This is to certify that the

thesis entitled
FREQUENCY RESPONSE PROCESSING FOR A VARIETY
OF SYSTEM MODELS

presented by

Gerald A. Gregorski

has been accepted towards fulfillment
of the requirements for

Master's degree in 1985

R.C. Rosenberg

Major professor

Date NOV. 14, 1985

FREQUENCY RESPONSE PROCESSING FOR A VARIETY OF SYSTEM MODELS

By

Gerald A. Gregorski

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering
1985

ABSTRACT

FREQUENCY RESPONSE PROCESSING
FOR A VARIETY OF SYSTEM MODELS

By

Gerald A. Gregorski

Frequency response methods are commonly employed in the design and analysis of dynamic systems. These techniques can be applied to various types of problems such as control system performance and resonance of vibratory systems. An interactive software package has been developed which will perform frequency response processing for most of the typically encountered methods of modeling dynamic systems. The system model can be in the form of a bond graph, a higher-order differential equation, standard vibrations matrices, or a transfer function. Output is in the form of Bode, Nyquist, or root locus plots.

ACKNOWLEDGMENTS

I would like to thank Dr. Ronald Rosenberg for the opportunity to undertake this work and the guidance to accomplish it and also the Case Center for Computer Aided Design for their material and financial support. I would especially like to express my gratitude to my wife and best friend, Christy, for her love, understanding, and support. Your contribution to this work, though unquantifiable, has been more valuable than any other.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	v
Chapter	
1. INTRODUCTION	1
2. THE PACKAGE: CAPABILITIES, STRUCTURE, AND DESIGN	4
2.1 Capabilities	4
2.2 Structure	6
2.3 Design	8
3. FREQUENCY RESPONSE PROCESSING: MODULE F	10
3.1 Structure And Flow	10
3.2 Internal Data Base	12
3.3 Mode Selection And Problem Entry	13
3.4 Interactive Transfer Function Input	15
3.5 Calculating The Results	16
3.6 Displaying And Saving The Results	18
3.7 Problem Statement Output And Filing	19
4. DIFFERENTIAL EQUATION CONVERSION: MODULE C	20
4.1 Structure And Flow	20
4.2 Internal Data Base	22
4.3 Mode Selection And Problem Entry	23
4.4 Interactive Differential Equation Input	24
4.5 Conversion To State Equation Form	25
4.6 Problem Statement Output And Filing	27

Chapter Page

5. EXAMPLES AND USER DOCUMENTATION 28

5.1 Electro-Hydraulic Servomechanism 29

5.2 Three Mass Oscillator 45

5.3 Parallel Geared Drive Shafts 57

6. SUMMARY AND RECOMMENDATIONS 63

LIST OF REFERENCES 65

APPENDICES

A. FILE ORGANIZATION AND CALLING TREES A1

B. MODULE F SOURCE CODE LISTINGS B1

C. MODULE C SOURCE CODE LISTINGS C1

D. UTILITY FILES LISTINGS D1

E. COMMON BLOCKS LISTINGS E1

LIST OF FIGURES

Figure 1.	The Package Structure and Flow	5
Figure 2.	Module F Structure and Flow	11
Figure 3.	Module C Structure and Flow	21
Figure 4.	Electro-Hydraulic Servomechanism	30
Figure 5.	Root Locus for Example 1	31
Figure 6.	Bode Plot for Example 1	32
Figure 7.	Three Mass Oscillator	46
Figure 8.	Frequency Response for Example 2	47
Figure 9.	Parallel Geared Drive Shafts	58
Figure 10.	ABDATA File for Example 3	59
Figure 11.	Frequency Response for Example 3	60

Chapter 1

INTRODUCTION

Frequency response methods are powerful tools for the design and analysis of a wide variety of dynamic systems. These systems can be modeled using a number of techniques, some of which are more suitable or convenient for certain types of systems. In order to use frequency response techniques, the system model must be put in the form of a scalar transfer function. A computer software package has been developed which will perform frequency response processing for systems represented by most of the commonly encountered types of models. The original system model can be in the form of a bond graph[1], a higher-order differential equation, standard vibrations matrices, or a transfer function. If the original model is not a scalar transfer function, then one is generated from the model using the desired input and output. Output of the software is in the form of Bode, Nyquist, or root locus plots. Since this software is a collection of "stand-alone" modules which can communicate through a shared data base and which will be integrated into a larger simulation software in the future, it has not been formally named. The software discussed in this thesis will therefore be referred to as "the package".

In addition to its flexible, multi-model capability, the package is also totally interactive, very robust, and simple to learn and use. The step-by-step, question and answer format allows anyone with a basic

knowledge of the subject of system dynamics to use the package with a minimum of instruction.

None of the numerical algorithms used in this package are new or experimental. They are all well established and documented. The contribution of this work has not been to develop new algorithms, but to incorporate proven routines into an integrated frequency response processing software with the aforementioned qualities: i.e. flexible, interactive, robust, and easy-to-use. Although the numerical algorithms will be referred to with only a minimum of discussion, the appropriate references and appendices will be cited for those desiring additional information. The contributions made toward software design and integration will be stressed throughout this thesis.

The package is written in FORTRAN and is currently implemented on a PRIME 750 computer in the A.H. Case Center for Computer Aided Design at Michigan State University. The basic design of the dialogue format and system structure for the package, as well as the numerical algorithms, were taken from SYSKIT[2], a linear systems analysis software written for micro computers by R.C. Rosenberg and E. Goodman. Much of the code for the package is either original or extensively re-written and modified to take advantage of the greater capabilities of the PRIME 750 in terms of memory, mass storage, and execution speed.

The capabilities, structure, and design of the package will be discussed in Chapter 2. The stand-alone modular concept and shared data base through the file management system will be highlighted. In Chapter 3 the stand-alone module for the frequency response and root locus

processing, Module F, will be discussed in detail. The module for converting ordinary differential equations to state equations, Module C, is covered in Chapter 4.

Chapter 5 consists of several detailed example problems. The examples are selected to illustrate the range of capabilities of the package as well as to give detailed instructions on the use of the various modules. This chapter is intended to be used as a "pull-out" section for user documentation. The integration of ENPORT[3] into the package, the module for bond graph input, is covered variously in Chapters 2,3, and 5. This work is summarized in Chapter 6 and some conclusions are drawn about the utility and applications of the package. Several recommendations for future development are also given.

Chapter 2

THE PACKAGE:
CAPABILITIES, STRUCTURE, AND DESIGN

2.1 CAPABILITIES

The package is capable of performing frequency response processing for physical dynamic systems modeled using bond graphs, higer-order differential equations, standard vibrations matrices, or transfer functions. A scalar transfer function is either generated from the original model using the desired input and output, or it is entered directly as a ratio of two polynomials. These polynomials can be described by their coefficients, roots, or the coefficients of lower-order factors. Both the original model description and the resulting transfer function can be stored in files for later retrieval and re-use and/or modification.

This scalar transfer function can be used to calculate the system's frequency response and root locus. The frequency response results can be plotted in rectangular form, magnitude and phase angle versus frequency, a Bode plot, or in polar form, a Nyquist plot. The root locus also can be plotted. Tabular output for both types of results can be displayed at the terminal, sent to a printer, or stored in a file. Tabular data also includes information such as the crossover frequency, gain and phase margins, and more.

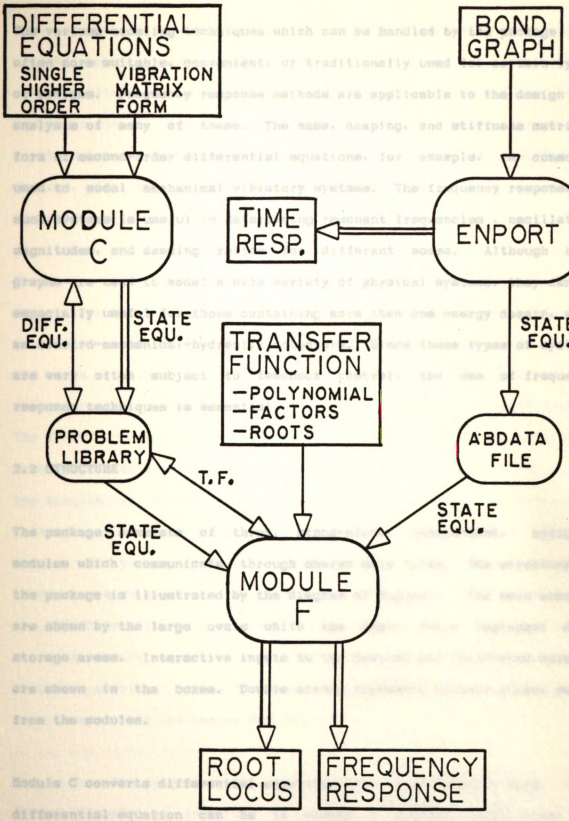


Figure 1. The Package Structure and Flow

The various modeling techniques which can be handled by the package are often more suitable, convenient, or traditionally used for certain types of systems. Frequency response methods are applicable to the design and analysis of many of these. The mass, damping, and stiffness matrices form of second-order differential equations, for example, is commonly used to model mechanical vibratory systems. The frequency response of such systems is useful in determining resonant frequencies , oscillation magnitudes, and damping ratios for different modes. Although bond graphs are used to model a wide variety of physical systems, they can be especially useful for those containing more than one energy domain, such as electro-mechanical-hydraulic actuators. Since these types of systems are very often subject to feedback control, the use of frequency response techniques is essential.

2.2 STRUCTURE

The package consists of three, stand-alone, independent, software modules which communicate through shared data files. The structure of the package is illustrated by the diagram of Figure 1. The main modules are shown by the large ovals while the small ovals represent data storage areas. Interactive inputs to the modules and calculated outputs are shown in the boxes. Double arrows represent primary output paths from the modules.

Module C converts differential equations into state equation form. The differential equation can be in either a single, higher-order or standard vibrations matrix form. User input consists of the coefficients for the single higher-order form and the elements of the

mass, damping, stiffness, and forcing matrices for the vibrations form. A state space representation of the system, A, B, C, and D matrices is automatically generated. Both the differential equations and the state equations can be saved in the problem library. The differential equation, the problem description, can be reloaded and modified if desired. The state space data can be read by Module F to generate a transfer function.

The second module of the package, ENPORT [3], is a commercial software package for the time response processing of bond graphs. The bond graph is entered through its bond, node, and junction structure description and state equations are generated from it. A time response of the selected output states to various input functions is the primary output. The ENPORT program has its own filing system for storing and retrieving bond graph models and time response information which is not shown in the diagram of Figure 1. This is a well established and documented software and a detailed discussion of it is beyond the scope of this thesis. Of primary concern is its ability to communicate with the main processing module through the sharing of state space data. An option in the ENPORT program which allows the user to store the state equations in a formatted data file, called an ABDATA file, is utilized to accomplish this data sharing. Once it's written to an ABDATA file, the state space data is available for use by Module F.

Module F is the frequency response processing module of the package. Direct, interactive input to this module is in the form of a transfer function. The transfer function, as discussed earlier, is input as the ratio of two polynomials which can be described by their coefficients.

roots, or coefficients of lower-order factors. The file management system, which is common to both Modules F and C, allows Module F to read the state equation files created by Module C and stored in the problem library. Module F is also able to read the ABDATA files created by ENPORT. The state space data from either module can then be used to generate a transfer function. The transfer function, whether entered interactively or created from state equation data, can be saved in the problem library for later retrieval. Once reloaded it can be processed with or without modifications. Primary outputs from Module F are the root locus and frequency response results as discussed in the previous section. Both Modules C and F will be discussed later in more detail.

2.3 DESIGN

While the ideas discussed in this section are incorporated in all three modules, they will only be examined in the context of Modules C and F. From the point of view of the package design, the ENPORT module can be thought of as a black box with a bond graph description as input and an ABDATA file as output. Module F simply reads the file and checks for proper data format and problem dimensions.

The package is designed to be totally interactive and easy to use and learn. It is intended that a new user could become relatively competent in its use rather quickly with little chance of getting into trouble. A menu driven, question-and-answer format with safe default answers is used throughout. All inputs from the user are checked for valid type and range. The software will not act on an improper response to a prompt. Extensive error condition checking is also employed. While it

can't be guaranteed as being fool proof, the package is designed to be very robust. It seems highly unlikely that anyone using the package would encounter any problems with error conditions.

Common block storage of internal data has been used extensively. Most arrays within the common blocks are dimensioned by variable parameters which are defined in a control block. It is a simple task to change various package parameters such as maximum problem order or the number of computed data points that can be stored. It is only necessary to change a single value and then re-compile and link the software.

Chapter 3

FREQUENCY RESPONSE PROCESSING: MODULE F

The frequency response and root locus calculations for the package are performed in Module F. This is an independent, stand-alone software which communicates with the other modules of the package through data files stored in the users directory. Its operation is completely interactive using a combination of question-and-answer and menu driven formats. The design and general operation of Module F will be discussed in this chapter. A detailed description of its operation, along with examples, will be postponed until Chapter 5.

3.1 STRUCTURE AND FLOW

The structure and flow paths of Module F are illustrated in the diagram of Figure 2. Each of the blocks in the diagram performs a specific function through a series of related operations. The program begins in the upper left hand block where the internal data base is initialized. Normal flow proceeds sequentially in a counter-clockwise direction following the heavy arrows. A branching point is encountered upon exit from each of the blocks. The user can either proceed on to the next block, the default, or branch off to the main menu from these points.

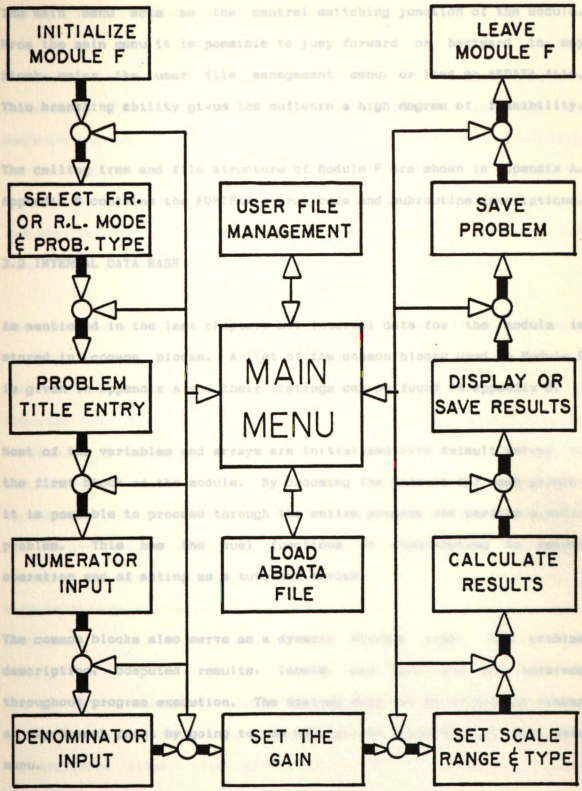


Figure 2. Module F Structure and Flow

The main menu acts as the central switching junction of the module. From the main menu it is possible to jump forward or backward to any block, enter the user file management menu, or load an ABDATA file. This branching ability gives the software a high degree of flexibility.

The calling tree and file structure of Module F are shown in Appendix A. Appendix B contains the FORTRAN source code and subroutine descriptions.

3.2 INTERNAL DATA BASE

As mentioned in the last chapter, all internal data for the module is stored in common blocks. A list of the common blocks used in Module F is given in Appendix A and their listings can be found in Appendix E.

Most of the variables and arrays are initialized with default values in the first block of the module. By choosing the default for each prompt, it is possible to proceed through the entire program and perform a valid problem. This has the dual functions of contributing to robust operation and of acting as a tutorial device.

The common blocks also serve as a dynamic storage area. The problem description, computed results, labels, and more are all retained throughout program execution. The desired data can be altered or viewed at any branch point by going to the appropriate block through the main menu.

3.3 MODE SELECTION AND PROBLEM ENTRY

The Module F program operates in either a frequency response or root locus mode. The mode is selected in the second block of the normal flow sequence and can be switched at any time by returning to this block. All internal data such as the problem description and title is retained when switching modes. This makes it possible, for example, to compute the root locus of a transfer function, select the desired gain, and compute the frequency response of the same transfer function.

The second block is also where the method of problem entry is selected. The problem can be entered in one of four ways: the transfer function can be input interactively, a data file containing a previously stored transfer function can be reloaded, an ENPORT generated ABDATA file can be read, or a state space data file created by Module C can be loaded. The loading of transfer function or state space data files is performed by the user file management system. The other file management options are not available at this point because the menu is bypassed, making its operation transparent to the user. If the file management menu is entered through the main menu, the full range of options becomes available.

The user file management system menu contains several options for performing various tasks such as creating, deleting, loading, or listing the available files. The system manages both the problem and results files and operates in one of two modes depending on which category of files is to be handled. If the filing system is entered through the main menu, the operating mode is selected by the user. If the filing

system menu is bypassed as in the second block, the appropriate mode is automatically set. A separate library file containing the filenames, associated problem titles, and file types is used to bookkeep each category of file. The file type is a code which identifies the type of data contained in the file. This code is used to determine which files can be loaded at a particular point and how they are to be handled if loaded. The ABDATA files are not handled by the filing system because ENPORT simply writes them to the user directory without keeping any record of them.

When a state space problem file or an ABDATA file is loaded, a transfer function is automatically generated from the state equations. If the state equations have multiple inputs and/or outputs, i.e. the column dimension of the input matrix and/or the row dimension of the output matrix is greater than one, the user selects which is to be used. If no output matrix exists, one is formed as an n-dimensional identity matrix, and the user selects the state to be used as output. The transfer function is then formed using the following method[4]. The matrix, [SI-A] is formed, where S is the Laplace operator, I is an n-dimensional identity matrix, and A is the state feedback matrix. The inverse of [SI-A] is then pre-multiplied by the selected row of the output matrix and post-multiplied by the selected column of the input matrix.

The title for the problem can be altered or viewed in the third block. If a problem file is loaded in the second block, the default title set at initialization is replaced by the title contained in the problem file. The current title is retained until changed and is always saved or listed with the problem and any results. The title not only

describes the problem, but also associates the results with the problem from which they were generated.

3.4 INTERACTIVE TRANSFER FUNCTION INPUT

The transfer function is in the form of a ratio of two polynomials multiplied by a gain constant.

$$T(S)=K*N(S)/D(S)$$

Each of the polynomials can be described by its coefficients, roots, or coefficients of several lower-order factors. If the polynomial is entered by its coefficients, a root finding routine[5] determines the roots. If it is entered by its roots, the coefficients are computed by another routine. For factored input, the roots of each factor are found and are used to generate the coefficients of the polynomial. Since the coefficients are normalized with respect to that of the highest power, a system gain is generated from the product of the highest-order coefficients for each factor.

Each of the three methods of input yields both the polynomial coefficients and roots. The factored form is not automatically generated because it is non-unique and not required for calculations. If a polynomial is initially described by factored data and the roots or coefficients are later altered, the factored data will still exist and be available, but it will no longer be associated with the problem.

The transfer function description can be input, modified, or viewed using the fourth, fifth, and sixth blocks. The user should be aware that any time one of the polynomial input blocks is entered the entire

polynomial data base is re-generated. The data associated with the polynomial description method selected when entering the block is used to generate the rest of the data base. The user should be certain that the polynomial data is correct before leaving one of the polynomial input blocks.

3.5 CALCULATING THE RESULTS

The range and type of scale used for the calculations is selected in the seventh block of the sequence and is used for both problem modes. The range of frequencies for frequency processing or gains for root locus calculations is entered along with the number of data points desired over this range. The frequency and gain values for the calculations can be generated on either a logarithmic or linear scale.

In the frequency processing mode[6], each polynomial is represented as a product of first-order factors of the form, (S-ROOTi), where each ROOTi is a root of the polynomial and is , in general, complex. If we replace S, the Laplace operator, by jWi, where Wi is a frequency, each factor becomes a complex number. These complex numbers can be represented in polar form by their magnitude and angle. The product of the numerator magnitudes, gain constant, and any system gain, divided by the product of the denominator magnitudes gives the the magnitude of the sinusoidal response for each frequency. The sum of the numerator angles minus the sum of the denominator angles gives the phase angle for each frequency.

The calculated frequency response is valid for any transfer function entered. It is the users responsibility to be aware of what the transfer function represents and to interpret the results accordingly. For example, much of the information given with the tabulated results, such as phase and gain margins, is only applicable to feedback control system analysis using the open-loop transfer function.

The applications for the root locus mode[6] are more restricted. It is used to determine the closed-loop poles of a feedback control system, as a function of system gain, from its open-loop transfer function. It is important that the user understand that the root locus results are only valid for an open-loop transfer function which can be closed by a feedback loop.

The open-loop transfer function is in the form of, To(S)=k*G(S)*H(S), where G(S) is the feedforward block, H(S) is the feedback block, and K is the forward path gain. The closed-loop transfer function is in the form of, Tc(S)=K*G(S)/[1+K*G(S)*H(S)]. The closed-loop poles are the solutions of the equation, K*G(S)*H(S)=-1. This can be put in the form of, K*N(S)+D(S)=0, where N(S) and D(S) are the numerator and denominator polynomials of the open-loop transfer function. The numerator coefficients are multiplied by the gain and a new polynomial is formed by combining these with the denominator coefficients. The roots of this polynomial are found as the gain is varied to generate the root locus for the closed-loop system.

3.6 DISPLAYING AND SAVING THE RESULTS

Once the results are calculated, flow proceeds to the ninth block where they can be displayed or saved in various forms. The tabulated results can be viewed on the screen, sent to a line printer, or saved in a file. The results can also be plotted on the screen and saved in graphics output files. The frequency response results can be plotted in rectangular form, magnitude and phase versus frequency, a Bode plot, or in polar form, a Nyquist plot. The root locus can, of course, also be plotted.

In addition to the problem title, a separate title can be given to each set of results. This can be useful in identifying any special conditions under which the particular set of results was generated from the problem. The frequency or gain range for results output can be any continuous subset of the total range used for calculations.

The results are stored in data files using the user file management system operating in the results mode. The filing system menu is bypassed at this point and its operation is transparent to the user, just as in the loading of problem files in the second block. The graphics output files are simply created in the users directory and are not handled by the filing system. They have no further use as far as the package is considered and are usually deleted after a hardcopy output is generated.

3.7 PROBLEM STATEMENT OUTPUT AND FILING

The user is given the option of filing and/or listing the problem description in the tenth block of the sequence. The problem statement consists primarily of the polynomial coefficients and roots and the coefficients of any factors. The filing of the problem statement is done transparently through the file management system at this point. The final block gives the option of either leaving the module or returning to the main menu for further work.

Chapter 4

DIFFERENTIAL EQUATION CONVERSION: MODULE C

Ordinary differential equations, in either single, higher-order or vibrations matrix form, are converted to state equation form in Module C. This is an independent, stand-alone software which communicates with the other modules of the package through data files stored in the users directory. Its operation is completely interactive using a combination of question-and-answer and menu driven formats just as in Module F. Since Modules C and F are very similar in terms of structure, flow, design, and operation, much of the material in this chapter is also covered in Chapter 3. However, since Module C operates independently of the other modules in the package, a discussion of its design and operation should also be able to stand alone. A detailed description of its operation through examples and interactive sessions will be deferred till Chapter 5.

4.1 STRUCTURE AND FLOW

The structure and flow paths of Module C are illustrated in the diagram of Figure 3. A specific task, involving a series of related operations, is performed in each of the blocks of the diagram. Normal program flow begins in the upper right-hand block and proceeds sequentially following the heavy arrows.

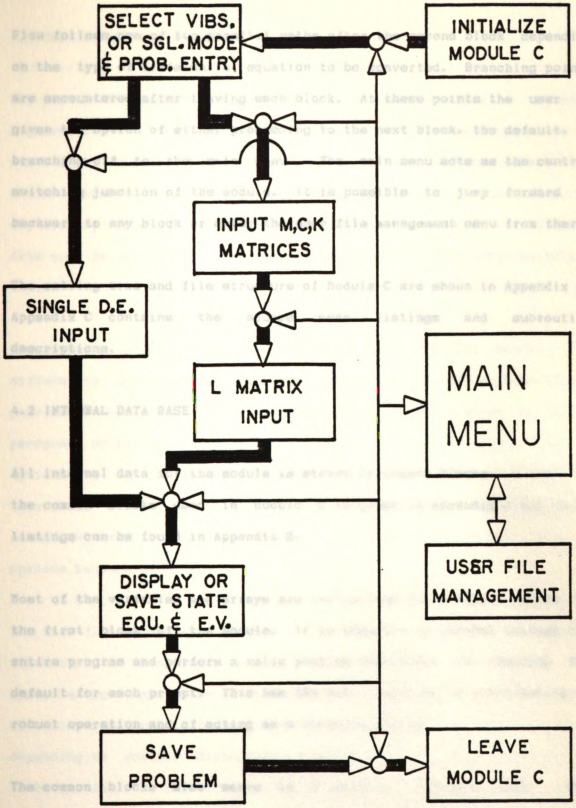


Figure 3. Module C Structure and Flow

Flow follows one of two parallel paths after the second block depending on the type of differential equation to be converted. Branching points are encountered after leaving each block. At these points the user is given the option of either proceeding to the next block, the default, or branching off to the main menu. The main menu acts as the central switching junction of the module. It is possible to jump forward or backward to any block or enter the user file management menu from there. The calling tree and file structure of Module C are shown in Appendix A. Appendix C contains the source code listings and subroutine descriptions.

4.2 INTERNAL DATA BASE

All internal data for the module is stored in common blocks. A list of the common blocks used in Module C is given in Appendix A and their listings can be found in Appendix E.

Most of the variables and arrays are initialized with default values in the first block of the module. It is possible to proceed through the entire program and perform a valid problem conversion by choosing the default for each prompt. This has the dual functions of contributing to robust operation and of acting as a tutorial device.

The common blocks also serve as a dynamic storage area. The differential equation, state equations, labels, and more are all retained during program execution. The desired data can be altered or viewed by going to the appropriate block through the main menu.

4.3 MODE SELECTION AND PROBLEM ENTRY

The Module C program can be used to convert either a single, higher-order or vibrations matrix form of a differential equation to state equation form. The mode in which the program operates is selected in the second block of the flow sequence. The mode can be switched at any time by returning to this block through the main menu. All internal data such as problem descriptions and titles are retained when switching modes.

The method of problem entry is also selected in the second block. The differential equation can either be input interactively or read from a previously created data file. The loading of these problem files is performed by the user file management system. The other file management options are not available at this point because the menu is bypassed. This makes its operation transparent to the user. If the file management menu is entered through the main menu, the full range of options becomes available.

4.4 INTERACTIVE DIFFERENTIAL EQUATION INPUT

The user file management system contains several options for performing tasks such as creating, loading, or listing the available files. The system manages both types of differential equation problem files as well as state equation data files. The system operates in one of two modes depending on whether differential equation or state equation files are to be handled. If the filing system is entered through the main menu, the operating mode is selected by the user. If the filing system menu is bypassed, as in the second block, the appropriate mode is automatically set. A library file containing the filenames, associated

problem titles, and file types is used to bookkeep the users files. The file type is a code which identifies the type of data contained in the file. Each of the three types of files used in Module C has a different code number. This code is used to determine which files can be loaded at a particular point and how they are to be handled if loaded. Since Module C does not generate any computed results, all three types of files are bookkept by the problem library. State equation problem files are not allowed to be reloaded by Module C, but all other filing system operations can be performed on them.

The problem title can be altered or viewed in the third block which is not shown in Figure 3. If a problem file is loaded in the second block, the current default title is replaced by the title contained in the problem file. The current title is retained until changed and is always saved or listed with both the problem description and the state-space representation. The title not only describes the problem, but also links the differential equation with its state equation form.

4.4 INTERACTIVE DIFFERENTIAL EQUATION INPUT

Normal program flow proceeds down one of two parallel paths after the second block depending on which mode is selected. If the problem is in the form of a single, higher-order differential equation, the left path is followed. If the differential equation is in standard vibrations form, the right-hand path is taken. The single differential equation is input as the coefficients of the time derivatives of both the dependent variable and the input. The vibrations matrix form is input as the elements of the mass, damping, stiffness, and input matrices. The

matrices are input and modified using a menu driven set of operations. Options include changing a single element, an entire row or column, or viewing the entire matrix. In either case, the user only proceeds on to the next step when the data is satisfactory. Both of these input paths can be used to input, modify, or view the problem for the current mode. Only the path corresponding to the current problem mode is open. For example, if the program is in the vibrations mode, it is necessary to first change modes before it is possible to view the current single differential equation problem.

4.5 CONVERSION TO STATE EQUATION FORM

Any time one of the differential equation input paths are entered, the corresponding state space representation is automatically updated using the problem data for the current mode. If a problem file is loaded in the second block, new state space data is not generated until program flow goes through the proper input path. The data from the problem file becomes the new default values for the prompts and are selected accordingly. When using one of the input paths to only view the current data, care should be taken that it is not inadvertently altered. Only the default values, the current problem data, for the prompts should be used.

The state variables for the single, higher-order input mode are selected such that any derivatives of the input are eliminated and the first state variable is the dependent variable of the differential equation. The remaining state variables will be combinations of the first state variable and its derivatives and the input. The only state available

through the output matrix is the first since it is the only one of any real interest or practical value.

The vibrations form of the differential equation[7] is input as mass, damping, and stiffness square matrices of order N, and an NxM input matrix where M is the number of inputs. Since this is a set of N second-order differential equations, the conversion results in 2N state variables/equations. The state vector consists of the displacements and velocities of N masses. State variables are chosen such that the first N states are the mass displacements and the second N states are the time derivatives of the first, the mass velocities. The output matrix is generated such that only the first N states, the mass displacements, may be used as output. Only the mass displacements are usually of interest for frequency response processing.

In both of the above cases, the characteristic equation, determinant[SI-A]=0, is formed. A root finding routine[5], the same as in Module F, is used to find the roots of this equation, the system eigenvalues. The eigenvalues are included in the state space data. They are also used by Module F to generate the denominator polynomial of the systems transfer function.

Only one state space representation exists at any time. It is associated with the problem which has been most recently processed through one of the input paths. After the state equations are generated, normal program flow proceeds to the block where both input paths converge. At this point the state equations and eigenvalues can be displayed on the screen, sent to printer, and/or saved in a file for

use by another module. The saving of state space data in files is done through the file management sytem. The filing system menu is bypassed at this point and its operation is transparent to the user, just as in the second block. The problem title is part of the data in all three options in order to link the state space data with the problem from which it is was generated.

4.6 PROBLEM STATEMENT OUTPUT AND FILING

The user is given the option of filing and/or printing the differential equation problem description in the second to last block of the sequence. The problem statement consists of the coefficients of the derivatives for the single input mode, and the mass, damping, stiffness, and input matrices for the vibrations mode. The filing of the problem statement is done transparently through the file management system at this point. The final block of the sequence gives the option of either leaving the module or returning to the main menu for further work.

Chapter 5

EXAMPLES AND USER DOCUMENTATION

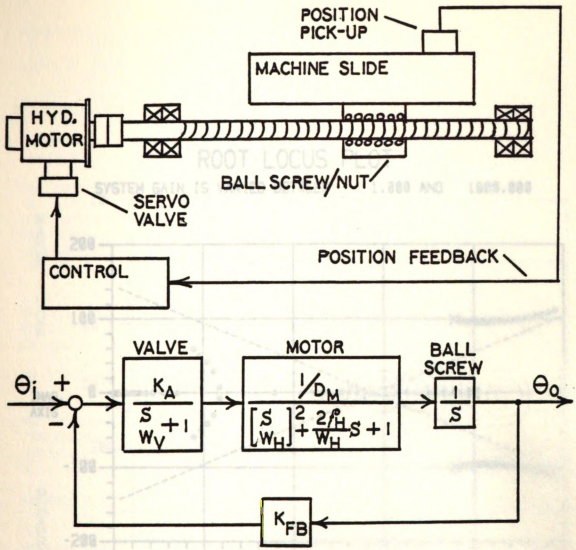
The capabilities and operation of the package will be illustrated in this chapter. The three example problems demonstrate both the capabilities of the package and the integration and operation of each of the individual modules. Each example consists of: a physical problem statement with required input information, the problem results, and an interactive problem session. Each interactive session is enhanced with comments and explanations and is intended to emphasize different features of the package and/or module. This chapter is meant to be used as a "pull-out" section for user documentation. It can be used alone or in combination with chapters 2, 3, and 4.

The first example deals with a typical automatic controls problem. It illustrates most of the main features of Module F in some detail. In the other two examples, only certain selected features of Module F will be shown in detail. The second example is a classic vibrations problem. It involves both Modules C and F. Many of the details of Module C are not shown because of its similarity with Module F. The final example illustrates the integration of ENPORT into the package through the use of ABDATA files.

5.1 ELECTRO-HYDRAULIC SERVOMECHANISM

A common type of electro-hydraulic servomechanism, such as that used in machine tools[8], is shown in Figure 4. A servo valve controls the fluid flow to a hydraulic motor. The motor drives a ballscrew which imparts linear motion to a machine slide. The slide position is sensed and summed with the input command to generate an error signal. This signal is the input to the servo valve. The transfer functions of the individual components are combined in a single block diagram. The open-loop transfer function is then determined using block diagram algebra.

This example demonstrates the operation of Module F in detail. The open-loop transfer function is entered interactively through its numerator and denominator polynomials. The root locus mode is selected and the closed-loop poles are computed for a range of gains. A plot of this root locus is shown in Figure 5. The processing mode is switched, a gain is selected from the root locus results, and the frequency response is computed. The Bode plot for the selected gain is shown in Figure 6. The problem statement is then saved in a file before leaving the module.



PARAMETERS:

$\zeta_H=0.1$, $W_H=100$, $W_V=200$, $K_{FB}=1$, $D_M=5$, $K_A=100$

OPEN-LOOP TRANSFER FUNCTION:

$$T(S)=\frac{\Theta_i}{\Theta_o}=\frac{0.2\cdot K_A}{S\left[.005\cdot S+1\right]\left[.0001\cdot S^2+.002\cdot S+1\right]}$$

Figure 4. Electro-Hydraulic Servomechanism

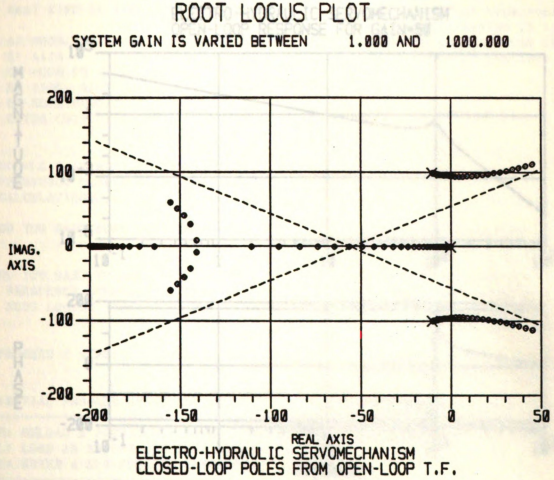


Figure 5. Root Locus for Example 1

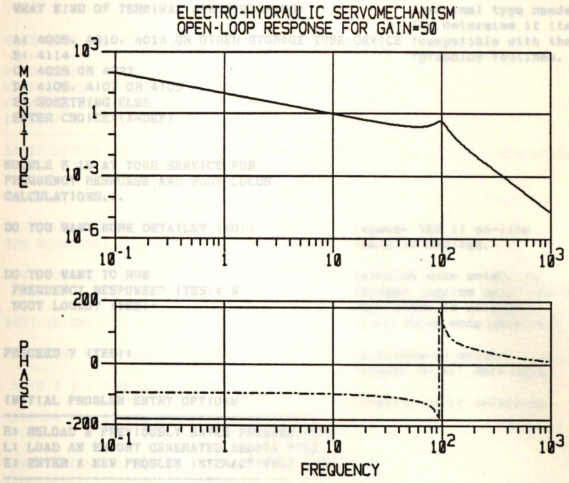


Figure 6. Bode Plot for Example 1

INTERACTIVE SESSION FOR EXAMPLE 1

```
OK, RUN77 SYSKIT>#MODULF                    !begin program execution.

 WHAT KIND OF TERMINAL ARE YOU USING?              !terminal type needed
                                                   !to determine if its
 A: 4006, 4010, 4014 OR OTHER STORAGE TUBE DEVICE  !compatible with the
 B: 4114                                           !graphics routines.
 C: 4025 OR 4027
 D: 4105, 4107 OR 4109
 E: SOMETHING ELSE
 ENTER CHOICE:(A=DEF)

MODULE F IS AT YOUR SERVICE FOR
FREQUENCY RESPONSE AND ROOT LOCUS
CALCULATIONS...

DO YOU WANT MORE DETAILS? (NO):             !answer YES if on-line
                                            !help is desired.

DO YOU WANT TO RUN                          !problem mode selection.
 FREQUENCY RESPONSE? (YES): N               !prompt toggles until one of
 ROOT LOCUS? (YES):                         !the modes is selected.
                                            !root locus mode selected.

PROCEED ? (YES):                            !a branching point.
                                            !answer NO for main menu.

INITIAL PROBLEM ENTRY OPTIONS               !problem entry selection.
----------------------------------------
R: RELOAD A PREVIOUSLY SAVED PROBLEM FILE
L: LOAD AN ENPORT GENERATED ABDATA FILE
E: ENTER A NEW PROBLEM INTERACTIVELY (=DEF)
----------------------------------------
ENTER OPTION (E):                           !interactive input selected.
                                            !the default option.

THE PROBLEM TITLE IS:                       !default problem title.

*** MODULE F PROBLEM ***

CHANGE THE TITLE? (NO): Y                   !enter a new title.


ENTER NEW TITLE ON ONE LINE:

ELECTRO-HYDRAULIC SERVOMECHANISM
```

```
THE PROBLEM TITLE IS:                          !display the new title.

ELECTRO-HYDRAULIC SERVOMECHANISM

CHANGE THE TITLE? (NO):                        !last chance to change it.

PROCEED? (YES):                                !another branch point.

NUMERATOR ENTRY OPTIONS                        !select the method for
--------------------------------               !defining the numerator.
P: SINGLE POLYNOMIAL (=DEF)
F: FACTORED FORM
R: ROOTS OF THE POLYNOMIAL
H: HELP
--------------------------------
ENTER OPTION (P):                              !input it as a single poly.

ORDER OF NUMERATOR? ( 1): 0                    !enter order of numerator.

THE NUMERATOR POLYNOMIAL:                      !display current numerator
                                               !polynomial (the default).
  5.000E+00 * S**0

WANT TO CHANGE IT? (NO): Y                     !option to change the poly.

ENTER THE NEW COEFFICIENTS:                    !enter the new coefficients.
                                               !default was 5.00
 S**0 ? ( 5.000E+00): .2

WANT TO SEE IT AGAIN? (NO):                    !options to look at the
                                               !polynomial and its roots.
WANT TO SEE THE ROOTS? (NO):

PROCEED ? (YES):                               !another branch point.

DENOMINATOR ENTRY OPTIONS                      !select the method for
--------------------------------               !defining the denominator.
P: SINGLE POLYNOMIAL (=DEF)
F: FACTORED FORM
R: ROOTS OF THE POLYNOMIAL
H: HELP
--------------------------------
ENTER OPTION (P): F                            !select factored input.

ORDER OF DENOMINATOR? ( 3): 4                  !complete denominator order.
```

```
ORDER OF FACTOR 1? ( 2): 1                    !order of first den. factor.

THE POLYNOMIAL FOR FACTOR  1:                 !first factor default poly.

  1.000E+00 * S**1
  1.000E+00 * S**0

WANT TO SEE ROOTS OF FACTOR  1? (NO):         !default roots.

WANT TO CHANGE FACTOR  1? (NO): Y             !enter new first factor.

ENTER THE NEW COEFFICIENTS:

 S**1 ? ( 1.000E+00):
 S**0 ? ( 1.000E+00): 0                       !change default coeff.

WANT TO SEE FACTOR  1 AGAIN? (NO): Y          !display new first factor.

THE POLYNOMIAL FOR FACTOR  1:

  1.000E+00 * S**1
  0.000E-01 * S**0

WANT TO SEE ROOTS OF FACTOR  1? (NO): Y       !show roots of first poly.


THE ROOTS FOR FACTOR  1:

  REAL PART       IMAG PART
 0.000E-01       0.000E-01

WANT TO CHANGE FACTOR  1? (NO):               !last chance to change it.

ORDER OF FACTOR 2? ( 2): 1                    !order for second den. fact.


THE POLYNOMIAL FOR FACTOR  2:                 !default poly. for second.

  1.000E+00 * S**1
  1.000E+00 * S**0

WANT TO SEE ROOTS OF FACTOR  2? (NO):

WANT TO CHANGE FACTOR  2? (NO): Y             !modify default poly.

ENTER THE NEW COEFFICIENTS:

 S**1 ? ( 1.000E+00): .005                    !enter new first coeff.
 S**0 ? ( 1.000E+00):

WANT TO SEE FACTOR  2 AGAIN? (NO): Y          !show second poly. again.
```

```
THE POLYNOMIAL FOR FACTOR  2:
  5.000E-03 * S**1
  1.000E+00 * S**0
WANT TO SEE ROOTS OF FACTOR  2? (NO): Y     !display its roots.

THE ROOTS FOR FACTOR  2:
 REAL PART       IMAG PART
-2.000E+02       0.000E-01
WANT TO CHANGE FACTOR  2? (NO):             !last chance to change it.

ORDER OF FACTOR 3? ( 2):                    !select default order.

THE POLYNOMIAL FOR FACTOR  3:               !display default poly.
  1.000E+00 * S**2
  1.000E+00 * S**1
  1.000E+00 * S**0

WANT TO SEE ROOTS OF FACTOR  3? (NO):


WANT TO CHANGE FACTOR  3? (NO): Y           !enter new third factor.

ENTER THE NEW COEFFICIENTS:

 S**2 ? ( 1.000E+00): .0001                 !change first two coeff.
 S**1 ? ( 1.000E+00): .002
 S**0 ? ( 1.000E+00):

WANT TO SEE FACTOR  3 AGAIN? (NO): Y        !look at it again.

THE POLYNOMIAL FOR FACTOR  3:
  1.000E-04 * S**2
  2.000E-03 * S**1
  1.000E+00 * S**0
WANT TO SEE ROOTS OF FACTOR  3? (NO): Y     !show roots of third fact.

THE ROOTS FOR FACTOR  3:
 REAL PART       IMAG PART
-1.000E+01      -9.950E+01
-1.000E+01       9.950E+01
```

```
WANT TO CHANGE FACTOR  3? (NO):          !last chance to change it.

WANT TO SEE ALL OF THE ROOTS? (NO): Y     !display all of the
                                          !denominator roots.
THE DENOMINATOR ROOTS (POLES):

 REAL PART      IMAG PART
 0.000E-01      0.000E-01
-2.000E+02      0.000E-01
-1.000E+01     -9.950E+01
-1.000E+01      9.950E+01

WANT TO SEE THE POLYNOMIAL? (NO): Y       !display the entire
                                          !denominator polynomial.
THE DENOMINATOR POLYNOMIAL:

  5.000E-07 * S**4
  1.100E-04 * S**3
  7.000E-03 * S**2
  1.000E+00 * S**1
  0.000E-01 * S**0

PROCEED ? (YES):                          !another branch point.

SET THE GAIN G...                         !gain of transfer function.
                                          !gain of 1 has no effect in
GAIN? ( 0.100E+01): :                     !the root locus mode.

PROCEED ? (YES):                          !another branch point.

SET THE K GAIN RANGE...                   !set range of gains for
                                          !root locus calculation.
 MINIMUM? ( 1.000E+00): :

 MAXIMUM? ( 1.000E+01): 1000

THE SCALING IS LOG                        !put gains on a log scale.
                                          !could change it to linear.
DO YOU WANT TO CHANGE IT? (NO): :


PROCEED ? (YES):                          !another branch point.

 NUMBER OF SOLUTION POINTS? ( 25): 400    !number of gains to be
                                          !generated across the range.
ENTER INTEGER FROM     3 TO   100: 50     !first entry out of bounds.
                                          !must be within limits.
```

```
PROCEED? (YES):


CALCULATING THE ROOT LOCUS.                    !processing the problem.
                                               !closed loop poles being
 ONE MOMENT, PLEASE...                         !computed for each gain.


RESULT OPTIONS                                 !options for displaying or
----------------------                         !saving the root locus data.
S:  DISPLAY ON SCREEN
L:  LIST TO PRINTER
W:  WRITE TO FILE
G:  PLOT ROOT LOCUS
R:  RETURN TO MAIN MENU
P:  PROCEED(=DEF)
----------------------
ENTER OPTION (P): S                            !display results on screen.


SET THE TITLE FOR RESULTS...                   !results title distinguishes
                                               !them from other results.
THE RESULTS HEADING IS:
*** MODULE F RESULTS ***                       !default results title.

CHANGE THE HEADING? (NO): Y                    !change the default title.

 ENTER A HEADING ON ONE LINE:
CLOSED-LOOP POLES FROM OPEN-LOOP T.F.

THE RESULTS HEADING IS:                        !display new title.
CLOSED-LOOP POLES FROM OPEN-LOOP T.F.

CHANGE THE HEADING? (NO):                      !last chance to change it.


SELECT THE GAIN LIMITS                         !any subset of results.
 FOR DISPLAY OR PLOTTING...

LOW  GAIN? ( 1.000E+00):                       !select entire range.
HIGH GAIN? ( 1.000E+03):


HIT <RETURN> TO START DISPLAY.                 !start of results output.
                                               !one screen full at a time.
```

```
PROCEED? (YES):

CALCULATING THE ROOT LOCUS.                 !processing the problem.
                                            !closed loop poles being
 ONE MOMENT, PLEASE...                      !computed for each gain.

RESULT OPTIONS                              !options for displaying or
-----------------------                     !saving the root locus data.
S:  DISPLAY ON SCREEN
L:  LIST TO PRINTER
W:  WRITE TO FILE
G:  PLOT ROOT LOCUS
R:  RETURN TO MAIN MENU
P:  PROCEED(=DEF)
-----------------------
ENTER OPTION (P): S                         !display results on screen.

SET THE TITLE FOR RESULTS...                !results title distinguishes
                                            !them from other results.
THE RESULTS HEADING IS:
*** MODULE F RESULTS ***                    !default results title.

CHANGE THE HEADING? (NO): Y                 !change the default title.

 ENTER A HEADING ON ONE LINE:
CLOSED-LOOP POLES FROM OPEN-LOOP T.F.

THE RESULTS HEADING IS:                     !display new title.
CLOSED-LOOP POLES FROM OPEN-LOOP T.F.

CHANGE THE HEADING? (NO):                   !last chance to change it.

SELECT THE GAIN LIMITS                      !any subset of results.
 FOR DISPLAY OR PLOTTING...

LOW  GAIN? ( 1.000E+00):                    !select entire range.
HIGH GAIN? ( 1.000E+03):


HIT <RETURN> TO START DISPLAY.              !start of results output.
                                            !one screen full at a time.
```

```
******************** MODULE F ROOT LOCUS DATA ********************

ELECTRO-HYDRAULIC SERVOMECHANISM

CLOSED-LOOP POLES FROM OPEN-LOOP T.F.

THE GAIN SCALE IS: LOG

THE RANGE OF GAINS IS  0.100E+01  TO  0.100E+04

THERE ARE  4 POLES AT:     REAL         IMAG.
                        0.000E+00   0.000E+00
                       -0.200E+03   0.000E+00
                       -0.100E+02  -0.995E+02
                       -0.100E+02   0.995E+02

THERE ARE  0 ZEROS AT:     REAL         IMAG.

THE POLE EXCESS =   4

THE REAL COORDINATE OF ASYMPTOTE ORIGIN = -0.550E+02

THE ASYMPTOTE ANGLES IN DEGREES:
                                    0.450E+02
                                    0.135E+03
                                    0.225E+03
                                    0.315E+03

WANT TO SEE MORE? (YES):                    !show another screen full.


    GAIN       REAL PART    IMAG. PART


  0.100E+01  -0.992E+01  -0.994E+02
             -0.992E+01   0.994E+02
             -0.200E+03   0.000E+00
             -0.200E+00   0.000E+00

  0.115E+01  -0.991E+01  -0.994E+02
             -0.991E+01   0.994E+02
             -0.200E+03   0.000E+00
             -0.231E+00   0.000E+00

  0.133E+01  -0.200E+03   0.000E+00
             -0.990E+01  -0.994E+02
             -0.990E+01   0.994E+02
             -0.266E+00   0.000E+00

  0.153E+01  -0.988E+01  -0.994E+02
```

```
                -0.988E+01    0.994E+02
                -0.200E+03    0.000E+00
                -0.306E+00    0.000E+00

WANT TO SEE MORE? (YES): N                  !end results output.


RESULT OPTIONS                              !return to output menu.
----------------------                      !all options operate the
S:  DISPLAY ON SCREEN                       !same as screen display.
L:  LIST TO PRINTER                         !could also plot the root
W:  WRITE TO FILE                           !locus and save it in a
G:  PLOT ROOT LOCUS                         !graphics file.
R:  RETURN TO MAIN MENU                     !just follow the prompts.
P:  PROCEED(=DEF)
----------------------
ENTER OPTION (P): R                         !return to main menu.


 MODULE F OPTIONS                           !main menu for Module F.
---------------------------------------     !can go anywhere from here.
 1: SELECT MODE AND PROBLEM ENTRY
 2: ALTER THE PROBLEM TITLE
 3: CHANGE THE NUMERATOR
 4: ALTER THE DENOMINATOR
 5: CHANGE THE GAIN
 6: ALTER FREQUENCY OR LOCUS SCALE
 7: CALCULATE RESULTS
 8: DISPLAY OR SAVE THE RESULTS
 9: PROBLEM OR RESULTS FILE MANAGEMENT
10: LOAD AN ENPORT ABDATA FILE
11: SAVE PROBLEM STATEMENT AND RETURN
12: CONTINUE FROM LAST ACTION (=DEF)
13: LEAVE THIS MODULE
---------------------------------------
PLEASE ENTER OPTION (12): 1                 !change program mode.

DO YOU WANT TO RUN                          !select freq. resp. mode.
 FREQUENCY RESPONSE? (YES):


PROCEED ? (YES): N                          !another branch point.
                                            !return to main menu after
                                            !changing modes.
 MODULE F OPTIONS
---------------------------------------
                                            !menu not shown here
                                            !for sake of brevity.
---------------------------------------
PLEASE ENTER OPTION (12): 5                 !select a new gain.


SET THE GAIN G...
```

```
GAIN? ( 0.100E+01): 50                    !set gain according to the
                                          !root locus results.

PROCEED ? (YES):                          !another branch point.


SET THE FREQUENCY RANGE...                !select the range and scale
                                          !for frequency response.
 MINIMUM? ( 1.000E-01): .1

 MAXIMUM? ( 2.000E+03): 1000

THE SCALING IS LOG

DO YOU WANT TO CHANGE IT? (NO):


PROCEED ? (YES):                          !another branch point.

 NUMBER OF SOLUTION POINTS? ( 50): 500    !number of frequencies
                                          !over selected range.
PROCEED? (YES):                           !another branch point.


CALCULATING THE FREQUENCY RESPONSE.       !performing calculations.

ONE MOMENT, PLEASE...


RESULT OPTIONS                            !back to the results menu.
----------------------
S:  DISPLAY ON SCREEN
L:  LIST TO PRINTER
W:  WRITE TO FILE
B:  BODE PLOT
N:  NYQUIST PLOT
R:  RETURN TO MAIN MENU
P:  PROCEED(=DEF)
----------------------
ENTER OPTION (P): S                       !display results to screen.


SET THE TITLE FOR RESULTS...              !new title for new results.

THE RESULTS HEADING IS:                   !old title is default.
CLOSED-LOOP POLES FROM OPEN-LOOP T.F.

CHANGE THE HEADING? (NO): Y               !enter a new title for
                                          !frequency response results.
 ENTER A HEADING ON ONE LINE:
OPEN-LOOP RESPONSE FOR GAIN=50

 THE RESULTS HEADING IS:                  !display new results title.
```

```
OPEN-LOOP RESPONSE FOR GAIN=50

CHANGE THE HEADING? (NO):

SELECT THE FREQUENCY RANGE                    !select any subset of the
FOR DISPLAY OR PLOTTING...                    !calculated results.

LOW  FREQUENCY? ( 1.000E-01):                 !select entire range.
HIGH FREQUENCY? ( 1.000E+03):

HIT <RETURN> TO START DISPLAY.                !start of screen display.

******************** MODULE F FREQUENCY RESPONSE ********************

ELECTRO-HYDRAULIC SERVOMECHANISM

OPEN-LOOP RESPONSE FOR GAIN=50

THE FREQUENCY SCALE IS: LOG

THE FREQUENCY RANGE IS   0.100E+00  TO  0.100E+04 RAD./SEC.

THE MINIMUM MAGNITUDE =  0.198E-04  AT  0.100E+04 RAD./SEC.

THE MAXIMUM MAGNITUDE =  0.100E+03  AT  0.100E+00 RAD./SEC.

THE CROSSOVER FREQUENCY =  0.101E+02

THE PHASE MARGIN =  0.859E+02

THE FREQUENCY AT -180 PHASE SHIFT =  0.953E+02

THE GAIN MARGIN =  0.224E+01  OR  0.699E+01 DECIBELS

WANT TO SEE MORE? (YES):

  FREQUENCY    MAGNITUDE    PHASE(DEG)  PHASE(RAD)

  0.100E+00    0.100E+03  -0.900E+02  -0.157E+01
  0.102E+00    0.982E+02  -0.900E+02  -0.157E+01
  0.104E+00    0.964E+02  -0.900E+02  -0.157E+01
  0.106E+00    0.946E+02  -0.900E+02  -0.157E+01
  0.108E+00    0.929E+02  -0.900E+02  -0.157E+01
  0.110E+00    0.912E+02  -0.901E+02  -0.157E+01
  0.112E+00    0.895E+02  -0.901E+02  -0.157E+01
  0.114E+00    0.879E+02  -0.901E+02  -0.157E+01
```

```
  0.116E+00    0.863E+02  -0.901E+02  -0.157E+01
  0.118E+00    0.847E+02  -0.901E+02  -0.157E+01
  0.120E+00    0.831E+02  -0.901E+02  -0.157E+01
  0.123E+00    0.816E+02  -0.901E+02  -0.157E+01
  0.125E+00    0.801E+02  -0.901E+02  -0.157E+01
  0.127E+00    0.787E+02  -0.901E+02  -0.157E+01
  0.129E+00    0.772E+02  -0.901E+02  -0.157E+01
  0.132E+00    0.758E+02  -0.901E+02  -0.157E+01
  0.134E+00    0.744E+02  -0.901E+02  -0.157E+01
  0.137E+00    0.731E+02  -0.901E+02  -0.157E+01
  0.139E+00    0.717E+02  -0.901E+02  -0.157E+01

WANT TO SEE MORE? (YES): N                    !end screen output.


RESULT OPTIONS                                !back to results menu.
----------------------
S:  DISPLAY ON SCREEN
L:  LIST TO PRINTER
W:  WRITE TO FILE
B:  BODE PLOT
N:  NYQUIST PLOT
R:  RETURN TO MAIN MENU
P:  PROCEED(=DEF)
----------------------
ENTER OPTION (P):                             !proceed on in program.


DO YOU WANT TO LIST                           !option to print out the
 THE CURRENT PROBLEM STATEMENT                !problem statement.
  ON THE PRINTER? (NO): Y


READY THE PRINTER                             !automatically goes to
 AND HIT <RETURN>...                          !the printer.

DO YOU WISH TO SAVE THE CURRENT               !save the problem in a file
PROBLEM STATEMENT IN A FILE? (NO): Y          !for later use.
                                              !the option to save the
*** ENTER NAME FOR THE SAVED FILE ***         !results in a file operates
                                              !very similar to this.
ENTER A FILE NAME (15 CHAR. OR LESS)
OR HIT <RETURN> TO EXIT. SERVO.PROB

YOU ENTERED FILE NAME: SERVO.PROB             !double check on name.
IS THAT CORRECT? (YES):

CHECKING DIRECTORY STATUS: PROBLEM*LIB        !library file created.

*** DIRECTORY FILE CREATED: PROBLEM*LIB


HIT <RETURN> TO RETURN TO MENU.               !hit return and ignore this.
```

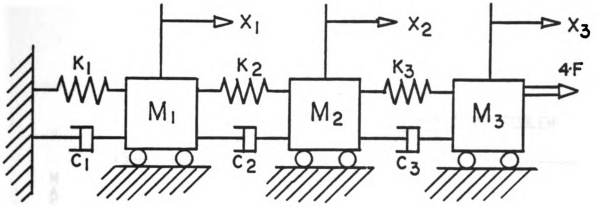
```
*** FILE STORED IN LIBRARY ***          !file storage confirmed.

HIT <RETURN> TO CONTINUE.

WANT TO LEAVE THIS MODULE? (YES):       !leave or return to main.

    SEE YOU LATER...                    !leaving Module F program.
```

5.2 THREE MASS OSCILLATOR

A system of three translating masses interconnected by springs and dampers is shown in Figure 7. This is a classic vibrations problem found in many texts on the subject[7]. Frequency processing of this system involves both Modules C and F.

This system is initially described in standard vibrations matrix form as shown in Figure 7. These matrices are entered interactively through Module C which generates an equivalent state space representation. The displacements of the masses are automatically selected as the first three state variables. The state space problem formulation is then saved in a file before leaving the module.

The frequency processing and problem file loading options are selected in Module F. The state space data stored previously in Module C is then loaded. The state variable to be used as output, displacement of the first mass, is selected and the transfer function is generated. The frequency response is calculated and the results displayed. Figure 8. shows the frequency response of the system with the position of the first mass as the output. The state space data file can be reloaded and a different state choosen as the transfer function output.



$$\begin{bmatrix} M_1 & 0 & 0 \\ 0 & M_2 & 0 \\ 0 & 0 & M_3 \end{bmatrix} \cdot \begin{bmatrix} \ddot{X}_1 \\ \ddot{X}_2 \\ \ddot{X}_3 \end{bmatrix} + \begin{bmatrix} C_1+C_2 & -C_2 & 0 \\ -C_2 & C_2+C_3 & -C_3 \\ 0 & -C_3 & C_3 \end{bmatrix} \cdot \begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \end{bmatrix}$$

$$+ \begin{bmatrix} K_1+K_2 & -K_2 & 0 \\ -K_2 & K_2+K_3 & -K_3 \\ 0 & -K_3 & K_3 \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} F \end{bmatrix}$$

PARAMETERS:

$M_1=2$, $M_2=M_3=1$, $K_1=K_2=K_3=4$,

$C_1=C_2=C_3=0.1$

Figure 7. Three Mass Oscillator

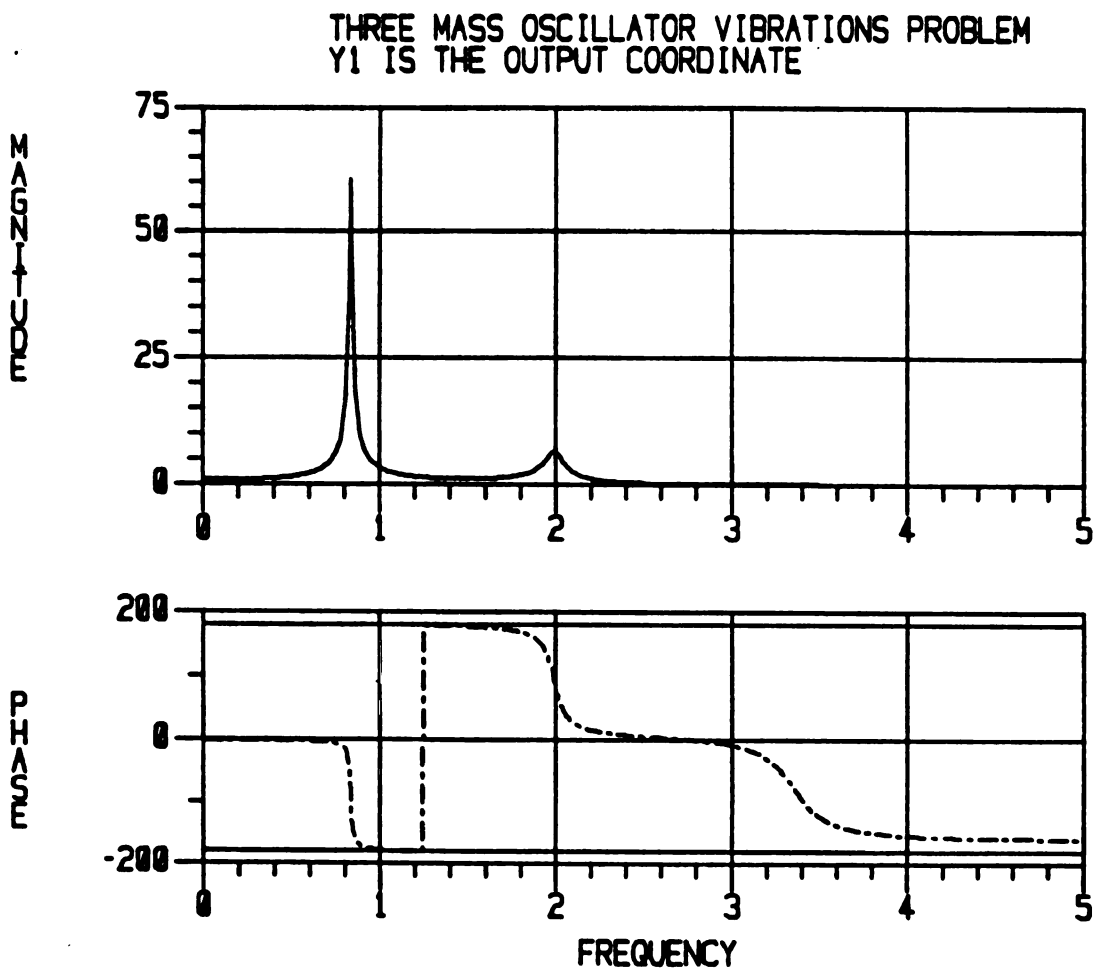


Figure 8. Frequency Response for Example 2

INTERACTIVE SESSION FOR EXAMPLE 2

```
OK, RUN77 SYSKIT>#MODULC                          !begin program execution.

 MODULE C IS AT YOUR SERVICE FOR
 CONVERTING DIFFERENTIAL EQUATIONS
 TO STATE EQUATION FORM...

 DO YOU WANT MORE DETAILS? (NO):                  !answer YES if on-line
                                                  !help is desired.

 DO YOU WANT TO WORK WITH                           !select problem mode.
  M,C,K MATRICES? (YES): N                          !prompt toggles until
  A SINGLE HIGHER-ORDER DIFF. EQU.? (YES): N        !a mode is selected.
  M,C,K MATRICES? (YES): Y


 PROCEED ? (YES):                                 !first branch point.

 ENTER A NEW PROBLEM INTERACTIVELY? (YES): N          !select input method
.
 RELOAD A PREVIOUSLY SAVED PROBLEM FILE? (YES): N     !only two choices.
 ENTER A NEW PROBLEM INTERACTIVELY? (YES):            !another toggle.


 THE PROBLEM TITLE IS:                            !default problem title.

 *** MODULE C PROBLEM ***

 CHANGE THE TITLE? (NO): Y                        !option to change title.


 ENTER NEW TITLE ON ONE LINE:                     !enter the new title.

THREE MASS OSCILLATOR VBRATIONS PROBLEM


 ENTER THE DIMENSION OF Z (2): 3                  !the number of coordinates.
                                                  !the default is 2.

 OPTIONS FOR THE M MATRIX                         !menu for operations which
 ------------------------                         !can be performed on the
  A: CHANGE ALL ENTRIES                           !mass matrix.
  R: CHANGE A ROW                                 !this same menu is used for
  C: CHANGE A COLUMN                              !every matrix input.
  E: CHANGE AN ENTRY                              !it wont be repeated here.
  Z: ZERO THE MATRIX
  L: LOOK AT THE MATRIX
  T: TO MODULE OPTIONS
```

```
 P: PROCEED  (=DEFAULT)
---------------------------

ENTER OPTION (P): L                          !display default mass matrix
                                             !on the screen.
THE M MATRIX IS:

         C1           C2           C3

R1:  0.2000E+01   0.4000E+01   0.0000E+00
R2:  0.4000E+01   0.9000E+01   0.0000E+00
R3:  0.0000E+00   0.0000E+00   0.1000E+01


PRESS <RET> KEY TO CONTINUE...               !return to matrix menu when
                                             !finished viewing matrix.

OPTIONS FOR THE M MATRIX
---------------------------                  !matrix menu not shown for
---------------------------                  !sake of brevity only.

ENTER OPTION (P): A                          !elect to change all of
                                             !the matrix elements.
                                             !enter a new matrix.
  M(1,1)? ( 2.000E+00): 2
  M(1,2)? ( 4.000E+00): 0                    !the new matrix is input
  M(1,3)? ( 0.000E-01): 0                    !element by element.


  M(2,1)? ( 4.000E+00): 0
  M(2,2)? ( 9.000E+00): 1
  M(2,3)? ( 0.000E-01): 0


  M(3,1)? ( 0.000E-01): 0
  M(3,2)? ( 0.000E-01): 0
  M(3,3)? ( 1.000E+00): 1

OPTIONS FOR THE M MATRIX                     !return to matrix menu
---------------------------                  !after mass matrix input.
---------------------------

ENTER OPTION (P): L                          !display new mass matrix.

THE M MATRIX IS:                             !display MASS matrix.

         C1           C2           C3

R1:  0.2000E+01   0.0000E+00   0.0000E+00
R2:  0.0000E+00   0.1000E+01   0.0000E+00
R3:  0.0000E+00   0.0000E+00   0.1000E+01
```

```
 PRESS <RET> KEY TO CONTINUE...                !return to matrix menu when
                                               !viewing is completed.
```

The damping, stiffness, and input matrices are entered in the same manner as the mass matrix. The same menu and procedure are used. The details of the interactive input of the other three matrices are omitted to avoid unnecessary repetition. The matrices are displayed as they were entered.

```
 THE C MATRIX IS:                              !display DAMPING matrix.

          C1           C2           C3

 R1:  0.2000E+00 -0.1000E+00  0.0000E+00
 R2: -0.1000E+00  0.2000E+00 -0.1000E+00
 R3:  0.0000E+00 -0.1000E+00  0.1000E+00

 THE K MATRIX IS:                              !display STIFFNESS matrix.

          C1           C2           C3

 R1:  0.8000E+01 -0.4000E+01  0.0000E+00
 R2: -0.4000E+01  0.8000E+01 -0.4000E+01
 R3:  0.0000E+00 -0.4000E+01  0.4000E+01

 ENTER THE DIMENSION OF F (2): 1               !enter number of inputs.

 THE L MATRIX IS:                              !display INPUT matrix.

          C1

 R1:  0.1000E+01
 R2:  0.0000E+00
 R3:  0.0000E+00
```

The equivalent state space representation is automatically generated. The first three states are the mass displacements and the other three states are the mass velocities. The state space data can now be viewed and/or saved in a problem file for later use by another module. The

original vibrations matrix form of the problem can also be filed for later use or modification by Module C.

```
 THE A MATRIX IS:                                    !state feedback matrix.

          C1            C2            C3

 R1:  0.0000E+00   0.0000E+00   0.0000E+00
 R2:  0.0000E+00   0.0000E+00   0.0000E+00
 R3:  0.0000E+00   0.0000E+00   0.0000E+00
 R4: -0.4000E+01   0.2000E+01   0.0000E+00
 R5:  0.4000E+01  -0.8000E+01   0.4000E+01
 R6:  0.0000E+00   0.4000E+01  -0.4000E+01

          C4            C5            C6

 R1:  0.1000E+01   0.0000E+00   0.0000E+00
 R2:  0.0000E+00   0.1000E+01   0.0000E+00
 R3:  0.0000E+00   0.0000E+00   0.1000E+01
 R4: -0.1000E+00   0.5000E-01   0.0000E+00
 R5:  0.1000E+00  -0.2000E+00   0.1000E+00
 R6:  0.0000E+00   0.1000E+00  -0.1000E+00


 THE B MATRIX IS:                                    !state input matrix.

          C1

 R1:  0.0000E+00
 R2:  0.0000E+00
 R3:  0.0000E+00
 R4:  0.0000E+00
 R5:  0.0000E+00
 R6:  0.4000E+01


 THE EIGENVALUES:

 VALUE     REAL          IMAGINARY
   1   -1.41144E-01  -3.35732E+00
   2   -1.41144E-01   3.35732E+00
   3   -4.99999E-02  -1.99938E+00
   4   -4.99999E-02   1.99938E+00
   5   -8.85623E-03  -8.41676E-01
   6   -8.85623E-03   8.41676E-01


 WANT TO PRINT THE A,B,C RESULTS AND          !print the state space
 EIGENVALUES ON THE PRINTER? (NO): Y          !formulation directly to
                                              !the line printer.
```

```
READY THE PRINTER
 AND HIT <RETURN>...                          !output sent after return.

DO YOU WISH TO SAVE THE A,B,C MATRICES        !store the state space data
AND EIGENVALUES IN A FILE? (NO): Y             !in a file for use with
                                               !Module F.
*** ENTER NAME FOR THE SAVED FILE ***

ENTER A FILE NAME (15 CHAR. OR LESS)          !data file name input.
OR HIT <RETURN> TO EXIT. OSCILL.ABC

YOU ENTERED FILE NAME: OSCILL.ABC             !check on file name input.
IS THAT CORRECT? (YES):

*** FILE STORED IN LIBRARY ***                !file storage confirmed.


HIT <RETURN> TO CONTINUE.


DO YOU WANT TO LIST                           !dump vibrations matrices
 THE CURRENT PROBLEM STATEMENT                !directly to line printer.
  ON THE PRINTER? (NO): Y


READY THE PRINTER
 AND HIT <RETURN>...

DO YOU WISH TO SAVE THE CURRENT               !option to save problem.
PROBLEM STATEMENT IN A FILE? (NO):


WANT TO LEAVE THIS MODULE? (YES):             !leave the program.


    SEE YOU LATER...
*** EXITED MODULE C ***
```

Module F is now used to generate a transfer function from the stored state space data and perform frequency processing. Most of the details of the operation of Module F are omitted here. Only those which are especially pertinent to this problem are included. Any gaps can be filled in by referring to the previous example. Module F execution begins at the problem entry selection.

```
 INITIAL PROBLEM ENTRY OPTIONS                !select method of entering
```

```
----------------------------------------   !the problem description.
R: RELOAD A PREVIOUSLY SAVED PROBLEM FILE
L: LOAD AN ENPORT GENERATED ABDATA FILE
E: ENTER A NEW PROBLEM INTERACTIVELY (=DEF)
----------------------------------------
ENTER OPTION (E): R                        !reload a state space file
                                           !saved in Module C.
*** ENTER NAME OF FILE TO BE LOADED ***

ENTER A FILE NAME (15 CHAR. OR LESS)       !enter name of data file.
OR HIT <RETURN> TO EXIT. OSCILL.ABC

YOU ENTERED FILE NAME: OSCILL.ABC          !confirm the file name.
IS THAT CORRECT? (YES):


CALCULATING THE TRANSFER FUNCTION          !transfer function creation
 FROM STATE VARIABLE DATA...               !taking place.

WHICH Y TO BE USED? (1): 1                 !row of output matrix to
                                           !be used for transfer funct.

DATA CONVERTED TO TRANSFER FUNCTION.       !successful conversion of
                                           !state space data confirmed.
*** USER FILE HAS BEEN LOADED ***


HIT <RETURN> TO CONTINUE.                  !return to continue.


PROCEED ? (YES):                           !another branch point.


THE PROBLEM TITLE IS:                      !display the new default
                                           !title entered through the
THREE MASS OSCILLATOR VIBRATIONS PROBLEM   !loaded data file.

CHANGE THE TITLE? (NO):                    !option to change title.


PROCEED? (YES):                            !another branch point.

NUMERATOR ENTRY OPTIONS                    !need to go through here
------------------------------             !in order to view the new
P: SINGLE POLYNOMIAL (=DEF)                !transfer function.
F: FACTORED FORM
R: ROOTS OF THE POLYNOMIAL
H: HELP
------------------------------
ENTER OPTION (P):                          !select polynomial input.
                                           !choose defaults for all
                                           !prompts so that problem
ORDER OF NUMERATOR? ( 2):                  !is not altered.
```

```
THE NUMERATOR POLYNOMIAL:                    !display the numerator.

  2.000E-02 * S**2
  1.600E+00 * S**1
  3.200E+01 * S**0


THE NUMERATOR ROOTS (ZEROS):

 REAL PART       IMAG PART
-4.000E+01      -1.105E-02
-4.000E+01       1.105E-02
```

View the denominator polynomial using the same procedure as for the numerator. Choose defaults to retain transfer function generated from the state space data.

```
ORDER OF DENOMINATOR? ( 6):                  !again choose all defaults
                                             !so that problem is only
                                             !viewed not altered.
THE DENOMINATOR POLYNOMIAL:

  1.000E+00 * S**6
  4.000E-01 * S**5
  1.603E+01 * S**4
  2.800E+00 * S**3
  5.606E+01 * S**2
  2.400E+00 * S**1
  3.200E+01 * S**0


THE DENOMINATOR ROOTS (POLES):

 REAL PART       IMAG PART
-1.411E-01      -3.357E+00
-1.411E-01       3.357E+00
-5.000E-02      -1.999E+00
-5.000E-02       1.999E+00
-8.856E-03      -8.417E-01
-8.856E-03       8.417E-01
```

The frequency response of this system can now be calculated and displayed as in the first example. A plot of the frequency response is shown in Figure 8. The state space data can be re-entered and a new transfer function generated using a different state as output. This can

be accomplished through the user file management system. The file management menu can be entered from the main menu at any branch point.

```
DO YOU WANT TO WORK WITH:                    !start of FILE MANAGEMENT.
 PROBLEM FILES? (YES): N                     !select the type of files
 RESULTS FILES? (YES): N                     !to be handled.
 PROBLEM FILES? (YES):                       !prompt toggles till YES.

PROBLEM FILER OPTIONS                        !user file management menu.
----------------------------------------
S: SAVE THE PROBLEM IN A FILE
U: UNSAVE (LOAD) A PROBLEM FILE
C: CHECK DIRECTORY AGAINST EXISTING FILES
L: LIST THE DIRECTORY CONTENTS
D: DELETE A PROBLEM FILE
A: ADD A FILE TO THE DIRECTORY
R: RETURN TO THE MAIN MENU (=DEFAULT)
----------------------------------------
ENTER OPTION (R): L                          !list the available files.
                                             !shown here only to
FILES FOR RELOADING MODULE T OR F:           !demonstrate option.
----------------------------------

 SERVO.PROB
ELECTRO-HYDRAULIC SERVOMECHANISM

 OSCILL.ABC
THREE MASS OSCILLATOR VIBRATIONS PROBLEM


HIT <RETURN> TO RETURN TO MENU.

PROBLEM FILER OPTIONS
----------------------------------------   !menu withheld for brevity.
----------------------------------------
ENTER OPTION (R): U                          !load the state data file.

*** ENTER NAME OF FILE TO BE LOADED ***     !input file name.

ENTER A FILE NAME (15 CHAR. OR LESS)
OR HIT <RETURN> TO EXIT. OSCILL.ABC

YOU ENTERED FILE NAME: OSCILL.ABC            !confirm file name entered.
IS THAT CORRECT? (YES):


CALCULATING THE TRANSFER FUNCTION            !generate transfer function.
 FROM STATE VARIABLE DATA...

WHICH Y TO BE USED? (1): 2                   !select output variable.
```

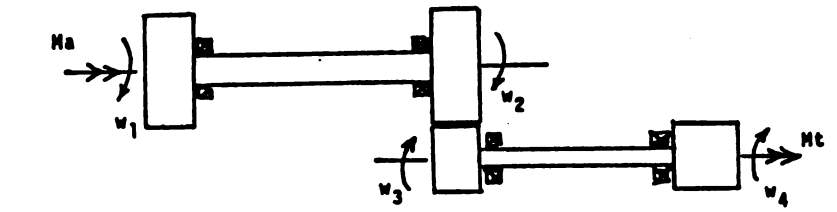
```
DATA CONVERTED TO TRANSFER FUNCTION.        !data conversion and file
                                            !loading confirmed.
*** USER FILE HAS BEEN LOADED ***

HIT <RETURN> TO CONTINUE.

PROBLEM FILER OPTIONS
-------------------------------------
-------------------------------------
ENTER OPTION (R):                           !return to main menu.
```

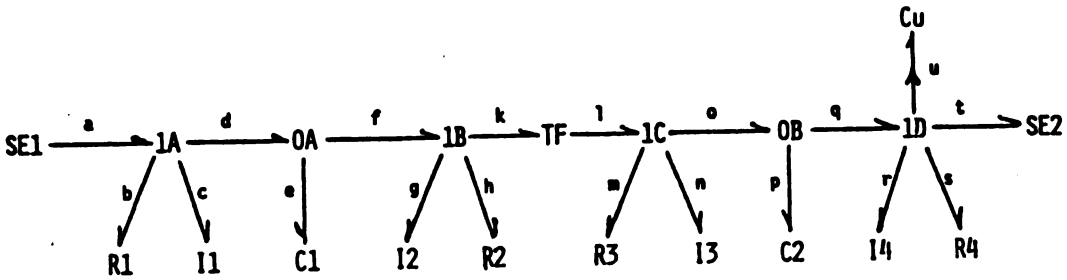
Once in the main menu the problem can proceed as in the first case with Y1 as output. This process can then be repeated for the third coordinate, Y3, as output.

5.3 PARALLEL GEARED DRIVE SHAFTS

The system shown in Figure 9 consists of two, geared, parallel shafts on bearing supports. An input load torque is applied to the left end with an output load torque at the right. The corresponding bond graph is also shown in Figure 9 along with the parameters and state variable definitions. This example is taken from reference 3 and the details of the bond graph processing using ENPORT can be found there. The ABDATA file containing the state space representation generated by ENPORT is shown in Figure 10. The package interfaces with ENPORT through this file.

Since Module F is covered thoroughly in the first two examples, most of the details of its operation will be omitted here. The option of loading an ABDATA file is selected as the method of problem entry. The angular displacement of the end mass, the sixth state variable, is selected as the output. The left hand torque input is choosen to be the input. The transfer function is generated between the two and is used to compute the frequency response shown in Figure 11.






```
THERE ARE   6 STATE VARIABLES
      AND   2 INPUT VARIABLES.

THE STATE VECTOR...

 X(  1) = P(c        )          ... momentum of I1
 X(  2) = Q(e        )          ... twist of C1
 X(  3) = P(g        )          ... momentum of I2
 X(  4) = Q(p        )          ... twist of C2
 X(  5) = P(r        )          ... momentum of I4
 X(  6) = Q(u        )          ... position of I4


THE INPUT VECTOR...

 U(  1) = E(a        )          ... driving torque
 U(  2) = E(t        )          ... load torque

The parameters are:

 C1:  K = 3200.  N/m          I3:  J =   20.  kg-m**2
 I1:  J =  100.  kg-m**2      C2:  K = 1200.  N/m
 I2:  J =  100.  kg-m**2      I4:  J =   40.  kg-m**2
 R1:  B =   25.  Ns           R3:  B =   25.  Ns
 R2:  B =   25.  Ns           R4:  B =   25.  Ns
                              TF:  m =    4.

                    The inputs are now:

                      SE1: driving torque
                      SE2: load torque
```

Figure 9. Parallel Geared Drive Shafts

```
File: SHAFTS.AB

PARALLEL GEARED DRIVE SHAFTS

The number of state variables:   6, the number of inputs:   2

 The A matrix

  1   -2.50000E-01 -3.20000E+03  0.00000E-01  0.00000E-01  0.00000E-01
       0.00000E-01

  2    1.00000E-02  0.00000E-01 -1.00000E-02  0.00000E-01  0.00000E-01
       0.00000E-01

  3    0.00000E-01  7.61905E+02 -1.01190E+00 -1.14286E+03  0.00000E-01
       0.00000E-01

  4    0.00000E-01  0.00000E-01  4.00000E-02  0.00000E-01 -2.50000E-02
       0.00000E-01

  5    0.00000E-01  0.00000E-01  0.00000E-01  1.20000E+03 -6.25000E-01
       0.00000E-01

  6    0.00000E-01  0.00000E-01  0.00000E-01  0.00000E-01  2.50000E-02
       0.00000E-01

 The B matrix

  1    1.00000E+00  0.00000E-01

  2    0.00000E-01  0.00000E-01

  3    0.00000E-01  0.00000E-01

  4    0.00000E-01  0.00000E-01

  5    0.00000E-01 -1.00000E+00

  6    0.00000E-01  0.00000E-01

The eigenvalues:

Value     Real        Imaginary
  1    0.00000E-01   0.00000E+00
  2   -4.24755E-01   9.13182E+00
  3   -4.24755E-01  -9.13182E+00
  4   -1.52011E-01   5.62825E+00
  5   -1.52011E-01  -5.62825E+00
  6   -7.33373E-01   0.00000E-01
```

Figure 10. ABDATA File for Example 3

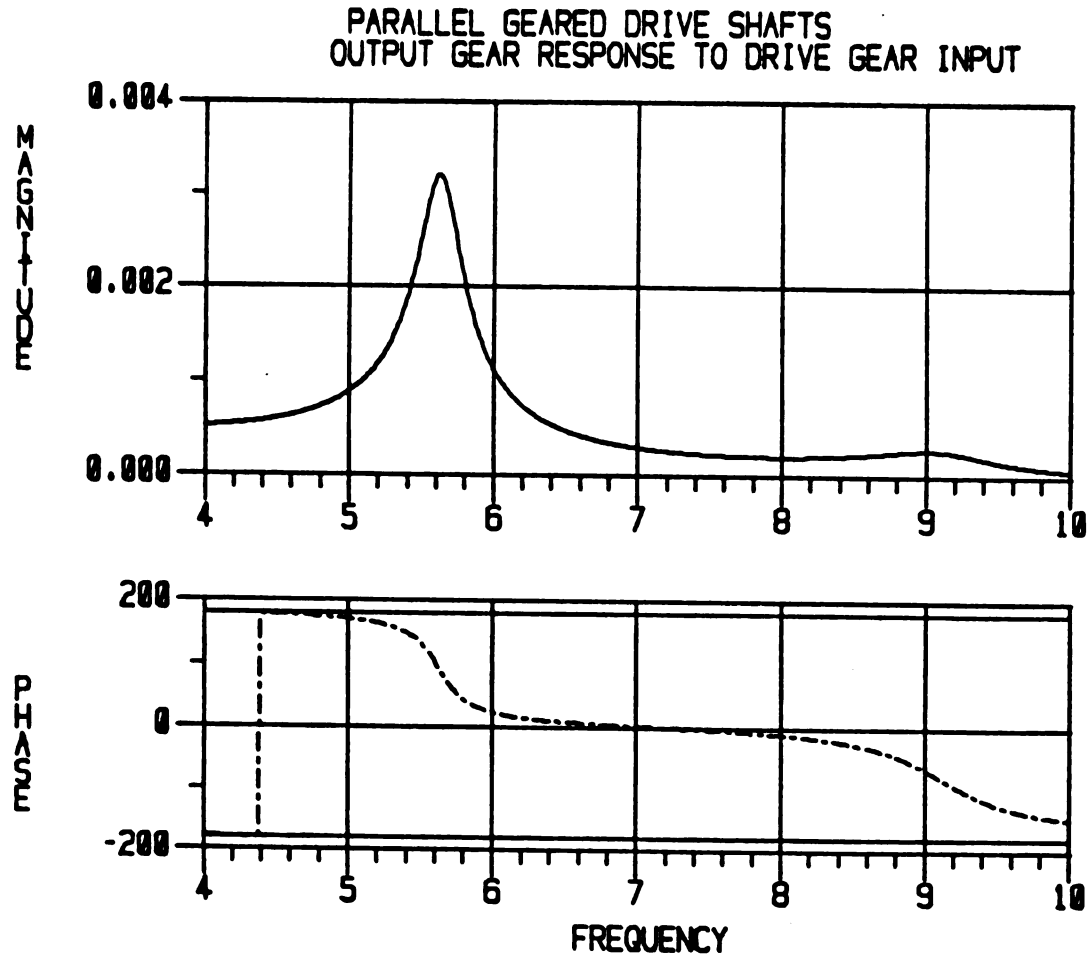


Figure 11. Frequency Response for Example 3

INTERACTIVE SESSION FOR EXAMPLE 3

Most of the details of the operation of Module F are omitted in this example. This session example begins at the selection of the problem input method. The option for loading a previously created ABDATA file is choosen. A transfer function is generated using the desired input and output.

```
INITIAL PROBLEM ENTRY OPTIONS                !problem entry selection.
----------------------------------------
R: RELOAD A PREVIOUSLY SAVED PROBLEM FILE
L: LOAD AN ENPORT GENERATED ABDATA FILE
E: ENTER A NEW PROBLEM INTERACTIVELY (=DEF)
----------------------------------------
ENTER OPTION (E): L                          !load an ABDATA file.

*** ENTER NAME OF THE ABDATA FILE ***

ENTER A FILE NAME (15 CHAR. OR LESS)         !name of ABDATA file to
OR HIT <RETURN> TO EXIT. SHAFTS.AB           !to be loaded.

YOU ENTERED FILE NAME: SHAFTS.AB             !confirm selection/spelling.
IS THAT CORRECT? (YES):

WHICH U TO BE USED AS INPUT? (1): 1          !select the input.

WHICH X TO BE USED AS OUTPUT? (1): 6         !select the output.


DATA CONVERTED TO TRANSFER FUNCTION.         !confirmation of conversion.


THE PROBLEM TITLE IS:                        !default title read from the
                                             !ABDATA file just loaded.
 PARALLEL GEARED DRIVE SHAFTS

CHANGE THE TITLE? (NO):
```

Display the numerator and denominator polynomial for the transfer function along with poles and zeroes. Use the polynomial entry blocks as described in the previous examples.

```
ORDER OF NUMERATOR? ( 0):

THE NUMERATOR POLYNOMIAL:

  9.143E+00 * S**0

ORDER OF DENOMINATOR? ( 6):

THE DENOMINATOR POLYNOMIAL:

  1.000E+00 * S**6
  1.887E+00 * S**5
  1.164E+02 * S**4
  1.371E+02 * S**3
  2.688E+03 * S**2
  1.943E+03 * S**1
  0.000E+00 * S**0

THE DENOMINATOR ROOTS (POLES):

 REAL PART      IMAG PART
-4.248E-01     -9.132E+00
-4.248E-01      9.132E+00
-1.520E-01     -5.628E+00
-1.520E-01      5.628E+00
-7.334E-01      0.000E+00
 0.000E+00      0.000E+00
```

The frequency response can now be computed and displayed as desired. The response for this system is shown in Figure 11. The state space data could be re-entered and a transfer function generated for a different output if desired.

Chapter 6

SUMMARY AND RECOMMENDATIONS

This software package allows frequency response processing for dynamic systems modelled by a variety of common techniques. The package consists of three independent, stand-alone software modules which communicate through a shared data base. Each of the modules operates using question-and-answer and menu driven interactive dialogue. None of the numerical algorithms used is new or experimental. Rather, only tried and proven routines were used. Of primary concern were software design and the integration of the individual modules into the package. Particular attention was paid to the qualities of ease-of-use, robustness, and flexibility, as well as future integration of other modules. All of the initial objectives of this work have been accomplished and the result is a powerful and dependable tool for dynamic system analysis and design. The package should prove useful not only to the student, but to the practicing engineer as well.

Future development should be initially directed to the re-design and integration of Module T. This module allows direct interactive input of state equations, converts transfer functions to state space form, and performs time response processing. The package was designed with the future integration of Module T as an important consideration. The basic building blocks for Module T are available and should be fairly easily assembled using the structure of Module F as a guide.

The next step would be to combine all four modules into a unified dynamic system modelling package. A common user file management system could be used which would handle differential equation, transfer function, and bond graph problem files, all state space data, and results for both frequency and time processing. ABDATA files would no longer be required. Certain redundancies such as having separate integration routines for both Module T and ENPORT could be eliminated if desired. The stand-alone modular concept could also be retained. Additional features such as block diagram algebra or the root locus for any system parameter would prove valuable.

LIST OF REFERENCES

1. Rosenberg, Ronald C., Karnopp, Dean C., 'Introduction to Physical System Dynamics', McGraw Hill, 1983

2. SYSKIT - Software Toolkit for Linear Systems - Operators Manual, Rosencode Associates, Inc., McGraw Hill

3. The ENPORT-5 Users Manual, Rosencode Associates, 1984

4. Ogata, Katsuhiko, 'Modern Control Engineering', Prentice-Hall, 1970

5. R.W. Hamming, 'Numerical Methods for Scientists and Engineers', McGraw Hill, 1962, pp. 356 - 359

6. Bollinger, John G., Harrison, Howard L., 'Introduction To Automatic Controls', International Textbook Co., 1969

7. Meirovitch, Leonard, 'Analytical Methods In Vibrations', MacMillan Co., 1967

8. Younkin, George, 'The Why of Industrial Servo Drives', Giddings & Lewis Machine Tool Co., Fon Du Lac, WI.

Appendix A

FILE ORGANIZATION AND CALLING TREES

FILES AND SUBROUTINES FOR MODULE F

MODULF B1
****
MAIN CALLING PROGRAM FOR MODULE F
CONTAINS NO SUBROUTINES

FBLOK1	B4	FBLOK2	B22	FBLOK3	B35
********		********		********	
FACTOR	B17	ENTERF	B22	FDISP	B37
GEGAIN	B12	FLEAVE	B34	FRESP	B42
GETDEN	B11	FPROB	B23	FSCALE	B35
GETERM	B5	FRSLT	B29	PHMARG	B44
GETNUM	B9	GETABD	B31	TL2GET	B46
INHELP	B14	PRINPY	B27		
INITLF	B6	SAVPRF	B26		
INPOLY	B14				
INROOT	B15				
POLY	B19				
SEMBL	B20				
SETUPF	B4				
TOPMDF	B8				
TTLGET	B6				
WHEREF	B9				
ZEROS	B20				

FBLOK4	B47	FBLOK5	B59	FBLOK6	B70
********		********		********	
LOCMAX	B57	CHREQ	B66	BPLOT	B70
LOGRES	B56	CHREQA	B63	DATAPR	B76
LOGSCL	B55	CONMAT	B62	NPLOT	B73
RDISP	B49	DET	B64	RPLOT	B74
RLOCUS	B47	GETTFN	B59		
		MPY	B65		
		PROOT	B67		

FILES AND SUBROUTINES FOR MODULE C

MODULC C1
****
MAIN CALLING PROGRAM FOR MODULE C.
CONTAINS NO SUBROUTINES.

CBLOK1	C4	CBLOK2	C13
********		********	
CLEAVE	C12	INNCK	C13
ENTERC	C8	INPUTL	C17
INITLC	C5	SGLINP	C19
SAVPRC	C11		
SETUPC	C4		
SHOWAB	C9		
TOPMDC	C6		
TTLGET	C9		
WHEREC	C7		
CBLOK3	C25	CBLOK4	C39
********		********	
CPROB	C25	CHREQA	C42
CRSLT	C28	DET	C43
DATAPR	C33	MLTYCK	C41
ELGET	C36	MMULT	C39
INPUTM	C34	PROOT	C44
OUTMAT	C36	SIMEQ	C40
PRINAB	C32	SORT	C46
PRINYM	C30		
PRMTX	C32		

UTILITY FILES
COMMON TO BOTH MODULES

IOUTIL	D1	UNIFIL	D13
********		********	
CHECKY	D9	ADAFIL	D21
CHRPCK	D9	CHKLIB	D19
FNAME	D1	DELFIL	D17
GETCHR	D7	LODFIL	D27
GETINT	D3	LSTLIB	D15
GETLIN	D8	SAVFIL	D24
GETREL	D4	UNIDRV	D13
MENU	D10		
MYCHAR	D11		
PROCED	D11		
YES	D10		

COMMON BLOCKS

**********	
CTRLBK	E1
EIGNBK	E8
FACTBK	E4
FILRBK	E10
FOUTBK	E5
INTGBK	E9
LABLBK	E1
PARMBK	E7
POLYBK	E2
RANGBK	E2
RESULT	E3
ROOTBK	E3
ROUTBK	E6
RSLTBK	E5
SGLIBK	E7
TERMBK	E9
VIBRBK	E6

CALLING TREE FOR MODULE F

```
MODULF
      SETUPF
      GETERM
      INITLF
      TOPMDF
      WHEREF
      ENTERF
            LODFIL
                  CHKLIB
                  FPROB
                        GETTFN
                              CONMAT
                                    CHREQA
                                          DET(FCN)
                                    CHREQ
                                    MPY
                                    PROOT
            GETABD
                  CONMAT
                        CHREQA
                              DET(FCN)
                        CHREQ
                        MPY
                        PROOT
      TTLGET
      GETNUM
            INPOLY
                  POLY
                  PROOT
                  ZEROS
            FACTOR
                  POLY
                  PROOT
                  ZEROS
                  SEMBL
            INROOT
                  ZEROS
                  SEMBL
                  POLY
            INHELP
      GETDEN
            INPOLY
```

```
            POLY
            PROOT
            ZEROS
      FACTOR
            POLY
            PROOT
            ZEROS
            SEMBL
      INROOT
            ZEROS
            SEMBL
            POLY
      INHELP
GEGAIN
FSCALE
FRESP
PHMARG
RLOCUS
      LOGSCL
            LOGRES
      PROOT
FDISP
      TL2GET
      DATAPR
      SAVFIL
            CHKLIB
            FRSLT
      BPLOT
      MPLOT
RDISP
      TL2GET
      LOCMAX
      DATAPR
      SAVFIL
            CHKLIB
            FRSLT
      RPLOT
UNIDRV
      SAVFIL
            CHKLIB
            FPROB
            FRSLT
      LODFIL
            CHKLIB
            FPROB
                  GETTFN
                        CONMAT
                              CHREQA
                                    DET(FCN)
                              CHREQ
                              MPY
                              PROOT
            FRSLT
      CHKLIB
```

```
      LSTLIB
            CHKLIB
      DELFIL
      ADAFIL
            CHKLIB
GETABD
      CONMAT
            CHREQA
                  DET(FCN)
            CHREQ
            MPY
            PROOT
SAVPRF
      PRINPY
            DATAPR
      SAVFIL
            CHKLIB
            FPROB
FLEAVE
```

CALLING TREE FOR MODULE C

```
MODULC
      SETUPC
      INITLC
      TOPMDC
      WHEREC
      ENTERC
            LODFIL
                  CHKLIB
                  CPROB
      TTLGET
      INMCK
            INPUTM
                  ELGET
                  OUTMAT
            SIMEQ
            MMULT
            CHREQA
                  DET(FCN)
            PROOT
            SORT
      INPUTL
            INPUTM
                  ELGET
                  OUTMAT
            MMULT
      SGLINP
            MLTYCK
            CHREQA
                  DET(FCN)
            PROOT
            SORT
      SHOWAB
            OUTMAT
            PRINAB
                  DATAPR
                  PRMTX
            SAVFIL
                  CHKLIB
                  CRSLT
      UNIDRV
            SAVFIL
                  CHKLIB
```

```
            CPROB
            CRSLT
      LODFIL
            CHKLIB
            CPROB
      CHKLIB
      LSTLIB
            CHKLIB
      DELFIL
      ADAFIL
            CHKLIB
SAVPRC
      PRINYM
            DATAPR
            PRMTX
      SAVFIL
            CHKLIB
            CPROB
CLEAVE
```

COMMAND FILES FOR COMPILING AND LINKING MODULES

```
FTN77 MODULF -FULLCHECK
FTN77 FBLOK1 -FULLCHECK
FTN77 FBLOK2 -FULLCHECK
FTN77 FBLOK3 -FULLCHECK
FTN77 FBLOK4 -FULLCHECK
FTN77 FBLOK5 -FULLCHECK
FTN77 FBLOK6 -FULLCHECK
SEG
LOAD SYSKIT>*MODULF
LO MODULF.BIN
LO FBLOK1.BIN
LO FBLOK2.BIN
LO FBLOK3.BIN
LO FBLOK4.BIN
LO FBLOK5.BIN
LO FBLOK6.BIN
LO SYSKIT>COMMON.DIR>IOUTIL.BIN
LO SYSKIT>COMMON.DIR>UNIFIL.BIN
LI AGII
LI TEK
LI F77LIB
LI VSP00*
LI
MAP 6
QUIT
CO -TTY
```

```
FTN77 MODULC -FULLCHECK
FTN77 CBLOK1 -FULLCHECK
FTN77 CBLOK2 -FULLCHECK
FTN77 CBLOK3 -FULLCHECK
FTN77 CBLOK4 -FULLCHECK
SEG
LOAD SYSKIT>*MODULC
LO MODULC.BIN
LO CBLOK1.BIN
LO CBLOK2.BIN
LO CBLOK3.BIN
LO CBLOK4.BIN
LO SYSKIT>COMMON.DIR>IOUTIL.BIN
LO SYSKIT>COMMON.DIR>UNIFIL.BIN
LI F77LIB
LI VSP00*
LI
MAP 6
QUIT
CO -TTY
```

Appendix B

MODULE F SOURCE CODE LISTINGS

```
C
C
C****************************** MODULF ********************************
C**********************************************************************
C
C
C  MAIN DRIVING PROGRAM FOR MODULE F.
C
      PROGRAM MODULF
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RESULT.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RSLTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      CALL GETERM
C
      CALL INITLF
C
      CALL TOPMDF
      IF (SYSFLG) THEN
        CALL FLEAVE
        IF (MODFLG) THEN
          GOTO 10
        ELSE
          GOTO 999
        ENDIF
      ENDIF
      GOTO 10
C
1000  MODFLG=.FALSE.
        CALL WHEREF(JUMPER)
        GO TO(10,20,30,40,50,60,70,80,90,95,100,110,115)JUMPER
C
C--- INITIALIZE
C
10    CALL ENTERF
      IF(MODFLG)GO TO 1000
```

```
C
C--- GET PROBLEM TITLE
C
20    CALL TTLGET
      IF (MODFLG)GOTO 1000
C
C--- GET NUMERATOR
C
30    CALL GETNUM
      IF(MODFLG)GO TO 1000
C
C--- GET DENOMINATOR
C
40    CALL GETDEN
      IF(MODFLG)GO TO 1000
C
C--- GET GAIN
C
50    CALL GEGAIN
      IF(MODFLG)GO TO 1000
C
C--- SET FREQUENCY OR LOCUS GAIN SCALE
C
60    CALL FSCALE
      IF(MODFLG)GO TO 1000
C
C--- FREQUENCY OR LOCUS CALCULATIONS
C
70    IF (FRQYES) THEN
        CALL FRESP
        IF(MODFLG) GO TO 1000
        CALL PHMARG
      ELSE
        CALL RLOCUS
        IF(MODFLG) GO TO 1000
      ENDIF
C
C--- OUTPUT DISPLAY
C
80    IF (FRQYES) THEN
        CALL FDISP
      ELSE
        CALL RDISP
      ENDIF
      IF(MODFLG)GO TO 1000
      GOTO 115
C
C--- PROBLEM OR RESULTS FILE MANAGEMENT
C
90    CALL UNIDRV
      IF (MODFLG) GOTO 1000
C
C--- LOAD AN ENPORT ABDATA FILE
C
```

```
95     CALL GETABD
       IF (MODFLG) GOTO 1000
C
C--- LEAVE BLOCK
C
115    CALL SAVPRF
       CALL FLEAVE
       IF(MODFLG)GO TO 1000
       GOTO 999
C
C--- SAVE PROBLEM DESCRIPTION ON DISK OR PRINTER
C
100    CALL SAVPRF
       GOTO 1000
C
C--- CONTINUE TO NEXT BLOCK
C
110    IF (IBLK.GE.8)THEN
         GOTO 1000
       ELSE
         IBLK=IBLK+1
         GOTO(10,20,30,40,50,60,70,115)IBLK
       ENDIF
C
 999   CONTINUE
       END
C
C
C*************************************************************************
C*************************************************************************
C
C
```

```
C
C
C****************************** FBLOK1 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Sets up and initializes Module F. Sets the problem
C          title and parameters as well as on-line help. Also
C          does all of the interactive input of the transfer
C          function. Also has main menu.
C
C--- CONTENTS:
C          SETUPF   block data to set permanent parameters
C          GETERM   determine terminal type for graphics
C          TTLGET   enter or change the problem title
C          INITLF   initialize parameters and data base
C          TOPMDF   on-line help for Module F
C          WHEREF   main menu display and option prompts
C          GETNUM   interactive input of the T.F. numerator
C          GETDEN   interactive input of the T.F. denominator
C          GEGAIN   input the system gain
C          INHELP   on-line help for problem entry options
C          INPOLY   interactive input of polynomial coefficients
C          INROOT   interactive input of the polynomial roots
C          FACTOR   input of num. or den. in factored form
C          POLY     displays a polynomial
C          ZEROS    displays the roots of a polynomial
C          SEMBL    generates a polynomials coeff. from its roots
C
C--- INDEX:
C          FACTOR
C          GEGAIN
C          GETDEN
C          GETERM
C          GETNUM
C          INHELP
C          INITLF
C          INPOLY
C          INROOT
C          POLY
C          SEMBL
C          SETUPF
C          TOPMDF
C          TTLGET
C          WHEREF
C          ZEROS
C
C
C****************************** SETUPF ******************************
C************************************************************************
C
C
C  SETS PARAMETERS FOR MODULE F WHICH ARE SPECIFIC
```

```
C  TO IT AND WHICH ARE TO REMAIN UNCHANGED.
C  WRITTEN BY J. GREGORSKI JANUARY,1985
C
      BLOCK DATA SETUPF
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      DATA WANTYP(1),WANTYP(2),PRBTYP/1,4,4/
      DATA HEADER(1),HEADER(2)/1,3/
      DATA MODULE/'F'/
C
      END
C
C
C****************************** GETERM ********************************
C**********************************************************************
C
C
C  DETERMINES TYPE OF USER TERMINAL AND USES INFORMATION
C  TO SET BAUD RATE AND TO PREVENT USE OF GRAPHICS ON
C  UNCOMPATIBLE TERMINALS.
C  MODIFIED BY J. GREGORSKI OCTOBER,1984
C
      SUBROUTINE GETERM
C
      INCLUDE 'SYSKIT>COMMON.DIR>TERMBK.TEXT'
C
      INTEGER ITERM
      LOGICAL OK
C
      CALL TERM$$(NTERM,NTYPE,NRATE,NERROR)
      NRATE = NRATE/10
C
10    WRITE(1,100)
100   FORMAT(/,/,'  WHAT KIND OF TERMINAL ARE YOU USING?',
     1   /,/,'  A: 4006, 4010, 4014 OR OTHER STORAGE TUBE DEVICE',
     2   /, '  B: 4114',
     3   /, '  C: 4025 OR 4027',
     4   /, '  D: 4105, 4107 OR 4109',
     5   /, '  E: SOMETHING ELSE',
     6   /, '  ENTER CHOICE:(A=DEF) ',$)
      CALL MENU(OK,ITERM,'A','B','C','D','E','','',
     2 '
      IF (.NOT.OK) GOTO 10
      GOTO (200,200,300,200,300) ITERM
C
C--- THESE TERMINALS ABLE TO PLOT CORRECTLY.
C
200   PLOTIT = .TRUE.
      RETURN
C
C--- NOT ABLE TO PLOT CORRECTLY.
C
300   PLOTIT = .FALSE.
```

```
      RETURN
      END
C
C
C****************************** TTLGET ********************************
C**********************************************************************
C
C
C  ENTER OR CHANGE THE TITLE FOR THE PROBLEM.
C  MODIFIED BY J. GREGORSKI AUGUST,1984
C
      SUBROUTINE TTLGET
C
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
C
      LOGICAL YES
C
      IBLK=2
C
10    WRITE(1,1010)TITLE1
1010  FORMAT(//' THE PROBLEM TITLE IS:',//,1X,A40//,
     2         ' CHANGE THE TITLE? (NO): ',$)
      IF (YES(15,'N')) THEN
        WRITE(1,1020)
1020    FORMAT(//' ENTER NEW TITLE ON ONE LINE:'/)
        READ(1,1030)TITLE1
1030    FORMAT(A40)
        GOTO 10
      ENDIF
C
      WRITE(1,2000)
2000  FORMAT(//' PROCEED? (YES):',$)
      IF (YES(23,'Y')) RETURN
      MODFLG=.TRUE.
      RETURN
      END
C
C
C****************************** INITLF ********************************
C**********************************************************************
C
C
C  SET PARAMETERS FOR DEFAULT PROBLEM FORMULATION AS
C  WELL AS INPUT LIMITS,COMPUTATION RANGE,TITLES,ETC...
C  WRITTEN BY J.GREGORSKI AUGUST,1984
C
      SUBROUTINE INITLF
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FACTBK.TEXT'
```

```
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
C--- START OF INITIALIZATION
C
      NPTS=25
      INDEG=1
      IDDEG=3
      GAIN=1.
      NFNUM=1
      NFDEN=1
C
C--- INITIALIZE ALL PROBLEM FORMULATION MATRICES
C
      DO 10 J=1,MAXDEG
        COEFFN(J)=1.
        RNR(J)=-1.
        RNI(J)=0.
        COEFFD(J)=1.
        RDR(J)=-1.
        RDI(J)=0.
        LNDEG(J)=2.
        LDDEG(J)=2.
        DO 10 I=1,MAXCOF
          CFNF(I,J)=1.
          CFDF(I,J)=1.
10    CONTINUE
      COEFFN(MAXCOF)=1.
      COEFFD(MAXCOF)=1.
C
C--- DEFAULT PROBLEM FOR POLYNOMIAL (=DEF) INPUT
C
      COEFFN(1)=5.
      COEFFN(2)=1.
      COEFFD(1)=0.
      COEFFD(2)=5.
      COEFFD(3)=2.
      COEFFD(4)=1.
C
C--- INITIALIZE OTHER PARAMETERS ETC....
C
      BLO=-1.E+15
      BHI=1.E+15
      SCMIN=1.
      SCMAX=10.
      PRBFIL=.TRUE.
      FRQYES=.TRUE.
      SYSFLG=.FALSE.
      MODFLG=.FALSE.
      NFCTFG=.FALSE.
      DFCTFG=.FALSE.
      LIBNAM='PROBLEM*LIB'
      TITLE1='*** MODULE F PROBLEM ***'
      TITLE2='*** MODULE F RESULTS ***'
```

```
C
      RETURN
      END
C
C
C****************************** TOPMDF ******************************
C************************************************************************
C
C
C  GIVES THE USER A BRIEF OVERVIEW OF MODULE F
C  CAPABILITITES AND INSTRUCTIONS FOR USE IF
C  SO DESIRED. OPTION TO LEAVE MODULE AFTER THE
C  MESSAGE IS DISPLAYED.
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE TOPMDF
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      LOGICAL YES
C
      WRITE(1,1077)
1077  FORMAT(//' MODULE F IS AT YOUR SERVICE FOR'
     2        /' FREQUENCY RESPONSE AND ROOT LOCUS'
     3        /' CALCULATIONS...',//,
     4' DO YOU WANT MORE DETAILS? (NO): ',$)
      IF (YES(7,'N')) THEN
C
C--- INTRODUCTORY TEXT
C
      WRITE(1,111)
111   FORMAT(//5X,'IN MODULE F YOU MAY ENTER',
     2       /,' TRANSFER FUNCTIONS IN EITHER POLYNOMIAL,',
     3       /,' ROOTS OF THE POLYNOMIAL, OR FACTORED FORM.')
      WRITE(1,112)
112   FORMAT(/,5X,'T(S)=GAIN*(NUM(S)/DEN(S)).')
      WRITE(1,113) MAXDEG
113   FORMAT(/,5X,'GAIN IS A REAL NUMBER.',
     2       /,5X,'NUM(S) AND DEN(S) ARE OF MINIMUM',
     3       /,5X,'ORDER 0 AND MAXIMUM ORDER ',I2,'.')
      WRITE(1,114)
114   FORMAT(/,5X,'THE FREQUENCY RESPONSE OR ROOT',
     2       /,' LOCUS WILL BE CALCULATED FOR A',
     3       /,' LOG OR LINEAR SCALE.')
      WRITE(1,115)
115   FORMAT(/,5X,'YOU MAY DIRECT THE SOLUTION TO',
     2       /,' THE SCREEN, PRINTER OR DISK. THE ',
     3       /,' PROBLEM MAY BE SAVED FOR LATER USE',
     4       /,' BY THIS OR ANOTHER MODULE.',
     5       //,' CONTINUE? (YES): ',$)
        IF (YES(22,'Y')) THEN
          GOTO 1000
        ELSE
          SYSFLG=.TRUE.
        ENDIF
```

```
      ENDIF
1000  RETURN
      END
C
C
C****************************** WHEREF ******************************
C********************************************************************
C
C
C  DISPLAYS MAIN MENU AND PROMPTS USER FOR
C  OPTION DESIRED.CHECKS THAT IT IS A VALID OPTION.
C  SENDS OPTION SELECTED TO MAIN AS JUMPER.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE WHEREF(JUMPER)
C
C--- MAIN MODULE OPTIONS
C
      WRITE(1,111)
111   FORMAT(//'  MODULE F OPTIONS',/,
     2' ----------------------------------------',/,
     3'  1: SELECT MODE AND PROBLEM ENTRY',/,
     4'  2: ALTER THE PROBLEM TITLE',/,
     5'  3: CHANGE THE NUMERATOR',/,
     6'  4: ALTER THE DENOMINATOR',/,
     7'  5: CHANGE THE GAIN',/,
     8'  6: ALTER FREQUENCY OR LOCUS SCALE')
      WRITE(1,912)
912   FORMAT('  7: CALCULATE RESULTS',/,
     2'  8: DISPLAY OR SAVE THE RESULTS',/,
     7'  9: PROBLEM OR RESULTS FILE MANAGEMENT',/,
     6' 10: LOAD AN ENPORT ABDATA FILE',/,
     3' 11: SAVE PROBLEM STATEMENT AND RETURN',/,
     4' 12: CONTINUE FROM LAST ACTION (=DEF)',/,
     5' 13: LEAVE THIS MODULE',/,
     6' ----------------------------------------')
      WRITE(1,113)
113   FORMAT(' PLEASE ENTER OPTION (12): ',$)
      JUMPER=12
      CALL GETINT(JUMPER,1,13,25)
      RETURN
      END
C
C
C****************************** GETNUM ******************************
C********************************************************************
C
C
C  INPUTS THE NUMERATOR OF THE TRANSFER FUNCTION
C  IN EITHER POLYNOMIAL, ROOTS, OR FACTORED FORM.
C  RE-WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE GETNUM
C
```

```
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FACTBK.TEXT'
C
      INTEGER ICHECK
      LOGICAL YES,OK,CHNGFG
C
      IBLK=3
      CHNGFG=.FALSE.
5     WRITE(1,1000)
1000  FORMAT(/' NUMERATOR ENTRY OPTIONS',
     1 /' ----------------------------',
     2 /' P: SINGLE POLYNOMIAL (=DEF)',
     3 /' F: FACTORED FORM',
     4 /' R: ROOTS OF THE POLYNOMIAL',
     5 /' H: HELP',
     6 /' ----------------------------',
     7 /' ENTER OPTION (P): ',$)
C
      CALL MENU(OK,IX,'P','F','R','H','','','
     2 '
      IF (.NOT.OK) GOTO 5
      IF (IX.EQ.4) GOTO 40
C
C--- ENTER THE ORDER OF THE NUMERATOR. WARNING ABOUT CHANGING
C--- THE PROBLEM IN POLY. OR ROOT INPUT IF FACTORED DATA EXISTS.
C
      IF((NFCTFG).AND.((IX.EQ.1).OR.(IX.EQ.3))) THEN
        WRITE(1,1045)
1045    FORMAT(//,'            *** WARNING *** ',/
     +  ' THE EXISTING FACTORED DATA WILL NO LONGER BE VALID IF ',/
     +  ' THE NUMERATOR ORDER,POLYNOMIAL, OR ROOTS ARE CHANGED!')
      ENDIF
      ICHECK=INDEG
      WRITE(1,4)INDEG
4     FORMAT(//,' ORDER OF NUMERATOR? (',I2,'): ',$)
      CALL GETINT(INDEG,0,MAXDEG,14)
      IF (ICHECK.NE.INDEG) CHNGFG=.TRUE.
C
      GOTO (10,20,30) IX
C
C--- SINGLE HIGHER-ORDER POLYNOMIAL INPUT
C
10    CALL INPOLY('N',INDEG,COEFFN,RNR,RNI,CHNGFG)
      IF (CHNGFG) NFCTFG=.FALSE.
      GOTO 999
C
C--- FACTORED INPUT
C
20    CALL FACTOR('N',INDEG,RNR,RNI,COEFFN,NFNUM,LNDEG,CFNF)
      NFCTFG=.TRUE.
      GOTO 999
C
```

```
C--- POLYNOMIAL ROOTS INPUT
C
30    CALL INROOT('N',INDEG,RNR,RNI,COEFFN,CHNGFG)
      IF (CHNGFG) NFCTFG=.FALSE.
      GOTO 999
C
C--- READ HELP FILE AND RETURN TO MENU
C
40    CALL INHELP
      GOTO 5
C
999   CONTINUE
      WRITE(1,12)
12    FORMAT(//,' PROCEED ? (YES): ',$)
      IF(.NOT.YES(22,'Y'))MODFLG=.TRUE.
C
      RETURN
      END
C
C
C****************************** GETDEN ********************************
C**********************************************************************
C
C
C  INPUTS THE DENOMINATOR OF THE TRANSFER FUNCTION
C  IN EITHER POLYNOMIAL, ROOTS, OR FACTORED FORM.
C  RE-WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE GETDEN
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FACTBK.TEXT'
C
      INTEGER ICHECK
      LOGICAL YES,OK,CHNGFG
C
      IBLK=4
      CHNGFG=.FALSE.
5     WRITE(1,1000)
1000  FORMAT(/' DENOMINATOR ENTRY OPTIONS',
     1 /' ----------------------------',
     2 /' P: SINGLE POLYNOMIAL (=DEF)',
     3 /' F: FACTORED FORM',
     4 /' R: ROOTS OF THE POLYNOMIAL',
     5 /' H: HELP',
     6 /' ----------------------------',
     7 /' ENTER OPTION (P): ',$)
C
      CALL MENU(OK,IX,'P','F','R','H','','','
     2 '
      IF (.NOT.OK) GOTO 5
      IF (IX.EQ.4) GOTO 40
```

```
C
C--- ENTER THE ORDER OF THE DENOMINATOR. WARNING ABOUT CHANGING
C--- THE PROBLEM IN POLY. OR ROOT INPUT IF FACTORED DATA EXISTS.
C
      IF((DFCTFG).AND.((IX.EQ.1).OR.(IX.EQ.3))) THEN
        WRITE(1,1045)
1045    FORMAT(//,'            *** WARNING *** ',/
     +  ' THE EXISTING FACTORED DATA WILL NO LONGER BE VALID IF ',/
     +  ' THE DENOMINATOR ORDER,POLYNOMIAL, OR ROOTS ARE CHANGED!')
      ENDIF
      ICHECK=IDDEG
      WRITE(1,4)IDDEG
4     FORMAT(//,' ORDER OF DENOMINATOR? (',I2,'): ',$)
      CALL GETINT(IDDEG,0,MAXDEG,12)
      IF (ICHECK.NE.IDDEG) CHNGFG=.TRUE.
C
      GOTO (10,20,30) IX
C
C--- SINGLE HIGHER-ORDER POLYNOMIAL INPUT
C
10    CALL INPOLY('D',IDDEG,COEFFD,RDR,RDI,CHNGFG)
      IF (CHNGFG) DFCTFG=.FALSE.
      GOTO 999
C
C--- FACTORED INPUT
C
20    CALL FACTOR('D',IDDEG,RDR,RDI,COEFFD,NFDEN,LDDEG,CFDF)
      DFCTFG=.TRUE.
      GOTO 999
C
C--- POLYNOMIAL ROOTS INPUT
C
30    CALL INROOT('D',IDDEG,RDR,RDI,COEFFD,CHNGFG)
      IF (CHNGFG) DFCTFG=.FALSE.
      GOTO 999
C
C--- READ HELP FILE AND RETURN TO MENU
C
40    CALL INHELP
      GOTO 5
C
999   CONTINUE
      WRITE(1,12)
12    FORMAT(//,' PROCEED ? (YES): ',$)
      IF(.NOT.YES(22,'Y'))MODFLG=.TRUE.
C
      RETURN
      END
C
C
C****************************** GEGAIN ******************************
C********************************************************************
C
C
```

```
C  INPUT SYSTEM GAIN AND CHECK FOR PROPER RANGE.
C
      SUBROUTINE GEGAIN
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
C
      LOGICAL YES
C
      IBLK=5
      WRITE(1,82)
82    FORMAT(//' SET THE GAIN G...')
      WRITE(1,84)GAIN
84    FORMAT(/,' GAIN? (',E10.3,'): ',$)
C
C---- READ GAIN FROM USER
C
      CALL GETREL(GAIN,BLO,BHI,11)
C
C---- EVALUATE SYSTEM GAIN IN FACTORED FORM
C
      GN=COEFFN(INDEG+1)
      GD=COEFFD(IDDEG+1)
      IF((GN.EQ.0).OR.(GD.EQ.0)) THEN
        WRITE(1,1010)
1010    FORMAT(//' WARNING *** ZERO GAIN IN THE',
     2          /' NUMERATOR OR DENOMINATOR. PLEASE',
     3          /' REVISE THE COEFFICIENTS.')
        MODFLG=.TRUE.
        RETURN
      ENDIF
      A=ALOG10(ABS(GAIN))
      B=ALOG10(ABS(GN))
      C=ALOG10(ABS(GD))
      IF ((A+B-C).GT.15.) THEN
        WRITE(1,1000)
1000    FORMAT(//' WARNING *** OVERALL SYSTEM GAIN',
     2          /' IS TOO LARGE. PLEASE REVISE THE',
     3          /' GAIN OR COEFFICIENTS.')
        MODFLG=.TRUE.
        RETURN
      ENDIF
      SYGAIN=GAIN*GN/GD
C
      WRITE(1,86)
86    FORMAT(//,' PROCEED ? (YES): ',$)
      IF(.NOT.YES(22,'Y'))MODFLG=.TRUE.
      CONTINUE
      RETURN
      END
C
C
```

```
C****************************** INHELP ******************************
C********************************************************************
C
C
C  GIVES THE USER A BRIEF OVERVIEW OF THE INPUT OPTIONS
C  FOR THE TRANSFER FUNCTON NUM./DEN. EXPLAINS THE USE OF
C  POLY.,ROOT, AND FACTORED INPUT.
C  WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE INHELP
C
      CHARACTER*3 ANSWER
C
      WRITE(1,10)
10    FORMAT(//' IN MODULE F THE TRANSFER FUNCTION CAN BE ENTERED'
     + /' INTERACTIVELY USING ONE OF THREE DIFFERENT METHODS.'
     + /' IT IS NOT NECESSARY TO USE THE SAME METHOD FOR BOTH'
     + /' THE NUMERATOR AND THE DENOMINATOR.')
      WRITE(1,20)
20    FORMAT(//' 1. SINGLE POLYNOMIAL:'
     + /'     INPUT OF A SINGLE, HIGHER-ORDER POLYNOMIAL. THE'
     + /'     POLYNOMIAL IS DESCRIBED BY ITS COEFFICIENTS.')
      WRITE(1,30)
30    FORMAT(//' 2. FACTORED FORM:'
     + /'     INPUT IS IN THE FORM OF ONE OR MORE FACTORS WHICH'
     + /'     ARE INPUT AS INDIVIDUAL POLYNOMIALS.')
      WRITE(1,40)
40    FORMAT(//' 3. ROOTS OF THE POLYNOMIAL:'
     + /'     INPUT OF THE ROOTS OF THE SINGLE POLYNOMIAL.')
      WRITE(1,50)
50    FORMAT(//' THE SINGLE POLYNOMIAL AND ITS ROOTS ARE '
     + /' AUTOMATICALLY UPDATED TO REFLECT CHANGES MADE THROUGH'
     + /' ANY OF THE INPUT OPTIONS. FACTORED DATA ENTERED WHILE'
     + /' IN OPTION 2 WILL NOT NOT BE UPDATED TO REFLECT CHANGES'
     + /' MADE LATER THROUGH THE OTHER OPTIONS. IF THIS OCCURS'
     + /' THE FACTORED DATA WILL NOT BE VALID UNTIL RE-ENTERED.')
C
      WRITE(1,1110)
1110  FORMAT(//,' HIT <RETURN> TO RETURN TO INPUT MENU. ',$)
      READ(1,'(A3)') ANSWER
C
      RETURN
      END
C
C
C****************************** INPOLY ******************************
C********************************************************************
C
C
C  INPUT AND DISPLAY COEFFICIENTS OF THE POLYNOMIAL.
C  USED BY GETNUM & GETDEN FOR INPUT IN POLYNOMIAL FORM.
C  GIVES THE OPTION TO CHANGE VALUES AND CHECKS FOR VALIDITY.
C  WRITTEN BY J. GREGORSKI APRIL,1985
C
```

```
C  LABEL   :ALPHA NAME OF VECTOR
C  PORDER  :ORDER OF POLYNOMIAL
C  VECTOR  :VECTOR OF COEFFICIENTS
C
      SUBROUTINE INPOLY(LABEL,PORDER,VECTOR,RPART,IPART,CHNGFG)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
C
      CHARACTER*1 LABEL
      INTEGER PORDER
      REAL VECTOR(MAXCOF),RPART(MAXDEG),IPART(MAXDEG),CHECK
      LOGICAL YES,CHNGFG
C
10    CALL POLY(LABEL,PORDER,VECTOR,1)
      WRITE(1,1040)
1040  FORMAT(//,' WANT TO CHANGE IT? (NO): ',$)
      IF(YES(15,'N'))THEN
30      WRITE(1,1060)
1060    FORMAT(/,' ENTER THE NEW COEFFICIENTS:',/)
        DO 40 I=PORDER+1,1,-1
          CHECK=VECTOR(I)
          WRITE(1,1080)I-1,VECTOR(I)
1080      FORMAT('  S**',I1,' ? (',1PE10.3,'): ',$)
          CALL GETREL(VECTOR(I),BLO,BHI,17)
          IF(CHECK.NE.VECTOR(I)) CHNGFG=.TRUE.
          IF(I.LE.PORDER)GO TO 40
          IF(VECTOR(I).NE.0)GO TO 40
          WRITE(1,1085)
1085      FORMAT(/,' LEADING COEFFICIENT MUST NOT',/
     +    ' BE ZERO. PLEASE START AGAIN!')
          GO TO 30
40      CONTINUE
        WRITE(1,1090)
1090    FORMAT(/,' WANT TO SEE IT AGAIN? (NO): ',$)
        IF(YES(11,'N'))GO TO 10
      END IF
C
      IF(PORDER.EQ.0)GO TO 6
      CALL PROOT(PORDER,VECTOR,RPART,IPART,1)
6     CONTINUE
      WRITE(1,7)
7     FORMAT(//,' WANT TO SEE THE ROOTS? (NO): ',$)
      IF(.NOT.YES(13,'N'))GO TO 999
      CALL ZEROS(LABEL,PORDER,RPART,IPART,1)
C
999   RETURN
      END
C
C
C****************************** INROOT ******************************
C********************************************************************
C
C
```

```
C  INPUT OF ROOTS OF NUM. OR DEN. POLYNOMIAL.
C  USED BY BOTH GETNUM & GETDEN.
C  WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE INROOT(LABEL,PORDER,RPART,IPART,VECTOR,CHNGFG)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      REAL RPART(MAXDEG),IPART(MAXDEG),VECTOR(MAXCOF),CHECK,RGAIN
      INTEGER PORDER
      LOGICAL OK,YES,CHNGFG
      CHARACTER*1 LABEL
C
20    CALL ZEROS(LABEL,PORDER,RPART,IPART,1)
      WRITE(1,1)
1     FORMAT(/,' CHANGE THEM? (NO): ',$)
      IF(YES(20,'N'))THEN
        IF(PORDER.GT.0)THEN
          I=1
          WRITE(1,2)
2         FORMAT(/,' ENTER THE NEW ROOTS:')
          CHECK=RPART(I)
10        WRITE(1,3)I,RPART(I)
3         FORMAT(/' REAL PART',I3,'? (',1PE10.3,'): ',$)
          CALL GETREL(RPART(I),BLO,BHI,12)
          IF(CHECK.NE.RPART(I)) CHNGFG=.TRUE.
          IF(I.LT.PORDER)THEN
            CHECK=IPART(I)
            WRITE(1,4)I,IPART(I)
4           FORMAT(' IMAG PART',I3,'? (',1PE10.3,'): ',$)
            CALL GETREL(IPART(I),BLO,BHI,12)
            IF(CHECK.NE.IPART(I)) CHNGFG=.TRUE.
            IF(IPART(I).NE.0.)THEN
              I=I+1
              RPART(I)=RPART(I-1)
              IPART(I)=-IPART(I-1)
              WRITE(1,11)I,RPART(I)
              WRITE(1,12)I,IPART(I)
11            FORMAT(/' REAL PART',I3,'= ',1PE10.3)
12            FORMAT(' IMAG PART',I3,'= ',1PE10.3)
            END IF
          ELSE
            IPART(I)=0.
          END IF
          I=I+1
          IF(I.LE.PORDER)GO TO 10
        END IF
        WRITE(1,5)
5       FORMAT(/,' WANT TO SEE AGAIN? (NO): ',$)
        IF(YES(14,'N'))GO TO 20
      END IF
C
C--- SAVE GAIN IF PROBLEM NOT CHANGED OR SET TO 1. IF IT HAS.
C--- THEN MULTIPLY POLYNOMIAL BY THE SAVED GAIN.
```

```
C
      IF (CHNGFG) THEN
        RGAIN=1.
      ELSE
        RGAIN=VECTOR(PORDER+1)
      ENDIF
      CALL SEMBL(PORDER,RPART,IPART,VECTOR,OK)
      IF(.NOT.OK)GO TO 20
      DO 50 I=1,PORDER+1
50      VECTOR(I)=RGAIN*VECTOR(I)
      WRITE(1,6)
6     FORMAT(/,' WANT TO SEE THE POLYNOMIAL? (NO): ',$)
      IF(.NOT.YES(5,'N'))GOTO 999
      CALL POLY(LABEL,PORDER,VECTOR,1)
C
999   RETURN
      END
C
C
C****************************** FACTOR ******************************
C************************************************************************
C
C
C  INPUT OF POLYNOMIAL IN FACTORED FORM.
C  USED BY BOTH GETNUM & GETDEN.
C  WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE FACTOR(LABEL,PORDER,RPART,IPART,VECTOR,
     + NFND,LNDDEG,CFNDF)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
C
      REAL RPART(MAXDEG),IPART(MAXDEG),VECTOR(MAXCOF)
      REAL ROOTR(MAXDEG),ROOTI(MAXDEG),COEFF(MAXCOF),
     + CFNDF(MAXCOF,MAXDEG),FGAIN
      INTEGER PORDER,FCTORD,NFACT,COUNT,NFND,LNDDEG(MAXDEG)
      LOGICAL OK,YES
      CHARACTER*1 LABEL
C
      COUNT=0
      NFACT=1
C
500   IF(PORDER.EQ.0) THEN
        FCTORD=0
      ELSEIF((PORDER-COUNT).EQ.1) THEN
        FCTORD=1
      ELSE
        FCTORD=LNDDEG(NFACT)
      ENDIF
      DO 130 J=1,MAXCOF
130     COEFF(J)=CFNDF(J,NFACT)
C
      WRITE(1,110) NFACT,FCTORD
```

```
110   FORMAT(/,' ORDER OF FACTOR',I2,'? (',I2,'): ',$)
      CALL GETINT(FCTORD,0,PORDER-COUNT,15)
C
C---- INPUT POLYNOMIAL FOR EACH FACTOR
C---- DISPLAY POLY. AND/OR ROOTS IF DESIRED
C
10    CALL POLY('F',FCTORD,COEFF,NFACT)
11    CALL PROOT(FCTORD,COEFF,ROOTR,ROOTI,1)
      WRITE(1,1040) NFACT
1040  FORMAT(/,' WANT TO SEE ROOTS OF FACTOR ',I2,'? (NO): ',$)
      IF (.NOT.YES(15,'N')) GOTO 99
      CALL ZEROS('F',FCTORD,ROOTR,ROOTI,NFACT)
99    CONTINUE
      WRITE(1,1050) NFACT
1050  FORMAT(/,' WANT TO CHANGE FACTOR ',I2,'? (NO): ',$)
      IF(YES(15,'N'))THEN
30      WRITE(1,1060)
1060    FORMAT(/,' ENTER THE NEW COEFFICIENTS:',/)
        DO 40 I=FCTORD+1,1,-1
          WRITE(1,1080)I-1,COEFF(I)
1080      FORMAT('  S**',I1,' ? (',1PE10.3,'): ',$)
          CALL GETREL(COEFF(I),BLO,BHI,17)
          IF(I.LE.FCTORD)GO TO 40
          IF(COEFF(I).NE.0)GO TO 40
          WRITE(1,1085)
1085      FORMAT(/,' LEADING COEFFICIENT MUST NOT',/
     +    ' BE ZERO. PLEASE START AGAIN!')
          GO TO 30
40      CONTINUE
        WRITE(1,1090) NFACT
1090    FORMAT(/,' WANT TO SEE FACTOR ',I2,' AGAIN? (NO): ',$)
        IF(YES(11,'N'))GO TO 10
        GOTO 11
      END IF
C
C---- STORE FACTOR ROOTS AND ADVANCE FACTOR AND ROOT COUNTERS
C---- ALSO STORE FACTORED FORMAT INFORMATION
C
      DO 75 J=COUNT+1,COUNT+FCTORD
        RPART(J)=ROOTR(J-COUNT)
        IPART(J)=ROOTI(J-COUNT)
75    CONTINUE
      LNDDEG(NFACT)=FCTORD
      DO 135 J=1,MAXCOF
135     CFNDF(J,NFACT)=COEFF(J)
      COUNT=COUNT+FCTORD
      NFACT=NFACT+1
      IF (COUNT.LT.PORDER) GOTO 500
C
C---- ALL FACTORS ENTERED
C
      NFND=NFACT-1
      WRITE(1,2090)
2090  FORMAT(/' WANT TO SEE ALL OF THE ROOTS? (NO): ',$)
```

```
      IF (.NOT.YES(15,'N')) GOTO 150
      CALL ZEROS(LABEL,PORDER,RPART,IPART,1)
C
C--- RETRIEVE THE GAIN FROM FACTORED DATA, GET THE POLYNOMIAL,
C--- AND MULTIPLY POLYNOMIAL BY THE FACTORED GAIN.
C
150   FGAIN=1.
      DO 100 J=1,NFND
100     FGAIN=FGAIN*CFNDF(LNDDEG(J)+1,J)
      CALL SEMBL(PORDER,RPART,IPART,VECTOR,OK)
      IF(.NOT.OK)GO TO 10
      DO 105 I=1,PORDER+1
105     VECTOR(I)=FGAIN*VECTOR(I)
      WRITE(1,6)
6     FORMAT(/,' WANT TO SEE THE POLYNOMIAL? (NO): ',$)
      IF(.NOT.YES(5,'N'))GOTO 999
      CALL POLY(LABEL,PORDER,VECTOR,1)
C
999   RETURN
      END
C
C
C****************************** POLY   ******************************
C*****************************************************************************
C
C
C  DISPLAYS THE COMPLETE POLYNOMIAL FOR THE NUMERATOR,
C  DENOMINATOR, OR INDIVIDUAL FACTORS.
C  WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE POLY(LABEL,PORDER,VECTOR,NFACT)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INTEGER PORDER,NFACT
      REAL VECTOR(MAXCOF)
      CHARACTER*1 LABEL
C
      IF (LABEL.EQ.'N') THEN
        WRITE(1,1000)
1000    FORMAT(//' THE NUMERATOR POLYNOMIAL:',/)
      ELSE IF (LABEL.EQ.'D') THEN
        WRITE(1,1010)
1010    FORMAT(//' THE DENOMINATOR POLYNOMIAL:',/)
      ELSE IF (LABEL.EQ.'F') THEN
        WRITE(1,1020) NFACT
1020    FORMAT(//' THE POLYNOMIAL FOR FACTOR ',I2,':',/)
      ENDIF
      DO 20 I=PORDER,0,-1
20      WRITE(1,1025)VECTOR(I+1),I
1025    FORMAT(2X,1PE10.3,' * S**',I1)
C
      RETURN
      END
C
```

```
C
C****************************** ZEROS  ******************************
C************************************************************************
C
C
C  DISPLAYS ZEROS OF THE POLYNOMIAL FOR THE NUMERATOR,
C  DENOMINATOR, OR AN INDIVIDUAL FACTOR.
C  RE-WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE ZEROS(LABEL,INDEG,RNR,RNI,NFACT)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INTEGER INDEG,NFACT
      REAL RNR(MAXDEG),RNI(MAXDEG)
      CHARACTER*1 LABEL
C
      IF (LABEL.EQ.'N') THEN
        WRITE(1,1000)
1000    FORMAT(//' THE NUMERATOR ROOTS (ZEROS):',/)
      ELSE IF (LABEL.EQ.'D') THEN
        WRITE(1,1010)
1010    FORMAT(//' THE DENOMINATOR ROOTS (POLES):',/)
      ELSE IF (LABEL.EQ.'F') THEN
        WRITE(1,1020) NFACT
1020    FORMAT(//' THE ROOTS FOR FACTOR ',I2,':',/)
      ENDIF
      WRITE(1,11)
11    FORMAT('  REAL PART       IMAG PART')
      DO 30 M=1,INDEG
        WRITE(1,8)RNR(M),RNI(M)
8       FORMAT(1X,1PE10.3,5X,1PE10.3)
30    CONTINUE
C
      RETURN
      END
C
C
C****************************** SEMBL  ******************************
C************************************************************************
C
C
C  GENERATES POLYNOMIAL COEFFICIENTS FOR DESIRED ROOTS.
C  CALLED FROM FACTOR.ASKS FOR NEW ROOTS IF REQUIRED
C  COEFFICIENTS BECOME TO LARGE.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE SEMBL(N,RR,RI,CF,OK)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INTEGER J(MAXCOF)
      REAL RR(MAXDEG),RI(MAXDEG),CF(MAXCOF)
      LOGICAL OK
C
      OK=.TRUE.
```

```
      CF(N+1)=1.0
      DO 14 M=1,N
       SUMR=0.0
       L=1
       J(1)=1
       GO TO 2
1      J(L)=J(L)+1
2      IF(L.EQ.M) GOTO 5
      DO 4 I=L,M-1
4       J(I+1)=J(I)+1
5     PRR=1.0
      PRI=0.0
      DO 7 I=1,N
      K=J(I)
      TEMPR=PRR
      PRR=-PRR*RR(K)+PRI*RI(K)
      PRI=-TEMPR*RI(K)-PRI*RR(K)
C
C---- MAGNITUDE PROTECTION
C
      IF(ABS(PRR).GT.1.E15)GO TO 50
      IF(ABS(PRI).GT.1.E15)GO TO 50
7     CONTINUE
      SUMR=SUMR+PRR
      DO 6 I=1,M
       L=M-I+1
       IF(J(L).LT.(N-M+L)) GOTO 1
6     CONTINUE
14    CF(N-M+1)=SUMR
      RETURN
C
50    WRITE(1,2000)
      OK=.FALSE.
2000  FORMAT(//' COEFFICIENT MAGNITUDE TOO LARGE.',
     2        /' PLEASE MODIFY THE ROOTS.')
      RETURN
      END
C
C
C*************************************************************************
C*************************************************************************
C
C
```

```
C
C
C****************************** FBLOK2 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Selection of the program mode and type of original
C          problem. Performs reading and writing of various problem
C          and results files as well output to the printer.
C
C--- CONTENTS:
C          ENTERF   mode selection and problem entry
C          FPROB    reads or writes problem files
C          SAVPRF   writes problem statement to a file or printer
C          PRINPY   used by SAVPRF to write to printer
C          FRSLT    reads or writes results files
C          GETABD   reads an ENPORT generated ABDATA file
C          FLEAVE   exit from Module F
C
C--- INDEX:
C          ENTERF
C          FLEAVE
C          FPROB
C          FRSLT
C          GETABD
C          PRINPY
C          SAVPRF
C
C
C****************************** ENTERF ******************************
C************************************************************************
C
C
C  BLOCK FOR PERFORMING DATA RELOAD FROM FILE, IF DESIRED.
C  ALSO DETERMINES IF ROOT LOCUS OR FREQUENCY RESPONSE
C  CALCULATIONS ARE TO BE PERFORMED.
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE ENTERF
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
      LOGICAL YES,OK
C
      IBLK=1
      WRITE(1,1010)
1010  FORMAT(//' DO YOU WANT TO RUN')
100    WRITE(1,1020)
1020  FORMAT('  FREQUENCY RESPONSE? (YES): ',$)
      IF (YES(11,'Y')) THEN
        FRQYES=.TRUE.
        GOTO 1
      ENDIF
```

```
      WRITE(1,1030)
1030  FORMAT('  ROOT LOCUS? (YES): ',$)
      IF (YES(19,'Y')) THEN
        FRQYES=.FALSE.
        GOTO 1
      ENDIF
      GOTO 100
C
1     MODFLG=.FALSE.
      WRITE(1,150)
150    FORMAT(//,' PROCEED ? (YES): ',$)
      IF(.NOT.YES(22,'Y')) THEN
        MODFLG=.TRUE.
        RETURN
      ENDIF
C
5     WRITE(1,1000)
1000  FORMAT(/' INITIAL PROBLEM ENTRY OPTIONS',
     1 /' ------------------------------------',
     2 /' R: RELOAD A PREVIOUSLY SAVED PROBLEM FILE',
     3 /' L: LOAD AN ENPORT GENERATED ABDATA FILE',
     4 /' E: ENTER A NEW PROBLEM INTERACTIVELY (=DEF)',
     8 /' ------------------------------------',
     6 /' ENTER OPTION (E): ',$)
C
      CALL MENU(OK,IX,'R','L','E','','','',
     2 '
      IF (.NOT.OK) GOTO 5
      GOTO (10,20,30) IX
C
C---- RELOAD A SAVED PROBLEM FILE
C
10    PRBFIL=.TRUE.
      LIBNAM='PROBLEM*LIB'
      CALL LODFIL
      GOTO 50
C
C---- LOAD AN ENPORT ABDATA FILE
C
20    CALL GETABD
      GOTO 50
C
C---- CONTINUE AND ENTER PROBLEM INTERACTIVELY
C
30    RETURN
C
50    MODFLG=.FALSE.
      WRITE(1,150)
      IF(.NOT.YES(22,'Y'))MODFLG=.TRUE.
      RETURN
      END
C
C
C****************************** FPROB  ******************************
```

```
C*********************************************************************
C
C
C  READS OR WRITES PROBLEM FILES FOR MODULE F.
C  CAN READ FILES CREATED BY MODULES T AND F.
C  FILES CREATED IN MODULE T ARE CONVERTED TO THE
C  TRANSFER FUNCTION FORMAT OF MODULE F.
C  CALLED BY SAVFIL OR LODFIL FROM UNIFIL.FTN
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE FPROB(FILNAM,READIN,ERROR)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FACTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      CHARACTER*15 FILNAM
      INTEGER IMOD
      LOGICAL BADFIL,BADDAT,READIN,ERROR
C
      LUN=5
      ERROR=.FALSE.
C
C--- CREATE NEW FILE OR READ OLD PROBLEM FILE.
C
      IF (.NOT.READIN) THEN
C
C--- BEGIN WRITING PROBLEM TO NEW FILE.
C
       OPEN(LUN,FILE=FILNAM,ERR=991,STATUS='UNKNOWN',
     2 FORM='UNFORMATTED')
       IMOD=4
         WRITE(LUN,ERR=991)IMOD
         WRITE(LUN,ERR=991)COEFFN,COEFFD
         WRITE(LUN,ERR=991)INDEG,IDDEG
         WRITE(LUN,ERR=991)GAIN
         WRITE(LUN,ERR=991)TITLE1
         WRITE(LUN,ERR=991)RNR,RNI,RDR,RDI
         WRITE(LUN,ERR=991)SYGAIN
         WRITE(LUN,ERR=991)SCMIN,SCMAX,NPTS,SCLOG
         WRITE(LUN,ERR=991)NFCTFG,DFCTFG
         IF(.NOT.NFCTFG) GOTO 10
         WRITE(LUN,ERR=991)NFNUM,LNDEG
         WRITE(LUN,ERR=991)CFNF
10       IF(.NOT.DFCTFG) GOTO 20
         WRITE(LUN,ERR=991)NFDEN,LDDEG
         WRITE(LUN,ERR=991)CFDF
20       ENDFILE(LUN)
         CLOSE(LUN,STATUS='KEEP')
         GOTO 999
       ELSE
C
```

```
C--- BEGIN READING INFORMATION FROM FILE SPECIFIED
C
      OPEN(LUN,FILE=FILNAM,ERR=991,STATUS='OLD',
     2 FORM='UNFORMATTED')
      READ(LUN,END=991,ERR=991)IMOD
C
C--- READ TRANSFER FUNCTION TYPE PROBLEM FILE
C
      IF(IMOD.EQ.4)THEN
        READ(LUN,END=991,ERR=991)COEFFN,COEFFD
        READ(LUN,END=991,ERR=991)INDEG,IDDEG
        READ(LUN,END=991,ERR=991)GAIN
        READ(LUN,END=991,ERR=991)TITLE1
        READ(LUN,END=991,ERR=991)RNR,RNI,RDR,RDI
        READ(LUN,END=991,ERR=991)SYGAIN
        READ(LUN,END=991,ERR=991)SCMIN,SCMAX,NPTS,SCLOG
        READ(LUN,END=991,ERR=991)NFCTFG,DFCTFG
        IF(.NOT.NFCTFG) GOTO 30
        READ(LUN,END=991,ERR=991)NFNUM,LNDEG
        READ(LUN,END=991,ERR=991)CFNF
30      IF(.NOT.DFCTFG) GOTO 40
        READ(LUN,END=991,ERR=991)NFDEN,LDDEG
        READ(LUN,END=991,ERR=991)CFDF
40      CLOSE(LUN,STATUS='KEEP')
C
      ELSE IF (IMOD.EQ.1) THEN
C
C--- STATE SPACE FORMULATION. USE GETTFN TO CONVERT.
C--- FILE IS CLOSED BY GETTFN
C
        CALL GETTFN(BADFIL,BADDAT,LUN)
        IF (BADFIL) GOTO 991
        IF (BADDAT) THEN
          WRITE(1,1010)
1010      FORMAT(//' UNABLE TO CONVERT THIS DATA TO A',
     2           /' TRANSFER FUNCTION SUCCESSFULLY.')
        ELSE
          WRITE(1,1015)
1015      FORMAT(//' DATA CONVERTED TO TRANSFER FUNCTION.')
        ENDIF
        SCMIN=0.
        IF (FRQYES) SCMIN=1.
        SCMAX=10.
        NPTS=11
        SCLOG=.FALSE.
      ELSE
        CLOSE(LUN,STATUS='KEEP')
        ERROR=.TRUE.
        WRITE(1,997)
997     FORMAT(//' **WARNING** THIS FILE IS NOT VALID',/,
     2    ' INPUT FOR MODULE F. I CANNOT READ IT.',/,
     3    ' PLEASE TRY ANOTHER PROBLEM FILE.')
      ENDIF
      ENDIF
```

```
      GOTO 999
C
991   ERROR=.TRUE.
      WRITE(1,993) FILNAM
993   FORMAT(//' *** ERROR ***  THE FILE: ',A15,/,
     2 ' IS NOT USABLE.  CHOOSE ANOTHER.  ')
      CLOSE(LUN,STATUS='KEEP')
C
999   RETURN
      END
C
C--- DUMMY PROBLEM READING SUBROUTINES FOR MODULES T AND C
C--- ADDED TO AVOID LOAD NOT COMPLETE FROM CALLS IN UNIFIL.
C
      SUBROUTINE CPROB(FILNAM,READIN,ERROR)
      CHARACTER*15 FILNAM
      LOGICAL READIN,ERROR
      WRITE(1,100)
100   FORMAT(//' THIS IS THE PROBLEM READER FOR MODULE C.'
     + /' YOU SHOULD NOT BE HERE!')
      ERROR=.TRUE.
      RETURN
      END
C
      SUBROUTINE TPROB(FILNAM,READIN,ERROR)
      CHARACTER*15 FILNAM
      LOGICAL READIN,ERROR
      WRITE(1,100)
100   FORMAT(//' THIS IS THE PROBLEM READER FOR MODULE T.'
     + /' YOU SHOULD NOT BE HERE!')
      ERROR=.TRUE.
      RETURN
      END
C
C
C****************************** SAVPRF ********************************
C**********************************************************************
C
C
C  IT QUERIES THE USER, AND WRITES INFORMATION TO A PRINTER
C  OR FILE IF REQUESTED, THEN RETURNS TO CALLING PROGRAM
C  FOR CONTINUATION OR EXIT.
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE SAVPRF
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
      LOGICAL YES
      CHARACTER*3 ANSWER
C
      WRITE(1,1000)
1000  FORMAT(//' DO YOU WANT TO LIST',
     2        /'  THE CURRENT PROBLEM STATEMENT',
     3        /'   ON THE PRINTER? (NO): ',$)
```

```
      IF (YES(14,'N')) THEN
        WRITE(1,1010)
1010    FORMAT(//' READY THE PRINTER',
     2          /'  AND HIT <RETURN>...',$)
        READ(1,'(A3)') ANSWER
        CALL PRINPY
      ENDIF
C
      WRITE(1,101)
101   FORMAT(/' DO YOU WISH TO SAVE THE CURRENT',/,
     2' PROBLEM STATEMENT IN A FILE? (NO): ',$)
      IF(YES(16,'N'))THEN
        PRBFIL=.TRUE.
        LIBNAM='PROBLEM*LIB'
        CALL SAVFIL
      ENDIF
C
      RETURN
      END
C
C
C****************************** PRINPY ********************************
C**********************************************************************
C
C
C  WRITES THE CURRENT PROBLEM DESCRIPTION DIRECTLY
C  TO THE PRINTER DURING PROGRAM EXECUTION.
C  DATAPR IS USED TO SET UP THE PRINTER.
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE PRINPY
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FACTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
C
      CHARACTER*6 SCALE
C
      LUN=13
      CALL DATAPR
      WRITE(LUN,1015)
1015  FORMAT(//' *** MODULE F PROBLEM DESCRIPTION ***')
      WRITE(LUN,1020) TITLE1
1020  FORMAT(//' THE PROBLEM TITLE IS:',//,1X,A40)
C
      IF(.NOT.NFCTFG)GOTO 13
      DO 21 NFACT=1,NFNUM
        WRITE(LUN,1031)NFACT
1031    FORMAT(//' NUMERATOR FACTOR ',I2,':',/)
        DO 21 I=LNDEG(NFACT),0,-1
21        WRITE(LUN,1040) CFNF(I+1,NFACT),I
```

```
1040  FORMAT(2X,1PE10.3,' * S**',I1)
13    WRITE(LUN,1030)
1030  FORMAT(//' THE NUMERATOR POLYNOMIAL:',/)
      DO 20 I=INDEG,0,-1
20      WRITE(LUN,1040) COEFFN(I+1),I
C
      IF(.NOT.DFCTFG)GOTO 14
      DO 31 NFACT=1,NFDEN
        WRITE(LUN,1051)NFACT
1051    FORMAT(//' DENOMINATOR FACTOR ',I2,':',/)
        DO 31 I=LDDEG(NFACT),0,-1
31        WRITE(LUN,1040) CFDF(I+1,NFACT),I
14    WRITE(LUN,1050)
1050  FORMAT(//' THE DENOMINATOR POLYNOMIAL:',/)
      DO 30 I=IDDEG,0,-1
30      WRITE(LUN,1040) COEFFD(I+1),I
      WRITE(LUN,1060) GAIN
1060  FORMAT(/' THE GAIN IS ',1PE10.3)
C
      WRITE(LUN,2010)
2010  FORMAT(//' THE NUMERATOR ROOTS:')
      WRITE(LUN,2020)
2020  FORMAT(/'  REAL PART       IMAG PART')
      DO 40 I=1,INDEG
40      WRITE(LUN,2030) RNR(I),RNI(I)
2030    FORMAT(1X,1PE10.3,5X,1PE10.3)
      WRITE(LUN,2040)
2040  FORMAT(/' THE DENOMINATOR ROOTS:')
      WRITE(LUN,2020)
      DO 50 I=1,IDDEG
50      WRITE(LUN,2030) RDR(I),RDI(I)
      WRITE(LUN,2050) SYGAIN
2050  FORMAT(/' THE SYSTEM GAIN IS  ',1PE10.3)
      IF (SCLOG) THEN
        SCALE='LOG   '
      ELSE
        SCALE='LINEAR'
      ENDIF
      IF (FRQYES) THEN
        WRITE(LUN,3004)
3004    FORMAT(//' FREQUENCY SCALING:')
      ELSE
        WRITE(LUN,3005)
3005    FORMAT(//' ROOT LOCUS SCALING:')
      ENDIF
      WRITE(LUN,3010) SCMIN,SCMAX,NPTS,SCALE
3010  FORMAT(/'  MINIMUM IS ',1PE10.3,
     2         /'  MAXIMUM IS ',1PE10.3,
     3         /'  IN ',I5,' POINTS ON A ',A6,' SCALE.')
C
      WRITE(LUN,4000)
4000  FORMAT(//' *** END OF PROBLEM STATEMENT ***')
      CLOSE(LUN)
C
```

```
      RETURN
      END
C
C
C***************************** FRSLT  ******************************
C*********************************************************************
C
C
C  READS OR WRITES RESULTS FILES FOR MODULE F.
C  USED FOR BOTH ROOT LOCUS AND FREQ. RESP.
C  MODE AND SCALE TYPE ARE CHANGED IF NECESSARY
C  TO ACCOMODATE RESULTS.
C  CALLED BY SAVFIL OR LODFIL FROM UNIFIL.FTN
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE FRSLT(FILNAM,READIN,ERROR)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RESULT.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RSLTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
C
      CHARACTER*15 FILNAM
      INTEGER IMOD
      LOGICAL READIN,ERROR
C
      LUN=5
      ERROR=.FALSE.
C
C--- CREATE NEW FILE OR READ OLD PROBLEM FILE.
C
      IF (.NOT.READIN) THEN
C
C--- BEGIN WRITING PROBLEM TO NEW FILE.
C
       OPEN(LUN,FILE=FILNAM,ERR=991,STATUS='UNKNOWN',
     2 FORM='UNFORMATTED')
       IMOD=4
       IF (FRQYES) THEN
         WRITE(LUN,ERR=991)IMOD
         WRITE(LUN,ERR=991)FRQYES
         WRITE(LUN,ERR=991)TITLE1,TITLE2
         WRITE(LUN,ERR=991)NPTS,SCLOG,SCMIN,SCMAX
         WRITE(LUN,ERR=991)WGC,PM,WPHC,GM
         WRITE(LUN,ERR=991)(W(I),I=1,NPTS)
         WRITE(LUN,ERR=991)(TRFMAG(I),I=1,NPTS)
         WRITE(LUN,ERR=991)(PHID(I),I=1,NPTS)
         WRITE(LUN,ERR=991)(PHI(I),I=1,NPTS)
       ELSE
         WRITE(LUN,ERR=991)IMOD
         WRITE(LUN,ERR=991)FRQYES
         WRITE(LUN,ERR=991)TITLE1,TITLE2
         WRITE(LUN,ERR=991)NPTS,SCLOG,SCMIN,SCMAX
```

```
        WRITE(LUN,ERR=991)IPOLEX,SIGMA
        WRITE(LUN,ERR=991)(KGAIN(I),I=1,NPTS)
        WRITE(LUN,ERR=991)((RR(J,I),I=1,NPTS),J=1,IPOLEX)
        WRITE(LUN,ERR=991)((RI(J,I),I=1,NPTS),J=1,IPOLEX)
        WRITE(LUN,ERR=991)(ANGLE(I),I=1,IPOLEX)
      ENDIF
      ENDFILE(LUN)
      CLOSE(LUN,STATUS='KEEP')
      GOTO 999
C
C--- BEGIN READING INFORMATION FROM FILE SPECIFIED
C
      ELSE
      OPEN(LUN,FILE=FILNAM,ERR=991,STATUS='OLD',
     2 FORM='UNFORMATTED')
      READ(LUN,END=991,ERR=991)IMOD
      READ(LUN,END=991,ERR=991)FRQYES
      IF(IMOD.EQ.4)THEN
        IF (FRQYES) THEN
          READ(LUN,END=991,ERR=991)TITLE1,TITLE2
          READ(LUN,END=991,ERR=991)NPTS,SCLOG,SCMIN,SCMAX
          READ(LUN,END=991,ERR=991)VGC,PM,VPHC,GM
          READ(LUN,END=991,ERR=991)(V(I),I=1,NPTS)
          READ(LUN,END=991,ERR=991)(TRFMAG(I),I=1,NPTS)
          READ(LUN,END=991,ERR=991)(PHID(I),I=1,NPTS)
          READ(LUN,END=991,ERR=991)(PHI(I),I=1,NPTS)
        ELSE
          READ(LUN,END=991,ERR=991)TITLE1,TITLE2
          READ(LUN,END=991,ERR=991)NPTS,SCLOG,SCMIN,SCMAX
          READ(LUN,END=991,ERR=991)IPOLEX,SIGMA
          READ(LUN,END=991,ERR=991)(KGAIN(I),I=1,NPTS)
          READ(LUN,END=991,ERR=991)((RR(J,I),I=1,NPTS),J=1,IPOLEX)
          READ(LUN,END=991,ERR=991)((RI(J,I),I=1,NPTS),J=1,IPOLEX)
          READ(LUN,END=991,ERR=991)(ANGLE(I),I=1,IPOLEX)
        ENDIF
      ELSE
        ERROR=.TRUE.
        WRITE(1,997)
997     FORMAT(//' **WARNING** THIS FILE IS NOT VALID',/,
     2    ' INPUT FOR MODULE F. I CANNOT READ IT.',/,
     3    ' PLEASE TRY ANOTHER RESULTS FILE.')
      ENDIF
        CLOSE(LUN,STATUS='KEEP')
      ENDIF
      GOTO 999
C
991   ERROR=.TRUE.
      WRITE(1,993) FILNAM
993   FORMAT(//' *** ERROR ***  THE FILE: ',A15,/,
     2 ' IS NOT USABLE.  CHOOSE ANOTHER.  ')
      CLOSE(LUN,STATUS='KEEP')
C
999   RETURN
      END
```

```
C
C--- DUMMY RESULTS READING SUBROUTINES FOR MODULES T AND C
C--- ADDED TO AVOID LOAD NOT COMPLETE FROM CALLS IN UNIFIL.
C
      SUBROUTINE CRSLT(FILNAM,READIN,ERROR)
      CHARACTER*15 FILNAM
      LOGICAL READIN,ERROR
      WRITE(1,100)
100   FORMAT(//' THIS IS THE RESULTS READER FOR MODULE C.'
     + /' YOU SHOULD NOT BE HERE!')
      ERROR=.TRUE.
      RETURN
      END
C
      SUBROUTINE TRSLT(FILNAM,READIN,ERROR)
      CHARACTER*15 FILNAM
      LOGICAL READIN,ERROR
      WRITE(1,100)
100   FORMAT(//' THIS IS THE RESULTS READER FOR MODULE T.'
     + /' YOU SHOULD NOT BE HERE!')
      ERROR=.TRUE.
      RETURN
      END
C
C
C****************************** GETABD ******************************
C*********************************************************************
C
C
C  READS AN ENPORT GENERATED ABDATA FILE AND USES
C  THE A AND B MATRICES AND EIGENVALUES TO CREATE
C  TRANSFER FUNCTION DESCRIPTION OF THE PROBLEM.
C  USER IS ASKED TO SELECT THE STATE VARIABLE FOR
C  THE OUTPUT AND THE COLUMN OF B FOR INPUT.
C  THIS ROUTINE IS BASED ON GETTFN SUBROUTINE.
C  WRITTEN BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE GETABD
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
C
      LOGICAL BADDAT,EXST,EXIT
      INTEGER DIMX,DIMU,NUMU,NUMX
      REAL BCOL(MAXDIM),CROW(MAXDIM)
      REAL A(MAXDIM,MAXDIM),B(MAXDIM,MAXDIM)
      REAL EVALRP(MAXDIM),EVALIP(MAXDIM)
      CHARACTER NAME*15,DUMMY*35,TEMPTL*40
      BADDAT=.FALSE.
      MODFLG=.TRUE.
C
```

```
C--- ENTER THE NAME OF THE ABDATA FILE TO LOAD
C
10    WRITE(1,1020)
1020  FORMAT(/' *** ENTER NAME OF THE ABDATA FILE *** ')
      CALL FNAME(NAME,EXIT)
      IF (EXIT) THEN
        WRITE(1,1025)
1025    FORMAT(/' LOADER EXITED. NO PROBLEM LOADED. ')
        RETURN
      ENDIF
C
C--- IF THE DESIRED ABDATA FILE EXISTS LOAD IT
C--- IF NOT THEN ASK FOR A NEW NAME
C
      INQUIRE(FILE=NAME,EXIST=EXST)
      IF (.NOT.EXST) THEN
        WRITE(1,1030)
1030    FORMAT(/' THIS FILE DOES NOT EXIST.')
        GOTO 10
      ENDIF
C
C--- OPEN THE FILE AND READ THE DATA
C--- DUMMY READS UNWANTED HEADINGS ETC...
C
      LUN=8
      OPEN(LUN,FILE=NAME,ERR=991)
      READ(LUN,'(//A)',ERR=991,END=991) TEMPTL
      READ(LUN,'(/A31,I4,A23,I4)',ERR=991,END=991)
     2 DUMMY,DIMX,DUMMY,DIMU
C
C--- CHECK TO SEE IF DIMENSION OF PROBLEM IS
C--- WITHIN THE RANGE COMPATIBLE WITH THIS SUB.
C
      IF ((DIMX.GT.MAXDIM).OR.(DIMU.GT.MAXDIM)) THEN
        WRITE(1,1045)
1045    FORMAT(/' *** THIS PROBLEM IS TOO LARGE ***')
        GOTO 991
      ENDIF
C
C--- READ THE A MATRIX
C
      READ(LUN,'(/)',ERR=991,END=991)
      DO 30 I=1,DIMX
        READ(LUN,1100,ERR=991,END=991) IZ,(A(I,J),J=1,DIMX)
1100    FORMAT(/1X,I3,2X,5E13.5:/,(6X,5E13.5))
30    CONTINUE
C
C--- READ THE B MATRIX IF IT EXISTS
C
      IF (DIMU.GT.0) THEN
        READ(LUN,'(/)',ERR=991,END=991)
        DO 40 I=1,DIMX
          READ(LUN,1100,ERR=991,END=991) IZ,(B(I,J),J=1,DIMU)
40      CONTINUE
```

```
      ENDIF
C
C--- READ THE EIGENVALUES
C
      READ(LUN,'(///)',ERR=991,END=991)
      DO 70 I=1,DIMX
        READ(LUN,1110,ERR=991,END=991) IZ,EVALRP(I),EVALIP(I)
1110    FORMAT(1X,I3,2X,E12.5,1X,E12.5)
70    CONTINUE
      CLOSE(LUN,STATUS='KEEP')
C
C--- DETERMINE THE INPUT AND OUTPUT
C
      NUMU=1
      IF (DIMU.GT.0) THEN
        WRITE(1,1080)
1080    FORMAT(/' WHICH U TO BE USED AS INPUT? (1): ',$)
        CALL GETINT(NUMU,1,DIMU,14)
      ELSE
        WRITE(1,1070)
1070    FORMAT(//' *** NO INPUT VARIABLES *** ')
        GOTO 991
      ENDIF
C
      NUMX=1
      IF (DIMX.GT.0) THEN
        WRITE(1,1040)
1040    FORMAT(/' WHICH X TO BE USED AS OUTPUT? (1): ',$)
        CALL GETINT(NUMX,1,DIMX,22)
      ELSE
        WRITE(1,1050)
1050    FORMAT(//' *** NO STATE VARIABLES *** ')
        GOTO 991
      ENDIF
C
C--- SET UP THE PROPER BCOL,CROW,DELEM
C
      DO 210 I=1,DIMX
        BCOL(I)=B(I,NUMU)
210     CROW(I)=0.0
      DELEM=0.0
      CROW(NUMX)=1.0
C
C
C--- GET THE TRANSFER FUNCTION HERE
C--- AND STORE IN POLYBK, ROOTBK
C
      IDDEG=DIMX
      CALL CONMAT(A,BCOL,CROW,DELEM,EVALRP,EVALIP,BADDAT)
C
      IF (BADDAT) THEN
        WRITE(1,1010)
1010    FORMAT(//' UNABLE TO CONVERT THIS DATA TO A',
     2           /' TRANSFER FUNCTION SUCCESSFULLY.')
```

```
        GOTO 991
      ELSE
        WRITE(1,1015)
1015    FORMAT(//' DATA CONVERTED TO TRANSFER FUNCTION.')
        TITLE1=TEMPTL
      ENDIF
      SCMIN=0.
      IF (FRQYES) SCMIN=1.
      SCMAX=10.
      NPTS=25
      SCLOG=.FALSE.
      GOTO 999
C
991   CLOSE(LUN,STATUS='KEEP')
      WRITE(1,993) NAME
993   FORMAT(//' *** ERROR ***  THE ABDATA FILE: ',A15,/,
     2 ' IS NOT USABLE AS INPUT TO MODULE F.  ')
C
999   RETURN
      END
C
C
C****************************** FLEAVE ********************************
C**********************************************************************
C
C
C  ASKS IF USER WANTS TO EXIT MODULE F.
C  IF YES THEN IT GIVES A MESSAGE AND EXITS.
C  IF NO THEN RETURN TO MAIN MENU.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE FLEAVE
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      LOGICAL YES
C
      IBLK=8
      WRITE(1,100)
100   FORMAT(//' WANT TO LEAVE THIS MODULE? (YES): ',$)
      IF (YES(5,'Y')) THEN
        WRITE(1,110)
110     FORMAT(//'     SEE YOU LATER...'/
     8' *** EXITED MODULE F ***')
        MODFLG=.FALSE.
      ELSE
        MODFLG=.TRUE.
      ENDIF
      RETURN
      END
C
C
C**********************************************************************
C**********************************************************************
C
```

```
C
C
C****************************** FBLOK3 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Sets the range for freq. resp. calculations and the
C          type of scale. Performs the freq. resp. calculations
C          and displays the results.
C
C--- CONTENTS:
C          FSCALE  sets freq./gain range and scale type
C          FDISP   displays frequency response results
C          FRESP   calculates the frequency response
C          PHMARG  determines phase & gain margins etc...
C          TL2GET  enter or change the results title
C
C--- INDEX:
C          FDISP
C          FRESP
C          FSCALE
C          PHMARG
C          TL2GET
C
C
C****************************** FSCALE ******************************
C************************************************************************
C
C
C  SETS THE MAX./MIN. FREQUENCY AND ROOT LOCUS
C  SCALE VALUES AND INPUTS THE TYPE OF SCALING.
C  LOG. OR LINEAR ,DEFAULT=LOG.
C
      SUBROUTINE FSCALE
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      LOGICAL YES
C
      IBLK=6
      SHI=1.E10
      IF (FRQYES) THEN
        SLO=1.E-10
C
C--- CHOOSE DEFAULT SCMIN AND SCMAX
C
        VMIN=SHI
        VMAX=0.
        DO 10 I=1,INDEG
          VALS=RNI(I)*RNI(I)+RNR(I)*RNR(I)
          IF (VALS.GT.0.) VAL=SQRT(VALS)
```

```
          IF (VALS.EQ.0.) VAL=0.
          IF (VAL.GT.VMAX) VMAX=VAL
          IF (VAL.LT.VMIN) VMIN=VAL
10        CONTINUE
        DO 20 I=1,IDDEG
          VALS=RDI(I)*RDI(I)+RDR(I)*RDR(I)
          IF (VALS.GT.0.) VAL=SQRT(VALS)
          IF (VALS.EQ.0.) VAL=0.
          IF (VAL.GT.VMAX) VMAX=VAL
          IF (VAL.LT.VMIN) VMIN=VAL
20        CONTINUE
        IF (VMIN.EQ.0.) VMIN=1.
        IF (VMAX.EQ.0.) VMAX=1.
        IF (VMIN.GT.VMAX) VMIN=VMAX
        SCMIN=VMIN*0.1
        SCMAX=VMAX*10.
        IF (SCMIN.LT.SLO) SCMIN=SLO
        IF (SCMAX.GT.SHI) SCMAX=SHI
        WRITE(1,1010)
1010    FORMAT(//' SET THE FREQUENCY RANGE...')
      ELSE
        SLO=-SHI
        WRITE(1,1020)
1020    FORMAT(//' SET THE K GAIN RANGE...')
      ENDIF
C
C--- SCMIN: MINIMUM SCALE VALUE
C
      WRITE(1,70)SCMIN
70    FORMAT(/,'  MINIMUM? (',1PE10.3,'): ',$)
      CALL GETREL(SCMIN,SLO,SHI,16)
C
C--- SCMAX: MAXIMUM SCALE VALUE
C
72    WRITE(1,69)SCMAX
69    FORMAT(/,'  MAXIMUM? (',1PE10.3,'): ',$)
      CALL GETREL(SCMAX,SCMIN,SHI,16)
      IF(SCMAX.EQ.SCMIN)GO TO 72
C
C--- SET TYPE OF SCALE
C
77    SCLOG=.TRUE.
      WRITE(1,74)'LOG    '
74    FORMAT(/,' THE SCALING IS ',A6)
      WRITE(1,75)
75    FORMAT(/,' DO YOU WANT TO CHANGE IT? (NO): ',$)
      IF(.NOT.YES(7,'N'))GO TO 76
      SCLOG=.FALSE.
      WRITE(1,74)'LINEAR'
      WRITE(1,75)
      IF(YES(7,'N'))GO TO 77
76    CONTINUE
C
      WRITE(1,67)
```

```
67    FORMAT(//,' PROCEED ? (YES): ',$)
      IF(.NOT.YES(22,'Y'))MODFLG=.TRUE.
      CONTINUE
      RETURN
      END
C
C
C***************************** FDISP  ******************************
C*********************************************************************
C
C
C  DISPLAYS FREQUENCY RESPONSE RESULTS IN EITHER
C  TABULAR OR GRAPHICAL FORM. TABULAR RESULTS CAN
C  BE WRITTEN TO THE SCREEN, PRINTER, OR A FILE.
C  GRAPHICS OUTPUT IS IN THE FORM OF BODE OR
C  NYQUIST PLOTS WHICH CAN BE SAVED IN GRAPHICS
C  FILES FOR LATER OUTPUT TO THE PRINTER.
C  WRITTEN BY J. GREGORSKI AUGUST,1984
C
      SUBROUTINE FDISP
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FOUTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RESULT.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>TERMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      LOGICAL YES,OK,EXST,AOK,EXIT
      CHARACTER*3 ANSWER
      CHARACTER*15 FILE
      CHARACTER*17 GFILE
      IBLK=7
      WMIN=SCMIN
      WMAX=SCMAX
      WTRMAX=WMAX
      WTRMIN=WMIN
      WLOG=SCLOG
      Z1=WMIN
      Z2=WMAX
C
14    WRITE(1,1200)
1200  FORMAT(/,' RESULT OPTIONS',/,
     2' ----------------------',/,
     3' S:  DISPLAY ON SCREEN',/,
     4' L:  LIST TO PRINTER',/,
     5' W:  WRITE TO FILE',/,
     6' B:  BODE PLOT',/,
     7' N:  NYQUIST PLOT')
      WRITE(1,1210)
1210  FORMAT(' R:  RETURN TO MAIN MENU',/,
     2' P:  PROCEED(=DEF)',/,
     3' ----------------------',/,
```

```
     3' ENTER OPTION (P): ',$)
      CALL MENU(OK,IX,'S','L','W','B','N','R','P','','
     2 'P',25)
      IF(.NOT.OK)GO TO 14
C
C--- CHECK TO SEE IF PLOT DESIRED AND ABLE TO DO IT.
C
      IF ((((IX.EQ.4).OR.(IX.EQ.5.)).AND.(.NOT.PLOTIT)) THEN
          WRITE(1,210)
210       FORMAT(/,' *** PLOTTING ROUTINES NOT COMPATIBLE',
     2    ' WITH THIS TERMINAL ***',//)
          GOTO 14
        ENDIF
C
      GO TO(100,100,100,100,100,98,99)IX
C
C--- GET TITLE2 FOR RESULTS
C
100   WRITE(1,1300)
1300  FORMAT(//,' SET THE TITLE FOR RESULTS...')
      CALL TL2GET
C
C--- GET FREQ LIMITS FOR DISPLAY OR STORE
C
      WRITE(1,1400)
1400  FORMAT(//,' SELECT THE FREQUENCY RANGE',/,
     2          ' FOR DISPLAY OR PLOTTING...',/)
      WRITE(1,1410)Z1
1410  FORMAT(' LOW  FREQUENCY? (',1PE10.3,'): ',$)
      CALL GETREL(Z1,WMIN,WMAX,10)
      IF (Z1.GT.Z2) Z2=Z1
      WRITE(1,1420)Z2
1420  FORMAT(' HIGH FREQUENCY? (',1PE10.3,'): ',$)
      CALL GETREL(Z2,Z1,WMAX,9)
C
C--- GENERATE DISPLAY VALUES USING FRESP OUTPUT AND THE
C--- DESIRED FREQUENCY LIMITS.
C
      II=0
      NPTOUT=0
      DO 250 I=1,NPTS
        IF(W(I).GT.Z2) GOTO 260
        IF(W(I).LT.Z1) GOTO 250
        II=II+1
        NPTOUT=NPTOUT+1
        WOUT(II)=W(I)
        MAGOUT(II)=TRFMAG(I)
        PHDOUT(II)=PHID(I)
        PHOUT(II)=PHI(I)
250   CONTINUE
260   CONTINUE
C
C--- GO TO PLOTTING SECTION IF SO DESIRED.
C
```

```
      IF (IX.GT.3) GOTO 200
C
C--- DETERMINE THE MAX. & MIN. MAGNITUDES AND THE
C--- ASSOCIATED FREQUENCIES FOR DISPLAY OF OUTPUT.
C
      TRMAX = MAGOUT(1)
      TRMIN = TRMAX
      WTRMAX = WOUT(1)
      WTRMIN = WTRMAX
      DO 310 I=2,NPTOUT
       IF (TRMAX.LT.MAGOUT(I)) THEN
         TRMAX = MAGOUT(I)
         WTRMAX = WOUT(I)
       ELSEIF (TRMIN.GT.MAGOUT(I)) THEN
         TRMIN = MAGOUT(I)
         WTRMIN = WOUT(I)
       ENDIF
310   CONTINUE
C
C--- SETUP OUTPUT TO SCREEN,PRINTER,OR RESULTS FILE.
C
      GOTO (110,120,130) IX
C
C--- OUTPUT TO THE SCREEN.
C
110   CONTINUE
      LUN=1
      WRITE(1,1110)
1110  FORMAT(//,' HIT <RETURN> TO START DISPLAY. ',$)
      READ(1,'(A3)') ANSWER
      GOTO 1000
C
C--- OUTPUT TO THE PRINTER.
C
120   CONTINUE
      LUN=13
        WRITE(1,1220)
1220    FORMAT(//,' READY THE PRINTER',
     2          /,' AND HIT <RETURN>....',$)
      READ(1,'(A3)') ANSWER
C
      CALL DATAPR
      GOTO 1000
C
C--- OUTPUT TO A RESULTS FILE.
C
130   CONTINUE
      WRITE(1,131)
131   FORMAT(/' DO YOU WISH TO SAVE THE',/,
     2' RESULTS IN A FILE? (NO): ',$)
      IF(YES(16,'N'))THEN
        PRBFIL=.FALSE.
        LIBNAM='RESULTS*LIB'
        CALL SAVFIL
```

```
      ENDIF
      GOTO 14
C
C--- OUTPUT OF RESULTS IN TABULAR FORM.
C
1000  CONTINUE
      WRITE(LUN,1005)
1005  FORMAT(//,' ********************',
     2' MODULE F FREQUENCY RESPONSE ',
     3'********************')
        WRITE(LUN,1010) TITLE1,TITLE2
1010    FORMAT(//,1X,A,//,1X,A)
      IF (.NOT.WLOG) THEN
        WRITE(LUN,1020)
1020    FORMAT(//,' THE FREQUENCY SCALE IS: LINEAR')
      ELSE
        WRITE(LUN,1030)
1030    FORMAT(//,' THE FREQUENCY SCALE IS: LOG')
      ENDIF
        WRITE(LUN,1040) WOUT(1),WOUT(NPTOUT)
1040  FORMAT(/,' THE FREQUENCY RANGE IS  ',
     2 E10.3,'  TO ',E10.3,' RAD./SEC.')
        WRITE(LUN,1050) TRMIN,WTRMIN
1050  FORMAT(/,' THE MINIMUM MAGNITUDE = ',
     2 E10.3,'  AT ',E10.3,' RAD./SEC.')
        WRITE(LUN,1060) TRMAX,WTRMAX
1060  FORMAT(/,' THE MAXIMUM MAGNITUDE = ',
     2 E10.3,'  AT ',E10.3,' RAD./SEC.')
      IF (WGC.LE.0) GOTO 1075
        WRITE(LUN,1025) WGC
1025  FORMAT(/,' THE CROSSOVER FREQUENCY = ',E10.3)
        WRITE(LUN,1035) PM
1035  FORMAT(/,' THE PHASE MARGIN = ',E10.3)
1075  IF (WPHC.LE.0) GOTO 1085
        WRITE(LUN,1045) WPHC
1045  FORMAT(/,' THE FREQUENCY AT -180 PHASE SHIFT = ',E10.3)
        WRITE(LUN,1055) GM,20*ALOG10(GM)
1055  FORMAT(/,' THE GAIN MARGIN = ',E10.3,'  OR ',
     2 E10.3,' DECIBELS')
C
1085  IF (IX.EQ.1) THEN
        WRITE(1,1065)
1065    FORMAT(/,' WANT TO SEE MORE? (YES): ',$)
        IF (.NOT.YES(25,'Y')) GOTO 14
        WRITE(1,*) '   '
        NLINE = 6
      ENDIF
C
        WRITE(LUN,1070)
1070  FORMAT(//,3X,'FREQUENCY',3X,'MAGNITUDE',3X,'PHASE(DEG)',
     2 2X,'PHASE(RAD)',//)
      DO 1001 I=1,NPTOUT
        WRITE(LUN,1080) WOUT(I),MAGOUT(I),PHDOUT(I),PHOUT(I)
1080    FORMAT(4(2X,E10.3))
```

```
C
      NLINE = NLINE+1
      IF ((IX.EQ.1).AND.(NLINE.EQ.25)) THEN
        WRITE(1,1065)
        IF (.NOT.YES(25,'Y')) GOTO 14
        WRITE(1,*) '    '
        NLINE = 0
      ENDIF
C
1001  CONTINUE
      CLOSE(LUN)
      IF (IX.EQ.1) THEN
        WRITE(1,1090)
1090    FORMAT(//,' HIT <RETURN> TO RETURN TO MENU. ',$)
      READ(1,'(A3)') ANSWER
      ENDIF
C
      GOTO 14
C
C--- PLOT EITHER A BODE OR NYQUIST PLOT TO THE SCREEN
C--- AND GIVE THE OPTION OF SAVING IT IN A FILE.
C
200   CONTINUE
      CALL INITT(NRATE)
      CALL ANMODE
        IF (IX.EQ.4) CALL BPLOT
        IF (IX.EQ.5) CALL NPLOT
205   WRITE(1,2010)
2010  FORMAT(' R:  RETURN TO DISPLAY MENU(=DEF)',/,
     2       ' S:  SAVE PLOT IN A GRAPHICS FILE',/,
     3       ' ENTER OPTION (R): ',$)
      CALL MENU(AOK,IXX,'R','S','','','','
     2 ')
      IF (.NOT.AOK) GOTO 205
      IF (IXX.EQ.1) THEN
        CALL ERASE
        CALL HOME
        GOTO 14
      ENDIF
C
C--- GET GRAPHICS FILE NAME, ADD 'G_' TO START,
C--- AND CHECK FOR EXISTENCE. THEN OPEN FILE
C--- AND REDRAW THE PLOT TO SAVE IT.
C
      CALL INITT(NRATE)
      CALL ANMODE
        WRITE(1,2020)
2020    FORMAT(' THE PLOT WILL BE REDRAWN AS IT IS SAVED!')
201   WRITE(1,2030)
2030  FORMAT(//,' ENTER THE NAME FOR THE GRAPHICS FILE. ')
      CALL FNAME(FILE,EXIT)
      IF (EXIT) GOTO 14
      GFILE = 'G_'//FILE
      INQUIRE(FILE=GFILE,ERR=201,EXIST=EXST)
```

```
C
      NLINE = NLINE+1
      IF ((IX.EQ.1).AND.(NLINE.EQ.25)) THEN
        WRITE(1,1065)
        IF (.NOT.YES(25,'Y')) GOTO 14
        WRITE(1,*) '    '
        NLINE = 0
      ENDIF
C
1001  CONTINUE
      CLOSE(LUN)
      IF (IX.EQ.1) THEN
        WRITE(1,1090)
1090    FORMAT(//,' HIT <RETURN> TO RETURN TO MENU. ',$)
      READ(1,'(A3)') ANSWER
      ENDIF
C
      GOTO 14
C
C--- PLOT EITHER A BODE OR NYQUIST PLOT TO THE SCREEN
C--- AND GIVE THE OPTION OF SAVING IT IN A FILE.
C
200   CONTINUE
      CALL INITT(NRATE)
      CALL ANMODE
        IF (IX.EQ.4) CALL BPLOT
        IF (IX.EQ.5) CALL NPLOT
205   WRITE(1,2010)
2010  FORMAT(' R:  RETURN TO DISPLAY MENU(=DEF)',/,
     2       ' S:  SAVE PLOT IN A GRAPHICS FILE',/,
     3       ' ENTER OPTION (R): ',$)
      CALL MENU(AOK,IXX,'R','S','','','','
     2 '
      IF (.NOT.AOK) GOTO 205
      IF (IXX.EQ.1) THEN
        CALL ERASE
        CALL HOME
        GOTO 14
      ENDIF
C
C--- GET GRAPHICS FILE NAME, ADD 'G_' TO START,
C--- AND CHECK FOR EXISTENCE. THEN OPEN FILE
C--- AND REDRAW THE PLOT TO SAVE IT.
C
      CALL INITT(NRATE)
      CALL ANMODE
        WRITE(1,2020)
2020    FORMAT(' THE PLOT WILL BE REDRAWN AS IT IS SAVED!')
201   WRITE(1,2030)
2030  FORMAT(//,' ENTER THE NAME FOR THE GRAPHICS FILE. ')
      CALL FNAME(FILE,EXIT)
      IF (EXIT) GOTO 14
      GFILE = 'G_'//FILE
      INQUIRE(FILE=GFILE,ERR=201,EXIST=EXST)
```

```
      IF (.NOT.EXST) GOTO 2050
        WRITE(1,2040)
2040    FORMAT(//,' FILE EXISTS---WANT TO OVERWRITE? (NO): ',$)
      IF (.NOT.YES(25,'N')) GOTO 201
2050  CALL INITT(NRATE)
      CALL ANMODE
      CALL OPENTK(GFILE,IERR)
        IF (IX.EQ.4) CALL BPLOT
        IF (IX.EQ.5) CALL NPLOT
      CALL CLOSTK(IERR)
        WRITE(1,2060) GFILE
2060    FORMAT(/,' THIS PLOT IS SAVED IN: ',A17)
      WRITE(1,2070)
2070  FORMAT(' HIT <RETURN> TO RETURN TO MENU. ',$)
      READ(1,'(A3)') ANSWER
      CALL ERASE
      CALL HOME
      GOTO 14
C
98    MODFLG=.TRUE.
99    RETURN
      END
C
C
C****************************** FRESP  ******************************
C************************************************************************
C
C
C  SUBROUTINE FRESP USES THE ROOTS OF A TRANSFER FUNCTION
C  TO  COMPUTE THE FREQUENCY RESPONSE . SUBROUTINE FRESP
C  IS PASSED POLYNOMIAL ROOTS IN COMMON BLOCK ROOTBK.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE FRESP
C
C------ INPUT :
C     (1) WMIN : THE MINIMUM VALUE OF FREQUENCY USED TO COMPUTE THE
C         FREQUENCY RESPONSE .
C     (2) WMAX : THE MAXIMUM VALUE OF FREQUENCY USED TO COMPUTE THE
C         FREQUENCY RESPONSE .
C     (3) NPTS :  THE NUMBER OF POINTS THAT WILL BE USED
C
C------ OUTPUT :
C         FREQUENCY, MAGNITUDE AND PHASE SAVED IN COMMON
C         BLOCK RESULT.
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RESULT.TEXT'
      REAL XMAG(MAXDEG),DMAG(MAXDEG)
      REAL IMAG
      LOGICAL WLOG,GO
```

```
C
      WMIN=SCMIN
      WMAX=SCMAX
      WLOG=SCLOG
      IF(NPTS.GT.MAXFRP) NPTS=MAXFRP
      WRITE(1,1040) NPTS
1040  FORMAT(/'  NUMBER OF SOLUTION POINTS? (',I3,'): ',$)
      CALL GETINT(NPTS,3,MAXFRP,5)
      CALL PROCED(GO)
      IF (.NOT.GO) THEN
        MODFLG=.TRUE.
        RETURN
      ENDIF
C
      WRITE(1,1000)
1000  FORMAT(//' CALCULATING THE FREQUENCY RESPONSE.',
     2       //' ONE MOMENT, PLEASE...',/' ')
      W(1)=WMIN
      XSTEP=NPTS-1
      PI=3.141592
C
      IF (.NOT.WLOG) THEN
C
C--- COMPUTE LINEAR SCALE OMEGA VALUES
C
        DOMEGA=(WMAX-WMIN)/XSTEP
        DO 30 I=2,NPTS-1
          W(I)=W(1)+(I-1)*DOMEGA
30        CONTINUE
        W(NPTS)=WMAX
      ELSE
C
C--- COMPUTE LOG SCALE OMEGA VALUES
C
        Y=(ALOG10(WMAX)-ALOG10(WMIN))/XSTEP
        Z=10.**Y
        DO 50 I=2,NPTS-1
          W(I)=W(I-1)*Z
50        CONTINUE
        W(NPTS)=WMAX
      ENDIF
C
C--- CALC MAGNITUDE AND PHASE OF THE TRANSFER FUNCTION
C
      XMAG(1)=1.0
      DMAG(1)=1.0
      B=1.0E-10
      NLOOP1=INDEG+1
      NLOOP2=IDDEG+1
      AST=0.0
      IF (SYGAIN.LT.0.0) AST=PI
      DO 100 NI=1,NPTS
      A=AST
      IF(INDEG.EQ.0) GOTO 130
```

```
      DO 80 LI=2,NLOOP1
      XMAG(LI)=SQRT((RNR(LI-1))**2+(W(NI)-RNI(LI-1))**2)*XMAG(LI-1)
      REAL=-RNR(LI-1)
      IF(ABS(REAL).LT.B)REAL=SIGN(B,REAL)
      IMAG=W(NI)-RNI(LI-1)
      A=A+ATAN2(IMAG,REAL)
80    CONTINUE
C
130   CONTINUE
      IF(IDDEG.EQ.0) GOTO 270
      DO 90 MI=2,NLOOP2
      DMAG(MI)=SQRT((RDR(MI-1))**2+(W(NI)-RDI(MI-1))**2)*DMAG(MI-1)
      IF(DMAG(MI).LT.B)DMAG(MI)=SIGN(B,DMAG(MI))
      REAL=-RDR(MI-1)
      IF(ABS(REAL).LT.B)REAL=SIGN(B,REAL)
      IMAG=W(NI)-RDI(MI-1)
      A=A-ATAN2(IMAG,REAL)
90    CONTINUE
C
270   CONTINUE
      TRFMAG(NI)=ABS(SYGAIN*(XMAG(NLOOP1)/DMAG(NLOOP2)))
      PHI(NI)=A
      PHID(NI)=A*57.3
100   CONTINUE
C
      RETURN
      END
C
C
C****************************** PHMARG ******************************
C********************************************************************
C
C
C  DETERMINES THE PHASE MARGIN,GAIN CROSSOVER FREQUENCY,
C  GAIN MARGIN,AND FREQUENCY AT -180 PHASE SHIFT FOR
C  THE COMPUTED FREQUENCY RESPONSE.
C  MODIFIED BY J. GREGORSKI SEPTEMBER,1984
C
      SUBROUTINE PHMARG
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RESULT.TEXT'
      REAL A,B,Y,Z,PHC,GC,WPHD
C
C--- FIND PHASE MARGIN=PM (WGC=CROSSOVER FREQ,
C--- PHC=PHASE SHIFT AT WGC)
C
      Z=TRFMAG(1)
      IF (Z.EQ.1.) THEN
        WGC=W(1)
        PHC=PHID(1)
        PM=180.+PHC
      ELSE
```

```
        DO 300 I=2,NPTS
          A=Z
          Z=TRFMAG(I)
          IF (Z.EQ.1.) THEN
            WGC=W(I)
            PHC=PHID(I)
            PM=180.+PHC
            GOTO 320
          ELSEIF ((A.LT.1..AND.Z.GT.1.).OR.
     +            (A.GT.1..AND.Z.LT.1.)) THEN
            WGD=W(I)-W(I-1)
            WGC=W(I)-WGD*(Z-1.)/(Z-A)
            PHC=(PHID(I)-PHID(I-1))*(WGC-W(I))/WGD+PHID(I)
            PM=180.+PHC
            GOTO 320
          ENDIF
300       CONTINUE
        PM=0.0
        WGC=0.0
        GOTO 320
      ENDIF
320   CONTINUE
C
C---- FIND GAIN MARGIN=GM (WPHC=FREQ.AT -180 PHASE SHIFT,
C---- GC=GAIN AT -180 PHASE SHIFT)
C
      Y=PHID(1)
      IF (Y.EQ.-180.) THEN
        WGC=W(1)
        GC=TRFMAG(1)
        IF(GC.EQ.0) GOTO 405
        GM=1.0/GC
        GOTO 420
      ELSE
        DO 400 I=2,NPTS
          B=Y
          Y=PHID(I)
          IF (Y.EQ.-180.) THEN
            WPHC=W(I)
            GC=TRFMAG(I)
            IF(GC.EQ.0) GOTO 405
            GM=1.0/GC
            GOTO 420
          ELSEIF ((B.LT.-180..AND.Y.GT.-180.).OR.
     +            (B.GT.-180..AND.Y.LT.-180.)) THEN
            WPHD=W(I)-W(I-1)
            WPHC=W(I)-WPHD*(Y+180.)/(Y-B)
            GC=(TRFMAG(I)-TRFMAG(I-1))*(WPHC-W(I))/WPHD+TRFMAG(I)
            IF(GC.EQ.0) GOTO 405
            GM=1.0/GC
            GOTO 420
          ENDIF
400       CONTINUE
      ENDIF
```

```
405   GM=0.0
      VPHC=0.0
420   CONTINUE
C
C
      RETURN
      END
C
C
C****************************** TL2GET ******************************
C********************************************************************
C
C
C  ENTER OR CHANGE THE TITLE FOR THE RESULTS.
C
      SUBROUTINE TL2GET
C
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      LOGICAL YES
C
10    WRITE(1,1010) TITLE2
1010  FORMAT(/' THE RESULTS HEADING IS:'/,1X,A40//,
     2        ' CHANGE THE HEADING? (NO): ',$)
       IF (YES(13,'N')) THEN
         WRITE(1,1000)
1000     FORMAT(/' ENTER A HEADING ON ONE LINE:')
         READ(1,1005)TITLE2
1005     FORMAT(A40)
         GOTO 10
       ENDIF
       RETURN
       END
C
C
C********************************************************************
C********************************************************************
C
C
```

```
C
C
C****************************** FBLOK4 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Generates gain values for both a Log and Linear
C          scale. Computes the root locus and displays the
C          results.
C
C--- CONTENTS:
C          RLOCUS  computes the root locus
C          RDISP   displays the root locus results
C          LOGSCL  generates gains on a Log scale
C          LOGRES  computes the gain values
C          LOCMAX  finds the limits of the root locus
C
C--- INDEX:
C          LOCMAX
C          LOGRES
C          LOGSCL
C          RDISP
C          RLOCUS
C
C
C****************************** RLOCUS ******************************
C************************************************************************
C
C
C  SUBROUTINE RLOCUS USES THE ROOTS OF A TRANSFER FUNCTION
C  TO  COMPUTE THE ROOT LOCI . SUBROUTINE RLOCUS
C  IS PASSED POLYNOMIAL ROOTS IN COMMON BLOCK ROOTBK.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE RLOCUS
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RSLTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      LOGICAL KLOG,GO
      REAL COEFDK(MAXCOF),ROOTR(MAXDEG),ROOTI(MAXDEG),KG,KDEL
      REAL KMIN,KMAX
C
      KMIN=SCMIN
      KMAX=SCMAX
      KLOG=SCLOG
      IF(NPTS.GT.MAXRLP) NPTS=MAXRLP
      WRITE(1,1040) NPTS
1040  FORMAT(/'  NUMBER OF SOLUTION POINTS? (',I3,'): ',$)
      CALL GETINT(NPTS,3,MAXRLP,5)
      CALL PROCED(GO)
```

```
      IF (.NOT.GO) THEN
        MODFLG=.TRUE.
        RETURN
      ENDIF
      NND=INDEG+1
      NDD=IDDEG+1
      NDUM=INDEG+IDDEG
      WRITE(1,1000)
1000  FORMAT(//' CALCULATING THE ROOT LOCUS.',
     2       //'  ONE MOMENT, PLEASE...',/' ')
C
C--- COMPUTE THE POLE EXCESS (P-Z) AND THE ASYMPTOTE
C--- ANGLES, ANGLE(I) .
C
      IPOLEX=IDDEG-INDEG
      IF (IPOLEX.EQ.0) GOTO 610
      IF (IPOLEX.LT.0) THEN
        WRITE(1,1010)
1010    FORMAT(/' MORE ZEROS THAN POLES.',
     2         /'  NO ROOT LOCUS.')
        MODFLG=.TRUE.
        GOTO 999
      ENDIF
      DO 499 NN=1,IPOLEX
      MM=NN-1
499   ANGLE(NN)=((2*MM+1)*3.141592)/IPOLEX
C
C--- COMPUTE THE ORIGIN OF THE ASYMPTOTES:
C--- SIGMA IS THE REAL COORDINATE OF ORIGIN OF ASYMPTOTES
C
      SUMZ=0.0
      DO 601 I=1,INDEG
601   SUMZ=SUMZ+RNR(I)
      SUMP=0.0
      DO 604 I=1,IDDEG
604   SUMP=SUMP+RDR(I)
      SIGMA=(SUMP-SUMZ)/IPOLEX
C
C--- MAKE BOTH NUMERATOR AND DENOMINATOR COEFFICIENT VECTORS
C--- TO BE THE SAME LENGTH : ( COEFFN , AND COEFFD ).
C
      DO 90 I=1,IPOLEX
90    COEFFN(NND+I)=0.0
610   CONTINUE
C
C--- SET GAIN VALUES INTO KGAIN VECTOR
C
      IF (KLOG) THEN
        CALL LOGSCL(KMIN,KMAX,NPTS,KGAIN)
      ELSE
        KDEL=(KMAX-KMIN)/(NPTS-1)
        DO 110 NS=1,NPTS-1
110       KGAIN(NS)=KMIN+KDEL*(NS-1)
        KGAIN(NPTS)=KMAX
```

```
      ENDIF
C
C--- COMPUTE THE ROOT LOCUS OF THE POLYNOMIAL
C       DK(S)= D(S) + K * N(S).
C
      DO 200 NS=1,NPTS
        KG=KGAIN(NS)
        DO 210 I=1,NDD
210       COEFDK(I)=KG*GAIN*COEFFN(I)+COEFFD(I)
C
C--- BE SURE TO AVOID LEAD COEF ZERO
C
        DO 215 I=1,NDD
          VAL=ABS(COEFDK(NDD))
          IF (VAL.GT.1.E-10) GOTO 218
          DO 216 J=NDD,2,-1
216         COEFDK(J)=COEFDK(J-1)
215       COEFDK(1)=0.0
        WRITE(1,2010)KG
2010    FORMAT(//' BAD POLYNOMIAL FOR GAIN= ',1PE10.3,
     2          /' CALCULATION TERMINATED.')
        MODFLG=.TRUE.
        GOTO 999
218     CALL PROOT(IDDEG,COEFDK,ROOTR,ROOTI,1)
        DO 220 I=1,IDDEG
          RR(I,NS)=ROOTR(I)
220       RI(I,NS)=ROOTI(I)
C         IF (IDDEG.GT.2) THEN
C           WRITE(1,2020) KG
C 2020      FORMAT('  GAIN= ',1PE10.3)
C         ENDIF
200   CONTINUE
C
999   RETURN
      END
C
C
C****************************** RDISP  ******************************
C*********************************************************************
C
C
C  DISPLAYS ROOT LOCUS RESULTS IN EITHER
C  TABULAR OR GRAPHICAL FORM. TABULAR RESULTS CAN
C  BE WRITTEN TO THE SCREEN, PRINTER, OR A FILE.
C  GRAPHICS OUTPUT IS IN THE FORM OF A ROOT
C  LOCUS PLOT WHICH CAN BE SAVED IN A GRAPHICS
C  FILE FOR LATER OUTPUT TO THE PRINTER.
C  WRITTEN BY J. GREGORSKI AUGUST,1984
C
      SUBROUTINE RDISP
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RSLTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
```

```
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROUTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>TERMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      REAL K1,K2,KMIN,KMAX,IMAGMN,IMAGMX
      LOGICAL YES,OK,EXST,AOK,EXIT
      CHARACTER*3 ANSWER
      CHARACTER*15 FILE
      CHARACTER*17 GFILE
C
      IBLK=7
      KMIN=SCMIN
      KMAX=SCMAX
      KLOG=SCLOG
      K1=KMIN
      K2=KMAX
      SOUT =SIGMA
      IOUT = IPOLEX
      DO 10 I=1,IOUT
10    AOUT(I) = ANGLE(I)
14    WRITE(1,1200)
1200  FORMAT(/,' RESULT OPTIONS',/,
     2' ---------------------',/,
     3' S:  DISPLAY ON SCREEN',/,
     4' L:  LIST TO PRINTER',/,
     5' W:  WRITE TO FILE',/,
     6' G:  PLOT ROOT LOCUS')
      WRITE(1,1210)
1210  FORMAT(' R:  RETURN TO MAIN MENU',/,
     2' P:  PROCEED(=DEF)',/,
     3' ---------------------',/,
     4' ENTER OPTION (P): ',$)
      CALL MENU(OK,IX,'S','L','W','G','R','P','','',
     2 'P',25)
      IF(.NOT.OK)GO TO 14
C
C--- CHECK TO SEE IF PLOT DESIRDED AND ABLE TO DO SO.
C
      IF (((IX.EQ.4).OR.(IX.EQ.5)).AND.(.NOT.PLOTIT)) THEN
          WRITE(1,270)
270       FORMAT(/,' *** PLOTTING ROUTINES NOT COMPATIBLE',
     2    ' WITH THIS TERMINAL ***',//)
          GOTO 14
        ENDIF
C
      GO TO(100,100,100,100,98,99)IX
C
C--- GET TITLE2 FOR RESULTS
C
100   WRITE(1,1300)
1300  FORMAT(//' SET THE TITLE FOR RESULTS...')
```

```
      CALL TL2GET
C
C--- GET GAIN LIMITS FOR DISPLAY OR STORE
C
      WRITE(1,1400)
1400  FORMAT(//,' SELECT THE GAIN LIMITS',/,
     2          '  FOR DISPLAY OR PLOTTING...',/)
      WRITE(1,1410)K1
1410  FORMAT(' LOW  GAIN? (',1PE10.3,'): ',$)
      CALL GETREL(K1,KMIN,KMAX,14)
      WRITE(1,1420)K2
1420  FORMAT(' HIGH GAIN? (',1PE10.3,'): ',$)
      CALL GETREL(K2,K1,KMAX,14)
C
C--- GENERATE DISPLAY VALUES USING RLOCUS OUTPUT AND THE
C--- DESIRED GAIN LIMITS.
C
      II=0
      NOUT=0
      DO 250 I=1,NPTS
        IF(KGAIN(I).GT.K2) GOTO 260
        IF(KGAIN(I).LT.K1) GOTO 250
        II=II+1
        NOUT=NOUT+1
        RROUT(1,II) = RR(1,I)
        RIOUT(1,II) = RI(1,I)
        KOUT(II) = KGAIN(I)
      DO 210 IXX=2,IDDEG
        RROUT(IXX,II) = RR(IXX,I)
        RIOUT(IXX,II) = RI(IXX,I)
210   CONTINUE
250   CONTINUE
260   CONTINUE
C
C--- GO TO PLOTTING SECTION IF SO DESIRED.
C
      IF (IX.GT.3) GOTO 200
C
C--- DETERMINE THE LIMITS OF ROOT LOCUS DATA AND
C--- MAXIMUM LIMIT FOR ASYMPTOTE PLOTS.
C
      CALL LOCMAX(REALMN,REALMX,IMAGMX)
      IMAGMN = -IMAGMX
C
      ZMAX = ABS(REALMN)
      IF (ABS(REALMX).GT.ZMAX) THEN
        ZMAX = ABS(REALMX)
        ENDIF
      IF (ABS(IMAGMX).GT.ZMAX) THEN
        ZMAX = ABS(IMAGMX)
        ENDIF
C
C--- SETUP OUTPUT TO SCREEN,PRINTER,OR RESULTS FILE.
C
```

```
      GOTO (110,120,130) IX
C
C--- OUTPUT TO THE SCREEN.
C
110   CONTINUE
      LUN=1
      WRITE(1,1110)
1110  FORMAT(//,' HIT <RETURN> TO START DISPLAY. ',$)
      READ(1,'(A3)') ANSWER
      GOTO 1000
C
C--- OUTPUT TO THE PRINTER.
C
120   CONTINUE
      LUN=13
        WRITE(1,1220)
1220    FORMAT(//,' READY THE PRINTER',
     2          /,' AND HIT <RETURN>....',$)
      READ(1,'(A3)') ANSWER
C
      CALL DATAPR
      GOTO 1000
C
C--- OUTPUT TO A RESULTS FILE.
C
130   CONTINUE
      WRITE(1,131)
131   FORMAT(/' DO YOU WISH TO SAVE THE',/,
     2' RESULTS IN A FILE? (NO): ',$)
      IF(YES(16,'N'))THEN
        PRBFIL=.FALSE.
        LIBNAM='RESULTS*LIB'
        CALL SAVFIL
      ENDIF
      GOTO 14
C
C--- OUTPUT OF RESULTS IN TABULAR FORM.
C
1000  CONTINUE
      WRITE(LUN,1005)
1005  FORMAT(//,' **********************',
     2 ' MODULE F ROOT LOCUS DATA ',
     3 '**********************')
        WRITE(LUN,1010) TITLE1,TITLE2
1010    FORMAT(//,1X,A,//,1X,A)
      IF (.NOT.KLOG) THEN
        WRITE(LUN,1020)
1020    FORMAT(//,' THE GAIN SCALE IS: LINEAR')
      ELSE
        WRITE(LUN,1030)
1030    FORMAT(//,' THE GAIN SCALE IS: LOG')
      ENDIF
        WRITE(LUN,1040) KOUT(1),KOUT(NOUT)
1040  FORMAT(/,' THE RANGE OF GAINS IS ',
```

```
     2 E10.3,'  TO ',E10.3)
      WRITE(LUN,1045) IDDEG
1045  FORMAT(/,' THERE ARE ',I2,' POLES AT:',4X,'REAL',8X,'IMAG.')
      DO 1055 I=1,IDDEG
        WRITE(LUN,1050) RDR(I),RDI(I)
1050    FORMAT(22X,E10.3,2X,E10.3)
1055  CONTINUE
C
      WRITE(LUN,1060) INDEG
1060  FORMAT(/,' THERE ARE ',I2,' ZEROS AT:',4X,'REAL',8X,'IMAG.')
      DO 1070 I=1,INDEG
        WRITE(LUN,1065) RNR(I),RNI(I)
1065    FORMAT(22X,E10.3,2X,E10.3)
1070  CONTINUE
C
      WRITE(LUN,1075) IOUT
1075  FORMAT(/,' THE POLE EXCESS = ',I3)
      WRITE(LUN,1080) SOUT
1080  FORMAT(/,' THE REAL COORDINATE OF ASYMPTOTE ORIGIN = ',E10.3)
      WRITE(LUN,1085)
1085  FORMAT(/,' THE ASYMPTOTE ANGLES IN DEGREES:')
      DO 1105 I=1,IOUT
        WRITE(LUN,1095) AOUT(I)*57.3
1095    FORMAT(33X,E10.3)
1105  CONTINUE
C
      IF (IX.EQ.1) THEN
        WRITE(1,1100)
1100    FORMAT(/,' WANT TO SEE MORE? (YES): ',$)
        IF (.NOT.YES(25,'Y')) GOTO 14
        WRITE(1,*) '   '
        NLINE = 6
      ENDIF
C
      WRITE(LUN,1115)
1115  FORMAT(//,5X,'GAIN',6X,'REAL PART',3X,'IMAG. PART',/)
      DO 1150 N=1,NOUT
        WRITE(LUN,1120) KOUT(N),RROUT(1,N),RIOUT(1,N)
1120  FORMAT(/,3(2X,E10.3))
      DO 1140 I=2,IDDEG
        WRITE(LUN,1130) RROUT(I,N),RIOUT(I,N)
1130    FORMAT(12X,2(2X,E10.3))
1140  CONTINUE
C
      NLINE = NLINE+IDDEG+1
      IF ((IX.EQ.1).AND.(NLINE.GE.25)) THEN
        WRITE(1,1100)
        IF (.NOT.YES(25,'Y')) GOTO 14
        WRITE(1,*) '   '
        NLINE = 0
      ENDIF
C
1150  CONTINUE
      CLOSE(LUN)
```

```
      IF (IX.EQ.1) THEN
        WRITE(1,1090)
1090    FORMAT(//,' HIT <RETURN> TO RETURN TO MENU. ',$)
      READ(1,'(A3)') ANSWER
      ENDIF
C
      GOTO 14
C
C--- PLOT THE ROOT LOCUS TO THE SCREEN AND GIVE
C--- THE OPTION OF SAVING IT IN A FILE.
C
200   CONTINUE
      CALL INITT(NRATE)
      CALL ANMODE
      CALL RPLOT
205   WRITE(1,2010)
2010  FORMAT(////,' R:  RETURN TO DISPLAY MENU(=DEF)',/,
     2       ' S:  SAVE PLOT IN A GRAPHICS FILE',/,
     3       ' ENTER OPTION:   ',$)
      CALL MENU(AOK,IXX,'R','S','','','','
     2 ')
      IF (.NOT.AOK) GOTO 205
      IF (IXX.EQ.1) THEN
        CALL ERASE
        CALL HOME
        GOTO 14
      ENDIF
C
C--- GET GRAPHICS FILE NAME, ADD 'G_' TO START,
C--- AND CHECK FOR EXISTENCE. THEN OPEN FILE
C--- AND REDRAW THE PLOT TO SAVE IT.
C
      CALL INITT(NRATE)
      CALL ANMODE
        WRITE(1,2020)
2020    FORMAT(' THE PLOT WILL BE REDRAWN AS IT IS SAVED!')
201   WRITE(1,2030)
2030  FORMAT(//,' ENTER THE NAME FOR THE GRAPHICS FILE. ')
      CALL FNAME(FILE,EXIT)
      IF (EXIT) GOTO 14
      GFILE = 'G_'//FILE
      INQUIRE(FILE=GFILE,ERR=201,EXIST=EXST)
      IF (.NOT.EXST) GOTO 2050
        WRITE(1,2040)
2040    FORMAT(//,' FILE EXISTS---WANT TO OVERWRITE? (NO): ',$)
      IF (.NOT.YES(25,'N')) GOTO 201
2050  CALL INITT(NRATE)
      CALL ANMODE
      CALL OPENTK(GFILE,IERR)
      CALL RPLOT
      CALL CLOSTK(IERR)
        WRITE(1,2060) GFILE
2060    FORMAT(////,' THIS PLOT IS SAVED IN: ',A17)
      WRITE(1,2070)
```

```
2070  FORMAT(' HIT <RETURN> TO RETURN TO MENU. ',$)
      READ(1,'(A3)') ANSWER
      CALL ERASE
      CALL HOME
      GOTO 14
C
98    MODFLG=.TRUE.
99    RETURN
      END
C
C
C****************************** LOGSCL ********************************
C**********************************************************************
C
C
C  CALCULATES THE GAINS ON A LOG SCALE
C  FROM KMIN TO KMAX, STORES POINTS IN KGAIN
C  NPTS .LE.MAXRLP, PROTECTED BY PARAMETER DEFINITION
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE LOGSCL(KMIN,KMAX,NPTS,KGAIN)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      REAL KMIN,KMAX,KGAIN(MAXRLP)
      REAL GMIN,GMAX,TGAIN(MAXRLP),KTOT,GSMALL
      INTEGER NPTS,NPOS,NNEG
C
      IF (NPTS.EQ.2) THEN
        KGAIN(1)=KMIN
        KGAIN(2)=KMAX
        RETURN
      ENDIF
C
      GSMALL=1.E-5
      GMIN=KMIN
      GMAX=KMAX
      IF(GMIN.GE.0.0)THEN
        CALL LOGRES(GMIN,GMAX,NPTS,KGAIN)
      ELSEIF(GMAX.LE.0.0) THEN
        GMIN=-KMAX
        GMAX=-KMIN
        CALL LOGRES(GMIN,GMAX,NPTS,TGAIN)
        DO 10 I=1,NPTS
10        KGAIN(I)=-TGAIN(NPTS+1-I)
      ELSE
        IF (NPTS.EQ.3) THEN
          KGAIN(1)=KMIN
          KGAIN(2)=0.0
          KGAIN(3)=KMAX
          RETURN
        ENDIF
C
C--- SPLIT NPTS INTO NPOS AND NNEG VALUES
C--- ASSIGN ZERO TO NPOS INTERVAL
```

```
C--- NPOS, NNEG PROPORTIONAL TO KMAX,KMIN
C
        KTOT=KMAX-KMIN
        NPOS=(NPTS*KMAX/KTOT)+1
        NNEG=NPTS-NPOS
        IF (NNEG.EQ.0) THEN
          NPOS=NPTS-1
          NNEG=1
        ELSEIF (NPOS.EQ.1) THEN
          NPOS=2
          NNEG=NPTS-2
        ENDIF
C
C--- COMPUTE THE POS INTERVAL GAINS
C
        GMIN=0.0
        CALL LOGRES(GMIN,GMAX,NPOS,TGAIN)
        DO 20 I=1,NPOS
20        KGAIN(NNEG+I)=TGAIN(I)
C
C--- COMPUTE NEG INTERVAL GAINS
C
        IF(NNEG.EQ.1) THEN
          KGAIN(1)=KMIN
        ELSE
          GMAX=-KMIN
          GMIN=GSMALL
          CALL LOGRES(GMIN,GMAX,NNEG,TGAIN)
          DO 30 I=1,NNEG
30          KGAIN(I)=-TGAIN(NNEG-I+1)
        ENDIF
      END IF
C
C--- CORRECT FOR ROUNDOFF
C
      KGAIN(1)=KMIN
      KGAIN(NPTS)=KMAX
      RETURN
      END
C
C
C***************************** LOGRES *******************************
C*********************************************************************
C
C
C  COMPUTES THE GAIN VALUES FOR ROOT LOCUS.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE LOGRES(GMIN,GMAX,NPTS,TGAIN)
C
C       GMIN: GE ZERO
C       GMAX: GT GMIN
C       NPTS: GE 2
C       TGAIN: GAIN VALUES STORED HERE
```

```
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      REAL TGAIN(MAXRLP),GMIN,GMAX,GSMALL
      INTEGER NPTS
C
      GSMALL=1.E-5
      TGAIN(1)=GMIN
      IF (NPTS.EQ.2) THEN
        TGAIN(2)=GMAX
      ELSE
        I1=2
        IF (GMIN.EQ.0.0) THEN
          IF (GMAX.GT.GSMALL) THEN
            GMIN=GSMALL
          ELSE
            GMIN=GMAX/100.
          ENDIF
          TGAIN(2)=GMIN
          I1=3
        ENDIF
C
C--- COMPUTE THE EFFECTIVE RANGE INTERVAL
C
        XSTEP=NPTS-I1+1
        Y=(ALOG10(GMAX)-ALOG10(GMIN))/XSTEP
        Z=10.**Y
C
C--- COMPUTE GAIN VALUES
C
        DO 50 I=I1,NPTS
50        TGAIN(I)=TGAIN(I-1)*Z
      ENDIF
      RETURN
      END
C
C
C****************************** LOCMAX ******************************
C*********************************************************************
C
C
C  COMPUTES REAL,IMAG LIMITS OF LOCUS DATA,
C  POLES AND ZEROS AUTOMATICALLY INCLUDED.
C  MODIFIED BY J. GREGORSKI AUGUST,1984
C
      SUBROUTINE LOCMAX(REALMN,REALMX,IMAGMX)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROUTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      REAL IMAGMX,REALMN,REALMX
C
      REALMN=1.E15
```

```
      REALMX=-1.E15
      IMAGMX=0.0
C
C--- GET LIMITS FOR OUTPUT DATA
C
      DO 100 N=1,NOUT
        DO 90 I=1,IDDEG
          TRY=RROUT(I,N)
          IF (TRY.LT.REALMN) THEN
            REALMN=TRY
          ELSEIF (TRY.GT.REALMX) THEN
            REALMX=TRY
          ENDIF
          TRY=RIOUT(I,N)
          IF (TRY.GT.IMAGMX) THEN
            IMAGMX=TRY
          ENDIF
90        CONTINUE
100     CONTINUE
C
C--- NOW INCLUDE POLES AND ZEROS
C
      DO 120 I=1,IDDEG
        TRY=RDR(I)
        IF (TRY.LT.REALMN) THEN
          REALMN=TRY
        ELSEIF (TRY.GT.REALMX) THEN
          REALMX=TRY
        ENDIF
        TRY=RDI(I)
        IF (TRY.GT.IMAGMX) THEN
          IMAGMX=TRY
        ENDIF
120     CONTINUE
      DO 140 I=1,INDEG
        TRY=RNR(I)
        IF (TRY.LT.REALMN) THEN
          REALMN=TRY
        ELSEIF (TRY.GT.REALMX) THEN
          REALMX=TRY
        ENDIF
        TRY=RNI(I)
        IF (TRY.GT.IMAGMX) THEN
          IMAGMX=TRY
        ENDIF
140     CONTINUE
C
      RETURN
      END
C
C
C*********************************************************************
C*********************************************************************
C
```

```
C
C
C****************************** FBLOK5 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Generates the transfer function from a state
C          space problem formulation. Also contains the
C          root finder used in poly. input and root locus.
C
C--- CONTENTS:
C          GETTFN   reads A,B problem & generates a T.F. from it
C          CONMAT   converts state equ. to a transfer function
C          CHREQA   generates characteristic poly. from an A matrix
C          DET      function calculates the determinant of a matrix
C          MPY      forms a scalar poly. from a matrix poly.
C          CHREQ    calculates ADJ(SI-A)
C          PROOT    finds the roots of a polynomial
C
C--- INDEX:
C          CHREQ
C          CHREQA
C          CONMAT
C          DET(FCN)
C          GETTFN
C          MPY
C          PROOT
C
C
C****************************** GETTFN ******************************
C************************************************************************
C
C
C  GETS TRANSFER FUNCTION BY READING FROM
C  A TYPE 1 (MODT) FILE AND CONVERTING
C  THE A,B,C, AND D MATRICES.
C  USES LOCAL VARIABLES FOR INPUT OF FILE
C  DATA. USES COMMON VBLS TO SAVE THE RESULTS.
C  N.B. MATRIX, VECTOR DATA BASE FOR T DESCRIPTION
C  DIMENSIONED TO MAX. ORDER=MAXDIM.
C  TRANSFER FCN DATA BASE FOR F DESCRIPTION
C  DIMENSIONED TO MAX. ORDER=MAXDEG.
C  *** BE CAREFUL ***
C  MODIFIED BY J.GREGORSKI DECEMBER,1984
C
      SUBROUTINE GETTFN(BADFIL,BADDAT,LUN)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
C
```

```
      LOGICAL BADFIL,BADDAT
      INTEGER DIMX,DIMU,DIMY,ITYPE(MAXDIM)
      INTEGER NUMU,NUMX,NUMY
      REAL BCOL(MAXDIM),CROW(MAXDIM)
      REAL A(MAXDIM,MAXDIM),B(MAXDIM,MAXDIM)
      REAL C(MAXDIM,MAXDIM),D(MAXDIM,MAXDIM)
      REAL KGAIN,KFB(MAXDIM),EVALRP(MAXDIM),EVALIP(MAXDIM)
      REAL UDEF(MAXDIM,10),IC(MAXDIM)
C
      BADFIL=.FALSE.
      BADDAT=.FALSE.
C
C--- READ REMAINING PORTION OF FILE NEEDED FOR
C--- GENERATION OF TRANSFER FUNCTION FROM A,B DATA.
C
      READ(LUN,ERR=991,END=991)TITLE1
      READ(LUN,ERR=991,END=991)DIMX,A
      READ(LUN,ERR=991,END=991)EVALRP,EVALIP
      READ(LUN,ERR=991,END=991)FBKFLG
      READ(LUN,ERR=991,END=991)KFB,KGAIN,IC
      READ(LUN,ERR=991,END=991)DIMU,B
      READ(LUN,ERR=991,END=991)ITYPE
      READ(LUN,ERR=991,END=991)UDEF
      READ(LUN,ERR=991,END=991)DIMY,C,D
      CLOSE(LUN,STATUS='KEEP')
C
C--- CHECK TO SEE IF DIMENSION OF PROBLEM IS
C--- WITHIN THE RANGE COMPATIBLE WITH THIS SUB.
C
      IF ((DIMX.GT.MAXDIM).OR.(DIMU.GT.MAXDIM).OR.
     2 (DIMY.GT.MAXDIM)) THEN
        WRITE(1,1045)
1045    FORMAT(/' *** THIS PROBLEM IS TOO LARGE ***')
        GOTO 990
      ENDIF
C
C--- SET DATA INTO TRANSFER FUNCTION FORMAT
C
      WRITE(1,1000)
1000  FORMAT(//' CALCULATING THE TRANSFER FUNCTION',
     2        /'  FROM STATE VARIABLE DATA...')
      IF (DIMX.EQ.0) THEN
        WRITE(1,1010)
1010    FORMAT(//' *** NO STATE VARIABLES *** ')
        GOTO 990
      ENDIF
      NUMX=1
      NUMY=1
      NUMU=1
      IF (DIMU.EQ.0) THEN
        WRITE(1,1020)
1020    FORMAT(//' *** NO INPUT VARIABLES *** ')
        GOTO 990
      ELSEIF (DIMU.GT.1) THEN
```

```
        WRITE(1,1030)
1030    FORMAT(/' WHICH U TO BE USED? (1): ',$)
        CALL GETINT(NUMU,1,DIMU,14)
      ENDIF
      IF (DIMY.EQ.0) THEN
        WRITE(1,1040)
1040    FORMAT(/' WHICH X TO BE USED',
     2         /'  AS OUTPUT? (1): ',$)
        CALL GETINT(NUMX,1,DIMX,22)
        DIMY=1
        DO 110 J=1,DIMX
110       C(1,J)=0.0
          C(1,NUMX)=1.0
        DO 120 J=1,DIMU
120       D(1,J)=0.0
      ELSEIF (DIMY.GT.1) THEN
        WRITE(1,1050)
1050    FORMAT(/' WHICH Y TO BE USED? (1): ',$)
        CALL GETINT(NUMY,1,DIMY,14)
      ENDIF
C
C--- SET UP THE PROPER A,BCOL,CROW,DELEM
C
      DO 210 I=1,DIMX
        BCOL(I)=B(I,NUMU)
210     CROW(I)=C(NUMY,I)
        DELEM=D(NUMY,NUMU)
C
C--- INCLUDE FEEDBACK EFFECTS HERE
C
      IF (FBKFLG) THEN
        IF (KGAIN.EQ.0.0) THEN
          WRITE(1,2010)
2010      FORMAT(/' ZERO CONTROLLER GAIN IN THE',
     2           /'  FEEDBACK SYSTEM. ')
          GOTO 990
        ENDIF
        DELEM=KGAIN*DELEM
        DO 230 I=1,DIMX
          BCOL(I)=KGAIN*BCOL(I)
230       CROW(I)=CROW(I)-DELEM
        DO 240 I=1,DIMX
          DO 240 J=1,DIMX
240         A(I,J)=A(I,J)-BCOL(I)*KFB(J)
      ENDIF
C
C--- GET THE TRANSFER FUNCTION HERE
C--- AND STORE IN POLYBK, ROOTBK
C
      IDDEG=DIMX
      CALL CONMAT(A,BCOL,CROW,DELEM,EVALRP,EVALIP,BADDAT)
      GOTO 999
C
990   BADDAT=.TRUE.
```

```
      GOTO 999
991   BADFIL=.TRUE.
999   RETURN
      END
C
C
C****************************** CONMAT ********************************
C**********************************************************************
C
C
C  THIS SUBROUTINE CONVERTS A,BCOL,CROW,DELEM
C  TO POLYNOMIAL TRANSFER FUNCTION FORMAT.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
C     COEFFN(I),COEFFD(I) POLYNOMIAL COEFFICIENTS
C     BCOL IS THE B VECTOR.
C     CROW IS THE C VECTOR (TRANSPOSE).
C
      SUBROUTINE CONMAT(A,BCOL,CROW,DELEM,EVALRP,EVALIP,BADDAT)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>RANGBK.TEXT'
      LOGICAL BADDAT
      REAL A(MAXDIM,MAXDIM),BCOL(MAXDIM),CROW(MAXDIM)
      REAL ADJA(MAXDIM,MAXDIM,MAXDIM)
      REAL EVALRP(MAXDIM),EVALIP(MAXDIM)
C
      GAIN=1.0
      DO 5 I=1,IDDEG+1
5       COEFFN(I)=0.0
C
C--- THIS SECTION SOLVES DET(SI-A) FOR DENOM POLY
C--- THIS IS THE CHARACTERISTIC EQUATION
C
      CALL CHREQA(A,IDDEG,COEFFD,BADDAT)
      IF (BADDAT) GOTO 999
C
C--- THIS SECTION SOLVES CROW-TRANSPOSE*(ADJ(SI-A))*BCOL
C--- FOR NUMERATOR POLY
C
      CALL CHREQ(A,ADJA,IDDEG,COEFFD)
      CALL MPY(BCOL,CROW,ADJA,IDDEG,COEFFN,INDEG,BADDAT)
      IF (BADDAT) GOTO 999
C
C--- HERE WE ADD IN THE EFFECT OF D
C
      IF (DELEM.NE.0.0) THEN
        DO 20 I=1,IDDEG+1
20        COEFFN(I)=COEFFN(I)+DELEM*COEFFD(I)
        DO 25 I=IDDEG+1,1,-1
          IF (COEFFN(I).NE.0.0) GOTO 28
25        CONTINUE
```

```
        BADDAT=.TRUE.
        GOTO 999
28      INDEG=I-1
      ENDIF
C
C--- MAKE SURE COEFF. ARE WITHIN RANGE
C
      DO 50 I=1,INDEG+1
        IF(COEFFN(I).LT.BLO) COEFFN(I)=BLO
        IF(COEFFN(I).GT.BHI) COEFFN(I)=BHI
        IF(ABS(COEFFN(I)).LT.(1./BHI)) COEFFN(I)=0
50    CONTINUE
C
      DO 60 I=1,IDDEG+1
        IF(COEFFD(I).LT.BLO) COEFFD(I)=BLO
        IF(COEFFD(I).GT.BHI) COEFFD(I)=BHI
        IF(ABS(COEFFD(I)).LT.(1./BHI)) COEFFD(I)=0
60    CONTINUE
C
C--- FILL IN FACTOR-FORMATTED DATA BASE (ROOTBK)
C
      SYGAIN=GAIN*COEFFN(INDEG+1)/COEFFD(IDDEG+1)
      DO 10 I=1,IDDEG
        RDR(I)=EVALRP(I)
10      RDI(I)=EVALIP(I)
      CALL PROOT(INDEG,COEFFN,RNR,RNI,1)
999   RETURN
      END
C
C
C****************************** CHREQA ******************************
C*********************************************************************
C
C
C  SUBROUTINE CHREQA DETERMINES COEFFICIENTS OF CHAR POLY OF A.
C  SEE J.L.MELSA: "AN ALGORITHM FOR THE DESIGN OF LINEAR STATE
C  VARIABLE FEEDBACK SYSTEMS", PROC ASILOMAR CONFERENCE
C  ON SYSTEMS AND CIRCUITS,MONTEREY, CALIFORNIA ,PP. 791-799,
C  NOV. 1967.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
C  N=IDDEG, C=COEFFD, A=A MATRIX
C
      SUBROUTINE CHREQA(A,N,C,BADDAT)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      LOGICAL BADDAT
      REAL A(MAXDIM,MAXDIM),C(MAXDIM+1)
      REAL B(MAXDIM,MAXDIM),D(150)
      INTEGER N,J(MAXDIM+1)
C
      NN=N+1
      DO 20 I=1,NN
20    C(I)=0.0
```

```
      C(NN)=1.0
      DO 14 M=1,N
      K=0
      L=1
      J(1)=1
      GO TO 2
1     J(L)=J(L)+1
2     IF(L-M)3,5,990
3     MM=M-1
      DO 4 I=L,MM
      II=I+1
4     J(II)=J(I)+1
5     DO 10 I=1,M
      DO 10 KK=1,M
      NR=J(I)
      NC=J(KK)
10    B(I,KK)=A(NR,NC)
      K=K+1
      D(K)=DET(B,M)
      DO 6 I=1,M
      L=M-I+1
      IF(J(L)-(N-M+L))1,6,990
6     CONTINUE
      M1=N-M+1
      DO 14 I=1,K
14    C(M1)=C(M1)+D(I)*(-1.0)**M
      GOTO 999
990   BADDAT=.TRUE.
C
999   RETURN
      END
C
C
C****************************** DET(FCN) ******************************
C**********************************************************************
C
C
C  FUNCTION DET CALCULATES THE DETERMINANT OF A MATRIX .
C  A IS THE MATRIX AND KC IS THE ORDER OF MATRIX A .
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      FUNCTION DET(A,KC)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      REAL A(MAXDIM,MAXDIM),B(MAXDIM,MAXDIM)
C
      IREV=0
      DO 1 I=1,KC
      DO 1 J=1,KC
1     B(I,J)=A(I,J)
      DO 20 I=1,KC
      K=I
9     IF(B(K,I))10,11,10
11    K=K+1
```

```
      IF(K-KC)9,9,51
10    IF(I-K)12,14,51
12    DO 13 M=1,KC
      TEMP=B(I,M)
      B(I,M)=B(K,M)
13    B(K,M)=TEMP
      IREV=IREV+1
14    II=I+1
      IF(II.GT.KC) GO TO 20
      DO 17 M=II,KC
      IF(B(M,I))19,17,19
19    TEMP=B(M,I)/B(I,I)
      DO 16 N=I,KC
16    B(M,N)=B(M,N)-B(I,N)*TEMP
17    CONTINUE
20    CONTINUE
      DET=1.0
      DO 2 I=1,KC
2     DET=DET*B(I,I)
      DET=(-1.0)**IREV*DET
      RETURN
51    DET=0.0
      RETURN
      END
C
C
C****************************** MPY  ******************************
C*************************************************************************
C
C
C  FORMS NUMERATOR POLY (COEF) OF ORDER NORD
C  N IS INPUT AS DIM ADJA.
C  SUBROUTINE MPY FORMS A SCALAR POLYNOMIAL FROM A MATRIX
C  POLYNOMIAL . SEE 'COMPUTER PROGRAMS FOR COMPUTATIONAL
C  ASSISTANCE IN THE STUDY OF LINEAR CONTROL THEORY' , BY
C  J.L. MELSA AND S.K. JONES , SECOND EDITION , 1973 , PP 152 .
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE MPY(BCOL,CROW,ADJA,N,COEF,NORD,BADDAT)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      REAL BCOL(MAXDIM),CROW(MAXDIM),COEF(MAXDIM+1)
      REAL ADJA(MAXDIM,MAXDIM,MAXDIM)
      REAL W(MAXDIM,MAXDIM)
      LOGICAL BADDAT
      INTEGER N,NORD
C
      DO 1 I=1,N
      DO 1 J=1,N
      W(J,I)=0.0
      DO 1 K=1,N
1     W(J,I)=W(J,I)+ADJA(I,J,K)*BCOL(K)
      DO 2 I=1,N
      COEF(I)=0.0
```

```
      DO 2 J=1,N
2     COEF(I)=COEF(I)+V(J,I)*CROV(J)
      COEF(N+1)=0.0
      DO 10 I=N,1,-1
        IF (COEF(I).NE.0.0) GOTO 92
10      CONTINUE
      BADDAT=.TRUE.
92    NORD=I-1
      RETURN
      END
C
C
C****************************** CHREQ ******************************
C*******************************************************************
C
C
C  MODIFIED TO CALCULATE ADJ(SI-A) ONLY.
C  ADJA(K,I,J)= ADJ(I,J) OF S**(K-1)
C  N IS DIMENSION OF A, COEF IS CHAR POLY.
C
C  SUBROUTINE CHREQ FINDS THE COEFFICIENTS OF THE
C  CHARACTERISTIC POLYNOMIAL USING THE LEVERRIER
C  ALGORITHM . SEE 'COMPUTER PROGRAMS FOR COMPUTATIONAL
C  ASSISTANCE IN THE STUDY OF LINEAR CONTROL THEORY' ,
C  J.L. MELSA AND S.K. JONES , SECOND EDITION ,1973 PP 145 .
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      SUBROUTINE CHREQ(A,ADJA,N,COEF)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INTEGER N
      REAL A(MAXDIM,MAXDIM),ATEMP(MAXDIM,MAXDIM)
      REAL PROD(MAXDIM,MAXDIM),COEF(MAXDIM+1)
      REAL ADJA(MAXDIM,MAXDIM,MAXDIM)
C
      DO 5 I=1,MAXDIM
        DO 5 J=1,MAXDIM
5         ATEMP(I,J)=0.0
      DO 65 I=1,N
65    ATEMP(I,I)=1.0
      DO 80 I=1,N
      DO 80 J=1,N
80    ADJA(N,I,J)=ATEMP(I,J)
C
      DO 40 I=1,N
      DO 40 J=1,N
40    ATEMP(I,J)=A(I,J)
      DO 10 I=1,N
      NNN=N-I
      IF(I.EQ.1)GO TO 55
C
      NP=NNN+1
      DO 90 II=1,N
      DO 90 J=1,N
```

```
90      ADJA(NP,II,J)=ATEMP(II,J)
        DO 15 J=1,N
        DO 15 K=1,N
        PROD(J,K)=0.0
        DO 15 L=1,N
15      PROD(J,K)=PROD(J,K)+(A(J,L)*ATEMP(L,K))
        DO 13 J=1,N
        DO 13 K=1,N
13      ATEMP(J,K)=PROD(J,K)
55      DO 10 J=1,N
10      ATEMP(J,J)=ATEMP(J,J)+COEF(N-I+1)
C
        RETURN
        END
C
C
C****************************** PROOT  ******************************
C*************************************************************************
C
C
C  SUBROUTINE PROOT  USES A MODIFIED BARSTOW METHOD TO FIND THE
C  ROOTS OF A POLYNOMIAL . SEE R. W. HAMMING: NUMERICAL METHODS
C  FOR SCIENTISTS AND ENGINEERS ,   MCGRAW-HILL BOOK COMPANY ,
C  INC . , 1962 , PP. 356-359 , FOR FURTHER INFORMATION .
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
C  N=ORDER POLYNOMIAL
C  A=COEFFICIENTS POLY
C  U=REAL PARTS ROOTS
C  V=IMAG PARTS ROOTS
C  IR=MAGICAL MYSTERY VARIABLE
C
        SUBROUTINE PROOT(N,A,U,V,IR)
C
        INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
        REAL A(MAXCOF),U(MAXDEG),V(MAXDEG),H(MAXCOF),
       2 B(MAXCOF),C(MAXCOF)
C
        IREV=IR
        NC=N+1
C
        DO 1 I=1,NC
1       H(I)=A(I)
        P=0.0
        Q=0.0
        R=0.0
3       IF(H(1))4,2,4
2       NC=NC-1
        IF(NC.LE.0) GOTO 100
        V(NC)=0.0
        U(NC)=0.0
        DO 1002 I=1,NC
1002    H(I)=H(I+1)
        GO TO 3
```

```
4       IF(NC-1)5,100,5
5       IF(NC-2)7,6,7
6       R=-H(1)/H(2)
        GO TO 50
7       IF(NC-3)9,8,9
8       P=H(2)/H(3)
        Q=H(1)/H(3)
        GO TO 70
9       IF(ABS(H(NC-1)/H(NC))-ABS(H(2)/H(1)))10,19,19
10      IREV=-IREV
        M=NC/2
        DO 11 I=1,M
        NL=NC+1-I
        F=H(NL)
        H(NL)=H(I)
11      H(I)=F
        IF(Q)13,12,13
12      P=0.0
        GO TO 15
13      P=P/Q
        Q=1.0/Q
15      IF(R)16,19,16
16      R=1.0/R
19      E=4.0E-7
        B(NC)=H(NC)
        C(NC)=H(NC)
        B(NC+1)=0.0
        C(NC+1)=0.0
        NP=NC-1
20      DO 49 J=1,1000
        DO 21 I1=1,NP
        I=NC-I1
        B(I)=H(I)+R*B(I+1)
21      C(I)=B(I)+R*C(I+1)
        IF(ABS(B(1)/H(1))-E)50,50,24
24      IF(C(2))23,22,23
22      R=R+1.0
        GO TO 30
23      R=R-B(1)/C(2)
30      DO 37 I1=1,NP
        I=NC-I1
        B(I)=H(I)-P*B(I+1)-Q*B(I+2)
37      C(I)=B(I)-P*C(I+1)-Q*C(I+2)
        IF (H(2))32,31,32
31      IF(ABS(B(2)/H(1))-E)33,33,34
32      IF(ABS(B(2)/H(2))-E)33,33,34
33      IF(ABS(B(1)/H(1))-E)70,70,34
34      CBAR=C(2)-B(2)
        D=C(3)**2-CBAR*C(4)
        IF(D)36,35,36
35      P=P-2.
        Q=Q*(Q+1.0)
        GO TO 49
36      P=P+(B(2)*C(3)-B(1)*C(4))/D
```

```
      Q=Q+(-B(2)*CBAR+B(1)*C(3))/D
49    CONTINUE
      E=E*4.
      GO TO 20
50    NC=NC-1
      V(NC)=0.0
      IF(IREV)51,52,52
51    U(NC)=1.0/R
      GO TO 53
52    U(NC)=R
53    DO 54 I=1,NC
54    H(I)=B(I+1)
      GO TO 4
70    NC=NC-2
      IF(IREV)71,72,72
71    QP=1.0/Q
      PP=P/(Q*2.0)
      GO TO 73
72    QP=Q
      PP=P/2.0
73    F=(PP)**2-QP
      IF(F)74,75,75
74    U(NC+1)=-PP
      U(NC)=-PP
      V(NC+1)=SQRT(-F)
      V(NC)=-V(NC+1)
      GO TO 76
75    IF(PP)81,80,81
80    U(NC+1)=-SQRT(F)
      GO TO 82
81    U(NC+1)=-(PP/ABS(PP))*(ABS(PP)+SQRT(F))
82    CONTINUE
      V(NC+1)=0.0
      U(NC)=QP/U(NC+1)
      V(NC)=0.0
76    DO 77 I=1,NC
77    H(I)=B(I+2)
      GO TO 4
C
100   RETURN
      END
C
C
C*********************************************************************
C*********************************************************************
C
C
```

```
      Q=Q+(-B(2)*CBAR+B(1)*C(3))/D
49    CONTINUE
      E=E*4.
      GO TO 20
50    NC=NC-1
      V(NC)=0.0
      IF(IREV)51,52,52
51    U(NC)=1.0/R
      GO TO 53
52    U(NC)=R
53    DO 54 I=1,NC
54    H(I)=B(I+1)
      GO TO 4
70    NC=NC-2
      IF(IREV)71,72,72
71    QP=1.0/Q
      PP=P/(Q*2.0)
      GO TO 73
72    QP=Q
      PP=P/2.0
73    F=(PP)**2-QP
      IF(F)74,75,75
74    U(NC+1)=-PP
      U(NC)=-PP
      V(NC+1)=SQRT(-F)
      V(NC)=-V(NC+1)
      GO TO 76
75    IF(PP)81,80,81
80    U(NC+1)=-SQRT(F)
      GO TO 82
81    U(NC+1)=-(PP/ABS(PP))*(ABS(PP)+SQRT(F))
82    CONTINUE
      V(NC+1)=0.0
      U(NC)=QP/U(NC+1)
      V(NC)=0.0
76    DO 77 I=1,NC
77    H(I)=B(I+2)
      GO TO 4
C
100   RETURN
      END
C
C
C*********************************************************************
C*********************************************************************
C
C
```

```
C
C
C****************************** FBLOK6 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Performs all of the graphical output as well as
C          output to printer queue.
C
C--- CONTENTS:
C          BPLOT   generates a Bode plot
C          NPLOT   generates a Nyquist plot
C          RPLOT   generates a root locus plot
C          DATAPR  sends output directly to printer queue
C
C--- INDEX:
C          BPLOT
C          DATAPR
C          NPLOT
C          RPLOT
C
C
C****************************** BPLOT  ******************************
C************************************************************************
C
C
C  SUBROUTINE BPLOT USES A SUBSET OF THE OUTPUT FROM THE
C  SUBROUTINE FRESP TO GENERATE THE BODE PLOT.
C  WRITTEN BY J. GREGORSKI AUGUST,1984
C
C--- INPUT:
C        (1)  'FOUTBK' : A COMMON BLOCK CONTAINING A SUBSET OF
C        OUTPUT FROM FRESP. IT IS GENERATED IN FDISP AND CONTAINS
C        MAGNITUDE, PHASE, AND FREQUENCY INFORMATION FOR THE
C        SELECTED DISPLAY RANGE OF FREQUENCIES.
C
C--- OUTPUT:
C        (1) THE OUTPUT IS IN GRAPHIC FORM ONLY .
C           A) MAGNITUDE AND PHASE VERSUS FREQUENCY.
C
      SUBROUTINE BPLOT
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FOUTBK.TEXT'
      DIMENSION F(10),G(10),H(10)
      DIMENSION ISTRIN(13),JSTRIN(16),KSTRIN(9)
      REAL W(MAXFRP),M(MAXFRP),PD(MAXFRP),PR(MAXFRP)
      INTEGER K,NSTEP,N
C
      DATA ISTRIN/10,10,10,10,77,65,71,78,73,84,85,68,69/
      DATA JSTRIN/10,10,10,10,10,10,10,10,10,10,10,80,72,65,
     & 83,69/
```

```
      DATA KSTRIN/70,82,69,81,85,69,78,67,69/
C
C--- READ IN THE OUTPUT OF FRESP FOR PLOTTING
C
      NSTEP = NPTOUT
      DO 10 IK=1,NSTEP
        W(IK) = WOUT(IK)
        M(IK) = MAGOUT(IK)
        PD(IK) = PHDOUT(IK)
        PR(IK) = PHOUT(IK)
10    CONTINUE
C
      DO 600 K=1,NSTEP
        N=(ABS(PD(K))+180.)/360.
        RN=N
        PD(K)=PD(K)-360.*SIGN(RN,PD(K))
        AZ=PD(K)
600   CONTINUE
C
      DO 50 II=1,10
        I=II-1
        F(II)=180.
        G(II)=-180.
        H(II)=W(1)+((W(NSTEP)-W(1))/9.)*I
50    CONTINUE
C
      IF (.NOT.WLOG) THEN
        GOTO 20
      ELSE
        GOTO 30
      ENDIF
C
C--- THIS IS THE PLOT ROUTINE FOR LINEAR OMEGA VALUES
C
C--- FOR MAGNITUDE VERSUS FREQUENCY
C
20    CALL RECOVR
      CALL BINITT
      CALL MOVER(50,0)
      CALL SLIMX(150,850)
      CALL SLIMY(400,700)
      CALL VLABEL(13,ISTRIN)
      CALL VLABEL(16,JSTRIN)
      CALL NOTATE(450,20,9,KSTRIN)
      CALL LINE(0)
      CALL NPTS(NSTEP)
      CALL CHECK(W,M)
      CALL DSPLAY(W,M)
C
C--- FOR PHASE VERSUS FREQUENCY
C
      CALL BINITT
      CALL SLIMX(150,850)
      CALL SLIMY(100,300)
```

```
      CALL LINE(2)
      CALL DLIMY(-180.,180.)
      CALL NPTS(NSTEP)
      CALL CHECK(W,PD)
      CALL DSPLAY(W,PD)
C
C--- PUT THE 180 DEGREE LINES ON PHASE PLOT
C
      CALL LINE(0)
      CALL NPTS(10)
      CALL CHECK(H,F)
      CALL CPLOT(H,F)
      CALL LINE(0)
      CALL NPTS(10)
      CALL CHECK(H,G)
      CALL CPLOT(H,G)
      CALL MOVABS(250,755)
      CALL CHARTK(TITLE1,1.0)
      CALL MOVABS(250,730)
      CALL CHARTK(TITLE2,1.0)
      CALL HOME
      CALL ANMODE
      GO TO 40
C
C--- THIS IS THE PLOT ROUTINE FOR LOG OMEGA VALUES
C
C--- PHASE VERSUS FREQUENCY
C
30    CALL RECOVR
      CALL BINITT
      CALL MOVER(50,0)
      CALL VLABEL(13,ISTRIN)
      CALL VLABEL(16,JSTRIN)
      CALL NOTATE(450,20,9,KSTRIN)
      CALL SLIMX(150,850)
      CALL SLIMY(100,300)
      CALL LINE(2)
      CALL XTYPE(2)
      CALL XLAB(2)
      CALL YTYPE(1)
      CALL YLAB(1)
      CALL DLIMY(-180.,180.)
      CALL NPTS(NSTEP)
      CALL CHECK(W,PD)
      CALL DSPLAY(W,PD)
C
C--- PUT THE 180 DEGREE LINES ON PHASE PLOT
C
      CALL LINE(0)
      CALL NPTS(10)
      CALL CHECK(H,F)
      CALL CPLOT(H,F)
      CALL LINE(0)
      CALL NPTS(10)
```

```
      CALL CHECK(H,G)
      CALL CPLOT(H,G)
C
C--- MAGNITUDE VERSUS FREQUENCY
C
      CALL BINITT
      CALL SLIMX(150,850)
      CALL SLIMY(400,700)
      CALL LINE(0)
      CALL XTYPE(2)
      CALL XLAB(2)
      CALL YTYPE(2)
      CALL YLAB(2)
      CALL NPTS(NSTEP)
      CALL CHECK(W,M)
      CALL DSPLAY(W,M)
C
      CALL BINITT
      CALL MOVABS(250,755)
      CALL CHARTK(TITLE1,1.0)
      CALL MOVABS(250,730)
      CALL CHARTK(TITLE2,1.0)
      CALL HOME
      CALL ANMODE
40    CONTINUE
C
      RETURN
      END
C
C
C****************************** NPLOT  ******************************
C*************************************************************************
C
C
C  SUBROUTINE NPLOT USES A SUBSET OF THE OUTPUT FROM THE
C  SUBROUTINE FRESP TO GENERATE THE NYQUIST PLOT.
C  WRITTEN BY J. GREGORSKI AUGUST,1984
C
C--- INPUT:
C        (1)  'FOUTBK' : A COMMON BLOCK CONTAINING A SUBSET OF
C        OUTPUT FROM FRESP. IT IS GENERATED IN FDISP AND CONTAINS
C        MAGNITUDE, PHASE, AND FREQUENCY INFORMATION FOR THE
C        SELECTED DISPLAY RANGE OF FREQUENCIES.
C
C--- OUTPUT:
C        (1) THE OUTPUT IS IN GRAPHIC FORM ONLY .
C            A) THE NYQUIST PLOT
C
      SUBROUTINE NPLOT
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FOUTBK.TEXT'
      DIMENSION XNY(MAXFRP),YNY(MAXFRP)
```

```
      DIMENSION MSTRIN(10)
      DIMENSION NSTRIN(22)
      REAL W(MAXFRP),M(MAXFRP),PD(MAXFRP),PR(MAXFRP)
      INTEGER NSTEP
C
      DATA MSTRIN/10,10,10,10,10,10,10,73,77,65,71,73,78,65,82,89,
     & 10,10,65,88,73,83/
      DATA NSTRIN/82,69,65,76,32,32,65,88,73,83/
C
C--- REASSIGN OUTPUT VARIABLES.
C
      NSTEP = NPTOUT
      DO 20 IJ=1,NSTEP
        W(IJ) = WOUT(IJ)
        M(IJ) = MAGOUT(IJ)
        PD(IJ) = PHDOUT(IJ)
        PR(IJ) = PHOUT(IJ)
20    CONTINUE
C
      DO 80 K=1,NSTEP
        XNY(K)=M(K)*COS(PR(K))
        YNY(K)=M(K)*SIN(PR(K))
80    CONTINUE
C
      CALL RECOVR
      CALL BINITT
      CALL SLIMY(150,700)
      CALL SLIMX(100,730)
      CALL NOTATE(400,40,10,MSTRIN)
      CALL HOME
      CALL VLABEL(22,NSTRIN)
      CALL NPTS(NSTEP)
      CALL CHECK(XNY,YNY)
      CALL DSPLAY(XNY,YNY)
      CALL MOVABS(250,755)
      CALL CHARTK(TITLE1,1.0)
      CALL MOVABS(250,730)
      CALL CHARTK(TITLE2,1.0)
      CALL HOME
      CALL ANMODE
C
      RETURN
      END
C
C
C****************************** RPLOT  ******************************
C*********************************************************************
C
C
C  SUBROUTINE RPLOT PLOTS THE ROOT LOCUS OUTPUT,
C  THE CALLING PROGRAM MUST PAGE THE SCREEN BY
C  USING AN INITT CALL AND THEN RETURN TO
C  ALPHANUMERIC MODE BY USING CALL ANMODE.
C  WRITTEN BY J. GREGORSKI AUGUST,1984
```

```
C
      SUBROUTINE RPLOT
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROUTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>ROOTBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>POLYBK.TEXT'
      DIMENSION U(MAXRLP*MAXDEG),V(MAXRLP*MAXDEG),X(2),Y(2)
      CHARACTER*33 NAME1
      CHARACTER*80 NAME5
C     DIMENSION ILABEL(14),JLABEL(9)
C
C     DATA ILABEL/73,77,65,71,73,78,65,82,89,10,80,65,82,84/
C     DATA JLABEL/82,69,65,76,32,80,65,82,84/
C
C--- FILL THE NAME VECTORS USED TO LABEL THE PLOT
C
      WRITE(NAME1,100)
100   FORMAT(15X,'ROOT LOCUS PLOT ')
      WRITE(NAME5,104) KOUT(1),KOUT(NOUT)
104   FORMAT('SYSTEM GAIN IS VARIED BETWEEN',F10.3,' AND ',F10.3)
C
C--- GENERATE 1-D PLOTTING ARRAYS FROM 2-D OUTPUT ARRAYS.
C
      K=0
      DO 300 I=1,IDDEG
      DO 350 J=1,NOUT
        K=K+1
        U(K) = RROUT(I,J)
        V(K) = RIOUT(I,J)
350   CONTINUE
300   CONTINUE
      M=NOUT*IDDEG
C
C--- LABEL THE PLOT
C
      CALL RECOVR
      CALL BINITT
      CALL MOVABS(0,750)
      CALL CHARTK(NAME1,1.5)
      CALL MOVABS(150,710)
      CALL CHARTK(NAME5,.9)
      CALL MOVABS(44,375)
C     CALL VLABEL(14,ILABEL)
      CALL CHARTK('IMAG.',.75)
      CALL MOVABS(44,350)
      CALL CHARTK('AXIS',.75)
      CALL MOVABS(500,70)
C     CALL HLABEL(9,JLABEL)
      CALL CHARTK('REAL AXIS',.75)
      CALL MOVABS(250,45)
      CALL CHARTK(TITLE1,1.0)
      CALL MOVABS(250,20)
```

```
      CALL CHARTK(TITLE2,1.0)
C
C--- PLOT THE ROOT LOCUS
C
      CALL SLIMX(175,900)
      CALL SLIMY(150,626)
      CALL NPTS(M)
      CALL SIZES(.45)
      CALL SYMBL(1)
C     CALL LINE(44)
      CALL LINE(-4)
      CALL CHECK(U,V)
      CALL DSPLAY(U,V)
C
C--- PLOT THE POLES AND ZEROS
C
      CALL SIZES(1.0)
      CALL LINE(-4)
      CALL SYMBL(2)
      CALL NPTS(IDDEG)
      CALL CHECK(RDR,RDI)
      CALL CPLOT(RDR,RDI)
C
      CALL SYMBL(4)
      CALL NPTS(INDEG)
      CALL CHECK(RNR,RNI)
      CALL CPLOT(RNR,RNI)
C
C--- DRAW IN THE ASYMPTOTES
C
      DO 400 J=1,IOUT
        X(1) = SOUT
        X(2) = SOUT+ZMAX*COS(AOUT(J))
        Y(1) = 0.0
        Y(2) = ZMAX*SIN(AOUT(J))
        CALL SYMBL(0)
        CALL NPTS(2)
C       CALL LINE(1212)
        CALL LINE(3)
        CALL CPLOT(X,Y)
400   CONTINUE
      CALL HOME
      CALL ANMODE
C
      RETURN
      END
C
C
C***************************** DATAPR ******************************
C*******************************************************************
C
C
C  THIS THING IS NOT TRANSPORTABLE ANYWHERE
C  THIS SUBROUTINE ALLOWS OUTPUT TO BE WRITTEN DIRECTLY
```

```
C  TO THE PRINTER QUEUE USING PRIMOS FUNIT=9 (LUN=13).
C  THE CALLING PROGRAM WRITES THE INFO. AND THEN MUST
C  CLOSE THE UNIT.
C  RE-WRITTEN BY J. GREGORSKI AUGUST,1984
C
      SUBROUTINE DATAPR
C
      INTEGER*2 INFO(12),BUF(1024),BUFL,CODE
      BUFL=1024
      INFO(1)=10
      INFO(2)=9
      INFO(3)=0
      INFO(4)=%120240
      INFO(5)=%120240
      INFO(6)=%120240
C
      CALL SPOOL$(INTS(2),'SPOOL_FROM_MODF',15,INFO,BUF,BUFL,CODE)
      CALL ERRPR$(K$IRTN,CODE,0,0,0,0)
C
      RETURN
      END
C
C
C*********************************************************************
C*********************************************************************
C
C
```

Appendix C

MODULE C SOURCE CODE LISTINGS

```
C
C
C****************************** MODULC ********************************
C**********************************************************************
C
C
C  MAIN PROGRAM FOR TIME DOMAIN CONVER-
C  SION OF M*Z''+C*Z'+K*Z=L*F AND
C  C(I)*Y'''+ETC=D(I)*U'' ETC
C  (ORDINARY NTH ORDER DIFF EQ)
C  TO X'=A*X+B*U
C  WRITTEN BY RCR'BERG AND KDW  10-1-82
C  REWRITTEN BY J.GREGORSKI 3-25-82
C
      PROGRAM MODULC
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>VIBRBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>SGLIBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      INTEGER JUMPER
C
      CALL INITLC
C
      CALL TOPMDC
      IF (SYSFLG) THEN
        CALL CLEAVE
        IF (MODFLG) THEN
          GOTO 10
        ELSE
          GOTO 999
        ENDIF
      ENDIF
      GOTO 10
C
1000   MODFLG=.FALSE.
      CALL WHEREC(JUMPER)
      GOTO(10,20,30,40,50,60,70,80,90,100)JUMPER
C
```

```
C--- INITIALIZE
C
10    CALL ENTERC
      IF (MODFLG) GOTO 1000
C
C--- PROBLEM TITLE
C
20    CALL TTLGET
      IF (MODFLG) GOTO 1000
25    IF (SGLIFG) GOTO 50
C
C--- M.C.K. INPUT
C
30    CALL INMCK
      IF (MODFLG) GOTO 1000
C
C--- L INPUT
C
40    CALL INPUTL
      IF (MODFLG) GOTO 1000
      GOTO 60
C
C--- C(I), D(I) INPUT
C
50    CALL SGLINP
      IF (MODFLG) GOTO 1000
C
C--- LOOK AT, SPOOL, OR FILE A,B AND E.V.
C
60    CALL SHOVAB
      IF (MODFLG) GOTO 1000
      GOTO 100
C
C--- PROBLEM OR RESULTS FILE MANAGEMENT
C
70    CALL UNIDRV
      IF (MODFLG) GOTO 1000
C
C--- LEAVE BLOCK
C
100   CALL SAVPRC
      CALL CLEAVE
      IF (MODFLG) GOTO 1000
      GOTO 999
C
C--- SAVE PROBLEM DESCRIPTION ON DISK OR PRINTER
C
80    CALL SAVPRC
      GOTO 1000
C
C--- CONTINUE TO NEXT BLOCK
C
90    IF (IBLK.GE.6) THEN
        GOTO 1000
```

```
      ELSE
        IBLK=IBLK+1
        GOTO(10,20,25,40,60,100) IBLK
      ENDIF
C
999   CONTINUE
      END
C
C
C*********************************************************************
C*********************************************************************
C
C
```

```
C
C
C****************************** CBLOK1 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Set up and initialize Module C. Gets problem title
C          and selection of mode and problem entry type.
C          Contains the main menu and on-line user help.
C          Also output or filing of the O.D.E. problem or
C          the associated state space formulation.
C
C--- CONTENTS:
C          SETUPC  block data to set permanent parameters
C          INITLC  initialize parameters and data base
C          TOPMDC  on-line help for Module C
C          WHEREC  main menu display and option prompts
C          ENTERC  mode selection and problem entry
C          TTLGET  enter or change the problem title
C          SHOWAB  display or save state equation data
C          SAVPRC  write problem statement to a file or printer
C          CLEAVE  exits Module C
C
C--- INDEX:
C          CLEAVE
C          ENTERC
C          INITLC
C          SAVPRC
C          SETUPC
C          SHOWAB
C          TOPMDC
C          TTLGET
C          WHEREC
C
C
C****************************** SETUPC ******************************
C************************************************************************
C
C
C  SETS PARAMETERS FOR MODULE C WHICH ARE SPECIFIC
C  TO IT AND WHICH ARE TO REMAIN UNCHANGED.
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      BLOCK DATA SETUPC
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      DATA WANTYP(1),WANTYP(2),PRBTYP/2,3,1/
      DATA HEADER(1),HEADER(2)/2,1/
      DATA MODULE/'C'/
C
      END
C
```

```
C
C****************************** INITLC ********************************
C**********************************************************************
C
C
C  SET PARAMETERS FOR DEFAULT PROBLEM FORMULATION AS
C  WELL AS INPUT LIMITS,COMPUTATION RANGE,TITLES,ETC...
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE INITLC
C
C--- SETUP PARAMETERS FOR DEFAULT PROBLEM.
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>VIBRBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>SGLIBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      SGLIFG=.FALSE.
      DIMM=2
      MATM(1,1)=2.0
      MATM(1,2)=4.0
      MATM(2,1)=4.0
      MATM(2,2)=9.0
      INVM(1,1)=.045
      INVM(1,2)=0.0
      INVM(2,1)=-2.0
      INVM(2,2)=1.0
      MATC(1,1)=1.0
      MATC(1,2)=0.0
      MATC(2,1)=0.0
      MATC(2,2)=1.0
      MATK(1,1)=2.0
      MATK(1,2)=5.0
      MATK(2,1)=5.0
      MATK(2,2)=7.0
      COLL=2
      MATL(1,1)=1.0
      MATL(1,2)=0.0
      MATL(2,1)=0.0
      MATL(2,2)=1.0
      ORDER=3
      COEF(1)=6.0
      COEF(2)=2.0
      COEF(3)=5.0
      COEF(4)=6.0
      INNMB=1
      INPT(1)=4.0
      INPT(2)=1.0
      BLO=-1.E+15
      BHI=1.E+15
      PRBFIL=.TRUE.
```

```
      SYSFLG=.FALSE.
      MODFLG=.FALSE.
      LIBNAM='PROBLEM*LIB'
      TITLE1='*** MODULE C PROBLEM ***'
C
      RETURN
      END
C
C
C***************************** TOPMDC ******************************
C*******************************************************************
C
C
C  THIS SUBROUTINE IS DESIGNED TO
C  DESCRIBE THE MODULE TO THE USER AND
C  DETERMINE IF THE USER WISHES TO USE
C  THE MODULE
C  WRITTEN BY ERIK GOODMAN, MODIFIED BY
C  KEN WHITE AND RON ROSENBERG
C  MODIFIED BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE TOPMDC
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
C
      LOGICAL YES
      CHARACTER*1 DUM
C
      WRITE(1,1077)
1077  FORMAT(//' MODULE C IS AT YOUR SERVICE FOR',
     2/' CONVERTING DIFFERENTIAL EQUATIONS',
     3/' TO STATE EQUATION FORM...',
     4//' DO YOU WANT MORE DETAILS? (NO): ',$)
      IF (YES(7,'N')) THEN
C
C--- INTRODUCTORY TEXT
C
      WRITE(1,1000)
1000  FORMAT(//' MODULE C DESCRIPTION',
     2/' ----------------------------------',
     3/'    MODULE C CONVERTS PROBLEMS DEFINED',
     4/' BY HIGHER-ORDER DIFFERENTIAL EQUATIONS',
     5/' TO STATE-VARIABLE FORM. TWO MODES ARE',
     6/' AVAILABLE.',
     7//' (1) STANDARD VIBRATION FORM:',
     8//'    M*Z'''' + C*Z'' + K*Z = L*F')
      WRITE(1,1100) MAXDIM/2,MAXDIM
1100  FORMAT(/' WHERE Z IS POSITION VECTOR (DIM 1-',I2,'),'
     2/'       F IS FORCING VECTOR (DIM 0-',I2,'),',
     3/'       M IS MASS MATRIX.',
     4/'       C IS DAMPING MATRIX.',
     5/'       K IS STIFFNESS MATRIX.',
     6/'   AND L IS INPUT MATRIX.')
```

```
      WRITE(1,1200)
1200  FORMAT(/' HIT <RET> TO CONTINUE...')
      READ(1,1300) DUM
1300  FORMAT(A1)
C
      WRITE(1,1400) MAXDIM,MAXDIM-1
1400  FORMAT(//' (2) SINGLE HIGHER-ORDER DIFFERENTIAL',
     A/'       EQUATION (EXAMPLE):',
     2//'    C2*Y'''' +C1*Y'' +C0*Y = D1*U'' +D0*U',
     3//' WHERE Y IS OUTPUT (SCALAR),',
     4/'       U IS INPUT (SCALAR),',
     5/'   AND C AND D ARE COEFFICIENTS.',
     6//' THE HIGHEST DERIVATIVE FOR Y IS ',I2,' AND',
     7/' FOR U IS ',I2,'.',
     8/' ---------------------------------------')
      WRITE(1,1500)
1500  FORMAT(/' PROCEED IN MODULE C? (YES): ',$)
      IF (.NOT.YES(11,'Y')) SYSFLG=.TRUE.
      ENDIF
C
      RETURN
      END
C
C
C****************************** WHEREC ******************************
C************************************************************************
C
C
C  DISPLAYS MAIN MENU AND PROMPTS USER FOR
C  OPTION DESIRED.CHECKS THAT IT IS A VALID OPTION.
C  SENDS OPTION SELECTED TO MAIN AS JUMPER.
C  RE-WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE WHEREC(JUMPER)
C
C--- MAIN MODULE OPTIONS
C
      WRITE(1,111)
111   FORMAT(//'  MODULE C OPTIONS',/,
     2' ---------------------------------------',/,
     3'  1: SELECT MODE AND PROBLEM ENTRY',/,
     4'  2: ALTER THE PROBLEM TITLE',/,
     5'  3: ALTER THE M,C,K MATRICES',/,
     6'  4: ALTER THE L MATRIX',/,
     7'  5: ALTER THE DIFFERENTIAL EQUATION',/,
     8'  6: VIEW OR SAVE THE A AND B MATRICES')
      WRITE(1,912)
912   FORMAT('  7: PROBLEM OR RESULTS FILE MANAGEMENT',/,
     2'  8: SAVE PROBLEM STATEMENT AND RETURN',/,
     3'  9: CONTINUE FROM LAST ACTION (=DEF)',/,
     4' 10: LEAVE THIS MODULE',/,
     5' ---------------------------------------')
      WRITE(1,113)
113   FORMAT(' PLEASE ENTER OPTION (9): ',$)
```

```
      JUMPER=9
      CALL GETINT(JUMPER,1,10,25)
      RETURN
      END
C
C
C****************************** ENTERC ******************************
C************************************************************************
C
C
C  BLOCK FOR PERFORMING DATA RELOAD FROM FILE, IF DESIRED.
C  ALSO DETERMINES WHICH TYPE OF DIFF. EQU. FORMAT IS
C  TO BE USED: VIBS OR HIGHER ORDER FORM.
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE ENTERC
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
      LOGICAL YES
C
C--- DETERMINE TYPE OF DIFF. EQU. TO WORK WITH.
C
      IBLK=1
      WRITE(1,1010)
1010  FORMAT(//' DO YOU WANT TO WORK WITH')
100   WRITE(1,1020)
1020  FORMAT('  M,C,K MATRICES? (YES): ',$)
      IF (YES(11,'Y')) THEN
        SGLIFG=.FALSE.
        GOTO 1
      ENDIF
      WRITE(1,1030)
1030  FORMAT('  A SINGLE HIGHER-ORDER DIFF. EQU.? (YES): ',$)
      IF (YES(19,'Y')) THEN
        SGLIFG=.TRUE.
        GOTO 1
      ENDIF
      GOTO 100
C
1     MODFLG=.FALSE.
      WRITE(1,150)
150   FORMAT(//,' PROCEED ? (YES): ',$)
      IF(.NOT.YES(22,'Y')) THEN
        MODFLG=.TRUE.
        RETURN
      ENDIF
C
C--- DETERMINE METHOD OF PROBLEM ENTRY DESIRED.
C
      MODFLG=.FALSE.
200   WRITE(1,2020)
2020  FORMAT(' ENTER A NEW PROBLEM INTERACTIVELY? (YES): ',$)
      IF (YES(11,'Y')) RETURN
```

```
      WRITE(1,2030)
2030  FORMAT(' RELOAD A PREVIOUSLY SAVED PROBLEM FILE? (YES): ',$)
      IF (.NOT.YES(19,'Y')) GOTO 200
      PRBFIL=.TRUE.
      LIBNAM='PROBLEM*LIB'
      CALL LODFIL
      WRITE(1,150)
      IF (.NOT.YES(22,'Y')) MODFLG=.TRUE.
      RETURN
C
      END
C
C
C****************************** TTLGET ********************************
C**********************************************************************
C
C
C  ENTER OR CHANGE THE TITLE FOR THE PROBLEM.
C  MODIFIED BY J. GREGORSKI AUGUST,1984
C
      SUBROUTINE TTLGET
C
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
C
      LOGICAL YES
C
      IBLK=2
C
10    WRITE(1,1010)TITLE1
1010  FORMAT(//' THE PROBLEM TITLE IS:',//,1X,A40//,
     2         ' CHANGE THE TITLE? (NO): ',$)
      IF (YES(15,'N')) THEN
        WRITE(1,1020)
1020    FORMAT(//' ENTER NEW TITLE ON ONE LINE:'/)
        READ(1,1030)TITLE1
1030    FORMAT(A40)
        GOTO 10
      ENDIF
C
      WRITE(1,2000)
2000  FORMAT(//' PROCEED? (YES):',$)
      IF (YES(23,'Y')) RETURN
      MODFLG=.TRUE.
      RETURN
      END
C
C
C****************************** SHOVAB ********************************
C**********************************************************************
C
C
C  THIS SUBROUTINE WILL ALLOW THE USER TO VIEW,
C  PRINT OUT, OR SAVE THE A,B,C MATRICES AND E.V.
```

```
C  WRITTEN BY KEN WHITE OCT 10,1982
C  RE-WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE SHOWAB
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>SGLIBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>VIBRBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>EIGNBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      CHARACTER*1 MNAME,DUM
      LOGICAL YES
C
      IBLK=5
C
C--- THIS BLOCK WILL CALL OUTMAT IF THE
C--- USER WANTS TO VIEW THE A,B,C MATRICES
C
      MNAME='A'
      WRITE(1,1010) MNAME
1010  FORMAT(//' DO YOU WISH TO VIEW THE'/1X,
     +A1,' MATRIX? (YES): ',$)
      IF(YES(23,'Y'))THEN
        CALL OUTMAT(MNAME,DIMX,DIMX,A)
      ENDIF
      IF(DIMU.GT.0)THEN
        MNAME='B'
        WRITE(1,1010) MNAME
        IF(YES(23,'Y'))THEN
          CALL OUTMAT(MNAME,DIMX,DIMU,B)
        ENDIF
      END IF
      MNAME='C'
      WRITE(1,1010) MNAME
      IF (YES(23,'Y')) THEN
        CALL OUTMAT(MNAME,DIMY,DIMX,C)
      ENDIF
C
C--- OPTION TO VIEW THE EIGENVALUES
C
      WRITE(1,1110)
1110  FORMAT(//' DO YOU WISH TO VIEW THE EIGENVALUES? (YES): ',$)
      IF (YES(17,'Y')) THEN
        WRITE(1,1100)
1100    FORMAT(3X,/' THE EIGENVALUES:')
        WRITE(1,2000)
2000    FORMAT(3X,/' VALUE',4X,'REAL',6X,'IMAGINARY')
        DO 70 I=1,DIMX
          WRITE(1,2020) I,EVALR(I),EVALI(I)
2020      FORMAT(1X,I3,2X,1PE12.5,1X,E12.5)
70      CONTINUE
      ENDIF
```

```
C
C--- OPTION TO OUTPUT A,B,C AND E.V. TO PRINTER
C
      WRITE(1,1000)
1000  FORMAT(//' WANT TO PRINT THE A,B,C RESULTS AND',
     2/' EIGENVALUES ON THE PRINTER? (NO): ',$)
      IF (YES(17,'N')) THEN
        WRITE(1,1020)
1020    FORMAT(/' READY THE PRINTER',
     2         /'  AND HIT <RETURN>...',$)
        READ(1,'(A1)') DUM
        CALL PRINAB
      ENDIF
C
C--- OPTION TO SAVE A,B,C AND E.V. IN A FILE
C
      WRITE(1,101)
101   FORMAT(/' DO YOU WISH TO SAVE THE A,B,C MATRICES',/,
     2' AND EIGENVALUES IN A FILE? (NO): ',$)
      IF(YES(16,'N'))THEN
        PRBFIL=.FALSE.
        LIBNAM='PROBLEM*LIB'
        CALL SAVFIL
      ENDIF
C
      RETURN
      END
C
C
C****************************** SAVPRC ******************************
C*********************************************************************
C
C
C  IT QUERIES THE USER, AND WRITES INFORMATION TO A PRINTER
C  OR FILE IF REQUESTED, THEN RETURNS TO CALLING PROGRAM
C  FOR CONTINUATION OR EXIT.
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE SAVPRC
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
      LOGICAL YES
      CHARACTER*3 ANSWER
C
      WRITE(1,1000)
1000  FORMAT(//' DO YOU WANT TO LIST',
     2        /'  THE CURRENT PROBLEM STATEMENT',
     3        /'   ON THE PRINTER? (NO): ',$)
      IF (YES(14,'N')) THEN
        WRITE(1,1010)
1010    FORMAT(//' READY THE PRINTER',
     2          /'  AND HIT <RETURN>...',$)
        READ(1,'(A3)') ANSWER
        CALL PRINYM
```

```
      ENDIF
C
      WRITE(1,101)
101   FORMAT(/' DO YOU WISH TO SAVE THE CURRENT',/,
     2' PROBLEM STATEMENT IN A FILE? (NO): ',$)
      IF(YES(16,'N'))THEN
        PRBFIL=.TRUE.
        LIBNAM='PROBLEM*LIB'
        CALL SAVFIL
      ENDIF
C
      RETURN
      END
C
C
C****************************** CLEAVE ******************************
C********************************************************************
C
C
C  ASKS IF USER WANTS TO EXIT MODULE C.
C  IF YES THEN IT GIVES A MESSAGE AND EXITS.
C  IF NO THEN RETURN TO MAIN MENU.
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE CLEAVE
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      LOGICAL YES
C
      IBLK=6
      WRITE(1,100)
100   FORMAT(//' WANT TO LEAVE THIS MODULE? (YES): ',$)
      IF (YES(5,'Y')) THEN
        WRITE(1,110)
110     FORMAT(//'     SEE YOU LATER...'/
     8' *** EXITED MODULE C ***')
        MODFLG=.FALSE.
      ELSE
        MODFLG=.TRUE.
      ENDIF
      RETURN
      END
C
C
C********************************************************************
C********************************************************************
C
C
```

```
C
C
C****************************** CBLOK2 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Interactive input of either a Vibs. form or a single
C          higher order differential equ. Conversion to state
C          space formulation occurs automatically.
C
C--- CONTENTS:
C          INMCK   inputs the M,C,K matrices , forms A & C
C          INPUTL  inputs the L matrix , forms B & D
C          SGLINP  inputs a single O.D.E. , forms A,B,C,D
C
C--- INDEX:
C          INMCK
C          INPUTL
C          SGLINP
C
C
C****************************** INMCK  ******************************
C************************************************************************
C
C
C  THIS SUBROUTINE IS DESIGNED TO INPUT
C  THE M,C,K MATRICES NEEDED TO SOLVE
C  VIBRATORY SYSTEMS IN THE TIME DOMAIN
C  WRITTEN BY KEN WHITE, AUG 2,1982
C  MODIFIED BY J. GREGORSKI MARCH,1985
C
C  INVMK   PRODUCT OF INVM AND THE K MATRIX
C  INVMC      ''       ''          C   ''
C  CPOLY   COEFFICIENTS FOR CHARACTERISTIC POLYNOMIAL
C  SIZE    THE DIMENSION OF THE M,C,K MATRICES
C
      SUBROUTINE INMCK
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>VIBRBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>EIGNBK.TEXT'
C
      LOGICAL FLAGM,GO,YES,BADDAT
      CHARACTER*1 MNAME
      INTEGER RI,CI,LR,EC,FLAG,SIZE,I,ILOW,IHIGH
      REAL XDOTT(MAXDIM+1),TEMPX(MAXDIM+1),INVMC(MAXDIM,MAXDIM),
     + INVMK(MAXDIM,MAXDIM),CPOLY(MAXCOF)
C
C--- CHECK TO SEE IF IN PROPER PROBLEM MODE
C
      IF (SGLIFG) THEN
        WRITE(1,110)
```

```
110     FORMAT(//'         *** WRONG INPUT MODE ***'/,
     +  ' CANNOT ALTER VIBS. FORM WHEN IN SINGLE D.E. MODE!')
        MODFLG=.TRUE.
        RETURN
       ENDIF
C
       IBLK=3
C
C--- PROMPT FOR MATRIX DIMENSION
C
       ILOW=1
       IHIGH=MAXDIM/2.
       SIZE=DIMM
11     WRITE(1,100)DIMM
100    FORMAT(/,' ENTER THE DIMENSION OF Z (',I1,'): ',$)
       CALL GETINT(SIZE,ILOW,IHIGH,10)
C
C--- IF THE MATRIX SIZE IS INCREASED, THEN
C--- DEFAULT VALUES ARE LOADED INTO THE
C--- NEW ELEMENTS
C
       IF(SIZE.GT.DIMM)THEN
         DO 60 RI=1,SIZE
           DO 60 CI=DIMM+1,SIZE
             MATM(RI,CI)=0.
             MATC(RI,CI)=0.
60           MATK(RI,CI)=0.
         DO 70 RI=DIMM+1,SIZE
           DO 75 CI=1,DIMM
             MATM(RI,CI)=0.
             MATC(RI,CI)=0.
             MATK(RI,CI)=0.
             MATL(RI,CI)=0.
75         CONTINUE
70       CONTINUE
         DO 80 I=DIMM+1,SIZE
           MATM(I,I)=1.0
           MATC(I,I)=1.0
80         MATK(I,I)=1.0
       ENDIF
C
C--- STORE THE CURRENT MATRIX DIMENSIONS
C--- INTO THE PROPER COMMON BLOCK VAR
C
       DIMM=SIZE
C
C--- NOW LOAD THE MATRICES
C--- THE MATRICES WILL BE FILLED ONE
C--- AT A TIME
C
       DO 10 I=1,3
         IF(I .EQ. 1) MNAME='M'
         IF(I .EQ. 2) MNAME='C'
         IF(I .EQ. 3) MNAME='K'
```

```
C
C--- INPUTM ALLOWS THE USER TO LOAD THE
C--- MATRICES WITH DATA
C
        IF(I.EQ.1)THEN
          CALL INPUTM(MNAME,DIMM,DIMM,MATM)
C
C--- DETERMINE IF M IS SINGULAR. IF NOT,
C--- GET ITS INVERSE
C
          CALL SIMEQ(MATM,XDOTT,DIMM,INVM,TEMPX,FLAG)
          FLAGM=.FALSE.
          IF(FLAG.EQ.0)THEN
            WRITE(1,1010)
1010        FORMAT(/' THE M MATRIX IS SINGULAR. DO YOU',
     2             /'  WANT TO CORRECT IT? (YES):',$)
            IF(YES(11,'Y'))THEN
              FLAGM=.TRUE.
            ELSE
              MODFLG=.TRUE.
            END IF
          ENDIF
        END IF
C
C--- IF INVM IS OKAY, LOAD THE K,C MATRICES
C
        IF((.NOT.FLAGM).AND.(.NOT.MODFLG))THEN
          IF(I.EQ.2)CALL INPUTM(MNAME,DIMM,DIMM,MATC)
          IF(I.EQ.3)CALL INPUTM(MNAME,DIMM,DIMM,MATK)
        END IF
10     CONTINUE
C
       IF(FLAGM)GOTO 11
       IF(.NOT.MODFLG)THEN
C
C--- NOW MULTIPLY K,C BY INVM
C
        CALL MMULT(INVM,MATC,INVMC,DIMM,DIMM,DIMM)
        CALL MMULT(INVM,MATK,INVMK,DIMM,DIMM,DIMM)
C
C--- FORM THE NEGATIVE OF INVMC AND INVMK
C
        DO 1 RI=1,DIMM
          DO 2 CI=1,DIMM
            INVMC(RI,CI)=-INVMC(RI,CI)
            INVMK(RI,CI)=-INVMK(RI,CI)
2         CONTINUE
1       CONTINUE
C
C--- NOW FORM THE A MATRIX. ITS DIMENSIONS
C--- WILL BE TWICE THAT OF THE M MATRIX.
C--- IT WILL BE FORMED FROM 4 SEPARATE
C--- MATRICES.
C
```

```
        DO 3 RI=1,DIMM
          LR=DIMM+RI
          DO 4 CI=1,DIMM
            EC=DIMM+CI
C
C--- THIS FILLS THE UPPER LEFT QUADRANT
C
            A(RI,CI)=0.
C
C--- THIS FILLS THE UPPER RIGHT QUADRANT
C
            IF(CI.EQ.RI)THEN
              A(RI,EC)=1.
            ELSE
              A(RI,EC)=0.
            END IF
C
C--- THIS FILLS THE LOWER LEFT QUADRANT
C
            A(LR,CI)=INVMK(RI,CI)
C
C--- THIS FILLS THE LOWER RIGHT QUADRANT
C
            A(LR,EC)=INVMC(RI,CI)
4         CONTINUE
3       CONTINUE
C
        DIMX=DIMM*2
        DIMY=DIMM
C
C--- FIND THE EIGENVALUES FOR THE A MATRIX
C
        BADDAT=.FALSE.
        CALL CHREQA(A,DIMX,CPOLY,BADDAT)
          IF (BADDAT) THEN
            WRITE(1,222)
222         FORMAT(/' BAD INPUT DATA. UNABLE TO CALCULATE THE',
     +       /' EIGENVALUES. PLEASE RE-ENTER THE PROBLEM!')
            GOTO 11
          ENDIF
        CALL PROOT(DIMX,CPOLY,EVALR,EVALI,1)
        CALL SORT(EVALR,EVALI,DIMX)
C
C---  STORE THE C MATRIX
C
        DO 500 RI=1,DIMY
          DO 499 CI=1,DIMX
499         C(RI,CI)=0.
500       C(RI,RI)=1.
C
C--- PREPARE TO LEAVE
C
        CALL PROCED(GO)
        IF(.NOT.GO)MODFLG=.TRUE.
```

```
      END IF
C
      RETURN
      END
C
C
C****************************** INPUTL ******************************
C********************************************************************
C
C
C  THIS SUBROUTINE WILL INPUT THE MATRIX
C  FOR THE VIBRATION SYSTEM FORCING VECTOR
C  WRITTEN BY KEN WHITE  OCT 12,1982
C  MODIFIED BY J. GREGORSKI MARCH,1985
C
C  SIZEL    NUMBER OF INPUTS
C  INVML    PRODUCT OF INVM AND MATL
C
      SUBROUTINE INPUTL
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>VIBRBK.TEXT'
C
      CHARACTER*1 MNAME
      LOGICAL GO
      INTEGER RI,LR,CI,IHIGH,ILOW,SIZEL
      REAL INVML(MAXDIM,MAXDIM)
C
C--- CHECK TO SEE IF IN PROPER PROBLEM MODE
C
      IF (SGLIFG) THEN
        WRITE(1,110)
110     FORMAT(//'         *** WRONG INPUT MODE ***'/,
     +  ' CANNOT ALTER VIBS. FORM WHEN IN SINGLE D.E. MODE!')
        MODFLG=.TRUE.
        RETURN
      ENDIF
C
      IBLK=4
C
C--- GET THE COLUMN SIZE FOR MATL
C
      ILOW=0
      IHIGH=MAXDIM
      SIZEL=COLL
      MNAME='L'
      WRITE(1,200)COLL
200   FORMAT(/,' ENTER THE DIMENSION OF F (',I1,'): ',$)
      CALL GETINT(SIZEL,ILOW,IHIGH,10)
      IF (SIZEL.EQ.0) GOTO 900
C
C--- LOAD DEFAULT VALUES INTO MATL IF THE
C--- SIZE IS INCREASED
```

```
C
      IF(SIZEL.GT.COLL)THEN
      DO 80 RI=1,DIMM
        DO 85 CI=COLL+1,SIZEL
          MATL(RI,CI)=0.
85      CONTINUE
80      CONTINUE
      END IF
C
C--- PUT MATL DIMENSIONS IN COMMON BLOCK
C
      COLL=SIZEL
C
C--- ALLOW THE USER TO LOAD MATL. FIRST,
C--- MAKE SURE THAT MATL IS NON-ZERO.
C
      DIMU=COLL
      IF(COLL.GT.0)THEN
      CALL INPUTM(MNAME,DIMM,SIZEL,MATL)
C
C--- MULTIPLY BY INVERSE M
C
      CALL MMULT(INVM,MATL,INVML,DIMM,DIMM,COLL)
C
C--- FORM THE B MATRIX. IT IS FORMED FROM
C--- TWO MATRICES
C
      DO 5 RI=1,DIMM
        LR=DIMM+RI
        DO 6 CI=1,COLL
C
C--- FIRST, FILL THE UPPER HALF
C
          B(RI,CI)=0.
C
C--- NOW FILL THE LOWER HALF
C
          B(LR,CI)=INVML(RI,CI)
6       CONTINUE
5     CONTINUE
      END IF
C
C--- SET THE D MATRIX
C
      DO 300 RI=1,DIMY
      DO 300 CI=1,DIMU
300     D(RI,CI)=0.
C
C--- READY TO EXIT
C
900   CALL PROCED(GO)
      IF(.NOT.GO)MODFLG=.TRUE.
C
      RETURN
```

```
      END
C
C
C****************************** SGLINP ********************************
C**********************************************************************
C
C
C  THIS SUBROUTINE INPUTS A SINGLE DIF-
C  FERENTIAL EQUATION, THEN FORMS THE A
C  AND B MATRIX FROM THIS EQUATION.
C  X1=Y BY DEFINITION; B IS A COLUMN VECTOR
C
C  WRITTEN BY KEN WHITE--AUG 10,1982
C  MODIFIED BY RCR'BERG--NOV 30,1982
C  MODIFIED BY J. GREGORSKI MARCH,1985
C
C  TEMP      COMPARES NEW VALUES OF ORDER AND INNMB WITH OLD
C  CPOLY     COEFFICIENTS OF THE CHARACTERISTIC POLYNOMIAL
C
      SUBROUTINE SGLINP
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>SGLIBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>EIGNBK.TEXT'
C
      INTEGER II,RI,CI,I,TEMP,ILESS1,IHIGH,ILOW
      REAL RESULT,INVLC,EXPN,CPOLY(MAXCOF)
      LOGICAL OK,GOOD,CHNGCF,YES,BADDAT
C
C--- CHECK TO SEE IF IN PROPER PROBLEM MODE
C
      IF (.NOT.SGLIFG) THEN
        WRITE(1,110)
110     FORMAT(//'         *** WRONG INPUT MODE ***'/,
     +  ' CANNOT ALTER SINGLE D.E. FORM WHEN IN VIBS. MODE!')
        MODFLG=.TRUE.
        RETURN
      ENDIF
C
      IBLK=4
C
C--- PROMPT FOR THE ORDER OF THE EQUATION
C
      ILOW=1
      IHIGH=MAXDIM
      TEMP=ORDER
11    WRITE(1,504)'Y',ORDER
      IF(YES(14,'N'))THEN
        WRITE(1,502)ORDER
        CALL GETINT(ORDER,ILOW,IHIGH,12)
C
C--- IF THE ORDER IS INCREASED, FILL THE
C--- NEW COEFS WITH DEFAULT VALUES
```

```
C
        IF(ORDER.GT.TEMP)THEN
          DO 45 I=TEMP+2,ORDER+1
            COEF(I)=1
45        CONTINUE
        ENDIF
      ENDIF
C
C--- PROMPT FOR THE COEFFICIENTS. FIRST,
C--- ALLOW THE USER TO VIEW THEM
C
      WRITE(1,511)'Y'
      IF(.NOT.YES(16,'Y'))GOTO 27
77    DO 80 I=ORDER+1,1,-1
        WRITE(1,512)COEF(I),'Y'
        DO 81 II=I-1,1,-1
          WRITE(1,513)
81      CONTINUE
80    CONTINUE
C
C--- ALLOW THE USER TO CHANGE THE VALUES
C
27    CHNGCF=.FALSE.
      WRITE(1,503)'Y'
      IF(YES(16,'N'))CHNGCF=.TRUE.
29    CONTINUE
C
      IF (CHNGCF) THEN
28      VAL=COEF(ORDER+1)
        WRITE(1,100)'C',ORDER,VAL
        CALL GETREL(VAL,BLO,BHI,23)
        IF (VAL.EQ.0) THEN
          WRITE(1,200)
200       FORMAT(/' *** THE LEADING COEFFICIENT',
     2/' MUST NOT BE ZERO.')
          GOTO 28
        ELSE
          COEF(ORDER+1)=VAL
        ENDIF
      ENDIF
      INVLC=1./COEF(ORDER+1)
C
      DO 10 I=ORDER,1,-1
        VAL=COEF(I)
        ILESS1=I-1
        IF(CHNGCF)THEN
          WRITE(1,100)'C',ILESS1,VAL
          CALL GETREL(VAL,BLO,BHI,23)
        END IF
C
C--- MAKE SURE THAT THE COEFFICIENT WILL
C--- REMAIN IN RANGE WHEN DIVIDED BY THE
C--- LEADING COEFFICIENT
C
```

```
        CALL MLTYCK(INVLC,VAL,GOOD,RESULT,EXPN)
C
C--- LOOP BACK TO CHANGE BAD DATA
C
        IF(.NOT.GOOD)THEN
          WRITE(1,1000) VAL,ORDER,'C',ILESS1
1000      FORMAT(/' *** THIS COEFFICIENT  (',1PE10.3,')',
     2    /' LEADS TO A NUMBER OUT OF RANGE.',
     3    /' MAKE C(',I2,') LARGER OR MAKE ',A1,'(',I2,')',
     4    /' SMALLER IN MAGNITUDE.')
            CHNGCF=.TRUE.
            GOTO 29
          ENDIF
          COEF(I)=VAL
C
C--- THE FIRST COL OF A IS LOADED HERE TO TAKE
C--- ADVANTAGE OF THE RESULT RETURNED BY MLTYCK
C
          A(ORDER+1-I,1)=-RESULT
10      CONTINUE
        IF (CHNGCF) THEN
          WRITE(1,511)'Y'
          IF (YES(16,'Y'))GOTO 77
        ENDIF
C
C--- PROMPT FOR THE ORDER  OF THE INPUT
C
        ILOW=0
        IHIGH=ORDER-1
        IF (INNMB.GT.IHIGH) INNMB=ILOW
        TEMP=INNMB
        WRITE(1,504)'U',INNMB
        IF(YES(14,'N'))THEN
          WRITE(1,103)INNMB
          CALL GETINT(INNMB,ILOW,IHIGH,15)
C
C--- LOAD DEFAULT VALUES INTO INPT IF THE
C--- THE NUMBER OF INPUTS IS INCREASED
C
          IF(INNMB.GT.TEMP)THEN
            DO 50 I=TEMP+1,INNMB+1
              INPT(I)=1.
50          CONTINUE
          ENDIF
        ENDIF
C
C--- CHECK IF INPUTS ARE TO BE CHANGED.
C--- FIRST ALLOW THE USER THE VIEW THEM
C
        WRITE(1,511)'U'
        IF(.NOT.YES(16,'Y'))GOTO 79
78      DO 82 I=INNMB+1,1,-1
          WRITE(1,512)INPT(I),'U'
          DO 83 II=I-1,1,-1
```

```
          WRITE(1,513)
83        CONTINUE
82      CONTINUE
C
C--- ALLOW THE USER TO CHANGE THE VALUES
C
79      CHNGCF=.FALSE.
        WRITE(1,503)'U'
        IF(YES(20,'N'))CHNGCF=.TRUE.
C
C--- ENTER NEW INPUTS
C
        DO 15 I=INNMB+1,1,-1
          VAL=INPT(I)
          ILESS1=I-1
          IF(CHNGCF)THEN
            WRITE(1,100)'D',ILESS1,VAL
            CALL GETREL(VAL,BLO,BHI,23)
          END IF
C
C--- MAKE SURE INPUTS REMAIN IN RANGE WHEN
C--- DIVIDED BY THE LEADING COEFFICIENT
C
          CALL MLTYCK(INVLC,VAL,GOOD,RESULT,EXPN)
C
C--- LOOP BACK TO CHANGE BAD DATA
C
          IF (.NOT.GOOD)THEN
            WRITE(1,1000) VAL,ORDER,'D',ILESS1
            CHNGCF=.TRUE.
            GOTO 29
          ENDIF
          INPT(I)=VAL
C
C--- SET NONZERO PART OF B VECTOR HERE
C
          B(ORDER+1-I,1)=RESULT
C
15      CONTINUE
        IF (CHNGCF)THEN
          WRITE(1,511)'U'
          IF (YES(16,'Y'))GOTO 78
        ENDIF
C
C--- NOW FORM THE REST OF THE A MATRIX
C
        DO 25 RI=1,ORDER
          DO 30 CI=2,ORDER
            IF(CI.EQ.RI+1)THEN
              A(RI,CI)=1.
            ELSE
              A(RI,CI)=0.
            END IF
30        CONTINUE
```

```
25    CONTINUE
C
C--- NOW FORM THE REST OF THE B VECTOR
C
      DO 35 I=1,ORDER-INNMB-1
        B(I,1)=0.
35    CONTINUE
C
C--- STORE DIMENSIONS OF A, B AND SET C,D.
C
      DIMX=ORDER
      DIMU=1
      DIMY=1
      C(1,1)=1.
      DO 40 I=2,DIMX
40      C(1,I)=0.
      D(1,1)=0.
C
C--- FIND THE EIGENVALUES FOR THE A MATRIX
C
      BADDAT=.FALSE.
      CALL CHREQA(A,DIMX,CPOLY,BADDAT)
        IF (BADDAT) THEN
          WRITE(1,222)
          FORMAT(/' BAD INPUT DATA. UNABLE TO CALCULATE THE',
     +       /' EIGENVALUES. PLEASE RE-ENTER THE PROBLEM!')
          GOTO 11
        ENDIF
      CALL PROOT(DIMX,CPOLY,EVALR,EVALI,1)
      CALL SORT(EVALR,EVALI,DIMX)
C
C--- FORMAT STATEMENTS
C
100   FORMAT(/,' ENTER ',A1,'(',I1,'). ITS CURRENT VALUE',/,
     2' IS(',1PE10.3,'): ',$)
103   FORMAT(/,' ORDER OF THE INPUT? (',I1,'): ',$)
502   FORMAT(/,' ORDER OF THE EQUATION? (',I1,'): ',$)
503   FORMAT(//,' CHANGE THE ',A1,' COEFFICIENTS? (NO): ',$)
504   FORMAT(/,' THE HIGHEST ',A1,' DERIVATIVE IS ',I1,'.',/,
     +' WANT TO CHANGE THE ORDER? (NO): ',$)
511   FORMAT(/,' DO YOU WANT TO SEE THE ',/,1X,A1,
     +' COEFFICIENTS? (YES): ',$)
512   FORMAT(/,6X,'(',1PE10.3,')*',A1,$)
513   FORMAT('''',$)
C
C--- PREPARE TO EXIT
C
      CALL PROCED(OK)
      IF(.NOT.OK)MODFLG=.TRUE.
C
      RETURN
      END
C
C
```

```
C**********************************************************************
C**********************************************************************
C
C
```

```
C**********************************************************************
C**********************************************************************
C
C
```

```
C
C
C****************************** CBLOK3 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Reads and writes problem and results files. Also
C          display or printing of O.D.E. or state equ. problem
C          and matrix input subs.
C
C--- CONTENTS:
C          CPROB    reads or writes problem files
C          CRSLT    writes the state equation file
C          PRINYM   writes the problem statement to the printer
C          PRMTX    outputs any matrix
C          PRINAB   writes the state equations to the printer
C          DATAPR   sends the output directly to the printer queue
C          INPUTM   input any matrix interactively
C          ELGET    gets individual array elements
C          OUTMAT   displays any matrix
C
C--- INDEX:
C          CPROB
C          CRSLT
C          DATAPR
C          ELGET
C          INPUTM
C          OUTMAT
C          PRINAB
C          PRINYM
C          PRMTX
C
C
C****************************** CPROB  ******************************
C************************************************************************
C
C
C  READS OR WRITES PROBLEM FILES FOR MODULE C.
C  CAN READ FILES CREATED BY MODULE C ONLY.
C  CALLED BY SAVFIL OR LODFIL IN UNIFIL.FTN
C  WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE CPROB(FILNAM,READIN,ERROR)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>SGLIBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>VIBRBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>EIGNBK.TEXT'
C
      CHARACTER*15 FILNAM
      INTEGER IMOD
```

```
      LOGICAL READIN,ERROR
C
      LUN=5
      ERROR=.FALSE.
C
C---- CREATE NEW FILE OR READ OLD PROBLEM FILE.
C
      IF (.NOT.READIN) THEN
        OPEN(LUN,FILE=FILNAM,ERR=991,STATUS='UNKNOWN',
     2  FORM='UNFORMATTED')
C
C---- BEGIN WRITING SINGLE D.E. PROBLEM TO NEW FILE.
C
        IF (SGLIFG) THEN
          IMOD=3
          WRITE(LUN,ERR=991)IMOD
          WRITE(LUN,ERR=991)TITLE1
          WRITE(LUN,ERR=991)COEF,INPT,ORDER,INNMB
          WRITE(LUN,ERR=991)A,B,DIMX,DIMU
          WRITE(LUN,ERR=991)EVALR,EVALI
          ENDFILE(LUN)
C
C---- BEGIN WRITING VIBS. FORM PROBLEM TO NEW FILE.
C
        ELSE
          IMOD=2
          WRITE(LUN,ERR=991)IMOD
          WRITE(LUN,ERR=991)TITLE1
          WRITE(LUN,ERR=991)MATM,MATC,MATK,MATL,INVM
          WRITE(LUN,ERR=991)DIMM,COLL
          WRITE(LUN,ERR=991)A,B,DIMX,DIMU
          WRITE(LUN,ERR=991)EVALR,EVALI
          ENDFILE(LUN)
        ENDIF
C
C---- ZERO THE C AND D MATRICES
C
      ELSE
        DO 100 I=1,MAXDIM
          DO 100 J=1,MAXDIM
            C(I,J)=0.
100         D(I,J)=0.
C
C---- BEGIN READING INFORMATION FROM FILE SPECIFIED
C
        OPEN(LUN,FILE=FILNAM,ERR=991,STATUS='OLD',
     2  FORM='UNFORMATTED')
        READ(LUN,END=991,ERR=991)IMOD
        READ(LUN,END=991,ERR=991)TITLE1
C
C---- READ SINGLE D.E. TYPE PROBLEM FILE
C
        IF(IMOD.EQ.3)THEN
          READ(LUN,END=991,ERR=991)COEF,INPT,ORDER,INNMB
```

```
          READ(LUN,END=991,ERR=991)A,B,DIMX,DIMU
          READ(LUN,END=991,ERR=991)EVALR,EVALI
          DIMY=1
          C(1,1)=1.
C
C---- READ VIBS. FORM TYPE PROBLEM FILE
C
        ELSEIF (IMOD.EQ.2)THEN
          READ(LUN,END=991,ERR=991)MATM,MATC,MATK,MATL,INVM
          READ(LUN,END=991,ERR=991)DIMM,COLL
          DIMY=DIMM
          DIMX=2*DIMM
          DO 200 I=1,DIMY
200         C(I,I)=1.
          READ(LUN,END=991,ERR=991)A,B,DIMX,DIMU
          READ(LUN,END=991,ERR=991)EVALR,EVALI
C
C---- INCORRECT INPUT FOR THIS MODULE
C
        ELSE
          ERROR=.TRUE.
          WRITE(1,997)
997       FORMAT(//' **WARNING** THIS FILE IS NOT VALID',/,
     2      ' INPUT FOR MODULE C. I CANNOT READ IT.',/,
     3      ' PLEASE TRY ANOTHER PROBLEM FILE.')
        ENDIF
C
       ENDIF
       CLOSE(LUN,STATUS='KEEP')
       GOTO 999
C
991    ERROR=.TRUE.
       WRITE(1,993) FILNAM
993    FORMAT(//' *** ERROR ***  THE FILE: ',A15,/,
     2 ' IS NOT USABLE.  CHOOSE ANOTHER.  ')
       CLOSE(LUN,STATUS='KEEP')
C
999    RETURN
       END
C
C---- DUMMY PROBLEM READING SUBROUTINES FOR MODULES T AND F
C---- ADDED TO AVOID LOAD NOT COMPLETE FROM CALLS IN UNIFIL.
C
       SUBROUTINE FPROB(FILNAM,READIN,ERROR)
       CHARACTER*15 FILNAM
       LOGICAL READIN,ERROR
       WRITE(1,100)
100    FORMAT(//' THIS IS THE PROBLEM READER FOR MODULE F.'
     + /' YOU SHOULD NOT BE HERE!')
       ERROR=.TRUE.
       RETURN
       END
C
       SUBROUTINE TPROB(FILNAM,READIN,ERROR)
```

```
      CHARACTER*15 FILNAM
      LOGICAL READIN,ERROR
      WRITE(1,100)
100   FORMAT(//' THIS IS THE PROBLEM READER FOR MODULE T.'
     + /' YOU SHOULD NOT BE HERE!')
      ERROR=.TRUE.
      RETURN
      END
C
C
C****************************** CRSLT  ******************************
C**********************************************************************
C
C
C  (READS OR) WRITES STATE-SPACE TYPE PROBLEM FILES
C  FOR MODULE C. STATE EQU. ARE THE PSEUDO RESULTS
C  FOR THIS MODULE.
C  CALLED BY SAVFIL OR LODFIL FROM UNIFIL.FTN
C  WRITTEN BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE CRSLT(FILNAM,READIN,ERROR)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>EIGNBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>INTGBK.TEXT'
C
      CHARACTER*15 FILNAM
      INTEGER IMOD
      LOGICAL READIN,ERROR
C
      LUN=5
      ERROR=.FALSE.
C
C--- CREATE NEW FILE OR READ OLD PROBLEM FILE.
C--- SET CERTAIN PARAMETERS FOR A,B DATA FORMAT.
C--- BEGIN WRITING PROBLEM TO NEW FILE.
C
      IF (.NOT.READIN) THEN
        OPEN(LUN,FILE=FILNAM,ERR=991,STATUS='UNKNOWN',
     2  FORM='UNFORMATTED')
C
        IMOD=1
        FBKFLG=.FALSE.
        DO 10 I=1,MAXDIM
          KFB(I)=1.0
          IC(I)=0.0
          ITYPE(I)=1.0
10        UDEF(I,1)=1.0
        DO 20 I=2,10
          DO 20 J=1,MAXDIM
20          UDEF(J,I)=0.0
        KGAIN=1.0
```

```
        TINIT=1.0
        TFINAL=10.0
        TINTEG=1.0
        TSTORE=1.0
        NSTEP=1
        NSTOR=10
        TITLE2='*** MODULE C A,B RESULTS ***'
C
        WRITE(LUN,ERR=991)IMOD
        WRITE(LUN,ERR=991)TITLE1
        WRITE(LUN,ERR=991)DIMX,A
        WRITE(LUN,ERR=991)EVALR,EVALI
        WRITE(LUN,ERR=991)FBKFLG
        WRITE(LUN,ERR=991)KFB,KGAIN,IC
        WRITE(LUN,ERR=991)DIMU,B
        WRITE(LUN,ERR=991)ITYPE
        WRITE(LUN,ERR=991)UDEF
        WRITE(LUN,ERR=991)DIMY,C,D
        WRITE(LUN,ERR=991)TINIT,TFINAL,TINTEG
        WRITE(LUN,ERR=991)TSTORE,NSTEP,NSTOR
        WRITE(LUN,ERR=991)TITLE2
        ENDFILE(LUN)
C
C--- BEGIN READING INFORMATION FROM FILE SPECIFIED
C--- ERROR IF NOT PROPER FILE TYPE FOR MODULE C
C
      ELSE
C
C--- WILL NOT ALLOW READING OF A,B PROBLEM FILE
C--- IN MODULE C. CODE TO PERFORM READ OF USEFULL
C--- PORTION OF A,B FILE IS INCLUDED IF DESIRED TO
C--- CHANGE THIS IN FUTURE BUT IS CURRENTLY BYPASSED.
C
        WRITE(1,150)
150     FORMAT(//'    *** ERROR *** ',/,
     +  ' NOT ALLOWED TO LOAD A,B PROBLEM  ',/,
     +  ' FILES WHILE IN MODULE C! ')
C
C        OPEN(LUN,FILE=FILNAM,ERR=991,STATUS='OLD',
C     2  FORM='UNFORMATTED')
C        READ(LUN,END=991,ERR=991)IMOD
C        IF(IMOD.EQ.1)THEN
C          READ(LUN,END=991,ERR=991)TITLE1
C          READ(LUN,END=991,ERR=991)DIMX,A
C          READ(LUN,END=991,ERR=991)EVALR,EVALI
C          READ(LUN,END=991,ERR=991)FBKFLG
C          READ(LUN,END=991,ERR=991)KFB,KGAIN,IC
C          READ(LUN,END=991,ERR=991)DIMU,B
C        ELSE
C          ERROR=.TRUE.
C          WRITE(1,997)
C997       FORMAT(//' **WARNING** THIS FILE IS NOT VALID',/,
C     2    ' INPUT FOR MODULE C. I CANNOT READ IT.',/,
C     3    ' PLEASE TRY ANOTHER A,B PROBLEM FILE.')
```

```
C        ENDIF
       ENDIF
       CLOSE(LUN,STATUS='KEEP')
       GOTO 999
C
991    ERROR=.TRUE.
       WRITE(1,993) FILNAM
993    FORMAT(//' *** ERROR ***  THE FILE: ',A15,/,
      2 ' IS NOT USABLE.  CHOOSE ANOTHER.  ')
       CLOSE(LUN,STATUS='KEEP')
C
999    RETURN
       END
C
C--- DUMMY RESULTS READING SUBROUTINES FOR MODULES T AND F
C--- ADDED TO AVOID LOAD NOT COMPLETE FROM CALLS IN UNIFIL.
C
       SUBROUTINE FRSLT(FILNAM,READIN,ERROR)
       CHARACTER*15 FILNAM
       LOGICAL READIN,ERROR
       WRITE(1,100)
100    FORMAT(//' THIS IS THE RESULTS READER FOR MODULE F.'
      + /' YOU SHOULD NOT BE HERE!')
       ERROR=.TRUE.
       RETURN
       END
C
       SUBROUTINE TRSLT(FILNAM,READIN,ERROR)
       CHARACTER*15 FILNAM
       LOGICAL READIN,ERROR
       WRITE(1,100)
100    FORMAT(//' THIS IS THE RESULTS READER FOR MODULE T.'
      + /' YOU SHOULD NOT BE HERE!')
       ERROR=.TRUE.
       RETURN
       END
C
C
C****************************** PRINYM ******************************
C*********************************************************************
C
C
C  WRITES EITHER THE Y,U DIFF EQ OR M,C,K,L ARRAYS
C  DIRECTLY TO THE PRINTER DURING PROGRAM EXECUTION.
C  DATAPR IS USED TO SET UP THE PRINTER.
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
       SUBROUTINE PRINYM
C
       INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
       INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
       INCLUDE 'SYSKIT>COMMON.DIR>SGLIBK.TEXT'
       INCLUDE 'SYSKIT>COMMON.DIR>VIBRBK.TEXT'
C
```

```
      INTEGER LUNPR
      CHARACTER*15 DU,DY
C
      LUNPR=13
      CALL DATAPR
      WRITE(LUNPR,1015)
1015  FORMAT(//' *** MODULE C PROBLEM DESCRIPTION ***')
      WRITE(LUNPR,900) TITLE1
900   FORMAT(//' THE PROBLEM TITLE:',//,1X,A40)
C
      IF (SGLIFG) THEN
        WRITE(LUNPR,1000)
1000    FORMAT(//' THE DIFFERENTIAL EQUATION:',$)
        WRITE(LUNPR,1010) 'Y'
1010    FORMAT(//' THE ',A1,' COEFFICIENTS ARE')
        DO 10 I=ORDER,0,-1
          WRITE(DY,'(15A1)')'Y',('''',J=I,1,-1)
          WRITE(LUNPR,1020) COEF(I+1),DY
1020      FORMAT(/,5X,'(',1PE10.3,')*',A,$)
10        CONTINUE
        IF (INNMB.GT.0) THEN
          WRITE(LUNPR,1010) 'U'
          DO 20 I=INNMB,0,-1
            WRITE(DU,'(15A1)')'U',('''',J=I,1,-1)
            WRITE(LUNPR,1020) INPT(I+1),DU
20          CONTINUE
        ENDIF
C
      ELSE
        WRITE(LUNPR,3000) DIMM
3000    FORMAT(//' THE NUMBER OF COORDINATES IS ',I2)
        WRITE(LUNPR,3010) 'M'
3010    FORMAT(//' THE "',A1,'" MATRIX:',/)
        CALL PRMTX(DIMM,DIMM,MATM)
        WRITE(LUNPR,3010) 'C'
        CALL PRMTX(DIMM,DIMM,MATC)
        WRITE(LUNPR,3010) 'K'
        CALL PRMTX(DIMM,DIMM,MATK)
        WRITE(LUNPR,3020)
3020    FORMAT(//' THE "INVERSE-M" MATRIX:',/)
        CALL PRMTX(DIMM,DIMM,INVM)
        WRITE(LUNPR,3030) COLL
3030    FORMAT(//' THE NUMBER OF INPUTS IS ',I2)
        IF (COLL.GT.0) THEN
          WRITE(LUNPR,3010) 'L'
          CALL PRMTX(DIMM,COLL,MATL)
        ENDIF
      ENDIF
C
      WRITE(LUNPR,5000)
5000  FORMAT(//' *** END OF PROBLEM STATEMENT ***')
      CLOSE(LUNPR)
      RETURN
      END
```

```
C
C
C****************************** PRMTX  ******************************
C*************************************************************************
C
C
C  PRINTS OUT ANY MATRIX GIVEN TO IT
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE PRMTX(NROW,NCOL,MTX)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
C
      INTEGER NROW,NCOL,LUNPR
      REAL MTX
      DIMENSION MTX(MAXDIM,MAXDIM)
C
      LUNPR=13
      DO 100 I=1,NROW
        WRITE(LUNPR,1000)(MTX(I,J),J=1,NCOL)
1000    FORMAT(3X,5E13.5:/,(3X,5E13.5))
100     CONTINUE
      RETURN
      END
C
C
C****************************** PRINAB ******************************
C*************************************************************************
C
C
C  WRITES THE A,B MATRICES AND E.V. DIRECTLY
C  TO THE PRINTER DURING PROGRAM EXECUTION.
C  DATAPR IS USED TO SET UP THE PRINTER.
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE PRINAB
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>EIGNBK.TEXT'
C
      INTEGER LUNPR
C
      LUNPR=13
      CALL DATAPR
      WRITE(LUNPR,1015)
1015  FORMAT(//' *** MODULE C PROBLEM CONVERSION ***')
      WRITE(LUNPR,1000) TITLE1
1000  FORMAT(//' THE PROBLEM TITLE:',//,1X,A40)
C
      WRITE(LUNPR,1010) 'X',DIMX
1010  FORMAT(//' THE LENGTH OF ',A1,' IS ',I2)
      WRITE(LUNPR,1020) 'A'
```

```
1020  FORMAT(/' THE ',A1,' MATRIX:',/)
      CALL PRMTX(DIMX,DIMX,A)
      WRITE(LUNPR,1010) 'U',DIMU
      IF (DIMU.GT.0) THEN
        WRITE(LUNPR,1020) 'B'
        CALL PRMTX(DIMX,DIMU,B)
      ENDIF
      WRITE(LUNPR,1010) 'Y',DIMY
      WRITE(LUNPR,1020) 'C'
      CALL PRMTX(DIMY,DIMX,C)
C
      WRITE(LUNPR,1100)
1100  FORMAT(3X,/' THE EIGENVALUES:')
      WRITE(LUNPR,2000)
2000  FORMAT(3X,/' VALUE',4X,'REAL',6X,'IMAGINARY')
      DO 70 I=1,DIMX
        WRITE(LUNPR,2020) I,EVALR(I),EVALI(I)
2020    FORMAT(1X,I3,2X,1PE12.5,1X,E12.5)
70    CONTINUE
C
      WRITE(LUNPR,5000)
5000  FORMAT(//' *** END OF PROBLEM STATEMENT ***')
      CLOSE(LUNPR)
      RETURN
      END
C
C
C***************************** DATAPR ******************************
C*******************************************************************
C
C
C  THIS THING IS NOT TRANSPORTABLE ANYWHERE
C  THIS SUBROUTINE ALLOWS OUTPUT TO BE WRITTEN DIRECTLY
C  TO THE PRINTER QUEUE USING PRIMOS FUNIT=9 (LUN=13).
C  THE CALLING PROGRAM WRITES THE INFO. AND THEN MUST
C  CLOSE THE UNIT.
C  RE-WRITTEN BY J. GREGORSKI AUGUST,1984
C
      SUBROUTINE DATAPR
C
      INTEGER*2 INFO(12),BUF(1024),BUFL,CODE
      BUFL=1024
      INFO(1)=10
      INFO(2)=9
      INFO(3)=0
      INFO(4)=%120240
      INFO(5)=%120240
      INFO(6)=%120240
C
      CALL SPOOL$(INTS(2),'SPOOL_FROM_MODF',15,INFO,BUF,BUFL,CODE)
      CALL ERRPR$(K$IRTN,CODE,0,0,0,0)
C
      RETURN
      END
```

```
C
C
C****************************** INPUTM ********************************
C**********************************************************************
C
C
C  USED FOR INPUTTING ANY MATRIX INTERACTIVELY
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
C  LABEL     CHARACTER NAME FOR MATRIX TO BE INPUT
C  DIMR      DIMENSION OF ROW SPACE
C  DIMC      DIMENSION OF COL SPACE
C  MATRIX    ACTUAL ARRAY TO BE INPUT
C
      SUBROUTINE INPUTM(LABEL,DIMR,DIMC,MATRIX)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
C
      CHARACTER*1 LABEL,DUM
      INTEGER DIMR,DIMC
      REAL MATRIX(MAXDIM,MAXDIM)
      LOGICAL OK,YES
C
C--- MENU FORMAT FOR MATRIX INPUT
C
1     WRITE(1,100) LABEL
100   FORMAT(//' OPTIONS FOR THE ',A1,' MATRIX',
     2/' -------------------------')
      WRITE(1,101)
101   FORMAT('  A: CHANGE ALL ENTRIES',
     2/'  R: CHANGE A ROW',
     3/'  C: CHANGE A COLUMN')
      WRITE(1,102)
102   FORMAT('  E: CHANGE AN ENTRY',
     2/'  Z: ZERO THE MATRIX',
     3/'  L: LOOK AT THE MATRIX')
      IF (LABEL.NE.'A') THEN
        WRITE(1,103)
103     FORMAT('  T: TO MODULE OPTIONS')
      ENDIF
      WRITE(1,104)
104   FORMAT('  P: PROCEED  (=DEFAULT)',
     2/' -------------------------')
5     WRITE(1,105)
105   FORMAT(/' ENTER OPTION (P): ',$)
      IF(LABEL.EQ.'A')THEN
        CALL MENU(OK,ICH,'A','R','C','E','Z','L','','P','
     2   'P',25)
      ELSE
        CALL MENU(OK,ICH,'A','R','C','E','Z','L','T','',
     2   'P',25)
      ENDIF
      IF(OK) THEN
```

```
       GOTO(10,30,40,50,70,60,80,1,999)ICH
      ELSE
       GOTO 1
      ENDIF
C
C--- ENTER ALL BY ROW-COL
C
10    DO 15 I=1,DIMR
        WRITE(1,'(/)')
        DO 15 J=1,DIMC
15        CALL ELGET(I,J,MATRIX(I,J),LABEL)
      GOTO 1
C
C--- ENTER A ROW
C
30    WRITE(1,140)
140   FORMAT(/' ENTER ROW INDEX: ',$)
      IR=1
      CALL GETINT(IR,1,DIMR,25)
      WRITE(1,'(/)')
      DO 35 J=1,DIMC
35      CALL ELGET(IR,J,MATRIX(IR,J),LABEL)
      GOTO 1
C
C--- ENTER A COLUMN
C
40    WRITE(1,150)
150   FORMAT(/' ENTER COL INDEX: ',$)
      JC=1
      CALL GETINT(JC,1,DIMC,25)
      WRITE(1,'(/)')
      DO 45 I=1,DIMR
45      CALL ELGET(I,JC,MATRIX(I,JC),LABEL)
      GOTO 1
C
C--- ENTER AN ELEMENT
C
50    WRITE(1,140)
      IR=1
      CALL GETINT(IR,1,DIMR,25)
      WRITE(1,150)
      JC=1
      CALL GETINT(JC,1,DIMC,25)
      WRITE(1,'(/)')
      CALL ELGET(IR,JC,MATRIX(IR,JC),LABEL)
      GOTO 1
C
C--- LOOK
C
60    CALL OUTMAT(LABEL,DIMR,DIMC,MATRIX)
      WRITE(1,600)
600   FORMAT(/' PRESS <RET> KEY TO CONTINUE...')
C
C--- DUMMY CHAR FOR CALL
```

```
C
      DUM='A'
      CALL GETCHR(DUM,1)
      GOTO 1
C
C--- ZERO MATRIX
C
70    WRITE(1,180)
180   FORMAT(/' ZERO THE MATRIX? (NO): ',$)
      IF(YES(16,'N')) THEN
       DO 74 IR=1,DIMR
        DO 74 ICOL=1,DIMC
74    MATRIX(IR,ICOL)=0.0
      WRITE(1,176)LABEL
176   FORMAT(/' MATRIX ',A,' HAS BEEN ZEROED.')
      ENDIF
      GOTO 5
C
C--- ESCAPE
C
80    MODFLG=.TRUE.
C
999   RETURN
      END
C
C
C****************************** ELGET  ******************************
C*********************************************************************
C
C
C  USED TO GET INDIVIDUAL ARRAY ELEMENTS.
C
      SUBROUTINE ELGET(I,J,VAL,LABEL)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>PARMBK.TEXT'
      CHARACTER*1 LABEL
C
      WRITE(1,100) LABEL,I,J,VAL
100   FORMAT(3X,A1,'(',I1,',',I1,')? (',1PE10.3,'): ',$)
      CALL GETREL(VAL,BLO,BHI,15)
      RETURN
      END
C
C
C****************************** OUTMAT ******************************
C*********************************************************************
C
C
C  THIS SUBROUTINE DISPLAYS ANY MATRIX
C  WITH SPECIFIED LABEL AND DIMENSION
C  WRITTEN BY DZUNG A. DANG  JULY 01, 1982
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
```

```
C  LABEL     NAME OF MATRIX
C  NROW      NUMBER OF ROWS
C  NCOLUM    NUMBER OF COLS
C  MAX       MAXIMUM SIZE OF MATRIX
C  NFIELD    NUMBER OF FIELDS OF MATRIX ELEMENTS
C  MARGIN    RIGHT MARGIN OF THE SCREEN
C  LIMIT     MAXIMUM NUMBER OF COLS CAN BE DISPLAYED WITH GIVEN MARGIN
C  FORM      FORMAT OF THE DISPLAYED MATRIX
C  I         ROW INDEX
C  J         COL INDEX
C
      SUBROUTINE OUTMAT(LABEL,NROW,NCOLUM,MATRIX)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
C
      CHARACTER*1 LABEL
      CHARACTER*20 FORM
      INTEGER I,J,NROW,NCOLUM,MAX,NFIELD,MARGIN,LIMIT
      REAL MATRIX(MAXDIM,MAXDIM)
C
      MAX=MAXDIM
      NFIELD=12
      MARGIN=40
      LIMIT=NCOLUM
      IF((NCOLUM*NFIELD).GT.MARGIN) LIMIT=(NCOLUM+1)/2
C
C--- SPECIFYING THE OUTPUT FORMAT OF THE MATRIX
C
      FORM='(1X,A1,I1,A1,3E12.4)'
      IF(NCOLUM.EQ.1) FORM='(1X,A1,I1,A1,E12.4)'
      IF((NCOLUM.EQ.2).OR.(NCOLUM.EQ.4)) FORM='(1X,A1,I1,A1,2E12.4)'
C
C--- DISPLAYING THE MATRIX
C
      WRITE(1,'(/,'' THE '',A1,'' MATRIX IS:'',/)') LABEL
      IF(NCOLUM.EQ.1) THEN
        WRITE(1,'(10X,''C1'',/)')
      ELSEIF((NCOLUM.EQ.2).OR.(NCOLUM.EQ.4)) THEN
        WRITE(1,'(10X,''C1'',10X,''C2'',/)')
      ELSE
        WRITE(1,'(10X,''C1'',10X,''C2'',10X,''C3'',/)')
      ENDIF
      DO 10, I=1,NROW
        WRITE(1,FORM) 'R',I,':',(MATRIX(I,J),J=1,LIMIT)
10    CONTINUE
      IF(NCOLUM.NE.LIMIT) THEN
        WRITE(1,'('' '')')
        IF(NCOLUM.EQ.4) THEN
          WRITE(1,'(10X,''C3'',10X,''C4'',/)')
        ELSEIF(NCOLUM.EQ.5) THEN
          FORM='(1X,A1,I1,A1,2E12.4)'
          WRITE(1,'(10X,''C4'',10X,''C5'',/)')
        ELSE
          WRITE(1,'(10X,''C4'',10X,''C5'',10X,''C6'',/)')
```

```
        ENDIF
        DO 20, I=1,NROW
          WRITE(1,FORM) 'R',I,':',(MATRIX(I,J),J=LIMIT+1,NCOLUM)
20      CONTINUE
      ENDIF
      WRITE(1,'(/)')
C
      RETURN
      END
C
C
C*********************************************************************
C*********************************************************************
C
C
```

```
C
C
C****************************** CBLOK4 ******************************
C************************************************************************
C
C
C--- DESCRIPTION:
C          Contains all computational routines. Performs both
C          scalar and matrix multiplication and matrix inversion.
C          Also finds and sorts eigenvalues.
C
C--- CONTENTS:
C          MMULT    multiplies two matrices
C          SIMEQ    finds the inverse of a matrix
C          MLTYCK   multiplies two numbers and checks if in range
C          CHREQA   finds the characteristic poly. of an A matrix
C          DET      function that gets the determinant of a matrix
C          PROOT    finds the roots of a polynomial
C          SORT     puts eigenvalues in ascending order
C
C--- INDEX:
C          CHREQA
C          DET(FCN)
C          MLTYCK
C          MMULT
C          PROOT
C          SIMEQ
C          SORT
C
C
C****************************** MMULT  ******************************
C************************************************************************
C
C
C  MULTS C=A X B ,REAL MATRICES.
C  ASSUMES ALL MATRICES (AND VECTORS) HAVE MAX DIMENSION
C  MAXDIM IN CALLING PROGRAM.
C  E. GOODMAN, JULY 19, 1982
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE MMULT(A,B,C,NRA,NCA,NCB)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
C
      REAL A(MAXDIM,NCA),B(MAXDIM,NCB),C(MAXDIM,NCB),TMP
      DO 100 K=1,NCB
        DO 80 I=1,NRA
          TMP=0.
          DO 60 J=1,NCA
            TMP=A(I,J)*B(J,K)+TMP
60        CONTINUE
          C(I,K)=TMP
80      CONTINUE
100   CONTINUE
```

```
C
      RETURN
      END
C
C
C****************************** SIMEQ  ******************************
C*********************************************************************
C
C
C  THIS SUBROUTINE FINDS THE INVERSE OF THE MATRIX A USING
C  DIAGONLIZATION PROCEDURES.
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE SIMEQ (A,XDOT,KC,AINV,X,IERR)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
C
      DIMENSION A(MAXDIM,MAXDIM),B(MAXDIM,MAXDIM),XDOT(MAXDIM+1),
     + X(MAXDIM+1),AINV(MAXDIM,MAXDIM)
C
      IERR=1
      DO 1 I=1,KC
        DO 1 J=1,KC
          AINV(I,J)=0
          B(I,J)=A(I,J)
1     CONTINUE
      DO 2 I=1,KC
        AINV(I,I)=1
        X(I)=XDOT(I)
2     CONTINUE
      DO 3 I=1,KC
        COMP=0
        K=I
        N=I
6       IF(ABS (B(K,I))-ABS (COMP))5,5,4
4       COMP=B(K,I)
        N=K
5       K=K+1
        IF(K-KC)6,6,7
7       IF(B(N,I))8,51,8
8       IF(N-I)51,12,9
9       DO 10 M=1,KC
          TEMP=B(I,M)
          B(I,M)=B(N,M)
          B(N,M)=TEMP
          TEMP=AINV(I,M)
          AINV(I,M)=AINV(N,M)
          AINV(N,M)=TEMP
10      CONTINUE
        TEMP=X(I)
        X(I)=X(N)
        X(N)=TEMP
12      X(I)=X(I)/B(I,I)
        TEMP=B(I,I)
```

```
        DO 13 M=1,KC
          AINV(I,M)=AINV(I,M)/TEMP
          B(I,M)=B(I,M)/TEMP
13      CONTINUE
        DO 16 J=1,KC
          IF(J-I)14,16,14
14        IF(B(J,I))15,16,15
15        X(J)=X(J)-B(J,I)*X(I)
          TEMP=B(J,I)
          DO 17 N=1,KC
            AINV(J,N)=AINV(J,N)-TEMP*AINV(I,N)
            B(J,N)=B(J,N)-TEMP*B(I,N)
17        CONTINUE
16      CONTINUE
3     CONTINUE
      RETURN
51    IERR=0
C
      RETURN
      END
C
C
C****************************** MLTYCK ********************************
C**********************************************************************
C
C
C  THIS SUBROUTINE IS DESIGNED TO TEST
C  TWO REAL NUMBERS TO DETERMINE IF
C  THEIR PRODUCT WILL BE IN RANGE. IT
C  WILL FIND THIS PRODUCT WHEN IT IS IN
C  RANGE. IF THE EXPONENT OF THE PRODUCT
C  IS LESS THAN -15, RESULT WILL EQUAL 0
C  AN EXPONENT GREATER THAN 15 IS OUT OF
C  RANGE.
C  WRITTEN BY KEN WHITE--AUG 19,1982
C
      SUBROUTINE MLTYCK(NUM1,NUM2,GOOD,RESULT,EXPN)
C
      REAL NUM1,NUM2,RESULT,EXPN
      LOGICAL GOOD
C
      GOOD=.TRUE.
      IF((ABS(NUM1).LT.1.0E-35).OR.(ABS(NUM2).LT.1.0E-35))THEN
        RESULT=0.0
      ELSE
        EXPN=ALOG10(ABS(NUM1))+ALOG10(ABS(NUM2))
        IF(EXPN.GT.15.)THEN
          GOOD=.FALSE.
        ELSE IF(EXPN.LT.-15.)THEN
          RESULT=0.0
        ELSE
          RESULT=NUM1*NUM2
        END IF
      END IF
```

```
      RETURN
      END
C
C
C****************************** CHREQA ********************************
C**********************************************************************
C
C
C  SUBROUTINE CHREQA DETERMINES COEFFICIENTS OF CHAR POLY OF A.
C  SEE J.L.MELSA: "AN ALGORITHM FOR THE DESIGN OF LINEAR STATE
C  VARIABLE FEEDBACK SYSTEMS", PROC ASILOMAR CONFERENCE
C  ON SYSTEMS AND CIRCUITS,MONTEREY, CALIFORNIA ,PP. 791-799,
C  NOV. 1967.
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
C  N=IDDEG, C=COEFFD, A=A MATRIX
C
      SUBROUTINE CHREQA(A,N,C,BADDAT)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      LOGICAL BADDAT
      REAL A(MAXDIM,MAXDIM),C(MAXDIM+1)
      REAL B(MAXDIM,MAXDIM),D(150)
      INTEGER N,J(MAXDIM+1)
C
      NN=N+1
      DO 20 I=1,NN
20    C(I)=0.0
      C(NN)=1.0
      DO 14 M=1,N
      K=0
      L=1
      J(1)=1
      GO TO 2
1     J(L)=J(L)+1
2     IF(L-M)3,5,990
3     MM=M-1
      DO 4 I=L,MM
      II=I+1
4     J(II)=J(I)+1
5     DO 10 I=1,M
      DO 10 KK=1,M
      NR=J(I)
      NC=J(KK)
10    B(I,KK)=A(NR,NC)
      K=K+1
      D(K)=DET(B,M)
      DO 6 I=1,M
      L=M-I+1
      IF(J(L)-(N-M+L))1,6,990
6     CONTINUE
      M1=N-M+1
      DO 14 I=1,K
14    C(M1)=C(M1)+D(I)*(-1.0)**M
```

```
      GOTO 999
990   BADDAT=.TRUE.
C
999   RETURN
      END
C
C
C****************************** DET(FCN) ****************************
C*********************************************************************
C
C
C  FUNCTION DET CALCULATES THE DETERMINANT OF A MATRIX .
C  A IS THE MATRIX AND KC IS THE ORDER OF MATRIX A .
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
      FUNCTION DET(A,KC)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      REAL A(MAXDIM,MAXDIM),B(MAXDIM,MAXDIM)
C
      IREV=0
      DO 1 I=1,KC
      DO 1 J=1,KC
1     B(I,J)=A(I,J)
      DO 20 I=1,KC
      K=I
9     IF(B(K,I))10,11,10
11    K=K+1
      IF(K-KC)9,9,51
10    IF(I-K)12,14,51
12    DO 13 M=1,KC
      TEMP=B(I,M)
      B(I,M)=B(K,M)
13    B(K,M)=TEMP
      IREV=IREV+1
14    II=I+1
      IF(II.GT.KC) GO TO 20
      DO 17 M=II,KC
      IF(B(M,I))19,17,19
19    TEMP=B(M,I)/B(I,I)
      DO 16 N=I,KC
16    B(M,N)=B(M,N)-B(I,N)*TEMP
17    CONTINUE
20    CONTINUE
      DET=1.0
      DO 2 I=1,KC
2     DET=DET*B(I,I)
      DET=(-1.0)**IREV*DET
      RETURN
51    DET=0.0
      RETURN
      END
C
C
```

```
C****************************** PROOT  ******************************
C*********************************************************************
C
C
C  SUBROUTINE PROOT  USES A MODIFIED BARSTOW METHOD TO FIND THE
C  ROOTS OF A POLYNOMIAL . SEE R. W. HAMMING: NUMERICAL METHODS
C  FOR SCIENTISTS AND ENGINEERS ,   MCGRAW-HILL BOOK COMPANY ,
C  INC . , 1962 , PP. 356-359 , FOR FURTHER INFORMATION .
C  MODIFIED BY J. GREGORSKI DECEMBER,1984
C
C  N=ORDER POLYNOMIAL
C  A=COEFFICIENTS POLY
C  U=REAL PARTS ROOTS
C  V=IMAG PARTS ROOTS
C  IR=MAGICAL MYSTERY VARIABLE
C
      SUBROUTINE PROOT(N,A,U,V,IR)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      REAL A(MAXCOF),U(MAXDEG),V(MAXDEG),H(MAXCOF),
     2 B(MAXCOF),C(MAXCOF)
C
      IREV=IR
      NC=N+1
C
      DO 1 I=1,NC
1     H(I)=A(I)
      P=0.0
      Q=0.0
      R=0.0
3     IF(H(1))4,2,4
2     NC=NC-1
      IF(NC.LE.0) GOTO 100
      V(NC)=0.0
      U(NC)=0.0
      DO 1002 I=1,NC
1002  H(I)=H(I+1)
      GO TO 3
4     IF(NC-1)5,100,5
5     IF(NC-2)7,6,7
6     R=-H(1)/H(2)
      GO TO 50
7     IF(NC-3)9,8,9
8     P=H(2)/H(3)
      Q=H(1)/H(3)
      GO TO 70
9     IF(ABS(H(NC-1)/H(NC))-ABS(H(2)/H(1)))10,19,19
10    IREV=-IREV
      M=NC/2
      DO 11 I=1,M
      NL=NC+1-I
      F=H(NL)
      H(NL)=H(I)
11    H(I)=F
```

```
      IF(Q)13,12,13
12    P=0.0
      GO TO 15
13    P=P/Q
      Q=1.0/Q
15    IF(R)16,19,16
16    R=1.0/R
19    E=4.0E-7
      B(NC)=H(NC)
      C(NC)=H(NC)
      B(NC+1)=0.0
      C(NC+1)=0.0
      NP=NC-1
20    DO 49 J=1,1000
      DO 21 I1=1,NP
      I=NC-I1
      B(I)=H(I)+R*B(I+1)
21    C(I)=B(I)+R*C(I+1)
      IF(ABS(B(1)/H(1))-E)50,50,24
24    IF(C(2))23,22,23
22    R=R+1.0
      GO TO 30
23    R=R-B(1)/C(2)
30    DO 37 I1=1,NP
      I=NC-I1
      B(I)=H(I)-P*B(I+1)-Q*B(I+2)
37    C(I)=B(I)-P*C(I+1)-Q*C(I+2)
      IF (H(2))32,31,32
31    IF(ABS(B(2)/H(1))-E)33,33,34
32    IF(ABS(B(2)/H(2))-E)33,33,34
33    IF(ABS(B(1)/H(1))-E)70,70,34
34    CBAR=C(2)-B(2)
      D=C(3)**2-CBAR*C(4)
      IF(D)36,35,36
35    P=P-2.
      Q=Q*(Q+1.0)
      GO TO 49
36    P=P+(B(2)*C(3)-B(1)*C(4))/D
      Q=Q+(-B(2)*CBAR+B(1)*C(3))/D
49    CONTINUE
      E=E*4.
      GO TO 20
50    NC=NC-1
      V(NC)=0.0
      IF(IREV)51,52,52
51    U(NC)=1.0/R
      GO TO 53
52    U(NC)=R
53    DO 54 I=1,NC
54    H(I)=B(I+1)
      GO TO 4
70    NC=NC-2
      IF(IREV)71,72,72
71    QP=1.0/Q
```

```
      PP=P/(Q*2.0)
      GO TO 73
72    QP=Q
      PP=P/2.0
73    F=(PP)**2-QP
      IF(F)74,75,75
74    U(NC+1)=-PP
      U(NC)=-PP
      V(NC+1)=SQRT(-F)
      V(NC)=-V(NC+1)
      GO TO 76
75    IF(PP)81,80,81
80    U(NC+1)=-SQRT(F)
      GO TO 82
81    U(NC+1)=-(PP/ABS(PP))*(ABS(PP)+SQRT(F))
82    CONTINUE
      V(NC+1)=0.0
      U(NC)=QP/U(NC+1)
      V(NC)=0.0
76    DO 77 I=1,NC
77    H(I)=B(I+2)
      GO TO 4
C
100   RETURN
      END
C
C
C****************************** SORT   ******************************
C*************************************************************************
C
C
C  PUTS EIGENVALUES IN ASCENDING ORDER
C  MODIFIED BY J. GREGORSKI APRIL,1985
C
      SUBROUTINE SORT(RR,RI,NORDER)
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
C
      DIMENSION RR(MAXDIM),RI(MAXDIM)
      DO 40 I=1,NORDER-1
         DO 30 J=1,NORDER-1
            IF(RR(J).LE.RR(J+1)) GO TO 25
            N=J+1
            RRTEMP=RR(J)
            RITEMP=RI(J)
            RR(J)=RR(N)
            RI(J)=RI(N)
            RR(N)=RRTEMP
            RI(N)=RITEMP
25       CONTINUE
30       CONTINUE
40    CONTINUE
      RETURN
      END
```

```
C
C
C**********************************************************************
C**********************************************************************
C
C
```

Appendix D

UTILITY FILES SOURCE CODE LISTINGS

```
C
C
C****************************** IOUTIL ********************************
C**********************************************************************
C
C
C--- DESCRIPTION:
C          Contains all I/O routines to insure robust
C          operation. All inputs are checked for validity
C          before being used.
C
C--- CONTENTS:
C          FNAME   reads input filename & checks for validity
C          GETINT  reads an integer number
C          GETREL  reads a real number
C          GETCHR  reads a character
C          GETLIN  reads an entire line
C          CHECKY  checks a string for all integer values
C          CHRPCK  writes an array of characters to a string
C          YES     logical function to read YES or NO response
C          MENU    reads menu option selected and checks if valid
C          PROCED  asks if you want to proceed
C          MYCHAR  function returns true ASCII code of a char.
C
C--- INDEX:
C          CHECKY
C          CHRPCK
C          FNAME
C          GETCHR
C          GETINT
C          GETLIN
C          GETREL
C          MENU
C          MYCHAR(FCN)
C          PROCED
C          YES(LGL FCN)
C
C
C****************************** FNAME  ********************************
C**********************************************************************
C
C
```

```
C  READS IN FILENAME.CHECKS FOR A VALID FILENAME.
C  CHECKS AGAIN WITH USER.IF NO NAME ENTERED
C  THEN EXIT SET TO TRUE.CALLING SEGMENT USES THIS
C  TO EXIT FROM TASK.
C  WRITTEN BY J. GREGORSKI MARCH,1985
C
      SUBROUTINE FNAME(FILNAM,EXIT)
C
      CHARACTER*1 LINE(40)
      CHARACTER*15 FILNAM
      LOGICAL YES,EXIT
      EXIT=.FALSE.
C
100   WRITE(1,89)
89    FORMAT(/' ENTER A FILE NAME (15 CHAR. OR LESS)',
     2/,' OR HIT <RETURN> TO EXIT. ',$)
      CALL GETLIN(ISTART,ISTOP,LINE,17)
C
C--- NOTE EXTRA COLUMNS READ TO DETECT TOO-LONG NAME...
C
      IF(ISTART.GE.16)THEN
      EXIT=.TRUE.
      GOTO 999
      ENDIF
C
C--- CHECK VALIDITY OF FILENAME TYPED IN.
C
      IF(ISTOP.GT.15.OR.LINE(ISTART).LT.'A'
     2   .OR.LINE(ISTART).GT.'Z')THEN
        WRITE(1,102)
102     FORMAT(/' FILE NAME MUST BE 1-15 ALPHANUMERIC',/,
     2   ' CHARACTERS (INCLUDING "." AND "/")',/,
     3   ' AND START WITH A LETTER. ')
        GO TO 100
      ENDIF
      DO 105 I=ISTART,ISTOP
C
C--- CHECK FOR ALPHA,DIGIT, PERIOD, OR SLASH...
C
        IF(LINE(I) .GE. '.' .AND. LINE(I) .LE. '9' .OR.
     2     LINE(I) .GE. 'A' .AND. LINE(I) .LE. 'Z')THEN
          GO TO 105
        ELSE
          WRITE(1,102)
          GO TO 100
        ENDIF
105   CONTINUE
C
C--- ASSEMBLE FILE NAME
C
      WRITE(FILNAM,111)(LINE(I),I=ISTART,ISTOP)
111   FORMAT(15A1)
C
C--- VERIFY NAME FOR WRITING OR READING
```

```
C
      WRITE(1,201) FILNAM
201   FORMAT(/' YOU ENTERED FILE NAME: ',A15,/,
     2  ' IS THAT CORRECT? (YES): ',$)
        IF(.NOT.YES(15,'Y'))GO TO 100
999   RETURN
      END
C
C
C****************************** GETINT ********************************
C**********************************************************************
C
C
C  THIS ROUTINE RETURNS AN INTEGER IN IVAR
C  READ FROM THE CONSOLE AND CHECKED FOR VALIDITY...
C
      SUBROUTINE GETINT(IVAR,LOWER,UPPER,LEN)
C
      CHARACTER*5 IZINT
      CHARACTER*1 LINE(40)
      INTEGER ISTART,ISTOP,IVAR,LOWER,UPPER,FRSTCH,LEN,LLEN
      LOGICAL JUNK
C
      LLEN=LEN
10    CALL GETLIN(ISTART,ISTOP,LINE,LLEN)
C
      IF(ISTART.EQ.LLEN+1) RETURN
C
C--- RETURN ON NO INPUT WITH DEFAULT VALUE PRESENT IN IVAR
C
C--- WE CHECK THE FIRST CHARACTER FOR +, -, OR NUMBER
C
      I=MYCHAR(LINE(ISTART))
C
C--- CHECK FOR + (ASCII=43)
C--- CHECK FOR - (ASCII=45)
C
C
      FRSTCH = ISTART
      IF(I.NE.43 .AND. I.NE.45) GOTO 15
      FRSTCH = FRSTCH+1
C
15    CALL CHECKY(LINE,FRSTCH,ISTOP,JUNK)
      IF(JUNK) GOTO 800
C
C--- NOW WE MAKE A CURSORY CHECK FOR RANGE...
C
      IF((ISTOP-ISTART).GT.3) THEN
        GOTO 800
      ENDIF
C
C--- NOW WE CAN SEND AN INTEGER BACK...
C
      WRITE(IZINT,300) (LINE(I),I=ISTART,ISTOP)
```

```
300   FORMAT(5A1)
C
C--- AND NOW WE GET THE INTEGER INTO AN INTEGER VARIABLE...
C
      READ(IZINT,400) I
400   FORMAT(BN,I5)
C
C--- WE CHECK AGAIN TO SEE IF INPUT IS WITHIN RANGE...
C
      IF(I.LT.LOWER .OR. I.GT.UPPER) THEN
        GOTO 800
        ENDIF
      IVAR = I
C
C
C---  AND NOW WE CAN GO HOME...
C
C
      RETURN
C
800   WRITE(1,100)LOWER,UPPER
100   FORMAT(/' ENTER INTEGER FROM ',I5,' TO ',I5,': ',$)
      LLEN=5
      GOTO 10
      END
C
C
C****************************** GETREL ******************************
C*********************************************************************
C
C
C  THIS ROUTINE GETS A LINE FROM THE CONSOLE,
C  LOOKS FOR A REAL NUMBER AND PERFORMS VALIDITY CHECKS.
C  IF IT DOESN'T LIKE WHAT THE USER HAS GIVEN IT, THE USER
C  WILL BE DUNNED INTO PRODUCING AN ACCEPTABLE REAL NUMBER.
C
      SUBROUTINE GETREL(RELVAR,LOWER,UPPER,LEN)
C
      LOGICAL JUNK
      INTEGER FRSTCH
      CHARACTER*1 IZDOT,IZE
      CHARACTER*1 LINE(40)
      CHARACTER*40 STRING
      INTEGER ISTART,ISTOP,LEN,LLEN
      REAL LOWER,UPPER,TEMP,X
C
C--- FRSTCH POINTS TO THE FIRST NUMERICAL CHAR (OR .)
C--- IDOT POINTS TO A DECIMAL POINT
C
      DATA IZDOT,IZE/'.','E'/
C
      LLEN=LEN
10    CALL GETLIN(ISTART,ISTOP,LINE,LLEN)
C
```

```
      IF(ISTART.EQ.LLEN+1) RETURN
C
      FRSTCH=ISTART+1
C
      I=MYCHAR(LINE(ISTART))
      IF(I.EQ.43    ) GOTO 20
      IF(I.EQ.45    ) GOTO 20
      FRSTCH=ISTART
C
      IF(I.EQ.46) GOTO  20
      IF(I.GE.48 .AND.I.LE.57) GOTO 20
C
C--- WE GOT A LOSER...
C
      GOTO 800
C
20    IDOT = 0
      IPOSE = 0
      DO 30 I=ISTART,ISTOP
        IF(LINE(I).EQ.IZDOT)IDOT=I
        IF(LINE(I).EQ.IZE)IPOSE=I
30    CONTINUE
C
      K = ISTOP
      IF(IPOSE.NE.0) THEN
C
C--- WE HAVE AN EXPONENT...
C
        J=IPOSE+1
        I=MYCHAR(LINE(J))
        IF(I.EQ.43.OR.I.EQ.45)  J=J+1
C
C--- J NOW POINTS TO THE FIRST # IN EXPONENT...
C
        CALL CHECKY (LINE,J,ISTOP,JUNK)
        IF(JUNK) GOTO 800
        IJK=ISTOP-J
        IF((IJK    ).GT.1) GOTO 800
        IF((IJK    ).EQ.1 .AND.MYCHAR(LINE(J)).GT.50)GOTO 800
C
C--- WE CHECK TO SEE THAT THERE IS SOME NUMBER AFTER EXP.
C
        IF((IJK    ).LT.0)GOTO 800
C
C--- CHECKS FOR OVERLARGE EXPONENT...
C
        K = IPOSE - 1
      ELSE
        IPOSE = ISTOP+1
C
C--- IPOSE POINTS TO THE PLACE AFTER MANTISSA..
C
      ENDIF
C
```

```
C--- NOW WE CHECK FOR A PERIOD..
C
      IF(IDOT.NE.0) THEN
C
C--- HERE WE CHECK FOR AN REAL NUMBER...
C
        I=IDOT-1
C
C--- I POINTS TO LAST POSSIBLE CHAR BEFORE '.'.
C
        CALL CHECKY(LINE,FRSTCH,I,JUNK)
        IF(JUNK)GOTO 800
C
C--- K POINTS TO LAST CHAR BEFORE EXPONENT...
C
        I=I+2
C
C--- I POINTS TO THE FIRST CHAR AFTER '.'
C
        CALL CHECKY(LINE,I,K,JUNK)
        IF(JUNK) GOTO 800
C
      ELSE
C
C--- HERE WE CHECK FOR AN INTEGER...
C--- (POSSIBLY WITH EXPONENT)
C
        CALL CHECKY(LINE,FRSTCH,K,JUNK)
        IF(JUNK)GOTO 800
C--- IF(IDOT.EQ.0 .AND.IPOSE.NE.0)GOTO 800
C--- LINE(K+1)='.'
C
      ENDIF
C
C--- NOW WE CHECK THE RANGE...
C
      IF(IDOT.EQ.0) THEN
        IPOWER=K-FRSTCH+1
      ELSE
        IPOWER=IDOT-FRSTCH
      ENDIF
C
C--- CHECK FOR LEADING ZEROES
C
      DO 40 IPOW=FRSTCH,K
        I=MYCHAR(LINE(I))
        IF(I.NE.46 .OR. I.NE.48) GOTO 50
        IPOWER=IPOWER-1
40    CONTINUE
50    CONTINUE
C
C--- NOW WE CHECK THE EXPONENT..
C
      IF(IPOSE.LT.ISTOP)THEN
```

```
        IEXP = MYCHAR(LINE(J)) - 48
        IF(J.NE.ISTOP) IEXP=IEXP*10+MYCHAR(LINE(ISTOP))-48
C
C--- CHECK FOR NEG EXPONENT AND ADD TO IPOWER..
C
        IF(LINE(J-1) .EQ.'-') IEXP=-IEXP
C
        IPOWER = IPOWER +IEXP
      ENDIF
C
      IF(IPOWER.GT.16) GOTO 800
      IF(IPOWER.LT.-14)GOTO 810
C
C--- NOW WE PACK IT BACK INTO A STRING..
C
      CALL CHRPCK(LINE,STRING,ISTART,ISTOP)
C
      READ(STRING,100) TEMP
100   FORMAT(BN,F10.0)
C
C
      IF (LOWER.GT.UPPER) THEN
        X=LOWER
        LOWER=UPPER
        UPPER=X
      ENDIF
      IF ((TEMP.LT.LOWER).OR.(TEMP.GT.UPPER)) GOTO 800
      RELVAR=TEMP
      RETURN
C
C--- THIS IS WHAT I CALL THE PAUL HAAS MEMORIAL
C--- 'PLEASE TRY AGAIN, DOORKNOB...' SECTION
C
800   WRITE(1,200)LOWER,UPPER
200   FORMAT(/' ENTER NUMBER FROM ',1PE10.3,
     2 /,' TO ',1PE10.3,': ',$)
      LLEN=23
      GOTO 10
810   WRITE(1,210)
210   FORMAT(/' *** NUMBER TOO SMALL. ENTER 0 OR',
     2       /' NUMBER WITH MAGNITUDE OVER 1.E-15.',
     3       /' ENTER NUMBER: ',$)
      LLEN=23
      GO TO 10
C
C
      END
C
C
C****************************** GETCHR ******************************
C*********************************************************************
C
C
C  ACCEPTS DEFAULT, CAR, IN AND LOOKS FOR LEGAL CAR. IN
```

```
C  LEN COLUMNS. RETURNS AS NEW VALUE OF CAR.  IF BLANK LINE,
C  RETURNS DEFAULT IN CAR UNCHANGED.
C
      SUBROUTINE GETCHR(CAR,LEN)
C
      INTEGER LEN,ISTART,ISTOP,LLEN
      CHARACTER*1 LINE(40),CAR
C
      LLEN=LEN
10    CALL GETLIN(ISTART,ISTOP,LINE,LLEN)
C
      IF(ISTART.GE.LLEN+1) RETURN
C
      I=MYCHAR(LINE(ISTART))
      IF(I.LT.32 .OR. I.GT.90) GOTO 800
C
      CAR = LINE(ISTART)
      RETURN
C
800   WRITE(1,100)
100   FORMAT(' ILLEGAL CHARACTER ENTERED. ',/,
     2'  PLEASE TRY AGAIN: ',$)
      LLEN=20
      GOTO 10
      END
C
C
C****************************** GETLIN ******************************
C********************************************************************
C
C
C  FINDS AND NUMBERS BLANKS IN INPUT LINE.
C  DETERMINES RANGE OF NON-BLANK CHARACTERS
C  IN STRING.
C
      SUBROUTINE GETLIN(ISTART,ISTOP,INPUT,LEN)
C
      CHARACTER*1 INPUT(40)
      INTEGER ISTOP,ISTART,LEN
C
C
      READ(1,100) (INPUT(I),I=1,LEN)
100   FORMAT(40A1)
C
C--- FIND THE FIRST NON-BLANK CHARACTER...
C
      ISTART = 0
10    ISTART = ISTART+1
        IF(INPUT(ISTART).EQ.' ' .AND.ISTART.LT.LEN+1) GOTO 10
C
C--- FIND THE FIRST BLANK AFTER ISTART..
C
      ISTOP = ISTART - 1
20    ISTOP = ISTOP +1
```

```
        IF(INPUT(ISTOP+1).NE.' ' .AND.ISTOP .LT.LEN+1) GOTO 20
C
C
      RETURN
      END
C
C
C****************************** CHECKY ******************************
C*********************************************************************
C
C
C  THIS ROUTINE CHECKS A STRING OF CHARACTERS
C  TO SEE IF THEY ARE ALL INTEGERS AND RETURNS A
C  LOGICAL VARIABLE (JUNK) .TRUE. IF THEY
C  ARE NOT.
C
      SUBROUTINE CHECKY (LINE,ISTART,ISTOP,JUNK)
C
      CHARACTER*1 LINE(40)
      INTEGER ISTART,ISTOP
      LOGICAL JUNK
C
      DO 10 I=ISTART,ISTOP
        J=MYCHAR(LINE(I))
        IF(J.LT.48 .OR. J.GT.57) THEN
          JUNK = .TRUE.
          RETURN
        ENDIF
10    CONTINUE
C
      JUNK = .FALSE.
      RETURN
      END
C
C
C****************************** CHRPCK ******************************
C*********************************************************************
C
C
C  WRITES ARRAY OF CHARACTERS TO A STRING.
C  STRING REVISED TO *32 TO WORK FOR FILENAMES PASSED.
C  USER SHOULD NOT CALL WITH ISTOP BEYOND 32.
C
      SUBROUTINE CHRPCK(LINE,STRING,ISTART,ISTOP)
C
      CHARACTER*1 LINE(40)
      CHARACTER*32 STRING
      INTEGER ISTART,ISTOP
C
      WRITE(STRING,100) (LINE(I),I=ISTART,ISTOP)
100   FORMAT(32A1)
C
      RETURN
      END
```

```
C
C
C****************************** YES(LGL.FCN) ************************
C*****************************************************************
C
C
C  SETS YES=.TRUE. IF RESPONSE=YES OR
C  SETS YES=.FALSE. IF RESPONSE =NO.
C  ASKS AGAIN IF NEITHER.
C
      LOGICAL FUNCTION YES(LEN,DEFCHR)
C
      CHARACTER*1 CHDF,DEFCHR
      INTEGER LEN,LLEN
C
      YES=.FALSE.
      LLEN=LEN
10    CHDF=DEFCHR
      CALL GETCHR(CHDF,LLEN)
      IF(CHDF.EQ.'Y')THEN
        YES=.TRUE.
        RETURN
      ELSE IF(CHDF.EQ.'N')THEN
        RETURN
      ELSE
        WRITE(1,100)
100     FORMAT(/' YES OR NO, PLEASE: ',$)
        LLEN=20
        GO TO 10
      ENDIF
      END
C
C
C****************************** MENU   ******************************
C*****************************************************************
C
C
C  READS MENU OPTION DESIRED AND CHECKS FOR VALIDITY.
C  RETURNS SELECTED OPTION IF OK.
C
      SUBROUTINE MENU(OK,I,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,
     2 DEFCHR,LEN)
C
      INTEGER LEN,LLEN,I
      LOGICAL OK
      CHARACTER*1 C(10),C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,CAR,
     2 DEFCHR
C
      C(1)=C1
      C(2)=C2
      C(3)=C3
      C(4)=C4
      C(5)=C5
      C(6)=C6
```

```
      C(7)=C7
      C(8)=C8
      C(9)=C9
      C(10)=C10
C
      LLEN=LEN
      N=1
5     CAR=DEFCHR
      CALL GETCHR(CAR,LLEN)
      DO 10 J=1,10
        IF(CAR.EQ.C(J))THEN
          IF(CAR.EQ.'
          I=J
          OK=.TRUE.
          RETURN
        ENDIF
10    CONTINUE
      IF(N.EQ.1)THEN
        N=2
        WRITE(1,100)
100     FORMAT(/' BAD OPTION.  ENTER NEW CHOICE: ',$)
        LLEN=13
        GO TO 5
      ELSE
        WRITE(1,110)
110     FORMAT(/' BAD OPTION AGAIN. ')
        OK=.FALSE.
        RETURN
      ENDIF
      END
C
C
C****************************** PROCED ******************************
C********************************************************************
C
C
C  PROCEED ROUTINE IS ROBUST, RETURNS T OR F. EDG.
C
      SUBROUTINE PROCED(ANSWER)
C
      LOGICAL ANSWER,YES
      WRITE(1,51)
51    FORMAT(/' PROCEED? (YES): ',$)
      IF(YES(23,'Y'))THEN
        ANSWER=.TRUE.
      ELSE
        ANSWER=.FALSE.
      ENDIF
      END
C
C
C****************************** MYCHAR(FCN) ************************
C********************************************************************
C
```

```
C
C  MYCHAR RETURNS 'TRUE' ASCII CODE OF CHAR
C  11/07/83 ZZ
C
      FUNCTION MYCHAR(CH)
C
      CHARACTER*1 CH
      MYCHAR = AND(ICHAR(CH),127)
      RETURN
      END
C
C
C*********************************************************************
C*********************************************************************
C
C
```

```
C
C
C****************************** UNIFIL ********************************
C**********************************************************************
C
C
C--- DESCRIPTION:
C          Performs management of both problem and results
C          user files. File names are stored in the library
C          file and the files reside in the users directory.
C
C--- CONTENTS:
C          UNIDRV  main driving program for file mgt.
C          LSTLIB  lists the library of user files
C          DELFIL  deletes a user file
C          CHKLIB  updates or creates the file library
C          ADAFIL  adds a filename to the library
C          SAVFIL  creates a user file
C          LODFIL  loads a user file
C
C--- INDEX:
C          ADAFIL
C          CHKLIB
C          DELFIL
C          LODFIL
C          LSTLIB
C          SAVFIL
C          UNIDRV
C
C
C****************************** UNIDRV ********************************
C**********************************************************************
C
C
C  MAIN CALLING ROUTINE FOR FILE MANAGEMENT SYSTEM.
C  CONTAINS MENU AND DIRECTS FLOW APPROPRIATELY.
C  WRITTEN BY J. GREGORSKI JANUARY,1985
C
      SUBROUTINE UNIDRV
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      CHARACTER*9 INSET
      LOGICAL OK,YES
C
C--- ASK USER FOR TYPE OF FILES THEY WANT TO ACCESS.
C--- USER IS BADGERED UNTIL THEY MAKE A DECISION.
C--- USE PROPER LIBRARY AND MENU FOR TYPE OF FILES.
C--- MADE TO BE COMPATIBLE WITH ALL 3 MODULES.
C
      IF (MODULE.EQ.'C') THEN
        LIBNAM='PROBLEM*LIB'
        INSET=' PROBLEM '
      ENDIF
```

```
C
      WRITE(1,110)
110   FORMAT(//' DO YOU WANT TO WORK WITH:')
C
100   IF (MODULE.EQ.'C') THEN
        WRITE(1,125)
125     FORMAT('  DIFFERENTIAL EQUATION PROBLEM FILES? (YES): ',$)
      ELSE
        WRITE(1,120)
120     FORMAT('  PROBLEM FILES? (YES): ',$)
      ENDIF
      IF (YES(19,'Y')) THEN
        PRBFIL=.TRUE.
        IF (MODULE.EQ.'C') GOTO 5
        LIBNAM='PROBLEM*LIB'
        INSET=' PROBLEM '
        GOTO 5
      ENDIF
C
      IF (MODULE.EQ.'C') THEN
        WRITE(1,135)
135     FORMAT('  STATE EQUATION PROBLEM FILES? (YES): ',$)
      ELSE
        WRITE(1,130)
130     FORMAT('  RESULTS FILES? (YES): ',$)
      ENDIF
      IF (YES(19,'Y')) THEN
        PRBFIL=.FALSE.
        IF (MODULE.EQ.'C') GOTO 5
        LIBNAM='RESULTS*LIB'
        INSET=' RESULTS '
        GOTO 5
      ENDIF
      GOTO 100
C
5     WRITE(1,1010) INSET,INSET,INSET,INSET
1010  FORMAT(/A9,'FILER OPTIONS',
     7 /' ---------------------------------------',
     A /' S: SAVE THE',A9,'IN A FILE',
     B /' U: UNSAVE (LOAD) A',A9,'FILE',
     1 /' C: CHECK DIRECTORY AGAINST EXISTING FILES',
     2 /' L: LIST THE DIRECTORY CONTENTS',
     3 /' D: DELETE A',A9,'FILE',
     5 /' A: ADD A FILE TO THE DIRECTORY',
     4 /' R: RETURN TO THE MAIN MENU (=DEFAULT)',
     8 /' ---------------------------------------',
     6 /' ENTER OPTION (R): ',$)
C
      CALL MENU(OK,IX,'S','U','C','L','D','A','R','',
     2 '
      IF (.NOT.OK) GOTO 5
      GOTO (10,20,30,40,50,60,70) IX
C
C--- SAVE A PROBLEM OR RESULTS
```

```
C
10    CALL SAVFIL
      GOTO 5
C
C--- LOAD A PROBLEM OR RESULTS
C
20    CALL LODFIL
      GOTO 5
C
C--- CHECK DIRECTORY
C
30    CALL CHKLIB
      GOTO 5
C
C--- LIST THE DIRECTORY
C
40    CALL LSTLIB
      GOTO 5
C
C--- DELETE A FILE
C
50    CALL DELFIL
      GOTO 5
C
C--- ADD A FILE
C
60    CALL ADAFIL
      GOTO 5
C
C--- RETURN TO MAIN MENU
C
70    MODFLG=.TRUE.
      RETURN
      END
C
C
C****************************** LSTLIB ******************************
C*********************************************************************
C
C
C  LSTLIB PRINTS A LIST OF THE STORED PROBLEM OR RESULTS FILE
C  LIBRARIES ON THE USER SCREEN.  LIBRARY NAMES ARE STORED
C  IN THE FILE LIBNAM. EACH PROBLEM OR RESULTS SET IS FILED
C  IN ITS OWN PERMANENT FILE.
C  CREATED BY J. GREGORSKI JANUARY,1985
C
      SUBROUTINE LSTLIB
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      LOGICAL FXST,FOUND,YES
      CHARACTER*9 MODNAM(5)
      CHARACTER*3 ANSWER
      LBLUN=8
```

```
C
C--- ALL 5 FOR COMPAT WITH ALL MODULES
C--- BOTH PROBLEMS AND RESULTS
C
      MODNAM(1)='T OR F:  '
      MODNAM(2)='C:       '
      MODNAM(3)='F:       '
      MODNAM(4)='T:       '
      MODNAM(5)='T,F OR C:'
C
C--- DETERMINE PROPER HEADING FOR FILE LISTING
C--- HEADERS SET BY INIT. PROCESS FOR MODULE (SETUP)
C
      IF (PRBFIL) THEN
        JXK=HEADER(1)
      ELSE
        JXK=HEADER(2)
      ENDIF
C
C--- SEE IF THERE ARE ANY LIBRARIES AT ALL
C
      INQUIRE(FILE=LIBNAM,EXIST=FXST)
C
      IF(FXST) THEN
        OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
     2  FORM='UNFORMATTED')
        READ(LBLUN,END=990,ERR=990) NUMUFL
        IF(NUMUFL.GT.0) THEN
          READ(LBLUN,END=990,ERR=990)(FLNAM(I),I=1,NUMUFL)
          READ(LBLUN,END=990,ERR=990)(TITLE(I),I=1,NUMUFL)
          READ(LBLUN,END=990,ERR=990)(TYPE(I),I=1,NUMUFL)
          CLOSE(LBLUN,STATUS='KEEP')
C
          FOUND=.FALSE.
          WRITE(1,180) MODNAM(JXK)
180       FORMAT(/' FILES FOR RELOADING MODULE ',A9,/,
     2            ' ------------------------------------')
          NLINE=3
          DO 200 I=1,NUMUFL
             IF ((((.NOT.PRBFIL).AND.(TYPE(I).EQ.PRBTYP))
     2       .OR.((PRBFIL).AND.(TYPE(I).EQ.WANTYP(1)))
     3       .OR.((PRBFIL).AND.(TYPE(I).EQ.WANTYP(2))))
     4       .OR.(TYPE(I).EQ.0)) THEN
                WRITE(1,201)FLNAM(I),TITLE(I)
201             FORMAT(/2X,A15,/,1X,A40)
                FOUND=.TRUE.
                NLINE=NLINE+3
                IF (NLINE.GE.25) THEN
                  WRITE(1,1010)
1010              FORMAT(/,' WANT TO SEE MORE? (YES): ',$)
                  IF (.NOT.YES(25,'Y')) GOTO 999
                  WRITE(1,*) '  '
                  NLINE=0
                ENDIF
```

```
            ENDIF
200         CONTINUE
            IF (.NOT.FOUND) THEN
               WRITE(1,182)
182            FORMAT(' (NO SUCH FILES IN THE LIBRARY)')
            ENDIF
C
          ELSE
            WRITE(1,1000)
1000        FORMAT(/' THERE ARE NO USER LIBRARY FILES.')
          ENDIF
C
C--- LIBNAM DOESNT EXIST, USE CHKLIB TO CREATE IT
C
        ELSE
          CALL CHKLIB
          WRITE(1,1000)
        ENDIF
        GOTO 999
C
990     WRITE(1,991)
991     FORMAT(//'           *** ERROR ***',/,
       2 ' THE LIBRARY OF FILES IS UNREADABLE!')
C
999     WRITE(1,995)
995     FORMAT(//,' HIT <RETURN> TO RETURN TO MENU. ',$)
        READ(1,'(A3)') ANSWER
        RETURN
        END
C
C
C****************************** DELFIL ********************************
C**********************************************************************
C
C
C  USED TO DELETE A PROBLEM OR RESULTS FILE FROM THE LIBRARY.
C  CREATED BY J. GREGORSKI JANUARY,1985
C
        SUBROUTINE DELFIL
C
        INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
        LOGICAL EXST,YES,EXIT
        CHARACTER*3 ANSWER
        LBLUN=8
C
C
1       CONTINUE
C
        WRITE(1,1000)
1000    FORMAT(/' *** ENTER NAME OF FILE TO BE DELETED *** ')
        CALL FNAME(NAME,EXIT)
        IF (EXIT) THEN
          WRITE(1,1020)
```

```
1020    FORMAT(/' DELETE EXITED. NO CHANGES MADE. ')
        RETURN
      ENDIF
C
C--- SEE IF NAME EXISTS
C
      INQUIRE(FILE=LIBNAM,EXIST=EXST)
      IF(.NOT.EXST) THEN
        WRITE(1,1010)
1010    FORMAT(/' *** NO FILES IN LIBRARY TO DELETE ***')
        WRITE(1,1020)
        RETURN
      ELSE
        OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
     2  FORM='UNFORMATTED')
        READ(LBLUN,END=990,ERR=990) NUMUFL
        IF(NUMUFL.GT.0) THEN
          READ(LBLUN,END=990,ERR=990)(FLNAM(I),I=1,NUMUFL)
          READ(LBLUN,END=990,ERR=990)(TITLE(I),I=1,NUMUFL)
          READ(LBLUN,END=990,ERR=990)(TYPE(I),I=1,NUMUFL)
          CLOSE(LBLUN,STATUS='KEEP')
          NLB=0
          DO 100 I=1,NUMUFL
            IF(FLNAM(I).EQ.NAME) THEN
              NLB=1
            ENDIF
  100     CONTINUE
        ELSE IF (NUMUFL.LT.0) THEN
          CLOSE(LBLUN,STATUS='KEEP')
          GOTO 990
        ELSE
          CLOSE(LBLUN,STATUS='KEEP')
          WRITE(1,1010)
          WRITE(1,1020)
          RETURN
        ENDIF
      ENDIF
C
      IF (NLB.EQ.0) THEN
        WRITE(1,1050)
1050    FORMAT(/' NAME DOESN''T EXIST, TRY AGAIN.')
        GOTO 1
      ELSE
        WRITE(1,1060)
1060    FORMAT(/' FILE FOUND. DELETE IT? (NO): ',$)
        IF (.NOT.YES(25,'N')) THEN
          GOTO 1
        ELSE
C
C--- HE REALLY WANTS TO DELETE IT!
C
          OPEN(9,FILE=NAME,STATUS='UNKNOWN',FORM='UNFORMATTED')
          CLOSE(9,STATUS='DELETE')
C
```

```
C--- REWRITE LIBNAM
C
          OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='UNKNOWN',
     2    FORM='UNFORMATTED')
          CLOSE(LBLUN,STATUS='DELETE')
C
          OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='NEW',
     2    FORM='UNFORMATTED')
          NUMUFL=NUMUFL-1
          DO 300 I=1,NUMUFL
            IF (I.GE.NLB) THEN
              FLNAM(I)=FLNAM(I+1)
              TITLE(I)=TITLE(I+1)
              TYPE(I)=TYPE(I+1)
            ENDIF
300       CONTINUE
          WRITE(LBLUN,ERR=990) NUMUFL
          WRITE(LBLUN,ERR=990)(FLNAM(I),I=1,NUMUFL)
          WRITE(LBLUN,ERR=990)(TITLE(I),I=1,NUMUFL)
          WRITE(LBLUN,ERR=990)(TYPE(I),I=1,NUMUFL)
          ENDFILE(LBLUN)
          CLOSE(LBLUN,STATUS='KEEP')
        ENDIF
      ENDIF
C
C--- WANT TO TRY ANOTHER ONE.
C
      WRITE(1,3000)
3000  FORMAT(/' *** FILE DELETION COMPLETED ***')
      WRITE(1,1080)
1080  FORMAT(/' WANT TO DELETE ANOTHER FILE? (NO): ',$)
      IF (.NOT.YES(25,'N')) THEN
        GOTO 999
      ELSE
        GOTO 1
      ENDIF
C
990   WRITE(1,991)
991   FORMAT(//'          *** ERROR ***',/,
     2 ' THE LIBRARY OF FILES IS UNREADABLE!')
C
999   WRITE(1,995)
995   FORMAT(//,' HIT <RETURN> TO RETURN TO MENU. ',$)
      READ(1,'(A3)') ANSWER
      RETURN
      END
C
C
C
C****************************** CHKLIB ******************************
C********************************************************************
C
C
C  CHKLIB UPDATES THE LIBRARY NAME FILE, LIBNAM. IF IT DOESN'T
```

```
C  EXIST IT IS CREATED.  ANY FILES THAT DO NOT EXIST ARE
C  DELETED FROM LIBNAM.
C  CREATED BY J. GREGORSKI JANUARY,1985
C
      SUBROUTINE CHKLIB
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      CHARACTER*40 STOTTL
      CHARACTER*15 STONAM
      INTEGER STOTYP
      CHARACTER*3 ANSWER
      LOGICAL EXST
      LBLUN=8
C
      WRITE(1,1001) LIBNAM
1001  FORMAT(/' CHECKING DIRECTORY STATUS: ',A)
C
      INQUIRE(FILE=LIBNAM,EXIST=EXST)
      IF(.NOT.EXST) THEN
        OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='NEW',
     2  FORM='UNFORMATTED')
        NUMUFL=0
        WRITE(LBLUN,ERR=990)NUMUFL
        ENDFILE(LBLUN)
        CLOSE(LBLUN,STATUS='KEEP')
        WRITE(1,1000) LIBNAM
1000    FORMAT(/' *** DIRECTORY FILE CREATED: ',A)
C
C--- LIBNAM EXISTS. CHECK DIRECTORY AGAINST USER FILES.
C
      ELSE
        OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
     2  FORM='UNFORMATTED')
        READ(LBLUN,END=990,ERR=990) NUMUFL
        IF (NUMUFL.EQ.0) THEN
          CLOSE(LBLUN,STATUS='KEEP')
          WRITE(1,1003)
1003      FORMAT(/' THE DIRECTORY IS EMPTY.')
        ELSE IF (NUMUFL.LT.0) THEN
          CLOSE(LBLUN,STATUS='KEEP')
          GOTO 990
        ELSE
          READ(LBLUN,END=990,ERR=990)(FLNAM(I),I=1,NUMUFL)
          READ(LBLUN,END=990,ERR=990)(TITLE(I),I=1,NUMUFL)
          READ(LBLUN,END=990,ERR=990)(TYPE(I),I=1,NUMUFL)
          CLOSE(LBLUN,STATUS='KEEP')
          NEX=0
          DO 100 I=1,NUMUFL
            INQUIRE(FILE=FLNAM(I),EXIST=EXST)
            IF(EXST) THEN
              NEX=NEX+1
              STONAM=FLNAM(I)
              STOTTL=TITLE(I)
```

```
              STOTYP=TYPE(I)
              FLNAM(NEX)=STONAM
              TITLE(NEX)=STOTTL
              TYPE(NEX)=STOTYP
              WRITE(1,1005) FLNAM(I)
1005          FORMAT(4X,' FILE FOUND: ',A)
            ELSE
              WRITE(1,1010) FLNAM(I)
1010          FORMAT(4X,' FILE  MISSING: ',A,
     1         /4X,' IT WILL BE DELETED FROM THE DIRECTORY.')
            ENDIF
100         CONTINUE
C
C--- IF DIRECTORY IS CORRECT, NEX=NUMUFL
C
          IF (NEX.NE.NUMUFL) THEN
            OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
     2      FORM='UNFORMATTED')
            CLOSE(LBLUN,STATUS='DELETE')
            OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='NEW',
     2      FORM='UNFORMATTED')
            WRITE(LBLUN,ERR=990)NEX
            WRITE(LBLUN,ERR=990)(FLNAM(I),I=1,NEX)
            WRITE(LBLUN,ERR=990)(TITLE(I),I=1,NEX)
            WRITE(LBLUN,ERR=990)(TYPE(I),I=1,NEX)
            ENDFILE(LBLUN)
            CLOSE(LBLUN,STATUS='KEEP')
          ENDIF
        ENDIF
      ENDIF
      GOTO 999
C
990   WRITE(1,991)
991   FORMAT(//'          *** ERROR ***',/,
     2 ' THE LIBRARY OF FILES IS UNREADABLE!')
C
999   WRITE(1,995)
995   FORMAT(//,' HIT <RETURN> TO RETURN TO MENU. ',$)
      READ(1,'(A3)') ANSWER
      RETURN
      END
C
C
C
C***************************** ADAFIL ******************************
C*******************************************************************
C
C
C  ADDS AN EXISTING FILE TO THE DIRECTORY LIST.
C  CREATED BY J. GREGORSKI JANUARY,1985
C
      SUBROUTINE ADAFIL
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
```

```
C
      LOGICAL EXST,YES,EXIT
      CHARACTER*3 ANSWER
      CHARACTER*40 ADDTTL
      CHARACTER*8 ENDTTL
      LBLUN=8
C
C--- GENERATE END OF DEFAULT TITLE FOR
C--- COMPATIBILITY WITH ALL MODULES.
C
      ENDTTL='MODULE '//MODULE
C
C--- ASSEMBLE DEFAULT TITLE FOR ADDED FILES
C
      WRITE(ADDTTL,2000) LIBNAM,ENDTTL
2000  FORMAT('*** ADDED TO ',A11,' BY ',A8,' ***')
C
1     CONTINUE
C
      WRITE(1,1000)
1000  FORMAT(/' *** ENTER NAME OF FILE TO BE ADDED ***')
      CALL FNAME(NAME,EXIT)
      IF (EXIT) THEN
        WRITE(1,1020)
1020    FORMAT(/' ADD FILE EXITED. NO CHANGES MADE.')
        RETURN
      ENDIF
C
C--- SEE IF FILE NAME EXISTS
C
      INQUIRE(FILE=NAME,EXIST=EXST)
      IF (.NOT.EXST) THEN
        WRITE(1,1010) NAME
1010    FORMAT(/' NO SUCH FILE FOUND: ',A)
        GOTO 1
      ENDIF
C
C--- SEE IF NAME EXISTS IN LIBRARY
C
      INQUIRE(FILE=LIBNAM,EXIST=EXST)
      IF(.NOT.EXST) THEN
        CALL CHKLIB
      ELSE
        OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
     2  FORM='UNFORMATTED')
        READ(LBLUN,END=990,ERR=990) NUMUFL
        IF(NUMUFL.GT.0) THEN
          READ(LBLUN,END=990,ERR=990)(FLNAM(I),I=1,NUMUFL)
          READ(LBLUN,END=990,ERR=990)(TITLE(I),I=1,NUMUFL)
          READ(LBLUN,END=990,ERR=990)(TYPE(I),I=1,NUMUFL)
          CLOSE(LBLUN,STATUS='KEEP')
          DO 100 I=1,NUMUFL
            IF(FLNAM(I).EQ.NAME) THEN
              WRITE(1,1040)
```

```
1040          FORMAT(/' THIS FILE ALREADY LISTED IN DIRECTORY.')
              GOTO 1
            ENDIF
100       CONTINUE
        ELSE IF (NUMUFL.LT.0) THEN
          CLOSE(LBLUN,STATUS='KEEP')
          GOTO 990
        ELSE
          CLOSE(LBLUN,STATUS='KEEP')
        ENDIF
      ENDIF
C
C--- ADD THE FILE NAME TO DIRECTORY
C
      NUMUFL=NUMUFL+1
      IF (NUMUFL.GT.MAXNUF) THEN
        NUMUFL=MAXNUF
        WRITE(1,1050) NUMUFL
1050    FORMAT(/' *** DIRECTORY IS FULL AT ',I4,' FILES.')
        WRITE(1,1020)
        RETURN
      ENDIF
C
C--- ADD DEFAULT TITLE AND NEUTRAL PRBTYP(=0)
C--- ALLOWS LISTING OF FILE BY ALL MODULES
C
      FLNAM(NUMUFL)=NAME
      TITLE(NUMUFL)=ADDTTL
      TYPE(NUMUFL)=0
C
C--- REWRITE LIBNAM WITH NEW FILE NAME ADDED
C
      OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
     2 FORM='UNFORMATTED')
      CLOSE(LBLUN,STATUS='DELETE')
C
      OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='NEW',
     2 FORM='UNFORMATTED')
      WRITE(LBLUN,ERR=990) NUMUFL
      WRITE(LBLUN,ERR=990)(FLNAM(I),I=1,NUMUFL)
      WRITE(LBLUN,ERR=990)(TITLE(I),I=1,NUMUFL)
      WRITE(LBLUN,ERR=990)(TYPE(I),I=1,NUMUFL)
      ENDFILE(LBLUN)
      CLOSE(LBLUN,STATUS='KEEP')
C
C--- WANT TO TRY ANOTHER ONE.
C
      WRITE(1,3000)
3000  FORMAT(/' *** FILE ADDITION COMPLETED ***')
      WRITE(1,1080)
1080  FORMAT(/' WANT TO ADD ANOTHER FILE? (NO): ',$)
      IF (.NOT.YES(25,'N')) THEN
        GOTO 999
      ELSE
```

```
        GOTO 1
      ENDIF
C
990   WRITE(1,991)
991   FORMAT(//'          *** ERROR ***',/,
     2 ' THE LIBRARY OF FILES IS UNREADABLE!')
C
999   WRITE(1,995)
995   FORMAT(//,' HIT <RETURN> TO RETURN TO MENU. ',$)
      READ(1,'(A3)') ANSWER
      RETURN
      END
C
C
C
C****************************** SAVFIL ******************************
C*********************************************************************
C
C
C  SAVES THE PROBLEM DESCRIPTION OR RESULTS IN A USER NAMED FILE.
C  CREATED BY J. GREGORSKI JANUARY,1985
C
      SUBROUTINE SAVFIL
C
      INCLUDE 'SYSKIT>COMMON.DIR>CTRLBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
      INCLUDE 'SYSKIT>COMMON.DIR>LABLBK.TEXT'
C
      CHARACTER*3 ANSWER
      LOGICAL LBXST,READIN,EXIT,YES,INLIB,EXST,ERROR
      INTEGER NFILE,FILTYP
      LBLUN=8
C
2     CONTINUE
      INLIB=.FALSE.
C
C--- CALL FOR THE FILE NAME
C
      WRITE(1,1010)
1010  FORMAT(/' *** ENTER NAME FOR THE SAVED FILE ***')
      CALL FNAME(NAME,EXIT)
      IF (EXIT) RETURN
C
C--- MAKE SURE NAME FILE EXISTS TO BE ACCESSED
C
15    INQUIRE(FILE=LIBNAM,EXIST=LBXST)
      IF(.NOT.LBXST) THEN
        CALL CHKLIB
        GOTO 15
      ENDIF
C
      OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
     2 FORM='UNFORMATTED')
      READ(LBLUN,END=990,ERR=990) NUMUFL
```

```
      IF(NUMUFL.EQ.0) THEN
        CLOSE(LBLUN,STATUS='KEEP')
        GOTO 600
      ELSEIF (NUMUFL.EQ.MAXNUF) THEN
        WRITE(1,1015)
1015    FORMAT(/' SORRY. NO MORE ROOM IN DIRECTORY FILE.')
        CLOSE(LBLUN,STATUS='KEEP')
        GOTO 999
      ELSEIF ((NUMUFL.LT.0).OR.(NUMUFL.GT.MAXNUF)) THEN
        CLOSE(LBLUN,STATUS='KEEP')
        GOTO 990
      ELSE
        READ(LBLUN,END=990,ERR=990)(FLNAM(I),I=1,NUMUFL)
        READ(LBLUN,END=990,ERR=990)(TITLE(I),I=1,NUMUFL)
        READ(LBLUN,END=990,ERR=990)(TYPE(I),I=1,NUMUFL)
        CLOSE(LBLUN,STATUS='KEEP')
      ENDIF
C
C--- CHECK IF FILE IS IN LIBRARY AND EXISTENCE OF FILE
C
      DO 100 I=1,NUMUFL
        IF (FLNAM(I).EQ.NAME) THEN
          INLIB=.TRUE.
          NFILE=I
          GOTO 600
        ENDIF
100   CONTINUE
C
600   INQUIRE(FILE=NAME,EXIST=EXST)
C
C--- IF FILE EXISTS, CHECK IF USER WANTS TO OVERWRITE
C--- AN IMPLIED OVERWRITE IF FILE DOES NOT EXIST
C--- ASK FOR ANOTHER FILE NAME IF NO OVERWRITE
C
      IF (EXST) THEN
        IF (INLIB) WRITE(1,1020)
1020    FORMAT(/' THIS FILE ALREADY EXISTS. SHALL I',
     2   /' OVERWRITE IT WITH A NEW ONE? (NO): ',$)
        IF (.NOT.INLIB) WRITE(1,1030)
1030    FORMAT(/' THIS FILE ALREADY EXISTS, BUT IT IS NOT',
     2  /' CURRENTLY LISTED IN THE FILE LIBRARY.',
     3  /' SHALL I OVERWRITE IT WITH A NEW ONE? (NO): ',$)
        IF (.NOT.YES(25,'N')) GOTO 2
      ENDIF
C
C--- INCREASE LIB. SIZE AND ADD FILE TO END
C--- IF FILE IS NOT LISTED IN THE LIBRARY
C
      IF (.NOT.INLIB) THEN
        NUMUFL=NUMUFL+1
        NFILE=NUMUFL
      ENDIF
C
C--- DETERMINE FILTYP FOR LISTING IN LIBRARY.
```

```
C
      IF ((MODULE.EQ.'C').AND.(PRBFIL)) THEN
        IF (SGLIFG) THEN
          FILTYP=WANTYP(2)
        ELSEIF (.NOT.SGLIFG) THEN
          FILTYP=WANTYP(1)
        ENDIF
      ELSE
        FILTYP=PRBTYP
      ENDIF
C
C--- REWRITE THE LIBRARY FILE.
C
      FLNAM(NFILE)=NAME
      TITLE(NFILE)=TITLE1
      TYPE(NFILE)=FILTYP
C
      OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
     2 FORM='UNFORMATTED')
      WRITE(LBLUN,ERR=990) NUMUFL
      WRITE(LBLUN,ERR=990)(FLNAM(I),I=1,NUMUFL)
      WRITE(LBLUN,ERR=990)(TITLE(I),I=1,NUMUFL)
      WRITE(LBLUN,ERR=990)(TYPE(I),I=1,NUMUFL)
      ENDFILE(LBLUN)
      CLOSE(LBLUN,STATUS='KEEP')
C
C--- OPEN NEW FILE AND WRITE PROBLEM OR RESULTS
C--- CALLS PROPER SUB. ACCORDING TO MODULE IN USE
C
      READIN=.FALSE.
C
      IF (PRBFIL) THEN
        IF (MODULE.EQ.'F') THEN
          CALL FPROB(NAME,READIN,ERROR)
        ELSEIF (MODULE.EQ.'T') THEN
          CALL TPROB(NAME,READIN,ERROR)
        ELSEIF (MODULE.EQ.'C') THEN
          CALL CPROB(NAME,READIN,ERROR)
        ENDIF
      ELSE
        IF (MODULE.EQ.'F') THEN
          CALL FRSLT(NAME,READIN,ERROR)
        ELSEIF (MODULE.EQ.'T') THEN
          CALL TRSLT(NAME,READIN,ERROR)
        ELSEIF (MODULE.EQ.'C') THEN
          CALL CRSLT(NAME,READIN,ERROR)
        ENDIF
      ENDIF
C
      IF (ERROR) GOTO 2
C
C--- STORAGE COMPLETED
C
      WRITE(1,3200)
```

```
3200  FORMAT(/' *** FILE STORED IN LIBRARY ***')
      GOTO 999
C
990   WRITE(1,991)
991   FORMAT(//'          *** ERROR ***',/,
     2 ' THE LIBRARY OF FILES IS UNREADABLE!')
C
999   WRITE(1,995)
995   FORMAT(//,' HIT <RETURN> TO CONTINUE. ',$)
      READ(1,'(A3)') ANSWER
      RETURN
C
      END
C
C
C
C****************************** LODFIL ********************************
C**********************************************************************
C
C
C  LOADS A PROBLEM OR RESULTS FROM A USER-NAMED FILE.
C  CREATED BY J. GREGORSKI JANUARY,1985
C
      SUBROUTINE LODFIL
C
      INCLUDE 'SYSKIT>COMMON.DIR>FILRBK.TEXT'
C
      LOGICAL LXST,FXST1,READIN,EXIT,ERROR
      LBLUN=8
C
C
C--- MAKE SURE DIRECTORY FILE EXISTS
C--- CREATE IT IF IT DOESN'T AND EXIT.
C
      INQUIRE(FILE=LIBNAM,EXIST=FXST1)
      IF(.NOT.FXST1) THEN
        CALL CHKLIB
        WRITE(1,1010)
1010    FORMAT(/' NO FILES IN LIBRARY, HENCE NO LOAD.')
        GOTO 999
      ENDIF
C
C--- SEE IF USER REQUEST IS IN DIRECTORY
C
10    WRITE(1,1020)
1020  FORMAT(/' *** ENTER NAME OF FILE TO BE LOADED *** ')
      CALL FNAME(NAME,EXIT)
      IF (EXIT) THEN
        WRITE(1,1025)
1025    FORMAT(/' LOADER EXITED. NO PROBLEM LOADED. ')
        RETURN
      ENDIF
C
      OPEN(LBLUN,FILE=LIBNAM,ERR=990,STATUS='OLD',
```

```
     2 FORM='UNFORMATTED')
      READ(LBLUN,END=990,ERR=990) NUMUFL
      IF(NUMUFL.GT.0) THEN
        READ(LBLUN,END=990,ERR=990)(FLNAM(I),I=1,NUMUFL)
        READ(LBLUN,END=990,ERR=990)(TITLE(I),I=1,NUMUFL)
        READ(LBLUN,END=990,ERR=990)(TYPE(I),I=1,NUMUFL)
        CLOSE(LBLUN,STATUS='KEEP')
        LXST=.FALSE.
        DO 100 I=1,NUMUFL
          IF(NAME.EQ.FLNAM(I)) LXST=.TRUE.
100     CONTINUE
      ELSE
        CLOSE(LBLUN,STATUS='KEEP')
        LXST=.FALSE.
      ENDIF
C
C--- IF USER FILE EXISTS LOAD IT IN
C
      IF(LXST) THEN
        INQUIRE(FILE=NAME,EXIST=LXST)
        IF (.NOT.LXST) THEN
          WRITE(1,1030)
1030      FORMAT(/' THIS FILE DOES NOT EXIST.')
          GOTO 10
        ENDIF
C
        READIN=.TRUE.
C
        IF (PRBFIL) THEN
          IF (MODULE.EQ.'F') THEN
            CALL FPROB(NAME,READIN,ERROR)
          ELSEIF (MODULE.EQ.'T') THEN
            CALL TPROB(NAME,READIN,ERROR)
          ELSEIF (MODULE.EQ.'C') THEN
            CALL CPROB(NAME,READIN,ERROR)
          ENDIF
        ELSE
          IF (MODULE.EQ.'F') THEN
            CALL FRSLT(NAME,READIN,ERROR)
          ELSEIF (MODULE.EQ.'T') THEN
            CALL TRSLT(NAME,READIN,ERROR)
          ELSEIF (MODULE.EQ.'C') THEN
            CALL CRSLT(NAME,READIN,ERROR)
          ENDIF
        ENDIF
C
        IF (ERROR) GOTO 10
C
        WRITE(1,1040)
1040    FORMAT(/' *** USER FILE HAS BEEN LOADED ***')
      ELSE
        WRITE(1,1060)
1060    FORMAT(/' THIS FILE NAME NOT IN DIRECTORY.')
        GOTO 10
```

```
      ENDIF
      GOTO 999
C
990   WRITE(1,991)
991   FORMAT(//'           *** ERROR ***',/,
     2 ' THE LIBRARY OF FILES IS UNREADABLE!')
C
999   WRITE(1,995)
995   FORMAT(//,' HIT <RETURN> TO CONTINUE. ',$)
      READ(1,'(A3)') ANSVER
      RETURN
      END
C
C
C*************************************************************************
C*************************************************************************
C
C
```

Appendix E

COMMON BLOCKS LISTINGS

```
C
C
C*** CTRLBK :  COMMON BLOCK TO CONTROL MAIN PROGRAM FLOW *********
C***           ALSO CONTAINS PARAMETER DEFINITIONS       *********
C*****************************************************************
C
C--- VARIABLES
C
C     SYSFLG    PROCEED FROM INSTRUCTIONS? YES=.FALSE.,NO=.TRUE.
C     MODFLG    PROCEED IN MAIN (.FALSE.),GOTO MENU (.TRUE.)
C     FRQYES    FREQUENCY RESPONSE (.TRUE.),ROOT LOCUS (.FALSE.)
C     IBLK      CONTROL FLOW IN MAIN,LOOK FOR END OF PROGRESSION
C     SGLIFG    HIGH ORDER DIFF. EQU.(.TRUE.),VIBS. FORM(.FALSE.)
C     FBKFLG    FEEDBACK EFFECTS IN A,B PROB. IF (.TRUE.)
C     MAXDEG    MAXIMUM DEGREE OF POLYNOMIALS (BOTH NUM. & DEN.)
C     MAXCOF    MAXIMUM NUMBER OF POLYNOMIAL COEFFICIENTS
C     MAXFRP    MAXIMUM NUMBER OF FREQ. RESP. RESULT POINTS
C     MAXRLP    MAXIMUM NUMBER OF ROOT LOCUS RESULT POINTS
C     MAXDIM    MAXIMUM DIMENSION OF STATE SPACE PROBLEM
C
      PARAMETER (MAXDEG=10,MAXCOF=MAXDEG+1,MAXFRP=500,
     +           MAXRLP=100,MAXDIM=10)
C
      INTEGER IBLK
      LOGICAL SYSFLG,MODFLG,FRQYES,SGLIFG,FBKFLG
C
      COMMON/CTRLBK/SYSFLG,MODFLG,FRQYES,SGLIFG,FBKFLG,IBLK
C
C*** END OF CTRLBK ************************************************
C
C
C
C
C*** LABLBK :  COMMON BLOCK FOR PROBLEM & RESULTS TITLES *********
C*****************************************************************
C
C--- VARIABLES
```

```
C
C     TITLE1    TITLE FOR PROBLEM STATEMENT
C     TITLE2    TITLE FOR RESULTS
C
      CHARACTER*40 TITLE1,TITLE2
C
      COMMON/LABLBK/TITLE1,TITLE2
C
C*** END OF LABLBK ***************************************************
C
C
C
C
C*** POLYBK :  COMMON BLOCK FOR DESCRIPTION OF POLYNOMIAL ********
C**********************************************************************
C
C--- VARIABLES
C
C     INDEG      DEGREE OF NUMERATOR POLYNOMIAL
C     IDDEG      DEGREE OF DENOMINATOR POLYNOMIAL
C     COEFFN     COEFFICIENTS OF NUMERATOR POLYNOMIAL
C     COEFFD     COEFFICIENTS OF DENOMINATOR POLYNOMIAL
C     GAIN       GAIN OF POLYNOMIAL
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      REAL COEFFN(MAXCOF),COEFFD(MAXCOF),GAIN
      INTEGER INDEG,IDDEG
C
      COMMON/POLYBK/COEFFN,COEFFD,INDEG,IDDEG,GAIN
C
C*** END OF POLYBK ***************************************************
C
C
C
C
C*** RANGBK :  COMMON BLOCK FOR OUTPUT RANGE AND SCALING *********
C**********************************************************************
C
C--- VARIABLES
C
C     SCMIN      MINIMUM VALUE OF FREQUENCY OR GAIN SCALE
C     SCMAX      MAXIMUM VALUE OF FREQUENCY OR GAIN SCALE
C     BLO        MINIMUM ACCEPTABLE VALUE FOR REAL NUMBER INPUT
```

```
C     BHI        MAXIMUM ACCEPTABLE VALUE FOR REAL NUMBER INPUT
C     NPTS       NUMBER OF SOLUTION POINTS FOR CALCULATIONS
C     SCLOG      SCALING OF FREQUENCIES OR GAINS (LOG IF =.TRUE.)
C
      REAL SCMIN,SCMAX,BLO,BHI
      INTEGER NPTS
      LOGICAL SCLOG
C
      COMMON/RANGBK/SCMIN,SCMAX,NPTS,SCLOG,BLO,BHI
C
C*** END OF RANGBK ***************************************************
C
C
C
C
C*** RESULT :  COMMON BLOCK FOR FREQUENCY RESPONSE RESULTS *******
C**********************************************************************
C
C--- VARIABLES
C
C     W          FREQUENCIES OF RESULTS POINTS
C     TRFMAG     MAGNITUDES OF RESPONSE
C     PHID       PHASE SHIFTS OF RESPONSE IN DEGREES
C     PHI        PHASE SHIFTS OF RESPONSE IN RADIANS
C     WGC        CROSSOVER FREQUENCY (W FOR MAG.=1)
C     PM         PHASE MARGIN (180+PHASE SHIFT AT WGC)
C     WPHC       FREQUENCY AT -180 PHASE SHIFT
C     GM         GAIN MARGIN (INVERSE OF MAG. AT WPHC)
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      REAL W(MAXFRP),TRFMAG(MAXFRP),PHID(MAXFRP),PHI(MAXFRP),
     +     WGC,PM,WPHC,GM
C
      COMMON/RESULT/W,TRFMAG,PHID,PHI,WGC,PM,WPHC,GM
C
C*** END OF RESULT ***************************************************
C
C
C
C
C
C*** ROOTBK :  COMMON BLOCK FOR ROOTS (& POLES) OF PROBLEM *******
C**********************************************************************
C
```

```
C--- VARIABLES
C
C     RNR        REAL PARTS OF NUMERATOR ROOTS
C     RNI        IMAGINARY PARTS OF NUMERATOR ROOTS
C     RDR        REAL PARTS OF DENOMINATOR ROOTS (POLES)
C     RDI        IMAGINARY PARTS OF DENOMINATOR ROOTS (POLES)
C     SYGAIN     SYSTEM GAIN
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      REAL RNR(MAXDEG),RNI(MAXDEG),RDR(MAXDEG),RDI(MAXDEG),SYGAIN
C
      COMMON/ROOTBK/RNR,RNI,RDR,RDI,SYGAIN
C
C*** END OF ROOTBK ************************************************
C
C
C
C
C*** FACTBK :  COMMON BLOCK FOR FACTORED INPUT FORMAT ************
C*****************************************************************
C
C--- VARIABLES
C
C     NFCTFG     .TRUE. IF N(S) HAS VALID FACTORED DATA
C     DFCTFG     .TRUE. IF D(S) HAS VALID FACTORED DATA
C     NFNUM      NUMBER OF FACTORS IN THE NUMERATOR
C     NFDEN      NUMBER OF FACTORS IN THE DENOMINATOR
C     LNDEG      LIST OF DEGREES OF NUMERATOR FACTORS
C     LDDEG      LIST OF DEGREES OF DENOMINATOR FACTORS
C     CFNF       ARRAY OF COEFFICIENTS OF NUMERATOR FACTORS
C     CFDF       ARRAY OF COEFFICIENTS OF DENOMINATOR FACTORS
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      LOGICAL NFCTFG,DFCTFG
      REAL CFNF(MAXDEG+1,MAXDEG),CFDF(MAXDEG+1,MAXDEG)
      INTEGER NFNUM,NFDEN,LNDEG(MAXDEG),LDDEG(MAXDEG)
C
      COMMON/FACTBK/NFCTFG,DFCTFG,NFNUM,NFDEN,
     +              LNDEG,LDDEG,CFNF,CFDF
C
C*** END OF FACTBK ************************************************
C
C
```

```
C
C
C*** RSLTBK :  COMMON BLOCK FOR ROOT LOCUS RESULTS ***************
C*****************************************************************
C
C--- VARIABLES
C
C     RR         REAL PARTS OF ROOT LOCI
C     RI         IMAGINARY PARTS OF ROOT LOCI
C     KGAIN      GAINS FOR ROOT LOCI
C     ANGLE      ANGLES OF LOCI ASYMPTOTES
C     SIGMA      ORIGIN OF LOCI ASYMPTOTES
C     IPOLEX     POLE EXCESS (#POLES-#ZEROS)
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      INTEGER IPOLEX
      REAL RR(MAXDEG,MAXRLP),RI(MAXDEG,MAXRLP),KGAIN(MAXRLP),
     +     ANGLE(MAXDEG),SIGMA
C
      COMMON/RSLTBK/RR,RI,KGAIN,ANGLE,SIGMA,IPOLEX
C
C*** END OF RSLTBK ************************************************
C
C
C
C
C*** FOUTBK :  COMMON BLOCK FOR FREQUENCY RESPONSE OUTPUT ********
C*****************************************************************
C
C--- VARIABLES
C
C     NPTOUT     NUMBER OF OUTPUT (DISPLAY) POINTS
C     WOUT       FREQUENCIES FOR OUTPUT (DISPLAY) RANGE
C     MAGOUT     MAGNITUDES OF RESPONSE
C     PHDOUT     PHASE SHIFTS OF RESPONSE IN DEGREES
C     PHOUT      PHASE SHIFTS OF RESPONSE IN RADIANS
C     WLOG       SCALING FOR FREQUENCY (LOG SCALE IF =.TRUE.)
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      INTEGER NPTOUT
      LOGICAL WLOG
      REAL WOUT(MAXFRP),MAGOUT(MAXFRP),PHDOUT(MAXFRP),PHOUT(MAXFRP)
C
      COMMON/FOUTBK/NPTOUT,WOUT,MAGOUT,PHDOUT,PHOUT,WLOG
C
C*** END OF FOUTBK ************************************************
C
```

```
C
C
C
C*** ROUTBK :  COMMON BLOCK FOR ROOT LOCUS OUTPUT ****************
C*****************************************************************
C
C--- VARIABLES
C
C     RROUT      REAL PARTS OF ROOT LOCI FOR OUTPUT (DISPLAY)
C     RIOUT      IMAG. PARTS OF ROOT LOCI FOR OUTPUT (DISPLAY)
C     KOUT       GAINS OF ROOT LOCI FOR OUTPUT (DISPLAY)
C     AOUT       ASSYMPTOTE ANGLES OF LOCI
C     SOUT       ORIGIN OF LOCI ASSYMPTOTES
C     NOUT       NUMBER OF OUTPUT (DISPLAY) POINTS
C     IOUT       POLE EXCESS (#POLES-#ZEROS)
C     ZMAX       GREATEST POINT ON PLOT (FOR PLOTTING ASSYMPTOTES)
C     KLOG       LOCI SCALING (LOG SCALE IF =.TRUE.)
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      INTEGER IOUT,NOUT
      LOGICAL KLOG
      REAL RROUT(MAXDEG,MAXRLP),RIOUT(MAXDEG,MAXRLP),KOUT(MAXRLP),
     +     AOUT(MAXDEG),SOUT,ZMAX
C
      COMMON/ROUTBK/RROUT,RIOUT,KOUT,AOUT,SOUT,
     +              NOUT,IOUT,ZMAX,KLOG
C
C*** END OF ROUTBK ************************************************
C
C
C
C
C
C*** VIBRBK :  COMMON BLOCK FOR VIBRATION FORM OF O.D.E. *********
C*****************************************************************
C
C--- THE MATRICES ARE DIMENSIONED 6X6 SO THAT OUTMAT CAN BE USED.
C--- THEIR ACTUAL DIMENSIONS ARE 3X3. (MESSAGE KEPT FROM ORIGINAL)
C
C--- VARIABLES
C
C     MATM       MASS MATRIX
C     MATC       DAMPING MATRIX
C     MATK       STIFFNESS MATRIX
```

```
C     MATL       INPUT MATRIX
C     INVM       INVERSE OF THE M MATRIX
C     DIMM       DIMENSION OF M,C,K MATRICES
C     COLL       NUMBER OF COLUMNS OF THE L MATRIX
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      INTEGER DIMM,COLL
      REAL MATM(MAXDIM,MAXDIM),MATC(MAXDIM,MAXDIM),
     +     MATK(MAXDIM,MAXDIM),MATL(MAXDIM,MAXDIM),
     +     INVM(MAXDIM,MAXDIM)
C
      COMMON/VIBRBK/MATM,MATC,MATK,MATL,DIMM,COLL,INVM
C
C*** END OF VIBRBK ***************************************************
C
C
C
C
C********** SGLIBK :  COMMON BLOCK FOR HIGHER ORDER D.E. *********
C**********************************************************************
C
C--- VARIABLES
C
C     COEF       COEFFICIENTS OF HIGHER ORDER D.E.
C     INPT       COEFFICIENTS FOR INPUT D.E.
C     ORDER      ORDER OF DIFFERENTIAL EQUATION
C     INNMB      ORDER OF INPUT DERIVATIVES
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      REAL COEF(MAXDIM+1),INPT(MAXDIM)
      INTEGER ORDER,INNMB
C
      COMMON/SGLIBK/COEF,INPT,ORDER,INNMB
C
C*** END OF SGLIBK ***************************************************
C
C
C
C
C
C
C*** PARMBK :  COMMON BLOCK FOR STATE-SPACE FORMULATION **********
C***    ALSO CONTAINS REAL NUMBER INPUT LIMITS          **********
C**********************************************************************
```

```
C
C--- VARIABLES
C
C     A          THE A MATRIX FOR STATE-SPACE FORMULATION
C     B          ''  B      ''       ''       ''
C     C          ''  C      ''       ''       ''
C     D          ''  D      ''       ''       ''
C     IC         VECTOR NEEDED TO WRITE A,B PROBLEM
C     UDEF       ARRAY FOR INPUTS FOR A,B PROBLEM
C     BLO        MINIMUM ACCEPTABLE VALUE FOR REAL NUMBER INPUT
C     BHI        MAXIMUM      ''       ''       ''
C     DIMX       NUMBER OF STATES (DIMENSION OF A)
C     DIMU       NUMBER OF INPUTS (COLUMN DIM. OF B AND D)
C     DIMY       NUMBER OF OUTPUTS (ROW DIM. OF C AND D)
C     ITYPE      VECTOR NEEDED TO WRITE A,B PROBLEM
C     KFB        VECTOR NEEDED TO WRITE A,B PROBLEM
C     KGAIN      GAIN OF THE SYSTEM
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      INTEGER DIMX,DIMU,DIMY,ITYPE(MAXDIM)
      REAL A(MAXDIM,MAXDIM),B(MAXDIM,MAXDIM),C(MAXDIM,MAXDIM),
     +     D(MAXDIM,MAXDIM),IC(MAXDIM),UDEF(MAXDIM,10),
     +     BLO,BHI,KFB(MAXDIM),KGAIN
C
      COMMON/PARMBK/DIMX,DIMU,DIMY,A,B,C,D,IC,UDEF,
     +              ITYPE,BLO,BHI,KFB,KGAIN
C
C*** END OF PARMBK ***************************************************
C
C
C
C
C
C*** EIGNBK :  COMMON BLOCK FOR STORAGE OF EIGENVALUES ***********
C*********************************************************************
C
C--- VARIABLES
C
C     EVALR      REAL PARTS OF EIGENVALUES
C     EVALI      IMAG. PARTS OF EIGENVALUES
C
C--- CTRLBK MUST PRECEED THIS BLOCK FOR PARAMETER ASSIGNMENTS
C
      REAL EVALR(MAXDEG),EVALI(MAXDEG)
C
      COMMON/EIGNBK/EVALR,EVALI
C
C*** END OF EIGNBK ***************************************************
C
```

```
C
C
C
C*** INTGBK :  COMMON BLOCK FOR INTEGRATION PARAMETERS   *********
C*****************************************************************
C
C--- VARIABLES
C
C     TINIT      INITIAL VALUE OF INTEGRATION RANGE
C     TFINAL     FINAL     ''          ''
C     TINTEG     AN INTEGRATION PARAMETER
C     TSTORE      ''       ''       ''
C     NSTEP       ''      ''       ''
C     NSTOR       ''       ''       ''
C
      REAL TINIT,TFINAL,TINTEG,TSTORE
      INTEGER NSTEP,NSTOR
C
      COMMON/INTGBK/TINIT,TFINAL,TINTEG,TSTORE,NSTEP,NSTOR
C
C*** END OF INTGBK ************************************************
C
C
C
C
C*** TERMBK :  COMMON BLOCK FOR USER TERMINAL PARAMETERS *********
C*****************************************************************
C
C--- VARIABLES
C
C     NTERM      NUMBER OF THE TERMINAL
C     NTYPE      TYPE OF TERMINAL
C     NRATE      BAUD RATE OF THE TERMINAL
C     NERROR     ERROR FLAG FROM CALL TERM$$
C     PLOTIT     .TRUE. IF TERMINAL IS COMPATIBLE WITH GRAPHICS
C
      LOGICAL PLOTIT
      INTEGER NTERM,NTYPE,NRATE,NERROR
C
      COMMON/TERMBK/NTERM,NTYPE,NRATE,NERROR,PLOTIT
C
C*** END OF TERMBK ************************************************
C
C
```

```
C
C
C*** FILRBK :  COMMON BLOCKS FOR FILE MGT.(UNIFIL) ***************
C*****************************************************************
C
C--- VARIABLES
C
C     LIBNAM    NAME OF USER FILE DIRECTORY BEING USED
C     NUMUFL    NUMBER OF USER FILES
C     MAXNUF    MAXIMUM NUMBER OF USER FILES
C     FLNAM     NAMES OF USER FILES IN DIRECTORY
C     NAME      CURRENT FILE NAME
C     TYPE      TYPES OF DATA FILES
C     TITLE     TITLES FOR DATA FILES
C     WANTYP    TYPES OF FILES USABLE BY MODULE (2)
C     PRBTYP    FILE TYPE FOR SAVING
C     PRBFIL    PROBLEM FILES (.TRUE.),RESULTS FILES (.FALSE.)
C     HEADER    HEADERS FOR LIBRARY LIST OF LOADABLE FILES
C     MODULE    MODULE CURRENTLY IN USE; SET IN SETUP*
C
      PARAMETER (MAXNUF=40)
C
      CHARACTER*1 MODULE
      CHARACTER*11 LIBNAM
      CHARACTER*15 FLNAM(MAXNUF),NAME
      CHARACTER*40 TITLE(MAXNUF)
      INTEGER TYPE(MAXNUF),WANTYP(2),PRBTYP,NUMUFL,HEADER(2)
      LOGICAL PRBFIL
C
      COMMON/FILRBK1/FLNAM,TITLE,NAME,LIBNAM,MODULE
      COMMON/FILRBK2/NUMUFL,TYPE,WANTYP,PRBTYP,PRBFIL,HEADER
C
C*** END OF FILRBK ************************************************
C
C
```