{\ \}$		expected value of

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:

$N \leq 50$

$NM \leq 5$

$NC \leq 5$

$ND \leq 15$

$NO \leq 50$

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;
                            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 AES200 (IFN, IFUNC, IAND, MZ, IPRT, WHEN)

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

SUBROUTINE AES200 CONTAINS THOSE FUNCTIONS WHICH INPUT DATA.
AES200 CALLS SUBROUTINES MATCHG, MATIN, MATPRT, 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, EGVCTR, 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, AND THE
FEED FORWARD, RICCATI, AND COVARIANCE EQUATIONS.
AES800 CALLS SUBROUTINES CONTRL, COVAR, EGVCTR, EIGEN, ESTMAT,
LYPCK, MATPRT, MODSHP, MXINV, 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 DAVISO 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, IPERN, 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    INVERSE 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,
1 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. XX, 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) (OF PLANT DISTURBANCE)
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,
1 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)
                (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)
                (IN MAGNITUDE AND PHASE ANGLE FORM)
        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)=TAT(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.

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 (AAA, 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.

INPUTS:

AAA	MATRIX FOR WHICH EIGENVECTORS ARE TO BE OBTAINED (N2,N2)
CPR	VECTOR OF REAL PARTS OF EIGENVALUES (N2) (OF AAA)
CPI	VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N2) (OF AAA)
N2	ACTUAL SIZE OF MATRIX AAA
IOP2	PRINT OPTION: 0, NO PRINT; 1, PRINT
N2MAX	MAXIMUM SIZE OF N2
ISEL	SELECTION OPTION
IHALF	SELECTION OPTION

OUTPUTS:

X	MODIFIED EIGENVECTOR MATRIX OF AAA (N2,N2)

TEMPORARY STORAGE:

TT	MATRIX (N2,N2)
EXT	MATRIX (N2,N2)
AR	VECTOR (N2)
AI	VECTOR (N2)
IPERN	INTEGER VECTOR (N2)
IPER	INTEGER VECTOR (N2)

****

```
 SUBROUTINE EIGEN (A, EIGR, EIGI, EX1, SSS, S, IA, IB, LEX, MEX,
1 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, NO WRITING TAKES PLACE
                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, ADBLE, 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 AES800.

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 (Z1, Z2, DD, HZ, G, 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
```

```
        AMP     AMPLITUDE
        PHA     PHASE
```

****

```
 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	IOUTTH COLUMN OF C (N)
DDCOF	DENOMINATOR COEFFICIENTS (N)
DNCOF	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, II, 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 II 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(II)
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)
        IN      MAXIMUM SIZE OF MATRIX A IN THE CALLING PROGRAM;
                IN = N, WHEN MATRIX A IS IN VECTOR STORAGE MODE.

OUTPUTS:

        A       OUTPUT MATRIX (N,N)

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

 SUBROUTINE ICRSP (EX1, C, ICMTX, AMPIN, DT, TIME, TYOUT, XNEW,
1 XOLD, TTIT, TTOP, TYTIT, IEXT, N, NOUT, NMAX, NOUTMX, ITRMX, IP,
2 NAME, IONPLT)

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

SUBROUTINE ICRSP COMPUTES MULTIPLE INITIAL CONDITION RESPONSES OF
THE SYSTEM:   XDOT=A*X   AND   TYOUT=C*X
BY SOLVING THE DIFFERENCE EQUATION:   XNEW=EX1*XOLD
THIS SUBROUTINE REQUIRES THAT THE STATE TRANSITION MATRIX,
EXP(A*DT), BE SUPPLIED AS INPUT MATRIX "EX1".  DESIRED INITIAL
CONDITION MAGNITUDES ARE SUPPLIED AS VECTOR 'AMPIN' AND THE
DESIRED INITIAL CONDITION-OUTPUT RESPONSE COMBINATIONS ARE
SELECTED BY APPROPRIATELY DEFINING ELEMENTS OF THE MATRIX 'ICMTX'.
ICRSP CALLS PLOTTING SUBROUTINES ONLY.  ICRSP IS CALLED BY
SUBROUTINE AES600.

INPUTS:

        EX1     STATE TRANSITION, EXP(A*DT), MATRIX (N,N)
        C       SYSTEM OUTPUT MATRIX (NOUT,N)
        ICMTX   MATRIX OF ZEROES AND ONES (N,NOUT).
                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*NOUT INITIAL CONDITION RESPONSES.
        AMPIN   VECTOR OF INPUT INITIAL CONDITION AMPLITUDES (N)
        DT      TIME STEP
        TTIT    PLOT TITLE (12)
        TTOP    PLOT TITLE (12)
        TYTIT   Y AXIS TITLE (4)
        N       ACTUAL SIZE OF STATE TRANSITION MATRIX
        NOUT    ACTUAL NUMBER OF POSSIBLE OUTPUTS
        NMAX    MAXIMUM SIZE OF N
        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 INITIAL CONTIDION. (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)
        IEXT    INTEGER VECTOR (N)
*****************************************************************
```

```
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, MXMLT, 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*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 STOGAGE 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:

        A       MATRIX TO BE TRANSPOSED (N,M)
        N       NUMBERS OF ROWS IN MATRIX A
                AND COLUMNS IN MATRIX R
        M       NUMBERS OF COLUMNS IN MATRIX A
                AND ROWS IN MATRIX R

OUTPUTS:

        R       RESULTANT MATRIX (M,N)

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



 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:

        A       UN-NORMALIZED SYSTEM MATRIX (N,N)
        B       UN-NORMALIZED CONTROL INPUT MATRIX (N,NC)
        C       UN-NORMALIZED OUTPUT MATRIX (NO,N)
        H       UN-NORMALIZED MEASUREMENT MATRIX (NM,N)
        Q       UN-NORMALIZED POWER SPECTRAL DENSITY MATRIX OF
                PLANT DISTURBANCE (N,N)
        RINV    UN-NORMALIZED INVERSE OF POWER SPECTRAL DENSITY
                MATRIX OF MEASUREMENT NOISE (NM,NM)
        D       UN-NORMALIZED DISTURBANCE INPUT MATRIX (N,ND)
        DOUT    UN-NORMALIZED FEED FORWARD MATRIX FOR
                NON-ZERO SET POINT REGULATOR (NO,NC)
        CSP     UN-NORMALIZED SET POINT OUTPUT MATRIX (NC,N)
        N       ACTUAL NUMBER OF STATES
        NC      ACTUAL NUMBER OF CONTROL INPUTS
        NO      ACTUAL NUMBER OF OUTPUTS
        NM      ACTUAL NUMBER OF MEASUREMENTS
        ND      ACTUAL NUMBER OF DISTURBANCE INPUTS
        NMAX    MAXIMUM SIZE OF N
        NCMAX   MAXIMUM SIZE OF NC
        NOMAX   MAXIMUM SIZE OF NO
        NMMAX   MAXIMUM SIZE OF NM
        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:

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)
RINV	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) (OF MATRIX A)
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 ANY SUBROUTINES. POLMPY IS CALLED BY SUBROUTINE DANSKY.

INPUTS:

X	POLYNOMIAL COEFFIENT VECTOR (NX)
Y	POLYNOMIAL COEFFIENT 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, II)

****

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
II	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 HARARYS 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,  XDOT=A*X + B*U,    AND,   Y=C*X
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 (NO,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 (NO,NC)  TEMPORARILY STORES EACH RESIDUE
                MATRIX BEFORE BEING PRINTED OUT

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



SUBROUTINE RICCHK (AAA, S, R, N, NMAX, N2MAX)

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

SUBROUTINE RICCHK COMPUTES THE RESIDUAL ERROR MATRIX FOR THE
RICCATI EQUATION
             S*AAA(22)*T + AAA(22)*S - S*AAA(12)*S + AAA(21) = 0
   OR,      -S*AAA(11) - AAA(11)*T*S - S*AAA(12)*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*AAA(22)*T + AAA(22)*S - S*AAA(12)*S + AAA(21)
=-(S*AAA(11) + AAA(11)*T*S - S*AAA(12)*S - AAA(21))
RICCHK CALLS SUBROUTINE MATPRT.  RICCHK IS CALLED BY SUBROUTINE
AES800.

INPUTS:

        AAA     HAMILTONIAN MATRIX (2 X N,2 X N)
        S       RICCATI SOLUTION MATRIX (N,N)
        N       ACTUAL SIZE OF MATRIX S
        NMAX    MAXIMUM SIZE OF N
        N2MAX   2 X NMAX
```

OUTPUTS:

R	RESIDUAL ERROR MATRIX (N,N)

****

SUBROUTINE RICSS (AAA, X, OUTPUT, CR, CI, TS, XR, EXT, TT, IPER,
1 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) (OF AAA)
CI	VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N2) (OF AAA)
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 (EX1, EX2, B, C, DOUT, IOMTX, AMPIN, DT, TIME,
1 TYOUT, XNEW, XOLD, ANR, TTIT, TTOP, TYTIT, IEXT, N, NIN, NOUT,
2 NMAX, NINMAX, NOUTMX, ITRMX, IP, NAME, IONPLT)
```

****

SUBROUTINE STP COMPUTES MULTIPLE STEP RESPONSES OF THE SYSTEM
      XDOT=A*X+B*U; TYOUT=C*X+DOUT*U
BY SOLVING THE DIFFERENCE EQ.
      XNEW=EX1*XOLD + EX2*B*AMPIN(L)
THIS SUBROUTINE REQUIRES THAT THE STATE TRANSITION MATRIX, EXP(A*DT), AND ITS INTEGRAL FROM TIME=0 TO TIME=DT, BE SUPPLIED AS INPUT MATRICES 'EX1' AND 'EX2'. DESIRED INPUT STEP MAGNITUDES ARE SUPPLIED AS VECTOR 'AMPIN' AND THE DESIRED STEP INPUT-OUTPUT RESPONSE COMBINATIONS ARE SELECTED BY APPROPRIATELY DEFINING ELEMENTS OF THE MATRIX 'IOMTX'. STP CALLS PLOTTING SUBROUTINES ONLY. STP IS CALLED BY SUBROUTINE AES600.

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 ZEROES 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 UZR901

****

THIS 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,
1 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(II) / U(JJ) = CC * ((S * I - A ) ** -1 ) * BB
II DENOTES DESIRED COMPONENT OF OUTPUT VECTOR
JJ DENOTES DESIRED COMPONENT OF INPUT VECTOR
ZEROES CALLS SUBROUTINES CONDI, EIGQR, HSBG, AND ARRAY.  ZEROES IS
CALLED BY SUBROUTINE AES700.

INPUTS:

         AA       SYSTEM MATRIX (N,N)
         BB       INPUT MATRIX (N,NUMBER OF POSSIBLE INPUTS)
         CC       OUTPUT MATRIX (NUMBER OF POSSIBLE OUTPUTS,N)
         DD       CONSTANT
         CONST    ITERATION CONSTANT
         N        ACTUAL SIZE OF MATRIX AA
         II       OUTPUT COMPONENT
         JJ       INPUT COMPONENT
         NMAX     MAXIMUM SIZE OF N
         NOMAX    MAXIMUM NUMBER OF OUTPUTS

OUTPUTS:

         ANR      VECTOR OF REAL PARTS OF EIGENVALUES (N)
         ANI      VECTOR OF IMAGINARY PARTS OF EIGENVALUES (N)
         L        NUMBER OF ZEROES
         ZERMAX   MAXIMUM EXPECTED VALUE OF TRANSFER FUNCTION
                  ZEROES
```

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

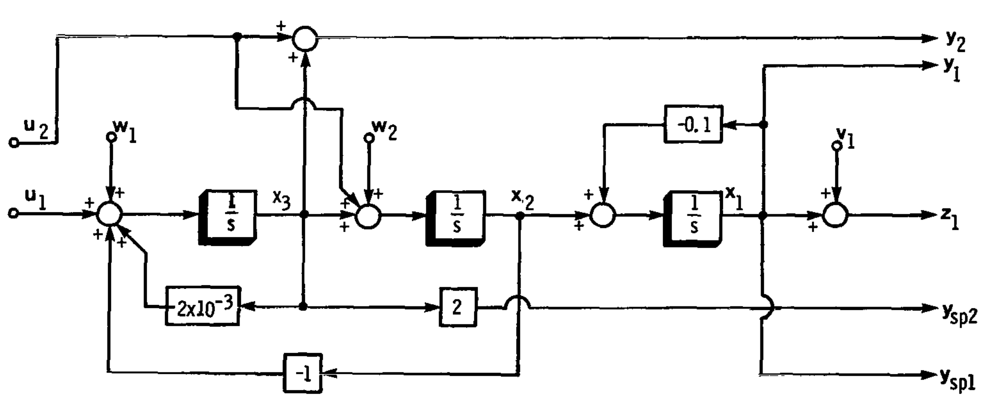
This appendix presents the results of two test cases. Test case I exercises almost all of the 77 available AESOP functions. Test case II shows how AESOP can be used in an interactive manner to design a control system.

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 ($\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3$) and state feedback as well as for Kalman filter feedback using a single noisy measurement, $\mathbf{z}_1$.

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.



State vector = $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$; control vector = $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$; measurement = z_1

Plant noise vector = $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$; measurement noise = v_1

Output vector = $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$; set-point vector = $\begin{pmatrix} y_{sp1} \\ y_{sp2} \end{pmatrix}$

Figure 14. – Block diagram of third-order system used for test case I.

TABLE XI. - TASKS PERFORMED IN TEST CASE I

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

Terminal Printout for Test Case I

```
AESRUN 1
 CANCELLED: OUT1 UNKNOWN.
OUT1 IS DDEFD TO THE HI-SPEED PRINTER
CG1 IS DDEFD TO IO UNIT 8
EG1 IS DDEFD TO IO UNIT 9
PFRUZ1 IS DDEFD TO IO UNIT 10
PFRUY1 IS DDEFD TO IO UNIT 11
PFRWZ1 IS DDEFD TO IO UNIT 12
PFRWY1 IS DDEFD TO IO UNIT 13
CFR1 IS DDEFD TO IO UNIT 14
PP1 IS DDEFD TO IO UNIT 15
SS1 IS DDEFD TO IO UNIT 16
FFG1 IS DDEFD TO IO UNIT 17
RUN1 IS LIBRARY GRAPHICS
RUN2 IS LIBRARY AESLIB
```

User invokes PROCDEF AESRUN to begin run

```
AESOP
DO YOU WISH TO MAKE PLOTS, Y OR N?
Y
ENTER THE PLOT NAME  - 8 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
I
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.
```

User calls the AESOP program

User requests maximum amount of data to be displayed at terminal

User elects to have the IFN function number string read in from storage

```
DDEF FT21F001,VS,DEC80

GO
 PROCEEDING(PCS): EXECUTION CONTINUES AT CHCRWC.(X'4CD4')
 &N1
 IFN =
101  201  401  402  403  801  802  301  803  804  805  806  807  808  809
810  302  811  303  812  813  814  815  816  817  818  819  820  501  502
503  504  505  506  507  508  509  510  511  512  513  514  515  516  517
518  519  520  521  522  523  524  525  601  602  603  604  701  702  703
704  705  210  404  801  301  809  819  405  999    0
 &END
```

User datadefs "dec80", the dataset containing the NAMELIST N1 for the test case, to unit 21. "GO" causes program to continue

```
FUNCTION  101
CHANGES TO REFS

DISPLAY REFS BEFORE MAKING CHANGES?
Y
&REFS
TSFTR= 1.0
DT= 0.50D-01
FI= 0.10D-01
DELF= 0.20D-01
ZERMAX= 100.0
AMPSP= 5*1.0
AMPSR= 5*1.0
AMPICX= 50*1.0
IF= 49
ISPACE= 1
IOUT= 1
IMEAS= 1
JINC= 1
JIND= 1
ITRMX= 100
NCURV= 2
LINLOG= 3
MSPY= 250*1
MSPYSP= 25*1
MSPU= 25*1
MSROLY= 250*1
```

Function 101 requested to demonstrate procedure for changing a "reference" parameter (in NAMELIST 'REFS')

```
MSROLX= 250*1
MICCLY= 2500*1
MICCLX= 2500*1
MICCLU= 250*1
MICOLY= 2500*1
MICOLX= 2500*1
&END
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)
  &REFS  ZERMAX=500.0D0  &END
&REFS                                          <------ Parameter ZERMAX increased from 100 to 500
TSFTR= 1.0
DT= 0.50D-01
FI= 0.10D-01
DELF= 0.20D-01
ZERMAX= 500.0
AMPSP= 5*1.0
AMPSR= 5*1.0
AMPICX= 50*1.0
IF= 49
ISPACE= 1
IOUT= 1
IMEAS= 1
JINC= 1
JIND= 1
ITRMX= 100
NCURV= 2
LINLOG= 3
MSPY= 250*1
MSPYSP= 25*1
MSPU= 25*1
MSROLY= 250*1
MSROLX= 250*1
MICCLY= 2500*1
MICCLX= 2500*1
MICCLU= 250*1
MICOLY= 2500*1
MICOLX= 2500*1
&END
ARE THERE ANYMORE CHANGES, Y OR N?
N

FUNCTION  201
PUT IN THE 3RD ORDER TEST CASE MATRICES BY USING SUBROUTINE MATIN
****** INPUT MATRICES ******              <------ 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

NN=

	1	2
1	1.000	0.0000
2	0.0000	3.000
3	2.000	0.0000

PCINV=

	1	2
1	1.000	0.0000
2	0.0000	0.1000D 00

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
DT= 0.50D-01
FI= 0.10D-01
DELF= 0.20D-01
ZERMAX= 500.0
AMPSP= 5*1.0
AMPSR= 5*1.0
AMPICX= 50*1.0
IF= 49
ISPACE= 1
IOUT= 1
IMEAS= 1
JINC= 1
JIND= 1
ITRMX= 100
NCURV= 2
LINLOG= 3
MSPY= 250*1
MSPYSP= 25*1
MSPU= 25*1
MSROLY= 250*1
MSROLX= 250*1
MICCLY= 2500*1
MICCLX= 2500*1
MICCLU= 250*1
MICOLY= 2500*1
MICOLX= 2500*1
&END
```

The following three functions perform analyses on the open-loop system:

```
FUNCTION  401
OPEN LOOP SYSTEM EIGENVALUES

**** MATRIX IS FOUND TO BE REDUCIBLE ****

 EIGENVALUES

        NAT FREQ (HZ)     ZETA

           0.1592           0.1000D-02
           0.1592D-01        1.000


FUNCTION  402
OPEN LOOP EIGENVECTORS AND MODE SHAPES
MODIFIED EIGENVECTOR MATRIX OF A

        1            2            3

 1   -1.087      -0.8933       1.000
 2   -1.001       0.9990      0.0000
 3    1.000        1.000      0.0000

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

For complex pair at 0.1592 Hz

```
        1            2            3

 1   0.9951      -174.4        1.000
 2   1.0000      -90.06       0.0000
 3    1.000      0.0000       0.0000
```

Magnitudes   Phase angles, deg

```
FUNCTION  403
OPEN LOOP SYSTEM ANALYSIS CALCULATIONS

SYSTEM CONTROLABILITY

        1            2

 1   0.5000       0.5000
 2   0.5730D-01    90.00
 3   0.9903     -0.9705D-01
```

For complex eigenvalue pair

```
SYSTEM OBSERVABILITY FOR (A AND H)

        1            2            3

 1   0.9951       174.4        1.000

SYSTEM OBSERVABILITY FOR (A AND C)

        1            2            3

 1   0.9951       174.4        1.000
 2    1.000      0.0000       0.0000

RESIDUES FOR (A, B, H) SYSTEM
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 FN/ZETA FORM
          NAT FREQ (HZ)     ZETA

            0.2382         0.7116
            0.4519          1.000
SS =

        1            2            3


1    19.24        10.42        1.301
2    10.42        9.203        1.819
3    1.301        1.819        1.647
```

```
KC =

        1            2            3

1    2.301        1.819        3.647
2    1.042        1.220       0.1819


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

         NAT FREQ (HZ)      ZETA

            0.2382           0.7116
            0.4519            1.000

FUNCTION  804
EIGENVECTORS AND MODE SHAPES WITH STATE FEEDBACK
MODIFIED EIGENVECTOR MATRIX OF AMBKC

        1            2            3

1  -0.9899      0.4246D-01   0.1494
2    1.000        1.000      -0.4092
3   0.2144       -1.042        1.000


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

        1            2            3

1   0.7006       -132.5       0.1494
2    1.000       0.0000      -0.4092
3   0.7519        123.4        1.000
```

Beginning of error checks for the Riccati solution matrix

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


 EIGENVALUES

         NAT FREQ (HZ)      ZETA

            0.1642          -1.000
            0.4929          -1.000
             4.133          -1.000
```

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

```
FUNCTION  806
SYMMETRY CHECK OF CONTROL RICCATI SOLUTION MATRIX, SS

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

        1            2            3

1   0.0000       0.0000       0.0000
2  -0.4475D-15   0.0000       0.0000
3  -0.3243D-14  -0.1465D-14   0.0000


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

	1	2	3
1	-0.1421D-13	-0.2842D-13	-0.1577D-13
2	-0.1754D-13	-0.1622D-13	-0.8438D-14
3	-0.8882D-15	0.4441D-15	0.6661D-15

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

FUNCTION 808
STORE CONTROL RICCATI SOLUTION MATRIX (SS) ON UNIT 16

Beginning of Riccati equation solution for Kalman filter problem

FUNCTION 809
DESIGN A KALMAN FILTER

EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM

NAT FREQ (HZ)	ZETA
0.2202	1.000
0.2862	0.4451

PP =

	1	2	3
1	2.882	4.442	1.482
2	4.442	11.76	8.865
3	1.482	8.865	18.37

KE =

	1
1	2.882
2	4.442
3	1.482

FUNCTION 810
STORE KALMAN FILTER GAINS (KE) ON UNIT 09

FUNCTION 302
FORM A-BKC-KEH MATRIX

FUNCTION 811
EIGENVALUES OF OPTIMAL CONTROLLER A-BKC-KEH

EIGENVALUES

NAT FREQ (HZ)	ZETA
0.6057	1.000
0.5366	0.6000

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

ATOT =

	1	2	3	4	5	6
1	-0.1000D 00	1.000	0.0000	0.0000	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	-2.301	-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	0.8181
6	1.482	0.0000	0.0000	-3.783	-2.819	-3.649

CTOT =

	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

DTOT =

	1	2
1	0.0000	0.0000
2	0.0000	1.000
3	1.000	0.0000
4	0.0000	0.0000
5	0.0000	0.0000
6	0.0000	0.0000

KCTOT =

	1	2	3	4	5	6
1	0.0000	0.0000	0.0000	-2.301	-1.819	-3.647
2	0.0000	0.0000	0.0000	-1.042	-1.220	-0.1819

HTOT =

	1	2	3	4	5	6
1	1.000	0.0000	0.0000	0.0000	0.0000	0.0000

FUNCTION 812
EIGENVALUES OF CONTROL SYSTEM WITH A FILTER IN THE FEEDBACK LOOP

EIGENVALUES

NAT FREQ (HZ)	ZETA
0.2202	1.000
0.2382	0.7116
0.2862	0.4451
0.4519	1.000

These are the eigenvalues of A - KE · H

These are the eigenvalues of A - B · KC

Beginning of error checks on Kalman filter error covariance matrix

FUNCTION 813
POSITIVE DEFINITENESS CHECK OF ERROR COVARIANCE MATRIX, PP

EIGENVALUES

NAT FREQ (HZ)	ZETA
0.1103	-1.000
1.138	-1.000
4.006	-1.000

Matrix PP is positive-definite

FUNCTION 814
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 =

	1	2	3
1	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000
3	-0.2996D-15	0.2004D-15	0.0000

FUNCTION 815
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

	1	2	3
1	-0.4441D-15	-0.3775D-14	-0.5958D-14
2	-0.4219D-14	-0.1155D-13	-0.1066D-13
3	-0.2849D-14	-0.1087D-13	-0.1066D-13

```
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


**************************************************
              COVARIANCE MATRICES
**************************************************


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= 0.65148D-12
NORMALIZED DIAGONAL ELEMENTS OF THE ERROR MATRIX =
-0.64970D-15 0.63051D-15-0.40617D-14
THE TRACE OF THE ERROR=-0.98575D-13
THE TRACE OF THE COVARIANCE=  66.806
TR(ERROR)/TR(COV.) =-0.14755D-14

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

KFF =

        1           2

 1    2.583       1.824
 2    1.164      -0.4090


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

Beginning of frequency response and Bode plot computation

```
FUNCTION  501
OPEN LOOP FREQUENCY RESPONSE OF MEASUREMENT  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              1.0002     *S**  1
  1.0000     *S**  0             0.10000D 00*S**  0
```

```
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                                    <-------- User specifies ZETA plotter to be used for plotting
 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              1.0002     *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 MEASUREMENT  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              1.0002     *S**  1
  1.0000    *S**  0             0.10000D 00*S**  0


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


FUNCTION  509
STORE OPEN LOOP FREQ. RESPONSE OF MEASUREMENT 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              1.0002     *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
-0.22204D-15*S**  2             4.9693     *S**  2
-0.22204D-15*S**  1             8.2882     *S**  1
 0.81810    *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
 -3.6470    *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 FREQ. 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		DENOMINATOR COEFFICIENTS	
0.00000	*S** 6	1.0000	*S** 6
0.00000	*S** 5	7.9536	*S** 5
0.00000	*S** 4	28.566	*S** 4
1.0000	*S** 3	62.639	*S** 3
7.8516	*S** 2	86.362	*S** 2
26.764	*S** 1	71.725	*S** 1
43.263	*S** 0	28.452	*S** 0

FUNCTION 518
PLOT CLOSED LOOP FREQ. 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		DENOMINATOR COEFFICIENTS	
0.00000	*S** 6	1.0000	*S** 6
0.00000	*S** 5	7.9536	*S** 5
0.00000	*S** 4	28.566	*S** 4
1.0000	*S** 3	62.639	*S** 3
7.8516	*S** 2	86.362	*S** 2
26.764	*S** 1	71.725	*S** 1
43.263	*S** 0	28.452	*S** 0

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

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		DENOMINATOR COEFFICIENTS	
0.00000	*S** 6	1.0000	*S** 6
-0.44409D-15	*S** 5	7.9536	*S** 5
-0.13323D-14	*S** 4	28.566	*S** 4
-0.39968D-14	*S** 3	62.639	*S** 3
-20.117	*S** 2	86.362	*S** 2
-6.8315	*S** 1	71.725	*S** 1
-24.088	*S** 0	28.452	*S** 0

FUNCTION 522
PLOT CLOSED LOOP FREQ. 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		DENOMINATOR COEFFICIENTS	
0.00000	*S** 3	1.0000	*S** 3
-20.117	*S** 2	7.8516	*S** 2
-6.8315	*S** 1	26.764	*S** 1
-24.088	*S** 0	43.263	*S** 0

FUNCTION 524
PLOT FREQ. RESPONSE OF CONTROL TO MEASUREMENT
FOR OPTIMAL CONTROLLER

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

← *Beginning of transient response calculations and plots*

FUNCTION 601
OBTAIN AND PLOT SELECTED OPEN LOOP STEP RESPONSES

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	0.9988	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	0.4988D-01	0.1248D-02	0.2080D-04
2	0.0000	0.4998D-01	0.1250D-02
3	0.0000	-0.1250D-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	0.9988	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	0.4988D-01	0.1248D-02	0.2080D-04
2	0.0000	0.4998D-01	0.1250D-02
3	0.0000	-0.1250D-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	0.9413D-03
2	-0.5251D-01	0.9369	0.3619D-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	0.9413D-03
2	-0.5251D-01	0.9369	0.3619D-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

← Beginning of transfer function gain and numerator zeros calculation

```
FUNCTION  701
GAIN AND ZEROES OF OPEN LOOP TRANSFER FUNCTION
RELATING MEASUREMENT  1 TO CONTROL INPUT  1

GAIN =     1.00000
NUMBER OF ZEROES =  0
REAL PARTS OF NUMERATOR ZEROES

         1

 1   0.0000

IMAGINARY PARTS OF NUMERATOR ZEROES

         1

 1   0.0000


FUNCTION  702
GAIN AND ZEROES OF OPEN LOOP TRANSFER FUNCTION
RELATING OUTPUT  1 TO CONTROL INPUT  1

GAIN =     1.00000


NUMBER OF ZEROES =  0
REAL PARTS OF NUMERATOR ZEROES

         1

 1   0.0000

IMAGINARY PARTS OF NUMERATOR ZEROES

         1

 1   0.0000


FUNCTION  703
GAIN AND ZEROES OF OPEN LOOP TRANSFER FUNCTION
RELATING MEASUREMENT  1 TO DISTURBANCE INPUT  1

GAIN =     1.00000
NUMBER OF ZEROES =  0
REAL PARTS OF NUMERATOR ZEROES

         1

 1   0.0000

IMAGINARY PARTS OF NUMERATOR ZEROES

         1

 1   0.0000


FUNCTION  704
GAIN AND ZEROES OF OPEN LOOP TRANSFER FUNCTION
RELATING OUTPUT  1 TO DISTURBANCE INPUT  1

GAIN =     1.00000
NUMBER OF ZEROES =  0
REAL PARTS OF NUMERATOR ZEROES


         1

 1   0.0000

IMAGINARY PARTS OF NUMERATOR ZEROES

         1

 1   0.0000
```

```
FUNCTION  705
GAIN AND ZEROES OF OPTIMAL CONTROLLER TRANSFER FUNCTION
RELATING CONTROL  1 TO MEASUREMENT  1

GAIN =     20.1170
NUMBER OF ZEROES =  2
REAL PARTS OF NUMERATOR ZEROES

        1

 1  -0.1699
 2  -0.1699

IMAGINARY PARTS OF NUMERATOR ZEROES

        1

 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 PP to show that results are identical to prior calculations

```
FUNCTION  210
CHANGE MATRICES A, B, 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, 49*0.0, 1.0, 0.0, -1.0, 48*0.0, 1.0, -0.20D-02, 2397*0.0
B= 2*0.0, 1.0, 48*0.0, 1.0, 198*0.0
C= 1.0, 100*0.0, 1.0, 2398*0.0
D= 2*0.0, 1.0, 48*0.0, 1.0, 698*0.0
H= 1.0, 249*0.0
DOUT= 51*0.0, 1.0, 198*0.0
CSP= 1.0, 10*0.0, 2.0, 238*0.0
QC= 20.0, 50*0.0, 1.0, 50*0.0, 10.0, 2397*0.0
NN= 1.0, 0.0, 2.0, 48*0.0, 3.0, 198*0.0
PCINV= 1.0, 5*0.0, 0.10D0, 18*0.0
QQ= 51*0.0, 2.0, 50*0.0, 20.0, 2397*0.0
RRINV= 1.0, 24*0.0
N= 3
NM= 1
NC= 2
ND= 2
NO= 2
&END
ENTER MATDAT CHANGES AS &MATDAT A(1,1)= , ETC. &END
  &MATDAT
  QC(1,1) = 500.0D0
  QC(2,2) = 9.0D0
  QC(3,3) = 40.0D0
  PCINV(1,1) = 1.5625D-2
  PCINV(2,2) = 1.2345679D-3
  NN(1,1) = 40.0D0
  NN(3,1) = 32.0D0
  NN(2,2) = 81.0D0
  &END
```

New values for weighting elements calculated off-line so that they correspond to normalized versions of those stored above as MATDAT

```
&MATDAT
A= -0.10D0, 49*0.0, 1.0, 0.0, -1.0, 48*0.0, 1.0, -0.20D-02, 2397*0.0
B= 2*0.0, 1.0, 48*0.0, 1.0, 198*0.0
C= 1.0, 100*0.0, 1.0, 2398*0.0
D= 2*0.0, 1.0, 48*0.0, 1.0, 698*0.0
H= 1.0, 249*0.0
DOUT= 51*0.0, 1.0, 198*0.0
CSP= 1.0, 10*0.0, 2.0, 238*0.0
QC= 500.0, 50*0.0, 9.0, 50*0.0, 40.0, 2397*0.0
NN= 40.0, 0.0, 32.0, 48*0.0, 81.0, 198*0.0
PCINV= 0.156250D-01, 5*0.0, 0.123456790D-02, 18*0.0
QQ= 51*0.0, 2.0, 50*0.0, 20.0, 2397*0.0
RRINV= 1.0, 24*0.0
N= 3
NM= 1
NC= 2
ND= 2
NO= 2
&END
ARE THERE ANYMORE CHANGES, Y OR N?
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 'NRMS' TO UNIT 34
 PROCEEDING: PAUSE  .  PROGRAM WILL PROCEED IF _RUN COMMAND ISSUED.

DDEF FT34F001,VS,ANRMS

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

NORMALIZING FACTORS ARE
&NRMS
SCX= 5.0, 3.0, 2.0, 47*0.0
SCU= 8.0, 9.0, 3*0.0
```

```
SCY= 4.0, 7.0, 10.0, 47*0.0
SCZ= 6.0, 4*0.0
SCYSP= 12.0, 11.0, 3*0.0
&END

NORMALIZED SYSTEM MATRICES

THE A MATRIX IS

        1             2             3

 1  -0.1000D 00   0.6000        0.0000
 2   0.0000       0.0000        0.6667
 3   0.0000      -1.500        -0.2000D-02

THE B MATRIX IS

        1             2

 1   0.0000       0.0000
 2   0.0000        3.000
 3    4.000       0.0000

THE C MATRIX IS

        1             2             3

 1    1.250       0.0000        0.0000
 2   0.0000       0.0000        0.2857

THE H MATRIX IS

        1             2             3

 1   0.8333       0.0000        0.0000

THE QQ MATRIX IS

        1             2             3

 1   0.0000       0.0000        0.0000
 2   0.0000       0.2222        0.0000
 3   0.0000       0.0000         5.000

THE RRINV MATRIX IS

        1

 1    36.00

THE D MATRIX IS

        1             2

 1   0.0000       0.0000
 2   0.0000       0.3333
 3   0.5000       0.0000

THE DOUT MATRIX IS

        1             2

 1   0.0000       0.0000
 2   0.0000        1.286

THE CSP MATRIX IS

        1             2             3

 1   0.4167       0.0000        0.0000
 2   0.0000       0.0000        0.3636


FUNCTION  801
DESIGN A LINEAR QUADRATIC REGULATOR
```

```
EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM
          NAT FREQ (HZ)      ZETA

            0.2382            0.7116}
            0.4519             1.000}
SS =
```

← Note that LQR eigenvalues are not changed by process of normalization (QC, NN, and PCINV were also normalized)

	1	2	3
1	481.1	156.3	13.01
2	156.3	82.83	10.91
3	13.01	10.91	6.588

KC =

	1	2	3
1	1.438	0.6821	0.9117
2	0.5789	0.4068	0.4042D-01

```
FUNCTION  301
FORM A-BKC MATRIX


FUNCTION  809
DESIGN A KALMAN FILTER


EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM
          NAT FREQ (HZ)      ZETA

            0.2202            1.000}
            0.2862           0.4451}
PP =
```

← Same Kalman filter eigenvalues as obtained previously

	1	2	3
1	0.1153	0.2961	0.1482
2	0.2961	1.307	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	2
1	3.874	2.509
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


NORMALIZING FACTORS
&NRMS
SCX= 5.0, 3.0, 2.0, 47*0.0
SCU= 8.0, 9.0, 3*0.0
SCY= 4.0, 7.0, 10.0, 47*0.0
SCZ= 6.0, 4*0.0
SCYSP= 12.0, 11.0, 3*0.0
&END
```

← Check shows that the following matrices are identical to those previously computed:

```
UNNORMALIZED SYSTEM MATRICES

THE KC MATRIX IS
```

```
         1            2            3

 1    2.301        1.819         3.647
 2    1.042        1.220        0.1819

THE KE MATRIX IS

         1

 1    2.882
 2    4.442
 3    1.482

THE KFF MATRIX IS

         1            2

 1    2.583         1.824
 2    1.164       -0.4090

THE PP MATRIX IS

         1            2            3

 1    2.882        4.442         1.482
 2    4.442        11.76         8.865
 3    1.482        8.865         18.37

STORE THE N1 FOR THIS RUN?  ← If new function number string has been formed, it could have been stored
N                                at this point for future use
TERMINATED:  STOP RETURN

PRINT OUT1,PRTSP=EDIT ← User requests output dataset (OUT1 in this case) to be printed out
```

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 $\mathbf{y}_{sp}$. They are selected to be $\mathbf{y}_{sp1} = \mathbf{x}_1$ and $\mathbf{y}_{sp2} = \mathbf{x}_3$ by proper definition of matrix **CSP**. 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 **(KFF)** and feedback **(KC)** 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 reponses 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 $|\mathbf{u}_1| \leq 5$ and $|\mathbf{u}_2| \leq 5$.

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 $\mathbf{y}_{sp1}$ and $\mathbf{y}_{sp2}$ 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 **QC** and **NN** are allowed to be zero, and **PCINV** is set equal to a 2×2 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 N1 FROM STORAGE?, the user replies "N." The program then prompts the user to enter function numbers from the terminal

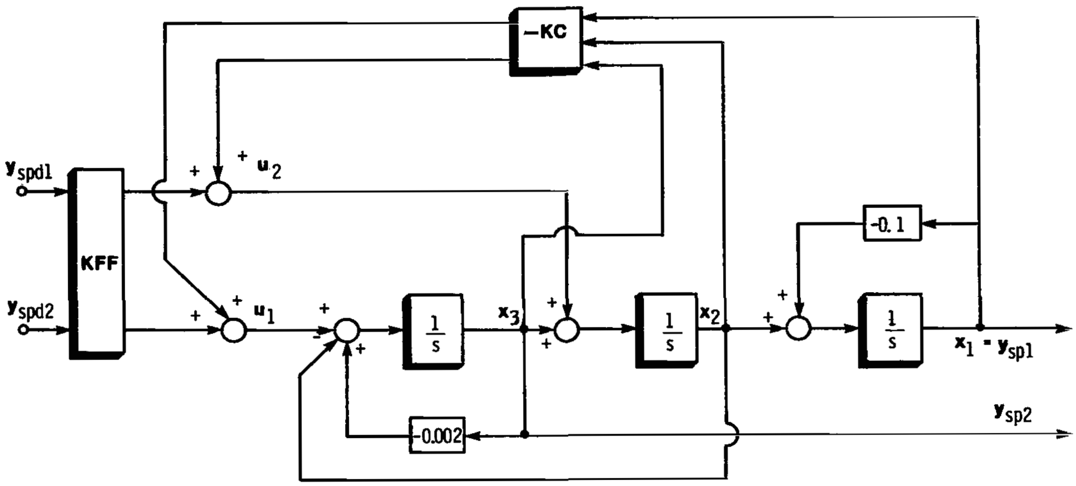


Figure 15. – Nonzero-set-point regulator.

```
&MATDAT
  N=3,NC=2,NM=1,NO=2,ND=2,
  A(1,1)=-0.1,0.,0.,A(1,2)=1.,0.,-1.,A(1,3)=0.,1.,-.2D-2,
  B(3,1)=1.,B(2,2)=1.,D(3,1)=1.,D(2,2)=1.,C(1,1)=1.,
  C(2,3)=1.,H(1,1)=1.,DOUT(2,2)=1.,CSP(1,1)=1.,CSP(2,3)=1.,
  PCINV(1,1)=1.,PCINV(2,2)=1.,QQ(2,2)=2.,QQ(3,3)=20.,RRINV(1,1)=1.
&END
&REFS
  DT=.24,
  MSPY=250*0,MSPYSP=25*0,MSPU=25*0,MSPYSP(1,1)=1,MSPYSP(2,2)=1
&END
```

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
go
CZAPB030      PROCEEDING(PCS): EXECUTION CONTINUES AT CHCRWC.(X'4CD4')
DISPLAY INPUT MATRICES?
n
```

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 CONPAR BEFORE MAKING CHANGES?
y
&CONPAR
QC= 2500*0.0
NN= 250*0.0
```

```
PCINV= 1.0, 5*0.0, 1.0, 18*0.0
&END
ENTER CONTROL WTS QC, NN AND/OR PCINV (NAMELIST CONPAR)
 &conpar  qc(1,1) = 0.001 &end
&CONPAR
QC= 0.10D-02, 2499*0.0
NN= 250*0.0
PCINV= 1.0, 5*0.0, 1.0, 18*0.0
&END
ARE THERE ANYMORE CHANGES, Y OR N?
n


FUNCTION  801
DESIGN A LINEAR QUADRATIC REGULATOR


EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM
         NAT FREQ (HZ)      ZETA

           0.1667D-01       1.000
           0.1593          0.2223D-01
SS =


        1             2             3


 1   0.4883D-02  0.6155D-03  0.4806D-02
 2   0.6155D-03  0.2122D-01  0.3902D-03
 3   0.4806D-02  0.3902D-03  0.2601D-01


KC =


        1             2             3


 1   0.4806D-02  0.3902D-03  0.2601D-01
 2   0.6155D-03  0.2122D-01  0.3902D-03



FUNCTION    0
TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS.(I3), ONE PER LINE
TO TERMINATE ENTER 999(LAST ENTRY MUST BE A RETURN)
301
819
604
```

The user then requests that step responses be plotted for a nonzero-set-point regulator. This requires, as prerequisites, forming of $\mathbf{A}-\mathbf{B}\cdot\mathbf{KC}$ and calculation of feedforward gain matrix **KFF**.

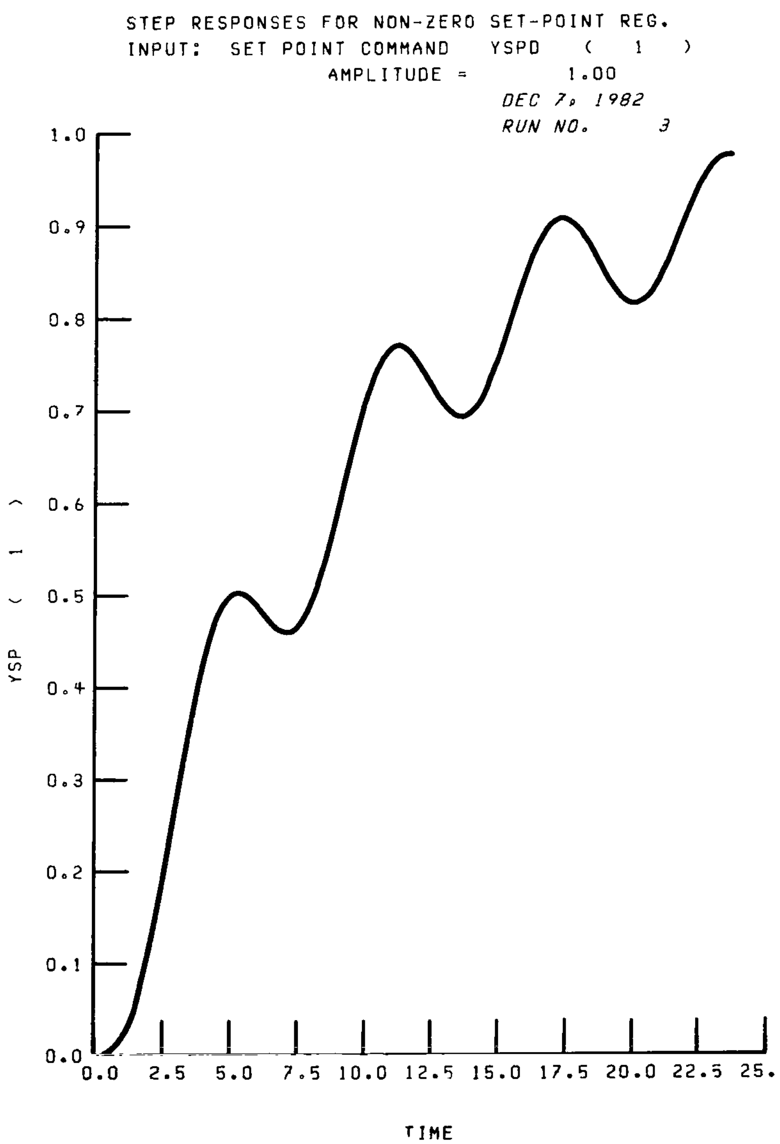
```
FUNCTION  301
FORM A-BKC MATRIX


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
```

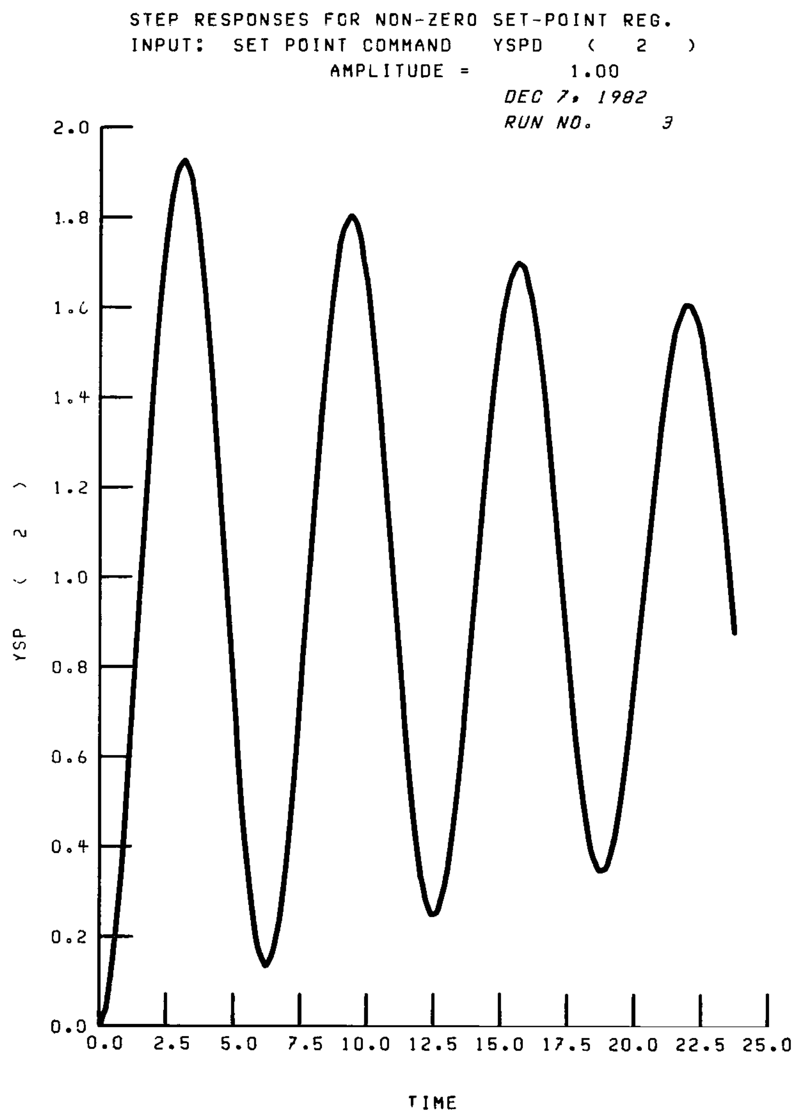
```
FUNCTION  604
OBTAIN AND PLOT SELECTED STEP RESPONSES
FOR THE NON-ZERO SET POINT LINEAR REGULATOR

STATE TRANSITION MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

        1            2            3


 1   0.9763       0.2343       0.2831D-01
 2  -0.2802D-03   0.9664       0.2362
 3  -0.1108D-02  -0.2365       0.9648

FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

        1            2            3


 1   0.2371       0.2839D-01   0.2276D-02
 2  -0.2841D-04   0.2371       0.2854D-01
 3  -0.1349D-03  -0.2857D-01   0.2369
```

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.

```
GRINT100     GRAPHICS DEVICE NOT DEFINED BY DDEF.
 ENTER UNIT NAME.  DEFAULT TO CANCEL.
la011
```

Two plots are then displayed: one for the response of $\mathbf{y}_{sp1}$ to a step in $\mathbf{y}_{spd1}$, and the other for $\mathbf{y}_{sp2}$ to a step in $\mathbf{y}_{spd2}$.


STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (1)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
YSP (1)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
TIME


STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (2)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
YSP (2)
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
TIME

With this low weighting on $\mathbf{x}_1$, 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.

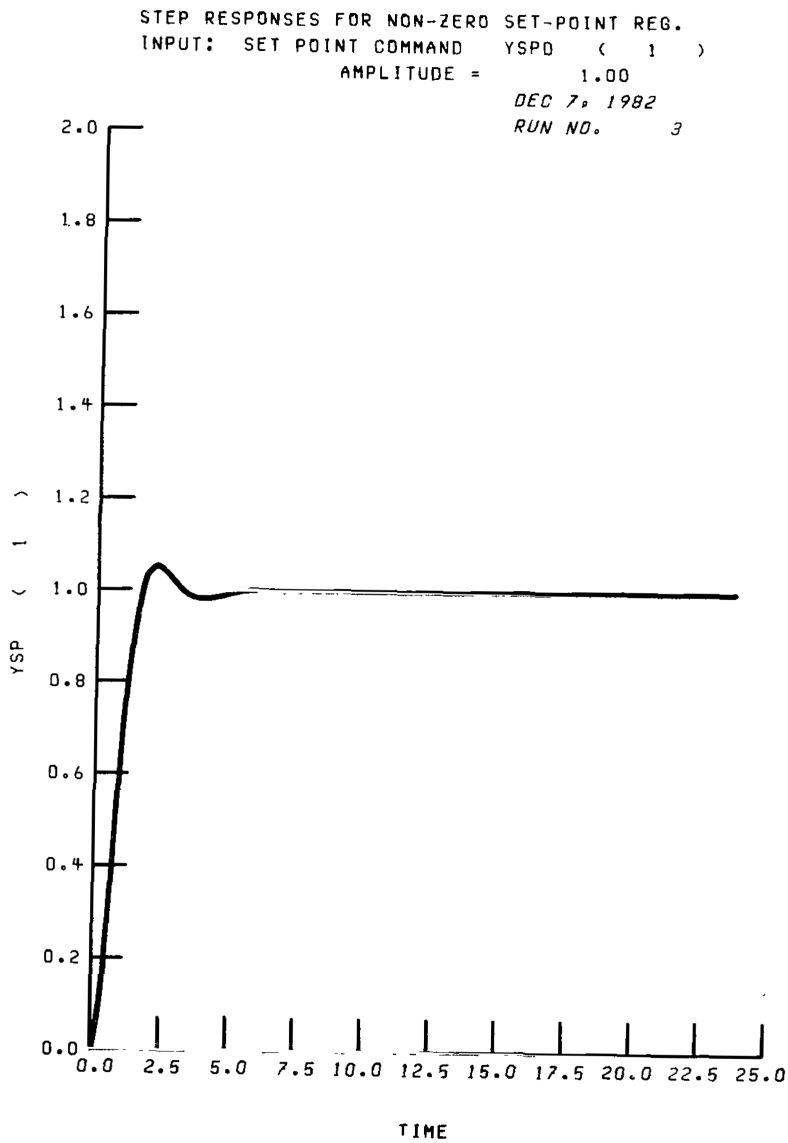
```
FUNCTION    0
TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS.(I3), ONE PER LINE
TO TERMINATE ENTER 999(LAST ENTRY MUST BE A RETURN)
:203
801
301
819
604
```

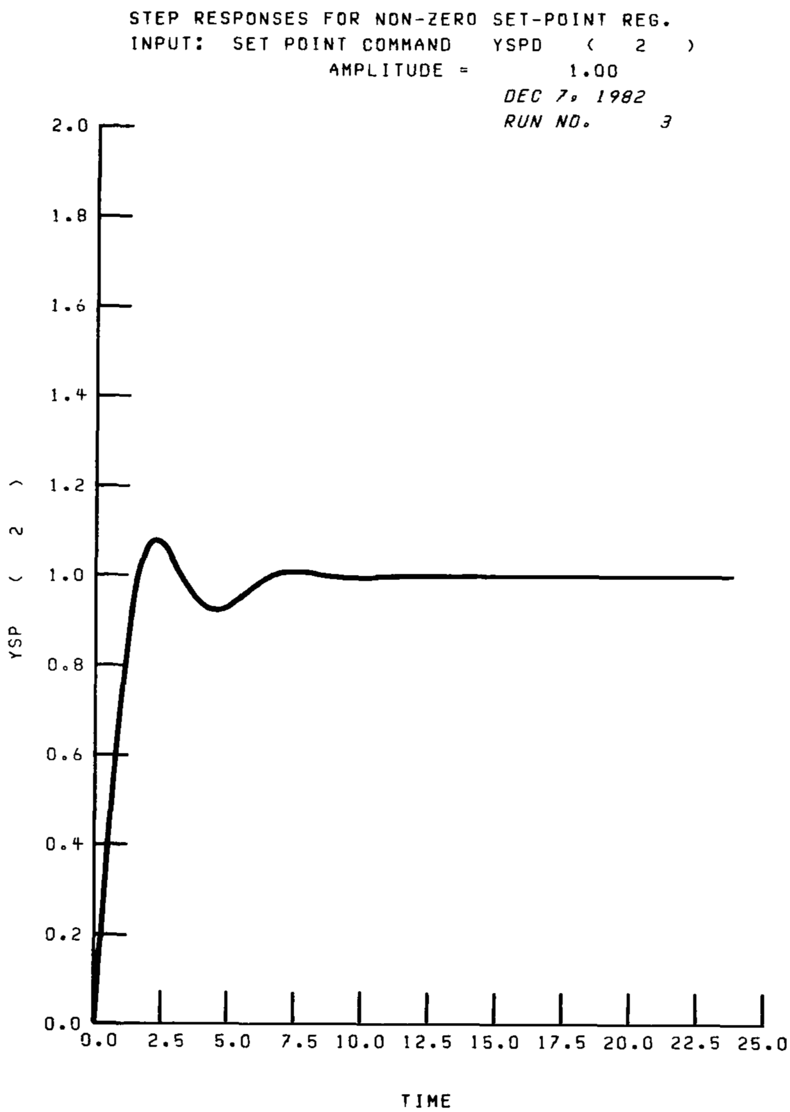
This time, the weight on $\mathbf{x}_1$ is increased to 1.0, and the LQR and feedforward gains are recomputed.

```
FUNCTION  203
DISPLAY CONPAR BEFORE MAKING CHANGES?
n
ENTER CONTROL WTS QC, NN AND/OR PCINV (NAMELIST CONPAR)
 &conpar  qc(1,1) = 10. &em#

 &conpar qc(1,1) = 10.0 &end
&CONPAR
QC= 10.0, 2499*0.0
NN= 250*0.0
PCINV= 1.0, 5*0.0, 1.0, 18*0.0
&END
ARE THERE ANYMORE CHANGES, Y OR N?
n


FUNCTION  801
DESIGN A LINEAR QUADRATIC REGULATOR


EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM
          NAT FREQ (HZ)      ZETA

            0.1392            1.000
            0.3027           0.5564
SS =


        1            2            3


 1    7.206        2.783       0.9032
 2    2.783        1.950       0.6620
 3   0.9032       0.6620       0.9392


KC =


        1            2            3


 1   0.9032       0.6620       0.9392
 2    2.783        1.950       0.6620


FUNCTION  301
FORM A-BKC MATRIX


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

```
KFF =

        1           2

 1    1.069       0.9412
 2    2.978      -0.3380


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

        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

FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

        1           2           3

 1    0.2315      0.2420D-01  0.6469D-03
 2   -0.6794D-01  0.1856      0.7636D-02
 3   -0.1475D-01 -0.3928D-01  0.2138
```

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
DEC 7, 1982
RUN NO. 3
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
YSP (1)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
TIME


STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (2)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
YSP (2)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
TIME

Actually, the 10-percent overshoot criterion is met, but $\mathbf{y}_{sp1}$ does not quite reach steady state in less than 5 seconds. Thus the user begins a third design iteration, choosing the weighting on $\mathbf{x}_1$ to be 10 this time.

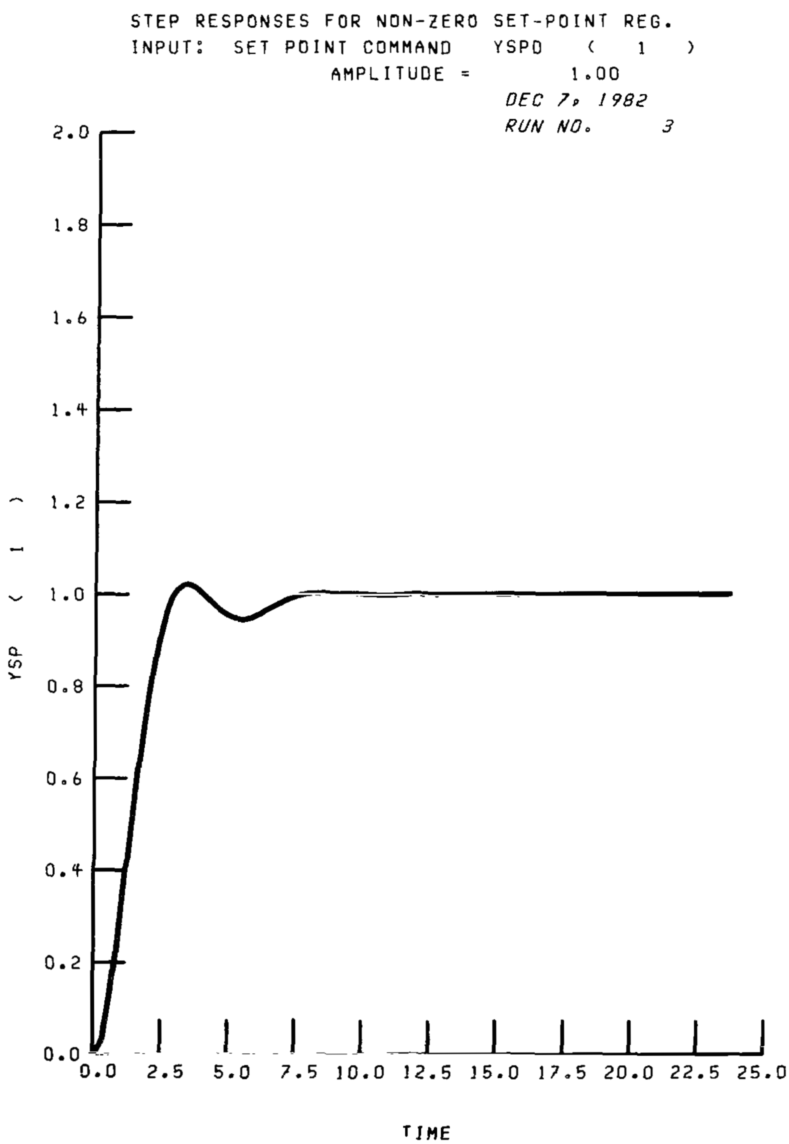
```
FUNCTION    0
TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS.(I3), ONE PER LINE
TO TERMINATE ENTER 999(LAST ENTRY MUST BE A RETURN)
203
801
301
819
604


FUNCTION  203
DISPLAY CONPAR BEFORE MAKING CHANGES?
n
ENTER CONTROL WTS QC, NN AND/OR PCINV (NAMELIST CONPAR)

&CONPAR
QC= 1.0, 2499*0.0
NN= 250*0.0
PCINV= 1.0, 5*0.0, 1.0, 18*0.0
&END
ARE THERE ANYMORE CHANGES, Y OR N?
n


FUNCTION  801
DESIGN A LINEAR QUADRATIC REGULATOR


EIGENVALUES OF REGULATOR OR FILTER IN FN/ZETA FORM
          NAT FREQ (HZ)      ZETA

            0.9524D-01        1.000
            0.2063           0.3879
```

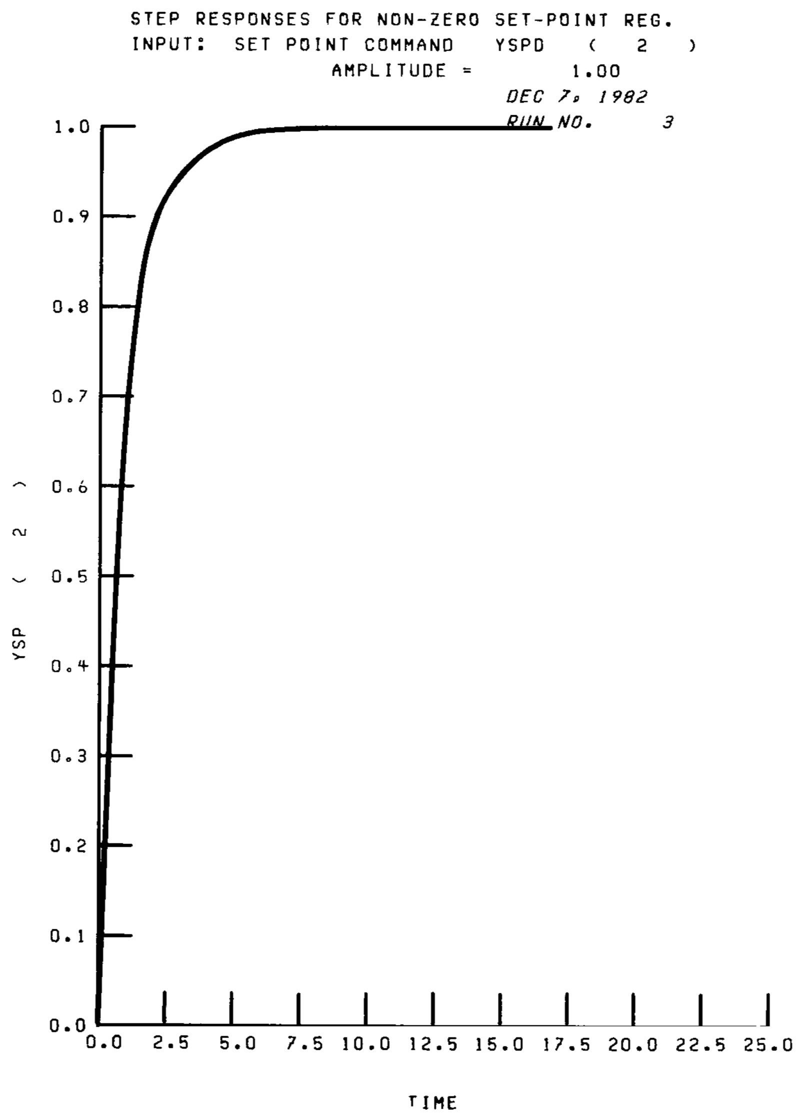
The preceding eigenvalues indicate that response time should now be acceptable as the slowest eigenvalue ($\lambda_1 = 0.1392$ Hz $= 0.875$ rad/sec) should give a settling time ($\sim 4/\lambda_1$) of about 4.5 seconds.

```
SS =

         1             2             3


 1     1.285       0.6844        0.5240
 2    0.6844       0.7483        0.3449
 3    0.5240       0.3449        0.7536

KC =

         1             2             3


 1    0.5240       0.3449        0.7536
 2    0.6844       0.7483        0.3449


FUNCTION  301
FORM A-BKC MATRIX


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

```
KFF =

        1            2

 1   0.6585       0.7556
 2   0.7592      -0.6551


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

        1            2            3

 1   0.9574       0.2138       0.1648D-01
 2  -0.1549       0.7967       0.1292
 3  -0.8876D-01  -0.2785       0.8125

FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

        1            2            3

 1   0.2356       0.2673D-01   0.1366D-02
 2  -0.1901D-01   0.2164       0.1662D-01
 3  -0.1208D-01  -0.3521D-01   0.2177
```

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
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
YSP (1)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
TIME


STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (2)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
YSP (2)
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0 22.5 25.0
TIME

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 **MSYSP** and **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.

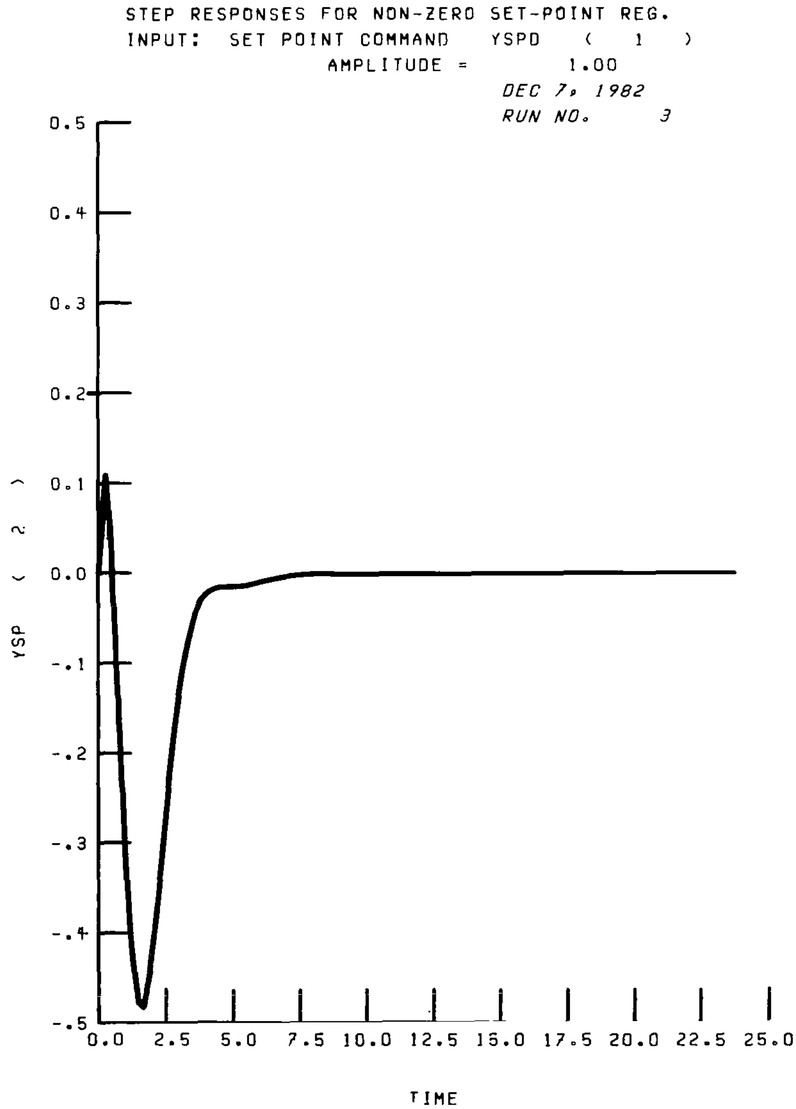
```
FUNCTION    0
TO COMPUTE FURTHER, ENTER NEXT FUNCTION NOS.(I3), ONE PER LINE
TO TERMINATE ENTER 999(LAST ENTRY MUST BE A RETURN)
101
604


FUNCTION  101
CHANGES TO REFS

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

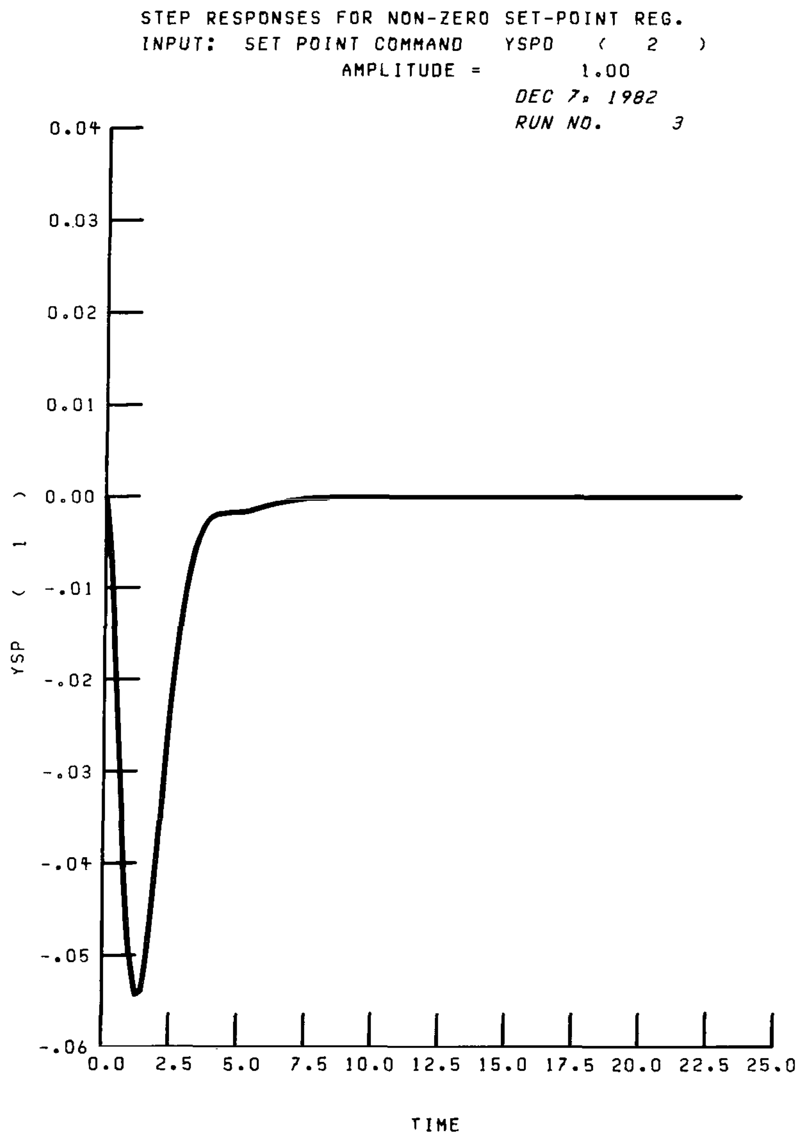
The program then displays the reference values and plots the requested transients.

```
&REFS  mspysp(1,1)=0,1,mspysp(1,2)=1,0,mspu(1,1)=1,1,mspu(1,2)=1,1 &end
&REFS
TSFTR= 1.0
DT= 0.240
FI= 0.10D-01
DELF= 0.10D-01
ZERMAX= 100.0
AMPSP= 5*1.0
AMPSR= 5*1.0
AMPICX= 50*1.0
IF= 96
ISPACE= 0
IOUT= 1
IMEAS= 1
JINC= 1
JIND= 1
ITRMX= 100
NCURV= 2
LINLOG= 2
MSPY= 250*0
MSPYSP= 0, 1, 3*0, 1, 19*0
MSPU= 2*1, 3*0, 2*1, 18*0
MSROLY= 250*0
MSROLX= 2*1, 8*0, 2*1, 238*0
MICCLY= 1, 2499*0
MICCLX= 2500*0
MICCLU= 250*0
MICOLY= 1, 2499*0
MICOLX= 1, 50*0, 1, 2448*0
&END
ARE THERE ANYMORE CHANGES, Y OR N?
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


        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
```

FORCED RESPONSE MATRIX OF LINEAR REGULATOR FOR TIME STEP 0.2400

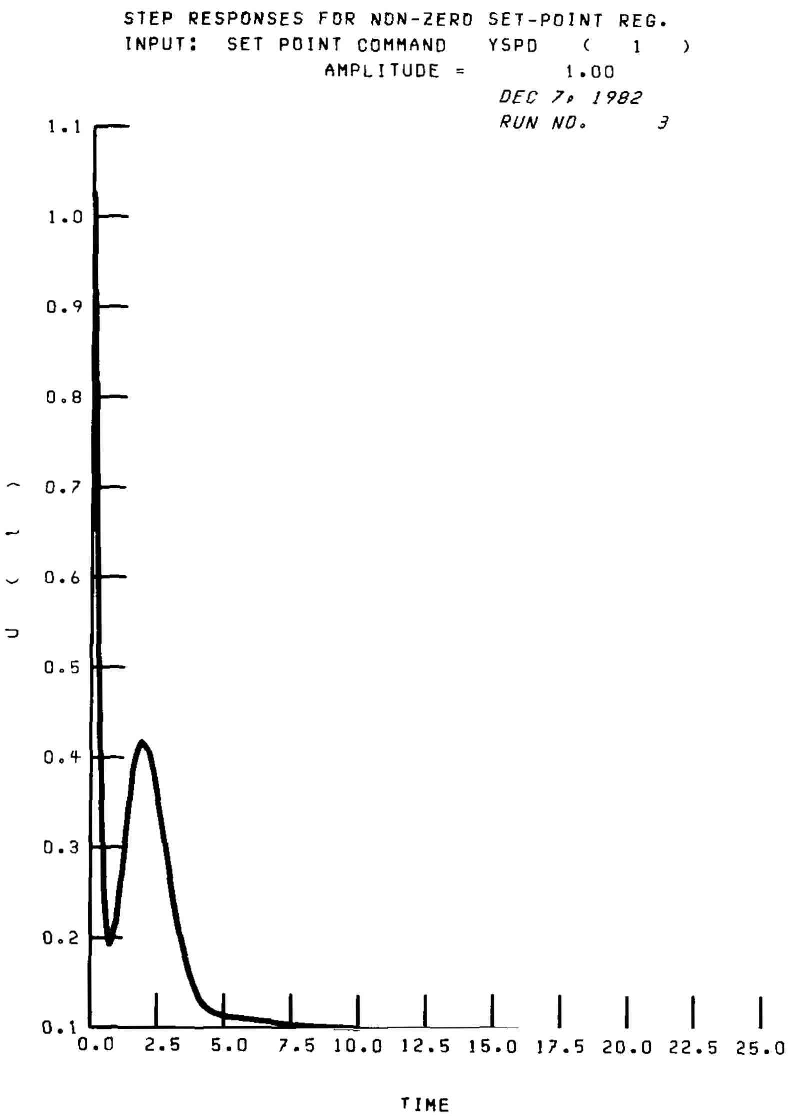
	1	2	3
1	0.2315	0.2420D-01	0.6469D-03
2	-0.6794D-01	0.1856	0.7636D-02
3	-0.1475D-01	-0.3928D-01	0.2138



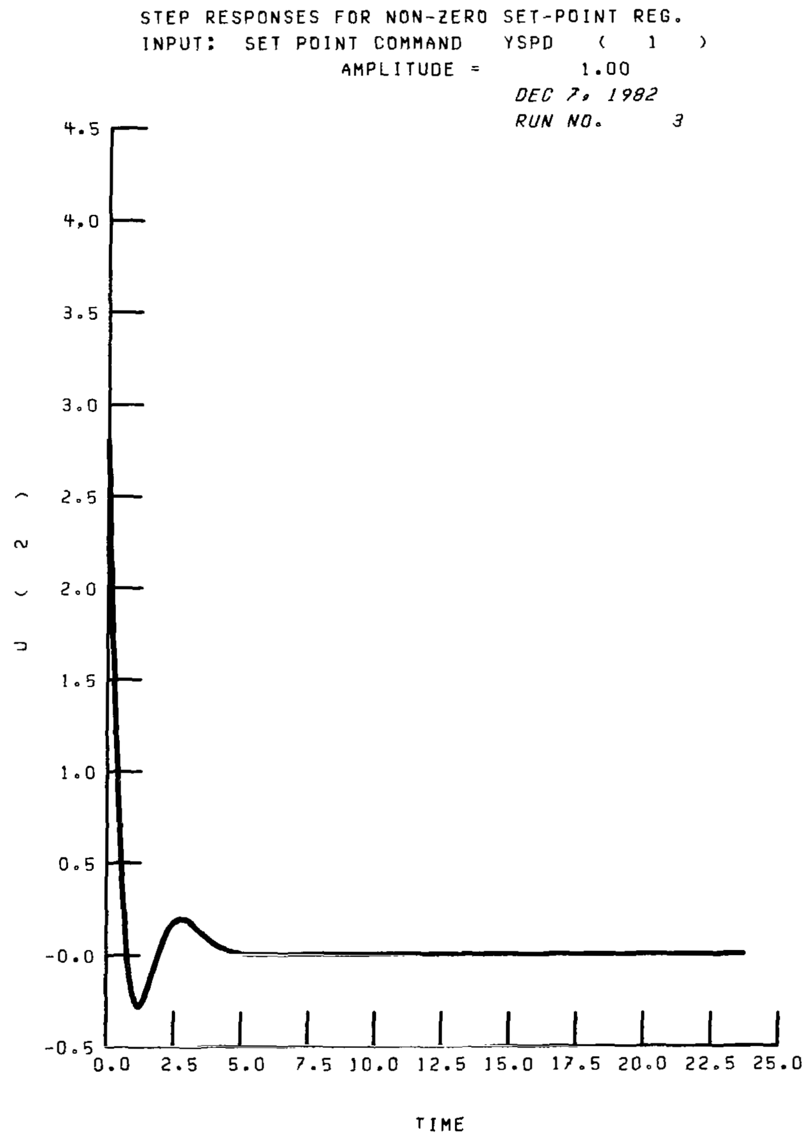


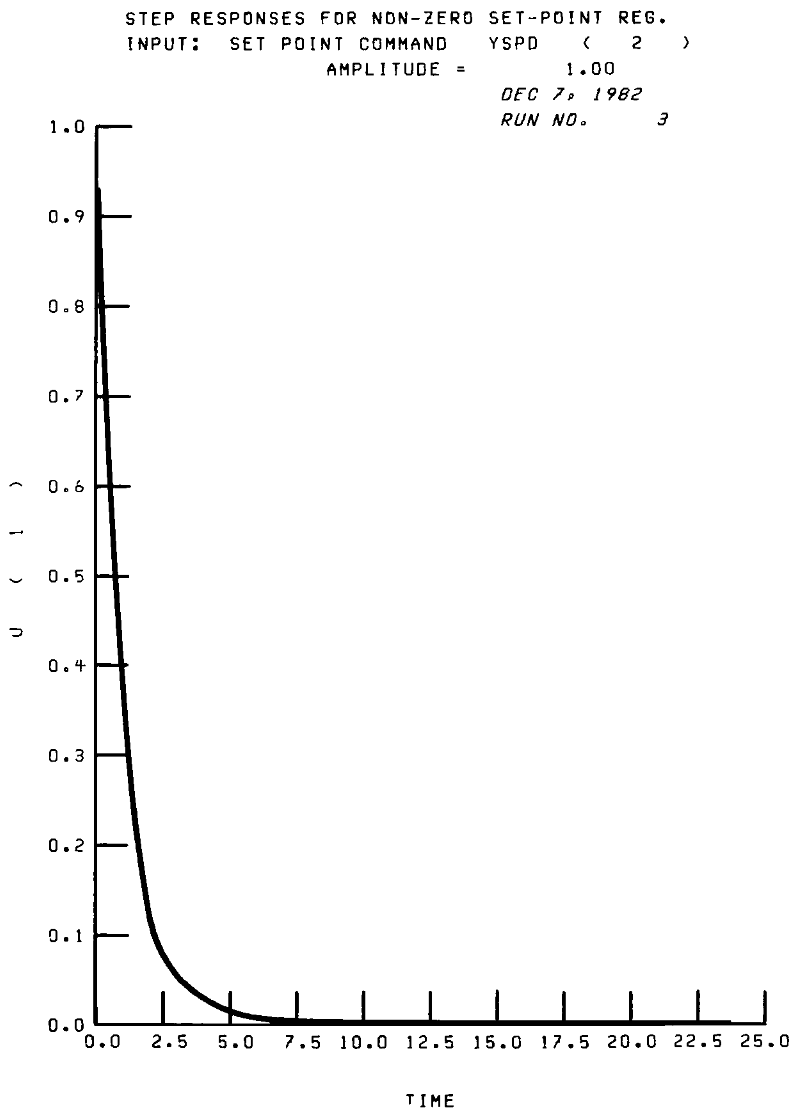
Note that there is some interaction between input 1 and output 2 but little between input 2 and output 1.

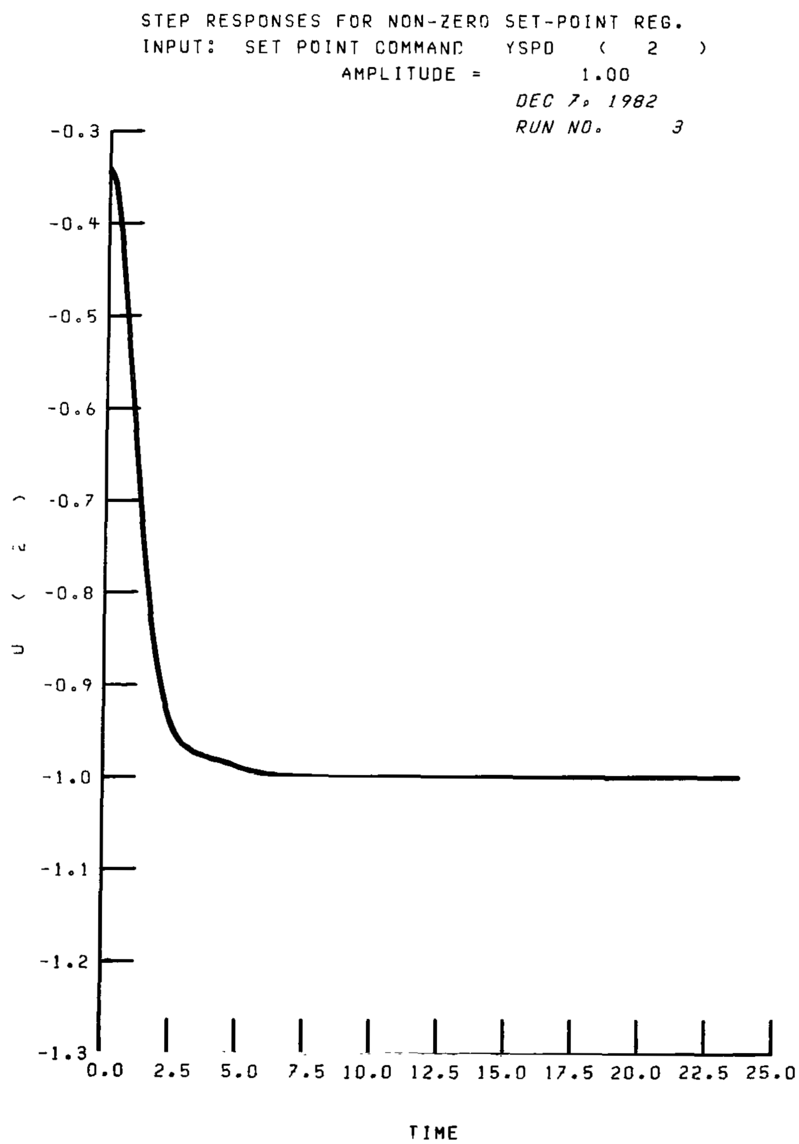
Finally, the four control responses are displayed. These responses confirm that design criterion 3 has been met. That is, both control signal absolute magnitudes remain less than 5 during a step response. The maximum excursion for $\mathbf{u}_1$ is $+1.07$ and that for $\mathbf{u}_2$ is $+2.98$, both 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
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
U (1)
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0 22.5 25.0
TIME


STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (1)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
-0.0
-0.5
U (2)
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0 22.5 25.0
TIME


STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (2)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
U (1)
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0 22.5 25.0
TIME



STEP RESPONSES FOR NON-ZERO SET-POINT REG.
INPUT: SET POINT COMMAND YSPD (2)
AMPLITUDE = 1.00
DEC 7, 1982
RUN NO. 3
U (2)
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
-1.0
-1.1
-1.2
-1.3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
TIME

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?
n
CHCRW400     TERMINATED: STOP RETURN
print outaa,prtsp=edit
CZABD050     PRINT BSN=0498,    800 LINES
off
B007         LOGOFF AT 14:40 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

(19) Residual error matrix for the Kalman filter Riccati equation
(20) 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.

```
AESOP --|
        |
        |-PLOTSUBS
        |
        |--AES100--|
        |(101,102) |
        |          |
        |          |--PREREQ
        |
        |--AES200--|
        |(203,204,)|
        |(205,206,)|
        |(207,208) |
        |          |
        |          |--PREREQ
        |          |
        |          |--MATPRT
        |          |
        |          |--MATIN --|
        |          |  (201)   |
        |          |          |
        |          |          |--MATPRT
        |          |
        |          |--MATRD --|
        |          |  (202)   |
        |          |          |
        |          |          |--MATPRT
        |          |
        |          |--MATCHG
        |             (209)
        |
        |--AES300--|
        |(301,302,)|
        |  (303)   |
        |          |
        |          |--PREREQ
        |          |
        |          |--MATPRT
        |
        |--AES400--|
        |          |
        |          |--PREREQ
        |          |
        |          |--MATPRT
        |          |
        |          |--EIGEN --|
        |          |  (401)   |
        |          |          |
        |          |          |--ARRAY
        |          |          |
        |          |          |--HSBG
        |          |          |
        |          |          |--EIGQR
        |          |          |
        |          |          |--SCALEA
        |          |          |
        |          |          |--CONDI --|
        |          |                     |
        |          |                     |--SCALEA
        |          |                     |
        |          |                     |--REDU
        |          |
        |          |--EGVCTR--|
        |          |  (402)   |
        |          |          |
        |          |          |--MATPRT
        |          |          |
        |          |          |--ARRAY
        |          |          |
        |          |          |--FACTR
        |          |          |
        |          |          |--PRMUTE
        |          |
        |          |--MODSHP--|
        |          |  (402)   |
        |          |          |
        |          |          |--MATPRT
        |          |
```

```
|         |--CTBL  --|
|         |  (403)   |
|         |          |
|         |          |--MATPRT
|         |          |
|         |          |--MXINV
|         |
|         |--OBSBL --|
|         |  (403)   |
|         |          |
|         |          |--MATPRT
|         |
|         |--RESI  --|
|         |  (403)   |
|         |          |
|         |          |--MATPRT
|         |
|         |--NRML
|         |  (404)
|         |
|         |--UNRML --|
|            (405)   |
|                    |
|                    |--MATPRT
|
|--AES500--|
|(503,506,)|
|(509,512,)|
|  (525)   |
|          |
|          |--PREREQ
|          |
|          |--FRSPNS--|
|          |(501,504,)|
|          |(507,510,)|
|          |(513,515,)|
|          |(517,519,)|
|          |(521,523) |
|          |          |
|          |          |--FRQP  --|
|          |          |          |
|          |          |          |--FRPOLY
|          |          |
|          |          |--BOLLIN--|
|          |                     |
|          |                     |--DAVISO
|          |                     |
|          |                     |--DANSKY--|
|          |                                |
|          |                                |--POLMPY
|          |
|          |--BODE  --|
|           (502,505,)|
|           (508,511,)|
|           (514,516,)|
|           (518,520,)|
|           (522,524) |
|                     |
|                     |-PLOTSUBS
|
|--AES600--|
|          |
|          |--PREREQ
|          |
|          |--MATPRT
|          |
|          |--DSCRT
|          |(601,602,)
|          |(603,604)
|          |
|          |--STP   --|
|          |(601,604) |
|          |          |
|          |          |-PLOTSUBS
|          |
|          |--ICRSP --|
|           (602,603) |
|                     |
|                     |-PLOTSUBS
|
|--AES700--|
|          |
|          |--PREREQ
|          |
|          |--MATPRT
|          |
|          |--ZEROES--|
|          |(701,702,)|
|          |(703,704,)|
|          |  (705)   |
|          |          |
|          |          |--ARRAY
|          |          |
|          |          |--HSBG
|          |          |
|          |          |--EIGQR
|          |          |
|          |          |--CONDI --|
|          |                     |
|          |                     |--SCALEA
|          |                     |
|          |                     |--REDU
```

```
|            |
|            |--GAIN
|            |(701,702,)
|            |(703,704,)
|            |  (705)
|            |
|--AES800--|
|(802,806,)|
|(808,810,)|
|(814,816) |
|            |--PREREQ
|            |
|            |--MATPRT
|            |
|            |--CONTRL--|
|            |  (801)    |
|            |           |--MATPRT
|            |           |
|            |           |--RICSS --|
|            |                      |--MATPRT
|            |                      |
|            |                      |--ARRAY
|            |                      |
|            |                      |--MXINV
|            |                      |
|            |                      |--ORDER
|            |                      |
|            |                      |--EIGQR
|            |                      |
|            |                      |--SCALEA
|            |                      |
|            |                      |--HSBG
|            |                      |
|            |                      |--EGCK   --|
|            |                      |
|            |                      |--MODSHP--|
|            |                      |          |--MATPRT
|            |                      |
|            |                      |--EGVCTR--|
|            |                                 |--MATPRT
|            |                                 |
|            |                                 |--ARRAY
|            |                                 |
|            |                                 |--FACTR
|            |                                 |
|            |                                 |--PRMUTE
|            |
|            |--EIGEN --|
|            |(803,805,)|
|            |(811,812,)|
|            |  (813)   |
|            |          |--ARRAY
|            |          |
|            |          |--HSBG
|            |          |
|            |          |--EIGQR
|            |          |
|            |          |--SCALEA
|            |          |
|            |          |--CONDI --|
|            |                     |--SCALEA
|            |                     |
|            |                     |--REDU
|            |
|            |--EGVCTR--|
|            |  (804)   |
|            |          |--MATPRT
|            |          |
|            |          |--ARRAY
|            |          |
|            |          |--FACTR
|            |          |
|            |          |--PRMUTE
|            |
|            |--MODSHP--|
|            |  (804)   |
|            |          |--MATPRT
|            |
|            |--RICCHK--|
|            |(807,815) |
|            |          |--MATPRT
|            |
|            |--ESTMAT--|
|            |  (809)   |
|            |          |--MATPRT
|            |          |
|            |          |--RICSS --|
|            |                     |--MATPRT
|            |                     |
|            |                     |--ARRAY
|            |                     |
|            |                     |--MXINV
```

```
|          |                     |
|          |                     |--ORDER
|          |                     |
|          |                     |--EIGQR
|          |                     |
|          |                     |--SCALEA
|          |                     |
|          |                     |--HSBG
|          |                     |
|          |                     |--EGCK  --|
|          |                     |
|          |                     |--MODSHP--|
|          |                     |          |
|          |                     |          |--MATPRT
|          |                     |
|          |                     |--EGVCTR--|
|          |                                |
|          |                                |--MATPRT
|          |                                |
|          |                                |--ARRAY
|          |                                |
|          |                                |--FACTR
|          |                                |
|          |                                |--PRMUTE
|          |
|          |--COVAR --|
|          |  (817)   |
|          |          |
|          |          |--MATPRT
|          |          |
|          |          |--ARRAY
|          |          |
|          |          |--LAPNV --|
|          |                     |
|          |                     |--MXADD
|          |                     |
|          |                     |--MXMLT
|          |                     |
|          |                     |--MXTRA
|          |                     |
|          |                     |--MXINV
|          |                     |
|          |                     |--DSCA
|          |
|          |--LYPCK --|
|          |  (818)   |
|          |          |
|          |          |--MATPRT
|          |          |
|          |          |--ARRAY
|          |          |
|          |          |--LAPNV --|
|          |                     |
|          |                     |--MXADD
|          |                     |
|          |                     |--MXMLT
|          |                     |
|          |                     |--MXTRA
|          |                     |
|          |                     |--MXINV
|          |                     |
|          |                     |--DSCA
|          |
|          |--MXINV
|            (819)
|
|--AES900--|
(901,902,)|
(903,904) |
           |
           |--UZR901
           |
           |--UZR902
           |
           |--UZR903
           |
           |--UZR904
```

Appendix F
Prerequisite Table

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

Function	Prerequisite function																											
	None	201 or 202	205 or 801	206 or 809	207 or 801	208 or 809	301	302	303	205, 206, 801, or 809	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									X																	
403												X																
404		X																										
405										X			X															
501		X																										
502														X														
503														X														
504		X																										
505															X													
506															X													
507		X																										
508																X												
509																X												
510		X																										
511																	X											
512																	X											
513							X																					
514																	X	X										
515							X																					
516																			X									
517									X																			
518																X				X								
519									X																			
520																	X				X							
521									X																			

TABLE XII. - Concluded.

Function	Prerequisite function																											
	None	201 or 202	205 or 801	206 or 809	207 or 801	208 or 809	301	302	303	205, 206, 801, or 809	401	402	404	501	504	507	510	513	515	517	519	521	523	801	803	809	817	819 or 209
522																						X						
523								X																				
524																							X					
525																							X					
601		X																										
602		X																										
603							X																					
604		X	X				X																					X
701		X																										
702		X																										
703		X																										
704		X																										
705								X																				
801		X																										
802			X																									
803							X																					
804		X																							X			
805					X																							
806					X																							
807																								X				
808																								X				
809		X																										
810				X																								
811								X																				
812									X																			
813						X																						
814						X																						
815																										X		
816																										X		
817		X																										
818																											X	
819							X																					
820																												X
901	X																											
902	X																											
903	X																											
904	X																											

References

1. Lehtinen, B.; and Lorenzo, C.F.: Space Shuttle Active-Pogo-Suppressor Control Design Using Linear Quadratic Regulator Techniques. NASA TP-1217, 1979.
2. Lehtinen, B.; DeHoff, R.L.; and Hackney, R.D.: Multivariable Control Altitude Demonstration on the F100 Turbofan Engine. J. Guidance Control. vol. 4, no. 1, Jan.-Feb. 1981, pp. 50-58.
3. Geyser, L.C.; and Lehtinen, B.: Digital Program for Solving the Linear Stochastic Optimal Control and Estimation Problem. NASA TN D-7820, 1975.
4. Lehtinen, B.; and Zeller, J.R.: Application of Quadratic Optimization to Supersonic Inlet Control. Automatica, vol. 8, no. 9, Sept. 1972, pp. 563-574.
5. Zeller, J.R.; et al.: Analytical and Experimental Performance of Optimal Controller Designs for a Supersonic Inlet. NASA TN D-7188, 1973.
6. Kalman, R.E.; and Englar, T.S.: A User's Manual for the Automatic Synthesis Program. NASA CR-475, 1966.
7. White, J.S.; and Lee, H.Q.: User's Manual for the Variable Dimension Automatic Synthesis Program (VASP). NASA TM X-2417, 1971.
8. Bryson, A.E., Jr.; and Hall, W.E., Jr.: Optimal Control and Filter Synthesis by Eigenvector Decomposition. (SUDAAR-436, Stanford Univ.; NASA Contract NAS2-5143.) NASA CR-152266, 1971.
9. Armstrong, E.S.: ORACLS – A System for Linear-Quadratic-Gaussian Control Law Design. NASA TP-1106, 1978.
10. Konar, A.F.; et al.: Digital Flight Control System for Tactical Fighters, Vol. 2: Documentation of the Digital Control Analysis Software (DIGIKON). Honeywell-08001-VOL-2, Honeywell, Inc., June 1974. (AFFDL-TR-73-119-VOL-2, AD-A002327.)
11. Special Issue – Computer-Aided Design of Control Systems. IEEE Control Systems Magazine, vol. 2, no. 4, Dec. 1982.
12. Kwakernaak, H.; and Sivan, R.: Linear Optimal Control Systems. Wiley-Interscience, 1972.
13. Smith, P.G.: Numerical Solution of the Matrix Equation. $\mathbf{AX}+\mathbf{XA}^T+\mathbf{B}=0$. IEEE Trans. Autom. Control, vol. AC-16, no. 3, June 1971, pp. 278-279.
14. Smith, R.A.: Matrix Equation $\mathbf{XA}+\mathbf{BX}=\mathbf{C}$. SIAM J. Appl. Math., vol. 16, no. 1, Jan. 1968, pp. 198-201.
15. Seidel, R.C.: Computer Programs for Calculation of Matrix Stability and Frequency Response from a State-Space System Description. NASA TM X-71581, 1974.
16. Davison, E.J.: A Computational Method for Finding the Zeroes of a Multivariable Linear Time Invariant System. Automatica, vol. 6, no. 3, May 1970, pp. 481-484.
17. Brockett, R.W.: Poles, Zeros, and Feedback; State Space Interpretation, IEEE Trans. Autom. Control, vol. 10, no. 2, Apr. 1965, pp. 129-135.
18. Kwatny, H.G.; and Fink, L.H.: Acoustics, Stability, and Compensation in Boiling Water Reactor Pressure Control Systems. IEEE Trans. Autom. Control, vol. AC-20, no. 6, Dec. 1975, pp. 727-739.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TP-2221		

4. Title and Subtitle AESOP: An Interactive Computer Program for the Design of Linear Quadratic Regulators and Kalman Filters	5. Report Date January 1984 6. Performing Organization Code 505-34-06
7. Author(s) Bruce Lehtinen and Lucille C. Geyser	8. Performing Organization Report No. E-1686 10. Work Unit No.
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135	11. Contract or Grant No. 13. Type of Report and Period Covered Technical Paper
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

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.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Feedback control; Computer-aided design; Multivariable control systems; Riccati equation; Linear systems; Frequency response; Kalman filter; Linear quadratic regulator	Unclassified - unlimited STAR Category 63

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	113	A06

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

NASA-Langley, 1984

THIS ENVELOPE CONTAINS THE
MICROFICHE REFERENCE SUPPLEMENT
TO NASA TP-2221
OUTPUT OF TEST CASE I
(SUPPLEMENT TO APPENDIX C)