SPECIFICATIONS OF A SOFTWARE ENVIRONMENT FOR THE COMPUTER-AIDED DESIGN OF CONTROL SYSTEMS

by

Michael Tessler

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering

Department of Electrical Engineering
McGill University
Montreal, Canada.

April 1985

ABSTRACT

In this thesis a software environment for the Computer-Aided Design (CAD) of multivariable control systems is specified. The main requirements for the CAD package was portability and user friendliness. This was achieved by incorporating software engineering techniques which determine how software should be specified, designed, implemented and documented. In addition, an examination of the target user and the design procedure was performed.

A portion of the package was implemented in order to validate the design specifications The scope of the implementation effort was limited to the development of the necessary software tools and a system identification module.

The result of the specification is a standardization of the structure of the CAD environment. This has several advantages such as increased maintainability, expandability and reliability, and in addition, eliminates redundant software development

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Dean Pierre R. Bélanger for his supervision and guidance of my work. I would, in addition like to thank him for his continued interest and encouragement in my work.

I am grateful to Michael Blauer for providing constructive criticism, useful discussions and support throughout this project.

I would like to thank several members of the Control Group for their friendship and stimulating discussions. Mark Readman, Alexis Aloneftis, Patrick Aboussouan, Michel Habib, Sean Meyn and Youssef Ghoneim.

I would also like to thank Yvan Leclerc and Mike Parker for providing assistance in using the computer facilities throughout this project.

This thesis is dedicated to Carmen and Eddie Tessler.

TABLE OF CONTENTS

design tasks but in addition, provides an environment whereby complex control algorithms are made available to the user unfamiliar with the many software details 4 . To accomplish this, several fields of study must be drawn from: computer science, software engineering, applied mathematics and control system engineering.

Presently, the development of CAD systems for control applications has followed the bottom-up philosophy. What this implies is that the emphasis is on the development of algorithms and the associated computational subroutines. Therefore, the development of the CAD system involves first assembling a set of computational subrouitnes and, subsequently constructing a user interaction to contain them. Several problems arise from this type of development. First, the computational software if not developed with integration in mind, will be difficult to incorporate in a package, since the data structures may be inconsistent. Secondly, the user interface may be difficult to use, since it has been designed to satisfy the needs of the computational software instead of the user's needs. Thirdly, this type of development leads to redundant development of many functions which are needed throughout the package. In addition, maintenance and expansion of such a system is quite difficult since they usually do not have a uniform structure.

As a result, the main objective of this thesis is to specify a software environment for the computer-aided design and analysis of multivariable control systems. In addition, the package should have the following features: portability, expandability, reliability and user friendliness. This is accomplished by incorporating software engineering concepts and techniques which determine how the software should be specified, designed, implemented and documented. Furthermore, an examination of the target user and the design task is required, since the success of the user interface depends on these features.

The thesis is divided as follows: Chapter 2 examines the use of CAD for control applications and in addition, a set of general package specifications are presented. Chapter 3 is an overview of the general design issues of a CAD system for control.

In chapter 4, the package specifications are detailed. A description of Module One is provided in Chapter 5. Chapter 6 provides a sample interactive session. Conclusions are provided in Chapter 7. Finally, Appendix A contains the documentation of the software tools developed during this project.

Notation: `Throughout this thesis this style of type is used to indicate software code.`

This cannot work because the number of equations is missing, and therefore we need ;

```
CALL solve ( A , X , B , N )
```

Some of the shortcoming of this second design are;

1. The matrix A must be destroyed; otherwise there is no working space for the subroutine solve to use.
2. The user is not warned if A is singular or if the subroutine solve failed.
3. The dimensions of the Fortran arrays for A, X, B are not given
4. If there are two systems of equations $Ax_1 = b_1$ and $Ax_2 = b_2$ with the same matrix, then A is factored twice.
5. The computation of A^{-1} is not efficient because of the lack of storage space.

Although this is a simple design, it lacks flexibility, error recovery and has Fortran language problems.

The following design has fewer shortcomings;

```
CALL solve ( A, X , B, N , I , J , K, M, M1 )
```

where

I is the number of rows of A, B

J is the number of columns of A

and the rows of X

K is the number of columns of X, B

M is a switch to compute A^{-1}

M1 is flag for the singularity of A

After deciding what level of flexibility to offer to the user the third question is whether to provide an interactive shell The advantage of the shell is that it would eliminate for the user the wasteful and tedious task of constructing a calling program whenever he wants to solve a set of equations Therefore an interactive shell is built around the computational software and will contain the Global Dimensions of the variables (a Fortran requirement), the interaction or prompts for the data values, the control data (i.e whether to compute the inverse of A) and error detection. The error detection ensures that the input data is of at least the right type (integer, real, character, range ..) which would of course abnormally terminate the program and must be avoided at all costs An example of such a program is shown below;

```
Program InterfaceEquationSolver
Declarations
Prompts User for A, B, N, M ...
CALL solve ( A, X , B, N , I , J , K, M, M1 )
IF ( M1  NE  0 ) THEN
  Print out result
ELSE
  Print error message
ENDIF
END
```

This example demonstrates the tradeoff between developing powerful software and software which is easy to use. The more flexibility the user is allowed the more powerful the software, however, this also leads to software which is difficult to use for someone who wants to solve a simple problem.

Therefore the most important elements for the development of an interactive shell are the problem definition and the subsequent interpretation of this definition into specific set of objectives.

It was decided that the main objective of this project would be to concentrate on the interactive shell instead of the computational software, taking advantage of the extensive computational resources embodied in available software libraries, such as LINPACK [6], EISPACK [7], and RICPACK ([8],[9]). All this software is available commercially. The important characteristics of control computation which are relevant to an interactive interface shell are ,

1. large data sets
2. heavy computational load
3. requires user 'decisions' or expertize
4. large input/output requirements

In terms of computational software, the objective is to use existing software libraries, and therefore little discussion is necessary If algorithms and computational software were to be designed in house, a quality management system would have to be designed and implemented to ensure reliability of the software.

2.2 Survey of Existing Control CAD packages

In this section, a survey of the development and use of computer packages in the field of control and systems engineering is presented. The earliest development of computer aided control software was initiated by educational institutions in need of teaching and research tools [10]. The industrial sector with the need for the higher productivity offered by a control computer aided design package was the second motivating force for development and by 1980 a great many packages had been designed and were in use by industry.

it allows the user to define his own macro commands. However, the added flexibility adds complexity to the problem of error detection.

Most existing packages although designed for the expert user are structured as a compromise between these interfaces, where for example a C/L is used but the arguments of the command are prompted for. See Table 2.1 for a brief list of existing CAD packages and their features.

NAME	SCOPE	INTERFACE	LANGUAGE	APPLICATION
CLADP	Analysis and Design of Multivariable Control Systems	Q/A	Fortran IV	industry/ academic
MATRIXx	Control design, System identification, data analysis and Simulation	C/L	Fortran 77	industry/ academic
ASTROM	Modelling, Identification, Analysis, Simulation and Design of MIMO Systems	C/L	Fortran	industry/ academic
UMIST	Analysis, Design and Simulation of MIMO Systems	Q/A	Fortran	industry/ academic

TABLE 2.1

In brief, the diversity of Control CAD software is great, where the distinguishing factors are capability and structure. As a result of the large number of packages there exists a great overlap of software and duplication of effort by many institutions. The recently published Oak Ridge National Laboratory (ORNL) report [8] cites these reasons as the main motivators for constructing a standard control systems package and assembling software from diverse sources.

2.3 General Specifications

The main objective of the Control CAD package is to create an environment for the analysis and design of multivariable control systems. After an analysis of existing CAD packages and the preliminary requirements of the Control CAD package the following design goals were adopted;

1. Package must be developed so that it can be used easily, efficiently and confidently by an occasional user, and
2. Allow the user to fully deploy his intuition, skill and experience while making use of powerful computational tools.

The second step in the design procedure was to derive a set of specific package guidelines which would be used in the actual development stage In the early stages of development it was decided that the main characteristics of the CAD package should be portability, expandability, reliability and user friendliness. After examining these characteristics the following design guidelines were drafted;

1. **Portability:** Package should be as portable as possible (written in Fortran 77).
2. **Modular Software :** Facilitates changes, extensions, and allows others to easily understand the structure of the package.
3. **Expandability:** The package should allow flexibility in adding new algorithms, relevent software and modifications to existing software.
4. **Error Detection:** Complete error detection should be provided to avoid frustration and loss of time for the user. Every level of the package should employ error detection routines and return to the user the source of the error.

Chapter 3

GENERAL DESIGN ISSUES

In the first stages of development an attempt was made to identify the many issues which would influence the design and implementation of the CAD package. In this chapter some of the general design issues are presented. The first section will examine the problem of portability and discuss ways to reduce the effect of non-portable segments. The question of user friendliness and interaction is discussed in the second section and in addition the error detection problem is examined. The third section deals with the software design methodology, and furthermore, a list of guidelines for proper documentation is included.

3.1 Portability

A program is said to be portable if it can be transported from one computer to another with relatively little effort. Portability is thus of key importance in software development. This is of course true in the development of the Control CAD package because of the varied environments under which the package is expected to operate. It is therefore the duty of the designer to devise ways to allow programs written for one computer to run on another.

As a first step in the study of portability let us examine how the application software interacts with the programming environment. In Figure 3.1 the programming environment is shown to be comprised of several independent layers, forming a hierarchical structure, where each level communicates to the layers directly below.

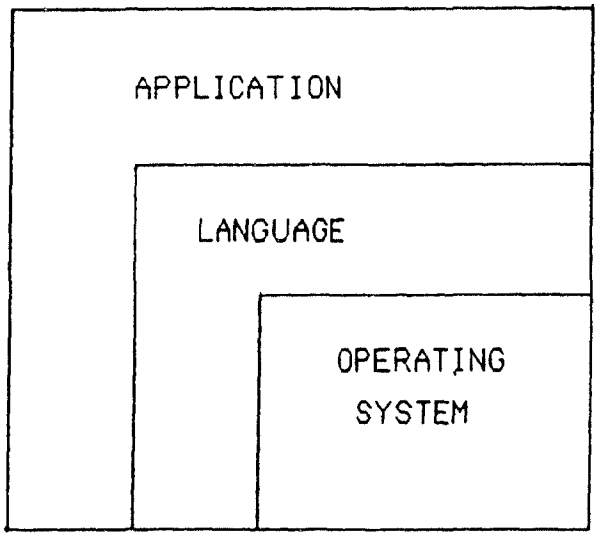


Fig 3.1

The highest layer is the application program which is dependent on the language layer, which in turn is dependent on the operating system services layer.

Although portability entails many programming details, a limited directed discussion of its effects on the package is undertaken. The approach chosen follows the previous discussion and hence this section is divided into the following areas:

1. programming language
2. operating system services

In addition a brief overview of the special problems related to the portability of graphical software is presented.

3.1.1 Programming Language

Fortran was the natural choice for the programming language given the characteristics of control problems and this selection has been recently substantiated by the recommendations of the ORNL report [8]. However, two problems of portability still exist. The first problem is the current standard of the Fortran language is Fortran 77 which is no more than a common subset and many installations offer language extensions, some of which are widely used. Therefore non-standard constructions must be avoided since they will run on different machines with different effects.

The second problem with respect to portability is the operating system dependent features of Fortran 77 [4]. These features are most evident when dealing with Fortran Input-Output commands. Therefore we shall limit ourselves to the operating system dependent features which affect the implementation of Fortran 77 I/O commands. What this means is that the designer need not concern himself with the existence of a particular I/O command since this is well documented the American National Standards Institute (ANSI) standard. However, the problem that the designer must address is the one of how the O/S handles the specific Fortran 77 I/O features

The first feature of the operating system which affects the implementation of Fortran 77 I/O commands is the directory structure used by the O/S. In order to minimize the effect of varied directory structures on portability, the directory structure of several operating systems are examined. Under the UNIX operating system the directory structure is tree-like and files are specified by pathnames. This approach is similar under the VMS (Digital Equipment Corp.) operating system where the directory structure is again tree-like but of course the descriptors are not the same. These two systems are completely different from some IBM operating systems where there is no tree structure and the reference to files is completely different. For example if a Fortran file 'filename' was to be stored under these different systems the three references would look like;

3.1.2 Operating System Services

The operating system offers a wide variety of services and these are available through the use of system calls resident in application programs. These services include input-output, error recovery and other general services (e.g random number generator).

In general to ensure portability the direct usage of operating system calls in application programs would have to be disallowed. However in some cases where features cannot be otherwise implemented, their limited use has to be accommodated.

To examine this issue let us examine two important areas of O/S services; run time error recovery and Input-Output. As specified in Section 2.3 the package should employ error detection and return to the user the source of the error What this means is that the package should be equipped with run time error recovery facilities. A run time error is produced when an abnormal event is detected by the processor, for example, a floating point exception. On most operating systems the default action is to clean up and abort, however the programmer may on some operating systems write an alternative error handling routine. However, as a result of the inability of Fortran 77 to trap or recover from run time errors, the use of operating system services for this purpose is necessary. In addition, it must be remembered that not only is the error recovery procedure non-portable but the cause of the error is machine dependent. For example, a floating point exception error can be caused by a floating point overflow which is dependent on the word length of the machine.

Unfortunately, the use of system services for error recovery introduces a non-portable portion and the problem is how to minimize its effects on the overall portability of the CAD package. The solution used was to segment the non-portable system calls by the creation of dummy subroutines. That is, within the package whenever error recovery is needed a call is made to a dummy subroutine which in turn contains the system call. This ensures that the call to the error recovery procedure would

not have to modified but only the contents of the dummy subroutine. This solution not only segments the non-portable code but in addition minimizes the effect on the overall package.

The second O/S service to be considered is I/O. Again as stated in Section 2.3 the requirement is to eliminate from the user the duties of file management. However in this case, although Fortran is not capable of performing all the necessary functions that are required, the use of system services cannot be allowed. The solution in this case is to use the full capabilities of Fortran I/O and in addition leave certain functions (such as the printing and purging of files) as the responsibility of the user. This eliminates the need for system calls within the code. The justification for this solution is that the usage of I/O system calls would further complicate the portability of the package because of the different directory structures. Therefore because of the large problems which would be caused with respect to portability the usage of system calls for the purpose of I/O must be disallowed.

3.1.3 Graphics

Graphics has been an integral part of control theory as witnessed by the many classical techniques such as Nyquist , Frequency Response, and Root Locus plots. Graphical output has always given the control engineer the intuitive information sometimes lost in extensive numerical solutions. Therefore it is realized that computer graphics can be a useful and powerful aid in the design of control systems. However graphics represents the greatest barrier to portability because of the present lack of standardization. In this section we will examine graphics as a special kind of application program and its related portability problems. As in all other cases where one deals with portability problems the goal is to isolate the non-portable segments from the portable ones. In order to accomplish this an examination of the structure of graphics software is undertaken.

In most situations the structure of the graphics software resembles Figure 3.2 . The CAD shell calls the graphics shell which in turn calls graphics primitives resident

in the support software. The CAD and graphics shell are part of the application layer as discussed previously. The supporting software is a collection of these primitives such as 'MOVE (X , Y) ' which has the effect of moving the cursor from its present position to the position (X , Y). These graphics primitives transmit, through the use of an operating system interface to the graphics device, the control characters containing the graphics information.

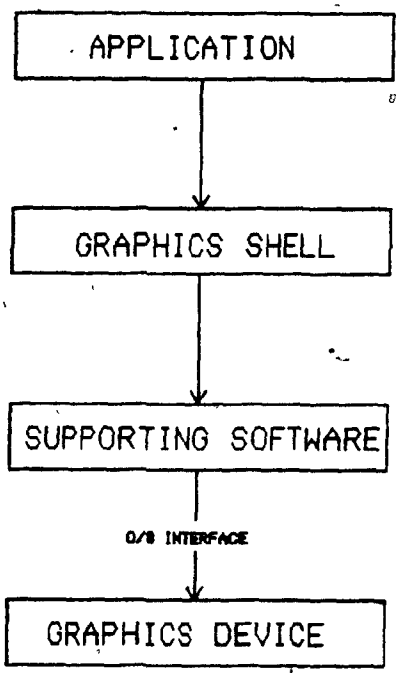


Fig 3.2

Device and O/S dependencies are the two features which results in graphics software being non-portable. Device dependency is clear: certain support software was designed to support certain graphics devices. For example, PLOT 10 was designed to

support a series of Tektronix terminals and only those or others which emulate this series of terminals will function properly. Furthermore, most support systems use different syntax for their graphics primitives. On the other hand O/S dependency is a result of the connection between the support software and the graphics device, which in most cases is accomplished through an O/S interface. As an example under the UNIX system the writing to a device is synonymous to writing to a special file. However, under the VMS operating system the link between graphics primitives and the graphics device entails the use of system calls to assign the output channel.

Two approaches for localizing these non-portable segments were suggested. The first solution was to adopt a GKS (Graphics Kernel System) [15] standard which is described in Figure 3.3 . The principle of the GKS solution is that the support software is device independent and portable. This solution is structured in layers where the graphics shell calls the device independent support software. In turn, the support software calls a device driver. Finally, through the use of an O/S interface, the control sequence (graphics information) is transmitted to the graphics device. Hence, the GKS solution entails using specific drivers for specific devices and different interfaces for different operating systems.

Another approach and the one which was decided upon, is based on the assumption that a small set of graphics primitives would be adequate to support the graphics requirements of the CAD package. The solution is to use generic names for the primitives and in this way make calls to dummy subroutines which contains the call to the associated primitive in the resident support software. For example, if the dummy subroutine PLOTTO (X,Y) was created, then this subroutine would simply call the PLOT 10 subroutine DRAWA (X,Y). As a result, throughout the application layer the routine PLOTTO would be referenced instead of DRAWA. Therefore with this solution, the support software would have to be substituted for a device of a different class by modifying the calls within the dummy subroutines. Of course, this method

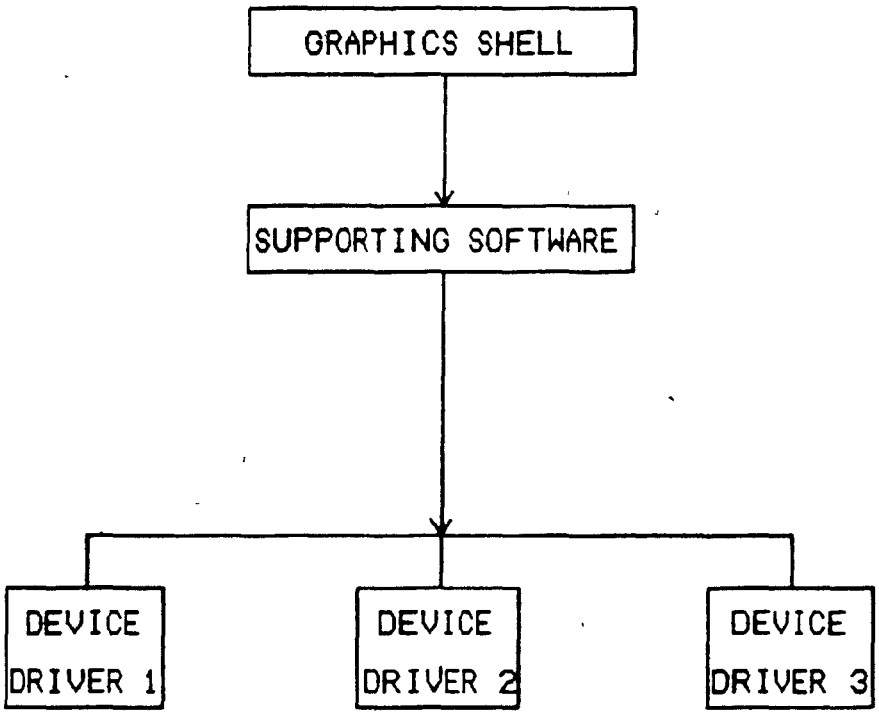


Fig 3.3

still forces the designer to create an operating system interface for each system. This solution is shown in Figure 3.4.

Throughout this section several problems related to portability were examined. The overall strategy in this respect is:

1. limited use of non-portable features (except when Fortran 77 is unable to accomplish desired feature)

user attributes such as education, CAD experience and how frequently the system will be used by the target user The following specifications were used to describe the target user for the CAD package

1. The user is a control engineer with a clear understanding of the theoretical formulations of the features included in the package.
2. The user has little CAD experience
3. The user will be an occasional user of the CAD facilities

3.2.2 Interface Dialog

The main problem with respect to interface dialog is complex input requirements and uninformative output In this section a brief discussion of the factors which influence the interface dialog are examined and some of the associated design decisions are presented. This presentation is broken into three areas; type of interface, input and output requirements.

As discussed previously, two commonly used interfaces are, the Question-Answer (Q/A) and the Command Language (C/L). A feature of the Q/A type dialog is a hierarchical ordered menu structure which guides the user through the design procedure. Unfortunately, this interface can overdetermine the sequence of tasks as shown in Figure 3.5 In this scheme, the progression from option X to option Y involves the traversal of the whole tree. On the other hand a C/L system allows for easy movement within the system, but of course is hard to use by the novice user. The choice of which type of interface to employ depends strongly on the user profile and therefore the Q/A dialog was chosen for the package.

The second area to be examined is interactive input requirements. There are two types of input data, data to be manipulated and parameters to control program

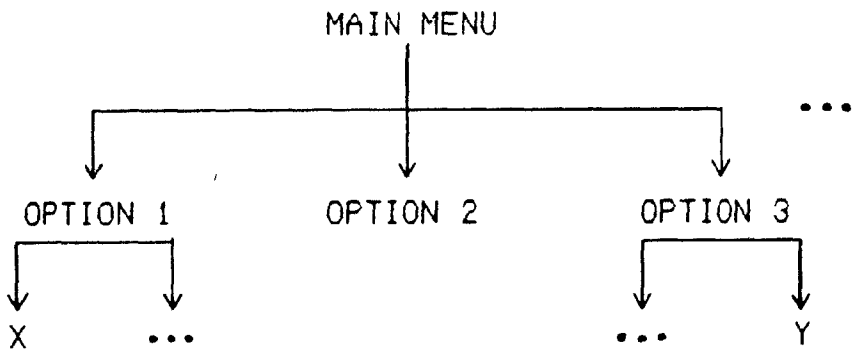


Fig 3.5

operations. One way to simplify things is to set certain control parameters to preassigned default values. This is especially useful for parameters or data which usually take on certain values, or only used in exceptional cases The ability to modify these parameters must exist if the user wants an unusual value. In the interactive environment of the CAD package the user is given considerable help by the program to input a series of parameters and/or data values. This help is in the form of a series of cues or messages which ensures that all input required is provided and is in the correct order Equally important, it also provides a framework for input error detection where each entry is checked and corrections can be asked for if an error is detected.

At present most CAD packages allow user interaction via keyboard commands. An alternative, because of the recent popularity of the light pen and the mouse, is to use input graphics for manipulating the program flow and perhaps for problem description input Graphical input offers an alternative to typing commands to determine program action. The user needs to be able to choose any of the many actions

that are appropriate and valid at a given time in the program. Graphically, these possible actions may be displayed as menus or light buttons on the display The user then merely points at the desired action, rather than typing the analogous command on the keyboard. In addition, graphical input can be used for problem definition such as system descriptions (block diagram interconnection) and perhaps to do graphical design by input constraint functions. However, because of the hardware dependence of such devices this feature would add greatly to the portability problem. Therefore, the use of input graphics for the CAD package was not considered.

A discussion of the output requirements of the CAD package is the third area of interest with respect to the interface dialog. Several specifications were used to describe the output requirements. The first of these is what kind of output is required. The type refers to the form of the actual output , for example text, graphical, or machine readable. The second is what is the quality of the output and this refers to factors such as permanence, availability and convenience. Within the package all possible output was examined in this context.

One of the major problems designers of CAD systems are faced with is the high volume of output. Several techniques for reducing the quantity of output were used and are listed below;

- examination of what data is really needed
- refine the problem so that the output expresses more concisely what is needed
- remove intermediate output inserted at an earlier stage to aid development
- examine formats of numerical output to determine whether the precision provided for is required
- only output a summary

Another problem specific to output is display capacity. Display capacity is limited, which implies that when the output is greater than the capacity of the terminal

is expected by the software to be an integer and the user responds with a character The user at this point must be warned that he has entered the wrong type of value and a chance for him to reenter the data is given.

Semantic error detection on the other hand involves analyzing the data to see if what the user has asked to do makes sense. In some cases this is not difficult but the order of complexity quickly rises if absolute reliability is desired. One criterion which can be used is to ensure that the system never crashes. This criterion although valid would not inform the user as to the reliability of the answers which have been computed. Semantic error detection (meaning of the information) can be broken into three classes of detection , limits, flow and computational . The limits are the easiest to enforce and deal with computational limits such as number of iteration of a particular procedure. For example in a generalized least squares identification scheme the number of iteration of the identification algorithm would be prompted from the user. This is a case where a hard limit can be set by the software to disallow relatively large number of iterations. The flow error detection checks whether the right sequences of things have been done in the right order. The third type of error detection is computational such as singularity checks of matrices.

The idea of error detection and recovery, which implies that the program recover from errors by interpreting what the user meant from the incorrect input in its context, makes considerable demands from the program Therefore in an interactive environment the best possible way is to stop each time an error is detected and give the user the opportunity of reentering that item or set of items immediately. This creates a cycle, where the user

$$\begin{array}{ccc} & ENTERS & \\ \swarrow & & \nwarrow \\ VALIDATES & \longrightarrow & CORRECTS \end{array}$$

This can be performed in an IF statement which tests for example that a number is positive: the algorithm for handling correct data can then become the THEN clause

while production of a diagnostic becomes the task of the ELSE clause. There are however cases where the input checking is nearly as complex as the algorithm to be performed and therefore should be included in the algorithm

In summary, the quality of the CAD system an its receptiveness by users hinges on the quality of the user profile. Therefore if an attempt is made to encompass too large a set of users the system becomes useless for everyone. In addition, programmers should make the minimum of assumptions about data coming from a user. Here it must be remembered that the type clause is also a hidden assumption which a user may be able to violate (eg alphanumeric characters in a numeric field). Furthermore, the error messages should be as comprehensible and helpful as possible. The type of error should be reported and the offending item(s) identified.

3.3 Software Design Methodology

Another of the main objectives is to produce a package which is easily modified and maintained. The ease of performing modifications and maintenance depends on the (1) structure of the design and (2) the standards of documentation. In this section these two areas will be examined.

3.3.1 Structure of Design

The main objective of this project is to design a flexible interactive CAD package that would interface to a variety of linear time invariant multivariable control software. Therefore, the interface can be used to not only implement commonly used design and analysis methods, but in addition to implement current techniques and even algorithms which might be developed in the future This results in a requirement that the package be extremely flexible so that new procedures can easily be accomodated.

One approach to accomplish the task of creating reliable and flexible software is to use the structured design approach. Structured design is a method whereby the

1. How to use the program
2. State of the Project
3. Overall Specifications
4. Models used to subdivide the program
5. Computational requirements (storage and time)
6. Flow of Control and Flow of Data through the program
7. Detailed Description of Data
8. Meaning of the error messages
9. Performance Evaluation

In addition, computational software should include the following;

1. Proper explanation of algorithm
2. References for Theory
3. Numerical Stability and Robustness of subroutine
4. Sample graphical output and recommendations for package graphics

These standards must be strictly adhered to, if a reliable and maintainable package is to be constructed.

Chapter 4

PACKAGE SPECIFICATIONS

In this chapter, the design and specifications of the CAD package are presented. The chapter is divided as follows: The first section provides a description of the functional structure of the package. The importance of the functional structure is it can be regarded as how the user perceives the structure of the package. The second section presents the software structure. The software tools which were required and implemented for the package are presented in the third section. The fourth section outlines the resource requirements for the CAD package and identifies possible hardware configurations.

4.1 Functional Structure

The first step in the design of the package was to decompose it into a set of modules. A requirement for this was that the decomposition should lead, in a logical manner, to a package structure which would resemble the control system design procedure, as closely as possible. This requirement was enforced to ensure that the package could be easily understood and used by control engineers. In order to accomplish this, a model for the design procedure was required, and the result is shown in Figure 4.1. This model represents the procedure as a sequence of functions with cyclical iterations in going from measured input-output data to a performance analysis of the plant and the controller.

This model allowed for the decomposition of the package into four distinct modules. These are;

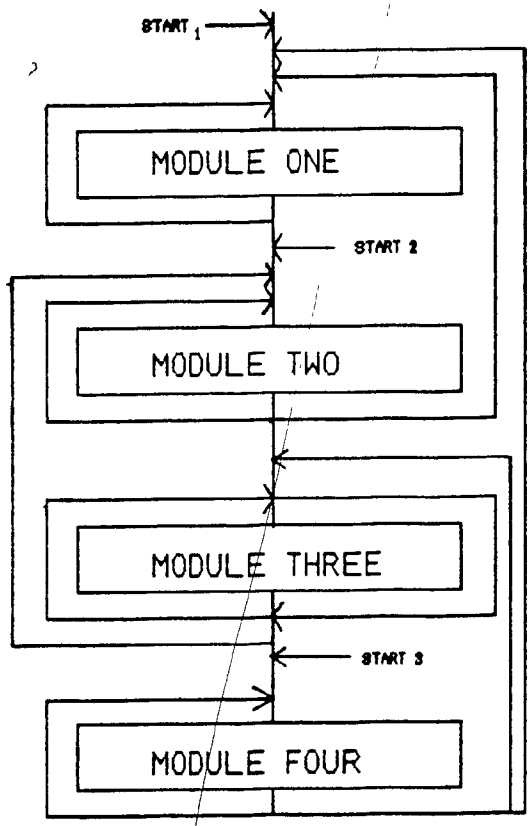


Fig 4.2

This structure allows the user to enter the package at three possible entry points. These are based on what type of apriori information the user has of the plant and/or the controller. The requirments for these entry points are:

1. Plant Input-Output Data
2. Model for the Plant
3. Model for the Controller

where each of these refers to the entry number shown in Figure 4.2.

In addition several iterative paths have been defined to allow the user to modify options and system data until a satisfactory result is achieved. Furthermore, a short-cut path has been defined, so that the user can manually enter both the plant and a controller and execute a performance analysis on both.

4.2 Software Structure

This section describes the software structure of the package. Because of the user interface design decisions reached in Chapter 3 the package has been configured in a tree structure with a hierarchical menu system. This configuration is shown in Figure 4.3.

From Figure 4.3 it can be seen that the software has been divided into five levels and each of these has been given a specific duties and responsibilities. A list of the levels along with their assigned responsibilities is given below.

LEVEL	RESPONSIBILITY
Overhead	Global Declarations Package Initialization Initialization of Error Routines File Handling Module Selection
Module	Provide Choice of Operations Ensure Entry Requirement is Fulfilled
Submodule	Prompts for all Common Elements Provide Subchoices
Computational Overhead	Prompts for Specific Algorithm Information Call Computational Routine Display Results (Numerical and Graphical) Set up Appropriate Values for Computational Routine
Computational	Performs Computation

By enforcing these preassigned responsibilities on each of the levels the understanding of the package structure is simplified by ensuring uniformity on any particular level.

are data structures which contain the information regarding the realization of the plant and/or controller Data files are files which contain large data sets which can be externally provided (e.g measured input-output data) or generated by the modules.

Along with the creation of these data structures is a need for standardization of data transfers between modules Therefore, guidelines were specified and are listed below:

1. Package initialization data is available to the modules through named common blocks.
2. System description data structures are declared in the overhead program (which serves as the Fortran main program) and passed down to the corresponding modules (wherever necessary) as subroutine arguments
3. Data files are read into the package and must have a predefined format or may be generated by the package.

These guidelines must be adhered to ensure consistency and software which can be easily modified.

To implement the package five intermodule data structures have been defined and are shown in Figure 4.4 These are:

1. Input-Output Data File
2. Discrete Time Transfer Function
3. Continuous/Discrete Time State Space realization of the Plant
4. Continuous/Discrete Time State Space realization of the plant and the Controller
5. Simulation Data File

Figure 4.4 shows which of these structures must be available to particular modules. (An assumption that has been made is that all procedures in Module Three 'Controller Synthesis' can be implemented using the state space realization of the plant.)

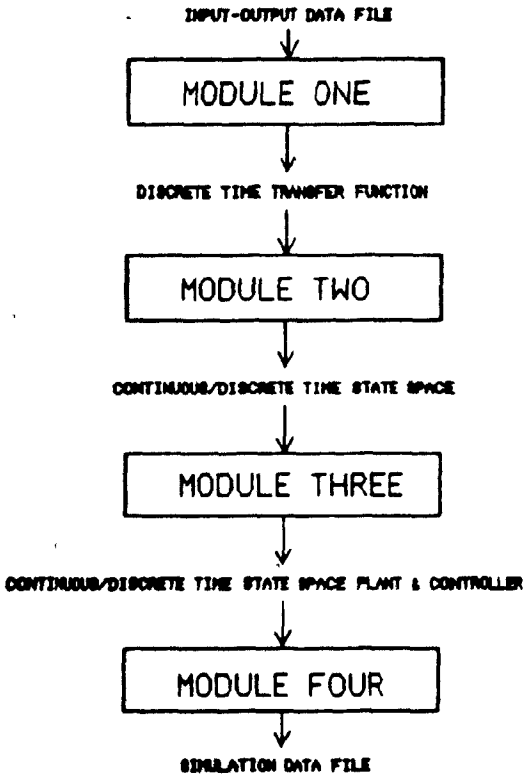


Fig 4.4

4.2.2 Internal File Structure

One of the main features of the package is its interactive nature, which implies that the terminal (alphanumeric or graphics) is the primary output device. However, there is a need for a more permanent record of the design results and this is accomplished through the use of files. (Note: These files are distinct from the previously defined data files.)

usersavefile m3	storage for Module 3
usersavefile.m4	storage for Module 4
usersavefile dry	diary file
usersavefile pnt	print file

The first five of these files are used internally by the package for storage and restarting of procedures The diary file is used to record the design procedure. The print file is used to record the results displayed on the terminal, and therefore serves as the permanent record of the design results. On any subsequent design session, the user has the choice of restarting a previous session (by simply selecting his previous filename) or starting a new session.

The advantage of this implementation is it reinforces the previously stated portability objective by respecting the design decisions stated in Chapter 3. In addition, it is easy to use because a large portion of the internal file structure is transparent to the user.

4.3 Software Tools

Once the package structure was specified, an attempt was made to identify some of the software tools which would be necessary The approach adopted was to consider the software tools as general-purpose functions, which are defined as functions which are required repeatedly by the modules [1], such as device interfaces, output functions and user interfaces As a result of this approach the construction of the package is reduced to simply using a combination of appropriate general-purpose functions and necessary special purpose functions. This approach has the advantage of ensuring uniformity from the user's viewpoint and reduces redundant development of similar functions. In addition, changes to general-purpose functions can be made without affecting other parts of the system and development of the package is facilitated. However, a drawback is the development of the overall package cannot proceed until development of these functions is complete.

In this section some of the software tools which were identified and developed for the package are examined. These include user interface routines, graphics, error detection and help facilities. All of these are used extensively and form the core of the package.

4.3.1 User Interface

The first of the general-purpose functions identified were for the purpose of user interaction and the development of these was broken into two categories; input prompts and output formatting functions. In the case of input prompt functions, it was realized that four routines would be needed and these are:

1. `Yesno`
2. `Prompt`
3. `Menudrive`
4. `Readmatrix`

The yesno routine prompts the user for a yes/no response to a specified question. The prompt routine prompts the user for a variable of a specified type (double precision, real, integer or character) to a specified question. The `menudrive` displays a menu of specified choices and prompts the user for his desired selection. The `readmatrix` routine prompts the user for the values of each entry in a matrix of specified size and type.

Extreme care was exercised in the design of these routines with respect to syntactic error detection. Therefore all of these routines check that the right type of data has been supplied by the user. If an error is detected an appropriate error message is displayed and the user is given the opportunity to reenter his response. This feature was one of the motivators for creating general purpose prompting routines instead of implementing them directly in the code wherever necessary. This

approach reduces redundancy in the code but of course execution is slower because of the subroutine call. This drawback was considered, but it was felt that speed was a relatively unimportant issue when dealing with user interaction.

The second category of user interface routines are output formatting functions. The following three functions were created;

1. `Pmatrix`
2. `Modpmatrix`
3. `Integerbox`

The `pmatrix` routine displays matrices. In order to handle large matrices this routine displays the matrix as several submatrices. The `modpmatrix` is similar but allows for special characters to be inserted in specified entries in the matrix (This is used to indicate that an element of a matrix was not calculated.) `Integerbox` displays integer variables in tabular form.

Access to all of the user interface routines is through Fortran subroutine calls.

4.3.2 Graphics

As discussed in Chapter 3, graphics is one of the major elements in a successful CAD package. This subsection presents the requirements and capabilities of the graphics utility.

A preliminary step in the design of the graphics utility was to determine which features would be required. It was realized that the graphics utility would have to have the capability of plotting such functions as, Bode (Magnitude and Phase), Nichols , Nyquist and Time Functions.

Therefore, several interactive graphical operators would be required. These are:

1. Overlaying
2. Windowing
3. Zooming

one for the graphics display and the second for the interactive dialog with the user (alphanumeric terminal). The following menu appears on the alphanumeric terminal after the first default screen is displayed.

1. Go to the previous default Screen
2. Return to the default Screen
3. Go to the next default Screen
4. Create a screen
5. Alter a graph on the present Screen
6. Zoom in on plot from the present screen
7. Overlay plot on present window
8. Status Summary
9. Stop (Graphics Session)

The first three of these commands allows the user to easily move between default screens. The 'create a screen' command allows the user to select up to four plots and display them on a single screen. The 'alter' command provides the user with the ability to interactively modify the range of the axes, labels and the origin of any plot on the present screen. The 'zoom' command takes the specified plot and displays it on the full screen. The user can overlay several plots on the same set of axes by selecting the 'overlay' command. The 'status summary' command provides the user with information, such as the total number of plots, the number of screen and overlay information.

The graphics utility, because of its size and complexity, was itself designed using the approach of general-purpose functions. The previously described interface routines were used to provide the user with interactive dialog to manipulate the display. In addition, several graphical general-purpose functions were created such as axis plotting and labelling

Portability was achieved by the approach presented in Chapter 3 Tektronix Plot 10 was used as the graphical support software and an operating system dependent interface was created. These two features would have to be modified if an alternate operating system and graphics device were used.

4.3.3 Error Detection and Recovery

The third software tool and a general requirement of the package is run-time error detection and recovery This utility increases the user's confidence in the system and contributes to user friendliness

In general an error detection and recovery utility must attempt to perform as many of the following features as possible.

- advise user that an arithmetic error has occurred
- advise of where the error has occurred
- inform of what caused the error
- set up the appropriate conditions so that execution can continue

The first of these features is performed by the operating system when executing the default error handling routine. However, as one tries to replace the default action by a more comprehensive error handler (increasing reliability) the complexity quickly rises

To design this utility it was decided to divide the problem into designing two handlers, one for arithmetic errors and another for input-output errors The general scheme with regards to arithmetic errors is to inform the operating system that an alternate handler exists and therefore upon detection of an error this routine should be executed. The preliminary objective of the utility was to provide global error detection and recovery for all arithmetic errors. However, it was realized that because the recovery procedure is so closely related to specific algorithms, global recovery could not be achieved Therefore, it was decided to construct two types

pointer from the source menu and opens a specified file which contains the relevant help text. The text is then displayed on the terminal.

The advantage of this implementation is that the help text can be easily modified since they are not in the program code but in special files. This allows for the simple maintenance of the help facility.

Throughout this section the software tools which were constructed for the package are described. The documentation for all these routines specifying the exact format can be found in Appendix A.

4.4 Computational Software

This section describes the computational software which would be required to implement an operational package. This is specified by first presenting a list of proposed features. Secondly, a list of software libraries presently available.

4.4.1 Proposed Features

Module One **Identification**
- Digital Filtering
- Statistical Analysis
- Least Squares Identification Algorithm
- Maximum Likelihood
- Cramer Rao
- Residuals and Model Output

Module Two **Models and Realizations**
- Model Conversions
  - Matrix Fraction Description $\rightleftharpoons$ State Space Description
  - State Continuous Time $\rightleftharpoons$ State Sampled Data
- Observability and Controllability
- Balanced Realizations
- Model Entry/Modification
- Poles and Zeros
- Model Reduction

Module Three **Controller Synthesis**
- State Feedback
  - Pole Placement (Discrete and Continuous)
  - Linear Quadratic (Discrete and Continuous)
- H^∞ design
- Kalman Filtering and Loop Transfer Recovery
- Continuous to Discrete Conversion of Quadratic Criterion

Module Four **Performance Analysis**
- Root Locus (any parameter)
- Nyquist
- Time Responses: deterministic/stochastic
- Bode Plot
  - Complementary Sensitivity
  - Sensitivity
  - Open Loop
  - Closed Loop
  - Max/Min Singular Values
- Variance Analysis
- Controller Entry/Modification

4.4.2 Software Libraries

The following is a list of software subroutines which are required to implement the proposed features of the package. This list was constructed from the recommendations of the ORNL report [8].

1. Linpack This is a software library used to solve linear equations and the linear least squares problem. It not necessary to include the whole package, and therefore a subset which would be adequate for the package is presented.

DGECO DGEFA DGESL DGEDI	Solve $Ax = b$, estimate condition, A^{-1}, det$A$ for general A
DPOCO DPOFA DPOSL DPODI	Solve $Ax = b$, estimate condition, A^{-1}, det$A$ for $A = A^T > 0$

DHQRORT	Modification to HQR2 to compute real Schur form (traingular)
DCBAL, DCORTH, DCOMQR, DCOMQR2, DCBABK2	Eigenvalues and eigenvectors of $A \in C^{n \times n}$
DTRED1, DTQLRAT, DTRED2, DTQL2	Eigenvalues and eigenvectors of $A = A^T \in R^{n \times n}$
DMINFIT, DSVD	Compute singular value decomposition of A and/or solve linear least sq. problem
DQZHES, DQZIT, DQZVAL, DQZVEC	Real generalized eigenvalues and/or eigenvectors

4. Miscellaneous Laub Software

DFRMG, ZHECO, ZHEFA, ZHESL, DCL1, ZLINRM, DHETR	Double precision frequency response software given A, B, C compute $G(z) = C(zI - A)^{-1}B$ $z \in C$
HQR3, EXCHNG, QRSTEP, SPLIT	Order real Schur form

4.5 Resources

In this section an estimate of the resources required is presented with the aim of determining the hardware needed to support the package. To examine this issue two characteristics of control computation were examined; memory requirements and computational load.

Memory requirement estimates were prepared for the package and are listed below (in bytes).

Module Name	Fixed Overhead	Small Problem	Large Problem
Module O	200,000	n/a	n/a
Module 1	100,000	200,000	600,000
Module 2	100,000	200,000	600,000
Module 3	100,000	200,000	600,000
Module 4	100,000	200,000	600,000
Graphics Shell	160,000	n/a	n/a
Plot Ten	26,000	n/a	n/a
Eispack (subset)	81,000	n/a	n/a
Linpack	6,500	n/a	n/a
O/S Math Library	70,000	n/a	n/a
Totals	943,500	800,000	2,900,000

(n/a - not applicable, small problem -2 inputs 2 outputs, large problem - 8 inputs 8 outputs). Therefore, the total memory needed varies from 1.8 Megabytes to 4.0 Megabytes depending on the size of the problem.

Because of the decision to concentrate on the interactive shell, computational information for the algorithms was not investigated and of course would be highly dependent on the algorithms chosen. However, the computational load must be considered in implementation, especially if the package is to be implemented in an environment where computational resources are scarce. This is of great importance if the interactive nature of the package is to be preserved.

Based on this, two possible hardware configurations are presented, depending on the environment chosen. If the package is to be used in a multi-user multi-tasking environment, a configuration similar to the developmental environment would be adequate. The environment consisted of:

Chapter 5

MODULE ONE

In this chapter Module One, the identification module is described. The approach adopted within this chapter is similar to the one used in Chapter 4 and therefore, the chapter is divided as follows: The first section presents an overview of Module One by examining the functional and software structure. The remaining four sections discuss in detail the structure of each of the submodules.

5.1 Overview of Module One

5.1.1 Functional Structure

The first step in the design of Module One was to decompose the module so that the structure would resemble the identification procedure. To accomplish this a model for the procedure was created and is shown in Figure 5.1.

The model represents the procedure as a sequence of functions, using as input the input-output data and producing as the output a discrete-time model. From this model it was decided to decompose the identification module into four submodules:

1. Digital Filtering
2. Statistical Analysis
3. Parameter Estimation
4. Input-Output Plots

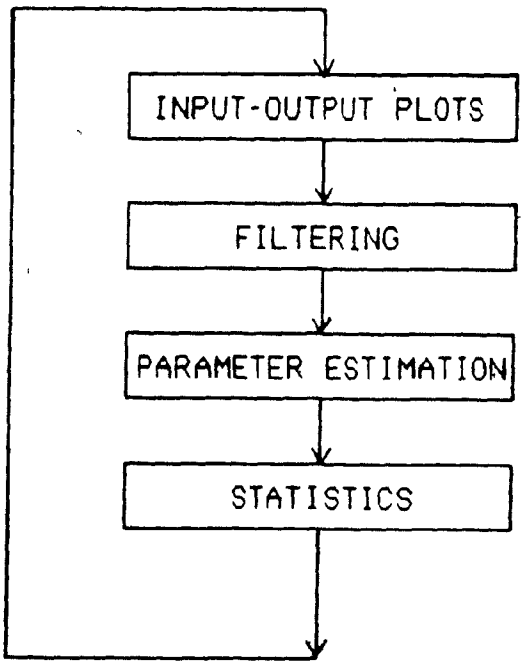


Fig 5.1

The second step in specifying the module is to define the functional interconnection structure. This refers to how the user perceives the structure of the module(i.e. the interconnection of the submodules). To define this an analysis of the tasks required to complete an identification procedure was undertaken and the resulting interconnection structure is shown in Figure 5.2.

This structure allows the user to enter the module at one point and the associated entry requirement is a data file. In addition, the structure includes several iterative paths to enable the user to modify certain parameters until a satisfactory result is achieved. Another path enables the user to filter the measured input-output data and to subsequently perform the parameter estimation procedure on the filtered data. Several additional paths have also been defined to allow the user to apply the

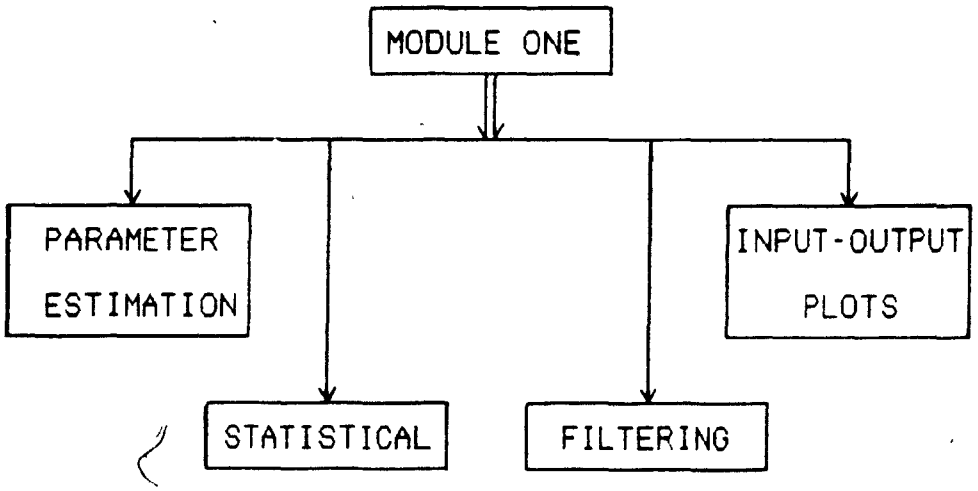


Fig 5.3

As described previously, each of the five software levels has certain assigned responsibilities. Let us now examine how these translate into specific features which must be included at each level of Module One. At the module level, the features which must be performed are:

1. Prompt for name of data file
2. Read input-output data from data file and store in an appropriate array
3. Provide entry to submodules and make available the input-output data

A flow diagram of the module level, demonstrating these features, is shown in Figure 5.4.

In order to detail the responsibilities of the submodules we must first examine the input/output requirements of the computational levels. In Figure 5.5 a block diagram representation of the computational levels is illustrated. The input, as shown, is composed of data and option parameters and the output is the result of the particular algorithm. In general, it is the responsibility of the submodule level to prompt the user for all data not produced by previous computations (i.e. the output of one

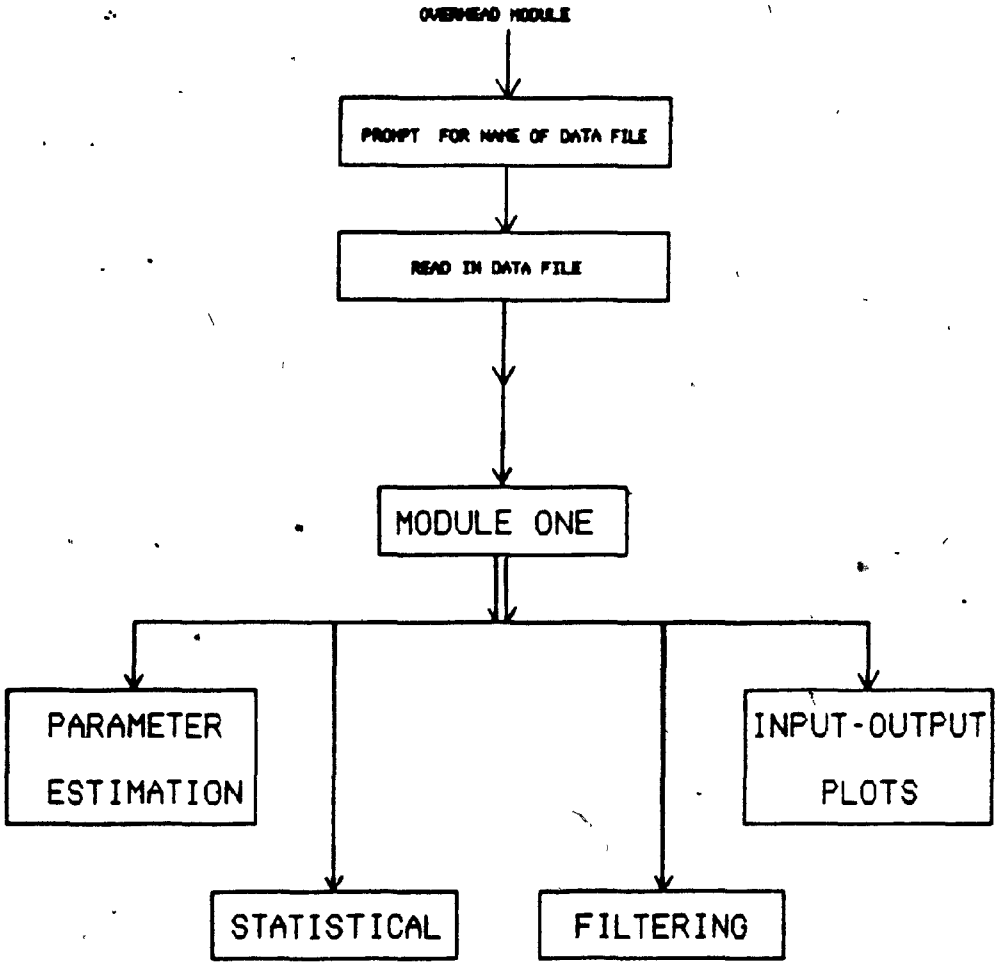


Fig 5.4

procedure becomes the input to another procedure) and all option parameters. The specific responsibilities for each submodule are discussed later in this chapter.

The next element in the specification of the software structure is to determine which internal data structures would be required at the module level. The input to the

u_i - is the i th input of the system

y_i - is the i th output of the system

n - is the number of inputs of the system

m - is the number of outputs of the system

l - is the number of data points

The output of Module One is a discrete-time multivariable transfer function of the form:

$$Ay = Bu + Ce$$

where A, B and C are matrices whose entries are polynomials in the delay operator. The main goal of this module is to identify the parameters of these polynomials and these results are stored in three arrays shown below:

$$A = \begin{pmatrix} a_{1,1} \\ a_{1,2} \\ \vdots \\ a_{1,m} \\ \vdots \\ a_{m,m} \end{pmatrix} \qquad B = \begin{pmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,n} \\ \vdots \\ b_{m,n} \end{pmatrix} \qquad C = \begin{pmatrix} c_{1,1} \\ c_{1,2} \\ \vdots \\ c_{1,m} \\ \vdots \\ c_{m,m} \end{pmatrix}$$

where

$$a_{i,j} = \begin{pmatrix} a^0_{i,j} & a^1_{i,j} & \dots & a^{r_{i,j}}_{i,j} \end{pmatrix}$$
$$b_{i,j} = \begin{pmatrix} b^0_{i,j} & b^1_{i,j} & \dots & b^{s_{i,j}}_{i,j} \end{pmatrix}$$
$$c_{i,j} = \begin{pmatrix} c^0_{i,j} & c^1_{i,j} & \dots & c^{t_{i,j}}_{i,j} \end{pmatrix}$$

and

$$R = \begin{pmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,m} \\ r_{2,1} & r_{2,2} & \dots & r_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m,1} & r_{m,2} & \dots & r_{m,m} \end{pmatrix}$$

$$S = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m,1} & s_{m,2} & \cdots & s_{m,n} \end{pmatrix}$$

$$T = \begin{pmatrix} t_{1,1} & t_{1,2} & & t_{1,m} \\ t_{2,1} & t_{2,2} & & t_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m,1} & t_{m,2} & & t_{m,m} \end{pmatrix}$$

5.2 Digital Filtering

The first submodule performs various filtering operations on each input and output data set and the submodule accepts three types of data as input, original, filtered and residual In addition, the submodule supports the following types of filters:

1. Low Pass
2. High Pass
3. Band Pass
4. Band Stop
5. Trend Removal

Furthermore, the filters can be specified in three ways:

1. Default Case (eg. 10th order Chebyshev)
2. Specify Order
3. Specify Ripple Characteristics

To contain the above option parameters an array `filtercontrol` was defined and has the following form:

$$\texttt{filtercontrol} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_9 \end{pmatrix}$$

where each f_i is a vector of dimension $(n+m)$ and where each each of these vectors contains the following:

f_1 - Filter Selection

f_2 - Sampling Frequency

f_3 - Cutoff Frequency for Low/High Pass Filters

f_4 - Lower Cutoff Frequency for Band Pass/Stop Filters

f_5 - Upper Cutoff Frequency for Band Pass/Stop Filters

f_6 - Pass Band Ripple

f_7 - Stop Band Attenuation

f_8 - Stop Band Transition Bandwidth

f_9 - Order of Filters

It is the responsibility of the submodule to provide adequate interaction in order to fill the `filtercontrol` array.

The results of this submodule are placed in an array `filtered` dimensioned $(n+m,l)$. This array has the same form as `original` and is shown below:

$$\texttt{filtered} = \begin{pmatrix} fu_1(1) & \dots & fu_n(1) & fy_1(1) & \dots & fy_m(1) \\ fu_1(2) & \dots & fu_n(2) & fy_1(2) & \dots & fy_m(2) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ fu_1(l) & \dots & fu_n(l) & fy_1(l) & \dots & fy_m(l) \end{pmatrix}$$

A flow diagram of this submodule is illustrated in Figure 5.6.

5.3 Statistical Analysis

The second submodule performs statistical calculations on each input and output data set of the original or filtered data and/or each residual data set. The features which are supported include:

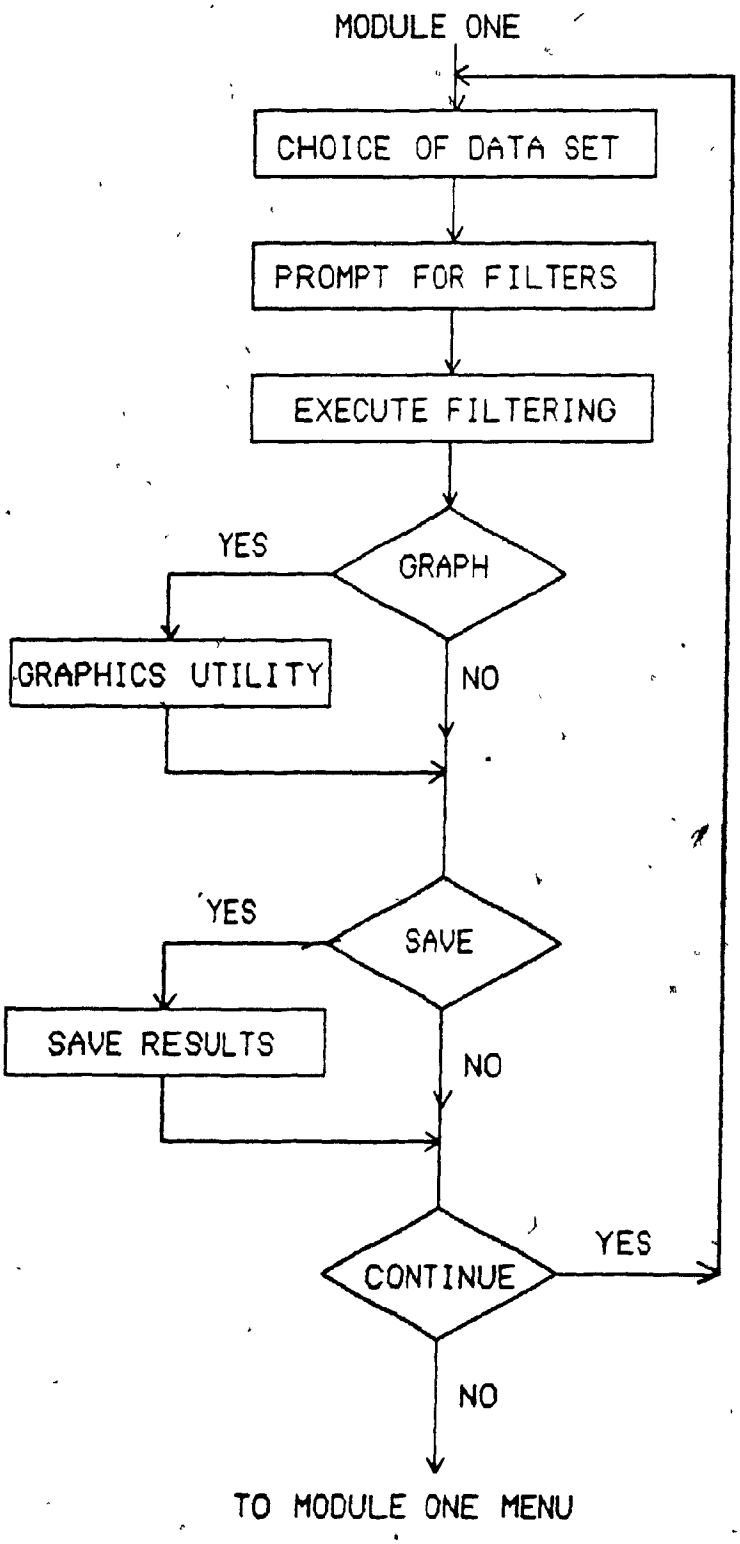

MODULE ONE
CHOICE OF DATA SET
PROMPT FOR FILTERS
EXECUTE FILTERING
GRAPH
YES
NO
GRAPHICS UTILITY
SAVE
YES
NO
SAVE RESULTS
CONTINUE
YES
NO
TO MODULE ONE MENU

Fig 5.6

Similarly, two other arrays were created:

covarcontroloo of dimension (m, m) which controls the covariance operation between outputs.

covarcontrolio of dimension (n, m) which controls the covariance operation between inputs and outputs

The results of this submodule are stored in a set of arrays. The first of these store the results of the mean, variance and standard deviation computations

mean is a vector of dimension $(n - m)$

variance is a vector of dimension $(n - m)$

standev is a vector of dimension $(n - m)$

In addition, the following three arrays are used to store the results of the covariance calculations

covarianceii of dimension (n, n) which stores the results of the covariance calculation between inputs. (i.e. the (i, j) th element is the covariance between the i th input and the j th input)

covarianceoo of dimension (m, m) which stores the results of the covariance calculation between outputs

covarianceio of dimension (n, m) which stores the results of the covariance calculation between inputs and outputs

Furthermore, an array was created to store the results of the discrete fourier transform

dft is an array of dimension $(l, n - m)$

A flow diagram of this submodule is illustrated in Figure 5.7

5.4 Parameter Estimation

This submodule performs parameter estimation and uses as input the original or filtered input-output data sets [7]. The submodule in addition, has the following features:

1. Least Squares or Maximum Likelihood Algorithms
2. Specify Model Order by Specifying Order of each Polynomial
3. Specify Starting Sample Point for Estimation
4. Multivariable Delays
5. Computation of Residual
6. Computation of Model Output (the output of the identified model when the same input data set is applied)
7. Computation of Akaike Information Criteria

In order to implement this submodule, the ability to modify the previously defined R, S and T matrices must exist. This allows the user to select the order of each polynomial of the model to be identified. In addition, an array delay was created to store the delays of each input which effect each output, and has the following form.

$$\mathbf{delay} = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,m} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,m} \\ \vdots & \cdot & \ddots & \cdot \\ d_{n,1} & d_{n,2} & \cdots & d_{n,m} \end{pmatrix}$$

where ,

$d_{i,j}$ - is the delay from the i th input to the j th output.

To store the results of the identification procedure several arrays are used. First, the parameters of the model are placed in the previously defined A, B and G matrices. The results of the residual computation are stored in the following array:

$$\mathbf{residual} = \begin{pmatrix} r_1(1) & r_2(1) & \cdots & r_m(1) \\ r_1(2) & r_2(2) & \cdots & r_m(2) \\ \vdots & \cdot & \ddots & \vdots \\ r_1(l) & r_2(l) & \cdots & r_m(l) \end{pmatrix}$$

The result of the model output computation are stored in an array of the following form:

$$\texttt{modelout} = \begin{pmatrix} m_1(1) & m_2(1) & \dots & m_m(1) \\ m_1(2) & m_2(2) & \dots & m_m(2) \\ \vdots & \vdots & \ddots & \vdots \\ m_1(l) & m_2(l) & \dots & m_m(l) \end{pmatrix}$$

A flow diagram of this submodule is illustrated in Figure 5.8a,b.

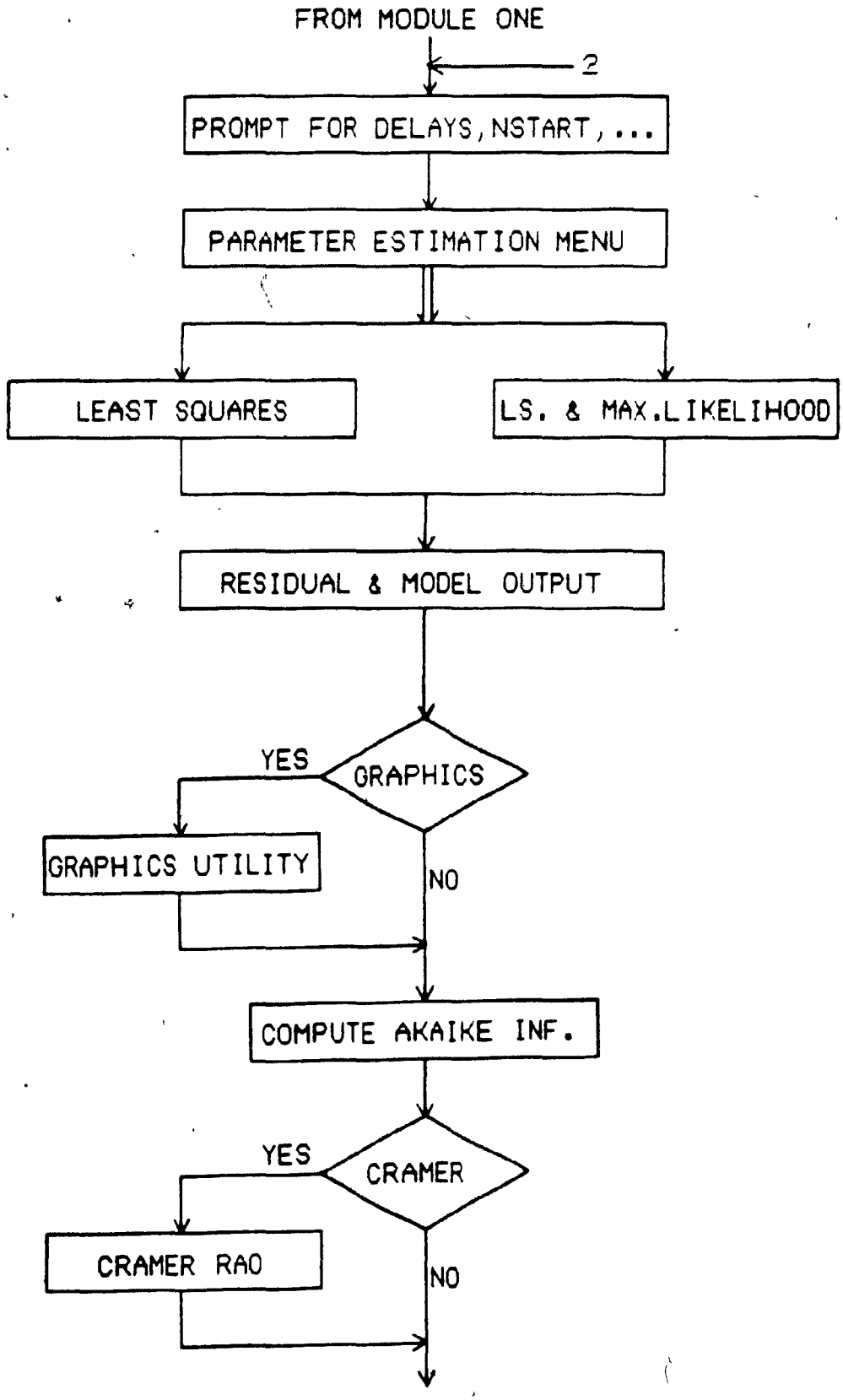

FROM MODULE ONE
2
PROMPT FOR DELAYS, NSTART, ...
PARAMETER ESTIMATION MENU
LEAST SQUARES
LS. & MAX.LIKELIHOOD
RESIDUAL & MODEL OUTPUT
YES
GRAPHICS
GRAPHICS UTILITY
NO
COMPUTE AKAIKE INF.
YES
CRAMER
CRAMER RAO
NO

Fig 5.8a

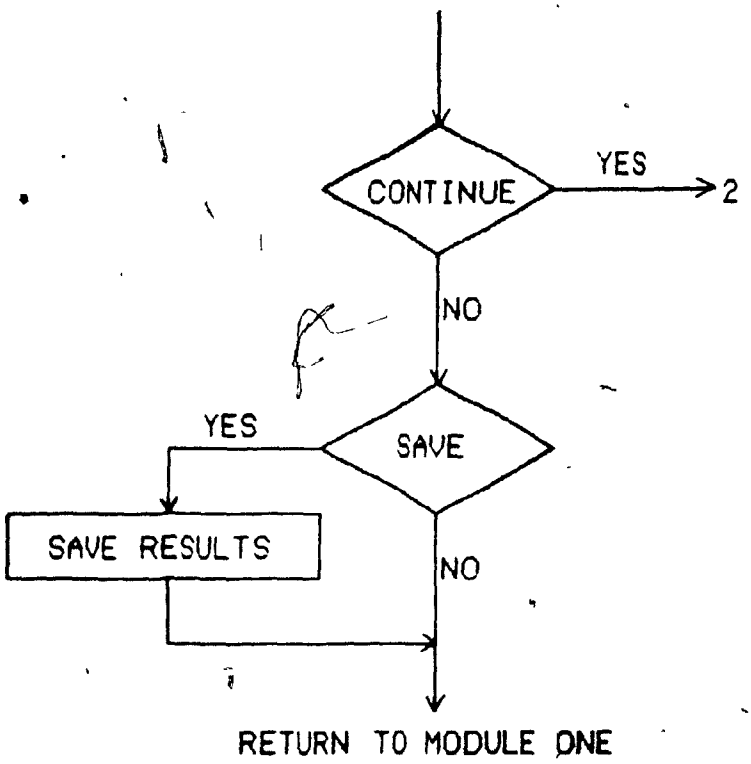
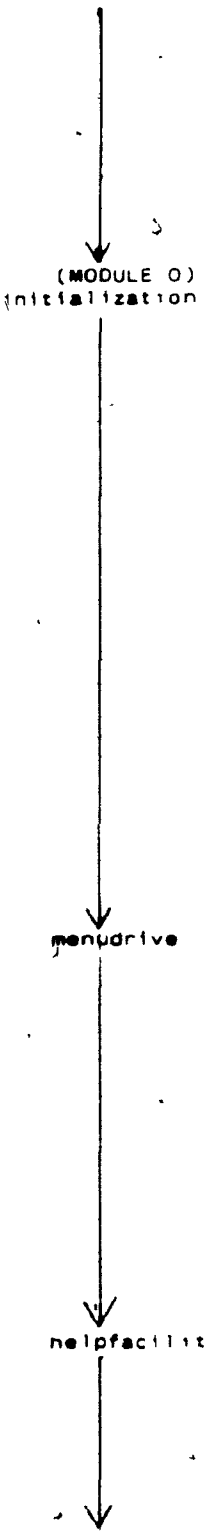


Fig 5.8b

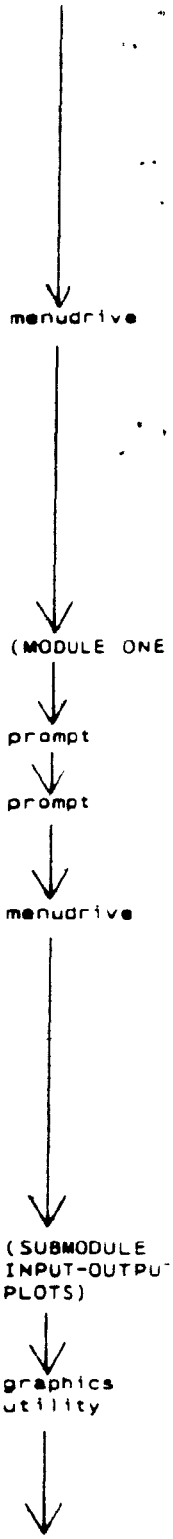
5.5 Input-Output Plots

This submodule allows the user to display the input-output data directly from the main menu of Module One. The submodule uses as input the original array and subsequently calls the graphics utility.




```
**********************************
*                                *
*   WELCOME TO McGILL s          *
*   CONTROL C.A D  PACKAGE       *
*                                *
*   VERSION 1 0                  *
*   AUGUST 1984                  *
*                                *
**********************************


                    ** ATTENTION **
        Note you may access the help facility from
        any menu by simply entering a  ?  or 'h' or  help
 Do you have a user save file ?
n
   In order to open a new user save file
 Enter the usrsavefile along with the directory
user.d/thesis


              MAIN MENU
              ---------

    1) IDENTIFICATION

    2) MODEL REP. & ANAL.

    3) CONTROLLER DESIGN

    4) ANAL TIME and FREQ

    5) QUIT

  Enter the number corresponding to the item of your choice
help

  **********
  helpfiles.d/mainmenu
  **********
  mainmenu
********** helpfacility **********
At this point the user is able to proceed to one of the 4 modules which
are listed below;
        1  Identification, Filtering and Statistical Operations
```

```
                - the entry requirement is a data-file
        2. Model Representations- Analysis, Entry and Conversions
                - the entry requirement is a system representation
        3  Controller Design
                - must be predeeded by mod2
        4  Simulation
 TO CONTINUE HIT CR



              MAIN MENU
             ------------------

  1) IDENTIFICATION

  2) MODEL REP  & ANAL.

  3) CONTROLLER DESIGN

  4) ANAL TIME and FREQ

  5) QUIT

  Enter the number corresponding to the item of your choice
 1



  Enter the number corresponding to the item of your choice
          In order to enter MODULE ONE you must provide
          a file containing Input/Output DATA
 Please enter the name of the datafile
  /data d/iodata



              MOD1 MENU (identification)
             ------------------

  1) FILTER

  2) STATISTICS

  3) PARAMETER ESTIMATION

  4) INPUT-OUTPUT PLOTS

  5) BACK TO MAIN MENU

  Enter the number corresponding to the item of your choice
 4



 ------------------------
 '' THE GRAPHICS SHELL ''
 ------------------------
 ------------------------
  EXECUTING   PLEASE WAIT
 ------------------------
```



```
                --GRAPHICS MENU--
               ------------------

   1) Go to the PREVIOUS default screen
   2) RETURN to the working default screen
   3) Go to  the  NEXT  default  screen
   4) CREATE a screen
   5) ALTER a graph on present screen
   6) ZOOM in on plot from the present screen
   7) OVERLAY plot on a present window
   8) Status SUMMARY
   9) STOP
-------------------------------------
  Enter the number corresponding to the item of your choice
9
```

menudrive

```
                MOD1 MENU (identification)
               ----------------------------

   1) FILTER
   2) STATISTICS
   3) PARAMETER ESTIMATION
   4) INPUT-OUTPUT PLOTS
   5) BACK TO MAIN MENU

  Enter the number corresponding to the item of your choice
1
```

menudrive

(SUBMODULE FILTER)

```
                  Filtered Data
                 ---------------

   1) ORIGINAL DATA
   2) RESIDUAL DATA
```

menudrive

3) FILTERED DATA

4) BACK TO MOD1 MENU

On which data set do you wish to filter
Enter the number corresponding to the item of your choice
1

menudrive

---

1) Default Case

2) Specify Order

3) Specify Xp

How would you like to specify your filter ?
Enter the number corresponding to the item of your choice
1

menudrive

FILTER OPERATIONS

1) LOW PASS FILTER

2) HIGH PASS FILTER

3) BAND PASS FILTER

4) BAND STOP FILTER

5) TREND REMOVAL

6) NO OPERATION

Enter the operation desired for input 01 :
Enter the number corresponding to the item of your choice
1

prompt

Please enter the sampling frequency in cycles/unit time

prompt

Please Enter the cutoff frequency for the LOW PASS FILTER

menudrive

FILTER OPERATIONS

1) LOW PASS FILTER

2) HIGH PASS FILTER

3) BAND PASS FILTER

```
   Filter Routines

Do you wish graphical output ??
n


                MODEL MENU (identification)
                ---------------------

    1) FILTER

    2) STATISTICS

    3) PARAMETER ESTIMATION

    4) INPUT-OUTPUT PLOTS.

    5) BACK TO MAIN MENU

  Enter the number corresponding to the item of your choice
3


                   DATA SELECTION MENU
                   -------------------

    1) ORIGINAL IO DATA

    2) FILTERED IO DATA

you can perform parameter estimation on  the above data sets
  Enter the number corresponding to the item of your choice
2
Enter the number of data points
200
Enter the number of inputs of the system
2
Enter the number of outputs of the system
2
Enter Starting Sample Number for estimation
5
```

NDATA	NU	NY	NSTRT
200	2	2	5

```
Are you satisfied with these numbers ? (yes/no)
Y
   The following is a prompt for multivariable delays
  These delays are prompted for in matrix  form.

  The element ( 1 , 2 ) represents the delay of input 1  which effects
          output 2 and so on
Enter the delays for the appropriate input and output          01 , 01 )
1
Enter the delays for the appropriate input and output          01 , 02 )
0
Enter the delays for the appropriate input and output          02 , 01 )
```

prompt

menudrive

(SUBMODULE PARAMETER ESTIMATION)

menudrive

prompt

prompt

prompt

prompt

integerbox

readmatrix

```
1
Enter the delays for the appropriate input and output                02 , 02 )
0

                    Delay Matrix
          1     2
     ***********************************************************************
   1 *     1     0
   2 *     1     0
Are you satisfied with these values ? (yes/no)
Y

  Enter the order of the A polynomial corresponding  to the matrix ARMAX
        model .  < Ay = Bu + Ce >


Enter the order of the corresponding A polynomial                    01 , 01 )
1
Enter the order of the corresponding A polynomial                    01 , 02 )
2
Enter the order of the corresponding A polynomial                    02 , 01 )
3
Enter the order of the corresponding A polynomial                    02 , 02 )
4

                    The order of the A polynomials
          1     2
     ***********************************************************************
   1 *     1     2
   2 *     3     4
Are you satisfied with these values ? (yes/no)
y

  Enter the order of the B polynomial corresponding  to the matrix ARMAX
        model .  < Ay = Bu + Ce >
Enter the order of the corresponding B polynomial                    01 , 01 )
2
Enter the order of the corresponding B polynomial                    01 , 02 )
3
Enter the order of the corresponding B polynomial                    02 , 01 )
4
Enter the order of the corresponding B polynomial                    02 , 02 )
5

                    The order of the B polynomials
          1     2
     ***********************************************************************
   1 *     2     3
   2 *     4     5
Are you satisfied with these values ? (yes/no)
Y
```

readmatrix

readmatrix

menudrive

```
            IDENTIFICATION PROCEDURES
            -------------------------

   1) LEAST SQUARES

   2) LEAST SQ  - MAXIMUM LIKELIHOOD

   3) BACK TO PREVIOUS MENU
```

```
                MOD1 MENU (identification)
                --------------------------

   1) FILTER

   2) STATISTICS

   3) PARAMETER ESTIMATION

   4) INPUT-OUTPUT PLOTS

   5) BACK TO MAIN MENU

  Enter the number corresponding to the item of your choice
2
```

menudrive

(SUBMODULE STATISTICS)

```
                DATA SELECTION MENU
                -------------------

   1) ORIGINAL

   2) FILTERED

   3) RESIDUAL

   4) BACK TO MAIN MENU MODULE ONE

 On which data set do you wish to perform statistical calculations on
  Enter the number corresponding to the item of your choice
2
Do you wish to calculate Mean, Variance or Standard Dev
y

Do you wish to calculate the MEAN of input  1
y
Do you wish to calculate the MEAN of input  2
y
Do you wish to calculate the MEAN of output  1
n
Do you wish to calculate the MEAN of output  2
n
Do you wish to calculate the VARIANCE of input  1
y
Do you wish to calculate the VARIANCE of input  2
y
Do you wish to calculate the VARIANCE of output  1
n
```

menudrive

prompt

prompt

prompt

prompt

prompt

prompt

prompt

```
Do you wish to calculate the VARIANCE of output  2                    prompt
n
Do you wish to calculate the STANDARD DEVIATION of  input  1          prompt
y
Do you wish to calculate the STANDARD DEVIATION of  input  2          prompt
y
Do you wish to calculate the STANDARD DEVIATION of  input  1          prompt
n
Do you wish to calculate the STANDARD DEVIATION of  input  2          prompt
n
Do you wish to calculate cross variances ?                            prompt
n
  Statistical Routines                                                modopmatrix

                    Calculation of Mean - Inputs
                          1
      ********************************************************************
     1*    0           d+00
     2*    0           d+00
  press enter for the next page

                    Calculation of Mean - Outputs
                          1
      ********************************************************************
     1*           ********
     2*           ********
  press enter for the next page

                    Calculation of Variance-Inputs
                          1
      ********************************************************************
     1*    0           d+00
     2*    0           d+00
  press enter for the next page

                    Calculation of VARIANCE - Outputs
                          1
      ********************************************************************
     1*           ********
     2*           ********
  press enter for the next page

                    Calculation of Standard Deviation -Inputs
                          1
      ********************************************************************
     1*    0           d+00
     2*    0           d+00
  press enter for the next page

                    Calculation of Standard Deviation -Outputs
                          1
      ********************************************************************
     1*           ********
     2*           ********
  press enter for the next page

Do you wish to calculate the Discrete Fourier Transform
n
Do you wish to copy these results into your .pnt file ?
y


          MAIN MENU                                                   menudrive
          ---------
```

6.2 User Save Files

Print File

```
*********  user d/thesis pnt                             *******************
************* PARAMETER ESTIMATION ******************

==================================================================

LEAST SQUARES

| NDATA    || NU        || NY        || NSTRT    ||
--------------------------------------------------
|      200 ||         2 ||         2 ||        5 ||
--------------------------------------------------

                    Delay Matrix
       1      2
*************************************************************************
  1 *      1      0
  2 *      1      0

                    The Order of the A Polynomials
       1      2
*************************************************************************
  1 *      1      2
  2 *      3      4

                    The Order of the B Polynomials
       1      2
*************************************************************************
  1 *      2      3
  2 *      4      5

                    The Order of the C  Polynomials
       1      2
*************************************************************************
  1 *      0      0
  2 *      0      0

                    The A polynomial estimates
                          1                      2                      3
*************************************************************************
  1*   0             d+00             ********               ********
  2*   0.            d+00    0.             d+00             ********
  3*   0.            d+00    0.             d+00    0.          d+00
  4*   0.            d+00    0.             d+00    0           d+00

                    The A polynomial estimates
                          4
*************************************************************************
  1*           ********
  2*           ********
  3*           ********
  4*   0.         d+00

                    The B polynomial estimates
                          1                      2                      3
*************************************************************************
  1*   0             d+00    0              d+00             ********
  2*   0.            d+00    0.             d+00    0           d+00
  3*   0.            d+00    0.             d+00    0           d+00
  4*   0.            d+00    0.             d+00    0.          d+00

                    The B polynomial estimates
                          4                      5
*************************************************************************
  1*           ********               ********
  2*           ********               ********
```

```
   3*    0          d+00          ********
   4*    0          d+00    0           d+00

                NORMALIZED VARIANCE RESIDUAL
                          1
     ***************************************************************************
   1 *    0.          d+00
   2 *    0           d+00
The Akiake Criterion Number is  0
********** end of parameter estimation results =======

******* STATISTICAL INFORMATION ********

                Calculation of Mean - Inputs
                          1
     ***************************************************************************
   1*    0           d+00
   2*    0           d+00

                Calculation of Mean - Outputs
                          1
     ***************************************************************************
   1*           ********
   2*           ********

                Calculation of Variance-Inputs
                          1
     ***************************************************************************
   1*    0           d+00
   2*    0.          d+00

                Calculation of VARIANCE - Outputs
                          1
     ***************************************************************************
   1*           ********
   2*           ********

                Calculation of Standard Deviation -Inputs
                          1
     ***************************************************************************
   1*    0.          d+00
   2*    0           d+00

                Calculation of Standard Deviation -Outputs
                          1
     ***************************************************************************
   1*           ********
   2*           ********
********** END OF STATISTICAL RESULTS ===============
```

Diary File

```
************  user.d/thesis.dry                    **************
HELP
IDENTIFICATION
INPUT-OUTPUT PLOTS
STOP
FILTER
ORIGINAL DATA
 Default Case
LOW PASS FILTER
LOW PASS FILTER
LOW PASS FILTER
LOW PASS FILTER
PARAMETER ESTIMATION
FILTERED IO DATA
LEAST SQUARES
STATISTICS
FILTERED
BACK TO MAIN MENU
QUIT
```

Two aspects of this work may be considered for future research. The first of these involves completing a working prototype of the package as specified. This would involve assembling existing computational software and incorporating in into the structure and developing software which is not available. In addition, an interactive shell would have to be designed and implemented. The second is the extension of the present package to include other modules which would perform additional features. An example of one is to design and implement an input graphics facility to build systems from block diagrams and compute the closed loop transfer function. This could be viewed as a preliminary step to Module Two, which expects a single transfer function for the plant.

REFERENCES

[1] S.D. Hester. D L. Parnas and D.F. Utter, "Using documentation as a software design medium", *The Bell System Technical Journal*, Vol. 60, No. 8, pp. 1941-1977, October 1981

[2] C.L. McGowan, J.R. Kelly, "*Top down structured programming technique*", Mason/Charter Publishers, 1975

[3] V. Begg, "*Developing expert CAD systems*", Unipub, 1984

[4] C.J. Herget and A.J. Laub, "Editorial: special issue on computer-aided design of control systems", *Control Systems Magazine*, Vol. 2, No. 4, pp. 2-3, December 1982

[5] J R. Rice, "*Matrix computations: mathematical software*", McGraw-Hill, 1981

[6] J J Dongarra et al , "*LINPACK Users' Guide*", SIAM, 1979

[7] B.T. Smith et al., "*Matrix eigensystem routines- EISPACK*", Springer-Verlag, 1976

[8] *Issues in the design of a computer-aided systems and control analysis and design environment*, Oak Ridge National Laboratory 1984

[9] W.F. Arnold and A.J Laub, "Generalized eigenproblem algorithms and software for algebraic ricatti equations", *Proc. of the IEEE*, Vol. 72, No. 12, pp. 1746-1754, December 1984

[10] W.J.M. Lemmens and A.J.W. Van Den Boom, "Interactive computer programs for education and research: A survey", *Automatica*, Vol. 15, No. 15, pp. 113-121, 1979

[11] K.J. Astrōm, "Computer aided modeling, analysis and design of control systems - A perspective", *Control Systems Magazine*, Vol. 3, No. 2, pp 4-16, May 1983

APPENDIX A

```
C-------------------------------------------------------------------
C-------------McGill-UNIVERSITY-CONTROL-CAD--PACKAGE----------------
C-------------------------------------------------------------------

MODULE NAME: yesno

C-------------------------------------------------------------------

CALLING SEQUENCE: CALL yesno ( message, result, display, keyboard )

C-------------------------------------------------------------------

PURPOSE:

This subroutine is a yes/no prompter.  It displays a message provided
by the calling program and then prompts the user for a yes or a no
response.


C-------------------------------------------------------------------
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C CALLED SUBROUTINES:
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C COMMON BLOCKS, :
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C IMPORTANT VARIABLES,PARAMETERS :
CHARACTER*80     message          This string is displayed before the
                                  user is prompted for a yes or a no
LOGICAL          result           If the user inputs a yes ==>
                                  result = .TRUE.  if the user inputs a
                                  no ==> result = .FALSE.
INTEGER          display          the unit number of the device that
                                  this routine has to write to.
                 keyboard         The unit number of the device that
                                  this routine has to read from.


C-------------------------------------------------------------------
C COMPUTING PRINCIPLE,METHOD,ALGORITHM :
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C REFERENCES :
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C COMMENTS :
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C AUTHOR,DATE CREATED : Michael Tessler and Babak Daneshrad
                        July 1984
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C UPDATES :
C-------------------------------------------------------------------

C-------------------------------------------------------------------
```

```
C--------------McGill-UNIVERSITY-CONTROL-CAD--PACKAGE-----------------
C-----------------------------------------------------------------------

MODULE NAME: PROMPT

C-----------------------------------------------------------------------

CALLING SEQUENCE: CALL prompt ( auto0, type, message, dblp,
                                decimal, whole,
                                string, display, keyboard )

C-----------------------------------------------------------------------
PURPOSE:
This subroutine prompts the user for an integer, real,
double-precision, or string variable.  It tries to read the variable
with the appropriate format. If an error is encountered then the user
is prompted again, until the format is satisfied.

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C CALLED SUBROUTINES:
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C COMMON BLOCKS, :
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C IMPORTANT VARIABLES,PARAMETERS :
LOGICAL          auto0          Upon detection of a <CR> without an
                                entry the user is re-prompted if
                                auto0 = .FALSE. If auto0 = .TRUE.
                                then the user is asked "Do you wish to
                                change the variable ?" if the answer
                                to this is 'yes' then the user is
                                reprompted, if 'no' then no read is
                                executed and thus the variable that
                                is supposed to be read by the routine
                                is not read.
CHARACTER*4      type           The type of variable that is desired
                                type=   'int' ==> integer
                                'real' ==> real
                                'dbl' ==> double precesion
                                if type = none of the 3 above entries
                                ==> character string
CHARACTER*80     message        This character string is displayed
                                before the user is prompted for an
                                entry.
REAL*8           dblp           If a double precision number is to be
                                read in then the result of the read
                                is returned to the calling routine by
                                'dblp'.
REAL             decimal        If a  REAL  number is to be read in
                                then the result of the read is
                                returned to the calling routine by
                                'decimal'.
INTEGER          whole          If an INTEGER  number is to be read in
                                then the result of the read is
                                returned to the calling routine by
                                'whole'.
CHARACTER*80     string         If a CHARACTER STRING is to be read in
                                then the result of the read is
                                returned to the calling  routine by
```

```
                                        'string'.
INTEGER         display                 The unit number of the device that
                                        this routine has to write to .
    "           keyboard                The unit number of the device that
                                        this routine has to read from.
C-------------------------------------------------------------------------
C
C-------------------------------------------------------------------------
C COMPUTING PRINCIPLE,METHOD,ALGORITHM :
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C REFERENCES :
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C COMMENTS :
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C AUTHOR,DATE CREATED : Michael Tessler and Babak Dageshrad
                        July 1984
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C UPDATES :
C-------------------------------------------------------------------------


C-------------------------------------------------------------------------
C-------------McGill-UNIVERSITY-CONTROL-CAD--PACKAGE-------------------
C-------------------------------------------------------------------------

MODULE NAME: menudrive

C-------------------------------------------------------------------------
CALLING SEQUENCE: CALL menudrive ( message, name, items,
                                   entries, command,
                                   display, keyboard, choice )

C-------------------------------------------------------------------------
PURPOSE:

This subroutine takes a maximum of 9 character strings; which should
be provided to it by the calling routine, creates a menu by displaying
them in menu format (placing numbers in front of them). The user is
then asked to enter the number corresponding to his choice. Both the
number of his choice and the character string that he chose are then
passed on to the calling routine.

C-------------------------------------------------------------------------
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C CALLED SUBROUTINES: produce
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C COMMON BLOCKS, :
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C IMPORTANT VARIABLES,PARAMETERS :
CHARACTER*80    message             This string is displayed before the
                                    user is asked to choose one of the
                                    menu entries.
```

```
CHARACTER*40     name              The name of the menu.
INTEGER          entries           The number of entries to the menu
                                   ( max 9 ).
CHARACTER*40     items (entries)   The entries to the menu should be
                                   stored in this vector and then passed
                                   onto the routine.
INTEGER          choice            Is the number corresponding to the
                                   menu item that was chosen by the user.
CHARACTER*40     command           On output command = items ( choice ).
INTEGER          display           The unit number of the device that
                                   this routine has to write to .
   "             keyboard          The unit number of the device that
                                   this routine has to read from.

C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C COMPUTING PRINCIPLE,METHOD,ALGORITHM :
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C REFERENCES :
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C COMMENTS :
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C AUTHOR,DATE CREATED : Michael Tessler and Babak Daneshrad
                        July 1984
C-------------------------------------------------------------------------

C-------------------------------------------------------------------------
C UPDATES :
C-------------------------------------------------------------------------
```

```
C-----------------------------------------------------------------------
C-------------McG111-UNIVERSITY-CONTROL-CAD--PACKAGE--------------------
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

MODULE NAME: INTEGER-BOX
C-----------------------------------------------------------------------

CALLING SEQUENCE: CALL integerbox ( charelem,   number1,
                                    number2,    number3,
                                    number4,    number5,
                                    number6,    numberelem,
                                    display )

C-----------------------------------------------------------------------
PURPOSE:
--------
This subroutine takes in 6 different integers and puts them in a box
format for the purpose of displaying data to the user during program
execution.

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C CALLED SUBROUTINES:
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C COMMON BLOCKS, :
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C IMPORTANT VARIABLES,PARAMETERS :

INTEGER          number1-6       These are the 6 numbers that the
                                 program will put into a box format.
   "             numberelem      Is the total number of items to be
                                 put into a box format
                                 0 < numberelem < 6 .
   "             display         The unit number associated with the
                                 display on the terminal.
CHARACTER*8      charelem (6)    This is a character array that on the
                                 input contains the heading for each
                                 number.  This character-string is
                                 printed above the number in the box
                                 to describe it.
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C COMPUTING PRINCIPLE,METHOD,ALGORITHM :
C-----------------------------------------------------------------------

The 6 integers are put into a vector of dimension 6 ( called box ).
The elements of the 2 vectors 'box', and 'charelem' are printed out
with variable formating to allow for the case of having fewer than 6
integers to print out.

C-----------------------------------------------------------------------
C REFERENCES :
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C COMMENTS :
C-----------------------------------------------------------------------
```

```
C-----------------------------------------------------------------------
C AUTHOR,DATE CREATED :     Mike Tessler
                            August 1984
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C UPDATES :
C-----------------------------------------------------------------------
```

```
C-----------------------------------------------------------------------
C--------------McGill-UNIVERSITY-CONTROL-CAD--PACKAGE-------------------
C-----------------------------------------------------------------------

MODULE NAME: READMATRIX

C-----------------------------------------------------------------------

CALLING SEQUENCE:
        CALL readmatrix ( type,          promptmessg,    title,
     1                    heading,       headerflag,     frmt,
     1                    fieldlenth,    dmatrix,        rmatrix,
     1                    imatrix,       nrow,           ncol,
     1                    display,       keyboard,       lowerange,
     1                    upperange,     rangeflag )

C-----------------------------------------------------------------------

PURPOSE:

This subroutine will read in a matrix, display for the user the
matrix that he has just entered, and asks if he wants to correct any
element.  The writting of the matrix onto the screen is done by
another subrbutine called PMATRIX.

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C CALLED SUBROUTINES:   convert, prompt, pmatrix, dblpmatrix
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C COMMON BLOCKS.
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C IMPORTANT VARIABLES,PARAMETERS :
INTEGER          display          The unit number of the output terminal
   "             keyboard         The unit number of the input terminal
   "             nrow             The number of rows of the matrix to
                                  be read in.
   "             ncol             The number of columns of the matrix
                                  to be read in.
   "             lowerange,       The option of restricting the input
                 upperange        to a given range is available
                                  lowerange is the lower limit of the
                                  inputs & upperange is the upper limit
                                  of the inputs.
LOGICAL          rangeflag        If you want to restrict the input to
                                  a certain range 'rangeflag' must be
                                  set to .TRUE. .
   "             headerflag       If a special set of instructions are
                                  to be given before prompting the user
                                  headerflag =  .TRUE. ( the
                                  instruction itself should be stored
                                  in header ).
CHARACTER*320    heading          Is a variable that will be displayed
                                  after every 4-6 prompts (only if
                                  headerflag=.TRUE.) it usually
                                  containes a set of special
                                  instructions.
CHARACTER*4      type             The type of matrix that is to be read
                                  in, ie. type = 'int' ; integer matrix
                                  to be read = 'real', or 'dblp' mean
```

```
                                        real and double precision matricies
                                        respectively.
CHARACTER*80    promptmessage           Each time the user is prompted for an
                                        entry this message is displayed.
CHARACTER*80    title                   The title of the matrix.
CHARACTER*10    frmt                    The format that you wish the elements
                                        of the matrix to be presented in
                                        ie. frmt = 'I10', or 'D17.8', or
                                        'F11.4', 'F6.3'
CHARACTER*4     fieldlenth              The number of character spaces that
                                        each element is going to take upon
                                        formated  display. fieldlenth is
                                        dependant on 'frmt' in the following
                                        manner for the examples; fieldlenth
                                        = '10X', '17X', '11X',
                                        and '06X' respectively
INTEGER         imatrix( nrow, ncol )
REAL            rmatrix( nrow, ncol )
REAL*8          dmatrix ( nrow, ncol )
                                        depending on what type of a matrix
                                        you are reading in the result will go
                                        into the appropriate matrix
                                                integer ==> imatrix
                                                real    ==> rmatrix
                                           double precision ==> dmatrix
                                        the user must keep in mind that in
                                        not to confuse the passing of arrays
                                        to a subroutine and retrieving them
                                        all three matricies must be declared
                                        in the calling program, even if they
                                        are not used ( suggest the using
                                        of scratch arrays ).

C-------------------------------------------------------------------

C-------------------------------------------------------------------
C COMPUTING PRINCIPLE,METHOD,ALGORITHM :
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C REFERENCES :
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C COMMENTS :
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C AUTHOR,DATE CREATED : Michael Tessler and Babak Daneshrad
                                    July 1984
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C UPDATES :
C-------------------------------------------------------------------
C-------------------------------------------------------------------

C-------------------------------------------------------------------
C-------------McGILL-UNIVERSITY-CONTROL-CAD--PACKAGE----------------
C-------------------------------------------------------------------

MODULE NAME; PMATRIX
C-------------------------------------------------------------------

CALLING SEQUENCE: CALL pmatrix ( matrix, nrow, ncol, fmt, fieldlenth,
                                 title, readflag, matrxtype, display,
```

```
                                        keyboard )

C-----------------------------------------------------------------------
PURPOSE:

This subroutine prints out a real or integer matrix with a specified
format provided by the calling routine.  It provides titling, labling
and the works.  The routine can be used alone or in conjunction with
the readmatrix routine.  In the second case the programmer should
only worry about the pass parameters to the readmatrix routine.

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C CALLED SUBROUTINES:   yesno,  prompt
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C COMMON BLOCKS, :
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C IMPORTANT VARIABLES, PARAMETERS :
REAL            matrix ( nrow, ncol )
                                The INTEGER or REAL matrix to be
                                printed out.
INTEGER         nrow            The number of rows of the matrix to
                                be printed.
   "            ncol            The number of columns of the matrix
                                to be printed.
CHARACTER*10    fmt             The format that you wish the elements
                                of the  matrix to be presented in
                                ie. frmt = 'I10', or 'D17.8', or
                                    'F11.4', 'F8.3'
CHARACTER*4     fieldlenth      The number of character spaces that
                                each element is going to take upon
                                formated display. fieldlenth is
                                dependant on 'frmt' in the following
                                manner for the examples; fieldlenth =
                                '10X', '17X', '11X',
                                and '06X' respectively
CHARACTER*80    title           The title of the matrix.
LOGICAL         readflag        Because this routine can be used in
                                conjunction with the readmatrix
                                routine it has some features that are
                                meant to be used with that routine.
                                If readflag = .TRUE. then
                                those features are executed.
                                Generally when using this routine by
                                itself set readflag = .FALSE
CHARACTER*4     matrxtype       The type of matrix to be printed out
                                   matrxtype = 'int' ==> integer
                                   matrxtype = 'real' ==> real
INTEGER         display         The unit number of the output terminal
   "            keyboard        The unit number of the input terminal.

C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C COMPUTING PRINCIPLE,METHOD,ALGORITHM :
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C REFERENCES :
C-----------------------------------------------------------------------
```

```
C-----------------------------------------------------------------------
  COMMENTS :

It should be noted that this routine is only designed for the
printing of integer and real matricies.  However if you would like to
print a double-precision matrix you should use the dblpmatrix routine
which is the exact same thing except that the matrix is declared as a
double-precision matrix.

C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C AUTHOR,DATE CREATED : Michael Tessler and Babak Daneshrad .
                                        July 1984
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C UPDATES :
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C--------------McG111-UNIVERSITY-CONTROL-CAD--PACKAGE-------------------
C-----------------------------------------------------------------------
```

```
C-----------------------------------------------------------------------
C--------------McGill-UNIVERSITY-CONTROL-CAD--PACKAGE-----------------
C-----------------------------------------------------------------------

MODULE NAME: the GRAPHICS_SHELL file
C-----------------------------------------------------------------------
C CALLING SEQUENCE:
C       There are two similar versions of the
C       graphics_shell.f (as well as subs.f) file.
C       The one located in the directory /usr/nonvlsi/control1/graphics
C       has the calling sequence:
C            CALL GRAPHSHELL( A,                        =data matrix
C       &         ma, na,                               =dimensions of A
C       &         Controlmatrix,                        =user supplied control
C                                                        matrix
C       &         mc,nc,                                =dimensions of Controlmatrix
C       &         desirebubble, )                       =positive -->bubblesort data
C                                                        negative -->do not "   "
C       The new (and improved) version of the graphics shell
C       routine, located in /usr/nonvlsi/control1/graphics/changes , has
C       this calling sequence:
C            CALL GRAPHSHELL( A,                        =data matrix
C       &         ma, na,                               =dimensions of A
C       &         Controlmatrix,                        =user supplied control
C                                                        matrix
C       &         mc,nc,                                =dimensions of Controlmatrix
C       &         desirebubble,                         =positive -->bubblesort data
C                                                        negative -->do not "   "
C       &         labelofuser,                          =A character*20 array
C                                                        of size labelsize
C       &         labelsize,                            =dimension of labelofuser
C       &         desireownlabel )                      =positive -->use user
C                                                        supplied labels.
C                                                        negative -->use default
C                                                        labels.
C-----------------------------------------------------------------------
PURPOSE: This file contains interactive routines to be used in
         conjunction with the graphics utility routines located in
         the files SUBS.F , MAIN.F & PDF.F .
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C COMMON BLOCKS, : NONE
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C IMPORTANT VARIABLES,PARAMETERS : The following are a listing of
C                the variables used within the subroutines that compose
C                the GRAPHICS_SHELL.F file ::===>


C REAL*8:        A: This is the data matrix whose columns contain
C                       the data points.
C                Controlmatrix: This is the matrix that the user
C                       supplies. It usually has only around 5 rows.
C                desirebubble: This is an input flag. If it is positive
C                       then the data points will be bubblesorted ; if
C                       desirebubble is negative, bubblesorting will
C                       be surpressed.
C                Control: This is the internally used control-matrix.
C                Ctrl: This is a working control-matrix. It is eventually
C                       sent to the routine GRAPH which interprets it
C                       and plots a screen full of graphs.
C                Ctrl2: This is a temporary matrix that usually holds
```

```
C                    the contents of Ctrl when it is being redimensioned.
C                retrnerror: An internally used flag. If positive, then
C                    an error was encountered .
C                xmode: This is a flag that indicates the format of the
C                    graphs on the screen.
C                Asdf: This is a 5*5 array containing all the possible
C                    screen coordinates that a screen of plots could have.
C                    4*4 is really enough for this matrix.
C INTEGER:       ma , na : These are the dimensions of the data matrix, A.
C                mcm , ncm: These are the dimensions of the user supplied
C                    control-matrix, Controlmatrix.
C                mc , nc: These are the dimensions of the internally used
C                    control-matrix, Control. Note that ( nc := ncm ).
C                nctrl: This is the number of columns of the abbreviated
C                    control-matrix, Ctrl. Note that it has mc rows.
C                nscreens: The total number of screens.
C                nwindows: The total number of windows (or plots (or graphs)).
C                nextra: The number of windows on the last screen.
C                screennumber: The present screen's number.
C                Ipoint: This is an integer vector of size nc. It is a pointer
C                    to the columns of the matrix Control, giving
C                    an indication as to the location of each of the
C                    plots. For example if Ipoint = (1, 3, 5, 6) this
C                    would mean that the information concerning the
C                    first window starts on column 1 (naturally!) and
C                    the control info for the second plot starts at
C                    column 3, the third plot at column 5 and finally the\
C                    info for the fourth plot starts at column 6  Note
C                    that from this, one an deduce that window one has
C                    (3-1) or 2 graphs on it and simailarily window 2
C                    contains (5-3)=2 funcions and the third window
C                    contains only one function. The last plot contains
C                    (nc - 6 ) + 1 plots on it.
C                    WARNING !!!!!!!!: It is important to note that
C                    this vector has nothing whatsoever to do with
C                    the integer vector of the same name that is found
C                    within the files MAIN.f and SUBS.F. The are COMPLETELY
C                    independent.
C                windows: The number of windows on the present screen.
C                plot1,plot2 : Temporary variables used in the OVERLAY routines.
C                column: a temporary column pointer.
C                newctrl (sometimes also called newnctrl): This is the
C                    new value that nctrl assumes whenever extra columns
C                    are added onto Ctrl.
C                naugctrl: The amount nctrl should be augmented by , or in
C                    other words, the number of extra columns that are
C                    to be  added onto Ctrl.
C LOGICAL:       abort: A flag indicating that a subroutine was aborted.
C                addmore: A flag used in the CREATESCREEN routines indicating
C                    that the user wishes to add more windows.
C                result,auto0 : Used by Bob's interactive utility routines.
C CHAR*80:       message: Used by the utility routines
C                Label(4) : A set of labels for up to four plots on a window
C                    that is eventually sent to the GRAPH routine.
C CHAR*40:       name,
C                item(20),
C                command : Used by the interactive utility routines.
C CHAR*15:       Labellist(5) : A set of elementary labels for each of the
C                    five different types of graphs that this package
C                    is capable of plotting.
C CHAR*4:        type: Used by the utility routines.
C
C ------------
C The following are variables that are used by the new graphics routines
C that are located in /usr/nonvisi/control1/graphics/changes
C
```

```
C REAL*8          XMINMAX(2,1000) . This is a COMMON block array
C                                 that is used in the FINDMAX subroutine
C                                 It keeps track of the min's and max's
C                                 of the columns of the data matrix, A,
C                                 so that they do not have to be
C                                 recalculated over and over again
C                 desireownlabel: If this flag is positive, then the
C                                 graphics program will use the labels
C                                 that the user supplies for the first
C                                 "labelsize" plots.
C CHARACTER*20:   labelofuser:    This is a character array of size
C                                 "labelsize", containing the labels
C                                 that the user desires for the first
C                                 "labelsize" plots.
C INTEGER:        labelsize:      The number of labels in the character
C                                 array Labelofuser. (i.e. It is
C                                 the dimension of Labelofuser)
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C AUTHOR,DATE CREATED : Vasu Iyengar & Ajit Nilakantan ; Summer 1984
C-----------------------------------------------------------------------

C=======================================================================
```