RL-TM-96-8
In-House Report
June 1997

A STATISTICAL PATTERN RECOGNITION TOOL

Shaun P. Montana

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

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

RL-TM-96-8 has been reviewed and is approved for publication.

APPROVED:

RONALD W. CWIRKO
Chief, Signal Intelligence Division
Intelligence & Reconnaissance Directorate

FOR THE COMMANDER:

JOSEPH CAMERA, Technical Director
Intelligence & Reconnaissance Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify Rome Laboratory/IRAP, Rome, NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1997	In-House, Jun 96 - Aug 96

4. TITLE AND SUBTITLE

A STATISTICAL PATTERN RECOGNITION TOOL

5. FUNDING NUMBERS

PE - 62702F
PR - 4594
TA - 15
WU - 2F

6. AUTHOR(S)

Shaun P. Montana

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Rome Laboratory/IRAP
32 Hangar Road
Rome, NY 13441-4114

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory/IRAP
32 Hangar Road
Rome, NY 13441-4114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TM-96-8

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Andrew J. Noga, IRAP, 315-330-4581. The author performed this work while employed under the summer Engineering Aide Program at Rome Laboratory.

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; Distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The development of a MATLAB based statistical pattern recognition software package (referred to as STATPACK) was begun. Initial developments include a two-dimensional coordinate projection capability, and a two-dimensional eigenvector projection capability. The operating system's directory structure has been utilized to allow for multiple levels of data separation algorithms in subsequent development.

14. SUBJECT TERMS

coordinate vector projections, eigenvector projections, pattern recognition

15. NUMBER OF PAGES

46

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	U/L

Table of Contents

1.0 Introduction

This report summarizes in-house work performed under the Rome Laboratory Summer Engineering Aide program, during the June 1996 to August 1996 time frame. During this time, a statistical analysis package (STATPACK) has been developed for general use in pattern recognition problems. STATPACK provides a user with the capacity to perform statistical pattern analysis and recognition functions similar to those available in the On-line Pattern Analysis and Recognition System (OLPARS) [1] but in the MATLAB environment [2]. Data is loaded in and analyzed using functions similar to those developed for OLPARS, but these functions are written in MATLAB code as opposed to the FORTRAN that OLPARS used.

The MATLAB environment is designed to handle the storage, processing, and display of data represented in matrix form. MATLAB can plot in two and three dimensions, with various markers for scatter plots. MATLAB also has the ability to display markers in numerous colors, both on the screen and in printouts of plots. Taken together, these and other abilities make MATLAB the ideal environment for the implementation of classical pattern recognition algorithms. STATPACK has been developed in an effort to bring the usefulness of OLPARS to a user-friendly, graphically oriented, matrix math environment, which readily lends itself to further extensions.

2.0 STATPACK Development

For test and demonstration purposes, the OLPARS package contains the data file "nasa.dat" which is used throughout their Instruction Manual and User's Guide in examples and

demonstrations. In a similar fashion this file was also used as the basic data file throughout STATPACK's development. This data file follows a standard OLPARS format, which is necessary for the proper usage of STATPACK. The format is as follows: an eight character maximum length class name at the beginning of each vector, followed by comma separated (or delimited) data, with a semi colon at the end of each vector. (Even if a class name is not eight characters long, the area of a class name must occupy eight spaces so that all class names are the same length.) A portion of the nasa.dat file is shown below:

```
soy    ,169.,173.,193.,191.,164.,163.,189.,166.,183.,173.,159.,178.;
soy    ,172.,176.,194.,193.,165.,164.,189.,167.,182.,170.,156.,174.;
soy    ,169.,173.,193.,190.,165.,163.,188.,165.,179.,164.,163.,177.;
/*
```

While nasa.dat followed this format exactly and STATPACK was developed to accept comma delimited files, data files need not be comma delimited. At the end of the data file is the standard OLPARS end of file (EOF) character, a slash followed by an asterisk (/*).

2.1 Data File Considerations

While MATLAB does have the ability to read in comma delimited files like nasa.dat, simply reading this data into a matrix was not sufficient for STATPACK's design. In the early stages of STATPACK's development each vector was read in, the commas and semi colons were removed from the data and the resultant data was entered into an ASCII text file. This process worked well for files such as nasa.dat, but would be insufficient for larger files. A data file larger than nasa.dat would occupy excessive amounts of disk space once it was transferred to ASCII text. The augmentation of a vector ID to the data, a number that could be used to identify a

vector in subsequent processing, to the data increased the amount of memory needed to store a large data file. In order to read from such a text file, the entire file would have to be read into a matrix line by line before any data could be processed. The amount of time required to read in data in this manner was likewise excessive for large data files. A further difficulty in using these ASCII files was establishing connection between the vectors and their classes. Creating another ASCII text file that contained associated class names only resulted in more memory used up, especially for larger data files. Instead of reading data to ASCII text files, an alternative method was developed using one of MATLAB's features.

MATLAB has the ability to save data in a matrix in a .mat file format [3]. This uses binary representation for the data instead of ASCII text representation, and takes up less disk space. The data is also loaded by a simple 'load <filename>' command, instead of being read in line-by-line. Using .mat files would solve the problems of excessive disk space and excessive time. Additionally, it would be easier to correlate vectors in a matrix with their class names, by placing this information in a separate matrix. Even large data files could be stored in a matrix that occupied less memory than ASCII text files and loaded back relatively quickly for the processing of the data.

2.2 Data File Conversion: FILEIN

The STATPACK routine FILEIN reads in data line-by-line from an ASCII text file and removes the class name, commas (if they exist), and semi-colons from the data. Three main files are created in the FILEIN process: transfer.dat, w_tre.mat, and c_tre.mat. The first file created,

transfer.dat, contains the class name of each vector, and the corresponding vector ID. A sample from a transfer.dat file is shown below:

```
soy 1
soy 2
soy 3
```

It is an ASCII text file and is used primarily in the creation of c_tre.mat and three other files to be explained later. The file w_tre.mat is a binary file of a matrix named w_tre which contains the data itself followed by a column of vector IDs and a column of an on/off switch. This switch is initially a column of ones (the default value, standing for 'on') that can be changed to a column of zeros (standing for 'off') and is used by various structure analysis functions. A sample w_tre matrix is shown below:

```
169  173  193  191  164  163  189  166  183  173  159  178  1  1
172  176  194  193  165  164  189  167  182  170  156  174  2  1
169  173  193  190  165  163  188  165  179  164  163  177  3  1
```

The vector IDs are a system of identifying a particular vector at any time. The first vector read in is assigned the number one; the second, the number two, and so forth. The vector IDs are an association between w_tre and c_tre. The file c_tre.mat, the third major file created, contains a matrix called c_tre which holds a class filename for every vector, with the vector ID implicit in the row of the matrix the class filename is in. A sample c_tre matrix is shown below:

```
soy.s_1
soy.s_1
soy.s_1
```

The class filename consists of the classname with a three character extension. The first two characters of this extension are based on the case of the first letter of the classname. If the first

letter is lowercase, the first two characters of the extension are the letter followed by an underscore. If the first letter is uppercase, then each of the first two characters of the extension are the same letter. The third character of the extension is one of the first fifteen hexadecimal digits, which corresponds to the color used in plots to represent a particular class.

The class filename convention was created for the way a data tree is displayed in STATPACK, using the MATLAB routine UIGETFILE. FILEIN creates directories under a main directory, \statpack\data, by taking the name of the data file and replacing the extension with three zeros (nasa.000, for example, for nasa.dat). This constitutes the main data node for a particular data set. Subsequent subnodes would be set up in the same fashion, except that the next highest number would be used (001, 002, etc.). In this directory is the class filename for each separate class (which are listed on the left hand side using UIGETFILE; the current node and other potential nodes are listed on the right hand side). OLPARS used a more standard data tree format, with the actual tree drawn out. The UIGETFILE setup constitutes a more organized data tree, in a format that any Windows user is already very familiar with.

2.2.1 Data Subdirectories

Three smaller files are created by FILEIN through the function ISUNIQUE. ISUNIQUE actually develops the class filename that is stored in c_tre by reading in the classname from transfer.dat. ISUNIQUE also generates the matrices classlist, file_list, and colorlist, and uses these to keep track of the different classes used and filenames developed, so that new filenames are created for classes only when a new class is encountered. The matrix classlist, saved as

c_list.mat, contains a list of each different classname from the original data file. The matrix file_list, saved as filelist.mat, is the same as classlist, except that it contains the individual filenames instead of the classname. The matrix colorlist, saved as clr_list.mat, is a list of the fifteen possible colors than can be used, with the hexadecimal digit of the class filename extension corresponding to the row containing the color assignment for that class. A text file, colors.txt, is included to tell the user what the RGB (red-green-blue) triples in colorlist represent.

The six files created are hidden by FILEIN so that when the data tree is viewed, only the "filenames" that represent the different classes of data are seen. This is done through the usage of DOS .bat files, which are created by FILEIN, executed, and subsequently deleted. FILEIN then updates a file called CDIRECT with the name of the new data node/directory that has just been created. CDIRECT is what will be referred to as a "dynamic .m" file. These are .m files that exist when MATLAB is started, but are modified by other .m files prior to execution. At startup, MATLAB goes through a path of directories in which it looks for all potential .m files the program can run. Creating a new .m file by an existing .m file is thus possible, but the new file will not be included in the path unless it exists during startup. Therefore, CDIRECT needs to be created by the user as an empty file before STATPACK is first used and is amended whenever necessary, with no static code. Once it is created, and after FILEIN is performed, it contains the single line "cd c:\statpack\data\<current node>". The final part of FILEIN uses UIGETFILE to put up a 'File-Open' style window which displays the data tree with the newly created node selected and the "files" under that node. Exiting this viewer is done by selecting one of the files and hitting the 'OK' button or simply hitting 'Cancel'.

2.3 Two Dimensional Projections: S2CRDV, S2EIGV

The two main processes that can be performed after data has been loaded in are S2CRDV and S2EIGV. A variety of functions were available in OLPARS, including structural analysis functions (other than S2CRDV and S2EIGV), logic design functions, data tree functions, logic tree functions, display functions, and information functions. S2CRDV and S2EIGV were chosen for STATPACK as they were two of the simplest OLPARS functions and two of the most useful in terms of data manipulation and analysis. These functions were also the most likely to take advantage of MATLAB's graphical capabilities, as opposed to logical classifiers which would use more uicontrols and other Windows-style interfaces instead of graphics features. Plots also existed for the nasa.dat data file that had been generated by using OLPARS's S2EIGV function, so that the results of STATPACK's S2EIGV could be compared to a known result. Further details regarding the usage of STATPACK are given in the proceeding sections.

3.0 Initialization and Overview of STATPACK

Before running STATPACK, a directory must be created on the user's hard drive called 'statpack'. Two subdirectories must then be created: 'data' and 'source'. All source code is copied into 'source'. Two empty files must be created in the tools subdirectory of source: cdirect.m and d_return.m. An empty matrix called node_list must be created and saved as nodelist.mat. (Before processing, this file should be hidden by changing its attributes.) These files are updated by FILEIN and other functions but must exist beforehand. Finally, the function SP_MOVIE must be run. This creates the movie that is seen when STATPACK starts up. This function is used only

once, the first time the program is used after a new installation. It must be run while the current MATLAB directory is '\statpack\data' as the first command for STATPACK is to change to this directory.

An analysis session is initiated by running the STATPACK function. The directory is changed to '\statpack\data' and two screens with no menus and black backgrounds are created. The larger one becomes the main screen and the smaller is used to show the movie. The movie is then loaded and played. When it is finished, the window containing it closes and the main screen is initialized with the title 'STATPACK Main Screen' and the STATPACK menus. A box of text welcoming the user and providing information on how to proceed is displayed in the center of the screen. The user can then access various commands through the STATPACK menus.

3.1 STATPACK Menus

Currently on the STATPACK main screen there are six menus that a user can access to perform various tasks and execute functions (see Appendix A, Sample Menus, Submenus, and Uicontrols). 'File' contains three submenus. 'File In' invokes the FILEIN command, and 'File Out' (which is not enabled) is included in anticipation of a FILEOUT command. 'Exit' exits the program and closes any STATPACK related windows that are open. The next menu, 'Data Node', has three submenus. 'Selection' invokes the CH_CDRCT command, which allows the user to change CDIRECT and thus select a new node at which to work. This is done by putting up a window that contains a popup menu with all the nodes that exist whatever the current node is, a major node or subnode. 'Draw Tree' uses UIGETFILE to display the available nodes and the files

contained in the current node. This viewer is exited by hitting the 'Cancel' button. 'Current' displays the name of the current node in a window. The third menu, 'Analysis', contains two submenus: '1D Structure' and '2D Structure'. The submenus of '1D Structure', 'Coordinate Projection' and 'Eigenvalue Projection', are currently not enabled as these functions have not been written for STATPACK. The submenus of '2D Structure' have the same names but are enabled, beginning either S2CRDV or S2EIGV, respectively. The fourth menu is 'Classify', which has no enabled submenus as STATPACK currently has no logical classifiers to divide data. The fifth menu, 'Help', contains a list of all the functions which are currently listed under the four previous menus. A help display for FILEIN is currently the only enabled submenu for this menu. The sixth menu, 'About', displays a screen with information that is pertinent to STATPACK and it's creation.

3.2 Major Functions: S2CRDV, S2EIGV

In addition to FILEIN, STATPACK currently has two major functions: S2CRDV and S2EIGV. These functions allow the user to analyze the data that FILEIN loads. Both produce two-dimensional plots and are referred to as structural analysis functions. The primary difference between the two functions is that S2CRDV is a basic coordinate projection while S2EIGV computes the covariance matrix of the data, and then computes the eigenvalues and eigenvectors for data projections.

3.2.1 Two Dimensional Projection: S2CRDV

S2CRDV starts by loading in the pertinent data contained in w_tre.mat and the "filenames" contained in c_tre.mat and filelist.mat. The column of vector IDs and the on/off switch column are removed from w_tre, leaving only the data. The file filelist.mat is then accessed for the list of "filenames" at that node. The program empties these files of anything they might contain and then puts the vector IDs of the vectors of each class into the appropriate files. This information is used later in plotting the projection. Two matrices which are used for plotting purposes are then initialized. The first, file_order, is the order in which in classes are plotted. This is changed at need by HIDESHOW. The second, hide_show, is a row vector of ones (the default value, standing for 'show') used to determine whether a class is visible on the plot ('shown') or plotted in a dark gray ('hidden'), and is also changed as necessary by HIDESHOW. The data itself is then transposed to determine the number of features in the data set and for the selection of features. A window, called 'Feature Selection', is generated, the size of which is based on the number of features. (A larger number of features requires a correspondingly large window size.) A checkbox is displayed for each potential feature. STATPACK has the ability to display up to 236 checkboxes for different features in the largest window. For large data sets, it is recommended that the user have a large amount of available memory for the displaying of such a large number of checkboxes. Each checkbox is generated by a MATLAB command called UICONTROL; thus, 236 checkboxes requires 236 calls to the UICONTROL function and 236 independently supported checkbox objects are displayed.

The user is asked to select two features to use in the projection. The user is warned that if

more than two are selected, only the first two will be used. Upon hitting the 'Done' button, the next section of S2CRDV code cycles through and determines which features the user selected for the projection. If not enough features are selected, an error window is displayed, and informs the user of the error. The program then returns to the 'Feature Selection' window. When the user has selected two of the features, two column vectors are generated, with a column per number of features, all entries being zero. The first feature selected is indicated by a one placed in the corresponding column in the column vector. (For example, if the user selected feature 8 and feature 11, the first column vector would contain a one in the eighth column and the second column vector would contain a one in the eleventh column.) The vectors are combined into a matrix which is used to project the data. (See Section 4.0, Mathematical Considerations, for more information.) A plot window called 'Feature Projection Plot' is generated (with a black background). The D_RETURN function is then updated with the name of the function that is currently in use (the S2CRDV function). This allows for returning to the correct program when a function that changes a plot is invoked. On the plot window, a set of menus additional to the standard menubar are created, each containing submenus that can manipulate the window or the data. The projection is calculated (see Section 4.0, Mathematical Considerations) and a set of axes is generated. The axes are labeled by the two features selected, and a title is given to the plot. This title displays the node name, the date on which the plot was produced, and the time at which the plot was produced. Each of the generated data points is then graphed as a black dot, invisible on the black background of the screen. A sample coordinate projection is shown in Figure 3.2.1-1.

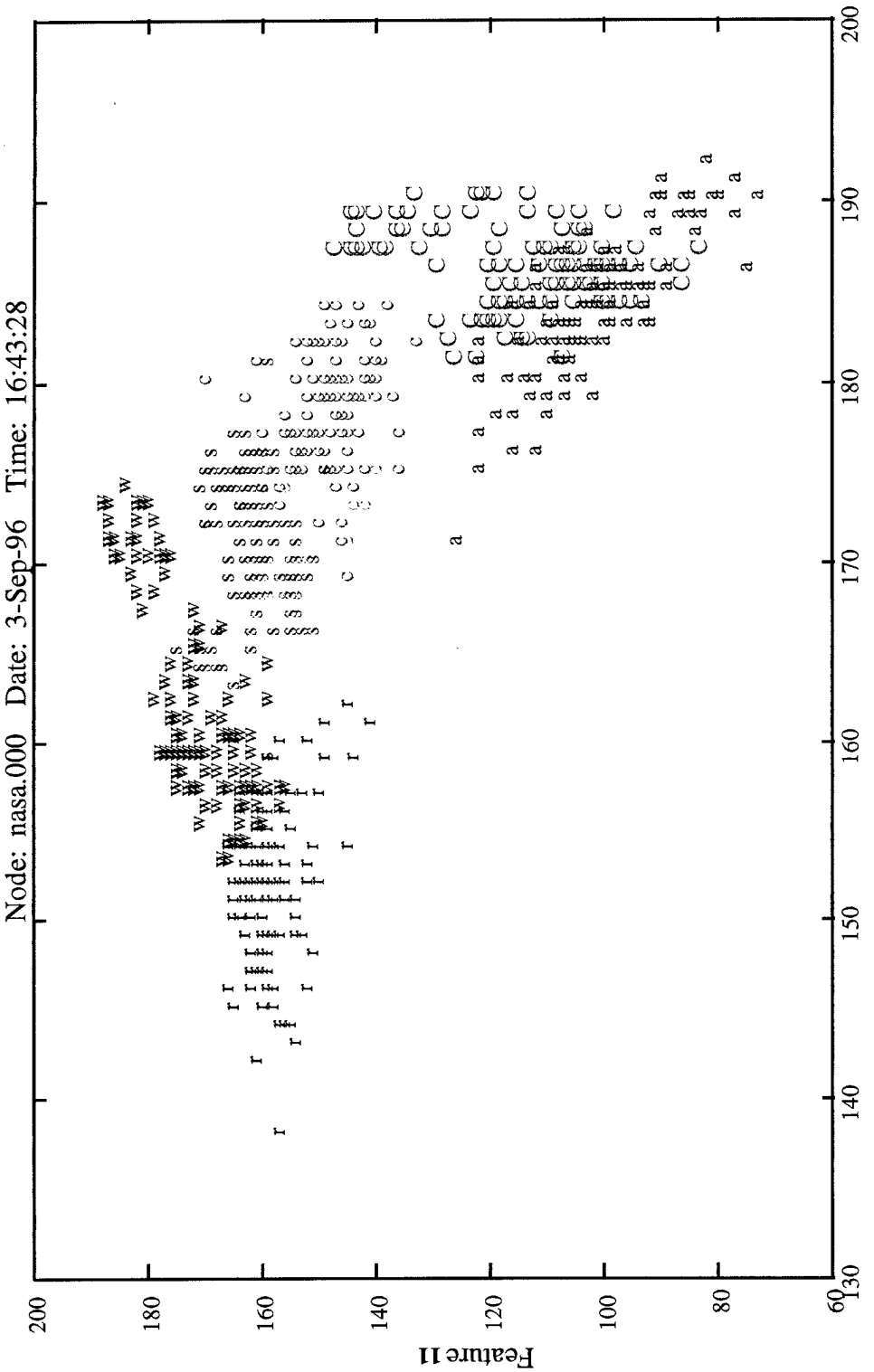


Figure 3.2.1-1. Sample coordinate vector plot.

The class filenames are now accessed according to file_order for the vector IDs they contain. The vectors for each class are then removed from the data and the projection for just that class is calculated. Then, the class filename itself is parsed to determine the letter that will be placed on the plot for that class, and the color the letter will appear in. The matrix hide_show is then accessed to see if this class has been selected to be 'hidden' or 'shown'. If 'shown' it will appear with the correct letter and color. If 'hidden' it will appear on the plot with the correct letter, but will be graphed in a dark gray. All the matrices created in this space are then cleared and the loop continues until all classes have been plotted. Any .mat files that may have been created by S2CRDV are hidden in the same manner as those created by FILEIN. Finally, the user can learn the ID of a vector by clicking on it's letter on the plot. (A further discussion of this capability and the plot's menus can be found in Section 3.3, S2CRDV, S2EIGV Plot Menu Commands.)

3.2.2 Two Dimensional Projection: S2EIGV

S2EIGV begins in the same way as S2CRDV, with the data loaded in and set up, the vector IDs entered into the "filenames", and hide_show and file_order initialized. The data is transposed and the number of features is again determined. A window called 'Feature Elimination Selection' is set up with the number of checkboxes equal to the number of features, and the necessary window size. The limitations are also the same, with the maximum number of features displayable being 236. The user is told to select any features to be eliminated from the covariance calculation. The user is also warned that at least two features must be left on or the program will

not be able to continue.

The code then checks to make sure that at least two features were left on and displays a user error window if not, informing the user of the error made and then returning to 'Feature Elimination Selection'. Subsequently, the code updates the data accordingly if a feature was selected to be eliminated. It then checks the on/off switch column of w_tre to see if a vector or vectors were selected to not be included in the covariance calculation. The data is updated accordingly and the covariance matrix is calculated. The eigenvalues and eigenvectors of this covariance matrix are then calculated and the eigenvalues are arranged in descending order. The eigenvectors are then ordered so as to correspond to their eigenvalues. A second window, called 'Eigenvalue Selection', is then created, with checkboxes matching the number of eigenvalues (which is also equivalent to the number of features). Again, the limitations are the same, and the size is dependent on the number of eigenvalues. The user is told to select two eigenvalues and that if more are selected, only the first two will be used.

The code now proceeds to check to make sure only two eigenvalues were selected and displays an error window if this does not occur. The program would then return to the 'Eigenvalue Selection' window. If two eigenvalues were selected, the first two are determined and their corresponding eigenvectors are put into column matrices. These individual matrices are then combined into one matrix which is used in the data projection.

The last section of the S2EIGV code displays a plot window (also with a black background) called 'Eigenvector Projection Plot', which has a set of menus in addition to the standard menus. (For more information regarding these menus, see Section 3.3, S2CRDV,

S2EIGV Plot Menu Commands.) The data projection is calculated and each data point is plotted initially as a black dot to establish plot dimensions and axes. The D_RETURN function is then updated with the name of the function that is currently in use (the S2EIGV function). Any .mat files that have been created by S2EIGV are then hidden, in the same process used in FILEIN and S2CRDV. The axes are labeled with the two eigenvalues, and a title is placed on the plot. This title contains the node name, the number of features used, and the date and time of the plotting.

The class filenames are accessed according to file_order for the vector IDs they contain. The vectors for each class are then removed from the data and the projection for just that class is calculated. Then, the class filename itself is interpreted to determine the letter that will be placed on the plot for that class, and the color the letter will appear in. The matrix hide_show is then accessed to see if this class has been selected to be 'hidden' or 'shown'. If 'shown' it will appear with the correct letter and color. If 'hidden', it will appear on the plot with the correct letter, but will be graphed in a dark gray. All the matrices created in this space are then cleared and the loop continues until all classes have been plotted. As with the 'Feature Projection Plot', the user can select the letter that represents a vector to identify it. A sample eigenvector projection is shown in Figure 3.2.2-1, using the two largest eigenvalues.

Of the two Structure Analysis functions, S2EIGV often produces a more distinct separation of classes. In the standard test set nasa.dat, for S2CRDV the "best" overall separation of data occurs by selecting features 8 and 11. For S2EIGV, the "best" overall separation occurs by selecting the two largest eigenvalues and using all 12 features in the covariance calculation.

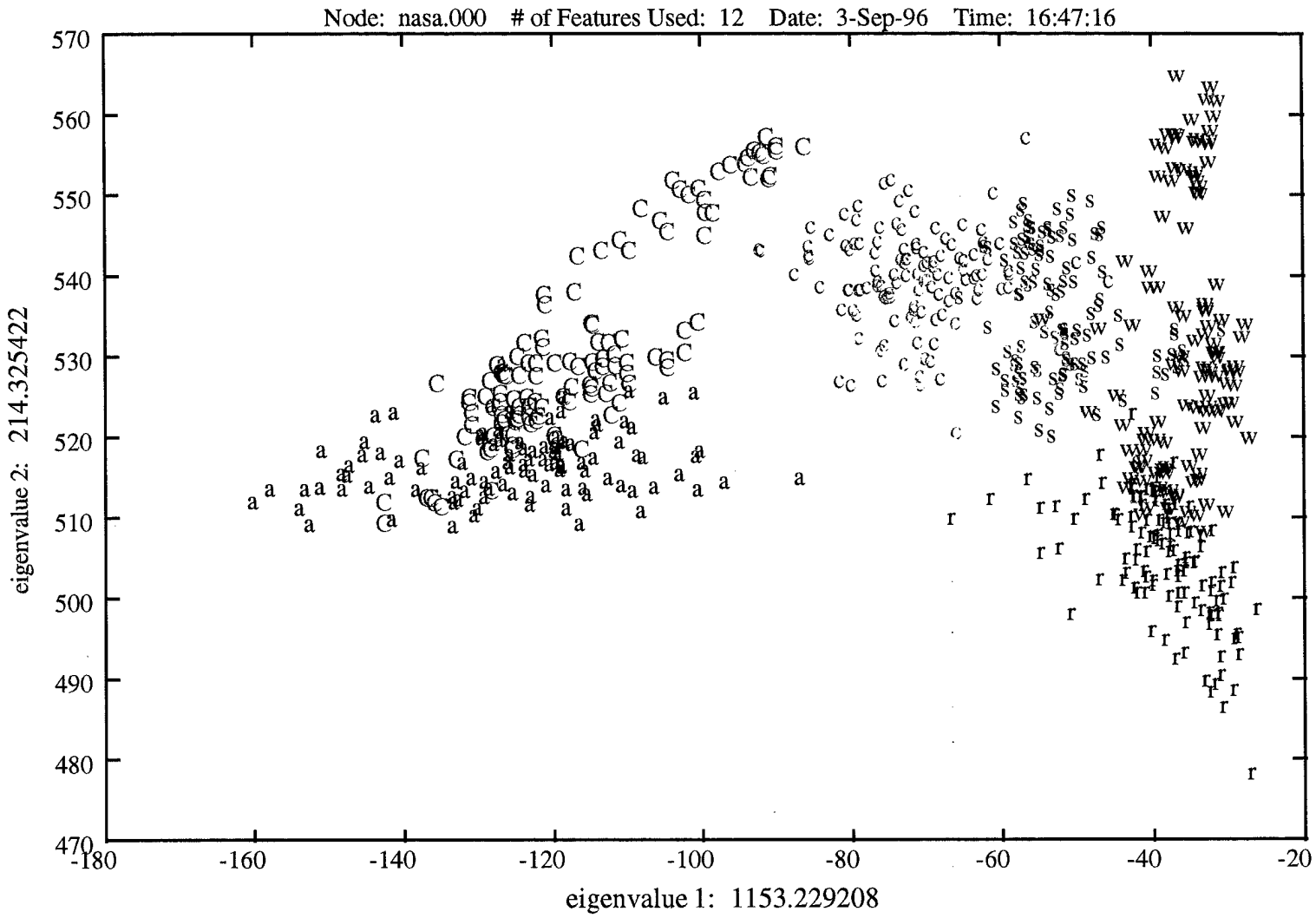


Figure 3.2.2-1. Sample eigenvector plot.

3.3 S2CRDV, S2EIGV Plot Menu Commands

The 'Feature Projection Plot' and 'Eigenvector Projection Plot' windows each include the standard menubar and a list of additional menus with various commands. The standard menubar consists of four menus: 'File', 'Edit', 'Window', and 'Help'. The 'File' menu contains six different options: 'New Figure', 'New' with three submenus of 'M File', 'Figure', and 'Model', 'Close', 'Print', 'Printer Setup', and 'Exit MATLAB'. 'Edit' contains three different options: 'Copy', 'Copy Options', and 'Clear Figure'. 'Copy' and 'Copy Options' deal with saving the current plot for importation to a document or presentation. Under the 'Windows' menu is a list of the current windows created by MATLAB. This allows the user to switch back and forth between various insert windows. The last standard menu, 'Help', contains the 'Table of Contents' and 'Index' of MATLAB Help.

The next menu for both plots is called 'View'. This contains four commands. 'Zoom On' activates the ZOOM function, which allows the user to select an area of a plot to view more closely. It is initialized upon plotting. 'Zoom Off' is initialized only when 'Zoom On' has been activated and turns the ZOOM function off. 'Zoom Out' returns the window to it's original settings and ends the ZOOM function. 'Square Axes' will set the length of the axes to be the same, based on whichever is larger. 'Zoom Out' will negate the effects of 'Square Axes'.

The next menu, 'Select', varies for the different functions. For S2CRDV, there are two submenus. 'Features' allows the user to re-select the features the data will be projected onto. 'Vectors' invokes the VIDCLICK function that is also invoked as the last action of S2CRDV and S2EIGV. This function allows the user to use the mouse to select a vector with the vector's ID

displayed in a window. The user then has three options: 'OK' allows the user to select another vector, 'End' ends the function, and 'Hide/Show' invokes the HIDESHOW function. This allows the user to 'hide' and 'show' various classes in the case where classes overlap. Any classes that are 'hidden' are graphed in a dark gray and cannot be selected by VIDCLICK. Selecting the 'Hide/Show' button will negate the effects of any use of ZOOM or 'Square Axes'.

The third additional menu for both S2CRDV and S2EIGV, 'Plot', contains five different commands. 'Refresh' uses the MATLAB command REFRESH to clear the screen of any errors in displays (but not data errors). 'Refresh' will not affect any zooms, 'Square Axes', labels, classes that have been 'hidden', or vectors that have been turned off (for S2EIGV only). 'Re-Plot' replots the data, and negates the effect of any zooms, 'Square Axes', or labels. Classes that have been hidden and vectors that have been turned off (for S2EIGV only) are not effected by this command. This does not change the features or eigenvalues selected. 'New Plot' allows you to construct a totally new plot and negates all potential menu commands. 'Label Plot' accesses the LABEL command to place text labels on the plot. 'Close All Plots' will close all plots and connected windows of the function that is currently operating.

3.3.1 Further Capabilities of the Eigenvector Projection Display

The 'Eigenvector Projection Plot' has five total menus in addition to the standard menubar. The 'View' and 'Plot' menus are the same as those in the 'Feature Projection Plot'. The 'Select' menu is the same with the exception that 'Features' is replaced by 'Eigenvalues' and the user can use this command to select two new eigenvalues. The other two menus are 'Eliminate' and

'Uneliminate'. Each contains two commands. The submenus of 'Eliminate' are 'Features' and 'Vectors'. The 'Features' submenu allows the user to select different features to be eliminated from the calculations and plots this new data. The 'Vectors' submenu allows the user to enter one or more vector IDs to have those vectors not included in the covariance calculation but otherwise included in the data. This is invoked via the function V_OFF, which makes use of the on/off switch column of w_tre. Any vectors selected to be 'off' are graphed in a dark gray but can still be selected through 'Select Vectors'. The 'Uneliminate' submenus are 'Vectors' and 'Reset All'. The 'Vectors' submenu invokes the opposite of V_OFF, the V_ON function, which allows the user to enter the IDs of vectors that were turned 'off' to be turned back 'on'. The 'Reset All' submenu turns 'on' all vectors that were turned 'off' and replots the data. Any of these commands would negate the effects of any zooms, 'Square Axes', or labels. Hidden classes would not be affected, nor would the features that were eliminated or the eigenvalues.

3.3.2 Vector and Class Selections

VIDCLICK is automatically invoked as the last process of S2CRDV and S2EIGV. This function uses the MATLAB function GINPUT to allow the user to click on a single vector. The coordinates selected are then compared to all the potential coordinates of all data points and the smallest difference between these results in the identification of the vector. If a vector has been 'hidden', clicking on it will result in the return of the ID of the vector which is closest to the point that was clicked on. Clicking on a blank part of the screen produces the same result. The ID is displayed in a window called 'Selected Vector' with three options. The first one, 'OK', allows the

user to click on another vector. The second, 'Hide/Show', invokes the HIDESHOW function. The third, 'End', turns VIDCLICK off.

HIDESHOW 'hides' or 'shows' classes as par the user's selections. A class that overlaps another can be 'hidden' for the user to view the covered class in its entirety. Any classes that are hidden cannot be selected by VIDCLICK. The hide_show matrix initialized near the beginning of S2CRDV or S2EIGV is first checked to see if any classes are already off. Then the number of classes in the data set is determined, and a figure window is created, showing a checkbox for each class. Size determination and limitations are the same as any other window containing checkboxes. Any classes that are 'hidden' will have their checkboxes checked; to 'show' a 'hidden' class, the check needs to be removed. Clicking on an empty checkbox will result in a class being 'hidden'. The user selects the 'Done' button in the corner to continue. The program will then return to the function that called it, either S2CRDV or S2EIGV, via the information provided in the D_RETURN function. The data with classes appropriately 'hidden' or 'shown' will then be plotted.

4.0 Mathematical Considerations

The covariance calculation performed during the execution of S2EIGV is accomplished by the function VAR_COV through the following process [4]. The process results are the same as those obtained through usage of the MATLAB function COV. Data read in through FILEIN must be transposed before the calculations of this method will work.

A data matrix of dimensions p x n (with p rows (or variates) and n columns (or samples))

is postmultiplied by a n x 1 column vector of ones and then multiplied by the scalar value $1/n$. This generates a sample mean of the data. By postmultiplying the sample mean by the (1 x n) transposed "ones" column vector, a matrix of means is obtained. Next, this matrix of means is subtracted from the original data, which produces a (p x n) matrix of deviations. Multiplying the matrix of deviations by its transpose, and then dividing by the scalar element (n - 1) results in the sample covariance matrix of the data.

For S2EIGV, the eigenvalues and eigenvectors of this covariance matrix are calculated and two eigenvalues are selected by the user. Their corresponding eigenvectors are used in the actual projection of the data. To obtain the projection, the data matrix or vectors are pre-multiplied by the two eigenvectors, which have been combined as a 2 x p matrix. For example, the nasa.dat data set which contains p = 12 features and n = 847 vectors would have 12 eigenvalues, each with it's own corresponding eigenvector. Two eigenvalues would be selected. Their corresponding eigenvectors (each of length 1 x 12) would be compiled in a 2 x 12 matrix. The original data set of size 847 x 12 would be transposed to a size of 12 x 847. Then the data set's transpose would be pre-multiplied by the eigenvector matrix, with the resulting 2 x 847 matrix containing, in each column, the respective x and y coordinates for the vector corresponding to that column. In STATPACK, the first row is used as the x coordinates, and the second row are the y coordinates. The function would then plot each of these points.

The same concept is used in the function S2CRDV, with the exception that this plot does not involve the calculation of a covariance matrix, eigenvalues, or eigenvectors. Using the same data set as an example, the user would select two of twelve features. Two column vectors of

zeros, each of length 1 x 12, would be generated, and ones would be placed in the columns corresponding to the features selected. (For example, if the first feature selected was Feature 1, a one would be placed in the first column of the first column vector. If the second feature selected was Feature 6, a one would be placed in the sixth column of the second column vector.) These column vectors are then combined into a matrix of length 2 x 12 and this matrix pre-multiplies the data, resulting in a 2 x 847 matrix with each column containing the respective x and y coordinates of the data points. Each of these points would then be plotted.

5.0 Support Files

The data used by S2CRDV and S2EIGV is stored, saved, updated, and resaved in .mat files which are hidden underneath their parent node. The files and their descriptions are given in Table 5.0-1.

.mat File:	Description:	Purpose:
c_fhlist.mat	list of uicontrol checkbox handles	used to identify the user selected features in S2CRDV
crd_data.mat	data transposed	used in the feature projection in S2CRDV
e_ehlist.mat	list of uicontrol checkbox handles	used to identify the eigenvalues selected by the user in S2EIGV
e_fhlist.mat	list of uicontrol checkbox handles	identify which features, if any, were selected by the user to be eliminated from the data calculation in S2EIGV
eig_data.mat	data transposed	used in eigenvector projection in S2EIGV
eig_proj.mat	eigenvalue projection data	projection in S2EIGV
eigvect1.mat eigvect2.mat	two individual eigenvectors	correspond to the selected eigenvalues in S2EIGV
feature1.mat feature2.mat	two features	selected by the user in S2CRDV
fet_proj.mat	feature projection data	used in feature projection in S2CRDV
file_ord.mat	list of the order in which files are accessed	updated by HIDESHOW so that 'hidden' classes are graphed first

Table 5.0-1: .mat files used by S2CRDV and S2EIGV

6.0 Conclusions and Future Development

Some basic structure analysis functions have been implemented in the MATLAB environment, for performing statistical pattern recognition on feature vector data files. Two dimensional plotting tools have been implemented, including the coordinate projection and the eigenvector projection. These tools have been supplemented with functionality such as vector selection, class selection, and zoom capabilities.

6.1 Future Development

FILEIN, S2CRDV, and S2EIGV are only three OLPARS-style functions that have been implemented in MATLAB. One dimensional plots from the functions S1CRDV and S1EIGV, (and also three dimensional plots,) can be implemented in a similar fashion to their two dimensional counterparts. DRAWBNDY, which allows the user to draw boundaries on plots for classification purposes, is another OLPARS-style function that would be useful. The resulting classifiers L1CDRV, L1EIGV, L2CRDV, and L2EIGV could then be used to divide data into subclasses.

OLPARS also had many general functions that would be useful. For example, FILEOUT performs the reverse of FILEIN, taking internal format vector arrays back to ASCII readable form. The OLPARS function ANYTHING, which lists all functions available to the user, is also a useful capability. More 'Help' items, and the ability to remove a node and associated information from a data tree are two more examples of general functions that can also be developed in the future.

References

[1] E. Haehn, D. A. Morris, "OLPARS User Manual," PAR Report #82-21, "OLPARS Software Reference Manual," PAR Report #82-20, "OLPARS Programmer and System Maintenance Manual," PAR Report #82-15, Pattern Analysis and Recognition Corporation, June 1982.

[2] *MATLAB User's Guide*, The Math Works Inc., August 1992.

[3] *MATLAB External Interface Guide*, The Math Works Inc., November 1994.

[4] R. A. Johnson, D. W. Wichern, *Applied Multivariate Statistical Analysis*, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

Appendix A:

Sample Menus, Submenus, and Uicontrols

Figure A-1: STATPACK Main Screen Menu

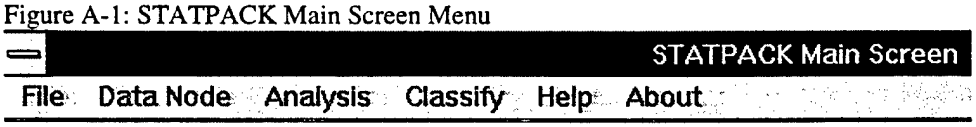


Figure A-2: STATPACK Menus

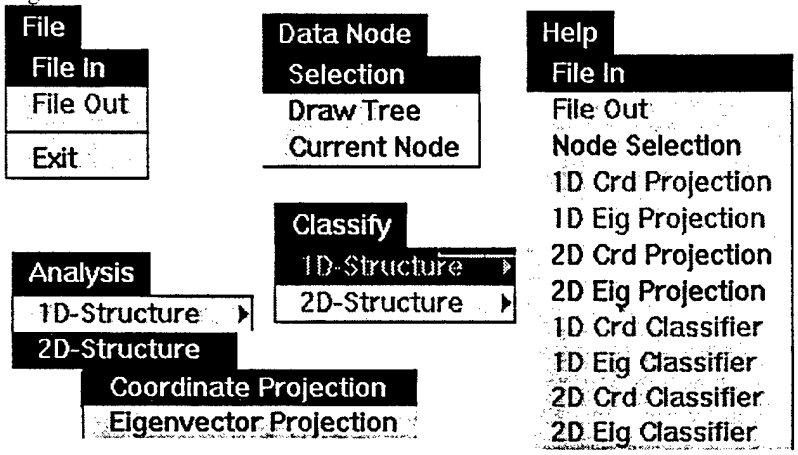


Figure A-3: Node Selection

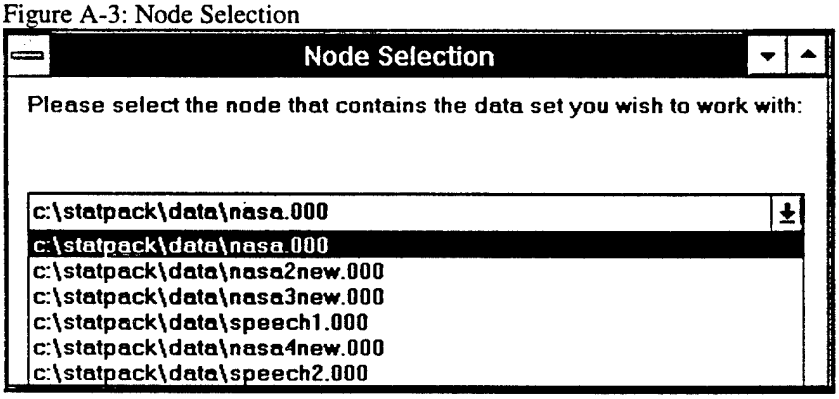


Figure A-4: Draw Tree

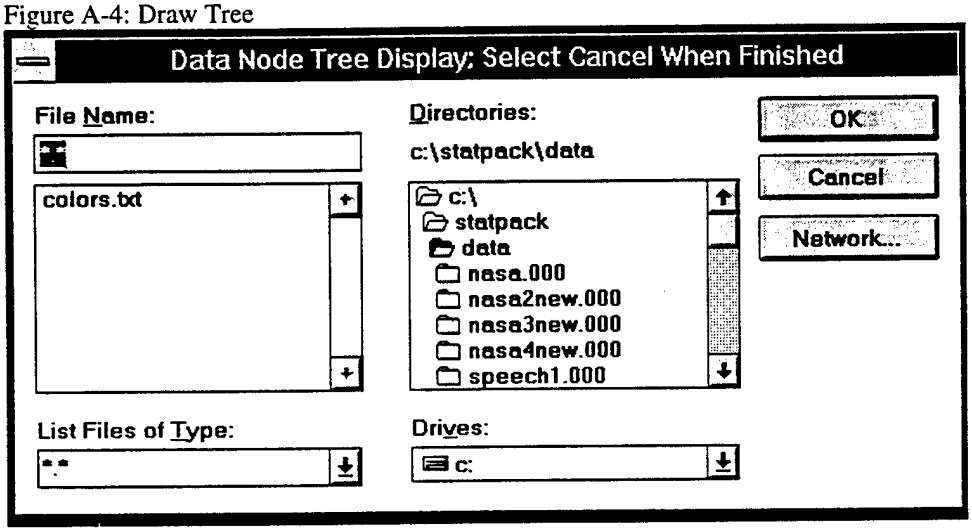


Figure A-5: Current Node

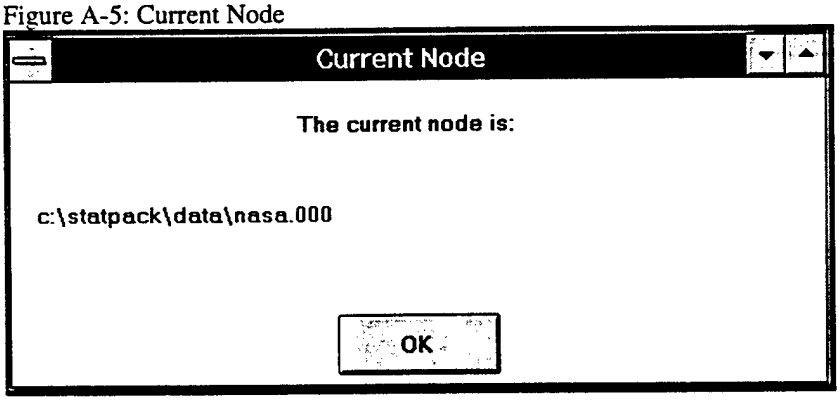


Figure A-6: Feature Selection

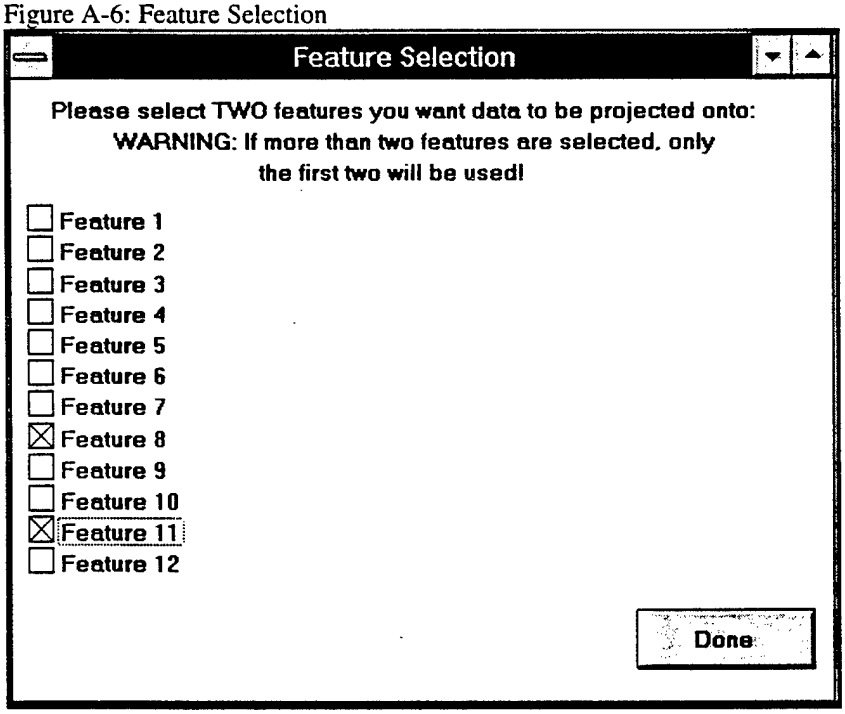


Figure A-7: Feature Projection Plot Menu and Submenus

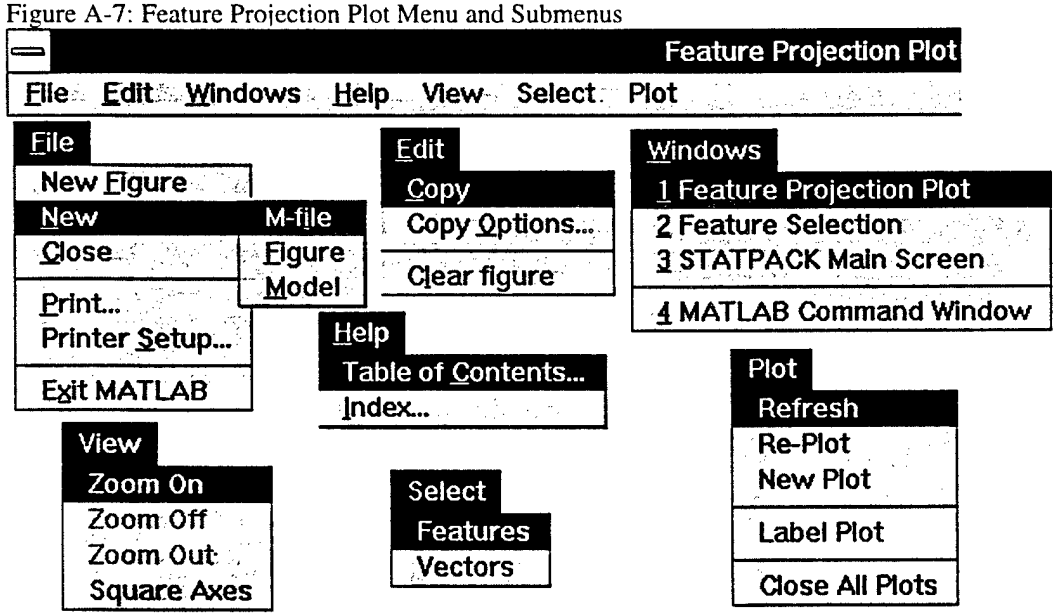


Figure A-8: Feature Elimination Selection

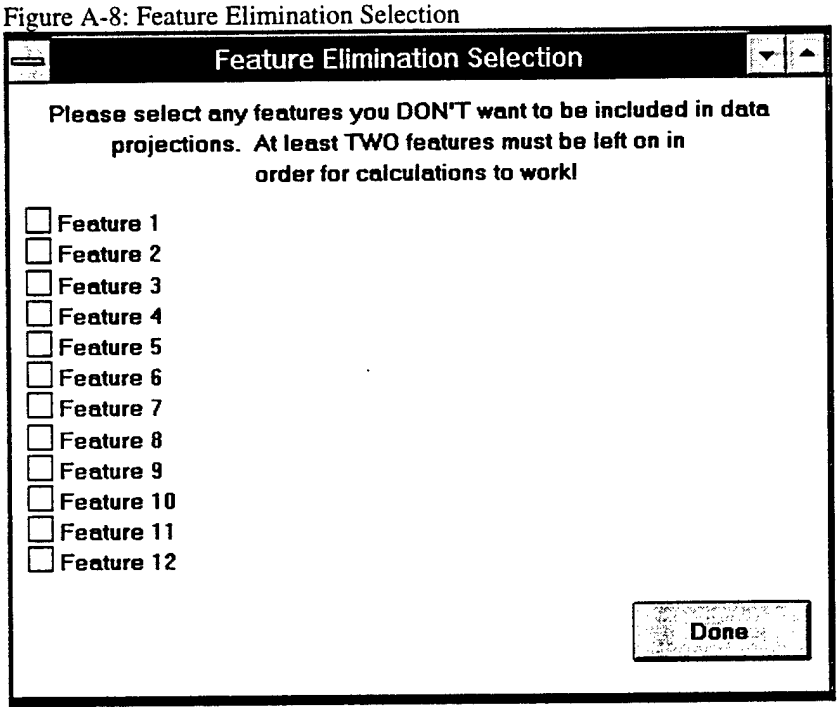


Figure A-9: Eigenvalue Selection

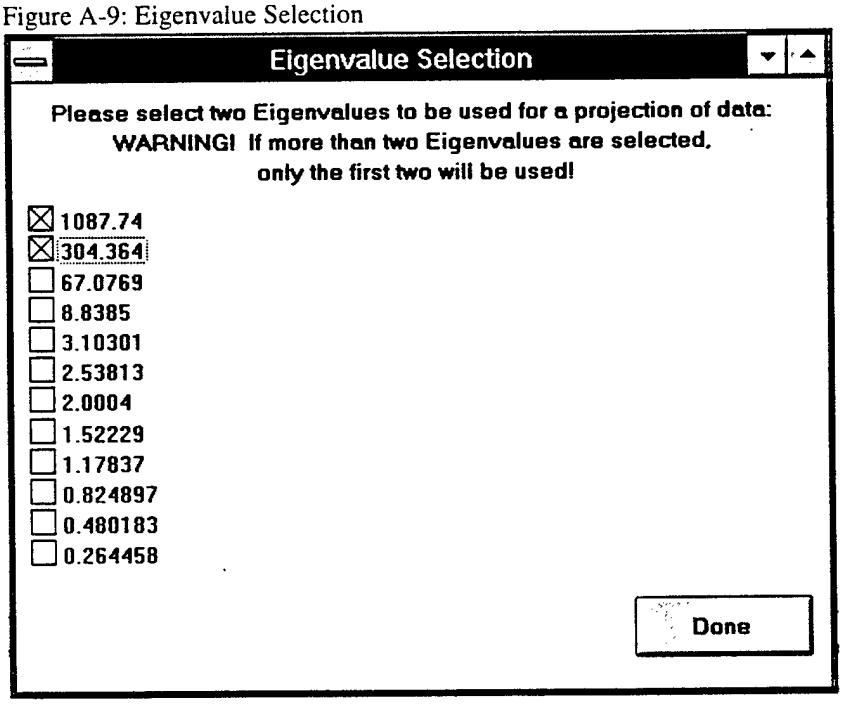


Figure A-10: Eigenvector Projection Plot Menu and Submenus (not found in Figure A-7)

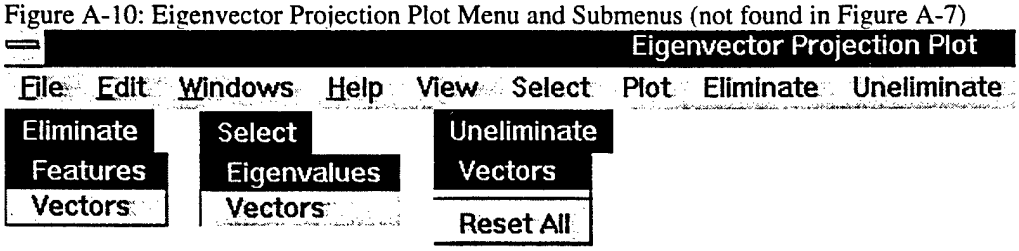


Figure A-11: Select Vector Window

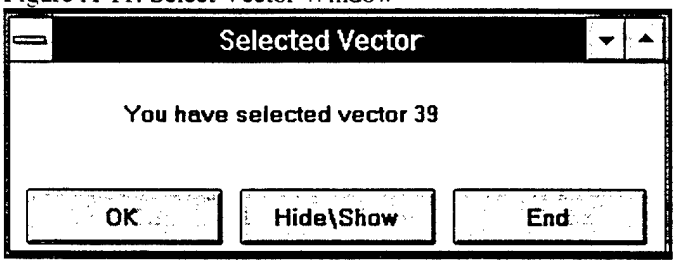


Figure A-12: Hide/Show Selection Window

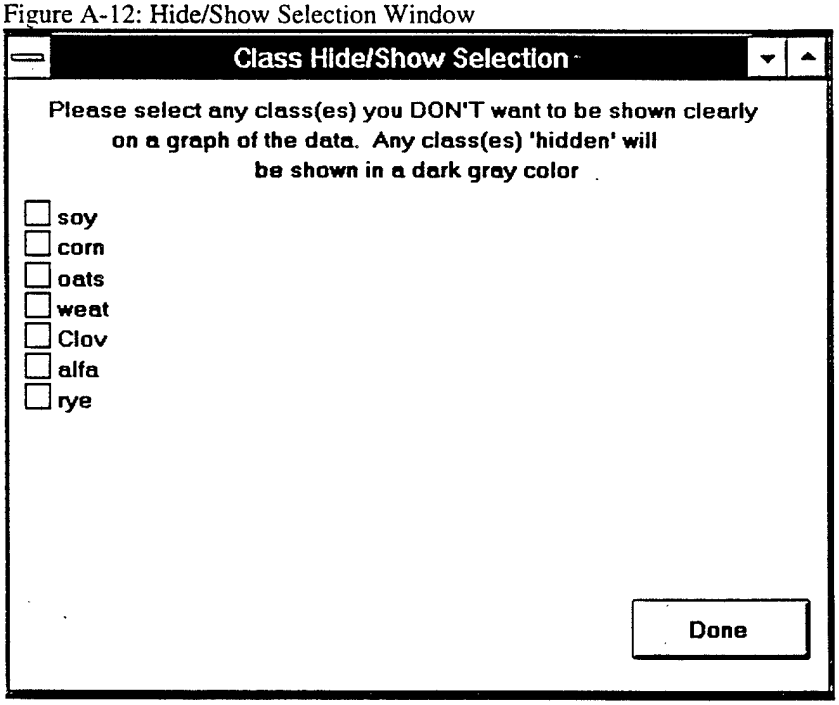
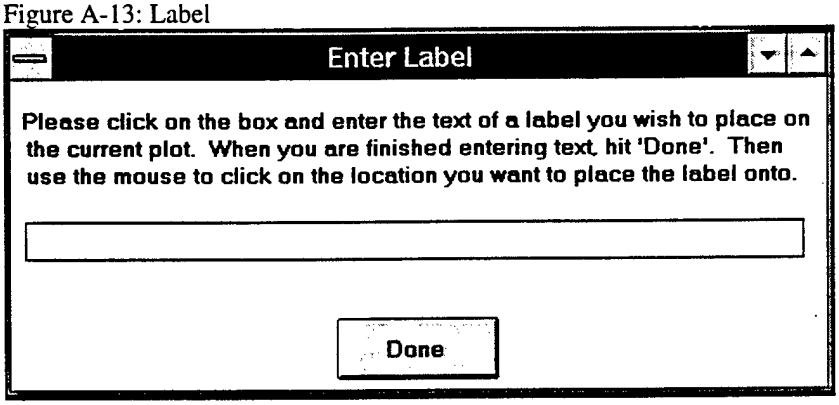


Figure A-13: Label



Appendix B:

List of Matlab Routines for STATPACK

The following is a list of the routines that were written in Matlab code for STATPACK:

statpack.m	sp_movie.m	cruller2.m
filein.m	stdoperr.m	fsize.m
waitbar2.m	stdclerr.m	isup_low.m
isunique.m	filelist.m	clrlist.m
ch_cdrct.m	ch_cdrt2.m	current.m
s2crdv.m	s2crdv_2.m	s2crdv_3.m
featerr.m	sq_axes.m	vidclick.m
near.m	hideshow.m	hdeshw_2.m
label.m	label_2.m	time.m
s2eigv.m	s2eigv_2.m	s2eigv_3.m
s2eigv_4.m	s2eigv_5.m	eig_ferr.m
var_cov.m	eigerr.m	v_off.m
v_off_2.m	v_off_m.m	v_on.m
v_on_2.m	v_on_m.m	reset_v.m
fi_help.m	about.m	

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.