A SOFTWARE PACKAGE FOR SYSTEM IDENTIFICATION

BY
ALI CHAABAN
Bachelor Of Science In
Electrical Engineering
University Of Tulsa
Tulsa, Oklahoma
1987

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1991

A SOFTWARE PACKAGE FOR
SYSTEM IDENTIFICATION

Thesis Approved:

Martin T Hagan
Thesis Advisor

Ronald P Rhoten

Richard L. Cummins

Thomas C. Collins
Dean of the Graduate College

ACKNOWLEDGMENTS

I thank God for the faith, persistence, and determination He gave me throughout my years of study. I also thank Him for surrounding me with gracious people away from home. My deepest love and appreciation goes with no doubt to my family, especially my mom and brothers and sisters (Adnan, Amer, Taha, Inwan, and Maha) who always supported and stood-by me. I dedicate this thesis to the memory of my father who inspired me throughout the years, and to him I say thanks for being there, dad, when I needed you most.

I am deeply indebted to Dr. Martin T. Hagan for his guidance, understanding, and encouragement. Thanks for being around when I needed you, thanks for the knowledge you gave me, and most of all, thanks for being a friend. I am thankful to Dr. R. P. Rhoten, Dr. R. L. Cummins, and Dr. G. M. Scheets for serving on my committee.

I would like to express my gratitude to my friends especially: Nidal Sammur, Mike Barkil, Eddy B. Boujaoude, Dia Abu-Al-Nadi, Adel Khoudir, Adel Abotteen, Ali Termos, and Rod McAbee for their help and friendship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGUERS

CHAPTER I

INTRODUCTION

System identification is the statistical modeling of dynamic systems based on observed data. Modeling such systems or processes can provide a probabilistic structure of the future behavior of the systems. In system identification one first identifies the structure of a model and then estimates its parameters.

Processes such as stock market prices or ozone pollution levels are much more useful if expressed recursively in a finite series of past observations and noise. The mathematical model can be expressed as

$$Z_t = \phi_1 Z_{t-1} + \cdots + \phi_p Z_{t-p} + e_t \tag{1.1}$$

Let us define a backward shift operation, B, such that

$$B^k Z_t = Z_{t-k} \tag{1.2}$$

Then the above AR model may be expressed as

$$\phi(B) Z_t = e_t \tag{1.3}$$

where

$$\phi(B) = 1 - B\phi_1 - \cdots - B^p \phi_p$$

Equation (1.2) is called the AutoRegressive , AR(p), process with order p. The AR model contains p+1 unknown parameters, one of which is the variance of the noise , e_t. An AR process can be stationary or nonstationary. For AR(p) process to be stationary, the impulse response must form a convergent series.

If Z_t is expressed in a finite number of previous e's, then the process would be called a Moving Average, MA(q) with order q, where q is the number of previous e's used to express Z_t

$$Z_t = e_t - \theta_1 e_{t-1} - \cdots - \theta_q e_{t-q} \tag{1.4}$$

If we make use of the B operator (1.4) becomes

$$Z_t = \theta(B) e_t \tag{1.5}$$

where

$$\theta(B) = 1 - B\theta_1 - \cdots - B^q \theta_q$$

A reasonable extension to the AR and MA models is the mixed model of the form

$$Z_t = \phi_1 Z_{t-1} + \cdots + \phi_p Z_{t-p} + e_t - \theta_1 e_{t-1} - \cdots - \theta_q e_{t-q} \tag{1.6}$$

and is called ARMA(p,q) model of order p and q. It is more flexible to use ARMA in modeling an actual time series. Again using the B operator we can simplify the ARMA equation as

$$\phi(B) Z_t = \theta(B) e_t \tag{1.7}$$

Another way of looking at (1.7) is by saying that Z_t and e_t are the output and input of a digital filter with $\phi^{-1}(B)\ \theta(B)$ as the transfer function, where the roots of $\phi(B)$ are the poles and the roots of $\theta(B)$

are the zeros of the filter. This filter is stable and the process is stationary only if the poles are inside the unit circle. Throughout this thesis, unless otherwise indicated, all processes are assumed to be stationary and zero mean.

In process identification we are concerned with estimating the orders p and q as well as the parameters ϕ's and θ's. Figure 1.1 shows a flowchart that can be used as a guide for process identification. This thesis will exhibit several procedures for ARMA identification and will describe an application program to implement these procedures. The application (MacModel) is a user-friendly menu-driven program for Macintosh computer systems. It has been designed for easy expandability. New identification procedures can be added simply.

Chapter II depicts the program flowchart and details its general structure. The examples included in chapter II provide information about start-up and use, and expansion of the program.

Chapter III includes discussion of the simulation of the ARMA process and white noise. It also contains examples on using the List, Plot, Print Text, and Print Graphics menus.

Chapter IV contains the theoretical derivation for the various order estimation methods, as well as the sub-program structure that is related to these methods. Examples are also provided to show the utilization of these methods using the program.

Parameter estimation methods are detailed in chapter V. Related program structure is described along with examples on parameter estimation.

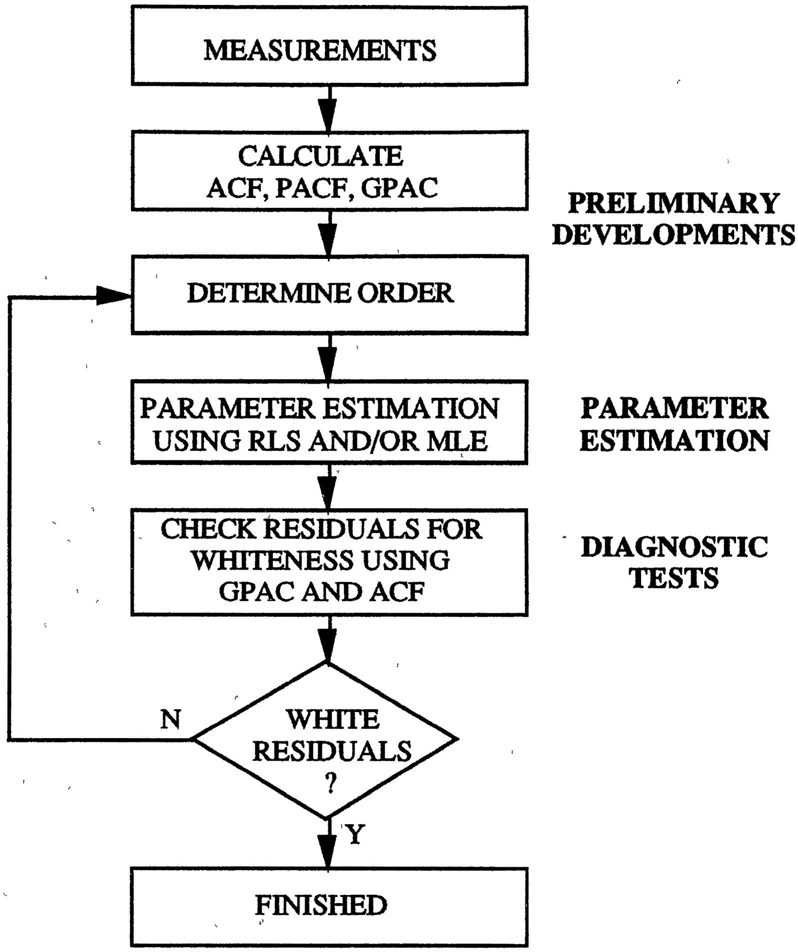

Figure 1.1. A guide for system identification

Chapter VI contains theory and examples of diagnostic testing. This chapter contains methods for checking the validity of the model obtained using the methods described in chapters IV and V. It also contains examples of processes that were overestimated or underestimated. Conclusions and recommendations are included in chapter VII.

CHAPTER II

PROGRAM STRUCTURE

Until April 1981, when Xerox introduced the 8010 "STAR", computers had unfriendly user interfaces where commands, with strict syntax, had to be typed in for every action needed. STAR introduced a bitmapped screen, windows, a mouse driven interface, and icons which were adopted later by others. Today mice, windows and icons are more common and are used by computer experts as well as casual users. This chapter discusses briefly the Macintosh structure, application programs structure, window structure, in addition to the expansion of the application program.

Logically the Macintosh is divided into sections as shown in Figure 2.1 [1]. Applications programs typically call the operating system and User Interface Toolbox routines. The routines available for use are divided according to function, into what are in most cases called "managers" of features that they support. Most managers are in the ROM part of the operating system and User Interface Toolbox.

The operating system is at the lowest level; it does basic tasks such as input and output, memory management, and interrupt handling. The User Interface Toolbox is a level higher above the operating system; it helps implement the standard Macintosh user interface applications. Users sometimes call the operating system

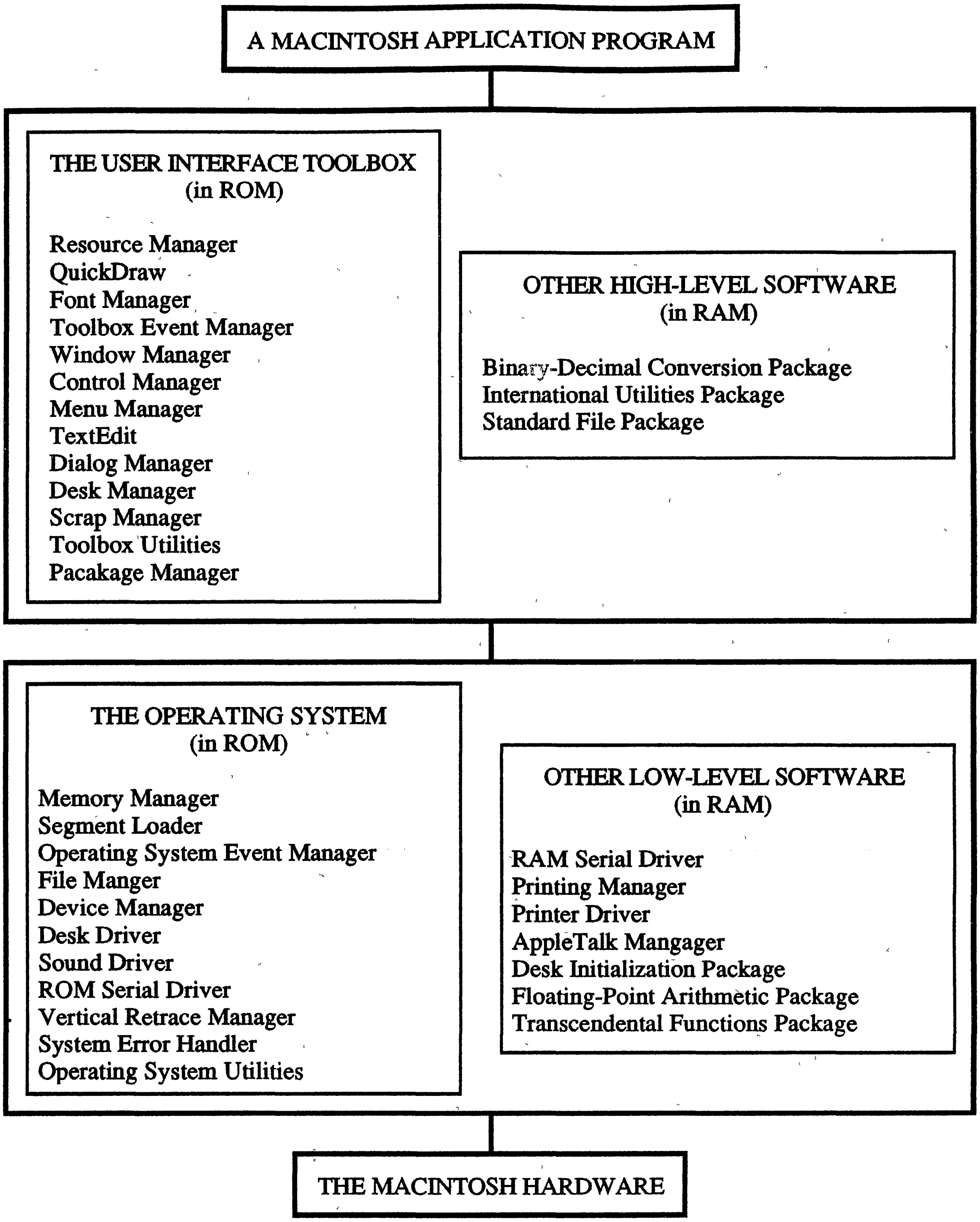

Figure 2.1 Overview of the Macintosh

directly, but for the most part the Toolbox calls the operating system to do low-level operations.

RAM-based software is available as well. In most cases this software performs specialized operations (such as floating-point arithmetic) that are not integral to the user interface but may be useful to some applications. For detailed description of the function of each manager and how managers interact see [1].

The Fortran Toolbox Interface, Toolbx, is a procedure that allows the access of about 500 different Macintosh resources. Toolbx may have different numbers of parameters when accessing different resources, as shown in the following two calls

```
call toolbx(NEWMENU, menuid, menutitle)
call toolbx(SELECTWINDOW, window)
```

the first parameter is reserved and is used to address the resource.

The first step in any Macintosh application is to initialize and allocate the menus (program main.for in Appendix B contains the menu initialization for the application MacModel). To allocate space for the first menu (the menu) the NEWMENU, ADDRESMENU and INSERTMENU functions of toolbx are called. For all of the other menus (such as file, edit,...etc) NEWMENU,APPENDMENU, AND INSERTMENU are called in the listed order.

Windows, which can be used for both text editing and displaying graphics, can be easily created and disposed of as the application runs. The characteristics of a window (such as if the window is highlighted, if it is visible, etc.) are stored in a record proprietary to that window (see Table A.1 for complete list of the window record).

The first part (the first 108 bytes) of any window record is reserved for the window's grafport. The grafport is a complete drawing environment that defines where and how graphic operations will take place in the window (see Table A.2 for grafport declaration and contents).

Changing any variable in the window record directly affects the corresponding window. Window records are also helpful in reading current information about a window such as window size as illustrated below

```
window_width  = portrect(3) - portrect(1)
window_height = portrect(4) - portrect(2)
```

with the portrect variable as defined in Table A.2. Or to check if the window is visible simply check if the variable "visible" is true (see Table A.1 for "visible").

The heart of every application program is its main event loop, which repeatedly calls the toolbox Event Manager to get events and then responds to them as appropriate. The most common event is a press of a mouse button; various actions are performed depending on where the mouse was pressed. Every event is represented internally by an event record containing all pertinent information about that event. The event record includes information such as the type of event and the mouse location (see Table A.3 for event record structure). The following example shows how to get the next event

```
toolbx(GETNEXTEVENT, eventmask, eventrecord)
```

eventmask = -1 to check all events (to filter in only certain events, check the corresponding value for eventmask in A.4). The programmer checks the "what" variable of the eventrecord to know what type of event had occurred and the "where" variable for the location of the mouse.

After detecting an event one should call the FINDWINDOW function

```
mouseloc=toolbx(FINDWINDOW,where,window)
```

to find out where the event occurred. See Table A.5 in Appendix A for interpretation of "mouseloc". The "window" parameter in the FINDWINDOW call is a returned pointer to the window (if any) that the event occurred in. The "where" parameter, obtained above from GETNEXTEVENT, is the location of the mouse at time of the event. Action should be taken according to which event occurred. For example, if the mouse was in a window, that window should be highlighted; or if the mouse was in the grow region of a window, action should be taking to resize that window.

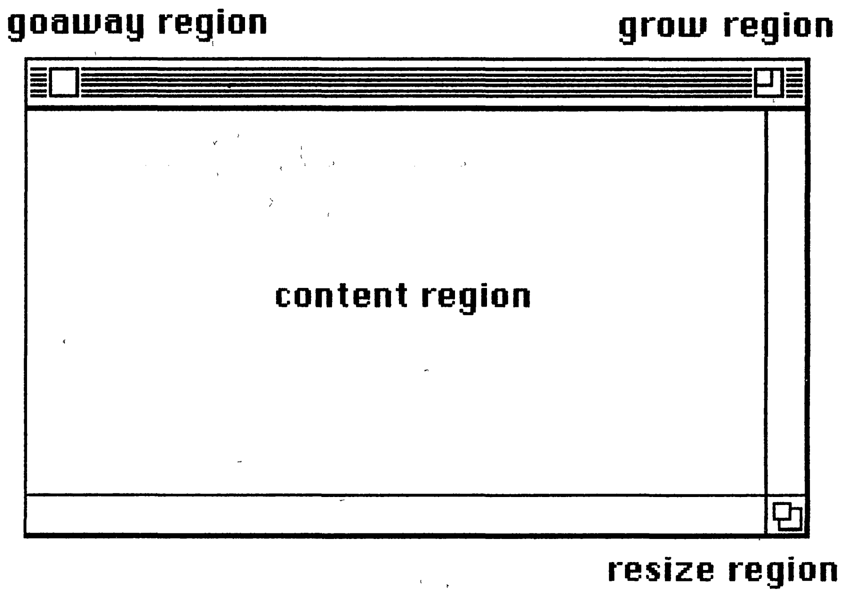

Figure 2.2 Window characteristics that can be accessed from a window record.

If mouseloc = 1 (i.e. mouse in menu bar) a user written subroutine should be called to display the pull-down menu and to select an item of the pull-down menu. In MacModel, menus.for (in Appendix B) is written for that purpose. Once a mouse-down event is detected in the menu bar area, the Menus subroutine takes care of displaying the pull-down menus and all the subsequent events.

The menus subroutine should first call the MENUSELECT function (or the MENUKEY function if command-key is hit) of the toolbx. MENUSELECT keeps control until the mouse button is released, tracking the mouse, pulling down menus as needed, and highlighting the menu item under the mouse cursor. Once the mouse button is released over a highlighted menu item, MENUSELECT returns the menuid (e.g. of File or Edit). With menuid and menu item at hand, the application should execute the desired routine or codes.

2.1 Program Structure and Flowchart

MacModel consists of a main routine and several subroutines. Most of the subroutines are executed in correspondence to a menu selection from the menu bar (see Figure 2.3). Each of the main-menu items has sub-menu items associated with it. Following is a brief description of each menu's contents:

Figure 2.3 MacModel menu bar

: Contains desk accessories. The list of accessories is expanded or reduced according to what is available, in the system folder. See Figure 2.4 for an menu.

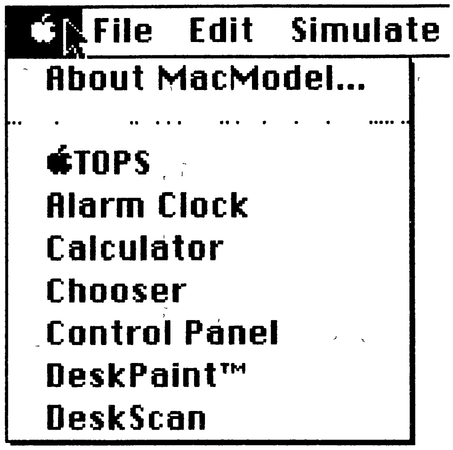

Figure 2.4 An menu

<u>File</u>: Contains List, Plot, Print Text, Print Graphics,Transfer and Quit. See Figure 2.5.

List: Lists a file to screen.

Plot: Plots a data file to screen.

Print Text: Sends a file (text) to printer.

Print Graphics: Sends the last graph or grayscale map to the printer (inactive initially until a graphic operation takes place).

Transfer: Allows the user to exit MacModel permanently to another application.

Quit: Permanently leaves MacModel and returns to Finder.

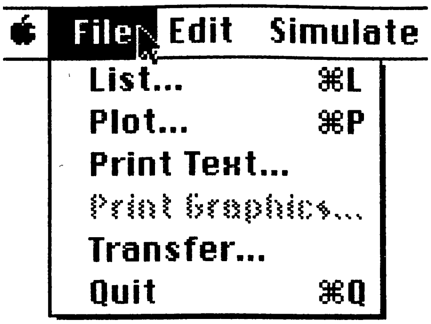

Figure 2.5 The File menu

Edit: Contains commands to manipulate text. See Figure 2.6.

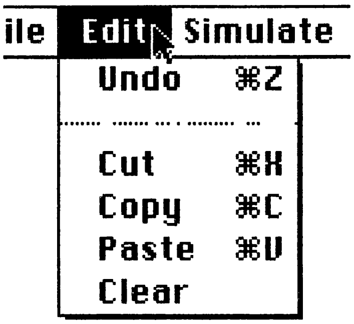

Figure 2.6 The Edit menu

Simulate: Contains commands to generate noise, gaussian or uniform, and to simulate ARMA processes. See Figure 2.7.

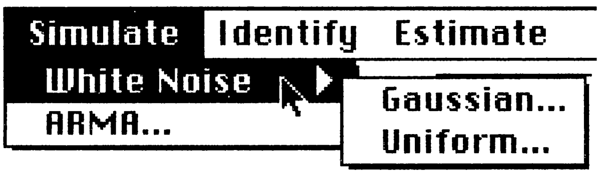

Figure 2.7 The Simulate menu and its submenu

Identify: Contains commands to generate autocorrelation, crosscorrelation and partial autocorrelation sequences as well as commands to generate generalized partial autocorrelation, S-array and R-array. See Figure 2.8.

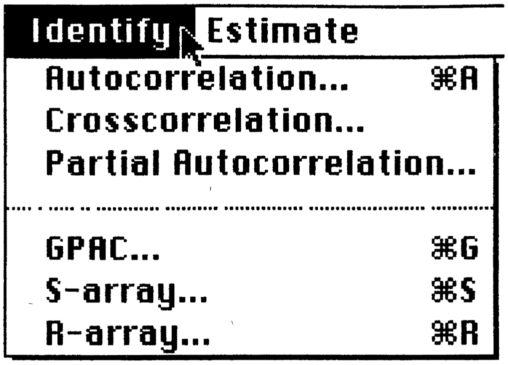

Figure 2.8 The Identify menu

<u>Estimate</u>: Contains commands to estimate parameters (using the Maximum Likelihood algorithm) of an ARMA process with known order. See Figure 2.9.

Figure 2.9 The Estimate menu

The main routine is basically divided into four sections which have been discussed previously:

1- Type definitions
2- Menu initialization and allocation
3- Windows creation
4- Main event processing loop.

The flowchart of the main program is shown in Figure 2.10

2.2 Start-Up and Use

To start up MacModel just select it using the mouse (click twice on the application icon). Make sure that the Macfortran run-time Libraries f77.rl and m81.rl are in the system folder or in the same folder with the application. MacModel is a standard Macintosh application, i.e. it has the same user interface any Macintosh application has.

2.3 Expansion of Program

There are two kinds of expansion possible; adding a new menu (pull-down menu), or adding a menu item in an already existing menu.

2.3.1 Add a New Menu

Allocate the new menu (in the main program) using the NEWMENU function of the toolbx, then append the menu items to that menu using APPENDMENU. Insert the menu anywhere you desire after the Apple menu. Now in the menus subroutine declare the needed variables (the best way to do this is to follow one of the other menu declarations), then insert in the Case statement the menu name. Code under the new menu a Case statement that has the menu items as the 'Case selectors' (e.g. the Edit menu would have Cut, Copy, Paste... as case selectors). The programmer is responsible for

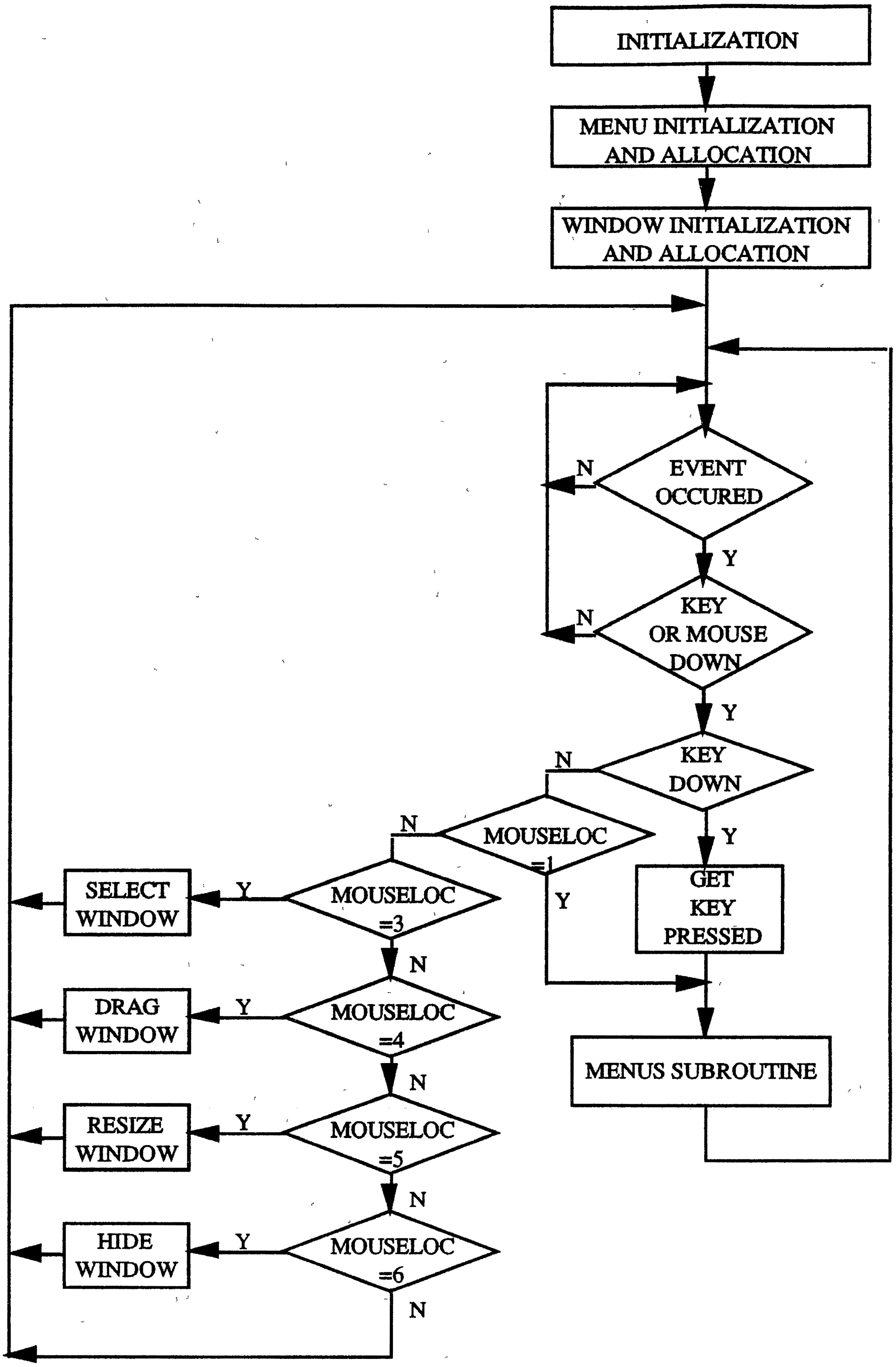

Figure 2.10 The main program flowchart

coding statements, or perhaps calling a subroutine to perform whatever is needed, inside the Case statement.

Let us assume that we want to add a new menu called NewMenu on the menu bar between the File menu and the Edit menu. Also assume that NewMenu has two menu items; Item1 and Item2. The first step is to declare the NewMenu right after the File menu in the main program as shown below

```
menuhandel(8) = toolbx(NEWMENU, 36, str255("NEWMENU"))
call toolbx(APPENDMENU, menuhandel(8),
                                   str255("Item1/1;Item2"))
call toolbx(INSERTMENU, menuhandel(8),0)
```

the '/1' after Item1 in the second call assigns the keys ⌘1 to this item. The second step is to declare NewMenu in the menu.for subroutine and assign it to 36 (or what ever number you used in the first call above. This number should be unique in the NEWMENU function calls). Then declare Item1 and Item2 and assign them to 1 and 2 as shown

```
parameter(Item1=1,Item2=2).
```

Now go to the event loop in the menu.for subroutine, and insert between the File and Edit Case statements the following:

```
case(NewMenu )
     case(Item1)
          call do_Item1()
     case(Item2)
          call do_Item2()
end select
```

2.3.2 Add a New Item to Existing Menu

Make the appropriate declaration, in the main program, and insert the new menu name where needed (e.g. between List and Plot in the File menu). Then in the menus subroutine (after the appropriate declarations) insert an additional Case selector in the Case statement that follows the menu name. Again code the appropriate statements after the new menu item selector of the Case statement. Let us say we want to add a menu item called Type between List and Plot in the File menu (see Figure 2.4). First make the following change in the main program

```
call toolbx(APPENDMENU, menuhandel(2),str255("List/L;
Type;Plot/P;Print Text;Print Graphics;Transfer;Quit/Q"))
```

Then in the menu.for subroutine change the declaration from

```
parameter (List=1,Plot=2,Print=3,Printg=4,
                                Transfer=5,Quit=6)
```

to

```
parameter (List=1,Type=2,Plot=3,Print=4,Printg=5,
                                Transfer=6,Quit=7)
```

Finally change the Case statement to

```
case(List)
     call do_list()
case(Type)
     call do_type()
case(Plot)
     call do_plot()
```

CHAPTER III

SIMULATION

Simulation is an important part of any system identification package. This chapter introduces the concept of simulation, in addition to instructions on plotting graphics and listing text to the screen or printer. The first menu, in MacModel, after the three standard menus (, File, and Edit) is the Simulate menu. Under this menu, there are commands to simulate noise, Gaussian or Uniform, ARMA, AR, and MA processes. List, Plot and print commands are in the File menu.

3.1 Simulation of Processes

The most basic use of a system description is to simulate the systems response to various input sequences, e.g. an input sequence e_t chosen by the user is applied to (1.6). Subroutine doarma.for, listed in Appendix B, does the ARMA, AR or MA processes simulation for a maximum AR order of 12 and a maximum MA order of 12. The input of the system is usually a white noise process. The noise is realized via a pseudo random number generator which is generated by subroutine downit.for that is also listed in Appendix B.
A commonly applied procedure that can be regarded as a test of the model validity, is to simulate the system, if feasible, with the actual

input and compare the simulated output with the actual output. Experimenting with models, is perhaps the most common use for simulation especially if the system in question is difficult to experiment on directly, such as aircraft or nuclear power stations.

3.2 Simulation Examples Using MacModel

The Simulate menu contains menu entries for generating noise and ARMA processes.

3.2.1 Noise Simulation

Choose "White Noise" from the "Simulate" menu; then select the desired noise distribution from the White Noise submenu. See Figure 3.1.

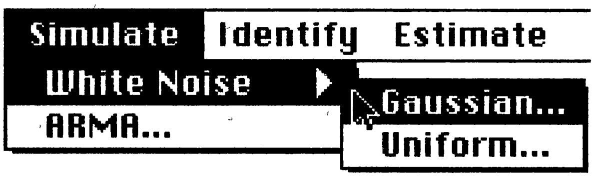

Figure 3.1 Selecting Gaussian noise from the Simulate menu

The user is next presented with the "Noise Parameters" dialog box as depicted in Figure 3.2. The shown values for the variance, mean, seed, and the number of points to be generated are set to the defaults and can be changed. The variance is restricted to be positive, and the maximum number of points is 4000. Exceeding the limit on the number of points causes a dialog box to be displayed

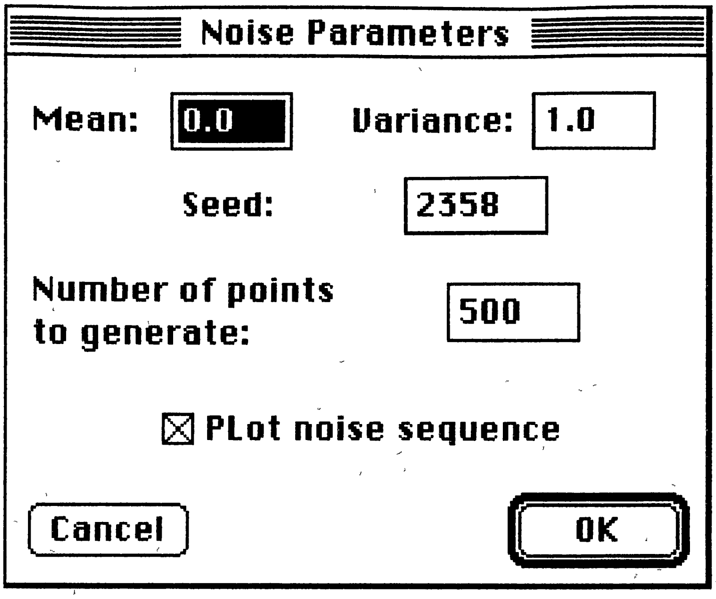

Figure 3.2 Noise Parameters dialog box

informing the user of the limit and setting the number of points to that limit. The "Plot noise sequence" check box in the Noise Parameters dialog box, if it remains selected, causes the subroutine dowhit.for to call subroutine plot.for (listed in Appendix B) which in turn plots the noise sequence to the screen as depicted in Figure 3.3. If the OK button is hit, the user is then prompted for saving the noise sequence to a file. If a cancel is hit at this point the sequence will not be saved and the user is given the option to plot it.

Pressing and dragging the lower-right corner of the window titled "Plot" (not the "TTY" window) causes the last graph to be resized and repainted to the screen. If a plot of another process is desired, choose plot from the File menu then select the process file from the File-Input Interface dialog box (see Figure 3.4). The user is next presented with the "Plot Parameters" dialog box and a choice to

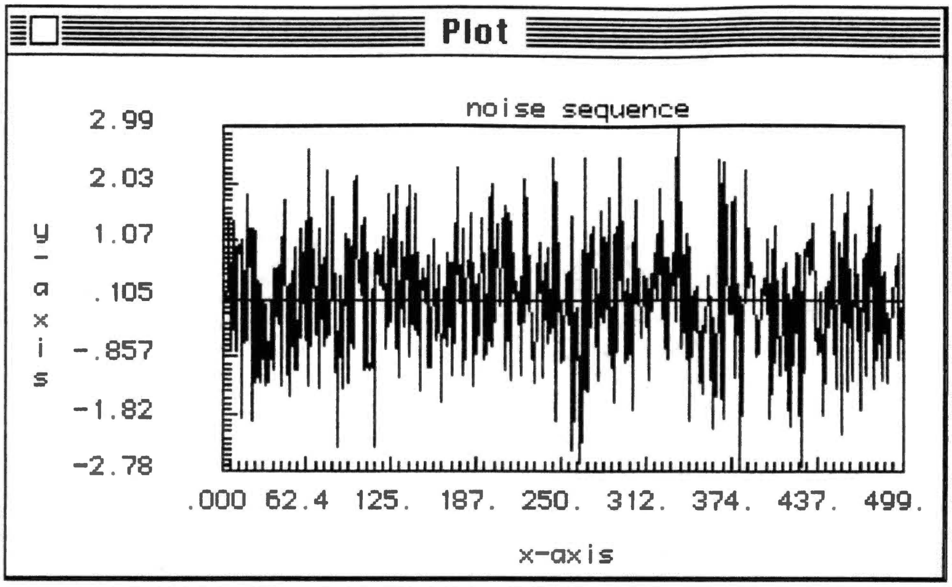

Figure 3.3 A Noise sequence

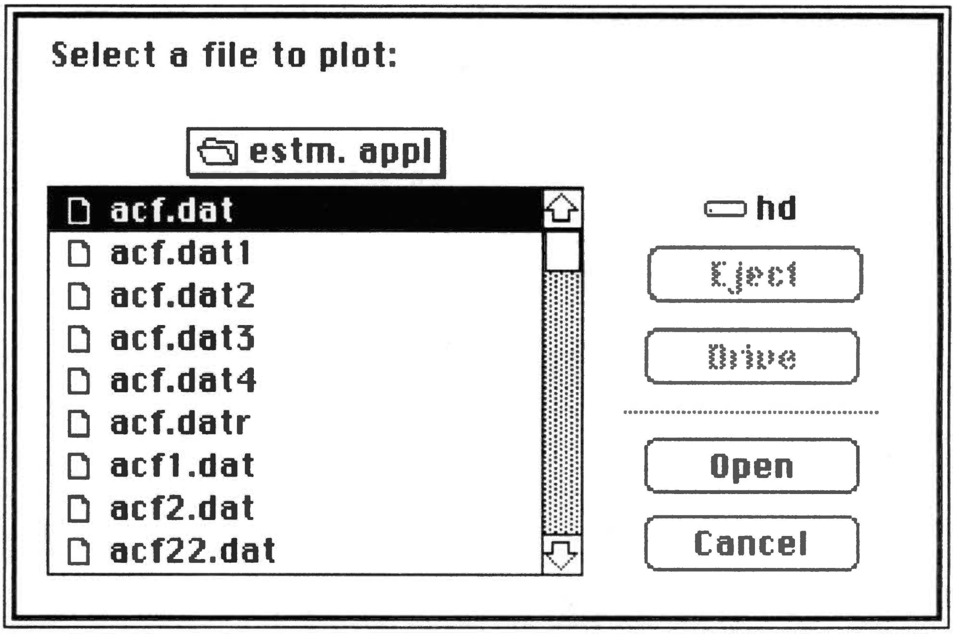

Figure 3.4 The File-Input Interface dialog box

plot any part of the process (see Figure 3.5). Clicking on the OK causes the data points to be plotted to the screen. If a hard copy is

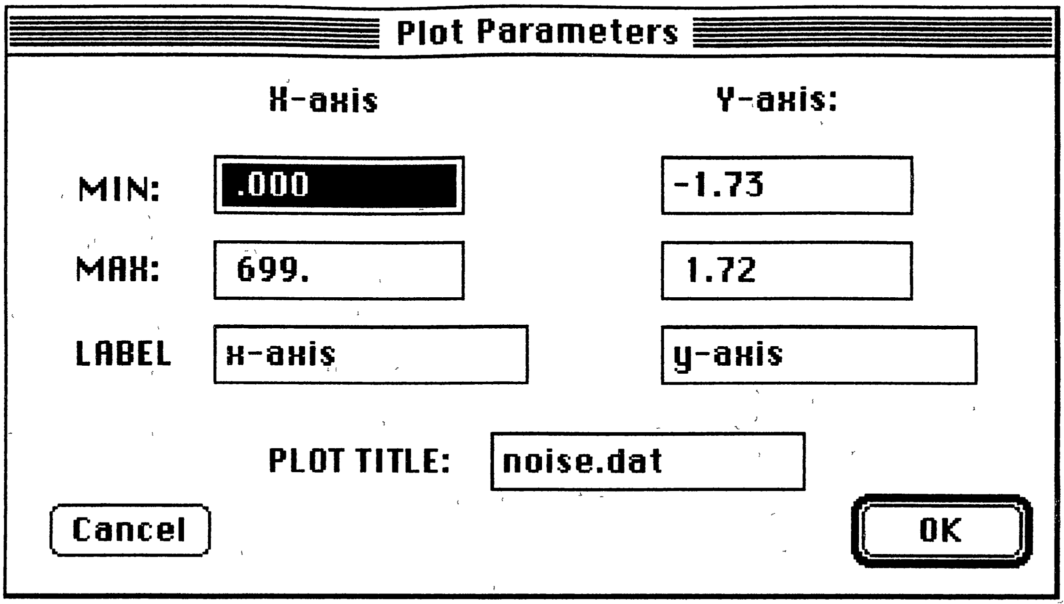

Figure 3.5 Plot Parameters dialog box

desired, choose Print Graphics from the File menu. This should print out the last graph displayed to the Plot window (even if the plot or graph is not currently shown in the window).

A list of the data points to the "TTY" window can be achieved by choosing List from the File menu, then selecting the appropriate data file from the File-Input Interface dialog box (similar to that shown in Figure 3.4). A printout of the data file is accomplished by choosing Print Text from the File menu. The data files generated by MacModel can be edited by any text editor. MacModel can use any data file for plotting as long as it has (x,y) format.

3.2.2 ARMA Processes Simulation

ARMA process simulation is carried out in the same manner as the Noise Simulation. Upon choosing "ARMA" from the "Simulate" menu, the "ARMA parameters" dialog box, containing the ARMA process order, p and q, and a plot check box, is displayed (see Figure 3.6).

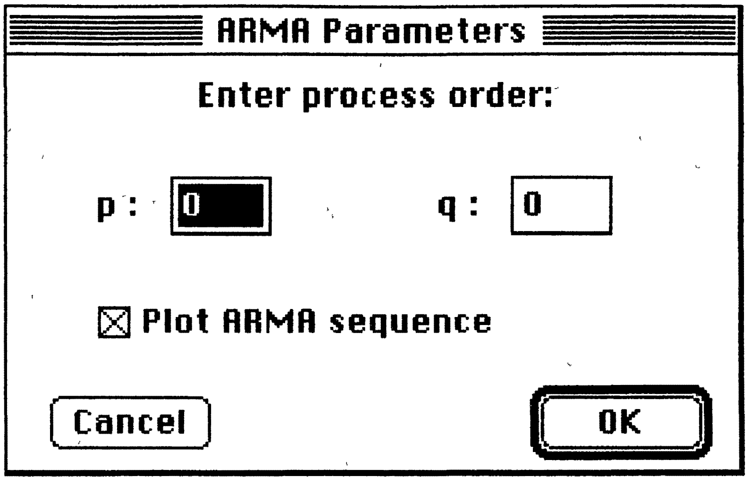

Figure 3.6 ARMA Parameters dialog box

Setting p to 0 simulates MA processes, and setting q to 0 simulates AR processes. The "Plot ARMA sequence" check button is dealt with in the same manner as the above mentioned "Plot Noise Sequence" button. Plotting, replotting, listing, printing, and editing the ARMA sequence are also possible, as they were with the noise sequence.

The user is responsible for checking the stability of the process. Unstable processes could cause numeric overflow when used as input to the ACF.

CHAPTER IV

MODEL IDENTIFICATION

Chapter I introduced the AR, MA, and ARMA models. Here we shall introduce the Autocorrelation Function (ACF), Partial Autocorrelation Function (PACF), Generalized Partial Autocorrelation function (GPAC) and S and R arrays, which are the tools for preliminary system identification. But first, let us touch on some material that deals with probability and statistical analysis which are crucial to preliminary system identification.

4.1 Preliminary Definitions

Expectations (mean and variance):

The mean is defined

$$\mu_z = E[Z] = \int_{-\infty}^{\infty} z\, f_z(z)\, dz \tag{4.1}$$

where f_Z is the probability density function of Z, and E[ ] is the expectation.

In practice, and in the absence of any probability density functions, we use the sample mean as shown in (4.2)

$$\hat{\mu}_z = \frac{1}{N}\sum_{t=1}^{N} Z_t \qquad (4.2)$$

Equation (4.2) is a good estimator if enough independent samples were taken.

The variance is defined

$$\begin{aligned}\sigma_z^2 &= E[(Z_t - \mu_z)^2] \\ &= E[Z_t^2] - \mu_z^2\end{aligned} \qquad (4.3)$$

The sample variance can be expressed as

$$\hat{\sigma}_z^2 = \frac{1}{N-1}\sum_{t=1}^{N}(Z_t - \hat{\mu}_z)^2 \qquad (4.4)$$

A process with constant statistics is said to be a stationary process. This is an important class of processes, and the preliminary system identification discussed in this chapter assumes stationarity. One way of testing for stationarity is by checking if the poles of the process are inside the unit circle. A nonstationary process can also be modeled by assuming some difference of the process to be stationary. Such models are called ARIMA, Autoregressive Integrated Moving Average, and will not be discussed here.

4.2 Model Identification Techniques

4.2.1 Correlation

The autocovariance of a process Z is defined

$$\begin{aligned}\gamma_z(k) &= \mathrm{cov}[Z_t, Z_{t+k}] \\ &= E[(Z_t - \mu_z)(Z_{t-k} - \mu_z)]\end{aligned} \tag{4.5}$$

and the autocorrelation as the normalized autocovariance

$$\rho_z(k) = \frac{\gamma_z(k)}{\gamma_z(0)} \tag{4.6}$$

Notice that the autocovariance and autocorrelation functions are functions of the time difference, k, not the actual time. This is true only for stationary processes.

Some properties of $\rho_Z(k)$ which also apply to $\gamma_Z(k)$

$$\begin{aligned}&1-\ \rho_z(k) = \rho_z(-k) \\ &2-|\rho_z(k)| \leq \rho_z(0)\end{aligned}$$

Also we have

$$\begin{aligned}\rho_z(0) &= 1 \\ \gamma_z(0) &= \sigma_z^2\end{aligned}$$

The sample autocovariance is defined as

$$\gamma_z(k) = \frac{1}{N-k}\sum_{t=1}^{N-k}(Z_t - \mu_z)(Z_{t+k} - \mu_z) \tag{4.7}$$

N is the total number of observations.

The crosscovariance is defined

$$\gamma_{zx}(k) = E[(Z_t - \mu_z)(X_{t+k} - \mu_x)] \tag{4.8}$$

where X_t and Z_t are assumed to be stationary processes, some of γ_{ZX} properties are

$$1-\ \gamma_{zx}(0) = \gamma_{xz}(0)$$
$$2-\ \gamma_{zx}(k) = \gamma_{xz}(-k)$$
$$3-\ |\gamma_{zx}(k)| \le \sqrt{\gamma_z(0)\gamma_x(0)}$$

If X_t and Z_t are two independent processes, then

$$\gamma_{zx}(k) = 0$$

The sample crosscovariance is defined as

$$\hat{\gamma}_{zx}(k) = \frac{1}{N-k}\sum_{t=1}^{N-k}(Z_t - \mu_z)(Z_{t+k} - \mu_z) \tag{4.9}$$

The sample autocorrelation and crosscorrelation algorithms are implemented in subroutine corlat.for which is listed in appendix B.

Consider the moving average process of (1.4) (where e_t is a white gaussian noise, i.e. $\rho_e(k) = 0$ for all $k \neq 0$ and $\rho_e(k) = \sigma^2{}_e$ for $k=0$)

$$Z_t = e_t - \theta_1 e_{t-1} - \cdots - \theta_q e_{t-q}$$

Also consider a shifted version of (1.4)

$$Z_{t-k} = e_{t-k} - \theta_1 e_{t-1-k} - \cdots - \theta_q e_{t-q-k}$$

Now take the expectation of $Z_t Z_{t-k}$. We get (assuming $\mu_e=0$)

$$\gamma_z(k) = E[Z_t Z_{t-k}] = E[(e_t - \theta_1 e_{t-1} - \cdots - \theta_q e_{t-q})(e_{t-k} - \theta_1 e_{t-1-k} - \cdots - \theta_q e_{t-q-k})]$$

$$\gamma_z(k) = \begin{cases} \sigma_e^2(-\theta_k + \theta_1\theta_{k+1} + \cdots + \theta_{q-k}\theta_q) & k = 1,2,\cdots,q \\ \sigma_e^2(1+\theta_1^2+\cdots+\theta_q^2) & k = 0 \end{cases} \qquad (4.10)$$

Keeping in mind that e_t is white gaussian noise with zero mean, it can be easily shown that for an MA(q) process

$$\gamma_z(k) = 0 \quad k > q$$
$$\rho_z(k) = 0 \quad k > q \qquad (4.11)$$

Equation (4.11) is very important because it indicates that any process with $\rho_Z(k)$, or $\gamma_Z(k)$, zero after certain time lag k = q, can be modeled as an MA(q) process. Figure (4.1) shows example of an autocorrelation function of an MA(2) process.

Now let us examine the autoregressive process (1.1)

$$Z_t = \phi_1 Z_{t-1} + \cdots + \phi_p Z_{t-p} + e_t$$

A significant property, shown below, of the autocorrelation function

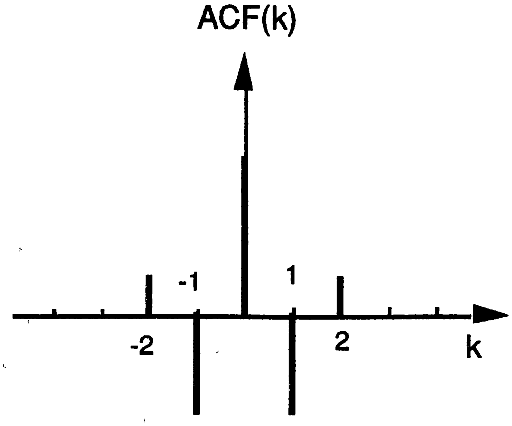

Figure 4.1 ACF of an MA(2) process

of the AR process is that it satisfies the same difference equation as does the AR process. By multiplying both sides of (1.1) by Z_{t-k} and taking the expectation we get

$$\gamma_z(k) = \phi_1\gamma_z(k-1)+\cdots+\phi_p\gamma_z(k-p)+\gamma_{ze}(k)$$

but

$$\gamma_{ze}(k) = 0 \qquad k > 0$$

because Z_t is dependent on present value of e_t only, and because e_t is white. The above equations lead to

$$\begin{aligned} &\gamma_z(k)-\phi_1\gamma_z(k-1)-\cdots-\phi_p\gamma_z(k-p)=0 \qquad k>0 \\ &\rho_z(k)-\phi_1\rho_z(k-1)-\cdots-\phi_p\rho_z(k-p)=0 \qquad k>0 \end{aligned} \tag{4.12}$$

Equations (4.12) for k= 1, 2, 3, ... are called the Yule-Walker equations and they shall come up later in our definitions and derivations.

4.2.2 Partial Autocorrelation Function (PACF)

Using the Yule-Walker equations, the PACF sequence can be defined as

$$\begin{bmatrix} \rho_z(0) & \rho_z(1) & \cdots & \rho_z(k-1) \\ \rho_z(1) & \rho_z(0) & \cdots & \rho_z(k-2) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_z(k-1) & \rho_z(k-2) & \cdots & \rho_z(0) \end{bmatrix} \begin{bmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{bmatrix} = \begin{bmatrix} \rho_z(1) \\ \rho_z(2) \\ \vdots \\ \rho_z(k) \end{bmatrix} \tag{4.13}$$

Let ϕ_{kk} be the PACF by definition. Solving for the PACF using the above Yule-Walker equations and Cramer's rule we get

$$\phi_{11} = \frac{\rho_z(1)}{\rho_z(0)}$$

$$\phi_{22} = \frac{\begin{vmatrix} \rho_z(0) & \rho_z(1) \\ \rho_z(1) & \rho_z(2) \end{vmatrix}}{\begin{vmatrix} \rho_z(0) & \rho_z(1) \\ \rho_z(1) & \rho_z(0) \end{vmatrix}}$$

In general ϕ_{kk} is nonzero for $k \leq p$ and zero otherwise. Also $\phi_{pp} = \phi_p$, the last parameter in the AR process. Let us look at an example identifying an AR(1) process using the PACF

Example 4.2.2.1

$$Z_t = 0.5\, Z_{t-1} + e_t \qquad \sigma_e^2 = 1$$

The ACF sequence for this process is

$\rho_k(k) = 1, .5, .25, .125, \ldots \qquad k = 0, 1, 2, 3, \ldots$

We know from (4.12) that the autocorrelation satisfies the above difference equation. So we get

$$\rho_t(k) = 0.5\ \rho_t(k-1) \qquad k > 0$$

Now let us calculate the PACF sequence

$$\phi_{11} = \frac{.5}{1}$$

$$\phi_{22} = \frac{\begin{vmatrix} 1 & .5 \\ .5 & .25 \end{vmatrix}}{\begin{vmatrix} 1 & .5 \\ .5 & 1 \end{vmatrix}} = \frac{0}{.75} = 0$$

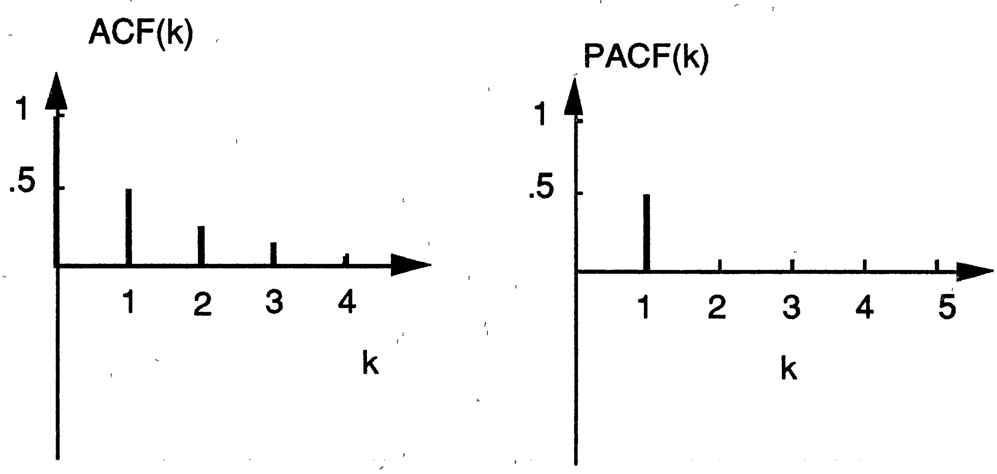

Figure 4.2 ACF and PACF of the AR(1) process in Example 4.2.2.1

This indicates that, theoretically at least, it suffices to examine the PACF to identify an AR process. The PACF function can be obtained recursively using Levinson's algorithm. Subroutine pacf.for contains the code for the algorithm and is listed in appendix B.

4.2.3 Generalized Partial Autocorrelation Function (GPAC)

The previous two sections showed that it suffices to examine the ACF to identify an MA(q) process. Whereas for AR(p) processes it was enough to examine the PACF. For ARMA identification the GPAC is utilized. The GPAC is an extension of the PACF, in fact the first row of the GPAC is the PACF sequence. Using the actual autocorrelation sequence in the GPAC calculations, one can uniquely determine the order of a stationary process. The Yule-Walker equations are also used here to produce the GPAC array.

By multiplying both sides of (1.6) by Z_{t-k} and taking the expectation we get

$$\gamma_z(k) = \phi_1\gamma_z(k-1)+\cdots+\phi_p\gamma_z(k-p)+\gamma_{ze}(k)-\theta_1\gamma_{ze}(k-1)-\cdots-\theta_q\gamma_{ze}(k-q)$$

It can be easily shown that (discussions here apply to the autocovariance as well as to the autocorrelation)

$$\rho_z(k)-\phi_1\rho_z(k-1)-\cdots-\phi_p\rho_z(k-p)=0 \qquad k>q \qquad (4.14)$$

This is an important equation because it shows that the ACF sequence, for k>q, satisfies the difference equation of the AR part of the process (i.e. only the $(q+1)^{st}$ and following of the Yule-Walker equations are satisfied).

Consider the following definitions

$$B(k,j) = \begin{bmatrix} \rho_z(j) & \rho_z(j-1) & \cdots & \rho_z(j-k+1) \\ \rho_z(j+1) & \rho_z(j) & \cdots & \rho_z(j-k+2) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_z(j+k-1) & \rho_z(j+k-2) & \cdots & \rho_z(j) \end{bmatrix}$$

$$\phi(k,j) = \begin{bmatrix} \phi^j_{k1} \\ \phi^j_{k2} \\ \vdots \\ \phi^j_{kk} \end{bmatrix} \qquad \rho(k,j) = \begin{bmatrix} \rho_z(j+1) \\ \rho_z(j+2) \\ \vdots \\ \rho_z(j+k) \end{bmatrix}$$

and let A(k,j) be equal B(k,j) with the last column replaced by $\rho(k,j)$. Using Cramer's rule the generalized partial autocorrelation function (GPAC) is defined

$$\phi^j_{kk} = \frac{|A(k,j)|}{|B(k,j)|}$$

where ϕ^j_{kk} is the last autoregresive coefficient of an assumed ARMA(k,j) process. For an ARMA(p,q) process

$$\phi^q_{kk} = 0 \qquad k > p \tag{4.15}$$

$$\phi^q_{kk} = \phi_p \qquad k = p \tag{4.16}$$

For an ARMA(p,q) process we could use q+1 through p+q of the Yule-Walker equations (4.12) to solve for the AR parameters. We could also use equations q+i through p+i+q to solve for the same parameters which would lead to

$$\phi^j_{pp} = \phi_p \qquad j \geq q \tag{4.17}$$

Now if we were to construct an array, the GPAC array, using j = 0, 1, 2, ... and k = 1, 2, ... , it would have the patterns (as indicated by (4.15), (4.16), (4.17)) shown in Table 4.1.

	1	...	p-1	p	p+1	p+2
0	ϕ_{11}^{0}	...	$\phi_{p-1,p-1}^{0}$	ϕ_{pp}^{0}	$\phi_{p+1,p+1}^{0}$	$\phi_{p+2,p+2}^{0}$
⋮	⋮		⋮	⋮	⋮	⋮
q−1	ϕ_{11}^{q-1}	...	$\phi_{p-1,p-1}^{q-1}$	ϕ_{pp}^{q-1}	$\phi_{p+1,p+1}^{q-1}$	$\phi_{p+2,p+2}^{q-1}$
q	ϕ_{11}^{q}	...	$\phi_{p-1,p-1}^{q}$	ϕ_{p}	0	0
q+1	ϕ_{11}^{q+1}	...	$\phi_{p-1,p-1}^{q+1}$	ϕ_{p}	u*	u
q+2	ϕ_{11}^{q+2}	...	$\phi_{p-1,p-1}^{q+2}$	ϕ_{p}	u	u

u* = undefined

Table 4.1 GPAC array patterns

At this point it is instructive to look at an example identifying an ARMA(1,1) process.

<u>Example 4.2.3.1</u>

$$Z_t - 0.5\ Z_{t-1} = e_t - 0.8\ e_{t-1} \qquad \sigma_e^2 = 1$$

Using (4.14) we get

$$\rho_z(k) = 0.5\ \rho_z(k-1) \qquad k \rangle 1$$

The actual autocorrelation is

$$\rho_z(k) = 1, -.214, -.107, -.054, \cdots \qquad k = 1, 2, \cdots$$

calculating the GPAC leads to Table 4.2.

q\p	1	2	3
0	-.214	-.16	-.124
1	.5	0	0
2	.5	0/0	0/0

Table 4.2 GPAC for ARMA(1,1)

$$Z_t - .5\ Z_{t-1} = e_t - .8\ e_{t-1}$$

Notice that the pattern is unique and clear when using the actual autocorrelation sequence.

Example 4.2.3.2 (from [2])

$$Z_t - 1.5\ Z_{t-1} + 1.21\ Z_{t-2} - .455\ Z_{t-3} = e_t + .2\ e_{t-1} + .9\ e_{t-2}$$

Table 4.3 gives the GPAC of the above process. It is clear that this process is ARMA(3,2). The AR order, p, is seen to be 3 because $\phi^2{}_{33}$, $\phi^3{}_{33}$, ... are equal and q must be 2 because of the zero pattern. The u pattern (u is actually 0/0, and in practice, where the autocorrelation is estimated, is a mix of large and small numbers) is also clear.

The GPAC can be calculated recursively using the S and R arrays. This will be discussed in the next section.

q\p	1	2	3	4	5	6	7
0	.845	-.706	.414	.299	-.304	-.145	.245
1	.606	-.458	.836	.683	-.434	-.646	.279
2	.391	-.070	.455	.000	.000	.000	.000
3	.328	2.073	.455	u*	u	u	u
4	1.356	-.119	.455	u	u	u	u
5	1.632	5.367	.455	u	u	u	u

u* = undefined

Table 4.3 GPAC of the above ARMA(3,2)

4.2.4 S and R Arrays

In the previous sections, we described time series utilizing the autocorrelation function. Gray and Foster [3] and Gray, Morgan and Houston [4] suggested an equivalent description of series utilizing the power spectrum which is based on a numerical technique called the G_n transform. The S and R arrays appear in the process of calculating the G-spectral estimator, and it is these (S and R) that are utilized for order estimation. This is explicitly discussed in Gray, Kelley, and McIntire [5], where they demonstrated this new approach of ARMA modeling.

A mathematical proof of this method is in [5]. Now let's list the S and R arrays as they are defined in [5].

Definition: Let m be an integer, h > 0, and f be a real-valued function. Also let $f_m = f(mh)$,

$$H_n\,[f_m] = \begin{vmatrix} f_m & f_{m+1} & \cdots & f_{m+n-1} \\ f_{m+1} & f_{m+2} & \cdots & f_{m+n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m+n-1} & f_{m+n} & \cdots & f_{m+2n-2} \end{vmatrix} \tag{4.18}$$

$H_0\,[f_m] \equiv 1$

and

$$H_{n+1}\,[1;f_m] = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ f_m & f_{m+1} & \cdots & f_{m+n-1} & f_{m+n} \\ f_{m+1} & f_{m+2} & \cdots & f_{m+n} & f_{m+n+1} \\ \vdots & \vdots & & \ddots & \vdots \\ f_{m+n-1} & f_{m+n} & \cdots & f_{m+2n-2} & f_{m+2n-1} \end{bmatrix} \tag{4.19}$$

then define

$$R_n(f_m) = \frac{H_n[f_m]}{H_{n+1}\,[1;f_m]} \tag{4.20}$$

and

$$S_n(f_m) = \frac{H_{n+1}[1;f_m]}{H_n[f_m]} \tag{4.21}$$

Pye and Atchinson [6] have shown that $R_n(f_m)$ and $S_n(f_m)$ can be calculated recursively by the following relations. Define

$S_0(f_m) = 1$ $\qquad$ $m=0, \pm1, \pm2, \ldots$

$R_1(f_m) = f_m$ $\qquad$ $m=0, \pm1, \pm2, \ldots$

then

$$R_{n+1}(f_m) = R_n(f_{m+1})\left[\frac{S_n(f_{m+1})}{S_n(f_m)} - 1\right] \tag{4.22}$$

and

$$S_n(f_m) = S_{n-1}(f_{m+1})\left[\frac{R_n(f_{m+1})}{R_n(f_m)} - 1\right] \tag{4.23}$$

for = 1, 2,... and m= 0, ±1, ±2,...

Subroutine gpac.for in appendix B has the code for the above algorithm. In the following discussion it is assumed that the process in discussion is stationary and that $\rho_m = \rho(m)$ is the autocorrelation as defined in (4.6).

Theorem 2 in [5] simply states that for an ARMA (p,q)

$$\begin{aligned} S_n(\rho_m) &= c_1 \\ S_n(\rho_{m_0-1}) &\neq c_1 \end{aligned} \qquad m > m_0$$

iff n = p and m_0 = q - p +1. Moreover,

$$c_1 = (-1)^p\left[1 - \sum_{k=1}^{p} \phi_k\right]$$

where ϕ_k's are the parameters of the AR part.

Theorem 3 in [5] states under the conditions of Theorem 2

$$\begin{aligned} S_n(\rho_m) &= c_2 \\ S_n\left(\rho_{m_1+1}\right) &\neq c_2 \end{aligned} \qquad m \leq m_1$$

iff $n = p$ and $m_1 = -q-p$. Moreover

$$c_2 = -\frac{c_1}{\phi_p}$$

Theorem 4 and 5 in [5] states that under the conditions of theorem 2 and replacing ρ_m by $(-1)^m\rho_m$; we would get the same patterns as specified in theorem 2 and 3 with c_1 and c_2 replaced by c_3 and c_4.

$$c_3 = (-1)^p\left(1-\sum_{k=1}^{p}(-1)^k\phi_k\right)$$

$$c_4 = (-1)^{p+1}\frac{c_3}{\phi_p}$$

Corollary 1 in [5], under the conditions of theorem 2, states

$$R_{n+1}((-1)^m\rho_m) = R_{n+1}(\rho_m) = 0 \qquad m \ge m_0, m \le m_1 - 1$$

and

$$R_{n+1}(\rho_{q-p}) \neq 0$$

$$R_{n+1}(\rho_{-q-p}) \neq 0$$

$$R_{n+1}((-1)^{q-p}\rho_{q-p}) \neq 0$$

$$R_{n+1}((-1)^{-q-p}\rho_{-q-p}) \neq 0$$

iff $n=p$, $m_0=q-p+1$ and $m_1=-q-p-1$. Table 4.4 and 4.5 show, respectively, the S and R array computed using (4.22) and (4.23).

m\n	1	$\cdots$	p	p+1
$-\ell$	$S_1(-\ell)$	$\cdots$	c_2	u^*
$-\ell+1$	$S_1(-\ell+1)$	$\cdots$	c_2	u
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$-q-p-2$	$S_1(-q-p-2)$	$\cdots$	c_2	u
$-q-p-1$	$S_1(-q-p-1)$	$\cdots$	c_2	$\pm\infty$
$-q-p$	$S_1(-q-p)$	$\cdots$	c_2	{2q non-
$\vdots$	$\vdots$		{2q non-	{constants
$q-p$	$S_1(q-p)$	$\cdots$	{constants	$-c_1$
$q-p+1$	$S_1(q-p+1)$	$\cdots$	c_1	u
$\vdots$	$\vdots$		$\vdots$	$\vdots$
j	$S_1(j)$	$\cdots$	c_1	u

$$u^* = c_2\left[\frac{0}{0}-1\right]$$

Table 4.4 S Array

m\n	1	$\cdots$	p	p+1
$-\ell$	$\rho_1(-\ell)$	$\cdots$	$R_1(-\ell)$	0
$-\ell+1$	$\rho_1(-\ell+1)$	$\cdots$	$R_1(-\ell+1)$	0
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$-q-p-2$	$\rho_1(-q-p-2)$	$\cdots$	$R_1(-q-p-2)$	0
$-q-p-1$	$\rho_1(-q-p-1)$	$\cdots$	$R_1(-q-p-1)$	0
$-q-p$	$\rho_1(-q-p)$	$\cdots$	$R_1(-q-p)$	Nonzero
$\vdots$	$\vdots$		$\vdots$	Nonzero
$q-p$	$\rho_1(q-p)$	$\cdots$	$R_1(q-p)$	Nonzero
$q-p+1$	$\rho_1(q-p+1)$	$\cdots$	$R_1(q-p+1)$	0
$\vdots$	$\vdots$		$\vdots$	$\vdots$
j	$\rho_1(j)$	$\cdots$	$R_1(j)$	0

$$u^* = c_2\left[\frac{0}{0}-1\right]$$

Table 4.5 R Array

m\n	1	⋯	p	p+1
$-\ell$	$S_1(-\ell)$	⋯	c_2	u^*
$-\ell+1$	$S_1(-\ell+1)$	⋯	c_2	u
⋮	⋮		⋮	⋮
$-q-2$	$S_1(-q-2)$	⋯	c_2	u
$-q-1$	$S_1(-q-1)$	⋯	c_2	$\pm\infty$
⋮	⋮	⋯	{2q non-	{2q non-
$q-1$	$S_1(q-1)$	⋯	{constants	{constants
q	$S_1(q)$	⋯	c_1	$-c_1$
$q+1$	$S_1(q+1)$	⋯	c_1	u
⋮	⋮		⋮	⋮
j	$S_1(j)$	⋯	c_1	u

$$u^* = c_2\left[\frac{0}{0} - 1\right]$$

Table 4.6 Shifted S Array

Woodward and Gray [2] suggest the use of the shifted S & R arrays, which is basically shifting the $(i+1)^{th}$ column i-1 times downward. Table 4.6 shows the shifted S array

The S array for the ARMA(3,2) of Example 4.2.3.2 is given in Table 4.7. An ARMA(3,2) can be easily identified from this array

The relationship between the GPAC and the S and R arrays is best described in [2] and is regulated by

$$\phi^j_{kk} = \frac{-S_k(\rho_{-k+j+1})}{S_k(\rho_{-k-j})} \qquad (4.24)$$

m\n	1	2	3	4	5	6
-6	-1.613	1.247	-.9154			
-5	-1.737	23.860	-9.154	u*		
-4	-4.052	2.915	-9.154	u	u	
-3	-3.556	22.437	-9.154	±∞	±∞	±∞
-2	-2.651	5.686	-7573	1.470	6.806	-7.911
-1	-2.184	4.456	-10.750	-10.450	7.143	-65.674
0	-1.845	3.148	-4.452	3.122	-2.173	2.489
1	-1.606	2.606	-6.334	-1.004	-3.527	-1.090
2	-1,391	1.578	-4.165	4.165	-4.165	4.165
3	-1.328	-6.044	-4.165	u	u	
4	-2.356	2.838	-4.165	u		
5	-2.632	-6.691	-4.165			

u* = undefined

Table 4.7 The shifted S array of ARMA(3,2) in Example 4.2.3.2

The S and R arrays can provide clearer patterns than the GPAC (as will be shown next), and they can also, along with the GPAC, be extended to the complex variable case, Using different frequencies to obtain more flexibility. For instance, $(-1)^m \rho_m = e^{j\pi m} \rho_m$ satisfies (4.14), In fact $\rho_m e^{j2\pi fm}$ (where f is the frequency) will always satisfy (4.14) [7]. S and R arrays have complex entries for $f \neq 0.5n$ (n=0,1,2,...).

The following example illustrates why sometimes the S array provides more information than the GPAC. Consider the ARMA(2,1) process [2]

$$Z_t - 0.5\, Z_{t-1} + 0.5\, Z_{t-2} = e_t - e_{t-1}$$

Table 4.8 shows the GPAC of the above ARMA using the actual autocorrelation sequence. We notice that the ϕ_p pattern starts in row zero rather than in the row one suggesting that the process might be ARMA(1,1). But the zero pattern, in row one, correctly identifies the process as ARMA(2,1). Practically, when using the sample ACF, it is difficult to recognize the small entries in row zero as nonzero numbers. In the S array of Table 4.9 there is little doubt that the process in question is ARMA(2,1).

4.3 Using MacModel for ACF, CCF, PACF, GPAC, S array, and R array

4.3.1 Example on ACF

To calculate the sample autocorrelation sequence, choose "Autocorrelation" under the "Identify" menu as shown in Figure 4.3.

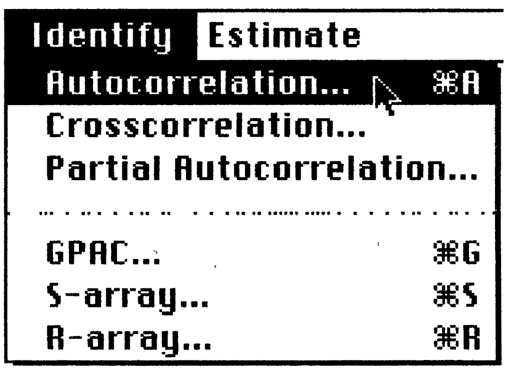

Figure 4.3 Selection of the Autocorrelation menu

q\p	1	2	3	4	5	6
0	0.0	-.5	-.333	-.25	-.2	.1.67
1	-∞	-.5	0.0	0.0	0.0	0.0
2	.5	-.5	u	u	u	u
3	-.5	-.5	u	u	u	u
4	1.5	-.5	u	u	u	u

Table 4.8 GPAC of an ARMA(2,1)

m\n	2
-5	2
-4	2
-3	2
-2	2
-1	3
0	1.5
1	1
2	1
3	1
4	1

Table 4.9 Column 2 of the shifted S array of an ARMA(2,1)

Then you will be prompted for the number of lags needed. The number of lags defaults to 20, but can be as high as 1000 (Entering a number higher than 1000 for the lags, results in a display of a dialog box informing you of the limit and that the number of lags is set to that limit). The option "Plot ACF sequence" is preselected, if you wish not to plot the sequence, click once on the check box. Next you will be prompted for the input file and then for the output file. Figure 4.4 shows the ACF of an ARMA(2,1).

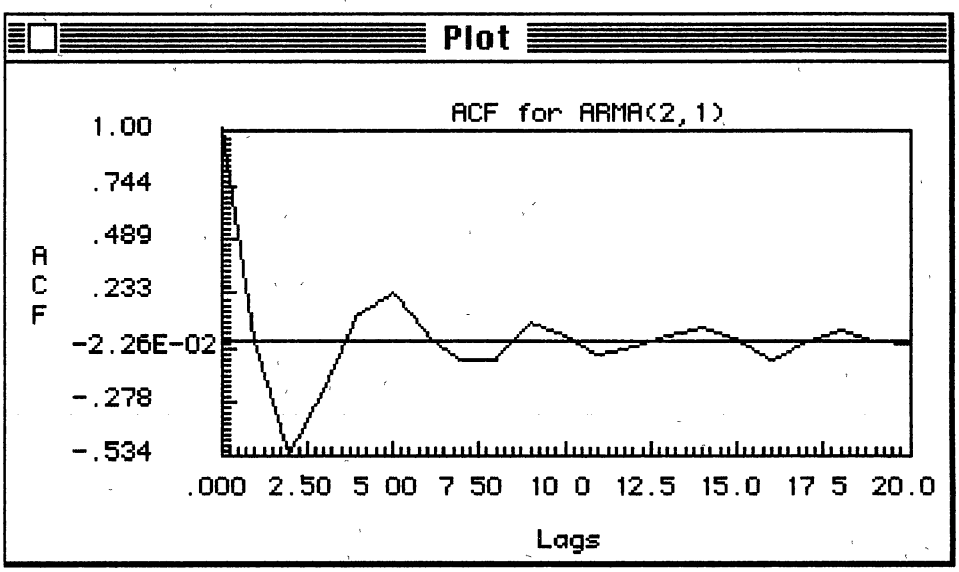

Figure 4.4 ACF of an ARMA(2,1)

4.3.2 <u>Example of CCF</u>

To calculate the crosscorrelation sequence, select the "Crosscorrelation " from the "Identify" menu as in Figure 4.5. The rest of the procedure is identical to that of the ACF with the exception of an extra input file. The order of the input files is

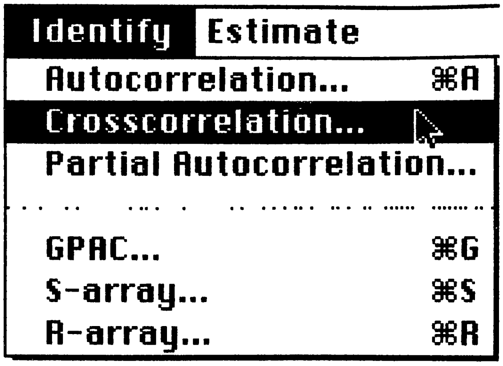

Figure 4.5 Selection of the Crosscorrelation menu

important since $\gamma_{zx} \neq \gamma_{xz}$. Figure 4.6 has the result of crosscorrelating the same ARMA process used in the ACF example with itself.

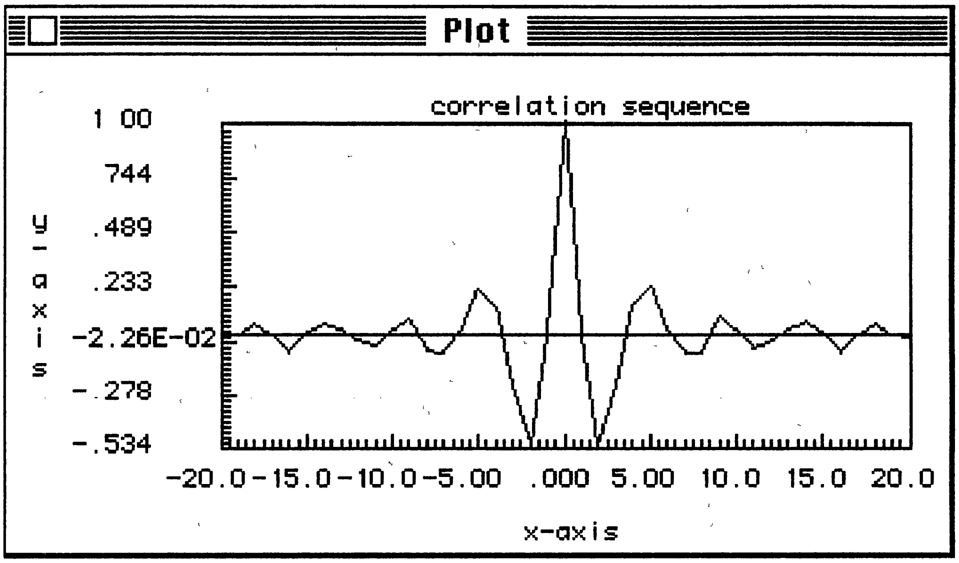

Figure 4.6 CCF of an ARMA(2,1)

4.3.3 Example of PACF

The process used in here is the same one used in Example 4.2.2.1. choose "Partial Autocorrelation" in the "Identify" menu (see Figure 2.8), then you will be prompted for the number of PACF lags.

Again a maximum of 1000 lags is allowed. The program will then prompt you for the ACF input file (If the number of PACF lags requested is not less than the number of the ACF lags, a dialog box will be displayed to inform you of the maximum number of lags allowed for the selected ACF sequence). If the Plot PACF sequence was checked, the results would be displayed to the screen as in Figure 4.7.

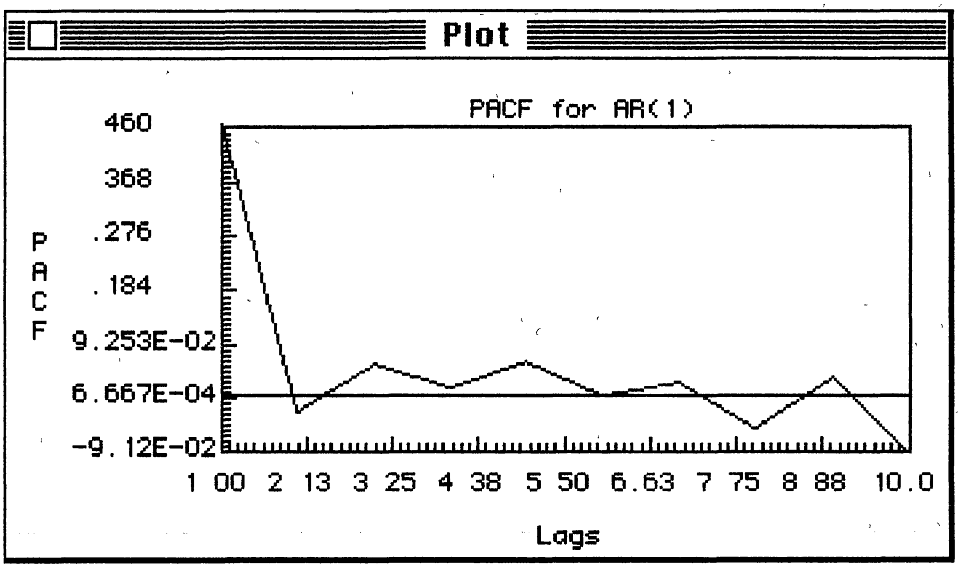

Figure 4.7 PACF of the process in Example 4.2.2.1

4.3.4 Grayscale Map and Examples on GPAC, S Array, and R Array

After calculating the arrays (GPAC, S, or R) the gpac.for subroutine calls the plotgsr.for subroutine. The plotgsr.for routine takes the array just calculated then prompts the user for parameters to map the array numbers to one of the 15 graylevels that it has. Plotgsr.for uses one of two functions to map the arrays: Sigmoid or

Linear.

Sigmoid

The Sigmoid uses the equation in Figure 4.8. The variable x ,in Figure 4.8, is an array entry and y is the new mapped entry. The a and b variables are the sigmoid function parameters. Selecting a and b is crucial because they control graylevel resolution. A low value for a will place relatively close entries in the array in the same graylevel bin. On the other hand, a higher value for a might place these same entries in different bins. One would choose a to be inversely proportional to the standard deviation of the array. Select b so that the data effectively uses the grayscale. The best value for b might be around the mean of the array. The default values for b and a will be the mean and 1/std respectively.

$$y = \frac{1}{1+e^{-a(x-b)}}$$

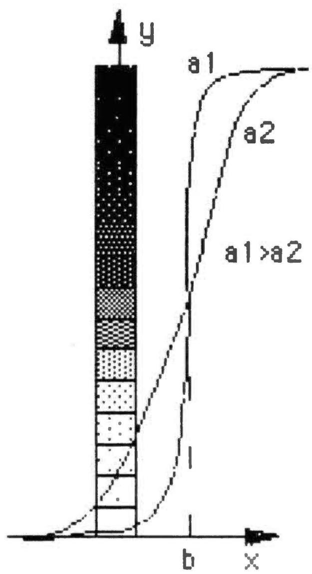

Figure 4.8 Equation and graylevel of the Sigmoid function

Linear

This Linear function maps the values of the array from the interval [a,b] to the graylevels. Its also clips the entries of the array to a minimum of min(a,b) and a maximum of max(a,b). Figure 4.9 shows the Equation and the graylevel of the linear function.

$$y = \frac{14}{b-a}(x-a)+1$$

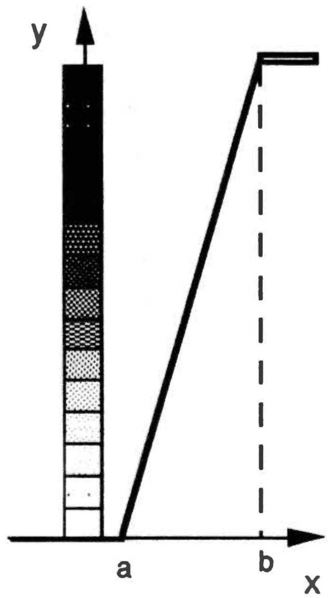

Figure 4.9 Equation and graylevel of the linear function

Choosing a and b close in value will increase the resolution of the graylevel inside [a,b]. Keep in mind, however, for this to be effective (a+b)/2 should be close to the mean. The default values for a and b will be the mean ± the standard deviation.

After displaying the graylevel map, plotgsr displays a dialog box (see Figure 4.10) for adjusting a and b. At this point a and b can redrawn. The magnitude of the increments can be adjusted interactively. When the grayscale map is satisfactory the "Done"

Figure 4.10 The dialog box for adjusting a and b

button is selected. Here are some helpful tips for adjusting a and b:

- If most entries become black or white, then the [a,b] interval is too small and it needs to be enlarged.
- If increasing (decreasing) a or b does not affect the graylevel map, then the mapped array entries are smaller (larger) than a or b, whichever is being changed, and thus the limit is reached.
- The chess pattern (shown in Figure 4.11) is the zero pattern. It replaces the patterns of the entries that fall in the zero bin (adjusting a or b may increase or decrease the bin size, and consequently array entries may enter or leave the zero bin).

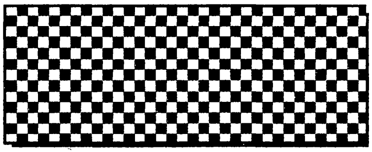

Figure 4.11 The chess pattern that indicates small values

To display the graylevel map of the GPAC (or for S or R arrays) just select the appropriate menu item under "Identify" (see Figure 2.8), select an ACF input file, enter the size of the array, select one of the functions (Sigmoid or Linear), enter a and b (or use the default), and then adjust a and b using the dialog box displayed above the array map. Figures 4.12, 4.13, and 4.14 show the grayscale GPAC, S array and R array of the ARMA(3,2) of Example

4.2.3.2 respectively. Note that the shifted S and R arrays will always be used.

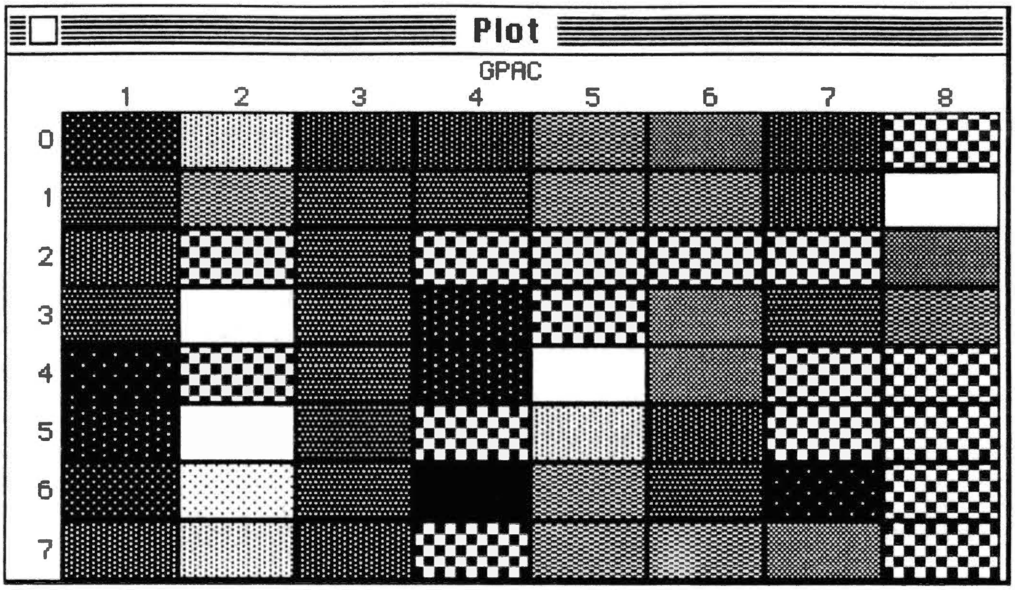

Figure 4.12 GPAC grayscale map of the ARMA in Example 4.2.3.2

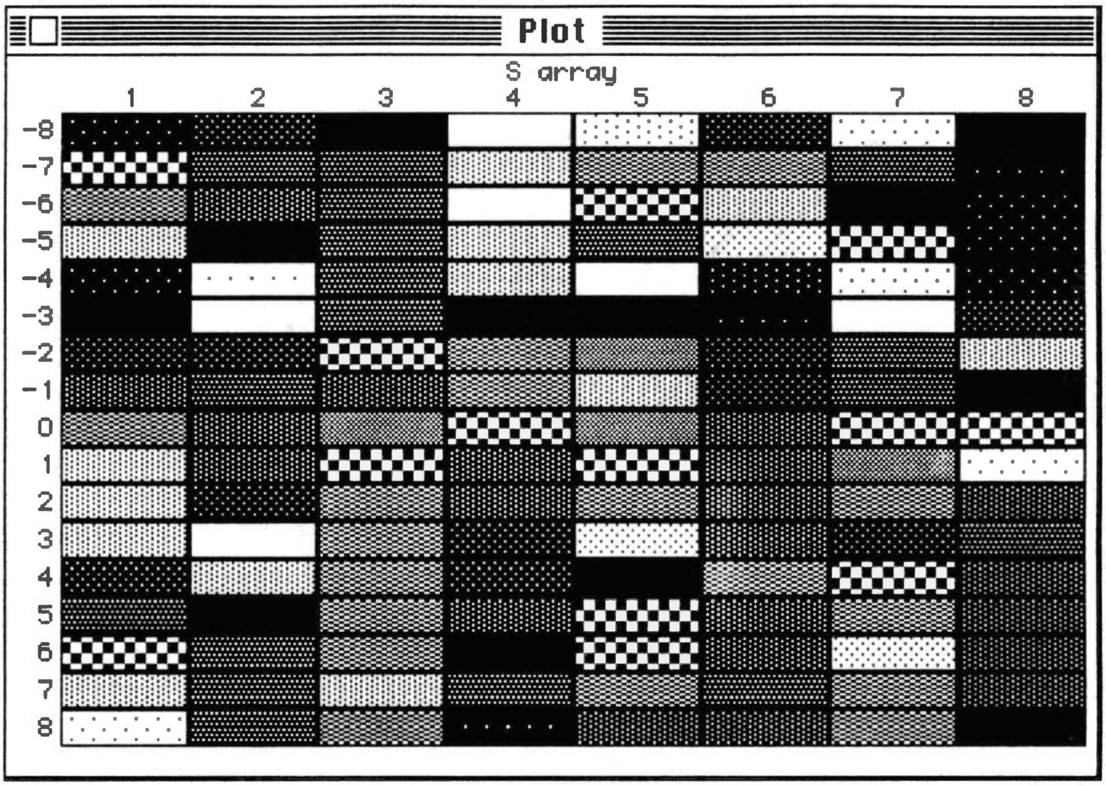

Figure 4.13 Shifted S array grayscale map of the ARMA in Example 4.2.3.2

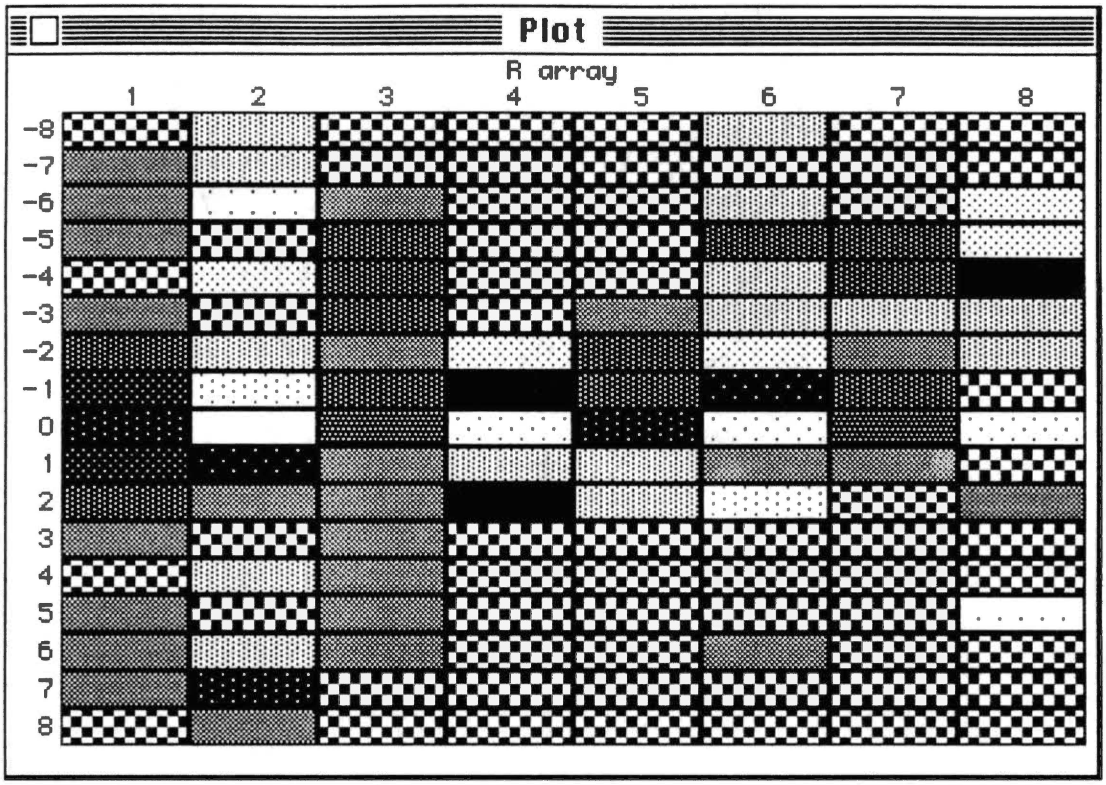

Figure 4.14 Shifted R array grayscale map of the ARMA in Example 4.2.3.2

CHAPTER V

PARAMETER ESTIMATION

Once the order of a process is identified, as described in chapter IV, the process parameters can be estimated, as depicted in the flowchart of Figure 1.1. The Maximum-Likelihood Estimation (MLE) procedure, which is well suited for estimating parameters when apriori knowledge of the parameters is not available, is described next.

5.1 Maximum Likelihood Estimation

Consider independent random variables $Z_1, Z_2, \ldots Z_N$ and observations $z_1, z_2, \ldots z_N$, where N is the number of observations. Also consider parameter vector θ. Let $f_Z(z;\theta)$ be the probability density function (pdf) of Z, and $f_{Z_1, \ldots, Z_N}(z_1, z_2, \ldots z_N;\theta)$ be the joint pdf of $Z_1, Z_2, \ldots Z_N$. Let us define

$$\ell(\theta) = f_{Z_1,Z_2,\cdots,Z_N}(z_1,z_2,\cdots,z_N,\theta) \tag{5.1}$$

as the likelihood function of θ. In most cases the pdf has exponential form , e.g. gaussian, so let us define

$$L(\theta) = \ln\left[\ell(\theta)\right] \tag{5.2}$$

Maximizing $L(\theta)$ is equivalent to maximizing $\ell(\theta)$ because the logarithm of $\ell(\theta)$ is monotic transformation of $\ell(\theta)$.

Substituting (5.1) into (5.2) and keeping in mind that $Z_1, Z_2, \dots Z_N$ are independent

$$L(\theta) = \ln\left[f_{Z_1}(z_1,\theta)\, f_{Z_2}(z_2;\theta)\cdots f_{Z_N}(z_N;\theta)\right] \tag{5.3}$$

maximizing $L(\theta)$ implies

$$\frac{\partial}{\partial\theta}L(\theta)\Big|_{\theta=\hat{\theta}_{ML}} = 0 \tag{5.4}$$

where $\hat{\theta}_{ML}$ is the value that maximizes $L(\theta)$, or $\ell(\theta)$.

Consider a taylor series expansion of $L(\theta)$ about $\hat{\theta}_{ML}$

$$L(\theta) = L(\hat{\theta}_{ML}) + \sum_{i=1}^{N}\frac{\partial}{\partial\theta_i}L(\theta)\Big|_{\theta=\hat{\theta}_{ML}}(\theta_i - \hat{\theta}_{i,ML}) + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial^2}{\partial\theta_i\partial\theta_j}L(\theta)\Big|_{\theta=\hat{\theta}_{ML}}(\theta_i - \hat{\theta}_{i,ML})(\theta_j - \hat{\theta}_{j,ML}) + \cdots \tag{5.5a}$$

For $\theta = \hat{\theta}_{ML}$ the second term of the right hand side of (5.5a) is zero (see (5.4)). Equation (5.5a) becomes

$$L(\theta) = L(\hat{\theta}_{ML}) + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial^2}{\partial\theta_i\partial\theta_j}L(\theta)\Big|_{\theta=\hat{\theta}_{ML}}(\theta_i - \hat{\theta}_{i,ML})(\theta_j - \hat{\theta}_{j,ML}) + \cdots \tag{5.5b}$$

We notice that (5.5b) is quadratic and can be written as

$$L(\theta) = L(\hat{\theta}_{ML}) + \frac{1}{2}(\theta - \hat{\theta}_{ML})^T J_0(\theta)\Big|_{\theta=\hat{\theta}_{ML}}(\theta - \hat{\theta}_{ML}) + \cdots \tag{5.5c}$$

where J_0 is called the Fisher information matrix and is defined as

$$J_0(\theta) = \frac{\partial^2}{\partial\theta_i \partial\theta_j}[L(\theta)|_{\theta = \hat{\theta}_{ML}}] \qquad i,j = 1,2,\cdots,N$$

Equation (5.5c) can be approximated as

$$L(\theta) \approx L(\hat{\theta}_{ML}) + \frac{1}{2}(\theta - \hat{\theta}_{ML})^T J_0(\theta)|_{\theta=\hat{\theta}_{ML}}(\theta - \hat{\theta}_{ML}) \qquad (5.6)$$

for $L(\hat{\theta}_{ML})$ to be maximum, the second term of (5.6) should be negative (i.e. the Fisher information matrix, J_0, should be negative definite).

Now consider the ARMA process in (1.6). Assuming e_t is normally distributed, zero mean, and independent, the pdf for (e_1, ..., e_N) can be written as

$$f_{e_1,\cdots,e_N}(e_1,\cdots,e_N) = \frac{1}{(2\pi)^{\frac{N}{2}}\sigma_e^N} e^{\frac{-\sum_{i=1}^{N} e_i^2}{2\sigma_e^2}} \qquad (5.7)$$

The logarithm of the pdf becomes

$$-\frac{1}{2\sigma_e^2}\sum_{i=1}^{N} e_i^2 - N\ln[\sigma_e] + \frac{N}{2}\ln[2\pi] \qquad (5.8)$$

Since the Z_t sequence is a one-to-one transformation of the e_t sequence (given certain initial conditions) and the Jacobian of the transformation is unity, (5.7) is also the likelihood function, where the e_t are computed from the Z_t sequence. Thus (5.8) is also the logarithm of the likelihood function. Minimizing (5.8) implies minimizing the sum of the squares

$$S(\theta) = \sum_{i=1}^{N} e_i^2$$

5.2 Maximum Likelihood Algorithm

The mle.for subroutine implements Marquardt's Algorithm for estimating ARMA process parameters. The algorithm, which avoids calculating the second derivatives, utilizes two methods: the Gauss-Newton linearization method, and the gradient method (also known as the steepest descent). The former procedure's advantage is that it tends to converge rapidly to the least squares estimates (convergence occurs in one iteration if s(θ) is quadratic in θ, normally it takes few iterations); the disadvantage is that it may never converge. The biggest advantage of Steepest descent is that it will always converge to the least square estimates (if the step size is chosen appropriately); the problem, however, is it may have a slow convergence rate. The following are the equations that govern the above mentioned methods:

Gauss-Newton

$$\Delta\theta = -\left[\frac{\partial^2}{\partial\theta^2} S(\theta)\right]^{-1} \frac{\partial}{\partial\theta} S(\theta)$$

Gradient method

$$\Delta\theta = -[\alpha]^{-1}\frac{\partial}{\partial\theta}S(\theta)$$

Marquardt

$$\Delta\theta = -\left[\frac{\partial^2}{\partial\theta^2}S(\theta)+\alpha I\right]^{-1}\frac{\partial}{\partial\theta}S(\theta)$$

For large α the first term to the right of the equal sign (in Marquardt's) is approximately $[1/\alpha]$, and the Marquardt algorithm reduces to the gradient method. For small α, the same term is approximately $[(\partial^2/\partial\theta^2)\,S(\theta)]^{-1}$ which reduces to the Gauss-Newton technique.

5.2.1 MLE Algorithm

Let $\boldsymbol{\theta}$ be the parameter vector of the AR and MA parts of the process

$$\boldsymbol{\theta} = [\phi_1 \cdots \phi_p \; \theta_1 \cdots \theta_q]^T$$

and let $\boldsymbol{\theta}^\circ$ be the current guess for $\boldsymbol{\theta}$ (assume $\boldsymbol{\theta}^\circ$=0). Also let $e_{t,o}=e_t(\boldsymbol{\theta}^\circ)$. The Taylor series expansion of e_t is

$$e_t = e_{t,o} + \sum_{l=1}^{p+q}\left.\frac{\partial e_t}{\partial\theta_l}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}(\theta_l - \theta_{l,o}) + \cdots \qquad (5.9)$$

Linearize e_t by considering only the first two terms. Now let

$$x_{l,t} = -\left.\frac{\partial e_t}{\partial\theta_l}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^0}$$

$$\mathbf{e}^0 = \left[e_1^0 \quad \cdots \quad e_N^0\right]^T$$

$$\mathbf{e} = \left[e_1 \quad \cdots \quad e_N\right]^T$$

Where t and i are the row number and column number respectively. Using vector format, (5.9) becomes

$$\mathbf{e} = \mathbf{e}^0 - \mathbf{x}[\boldsymbol{\theta} - \boldsymbol{\theta}^0]$$

or, if we want to minimize $S(\theta) = e^T e$,

$$[\boldsymbol{\theta} - \boldsymbol{\theta}^0] = [\mathbf{x}^T\mathbf{x}]^{-1}\mathbf{x}^T\mathbf{e}^0 \tag{5.10}$$

where $[\boldsymbol{\theta} - \boldsymbol{\theta}^\circ]$ is the correction term. Since e_t is a function of $\boldsymbol{\theta}$, to minimize $S(\boldsymbol{\theta})$, derived in section 5.1, we need to calculate the negative of the derivatives of e_t with respect to θ (i.e. $x_{i,t}$). The negative of the derivative is approximated as follows

$$x_{i,t} = \frac{\left[e_t\left(\left[\theta_{1,0},\cdots,\theta_{i,0},\cdots,\theta_{p+q,0}\right]\right) - e_t\left(\left[\theta_{1,0},\cdots,\theta_{i,0}+\delta_i,\cdots,\theta_{p+q,0}\right]\right)\right]}{\delta_i}$$

5.2.2 <u>Pseudo Code of MLE Algorithm</u>

<u>Notation:</u>

δ	Step size for derivative calculations (typically 0.01). δ could be different for each θ_i.
PI	Correction parameter (typically 0.01). Larger PI moves the correction toward the gradient method. PIMAX is the

maximum allowed value for PI before stopping the algorithm without convergence.

ITERMAX The maximum number of iteration allowed before stopping the execution of the algorithm without convergence.

ε Tolerance to check for parameter convergence (typically 0.001).

F_2 Acceleration factor (typically between 1 and 2).

Stage 1:

Calculate the derivative

$$x_{l,t} = \frac{\left[e_t\left(\left[\theta_{1,0},\cdots,\theta_{l,0},\cdots,\theta_{p+q,0}\right]\right) - e_t\left(\left[\theta_{1,0},\cdots,\theta_{l,0}+\delta_l,\cdots,\theta_{p+q,0}\right]\right)\right]}{\delta_l}$$

$$A_{l,j} = \sum_{l=n}^{n} x_{l,t}x_{j,t} \qquad \left\{\mathbf{A} = \left[A_{lj}\right] = \left[\mathbf{x}^T\mathbf{x}\right]\right\}$$

$$g_l = \sum_{l=n}^{n} x_{l,t}e_t \qquad \left\{\mathbf{g} = \mathbf{x}^T\mathbf{e}^0\right\}$$

$$D_l = \sqrt{A_{ll}}$$

Stage 2:

$$A^*_{l,j} = \frac{A_{lj}}{D_l D_j}$$ {normalize **A** for numerical stability}

$$A^*_{l,l} = 1 + PI$$ {combination of the two methods}

$$g^*_l = \frac{g_l}{D_l}$$ {normalize for numerical stability}

Solve the equation

$$\mathbf{A}^*\mathbf{h}^* = \mathbf{g}^*$$ $\{\mathbf{h}^* = [\boldsymbol{\theta} - \boldsymbol{\theta}^0]$. Change in $\boldsymbol{\theta}\}$

For $\mathbf{h}^*$ then scale back to obtain

$$h_j = \frac{h^*_l}{Dj}$$ {because $\mathbf{A}^*$ and $\mathbf{g}^*$ were normalized}

Calculate the new $\boldsymbol{\theta}$

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta}_{old} + \mathbf{h}$$

Stage 3:

Check for reduction of the sum of squares. If the sum of squares is not reduced, increase PI (move toward the steepest descent technique), otherwise reduce PI and check for parameter convergence

```
if S(θnew) < S(θold) then
   if |hi| < ε then              {check for parameter convergence}
      parameters converged
      done
   else
      θold = θnew
      PI = PI / F2               {move toward Gauss-Newton tech.}
      iter = iter + 1
      if iter > itermax then
         error message
         stop
      endif
   endif
else
   PI = PI * F2                  {move toward steepest descent tech.}
   if PI > PIMAX then
      error message
      stop
   endif
   GOTO stage 2
endif
GOTO stage 1
```

5.3 Using MacModel for MLE

Example 5.3.1

To calculate the Maximum Likelihood Estimate of an ARMA process choose "Maximum Likelihood" from the "Estimate" menu. You will be confronted with the "Maximum Likelihood Parameters" dialog box, which requests the process order and the maximum number of iterations. See Figure 5.1.

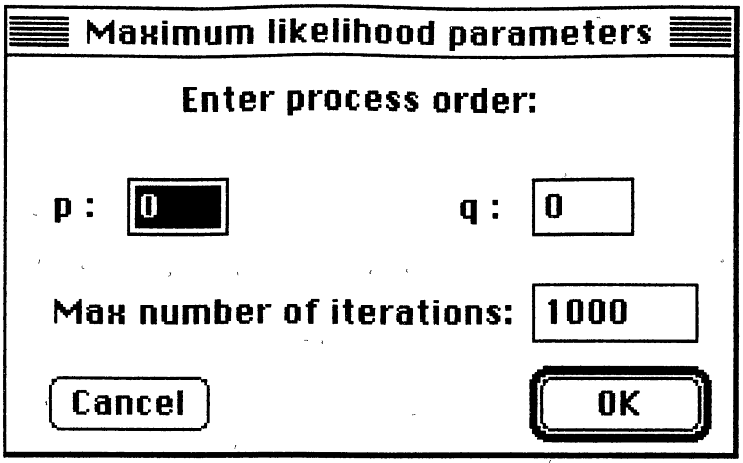

Figure 5.1 Maximum Likelihood Parameters dialog box

Next the program displays the iteration number and the sum of squares as they change with the routine execution. See Figure 5.2.

Upon finishing the execution, the MLE routine displays the "MLE Report" dialog box, which contains the results of the run (See Figure 5.3). Displayed in the top center of the report is the convergence information. The number of degrees of freedom (the number of data points minus the number of estimated parameters) is also displayed, along with the standard deviation of the residuals (e_t's) and the number of iterations completed. Each parameter (AR or MA) and its standard deviation are enclosed by parentheses. The

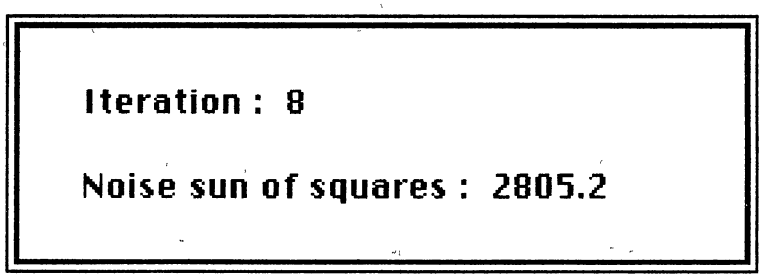

Figure 5.2 Iteration number and the sum of squares in MLE

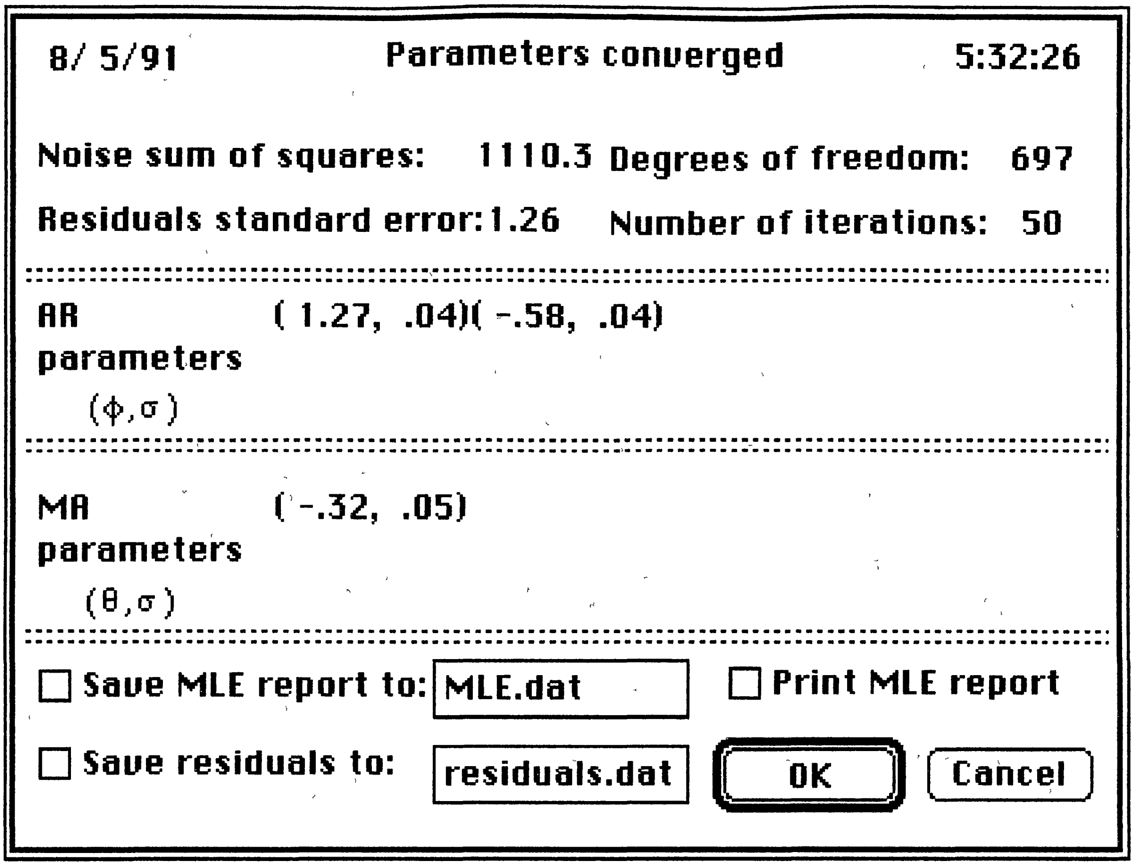

Figure 5.3 MLE results report

first pair of parentheses (in the AR region) enclose ϕ_1, the second enclose ϕ_2, etc. The MA parameters are displayed in a similar fashion in the MA region. The bottom part of the report contains three check boxes, two of which are for saving the report and the residuals. The third is for printing a hard copy of the report. This is equivalent to saving the report, then printing it by selecting "Print Text" from the File menu.

CHAPTER VI

DIAGNOSTIC TESTING

Once a model has been identified and the parameters estimated, diagnostic checks must be applied to check the adequacy of the model (see flowchart of Figure 1.1). There are many procedures to check for accuracy of an identified model, one of which is testing the whiteness of the residuals.

The residual-whiteness test is a procedure that checks if the residuals are truly white (see Chapter IV on how to obtain the residuals). This is done by inspecting the autocorrelation function, ρ_e. Theoretically, $\rho_e(k)=0$ for all $k\neq0$ and $\rho_e(0)=1$. Due to the use of the sample autocorrelation, the residuals will not look perfectly white. A rule of thumb to check for whiteness is to see if the autocorrelation is within 2 standard deviations of zero. The standard deviation of the sample autocorrelation function will be $1/\sqrt{N}$ where N is the number of data points [8].

Example

Consider the following AR(2) process

$$Z_t - 1.7\ Z_{t-1} + 0.72\ Z_{t-2} = e_t$$

Let's assume that we miss identified the process as AR(1). The MLE of the parameter of the AR(1) process for 1500 data points was found to be

$$Z_t - 0.98\, Z_{t-1} = e_t$$

Now let's perform some diagnostic checking. The ACF sequence of the residuals is shown in Figure 6.1.

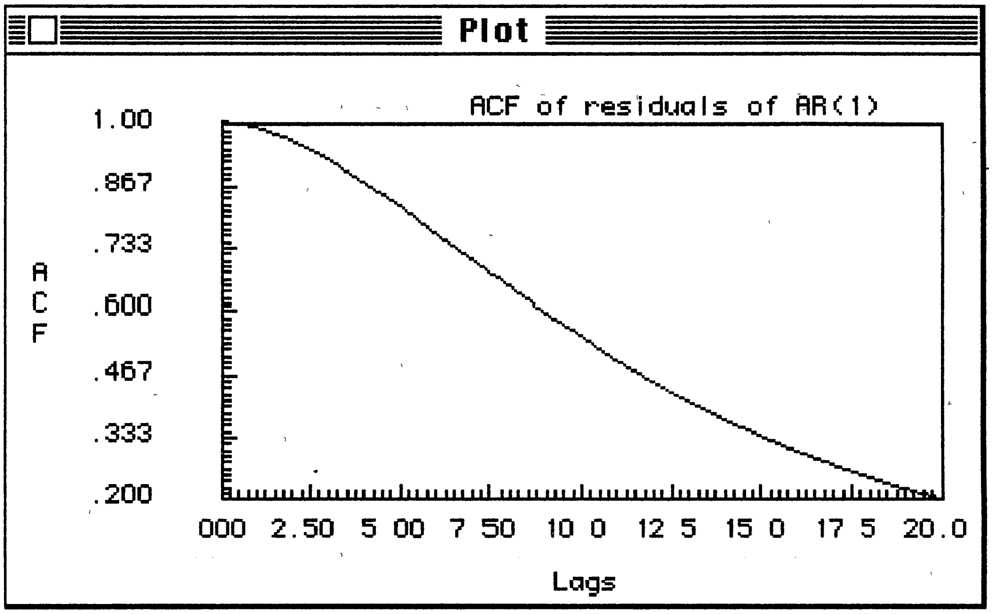

Figure 6.1 The ACF of the residuals

The residuals are not white since the ACF sequence is not confined to ± $2/\sqrt{N}$ for Lags>0 (N=1500 points in this example), in fact it looks like it is for an AR(1) since it has exponential shape with no oscillations. Furthermore the GPAC of the residuals in Figure 6.2 clearly identifies the process as AR(1). If we increase the order of

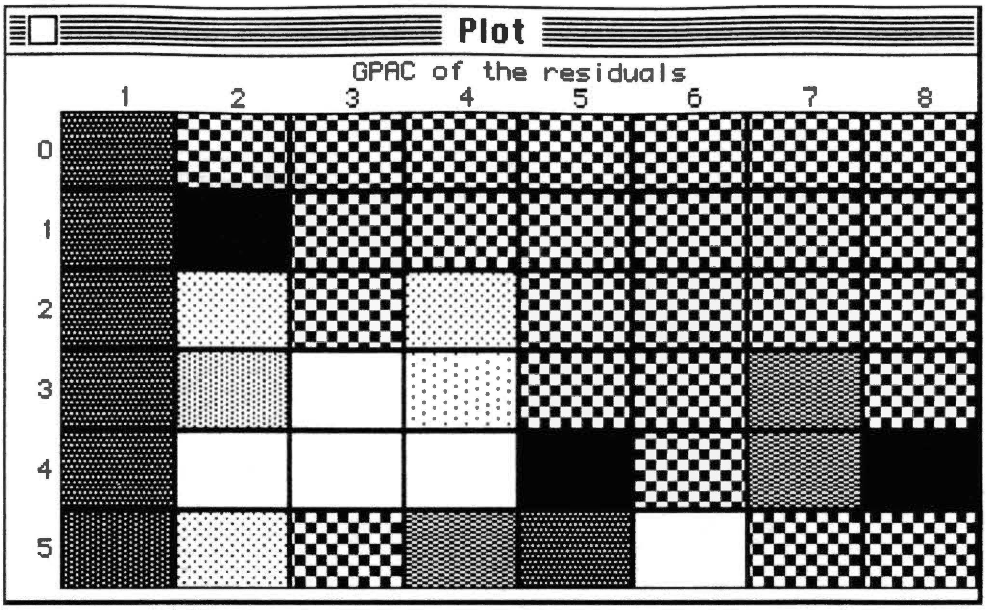

Figure 6.2 The GPAC of the residuals

the model by one, namely to an AR(2), the following parameter estimates are produced

$$Z_t - 1.69\, Z_{t-1} + 0.72\, Z_{t-2} = e_t$$

which are very close to the actual parameters.

When we perform diagnostic checks on the residuals of the AR(2) model, we get the ACF in Figure 6.3 and the GPAC in Figure 6.4. The ACF is confined to $\pm 2/\sqrt{N}$, and the first row of the GPAC is zero which is typical of a white noise process.

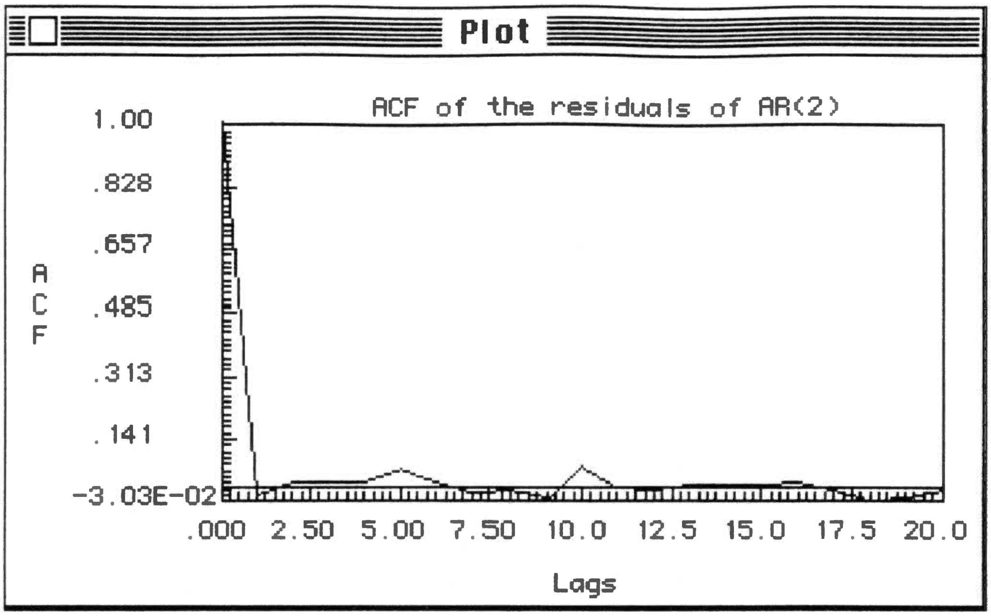

Figure 6.3 The ACF of the residuals of AR(2)

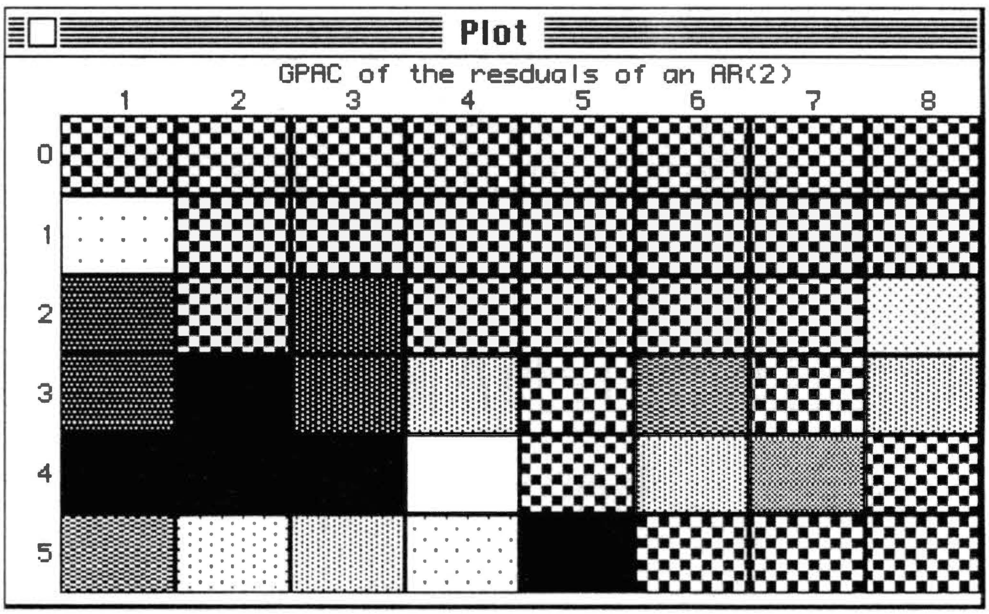

Figure 6.4 The GPAC of the residuals of AR(2)

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

This thesis discussed process identification from preliminary order determination through parameter estimation and diagnostic testing. From the diagnostic tests we learned about the model adequacy. The main thrust of this thesis was a Macintosh application program, called MacModel, written for system identification. MacModel is a menu-driven, user friendly program that conforms with Macintosh application standards.

Chapter II discussed the Macintosh Toolbox structure and the application program structure, with MacModel's main-routine flowchart depicted in Figure 2.10. It contained discussion on how to allocate, initialize, and customize menus to your needs. This was followed by an explanation on how to allocate and manipulate windows and their grafports. Easy to follow instructions on how to start-up, use, and expand MacModel were also included. Chapter III included white noise and ARMA processes simulation with examples on using MacModel. The program has plotting capabilities and uses grayscale to display graylevel array maps that make it easier to recognize patterns in the GPAC, S, and R arrays. A listing of all the routines is included in appendix B.

Chapters IV through VI included the theory behind system identification and parameter estimation. It was shown that for an MA(q) process it was sufficient to inspect the ACF sequence; For an AR(p) process the order p was determined by the PACF sequence only. For an ARMA(p,q) process, however, the GPAC had to be inspected to estimate the process order. Three patterns were formed in the GPAC: a vertical pattern, called the ϕ_p pattern, which occurs in the p^{th} column of the GPAC, a horizontal pattern, called the zero pattern, which occurs in the q^{th} row, and a block pattern, known as the 0/0 pattern, which starts to the right of the ϕ_p pattern and below the zero pattern.

The GPAC patterns were found not to stand-out well for some processes, so the S and R arrays method was explored. The S array has 4 patterns, which are the c_1, c_2, $\pm c_1$ and $\pm\infty$ patterns. Then the shifted S and R arrays where introduced to make it easier to recognize patterns. Complex S and R arrays were briefly discussed. It was found that $e^{j2\pi fm} \rho_m$ satisfies the same difference equation that does ρ_m. The complex arrays were found (for frequency $f \neq n/2$, $n=0,1,2,...$) to give twice the number of patterns of the real case. This is also true for the GPAC since (4.24) showed how to recursively calculate the GPAC from the arrays. Diagnostic testing was discussed in Chapter VI to aid in determining the adequacy of the identified model. Figure 1.1 depicts a flowchart that can be used as a guide for process identification.

In conclusion we recommend that the S and R arrays be explored further, especially the complex approach, as they are virtually unknown or unused in the engineering fields. The complex S and R arrays can give enhanced patterns and can be used to recursively calculate a complex GPAC. We also recommend continued development of MacModel. It can be easily expanded to include new system identification techniques. The use of graylevel maps to display the GPAC and S and R arrays seems to allow easier recognition of patterns. Further studies should be made of this capability.

Finally we recommend the addition or improvement of the following:

- Improve the routines to make them more numerically stable (e.g. the use of unusually large numbers may cause numerical overflows).
- Append a subroutine to find the poles of ARMA and AR processes to check for stability. Unstable processes often cause numerical overflow.
- Add the frequency as a new variable in the dialog box of Figure 4.10. The ability to adjust the frequency provides an extra degree of freedom in optimizing the graylevel map. This should enhance the patterns further. After adding the frequency, one would adjust it until a large variance of the array is obtained, then adjust a and b as discussed in section 4.3.4.
- Modify the program to include a preference menu and a profile. This would allow the users to customize the environment.
- Refresh the Plot and TTY windows.

- Modify plotgsr.for to allow the users to display a value in the array (GPAC, S, or R arrays) by clicking on the its corresponding graylevel in the map.
- Adjust the print manager to default to landscape instead of portrait upon choosing Print Graphics in the File menu.

REFERENCES

[1] Apple (1989). Inside Macintosh, New York: Addison Wesley.

[2] Woodward, W. A., and Gray, H. L. (1981). On the Relationship Between the S Array and the Box-Jenkins Methods of ARMA Model Identification. Journal of the American Statistical Association, Vol. 76, No. 375, Theory and Methods Section, p.p. 579-587.

[3] Gray, H. L., and Foster, M. R. (1975). Autoregressive, Maximum Entropy, and G-Spectral Estimation. The Search for Oil: Some Statistical Methods and Techniques, New York: Marcel Dekker, p.p.169-189.

[4] Gray, H. L., Houston, A. G., and Morgan, F.W. (1978). On G-Spectral Estimation, Proceeding of the 1976 Tulsa Symposium on Applied Time Series, New York: Academic Press.

[5] Gray, H. L., Kelley, G. D.,and McIntire, D. D. (1978). A New Approach to Arma Modeling, Commun. Statist. Vol B7, p.p. 1-78.

[6] Pye, W. C., and Atchison, T. A. (1973). An Algorithm for the Computation of the Higher Order G-Transformation. SIAM J. Numer. Anal, Vol. 10, No. 1, p.p. 1-7.

[7] Morton, M. J., and Gray, H. L. (1978). The G-Spectral Estimator. Journal of the American Statistical Association, Vol. 79, No. 387, Theory and Methods Section, p.p. 692-701.

[8] Box, G. E., Jenkins, G. M. (1967). Time Series Analysis Forecasting and Control, San Francisco: Holden day.

APPENDIXES

APPENDIX A

TYPE DECLARATION

Table A.1 window record structure

```
integer*1  windowrecord(156)
integer*2  grafport(54)
integer*2  windowkind
logical*1  visible
logical*1  hilite
logical*1  goawayflag
logical*1  spareflag
integer*4  strucrgn
integer*4  contrgn
integer*4  updatergn
integer*4  windowdefproc
integer*4  datahandle
integer*4  titlehandle
integer*2  titlewidth
integer*4  controllist
integer*4  nextwindow
integer*4  windowpic
integer*4  refcon
equivalence  (windowrecord(1),grafport)
equivalence  (windowrecord(109),windowkind)
equivalence  (windowrecord(111),visible)
equivalence  (windowrecord(112),hilite)
equivalence  (windowrecord(113),goawayflag)
equivalence  (windowrecord(114),spareflag)
equivalence  (windowrecord(115),strucrgn)
equivalence  (windowrecord(119),contrgn)
equivalence  (windowrecord(123),updatergn)
equivalence  (windowrecord(127),windowdefproc)
equivalence  (windowrecord(131),datahandle)
equivalence  (windowrecord(135),titlehandle)
equivalence  (windowrecord(139),titlewidth)
equivalence  (windowrecord(141),controllist)
equivalence  (windowrecord(145),nextwindow)
equivalence  (windowrecord(149),windowpic)
equivalence  (windowrecord(153),refcon)
```

Table A.2 Grafport structure

```
integer*2  grafport(54)
integer*2  device
integer*2  portbits(7)
integer*2  portrect(4)
integer*4  visrgn
integer*4  cliprgn
integer*1  bkpat(8)
integer*1  fillpat(8)
integer*2  pnloc(2)
integer*2  pnsize(2)
integer*2  pnmode
integer*1  pnpat(8)
integer*2  pnvis
integer*2  txfont
integer*2  txface
integer*2  txmode
integer*2  txsize
integer*4  spextra
integer*4  fgcolor
integer*4  bkcolor
integer*2  colrbit
integer*2  patstretch
integer*4  picsave
integer*4  rgnsave
integer*4  polysave
integer*4  grafprocs
equivalence  (grafport(1),device)
equivalence  (grafport(2),portbits(1))
equivalence  (grafport(9),portrect(1))
equivalence  (grafport(13),visrgn)
equivalence  (grafport(15),cliprgn)
equivalence  (grafport(17),bkpat(1))
equivalence  (grafport(21),fillpat(1))
equivalence  (grafport(25),pnloc(1))
equivalence  (grafport(27),pnsize(1))
equivalence  (grafport(29),pnmode)
equivalence  (grafport(30),pnpat(1))
equivalence  (grafport(34),pnvis)
equivalence  (grafport(35),txfont)
equivalence  (grafport(36),txface)
equivalence  (grafport(37),txmode)
```

```
equivalence   (grafport(38),txsize)
equivalence   (grafport(39),spextra)
equivalence   (grafport(41),fgcolor)
equivalence   (grafport(43),bkcolor)
equivalence   (grafport(45),colrbit)
equivalence   (grafport(46),patstretch)
equivalence   (grafport(47),picsave)
equivalence   (grafport(49),rgnsave)
equivalence   (grafport(51),polysave)
equivalence   (grafport(53),grafprocs)
```

Table A.3 Event Record Structure

```
integer*2  eventrecord(8)     ! overlying structure

    integer*2  what           ! type of event:
                              ! 0 = null
                              ! 1 = mouse down
                              ! 2 = mouse up
                              ! 3 = key down
                              ! 4 = key up
                              ! 5 = auto-key
                              ! 6 = update
                              ! 7 = disk inserted
                              ! 8 = activate
                              ! 9 = abort
                              ! 11 = I/O driver
                              ! 12 = application defined
                              ! 13 = application defined
                              ! 14 = application defined
                              ! 15 = application defined
    integer*4  message        ! extra event information:
                              ! keyboard: ASCII code in bits 0-7
                              ! disk inserted: drive number
                              ! Activate and deactive: window
                              pointer
                              ! Mouse: undefined
    integer*4  when           ! time of event in 60ths of seconds
    integer*2  where(2)       ! mouse location in global
                              coordinates
    integer*2  modifiers      ! state of mouse button and
                              modifier keys:
                              !     1 = activate/deactivate
                              !     2 = System/application
                              window
                              !     4 = unused
                              !     8 = unused
                              !    16 = unused
                              !    32 = unused
                              !    64 = unused
                              !   128 = mouse button
                              !   256 = Command key
                              !   512 = Shift key
                              !  1024 = Caps Lock key
```

```
                          2048 = Option key
                        !  4096 = unused
                        !  8192 = unused
                        ! 16384 = unused
                        ! 32768 = unused

equivalence  (eventrecord(1),what)
equivalence  (eventrecord(2),message)
equivalence  (eventrecord(4),when)
equivalence  (eventrecord(6),where(1))
equivalence  (eventrecord(8),modifiers)
```

Table A.4 eventmask values

-1 = all events
0 = null
1 = mouse down
2 = mouse up
3 = key down
4 = key up
5 = auto-key
6 = update
7 = disk inserted
8 = activate
9 = abort
10 = network
11 = I/O driver
12 = application defined
13 = application defined
14 = application defined
15 = application defined

Table A.5 mouseloc interpretation

```
integer mouseloc        ! mouse location from FINDWINDOW:
                        !   0 = none of the following
                        !   1 = in the menu bar
                        !   2 = in system window
                        !   3 = in content region
                        !   4 = in drag region
                        !   5 = in grow region
                        !   6 = in go away region
```

APPENDIX B

PROGRAM CODE LISTINGS

```
************************************************************************
*
* MacModel:  The main program.  It contains the initializations and the
*            event loop.
*
************************************************************************

      program MacModel

      implicit none                  ! helps keep us out of trouble

      include  hd:include files:Estm-menu.inc

*
*  Pointer for creating the menus
*

      integer*4 menuhandle(10)       ! pointer to menu
      integer*4 menuhandle1(2)


*
*  The following structure defines an event record for making calls to
*  theMacintosh event manager. Events are occurances such as the
*  detection ofa change in the state of the mouse button, capturing
*  keyboard input, anddetecting the insertion of a disk. You can specify
*  the specific eventsthat your application is interested in handling.
*
      integer*4 eventmask            ! specifies the events of interest
c                            !     1 = null
c                            !     2 = mouse down
c                            !     4 = mouse up
c                            !     8 = key down
c                            !    16 = key up
c                            !    32 = auto-key
c                            !    64 = update
c                            !   128 = disk inserted
c                            !   256 = activate
c                            !   512 = abort
c                            !  1024 = network
c                            !  2048 = I/O driver
c                            !  4096 = application defined
c                            !  8192 = application defined
c                            ! 16384 = application defined
c                            ! 32768 = application defined

      integer*2 eventrecord(8)       ! overlying structure

      integer*2 what                 ! type of event:
c                            !  0 = null
c                            !  1 = mouse down
c                            !  2 = mouse up
c                            '  3 = key down
c                            !  4 = key up
c                            '  5 = auto-key
c                            !  6 = update
c                            !  7 = disk inserted
c                            !  8 = activate
```

```
c                               !  9 = abort
c                               ! 10 = network
c                               ! 11 = I/O driver
c                               ! 12 = application defined
c                               ! 13 = application defined
c                               ! 14 = application defined
c                               ! 15 = application defined
      integer*4 message         ! extra event information:
c                               !  keyboard: ASCII code in bits 0-7
c                               !  disk inserted: drive number
c                               !  Activate and deactive: window pointer
c                               !  Mouse: undefined
      integer*4 when            ! time of event in 60ths of seconds
      integer*2 where(2)        ! mouse location in global coordinates
      integer*2 modifiers       ! state of mouse button and modifier keys:
c                               !     1 = activate/deactivate
c                               !     2 = System/application window
c                               !     4 = unused
c                               !     8 = unused
c                               !    16 = unused
c                               !    32 = unused
c                               !    64 = unused
c                               !   128 = mouse button
c                               !   256 = Command key
c                               !   512 = Shift key
c                               !  1024 = Caps Lock key
c                               !  2048 = Option key
c                               !  4096 = unused
c                               !  8192 = unused
c                               ! 16384 = unused
c                               ! 32768 = unused

      equivalence (eventrecord(1),what)
      equivalence (eventrecord(2),message)
      equivalence (eventrecord(4),when)
      equivalence (eventrecord(6),where(1))
      equivalence (eventrecord(8),modifiers)
      integer*4 flag


*
*  The following structure defines a window record for making calls to
*  theMacintosh window manager. The two windows defined by this program
*  willnot need to manipulate the window record, but the structure is
*  definedfor reference
*

      integer*4 tty_window          ! pointer to the tty window
      integer*4 plot_window         ! pointer to the graphics window
      integer*4 dialog_win          ! pointer to the dialog window
      integer*4 dialog_win1
      integer*4 plot_scrlbr
      common  /win/ tty_window,plot_window

      integer*1 tty_record(156)     ! tty window record
      integer*1 plot_record(156)    ! graphics window record
      integer*1 dialog_recd(170)    ! dialog window record
      integer*2 portrect(4)         ! to store the plot window size
```

```
      equivalence(plot_record(17),portrect(1))

      integer*2 port(54)         ! grafport
      integer*2 windowkind       ! type of window:
c                                !   0 = standard document window
c                                !   1 = alert box or modal dialog box
c                                !   2 = plain box
c                                !   3 = plain box with shadow
c                                !   4 = document window without size box
c                                !  10 = round cornered window
      logical*1 visible          ! true when window is visible
      logical*1 hilite           ! highlighted when true
      logical*1 goaway           ! has a go away box when true
      logical*1 spareflag        ! reserved
      integer*4 strucrgn         ! Quickdraw region
      integer*4 contrgn          ! Quickdraw region
      integer*4 updatergn        ! Quickdraw region
      integer*4 windowdefproc ! pointer to window definition function
      integer*4 datahandle       ! pointer for window definition function
      integer*4 titlehandle      ! pointer to title
      integer*4 titlewidth       ! width (in pixels) of title
      integer*4 controllist      ! pointer to control list for Control Mgr
      integer*4 nextwindow       ! pointer to next window in window list
      integer*4 windowpic        ! pointer to Quickdraw picture of contents
      integer*4 refcon           ! reference value field


      equivalence (tty_record(1),port)
      equivalence (tty_record(104),windowkind)
      equivalence (tty_record(109),visible)
      equivalence (tty_record(110),hilite)
      equivalence (tty_record(111),goaway)
      equivalence (tty_record(112),spareflag)
      equivalence (tty_record(113),strucrgn)
      equivalence (tty_record(117),contrgn)
      equivalence (tty_record(121),updatergn)
      equivalence (tty_record(125),windowdefproc)
      equivalence (tty_record(129),datahandle)
      equivalence (tty_record(133),titlehandle)
      equivalence (tty_record(137),titlewidth)
      equivalence (tty_record(139),controllist)
      equivalence (tty_record(143),nextwindow)
      equivalence (tty_record(147),windowpic)
      equivalence (tty_record(151),refcon)


*
*  Miscellaneous declarations
*

      integer*4 toolbx           ! the tool box interface
      character*256 str255       ! function to create a Pascal LSTRING
      integer*4 window           ! general purpose pointer
      integer*4 size,w,h,w1,h1         ! for growing windows
      integer*2 rect(4), rect1(4)    ! rectangle coordinates
      integer mouseloc           ! mouse location from FINDWINDOW:
```

```
c                              !   0 = none of the following
c                              !   1 = in the menu bar
c                              !   2 = in system window
c                              !   3 = in content region
c                              !   4 = in drag region
c                              !   5 = in grow region
c                              !   6 = in go away region

      logical   cnclflag
      real  dum
      integer*4 proc
      character*64 char_mtrx(3)


      integer*4 mypict, item_ptr,item_list(2)
      integer*4 itemhit, error
      integer*4 appllimit      ! address of current aplication limit
      integer*4 newlimit       ! new application limit
      parameter (appllimit=z'00000130')

      newlimit = LONG(appllimit)    ! get the current limit
      newlimit = newlimit-102400    !allocate an additional 100k of stack
      call toolbx(SETAPPLLIMIT,newlimit) !set the new application limit

*
*  Set up the event manager mask (you should accept responsibility for
*  allevents to insure that the event queue is flushed; some calls such
*  as MENUSELECT will not work properly if there are extra mouse up
*  events lying around):
*
      eventmask = -1

*
*  Close MacFortran I/O window (never make a DISPOSEWINDOW call on this
*  window):
*

      window = toolbx(FRONTWINDOW)
      call toolbx(CLOSEWINDOW,window)
      window = toolbx(FRONTWINDOW)
      call toolbx(HILITEWINDOW,window,.false.)
*
*  Build the menus (for details refer to "Menu Initialization and
*  Allocation" in "The MacFortran Tool Box Interface"):
*

*  Most Toolbox initialization required is done by runtime.  Only
*  Text Edit needs to be initialized here.  Also open resource file.

      call toolbx(TEINIT)
      error=toolbx(OPENRESFILE,char(8) // 'dlog.res')
      if (error=-1) write(9,*) 'Resource file busy or missing'

      menuhandle(1) = toolbx(NEWMENU,29,char(1) // char(z'14'))

c Setup the apple menu
      call toolbx(APPENDMENU,menuhandle(1),
     +              str255("About MacModel...;(-"))
```

```
      call toolbx(ADDRESMENU,menuhandle(1), 'DRVR')
      call toolbx(INSERTMENU,menuhandle(1),0)     ! insert at end of list

c Setup the File menu
      menuhandle(2) = toolbx(NEWMENU,30,str255("File"))
      call toolbx(APPENDMENU,menuhandle(2),str255("List.../L;
     +Plot.../P;Print Text...;(Print Graphics...;Transfer...;Quit/Q"))
      call toolbx(INSERTMENU,menuhandle(2),0)     ! insert at end of list

c Setup the Edit menu
      menuhandle(3) = toolbx(NEWMENU,31,str255("Edit"))
      call toolbx(APPENDMENU,menuhandle(3),
     +                str255("Undo/Z;(-;Cut/X;Copy/C;
     +Paste;Clear"))
      call toolbx(INSERTMENU,menuhandle(3),0)     ! insert at end of list

c Setup the Simulation menu and its submenu
      menuhandle(4) = toolbx(NEWMENU,32,str255("Simulate"))
      call toolbx(APPENDMENU,menuhandle(4),
     +     str255("!AWhite Noise/ ;ARMA..."))
      call toolbx(INSERTMENU,menuhandle(4),0)     ! insert at end of list

c 65 (decimal)=41 hex= A ASCII
      menuhandle(5) = toolbx(NEWMENU,65,str255("White Noise"))
                                             ! 65 decimal
      call toolbx(APPENDMENU,menuhandle(5),
     +     str255("Gaussian...;Uniform..."))
      call toolbx(INSERTMENU,menuhandle(5),-1)    ! -1 Hierarchial menu

c Setup the Correlation menu
      menuhandle(6) = toolbx(NEWMENU,33,str255("Identify"))
      call toolbx(APPENDMENU,menuhandle(6),
     +      str255("Autocorrelation.../A;Crosscorrelation...;
     +Partial Autocorrelation...;(-;GPAC.../G;S-array...;
     +R-array.../R"))
      call toolbx(INSERTMENU,menuhandle(6),0)     ! insert at end of list

c Setup the Estimation menu
      menuhandle(7) = toolbx(NEWMENU,34,str255("Estimate"))
      call toolbx(APPENDMENU,menuhandle(7),
     +                str255("Maximun Likelihood..."))
      call toolbx(INSERTMENU,menuhandle(7),0)     ! insert at end of list

      call toolbx(DRAWMENUBAR)
*
*  Create the windows, only display the TTY window; the SETPORT call
*  allows I/O to take place:
*

      tty_window = toolbx(PTR,tty_record)
      rect(1) = 40
      rect(2) = 3
      rect(3) = 338
      rect(4) = 503
      tty_window = toolbx(NEWWINDOW,tty_window,rect,
     +                 str255("TTY"),.false.,0,-1,.true.,0)
      call toolbx(SETPORT,tty_window)
      call toolbx(TEXTFONT,4)         ! monaco
```

```
      call toolbx(TEXTSIZE,9)        ! 9 point

      plot_window = toolbx(PTR,plot_record)
      rect(1) = 40
      rect(2) = 3
      rect(3) = 334
      rect(4) = 507
      plot_window = toolbx(NEWWINDOW,plot_window,rect,
     +            str255("Plot"),.false.,0,tty_window,.true.,0)
        call toolbx(SETPORT,plot_window)
      call toolbx(TEXTFONT,4)        ! monaco
      call toolbx(TEXTSIZE,9)        ! 9 point

c----------------------------------------------------------------------
c  main event processing loop
c  constraints on window dragging:
c----------------------------------------------------------------------
      rect(1) = 25
      rect(2) = 25
      rect(3) = 300
      rect(4) = 500
      mypict = 0

        do
          if (toolbx(GETNEXTEVENT,eventmask,eventrecord)) then

            select case (what)

              case (1)          ! mouse down

              mouseloc = toolbx(FINDWINDOW,where,window)

                if (mouseloc=1) then
                  flag=0
                  call menus(where,portrect,menuhandle(1),
     &                  port,mypict,flag,message)
                  if (mypict<>0) call toolbx(ENABLEITEM,
     +                      menuhandle(2),4) !enables print grapics menu


                else if (mouseloc=3) then
                  call toolbx(SELECTWINDOW,window)

                else if (mouseloc=4) then
                  call toolbx(DRAGWINDOW,window,where,rect)

                else if (mouseloc=5) then
                  h1 = portrect(3)-portrect(1)   !height
                  w1 = portrect(4)-portrect(2)   'width
                  size = toolbx(GROWWINDOW,window,where,rect)
                  w = size .and. z'ffff'   'high-word of size is the
                  h = shift(size,-16)      ' vertical measurements,
                                           ! horizontalis in the low.
                  if ((h1.NE.h).OR.(w1.NE.w)) then
                  call toolbx(SIZEWINDOW,window,w,h,.true.)

                  if (window=plot_window).and.(mypict<>0) then
                    call toolbx(SETPORT,plot_window)
```

```
                call clrwin(plot_window)
                rect1(1) = 0
                rect1(2) = 0
                rect1(3) = h
                rect1(4) = w
                call toolbx(DRAWPICTURE,mypict,rect1)
                call toolbx(SETPORT,tty_window)
              endif
                    endif
            else if (mouseloc=6) then
              if (toolbx(TRACKGOAWAY,window,where))
+                       call toolbx(HIDEWINDOW,window)
            end if

          case(3)                 !key down
           if (modifiers=384 .or. modifiers=1408) then
                                  ! 384=command,1408=caps lock command
            flag=1
            call menus(where,portrect,menuhandle(1),
&                   port,mypict,flag,message)
          end if
        case default

        end select

      end if
    repeat

  end
```

```
************************************************************************
*
* menus: a mouse down event was detected in the menu area; process menu
*      selection
*
************************************************************************

      subroutine menus(where,portrect,menuhandle,port,array_pict,
     +                   flag,message)

      implicit none

      include hd:include files:Estm-menu.inc

      integer*4 toolbx,array_pict,flag,message
      integer*4 error


      integer*4 tty_window          ! tty window pointer
      integer*4 plot_window         ! plot window pointer
      common  /win/ tty_window, plot_window !, pic_window

      integer   menuhandle, npts
      integer*4 menuhandle1(2)
      integer*2 where(2)       ! mouse location from the event record

      integer*2 port(54)       ! grafport
      integer*2 portrect(4)
      integer*4 mywindow, proc

*
*  variables for making menu selections
*
      integer*2 menuselection(2)    ! menu selection information
c                                   !  (1) = menuid
c                                   !  (2) = menu item number
      integer*4 menudata            ! for use left of equals sign
      integer*4 mnusel4             ! To convert menuselection(2)
                                    ! to 4 bytes
      character*20  itemname        ! to store the item name
      integer   refnum

      equivalence (menuselection,menudata)

*
*  Menu selection constants:
*

      integer Apple, Edit, File,aboutMM  ! menus
      integer Simulate, Identify         ! menus
      integer Estimate             ! menus
      integer List,Plot,Print,Printg         ! "File" menu selections
      integer Transfer,Quit
      integer White,G_Noise,U_Noise,ARMA ! "Simulate" menu selections
      integer ACF,CCF,PACF               ! "Identify" menu selections
      integer GPAC,S_array,R_array
      integer MLE                        ! "Estimate" menu selection
```

```
      parameter (Apple=29, File=30, Edit=31, Simulate=32)
      parameter (aboutMM=1)
      parameter (Identify=33, Estimate=34)
      parameter (List=1, Plot=2,Print=3, Printg=4, Transfer=5, Quit=6)
      parameter (White=65,ARMA=2)
      parameter (G_Noise=1,U_Noise=2)
      parameter (ACF=1, CCF=2,PACF=3)     ! 4 is the separating line
      parameter (GPAC=5,S_array=6,R_array=7)
      parameter (MLE=1)
      integer   i,j
      logical   editflag,doneflag,cnclflag,exist
      real  dum
      character opt*4
      character*64 filename,char_mtrx(3)

c-----------------------------------------------------------------------
*  The MENUSELECT tool box call handles the messy details of
highlighting
*  menus and menu selections, pulling menus down, and determining which
*  item was selected. The HILITEMENU call at the end of the select case
*  blocks actually unhighlights the menus.
c-----------------------------------------------------------------------
      if (flag.eq.1)  then
          opt='    '//char(message)
        menudata=toolbx(MENUKEY,opt)
      else
         menudata = toolbx(MENUSELECT,where)
      endif
      mnusel4= menuselection(2)             ! convert to 4 bytes
        select case (menuselection(1))

          case (Apple)                      ! AppleMenu
            call clrwin(tty_window)
            if(menuselection(2).EQ. aboutMM) then
               proc = 3
             dum=0
                 call dlog(8200,0,0,dum,0,0,0,menuhandle1,
     +                       cnclflag,proc,char_mtrx)
            else
               call toolbx(GETITEM, menuhandle,
     +                              mnusel4, itemname)
                 refnum = toolbx(OPENDSKACC, itemname)
              endif

          case (File)                      ! the "File" menu was selected

              select case (menuselection(2))

              case (List)
                call clrwin(tty_window)
                call list(tty_window)

              case (Plot)
                call toolbx(SETPORT,plot_window)
                call clrwin(plot_window)
                call subplt(array_pict,portrect,0,npts)
                call toolbx(SETPORT,tty_window) ! because it was set to
                call setcur(0)! plot_window in plot subroutine
```

```
    case (Print)
      call print

    case (Printg)
       rect(1)=word(plot_window+16)
       rect(2)=word(plot_window+18)
       rect(3)=word(plot_window+20)
       rect(4)=word(plot_window+22)
       call toolbx(SETPORT,plot_window)
         call toolbx(DRAWPICTURE, array_pict, rect)
       call printg(array_pict)
       call toolbx(SETPORT,tty_window)

    case (Transfer)
      call transfer

    case (Quit)
      inquire (file='MacModel.scratch',exist=exist)
      if (exist) then
        open(1,file='MacModel.scratch',status='old')
          close(1,status='DELETE')
      endif
      stop

  end select

case (Edit)

    editflag = toolbx(SYSTEMEDIT,mnusel4-1)

case (Simulate)            ! the "Simulation" menu was selected
   select case (menuselection(2))
    case(ARMA)
      call clrwin(plot_window)
      call doarma(array_pict,portrect)
      call setcur(0)
      call toolbx(SETPORT,tty_window)
    end select

case (White)          ! the "White Noise" menu was selected
   select case (menuselection(2))
    case (G_Noise)
      call clrwin(plot_window)
      call dowhit(0,array_pict,portrect)     !0: gaussian noise
    case (U_Noise)
      call clrwin(plot_window)
      call dowhit(1,array_pict,portrect)     !1: uniform noise
  end select
  call setcur(0)
  call toolbx(SETPORT,tty_window)
case (Identify)      ! the "Correlation" menu was selected

    select case (menuselection(2))

    case (ACF)
      call clrwin(plot_window)
      call corlat(1,array_pict,portrect)
```

```
          call toolbx(SETPORT,tty_window)

        case (CCF)
          call clrwin(plot_window)
          call corlat(2,array_pict,portrect)
          call toolbx(SETPORT,tty_window)

        case (PACF)
          call clrwin(plot_window)
          call pacf(array_pict,portrect)
          call toolbx(SETPORT,tty_window)

       case (GPAC)
          call clrwin(plot_window)
          call gpac(1,array_pict,portrect)
            call toolbx(DRAWPICTURE, array_pict, portrect)
          call toolbx(SETPORT,tty_window)

        case (S_array)
          call clrwin(plot_window)
          call gpac(2,array_pict,portrect)
            call toolbx(DRAWPICTURE, array_pict, portrect)
          call toolbx(SETPORT,tty_window)

        case (R_array)
          call clrwin(plot_window)
          call gpac(3,array_pict,portrect)
            call toolbx(DRAWPICTURE, array_pict, portrect)
          call toolbx(SETPORT,tty_window)

       end select


    case (Estimate)             ! "Estimate" menu was selected

        select case (menuselection(2))

        case (MLE)
          call clrwin(tty_window)
          call MLE

       end select

    case default            ! just playing with the mouse

  end select
  call toolbx(HILITEMENU,0)
end
```

```
***********************************************************************
*
* dlog:  Manages all the dialog boxes with the exception of the dialog
*        that adjusts a and b.
*
***********************************************************************

c w= the dlog id#
c esize= number of edit text fields
c ssize= number of static text fields
c array= data in edit fields
c okbutton= the item number of the ok button
c cancelbutton= the item number of the cancel button
c offset item number where the edit field starts  ie. if ok button is
c item #1 and cancelbutton is item #2 then the offset is 2.
c NOTE: always have the ok and cancel buttons the first buttons used, if
c you don't want to have either of these buttons to be a default, then
c make a "dummy" item #1and have the ok and cancel #2 and #3.  Or if you
c don't want to have a cancel buttonand you don't want the ok to be the
c default  button then make a "dummy" item #1 and set the cancelbutton
c to 0 and start the ok at item #2.
c userdef is a defined variable that contains misc information.
c nmbr is the array number 1=dim,2=oper,3=fri,4=val
c cnclflag returns false if the user wants to terminate the function
c proc is a flag that to indicate the calling procedure
c char_mtrx analogous to mtrx but for characters instead of real is
c used in plot.
c proc=0 calling routine is anyone except the listed below
c proc=1 calling routine is plotgsr
c proc=2 calling routine is plot
c proc=3 calling routine is menus
c proc=4 calling routine is estm_menu
c proc=5 calling routine is doarma
c proc=6 calling routine is dowhite
c proc=7 calling routine is corlat
c proc=8 calling routine is pacf
c proc=9 calling routine is doarma (p & q dialog box)
c proc=10 calling routine is plotgsr
c proc=11 calling routine is gpac (second time only -alert-)
c proc=12 calling routine is mle

      subroutine dlog(w,esize,ssize,aray,okbutton,cancelbutton,offset,
     &      userdef,cnclflag,proc,char_mtrx)
      implicit none
      include hd:include files:utilities.inc
      integer*2 rect(4),itemhit,itemtype,wrect(4)
      integer*4 disable
      parameter(disable=128)
      integer*4 j,i,offset,esize,ssize,pd,w,flag
      integer*4 hte,er,toolbx
      integer*4 okbutton, cancelbutton,longhit,window
      integer*4 userdef(2)
      integer*4 sigmoid,linear,proc,rdio_chk,check3,check4,check5
      parameter (sigmoid=3, linear=4)
      logical doneflag, cnclflag
      character*256 str255,frchar,powr,edfld(100),blnk,static(100)
      character*64 char_mtrx(*), temp
      real aray(*),freal
```

```
      data wrect /0,0,470,630/
      doneflag = .false.

      call toolbx(INITDIALOGS,0)
      window=toolbx(FRONTWINDOW)
      pd = toolbx(GETNEWDIALOG,w,0,-1)
      call toolbx(SETPORT, pd)
c Disable the unneeded edit fields if proc is doarma
      if (proc=5) then
         do I= userdef(1)+offset+1,12+offset
                                              !inhibit the AR edit fields
            call toolbx(GETDITEM,pd,i,itemtype,hte,rect)
              call toolbx(SETDITEM,pd,i,8+disable,hte,rect)
                                              !8: is  static field
         enddo
         do I= userdef(2)+offset+1+12,24+offset
                                              !inhibit the MA edit fields
            call toolbx(GETDITEM,pd,i,itemtype,hte,rect)
              call toolbx(SETDITEM,pd,i,8+128,hte,rect)
         enddo
        endif

      call toolbx(SHOWWINDOW,pd)

c Inserting bolding around 'ok' button
      if (okbutton<>-1) then
         call toolbx(GETDITEM,pd,okbutton,itemtype,hte,rect)
         call toolbx(PENSIZE,3,3)
         call toolbx(INSETRECT,rect,-4,-4)
         call toolbx(FRAMEROUNDRECT,rect,16,16)
      endif
      do 5 i=1,32
5           blnk=trim(blnk)//'          '
      do 10 i=1,esize
10      edfld(i)='          '

c Get handles to static text fields and extract strings
      do 15 i=1,ssize
           call toolbx(GETDITEM,pd,esize+offset+i,itemtype,hte,rect)
           call toolbx(GETITEXT,hte,static(esize+offset+i))
15    continue

c Add the numbers (actually they are characters to the edit text fields
c if the calling subroutine is PLOT or PLRGSR.
      if (proc=2).OR.(proc=10) then
         do 20 i=1,esize
            powr=str255(frchar(aray(i),'1PG9.3'))
            call toolbx(GETDITEM,pd,offset+i,itemtype,hte,rect)
            call toolbx(SETITEXT,hte,powr)
            powr=blnk
20       continue
      endif

c items 3,4,& 5 are the text items (plot labels) in plot dialog box
      if (proc=2) then
         do  i=1,3
            call toolbx(GETDITEM,pd,i+2,itemtype,hte,rect)
            call toolbx(SETITEXT,hte,char_mtrx(i))
```

```
         enddo
      endif
c item # 2 in plotgsr is a text item (array title).
      if (proc=10) then
            call toolbx(GETDITEM,pd,3,itemtype,hte,rect)
            call toolbx(SETITEXT,hte,char_mtrx(1))
      endif
c add the text items of mle report(results of mle routine).
      if (proc=12) then
          do i= 1,2           !char_mtrx(10:11) are lables for edit text
            call toolbx(GETDITEM,pd,i+5,itemtype,hte,rect)
            call toolbx(SETITEXT,hte,str255(char_mtrx(i+9)))
         enddo
         do i= 1, 9
            call toolbx(GETDITEM,pd,i+13,itemtype,hte,rect)
            call toolbx(SETITEXT,hte,str255(char_mtrx(i)))
         enddo
      endif
c Put the character strings back into the static text fields
      do 25 i=1,ssize
          call toolbx(GETDITEM,pd,esize+offset+i,itemtype,hte,rect)
          call toolbx(SETITEXT,hte,static(esize+offset+i))
25    continue
      if (proc=10) then
         write(temp,83) aray(3), aray(4)
83       format('Array STD: ',G9.4,'  Array mean: ',f8.3)
           call toolbx(GETDITEM,pd,13,itemtype,hte,rect)
         call toolbx(SETITEXT,hte,str255(temp))
        endif
      if (proc=11) then
         call toolbx(PARAMTEXT,str255(char_mtrx(1)),
     +                 str255(char_mtrx(2)),str255(char_mtrx(3)),
     +                                     str255(char_mtrx(4)));
        endif

      if (proc=6).OR.(proc=7).OR.(proc=8).OR.(proc=9) then
         call toolbx(GETDITEM,pd,3,itemtype,hte,rect)
         call toolbx(SETCTLVALUE,hte,1)
         rdio_chk = 1          !1 :plot sequence
      endif
      check3=0;   check4=0;   check5=0;

      if (esize<>0) then   !if there is edit fields, highlight the first
         if ((proc=5).and.(userdef(1)=0)) then  !skip the AR edit fields
            call toolbx(SELITEXT,pd,offset+12+1,0,10)
         else
            call toolbx(SELITEXT,pd,offset+1,0,10)
         endif
      endif
      if (proc=12) then ' highlight first edit field if proc is mle
         call toolbx(SELITEXT,pd,7,0,10)
      endif

c Main loop
      do

c Modaldialog handles all events that the user makes.
c Itemhit is the item number from ResEdit- there will be "offset" number
```

```
of items
      call toolbx(MODALDIALOG, 0, itemhit)
         if (proc=3) doneflag=.true.      ! About menu
         longhit = itemhit          ! 4 byte version of itemhit

         if (longhit .EQ. okbutton) then
            call extract(pd,offset,esize,edfld,aray)
            if (proc=2) then
               do  i=1,3           !for edit fields with text arguments
                  call toolbx(GETDITEM,pd,i+2,itemtype,hte,rect)
                  call toolbx(GETITEXT,hte,char_mtrx(i))
                 enddo
            endif
            if (proc=10) then      !for edit fields with text arguments
             call toolbx(GETDITEM,pd,3,itemtype,hte,rect)
             call toolbx(GETITEXT,hte,char_mtrx(1))
            endif
            if (proc=12) then
               do  i=1,2           !for edit fields with text arguments
                  call toolbx(GETDITEM,pd,i+5,itemtype,hte,rect)
                  call toolbx(GETITEXT,hte,char_mtrx(i+9))
               enddo
            endif
            doneflag=.true.
            cnclflag=.true.

         elseif (longhit .EQ.sigmoid).AND.(proc=1) then
            rdio_chk=sigmoid
            doneflag=.true.
            cnclflag=.true.

         elseif (longhit .EQ.linear).AND.(proc=1) then
            rdio_chk=linear
            doneflag=.true.
            cnclflag=.true.

         elseif (longhit .EQ. cancelbutton) then
            doneflag=.true.
            cnclflag=.false.

           elseif (longhit .EQ.3).AND.((proc=6).OR.
     +                  (proc=7).OR.(proc=8).OR.(proc=9))  then
            call toolbx(GETDITEM,pd,3,itemtype, hte,rect)
              if (rdio_chk=1) then
               call toolbx(SETCTLVALUE,hte,0)
             rdio_chk=0
            elseif (rdio_chk=0) then
               call toolbx(SETCTLVALUE,hte,1)
             rdio_chk=1
            endif

        elseif ((longhit=3).OR.(longhit=4).OR.(longhit=5)).AND.
     +               (proc=12) then
            call toolbx(GETDITEM,pd,longhit,itemtype, hte,rect)
              if (longhit=3) then
               if (check3=1) then
                  call toolbx(SETCTLVALUE,hte,0)
                check3=0
```

```
               elseif (check3=0) then
                  call toolbx(SETCTLVALUE,hte,1)
                check3=1
               endif
            elseif (longhit=4) then
             if (check4=2) then
                  call toolbx(SETCTLVALUE,hte,0)
                check4=0
               elseif (check4=0) then
                  call toolbx(SETCTLVALUE,hte,1)
                check4=2
               endif
            elseif (longhit=5) then
             if (check5=4) then
                  call toolbx(SETCTLVALUE,hte,0)
                check5=0
               elseif (check5=0) then
                  call toolbx(SETCTLVALUE,hte,1)
                check5=4
               endif
            endif
         endif
         if (doneflag) exit
        repeat                    ! Next dialog item.
        if (proc=12) then
           proc = check3+check4+check5
        else
           proc=rdio_chk
        endif
c Closes the dialog window and redisplays the main window
        call toolbx(DISPOSEDIALOG,pd)
        call toolbx(SETPORT,window)
        call toolbx(SELECTWINDOW,window)
      return
      end

c-------------------------------------------------------------------------
c Extract the characters from the edit fields
c-------------------------------------------------------------------------
      subroutine extract(pd,offset,esize,edfld,aray)
      implicit none
      include hd:include files:utilities.inc
      integer*2 rect(4),itemtype
      integer*4 i,esize,pd,offset,toolbx,hte
      character*256 str255,frchar,powr,edfld(100),blnk,static(100)
      real aray(*),freal

c Get handles to edit text fields and extract data from fields
      do 10 i=1,esize
         call toolbx(GETDITEM,pd,offset+i,itemtype,hte,rect)
         call toolbx(GETITEXT,hte,edfld(i))
10    continue

c Subroutine sort takes character data from the text edit fields and
c sorts them into a real array.
      call sortr(esize,edfld,aray)

      return
```

```
      end

c-----------------------------------------------------------------------
c Sorting routine to put the character array "edfld" into a real array
"aray"
c-----------------------------------------------------------------------
      subroutine sortr(esize,edfld,aray)
      implicit none
      integer*4 i,esize
      character*256 edfld(*),dl
      real freal,aray(*)

      do 10 i=1,esize
        dl=edfld(i)
        aray(i)=freal(dl(2:11))
        edfld(i)='          '
 10   continue
      return
      end
c-----------------------------------------------------------------------
c     This routine converts numeric data to character type
c     given the following information:
c
c-----------------------------------------------------------------------
        function frchar(qreal,qtype)
      implicit none
      integer*4 err
      character*(*) frchar
      character*(*) qtype
      character*16 temp
      character q1*9,q2*7
      real qreal

      if (qreal .LT. (1.0E-20).AND.qreal.GT.(-1.0E-20))then
          frchar=' '
      endif
c Set default conversion to F format
      q2= qtype
      q1='('//q2//')'

10    continue
      write(temp,q1,err=200)qreal
      frchar=temp
      return

200   continue
      frchar ='err in conv'
      return
      end
c-----------------------------------------------------------------------
c     This a function routine to convert ASCII data to a REAL format.
c-----------------------------------------------------------------------
      real function freal(temp)
      implicit none
      integer*4 i,j,k,fmt,err
      character*(*)temp
      character*20 ascii
```

```
      j=0
      k=0
      ascii='          '
      ascii=temp
c Find start and ending of number to convert
      do 20 i = 1, 20
         if(j.EQ.0.AND.ascii(i:i).NE.' ')j=i
         IF(j.NE.0.AND.ascii(i:i).EQ.' ')then
              k=i-1
              goto 40
           endif
20    continue

c Find decimal if one is not found put one there
40    continue
      if (k.EQ.0) goto 200
      do 50 i=j,k
         if(ascii(i:i).EQ.'.') goto 60
50    continue
      k=k+1
      ascii(k:k)='.'

c Internal read to convert character numbers to real data
60    continue
      read(ascii(j:k),fmt=100,err=200)freal
100   format(G10.4)
      goto 300

200     continue
      freal=0.0

300   continue
      return
      end
```

```
************************************************************************
*
* plt_gsr: Draws the grayscale map that corresponds to numbers in GPAC,
*          S, or R arrays.
*
************************************************************************

      subroutine plt_gsr(flag,array_pict,array,rows,columns,rect)
      implicit none

      integer flag
      integer*4 array_pict,toolbx
      integer ary_size,rows, columns,i,j,func_type
      parameter(ary_size=26)
      real theta,alpha
      real array(-2*ary_size:ary_size,0:ary_size), zero_num
      real nw_array(-2*ary_size:ary_size,0:ary_size)
      integer array_ptrn(0:2*ary_size,0:ary_size)
      character*64 filename,title
      character ch*2, str255*256
      integer*2 rect(4)
      logical cnclflag


      call ary_map(flag,cnclflag,func_type,rows,columns,title,
     +            alpha,theta,zero_num,array,nw_array)
      if (.NOT.cnclflag) return
      call num_to_ptrn(nw_array, array_ptrn, rows,
     +                    columns,zero_num)
      call pict_to_scrn(array_ptrn,rows, columns,title,
     +                     flag,array_pict,rect)
      call optmap(flag,func_type,rows,columns,alpha,theta,
     +              array_pict,array,rect,title)

      return
      end

************************************************************************
*
* ary_map: This routine apply the selected function on GPAC, S, or R
*          array entries and return the new array in nw_array
*
************************************************************************
      subroutine ary_map(flag,cnclflag,func_type,rows,columns,
     +              title,alpha,theta,zero_num,array,nw_array)
      implicit none

      integer*4 toolbx,menuhandle(2),flength
      integer flag, proc,sigmoid, limiter,func_type
      parameter(sigmoid=3,limiter=4)
      integer  g_pac, s_array, r_array
      parameter(g_pac=1, s_array=2, r_array=3)
      integer ary_size,rows, columns, i, j, n
        parameter(ary_size=26)
      real theta,alpha,ed_fld_data(4),zero_num,mean,var
      real array(-2*ary_size:ary_size,0:ary_size),dum
      real nw_array(-2*ary_size:ary_size,0:ary_size)
      character*64 char_mtrx(3),title
```

```
      character ch*2, str255*256
      logical cnclflag


c Calculate the array mean and variance
      n=0;   mean=0
      do i=1,columns
         do j=0,rows-1
            if array(j,i)>10 then
               dum=10
            elseif array(j,i)<-10 then
               dum=-10
            else
               dum=array(j,i)
            endif
            mean = mean + dum
            n = n + 1
         enddo
      enddo
      mean = mean/float(n)

      n=0;    var=0
      do i=1,columns
         do j=0,rows-1
            if array(j,i)>10 then
               dum=10
            elseif array(j,i)<-10 then
               dum=-10
            else
               dum=array(j,i)
            endif
            var = (dum - mean)**2 + var
            n = n + 1
         enddo
      enddo
      var = SQRT(var/float(n))               ! Standard deviation

      proc = 1                  !flag to indicate that pltsgr
                                !is the calling routine
      call dlog(5110,0,0,ed_fld_data,-1,2,4,menuhandle,
     +           cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) return
      func_type = proc  !proc is also returned as a flag
                        !to indicate which function (sigmoid
                        !or limiter) was selected

c set the title according to the array type
      if (flag=g_pac) then
         char_mtrx(1) = str255('GPAC')
      elseif (flag=s_array) then
         char_mtrx(1) = str255('S array')
      elseif(flag=r_array) then
         char_mtrx(1) = str255('R array')
      endif
      if (proc=sigmoid) then
         proc=10
         ed_fld_data(1) = 1/var
         ed_fld_data(2) = mean
```

```
         ed_fld_data(3) = var
         ed_fld_data(4) = mean
         call dlog(5120,2,3,ed_fld_data,1,2,3,menuhandle,
     +                cnclflag,proc,char_mtrx)
         if (.NOT.cnclflag) return
         alpha = ed_fld_data(1)
         theta  = ed_fld_data(2)
         title = char_mtrx(1)
         flength = ichar(title(1:1))
         title = title(2:flength+1)
         call lim_sigm(sigmoid,array,nw_array,rows,columns,
     +                      alpha,theta,zero_num)
      elseif (proc=limiter) then
         proc=10
         ed_fld_data(1) = mean-var
         ed_fld_data(2) = mean+var
         ed_fld_data(3) = var
         ed_fld_data(4) = mean
         call dlog(5130,2,3,ed_fld_data,1,2,3,menuhandle,
     +                cnclflag,proc,char_mtrx)
         if (.NOT.cnclflag) return
         alpha = ed_fld_data(1)
         theta  = ed_fld_data(2)
         title=char_mtrx(1)
         flength = ichar(title(1:1))
         title = title(2:flength+1)
         call lim_sigm(limiter,array,nw_array,rows,columns,
     +                      alpha,theta,zero_num)
      endif

      return
      end

************************************************************************
*
* num_to_ptrn: Changes GPAC, S, or R arrays entries to integer numbers
*              that correspond to certain patterns.
*
************************************************************************
      subroutine lim_sigm(flag,array,nw_array,rows,columns,
     +                      alpha,theta,zero_num)
      implicit none

      integer flag,ary_size,rows, columns, i, j
        parameter(ary_size=26)
      real theta,alpha,zero_num
      real array(-2*ary_size:ary_size,0:ary_size)
      real nw_array(-2*ary_size:ary_size,0:ary_size)


      if (flag=3) then  ! sigmoid
         do i=0, rows-1
            do j=1,columns
             nw_array(i,j) = 1/(1+exp(-alpha*(array(i,j)-theta)))
            enddo
         enddo
         zero_num = 1/(1+exp(-alpha*(0-theta))) ! # that corresponds
```

```
                                        ! to an actual zero
         return
      elseif (flag=4) then    ! limiter
         if (alpha=theta)  theta = theta +.1    ! Avoid zero division
         do i=0, rows-1
            do j=1,columns
               if (array(i,j)<min(alpha,theta)) then
                if (alpha>theta) then
                   nw_array(i,j)= 15
                else
                   nw_array(i,j)=1
                endif
              elseif (array(i,j)>max(theta,alpha)) then
                if (alpha>theta) then
                   nw_array(i,j)= 1
                else
                   nw_array(i,j)=15
                endif
              else
                  nw_array(i,j)=(14*(array(i,j)-alpha)/
     +                         (theta-alpha))+1
              endif
            enddo
         enddo
         zero_num=((-14*alpha)/(theta-alpha))+1     ! # that corresponds
                                         ! to an actual zero
         return
      endif
      end
***********************************************************************
*
* num_to_ptrn: Changes GPAC, S, or R arrays entries to integer numbers
*              that correspond to certain patterns.
*
***********************************************************************

      subroutine num_to_ptrn(array, array_ptrn, rows,
     +                        columns,zero_num)
      implicit none

      integer ary_size;      parameter(ary_size=26)
      real array(-2*ary_size:ary_size,0:ary_size)
      real  seg, max, min,zero_num
      integer array_ptrn(0:2*ary_size,0:ary_size)
      integer i, j, columns, rows,num_ptrn
      parameter(num_ptrn=15)


* Get maximum and minimum of array

      min = array(1,1)
      max = array(1,1)

      do j = 1, columns
         do i = 0, rows-1
            if (array(i,j)<min)  min = array(i,j)
            if (array(i,j)>max)  max = array(i,j)
         enddo
```

```
      enddo

      seg = float(max - min)/float(num_ptrn)

*
*------------------------------------------------------------
* Classify entries of array as one of 15 patterns
*------------------------------------------------------------

      do j = 1, columns
         do i = 0, rows-1
            if (array(i,j)>zero_num-seg/2.).AND.
     +                      (array(i,j)<zero_num+seg/2) then
                  array_ptrn(i,j) = 0
            elseif (array(i,j) < min+seg) then
                  array_ptrn(i,j) = num_ptrn
            elseif (min+seg<= array(i,j) .and.
     +                        array(i,j) < min+2*seg) then
                  array_ptrn(i,j) = num_ptrn-1
             elseif (min+2*seg<= array(i,j) .and.
     +                        array(i,j) < min+3*seg) then
                  array_ptrn(i,j) = num_ptrn-2
             elseif (min+3*seg<= array(i,j) .and.
     +                        array(i,j) < min+4*seg) then
                  array_ptrn(i,j) = num_ptrn-3
             elseif (min+4*seg<= array(i,j) .and.
     +                        array(i,j) < min+5*seg) then
                  array_ptrn(i,j) = num_ptrn-4
             elseif (min+5*seg<= array(i,j) .and.
     +                        array(i,j) < min+6*seg) then
                  array_ptrn(i,j) = num_ptrn-5
             elseif (min+6*seg<= array(i,j) .and.
     +                        array(i,j) < min+7*seg) then
                  array_ptrn(i,j) = num_ptrn-6
            elseif (min+7*seg<= array(i,j) .and.
     +                        array(i,j) < min+8*seg) then
                  array_ptrn(i,j) = num_ptrn-7
            elseif (min+8*seg<= array(i,j) .and.
     +                        array(i,j) < min+9*seg) then
                  array_ptrn(i,j) = num_ptrn-8
             elseif (min+9*seg<= array(i,j) .and.
     +                        array(i,j) < min+10*seg) then
                  array_ptrn(i,j) = num_ptrn-9
             elseif (min+10*seg<= array(i,j) .and.
     +                        array(i,j) < min+11*seg) then
                  array_ptrn(i,j) = num_ptrn-10
             elseif (min+11*seg<= array(i,j) .and.
     +                        array(i,j) < min+12*seg) then
                  array_ptrn(i,j) = num_ptrn-11
             elseif (min+12*seg<= array(i,j) .and.
     +                        array(i,j) < min+13*seg) then
                  array_ptrn(i,j) = num_ptrn-12
             elseif (min+13*seg<= array(i,j) .and.
     +                        array(i,j) < min+14*seg) then
                  array_ptrn(i,j) = num_ptrn-13
            elseif (min+14*seg<= array(i,j) .and.
     +                        array(i,j) <= max) then
                  array_ptrn(i,j) = num_ptrn-14
```

```
          else
              array_ptrn(i,j) = 16
            endif
        enddo
      enddo
      return
      end


***********************************************************************
*
* pict_to_scrn : draws picture of GPAC, S, or R arrays and maps it to
*                screen
*
***********************************************************************

      subroutine pict_to_scrn(array_ptrn, rows, columns,title,
     +                          flag,array_pict,rect1)
      implicit none

      integer*4 toolbx,flag
      integer ary_size;      parameter(ary_size=26)
      integer array_ptrn(0:2*ary_size,0:ary_size)
      integer columns, rows, i, j
      integer*4  array_pict, w, h,title_len,t_start
      integer*2  rect(4),rect1(4)
      character  str255*256, title*64

      integer TEXTFONT,TEXTFACE,TEXTMODE,TEXTSIZE,SPACEEXTRA,DRAWCHAR,
     +          DRAWSTRING,DRAWTEXT,CHARWIDTH,STRINGWIDTH,TEXTWIDTH,
     +          GETFONTINFO
      parameter (TEXTFONT=Z'88708000',TEXTFACE=Z'88808000',
     +             TEXTMODE=Z'88908000',TEXTSIZE=Z'88A08000',
     +             SPACEEXTRA=Z'88E10000',DRAWCHAR=Z'88308000',
     +             DRAWSTRING=Z'88430000',DRAWTEXT=Z'88511200',
     +             CHARWIDTH=Z'88D48000',STRINGWIDTH=Z'88C70000',
     +             TEXTWIDTH=Z'88651200',GETFONTINFO=Z'88B30000')

      integer   OPENPICTURE,KILLPICTURE,CLOSEPICTURE,DRAWPICTURE
        parameter (OPENPICTURE=Z'8F3B0000',KILLPICTURE=Z'8F510000',
     +             CLOSEPICTURE=Z'8F400000',DRAWPICTURE=Z'8F616000')
        integer   SETRECT,CLIPRECT,FILLRECT,FRAMERECT
      parameter (SETRECT=Z'8A731248',CLIPRECT=Z'87B30000',
     +             FILLRECT=Z'8A536000',FRAMERECT=Z'8A130000')
        integer MOVETO, HIDEPEN,SHOWPEN,PENSIZE
        parameter (MOVETO=Z'89309000',HIDEPEN=Z'89600000',
     +              SHOWPEN=Z'89700000',PENSIZE=Z'89B09000')
        INTEGER ERASERECT
        PARAMETER (ERASERECT=Z'8A330000')
      integer*1 pat0(8),pat1(8), pat2(8), pat3(8), pat4(8)
      integer*1 pat5(8), pat6(8), pat7(8), pat8(8)
      integer*1 pat9(8), pat10(8), pat11(8), pat12(8)
      integer*1 pat13(8), pat14(8), pat15(8),pat16(8)

c this pattern (pat16)is just to detect if a number in array_ptrn
c was not accounted for (for error detection and debug)

      data pat16 /b'00000000',
```

```
+              b'11111111',
+              b'00000000',
+              b'11111111',
+              b'00000000',
+              b'11111111',
+              b'00000000',
+              b'11111111'/

 data pat15 /b'00000000',
+              b'00000000',
+              b'00000000',
+              b'00000000',
+              b'00000000',
+              b'00000000',
+              b'00000000',
+              b'00000000'/

  data pat14 /b'00000000',
+              b'00000000',
+              b'00000000',
+              b'00010000',
+              b'00000000',
+              b'00000000',
+              b'00000000',
+              b'00000000'/

  data pat13 /b'10000000',
+              b'00000000',
+              b'00000000',
+              b'00000000',
+              b'00001000',
+              b'00000000',
+              b'00000000',
+              b'00000000'/

  data pat12 /b'10000000',
+              b'00000000',
+              b'00001000',
+              b'00000000',
+              b'10000000',
+              b'00000000',
+              b'00001000',
+              b'00000000'/


  data pat11 /b'10001000',
+              b'00000000',
+              b'00100010',
+              b'00000000',
+              b'10001000',
+              b'00000000',
+              b'00100010',
+              b'00000000'/

  data pat10 /b'10001000',
+              b'00100010',
+              b'10001000',
+              b'00100010',
```

```
+           b'10001000',
+           b'00100010',
+           b'10001000',
+           b'00100010'/

   data pat9 /b'11001100',
+           b'00110011',
+           b'11001100',
+           b'00110011',
+           b'11001100',
+           b'00110011',
+           b'11001100',
+           b'00110011'/

   data pat8 /b'10101010',
+           b'01010101',
+           b'10101010',
+           b'01010101',
+           b'10101010',
+           b'01010101',
+           b'10101010',
+           b'01010101'/

   data pat7 /b'01110111',
+           b'11011101',
+           b'01110111',
+           b'11011101',
+           b'01110111',
+           b'11011101',
+           b'01110111',
+           b'11011101'/

   data pat6 /b'10101010',
+           b'11111111',
+           b'01010101',
+           b'11111111',
+           b'10101010',
+           b'11111111',
+           b'01010101',
+           b'11111111'/

   data pat5 /b'01110111',
+           b'11111111',
+           b'11011101',
+           b'11111111',
+           b'01110111',
+           b'11111111',
+           b'11011101',
+           b'11111111'/

   data pat4 /b'01111111',
+           b'11111111',
+           b'11110111',
+           b'11111111',
+           b'01111111',
+           b'11111111',
+           b'11110111',
+           b'11111111'/
```

```
        data pat3 /b'01111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11110111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111'/

        data pat2 /b'11111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11101111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111'/

        data pat1 /b'11111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111',
     +            b'11111111'/
                         ! this pattern replaces the patterns of the
                         ! numbers within a seg/2 from zero
        data pat0 /b'11110000',
     +            b'11110000',
     +            b'11110000',
     +            b'11110000',
     +            b'00001111',
     +            b'00001111',
     +            b'00001111',
     +            b'00001111'/


      call toolbx(SETRECT, rect, 0, 0, 512, 342)
      call toolbx(CLIPRECT, rect)
      h = (rect1(3)-rect1(1)-21)/rows
      w = (rect1(4)-rect1(2)-20)/columns
      array_pict = toolbx(OPENPICTURE, rect1)
c Array title
c    monaco 9 point is 5 pixels horizontal and 7 vertical per character
         title_len = len(trim(title))
         t_start = (rect1(4)-rect1(2))/2 - 5*title_len/2
         call toolbx(MOVETO,t_start,9)
         write(9,90) title
  90       format(A64)
c print row indicies
         do i=0,rows-1
            call toolbx(SETRECT, rect,1*w-w,i*h,1*w,i*h+h)
            call toolbx(MOVETO,0,21+i*h+h/2+3)
            if (flag=1) then
             write(9,91) i
            elseif (flag=2).OR.(flag=3) then
```

```
            write(9,91) -rows/2+i
           endif
   91        format(i3)
        enddo
c print column indicies
        do j=1,columns+1
           call toolbx(SETRECT, rect,j*w-w,0*h,j*w,0*h+h)
           call toolbx(MOVETO,20+j*w-w/2-7,19)
           write(9,92) j
  92         format(i2)
        enddo
        call toolbx(MOVETO,20,19)

        do j=1, columns
           do i=0, rows-1
c 20 & 19 are the horizontal and vertical offset respectively
              call toolbx(SETRECT, rect,20+j*w-w,21+i*h,
     +                                    20+j*w,21+i*h+h)
              select case(array_ptrn(i,j))
            case(0)
               call toolbx(FILLRECT, rect, pat0)
            case(1)
               call toolbx(FILLRECT, rect, pat1)
            case(2)
               call toolbx(FILLRECT, rect, pat2)
            case(3)
               call toolbx(FILLRECT, rect, pat3)
            case(4)
               call toolbx(FILLRECT, rect, pat4)
            case(5)
               call toolbx(FILLRECT, rect, pat5)
            case(6)
               call toolbx(FILLRECT, rect, pat6)
            case(7)
               call toolbx(FILLRECT, rect, pat7)
            case(8)
               call toolbx(FILLRECT, rect, pat8)
            case default
               call toolbx(FILLRECT, rect, pat9)
            case(10)
               call toolbx(FILLRECT, rect, pat10)
            case(11)
               call toolbx(FILLRECT, rect, pat11)
            case(12)
               call toolbx(FILLRECT, rect, pat12)
            case(13)
               call toolbx(FILLRECT, rect, pat13)
            case(14)
               call toolbx(FILLRECT, rect, pat14)
            case(15)
               call toolbx(FILLRECT, rect, pat15)
            case default
               call toolbx(FILLRECT, rect, pat16)
            end select
            call toolbx(FRAMERECT, rect)
           enddo
        enddo
      call toolbx(CLOSEPICTURE, array_pict)
```

```
      call toolbx(DRAWPICTURE, array_pict, rect1)
      return
      end

************************************************************************
*
* optmap : Displays a dialogbox then it retrieves the new value of alpha
*          or theta before redrawing the grayscale map.
*
************************************************************************
      subroutine optmap(flag,func_type,rows,columns,alpha,
     +                  theta,array_pict,array,rect,title)
      implicit none
      include hd:include files:utilities.inc
      integer flag,rows, columns,array_pict,hte,func_type
      integer*2 itemhit,itemtype,ibox(4),rect(4),j
      integer*4 pd,toolbx,longhit,window,ihandle
      integer*4 done,less_a,more_a,less_b,more_b
      parameter(done=1,less_a=2,more_a=3,less_b=4,more_b=5)
      integer ary_size;          parameter(ary_size=26)
      real theta,alpha
      real freal1,zero_num,inc_alpha,inc_theta
      real array(-2*ary_size:ary_size,0:ary_size)
      real nw_array(-2*ary_size:ary_size,0:ary_size)
      integer array_ptrn(0:2*ary_size,0:ary_size)
      character*256 str255,static(4),edfld,alfa_theta
      character title*64, temp*10
      logical doneflag

      doneflag=.false.
      call toolbx(INITDIALOGS,0)
      window=toolbx(FRONTWINDOW)
      pd = toolbx(GETNEWDIALOG,5150,0,-1)
      call toolbx(SETPORT, pd)
      call toolbx(SHOWWINDOW,pd)

c Inserting bolding around 'ok' button
      call toolbx(GETDITEM,pd,done,itemtype,ihandle,ibox)
      call toolbx(PENSIZE,3,3)
      call toolbx(INSETRECT,ibox,-4,-4)
      call toolbx(FRAMEROUNDRECT,ibox,16,16)

      do 10 j=1,3
        call toolbx(GETDITEM,pd,j+7,itemtype,ihandle,ibox)
        call toolbx(GETITEXT,ihandle,static(j))
10    continue

      do 20 j=1,3
        call toolbx(GETDITEM,pd,j+7,itemtype,ihandle,ibox)
        if (j=1) then           ! display alpha on dialog box
           write(temp,90) alpha
           static(j) = str255('a='//temp)
        elseif (j=2) then       ! display theta on dialog box
           write(temp,90) theta
           static(j)=str255('b='//temp)
        endif
        call toolbx(SETITEXT,ihandle,static(j))
20    continue
```

```
      call toolbx(SELITEXT,pd,6,0,10)
      inc_alpha = 0.2
      inc_theta = 0.2
c   main loop
      do
        call toolbx(MODALDIALOG,0,itemhit)
                                    !itemhit is the item number from
ResEdit
        longhit = itemhit       ! 4 byte version.
        if (longhit .EQ. done) then
           doneflag=.true.
        elseif (longhit .EQ. less_a) then
           alpha=alpha-inc_alpha
        elseif (longhit .EQ. more_a) then
           alpha=alpha+inc_alpha
        elseif (longhit .EQ. less_b)  then
           theta=theta-inc_theta
        elseif (longhit .EQ. more_b) then
           theta=theta+inc_theta
        elseif (longhit .EQ. 6).OR.(longhit .EQ. 7) then
           call toolbx(GETDITEM,pd,longhit,itemtype,hte,ibox)
           call toolbx(GETITEXT,hte,edfld)
           if (longhit=6) inc_alpha= freal1(edfld(2:11))
           if (longhit=7) inc_theta= freal1(edfld(2:11))
           edfld = '          '
        endif
        if (longhit=less_a).OR.(longhit=more_a).OR.
     +        (longhit=less_b).OR.(longhit=more_b)  then
           if (alpha<-9.99) alpha=-9.99
           if (alpha>9.99)  alpha =9.99
           if (theta<-9.99) theta=-9.99
           if (theta>9.99) theta  =9.99
c Update alpha or theta
           if (longhit=less_a).OR.(longhit=more_a) then
              call toolbx(GETDITEM,pd,(longhit+6)/2*2,
     +                                  itemtype,hte,ibox)
            write(temp,90) alpha
  90        format(f5.2)
              temp='a='//temp
           else          !(longhit=less_b).OR.(longhit=more_b)
              call toolbx(GETDITEM,pd,(longhit+4)/2*2+1,
     +                                  itemtype,hte,ibox)
            write(temp,90) theta
            temp='b='//temp
           endif
           alfa_theta= temp
           call toolbx(SETITEXT,hte,str255(alfa_theta))

c Subroutine calls to update the grayscale map.
           call toolbx(SETPORT,window)
           call lim_sigm(func_type,array,nw_array,rows,columns,
     +                    alpha,theta,zero_num)
           call num_to_ptrn(nw_array, array_ptrn, rows,
     +                    columns,zero_num)
           call pict_to_scrn(array_ptrn,rows, columns,title,
     +                     flag,array_pict,rect)
                 call toolbx(DRAWPICTURE,array_pict,rect)
```

```
          endif

          call toolbx(SETPORT,pd)
          if (doneflag) exit
        repeat

        call toolbx(DISPOSEDIALOG,pd)
        call toolbx(SETPORT,window)
        call toolbx(SELECTWINDOW,window)
        return
        end

c-------------------------------------------------------------------------
c       This a function routine to convert ASCII data to a REAL format.
c         This function is duplicated in dlog.for.  Keep only one.
c-------------------------------------------------------------------------
        real function freal1(temp)
        implicit none
        integer*4 i,j,k,fmt,err
        character*(*)temp
        character*20 ascii

        j=0
        k=0
        ascii='                    '
        ascii=temp
c Find start and ending of number to convert
        do 20 i = 1, 20
           if(j.EQ.0.AND.ascii(i:i).NE.' ')j=i
           IF(j.NE.0.AND.ascii(i:i).EQ.' ')then
                k=i-1
                goto 40
             endif
20      continue

c Find decimal if one is not found put one there
40      continue
        if (k.EQ.0) goto 200
        do 50 i=j,k
           if(ascii(i:i).EQ.'.') goto 60
50      continue
        k=k+1
        ascii(k:k)='.'

c Internal read to convert character numbers to real data
60      continue
        read(ascii(j:k),fmt=100,err=200)freal1
100     format(G10.4)
        goto 300

200       continue
        freal1=0.0

300     continue
        return
        end
```

```
************************************************************************
*
* plot:  plots any given data (x,y) to the screen.  Also it returns
*        a picture handel of the plot.  The size of the plot is rect.
*
************************************************************************
      subroutine plot(plotpic,rect,proc,x,y,npts)
      implicit none
      include hd:Include files:utilities.inc

      integer*4 toolbx,xmajor,xminor,ymajor,yminor,npts
      integer*4 xaxsmin,xaxsmax,yaxsmin,yaxsmax,plotpic
      integer   i,eof,status,plotseq;parameter(eof=-1)
      integer*2 rect(4)
      integer*4 proc
      character*64 filename,xaxis,yaxis,title,prompt
      real x(*), y(*)

      data xmajor,xminor,ymajor,yminor/8,10,6,10/
                                  ! for short and long tick marks

      if (proc=0) then
         prompt = 'Select a file to plot:'
         call getfil('TEXT', filename,prompt)
         if (filename=' ') return

         call setcur(4)
         open(unit=2, file=filename)
         i=1; status=0
         while ((status<>eof).and.(i<4000))
            read(2,*,IOSTAT=status) x(i), y(i)
            i=i+1
         repeat
 100     npts = i-2
         close(2)
      endif

      xaxsmin=80
      xaxsmax=rect(4) - 15
      yaxsmin=25
      yaxsmax=rect(3) - 40
      xaxis='x-axis'
      yaxis='y-axis'
      if (proc=6) then
         title='noise sequence'
      elseif (proc=7) then
         title='correlation sequence'
      elseif (proc=8) then
         title='PACF sequence'
      elseif (proc=9) then
         title='arma sequence'
      else
         title=filename
      endif

      call plot_data(npts,x,y,xmajor,xminor,ymajor,yminor,xaxsmin,
     &      xaxsmax,yaxsmin,yaxsmax,xaxis,yaxis,title,rect,plotpic)
```

```
      call setcur(0)
      return
      end
c-----------------------------------------------------------------------
c
c     The program uses the following variables:
c
c     num- this is the number of data points that are to be plotted.
c     x(*)- this is the array for the x-axis points
c     y(*)- this is the array for the y-axis points
c     xmajor- the number of x-axis major divisions
c     xminor- the number of x-axis minor divisions
c     ymajor- the number of y-axis major divisions
c     yminor- the number of y-axis minor divisions
c     xaxsmin- the starting point for the x-axis in pixels
c     xaxsmax- the ending point for the x-axis in pixels
c     yaxsmin- the starting point for the y-axis in pixels
c     yaxsmax- the ending point for the y-axis in pixels
c     xaxis- the title for the x-axis
c     yaxis- the title for the y-axis
c     title- the title for the plot
c-----------------------------------------------------------------------
      subroutine plot_data(num,x,y,xmajor,xminor,ymajor,yminor,xaxsmin
     &       ,xaxsmax,yaxsmin,yaxsmax,xaxis,yaxis,title,rect,plotpic)
      implicit none
      include hd:Include files:utilities.inc
      integer*4 xticx,xticy,xticxm,yticx,yticy,yticym,toolbx,h,v,error
      integer*4 xmajor,xminor,ymajor,yminor,rk,dx,dy,xlen,xstart
      integer*4 xaxsmin,xaxsmax,yaxsmin,yaxsmax,plotpic,ylen,ystart
      integer*4 j,tlen,tstart,flength,num
      integer*2 rect(4),rect1(4)
      integer*4 menuhandle(2),proc
      character*1 ychar
      character*64 xaxis,yaxis,title,char_mtrx(3)
      character*256 str255
      real xmax,xmin,ymax,ymin,xmax2,ymax2,xmin2,ymin2
      real delx,dely,delx2,dely2,varx,vary
      real x(*),y(*),mtrx(4)
      logical cnclflag

      call automatic(x,y,num,xmax,xmin,ymax,ymin)
      xmax2=xmax
      xmin2=xmin
      ymax2=ymax
      ymin2=ymin
      mtrx(1)=xmin2
      mtrx(2)=xmax2
      mtrx(3)=ymin2
      mtrx(4)=ymax2
      char_mtrx(1)=str255(xaxis)
      char_mtrx(2)=str255(yaxis)
      char_mtrx(3)=str255(title)
      proc=2
      call setcur(0)
      call dlog(1300,4,6,mtrx,1,2,5,menuhandle,cnclflag,
     +            proc,char_mtrx)
        call setcur(4)
      if (.NOT.cnclflag) return          ! was the button hit Cancel?
```

```
      xaxis=char_mtrx(1)
      flength = ichar(xaxis(1:1))
      xaxis = xaxis(2:flength+1)
      yaxis=char_mtrx(2)
      flength = ichar(yaxis(1:1))
      yaxis = yaxis(2:flength+1)
      title=char_mtrx(3)
      flength = ichar(title(1:1))
      title = title(2:flength+1)
      xmin2=mtrx(1)
      xmax2=mtrx(2)
      ymin2=mtrx(3)
      ymax2=mtrx(4)
      delx=(xmax-xmin)/float(xmajor)
        delx2=(xmax2-xmin2)/float(xmajor)
        dely=(ymax-ymin)/float(ymajor)
        dely2=(ymax2-ymin2)/float(ymajor)

        call toolbx(TEXTFONT,4)      !monaco
        call toolbx(TEXTSIZE,9)      ! 9 pt.
      rect1(1) =0
      rect1(2) =0
      rect1(4) =512
      rect1(3) =342
      call toolbx(CLIPRECT, rect1)
      call toolbx(ERASERECT,rect)
      plotpic=toolbx(OPENPICTURE,rect)
c-----------------------------------
c set up the labels on axis
c-----------------------------------
c  the x axis
c     monaco 9 point is 5 pixels horizontal per character
      xlen=len(trim(xaxis))
      xstart=((xaxsmax-xaxsmin)/2 + xaxsmin) - 5*xlen/2
        call toolbx(MOVETO,xstart,yaxsmax+35)
        write(9,60)xaxis
60      format(A64)

c the y axis
c     monaco 9 point is 7 pixels vertical per character
      ylen=len(trim(yaxis))
      ystart=((yaxsmax-yaxsmin)/2 + yaxsmin) - 7*ylen/2
      h=xaxsmin-70
      v=ystart
        call toolbx(MOVETO,h,v)
      do 65 j=1,ylen
      ychar=yaxis(j:j)
        write(9,70)ychar
      v=ystart+11*j
        call toolbx(MOVETO,h,v)
65    continue
70      format(A1)

c  the title
      tlen=len(trim(title))
      tstart=((xaxsmax-xaxsmin)/2 + xaxsmin) - 5*tlen/2
      call toolbx(MOVETO,tstart,yaxsmin-3)
      write(9,75)title
```

```
75     format(A50)
c-----------------------------------------------------------------------
c set up the numbers on the axis
c-----------------------------------------------------------------------
c  the x axis
       call toolbx (MOVETO,xaxsmin-20,yaxsmax+15)
       do 90 j=1,xmajor+1
       varx=xmin2+(delx2*(float(j)-1.))
         write(9,80)varx
80       format(1PG9.3,$)
       h=(xaxsmin-20)+(j*(xaxsmax-xaxsmin)/xmajor)
         call toolbx(MOVETO,h,yaxsmax+15)
90       continue

c  the y axis
       call toolbx(MOVETO,xaxsmin-55,yaxsmax+2)
         do 110 rk=1,ymajor+1
         vary=ymin2+(dely2*(float(rk)-1.))
       write(9,100)vary
100        format(1PG9.3,$)
       v=(yaxsmax+2)-(rk*(yaxsmax-yaxsmin)/ymajor)
         call toolbx(MOVETO,xaxsmin-55,v)
110      continue
c-----------------------------------------------------------------------
c set up axis
c-----------------------------------------------------------------------
       call toolbx(MOVETO,xaxsmin,yaxsmin)
       call toolbx(LINETO,xaxsmin,yaxsmax)
       call toolbx(LINETO,xaxsmax,yaxsmax)
       call toolbx(LINETO,xaxsmax,yaxsmin)
       call toolbx(LINETO,xaxsmin,yaxsmin)
c-----------------------------------------------------------------------
c set up tick marks on x axis
c-----------------------------------------------------------------------
       do j =1, xmajor*xminor
          xticx=xaxsmin+j*(xaxsmax-xaxsmin)/(xminor*xmajor)
         yticx=yaxsmax
          call toolbx(MOVETO,xticx,yticx)
         if (mod(j,xminor)=0) then
             call toolbx(LINETO,xticx,yticx-5)        !Major tick marks
         else
           call toolbx(LINETO,xticx,yticx-3)         !Minor tick marks
         endif
       enddo
c-----------------------------------------------------------------------
c set up tick marks on y axis
c-----------------------------------------------------------------------
       do j =1, ymajor*yminor
          xticy=xaxsmin
          yticy=yaxsmax-j*(yaxsmax-yaxsmin)/(ymajor*yminor)
          call toolbx(MOVETO,xticy,yticy)
         if (mod(j,yminor)=0) then
             call toolbx(LINETO,xticy+5,yticy)        !Major tick marks
         else
           call toolbx(LINETO,xticy+3,yticy)         !Minor tick marks
         endif
       enddo
c-----------------------------------------------------------------------
```

```
c draw the y = 0 line (if it exists)
c-----------------------------------------------------------------------
      if (ymax2*ymin2.LT.0.) then
         dx=xaxsmin
         dy=int(float(yaxsmax)-float(yaxsmax-yaxsmin)*(0.0-ymin2)
     &      /(ymax2-ymin2))
         call toolbx(MOVETO,dx,dy)
         dx=xaxsmax
         call toolbx(LINETO,dx,dy)
      endif
c-----------------------------------------------------------------------
c plot points
c-----------------------------------------------------------------------
      dx=xaxsmin+(xaxsmax-xaxsmin)*(x(1)-xmin2)/(xmax2-xmin2)
      dy=yaxsmax-(yaxsmax-yaxsmin)*(y(1)-ymin2)/(ymax2-ymin2)
      h=dx
      v=dy
      call toolbx(MOVETO,h,v)
      do 160 j=2,num
      dx=int(float(xaxsmin)+float(xaxsmax-xaxsmin)*(x(j)-xmin2)
     &      /(xmax2-xmin2))
      dy=int(float(yaxsmax)-float(yaxsmax-yaxsmin)*(y(j)-ymin2)
     &      /(ymax2-ymin2))
      h=dx
      v=dy
      call toolbx(LINETO,h,v)
160   continue
c-----------------------------------------------------------------------
c  end of program
c-----------------------------------------------------------------------
      call toolbx(CLOSEPICTURE)
      call toolbx(DRAWPICTURE,plotpic,rect)
      return
      end

c-----------------------------------------------------------------------
c  subroutine automatic finds the max and min of the two data arrays
c-----------------------------------------------------------------------
      subroutine automatic(x,y,num,xmax,xmin,ymax,ymin)
      implicit none
      integer*4 j
      real xmax,xmin,ymax,ymin
      real x(*),y(*)
      xmax=x(1)
      xmin=x(1)
      ymax=y(1)
      ymin=y(1)
      do 190 j=2,num
         IF (x(j) .GT. xmax) xmax=x(j)
         IF (x(j) .LT. xmin) xmin=x(j)
         IF (y(j) .GT. ymax) ymax=y(j)
         IF (y(j) .LT. ymin) ymin=y(j)
190   continue
      IF (xmin = xmax) xmax = xmax + 1
      IF (ymin = ymax) ymax = ymax + 1
      return
      end
```

```
*********************************************************************
*
* dowhit: generates gaussian or unifom noise.
*          flag : 1 (uniform noise)
*             2 (gaussian noise)
*
**********************************************************************

      subroutine dowhit(flag,plotpic,rect)
      implicit none
      include hd:include files:utilities.inc

      integer  flag

      integer*4 toolbx, error, misc(2),plotpic
      integer*2 rect(4)
      integer  npts,i,seed,plotseq,proc
      real  noisebuf(4000),x(4000),mean,var,ed_fld_data(5)
      character temp*3
      integer*2 good
      character*64 FILENAME, prompt, origname,char_mtrx(4)
      logical cnclflag


      proc=6
      ed_fld_data(1) = 0.           !mean
      ed_fld_data(2) = 1.           !var
      ed_fld_data(3) = 5709         !seed
      ed_fld_data(4) = 500          !# of pts as displayed in dialog box
      call dlog(3100,4,4,ed_fld_data,1,2,3,misc,
     +          cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) return
      call setcur(4)
      mean = ed_fld_data(1)
      var  = ed_fld_data(2)
      seed = int(ed_fld_data(3))
      npts = int(ed_fld_data(4))
      plotseq=proc                  !proc returns 1 if plot is needed

      if (npts>4000) then
         proc=11
         char_mtrx(1)=''
         char_mtrx(2)='Cannot generate more than 4000 points.'
         char_mtrx(3)=''
         char_mtrx(4)='4000 points are being generated'
         call dlog(515,0,0,ed_fld_data,1,0,2,misc,
     +             cnclflag,proc,char_mtrx)
         npts=4000
      endif

      call white(mean,var,seed,flag,npts,noisebuf)
      prompt = "save noise sequence as:"
      origname = "noise.dat"
      call setcur(0)
      call putfil(FILENAME, prompt, origname)
      if (filename<>' ') then
         call setcur(4)
         open(1,file=filename,status='new')
```

```
         do i=1,npts
            write(1,*) i-1,',',noisebuf(i)
         enddo
         close(1)
      endif

      if (plotseq = 1) then
         do i=1,npts
            x(i)=i-1
         enddo
         call plot(plotpic,rect,6,x,noisebuf,npts)
      endif
      return
      end

************************************************************************
*
*  WHITE:  CALCULATES WHITE NOISE. THE USER IS PROMPTED FOR
*     THE MEAN, VARIANCE, AND SEED.
*
************************************************************************

      SUBROUTINE WHITE(MEAN, VAR, SEED, FLAG, N, E)

      implicit none
      INTEGER  SEED, FLAG,N,I,J
      REAL     RANDOM,MEAN,E(*),VAR

      IF (FLAG.EQ.1)  THEN                      !uniform noise
         DO  I=1,N
            E(I)= RANDOM(SEED)*SQRT(VAR*12.) + MEAN
         ENDDO
      ELSEIF (FLAG.EQ.0) THEN                   !gaussian noise
         DO 20 I=1,N
            E(I)=0
            DO J=1,12
               E(I)=E(I)+RANDOM(SEED)
            ENDDO
            E(I)=E(I)*SQRT(VAR*1.) + MEAN
         ENDDO
      ENDIF
      END

      REAL FUNCTION RANDOM(SEED)

      INTEGER SEED

      SEED=MOD(3125*SEED,65536)
      RANDOM= (SEED/65536.) - 0.5
      RETURN
      END
```

```
************************************************************************
*
* doarma:  Simulates ARMA, AR, and MA.
*
************************************************************************
      subroutine doarma(plotpic,rect)

      implicit none
      integer*4  toolbx,p_and_q(2),proc,dum,plotpic,plotseq
      integer*2  rect(4)
      integer p,q,i,npts,status,eof    ;parameter(eof=-1)
      character*64 filename, prompt, origname,char_mtrx(3)
      real   noisebuf(-20:4000,2),armabuf(-20:4000)
      real   phi(20),theta(20), ed_fld_data(25),x(4000),y(4000)
      logical cnclflag

*
* prompt the user for parameters needed
*

      proc=9                          !calling proc. is doarma
      ed_fld_data(1) = 0.             !AR order (p)
      ed_fld_data(2) = 0.             !MA order (q)
      call dlog(3200,2,3,ed_fld_data,1,2,3,p_and_q,
     +             cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) return
      p = int(ed_fld_data(1))
      q  = int(ed_fld_data(2))
      if (p>12) p=12
      if (p<0) p =0
      if (q>12) q=12
      if (q<0) q =0
      if ((p=0).and.(q=0)) return
      plotseq = proc

      do i=1,25
         ed_fld_data(i) = 0.0         !parameters
      enddo
      proc=5
      p_and_q(1) = p                  ! send p & q to display exactly
      p_and_q(2) = q                  ! p+q edit fields
      call dlog(3210,24,0,ed_fld_data,1,2,2,p_and_q,
     +             cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) return
      call setcur(4)
      dum=0
      do i=1,p
         phi(i) = ed_fld_data(i)          !AR parameters
         if (phi(i)<>0) dum=dum+1
      enddo
      do i=1,q
         theta(i) = ed_fld_data(12+i)          !MA parameters
         if (theta(i)<>0) dum=dum+1
      enddo
      if (dum=0) return               ! dum=0 if all parameters are zeros
      call setcur(0)
      prompt = 'Select a noise file input to ARMA:'
      call getfil("TEXT",filename,prompt)
```

```
      if (filename=' ') return
      call setcur(4)
*
* read data from 'filename' file
*
      open(unit=2, file=filename)
      i=0; status=0
      while ((status<>eof).and.(i<4000))
         read(2,*,IOSTAT=status) noisebuf(i,1), noisebuf(i,2)
         i=i+1
      repeat
      npts = i-1
      close(2)

      call arma(phi,p,theta,q,npts,noisebuf,armabuf)

*
* prompt the user for output file and save data to it
*
      prompt = "save ARMA sequence as:"
      origname = "arma.dat"
      call setcur(0)
      call putfil(filename,prompt,origname)
      if (filename<>' ') then
         call setcur(4)
         open(1,file=filename,status='new' )
         do i=0,npts-1
            write(1,*) noisebuf(i,1),',',armabuf(i)
         enddo
         close(1)
      endif
      if (plotseq=1) then
         do i=1,npts
            x(i) = noisebuf(i-1,1)
            y(i) = armabuf(i-1)
         enddo
         call plot(plotpic,rect,9,x,y,npts)
      endif
      call setcur(0)

      return
      end

**********************************************************************
*
*   ARMA:  CALCULATES AN ARMA SEQUENCE GIVEN ITS ORDERS AND
*          PARAMETERS.
*
**********************************************************************
      SUBROUTINE ARMA(PHI,P,THETA,Q,N,E,Z)

      IMPLICIT NONE
      INTEGER P,Q,I,K,N
      REAL    Z(-20:4000), E(-20:4000,2), PHI(20), THETA(20), ACC

        DO 10 I=-20,N-1
 10       Z(I)=0
```

```
      DO 20 I=-20,-1
20       E(I,2)=0

       DO 30 K=0,N-1
         DO 40 I=1,P
40         Z(K)=Z(K) +  PHI(I)*Z(K-I)
         DO 50 I= 1,Q
50         Z(K)=Z(K) -  THETA(I)*E(K-I,2)
30       Z(K)= Z(K) + E(K,2)
      END
```

```
*********************************************************************
*
*  CORLAT: PERFORMS AUTOCORRELATION AND CROSSCORRELATION
*
*********************************************************************

      subroutine corlat(flag,plotpic,rect)

      implicit none

      integer*2 rect(4)
      integer*4  flag,proc,plotseq,plotpic,cross,acf
      parameter(acf=1,cross=2)
      integer*4  toolbx, error, misc(2)
      integer  lags,npts,npts2,i,status,eof     ;parameter(eof=-1)
      character*64 filename,prompt,origname,char_mtrx(4)
      character*3 temp
      real   databuf1(0:4000,2),databuf2(0:4000,2),ed_fld_data
      real   x(1000),y(1000)
      logical cnclflag


      proc=7
      ed_fld_data = 0          !ACF lags
      call dlog(4100,1,1,ed_fld_data,1,2,3,misc,
     +          cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) return
      lags = int(ed_fld_data)
      plotseq=proc
      if (lags>1000) then
         proc=11
         char_mtrx(1)=''
         char_mtrx(2)='A maximum of 1000 lags can be generated.'
         char_mtrx(3)='1000 ACF lags are being generated.'
         char_mtrx(4)=''
         call dlog(515,0,0,ed_fld_data,1,0,2,misc,
     +             cnclflag,proc,char_mtrx)
      endif
      prompt = 'Select an input file for correlation:'
      call getfil("TEXT",filename,prompt)
      if (filename=' ') return

c read data from 'filename' file
      call setcur(4)
      open(unit=2, file=filename)
      i=0; status=0
      while ((status<>eof).and.(i<1000))
         read(2,*,IOSTAT=status) databuf1(i,1), databuf1(i,2)
         i=i+1
      repeat
      npts = i-2
      close(2)

      if (lags>=npts) then
         call setcur(0)
         proc=11
         lags = npts-1
         write(temp,92) lags
```

```
 92     format(I3)
        char_mtrx(1)=''
        char_mtrx(2)='Lags MUST be less than the number of'
        char_mtrx(3)='points.'
        char_mtrx(4)= temp//' lags will be generated.'
        call dlog(515,0,0,ed_fld_data,1,0,2,misc,
     +              cnclflag,proc,char_mtrx)
        call setcur(4)
      endif

      if (flag=cross) then                 ! for cross correlation only
        call setcur(0)
        prompt = 'Select a second input file for correlation:'
        call getfil("TEXT",filename,prompt)
        if (filename=' ') return
        call setcur(4)
        open(unit=2, file=filename)
        i=0; status=0
        while ((status<>eof).and.(i<1000))
          read(2,*,IOSTAT=status) databuf2(i,1), databuf2(i,2)
          i=i+1
        repeat
        npts2 = i-2
        close(2)
        if (npts>npts2) then
          do i=npts2,npts-1
            databuf2(i,2)=0
          enddo
        elseif (npts2>npts) then
          do i=npts ,npts2-1
            databuf1(i,2)=0
          enddo
        endif
        npts = max(npts,npts2)
      endif

      if (flag=acf) then
        call correlate(databuf1,databuf1,npts,lags,acf)
      else
        call correlate(databuf1,databuf2,npts,lags,cross)
      endif

*
* prompt the user for output file and save data to it
*
      prompt = "save correlation sequence as:"
      if (flag=acf) origname = "acf.dat"
      if (flag=cross) origname = "ccf.dat"
      call setcur(0)
      call putfil(filename,prompt,origname)
      if (filename<>' ') then
        call setcur(4)
        open(1,file=filename,status='new')
        if (flag=2) lags = 2*lags
        do i=0,lags
          write(1,*) databuf1(i,1),',',databuf1(i,2)
        enddo
        ENDFILE(1)
```

```
         close(1)
      endif

      if (plotseq=1) then
         if (flag=2) lags = 2*lags
         do i=1,lags+1
            x(i) = databuf1(i-1,1)
            ed_fld_data=x(i)
            y(i) = databuf1(i-1,2)
            ed_fld_data=y(i)
         enddo
         call plot(plotpic,rect,7,x,y,lags+1)
      endif
      call setcur(0)
      return
      end


**********************************************************************
*****
***** THIS ROUTINE SUBTRACTS THE MEAN FROM A GIVEN PROCESS BEFORE IT
***** CALCULATES THE AUTOCORRELATION SEQUENCE.
*****
**********************************************************************

       SUBROUTINE CORRELATE(E1, E2, N, LAG, FLAG)

       IMPLICIT NONE

       REAL E1(0:4000,2), E2(0:4000,2)
       REAL CORRP(0:1000), CORRN(0:1000),VAR1,VAR2
       INTEGER TAU, FLAG,I,N,LAG

       CALL SUBMEAN(E1,N)
       CALL SUBMEAN(E2,N)

       VAR1 = 0
       VAR2 = 0
       DO I = 0, N-1
         VAR1 = E1(I,2)**2 + VAR1
         VAR2 = E2(I,2)**2 + VAR2
       ENDDO
       VAR1 = SQRT(VAR1/N)
       VAR2 = SQRT(VAR2/N)

       DO TAU=0,LAG
        CORRP(TAU)=0.
        DO I=1,N-TAU
           CORRP(TAU) = (E1(I,2)*E2(I+TAU,2))+CORRP(TAU)
        ENDDO
        CORRP(TAU) = CORRP(TAU)/FLOAT(N-TAU)
        CORRP(TAU) = CORRP(TAU)/(VAR1*VAR2)
       ENDDO

       IF (FLAG.EQ.2) THEN
         DO 30 TAU=0,-LAG, -1
          CORRN(-TAU)=0.
```

```
      DO I=1, N+TAU
         CORRN(-TAU) = (E2(I,2)*E1(I-TAU,2))+CORRN(-TAU)
      ENDDO
      CORRN(-TAU) = CORRN(-TAU)/((VAR1 * VAR2)*(N-TAU))
   ENDDO

     DO  I =0, LAG-1
      E1(I,1) = -(LAG-I)
        E1(I,2) = CORRN(LAG-I)
   ENDDO
     DO  I=LAG, 2*LAG
      E1(I,1) = I-LAG
        E1(I,2) = CORRP(I-LAG)
   ENDDO
   ELSE
     DO I=0, LAG
      E1(I,2) = I
        E1(I,2) = CORRP(I)
   ENDDO
   ENDIF

   RETURN
   END

**********************************************************************
***
*** SUBMEAN: SUBTRACT THE MEAN FROM E SEQUENCE
***
**********************************************************************

   SUBROUTINE SUBMEAN(E,N)
   IMPLICIT NONE

   REAL E(0:4000,2), MEAN
   INTEGER  I,N

   MEAN = 0.
   DO I = 0,N-1
      MEAN = MEAN + E(I,2)
   ENDDO
   MEAN = MEAN/FLOAT(N)
   DO I = 1, N
      E(I,2) = E(I,2) - MEAN
   ENDDO
   RETURN
   END
```

```
************************************************************************
*
*  LEVINSON:  CALCULATES THE PARTIAL AUTOCORRELATION SEQUENCE USING
*             LEVINSON ALGORITHM. THE USER IS PROMPTED FOR ACF, # OF
*             LAGS, AND THE OUTPUT FILE.
*
************************************************************************
      SUBROUTINE LEVINSON (plotpic,rect)

      IMPLICIT NONE

      integer*4  toolbx,misc(2),plotpic
      REAL PACF(100),VARE(0:101),PHI(0:101,0:101)
      REAL Q(101),ACF(0:1000),DUM,ed_fld_data,x(1000),y(1000)
      integer*2  rect(4)
      INTEGER I, M, N, LAGP, LAG, STATUS,EOF,proc,plotseq
      PARAMETER(EOF = -1)
      CHARACTER*64 FILENAME,PROMPT,ORIGNAME,CHAR_MTRX(4)
      character temp*3
      logical cnclflag

      proc=8
      ed_fld_data = 0          !PACF lags
      call dlog(4300,1,1,ed_fld_data,1,2,3,misc,
     +          cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) return
      lagp = int(ed_fld_data)
      plotseq = proc
      prompt = 'Select an ACF input file to PACF:'
      call getfil("TEXT",filename,prompt)
      if (filename=' ') return
*
* read data from 'filename' file
*     call setcur(4)
      open(unit=2, file=filename)
      i=0; status=0
      while ((status<>eof).and.(i<100))
         read(2,*,IOSTAT=status) dum, acf(i)
         i=i+1
      repeat
      lag = i-1
      close(2)

      if ((LAG-1).LE.LAGP) THEN
         LAGP = LAG-2
         proc=11
         write(temp,92) LAGP
   92    format(I3)
         char_mtrx(1)='PACF lags SHOULD be 2 less than the ACF lags.'
         char_mtrx(2)= temp//' PACF lags are being generated.'
         char_mtrx(3)=''
         char_mtrx(4)=''
         call dlog(515,0,0,ed_fld_data,1,0,2,misc,
     +             cnclflag,proc,char_mtrx)
      endif
C
C INITIALIZATIONS
C
```

```
      VARE(0) = ACF(0)
      Q(1) = ACF(1)
      DO I = 0, LAG
           PHI(I,0) = -1.
      ENDDO

      PHI(1,1) = Q(1)/VARE(0)
      VARE(1)  = VARE(0)*(1-(PHI(1,1)**2))
C
C ALGORITHM'S MAIN LOOP
C
      DO M=1, LAGP
           Q(M+1) = 0.
           DO N = 0, M
              Q(M+1) = Q(M+1)+ ACF(M+1-N)*(-PHI(M,N))
           ENDDO
           PHI(M+1,M+1) = Q(M+1)/VARE(M)
           DO N = 1,M
              PHI(M+1,N)= PHI(M,N)-PHI(M+1,M+1)*PHI(M,M+1-N)
           ENDDO
           VARE(M+1) = VARE(M)*(1+PHI(M+1,M+1)**2)
      ENDDO
*
* prompt user for output file and save data to it
*
      prompt = "save PACF sequence as:"
      origname = "pacf.dat"
      call setcur(0)
      call putfil(filename,prompt,origname)
      if (filename<>' ') then
         call setcur(4)
         open(1,file=filename,status='new')
         do i = 1, lagp
            write(1,*) i,phi(i,i)
         enddo
         close(1)
      endif

      if (plotseq=1) then
          do i=1,lagp
            x(i) = i
            y(i) = phi(i,i)
         enddo
         call plot(plotpic,rect,8,x,y,lagp)
      endif
      call setcur(0)
      return
      end
```

```
*********************************************************************
*
*  THIS SUBROUTINE CALCULATES THE GENERALIZED PARTIAL AUTOCORRELATION
*  ARRAY GIVEN THE AUTOCORRELATION, ACF, SEQUENCE. THE USER IS PROMPTED
*  IN THE MAIN PROGRAM FOR THE ACF, # OF LAGS, AND THE OUTPUT FILE.
*
*********************************************************************
      SUBROUTINE GPAC(FLAG,ARRAY_PICT,RECT)
       IMPLICIT NONE
       INCLUDE hd:include files:Estm-menu.inc

       INTEGER*4 ARRAY_PICT,proc
       INTEGER*2 RECT(4)
       INTEGER ARY_SIZE
       PARAMETER(ARY_SIZE=26)
       REAL PHIKJ(-2*ARY_SIZE:ARY_SIZE,0:ARY_SIZE)
       real S(-2*ARY_SIZE:ARY_SIZE,0:ARY_SIZE)
       REAL ACF(0:1000), R(-2*ARY_SIZE:ARY_SIZE,0:ARY_SIZE)
       REAL PI, W, ed_fld_data(3)
       INTEGER I, J, K, dum, FLAG
       INTEGER KLAG, JLAG, LAG, STATUS
       INTEGER EOF, g_pac, s_array, r_array
       PARAMETER(EOF=-1, g_pac=1, s_array=2, r_array=3)
       INTEGER*4 toolbx, misc(2)
       CHARACTER*64 filename, prompt, origname,char_mtrx(4)
       character temp*3
       LOGICAL cnclflag

*********************************************************************
*****
*****      INITALIZATION OF S AND R ARRAYS
*****
*********************************************************************

C PROMPT THE USER FOR THE PARAMETERS

      IF (FLAG=G_PAC) THEN
         PROMPT = 'Select an ACF input file to GPAC:'
      ELSEIF (FLAG=S_ARRAY) THEN
         PROMPT = 'Select an ACF input file to S ARRAY:'
      ELSE
         PROMPT = 'Select an ACF input file to R ARRAY:'
      ENDIF
      CALL GETFIL("TEXT",filename,prompt)
      IF (FILENAME=' ') RETURN

      OPEN(UNIT=2, FILE=FILENAME)
      I=0; STATUS=0
      WHILE ((STATUS<>EOF).AND.(I<1000))
         READ(2,*,IOSTAT=STATUS) dum, ACF(I)
         I=I+1
      REPEAT
      LAG = I-2
      if .NOT.(lag<ary_size) lag =ary_size-1
                                 !to avoid array boundary error
      CLOSE(2)

      do
```

```
      proc=0
      ed_fld_data(1) = 8            !KLAG
      ed_fld_data(2) = 8            !JLAG
      ed_fld_data(3) = 0.0          !W
      call dlog(5100,3,3,ed_fld_data,1,2,2,misc,
     +          cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) return
      KLAG = int(ed_fld_data(1))
      JLAG  = int(ed_fld_data(2))
      W = ed_fld_data(3)

      if (KLAG+JLAG-1>LAG) then
         proc=11
         write(temp,92) LAG
 92      format(I3)
         char_mtrx(1)=''
         char_mtrx(2)='(width + height - 1) SHOULD be ≤ '//temp
         char_mtrx(3)=''
         char_mtrx(4)='('//temp//' = autocorrelation lags)'
         call dlog(514,0,0,ed_fld_data,1,0,2,misc,
     +             cnclflag,proc,char_mtrx)
        else
         exit
      endif
      repeat
        DO J = 0, ARY_SIZE
          DO I = -2*ARY_SIZE, ARY_SIZE
            R(I,J)=1
            S(I,J)=0
          ENDDO
       ENDDO

       PI = ATAN(1.)*4.
       DO  I= 0, LAG
          R(I,1) = COS(2.*PI*W*I) * ACF(I)/ACF(0)
          R(-I,1) = COS(2.*PI*W*(-I)) * ACF(I)/ACF(0)
       ENDDO
       DO I = 0, lag+1
          S(-I,0) = 1
          S(I,0)  = 1
       ENDDO
**********************************************************************
*****
*****    CALCULATING THE S AND R ARRAYS
*****
**********************************************************************
       DO 50  K = 1, LAG

          DO 30 J = (LAG-K),-(LAG+K),-1
             IF (R(J,K).EQ.0.)   R(J,K) = 1E-2
             S(J,K) = S(J+1,K-1) * (R(J+1,K)/R(J,K) -1.)
 30       CONTINUE

          DO 40 J = (LAG-K),-(LAG+K),-1
             IF (S(J,K).EQ.0.)   S(J,K)=1E-2
             R(J,K+1) = R(J+1,K) * (S(J+1,K)/S(J,K) -1)
 40       CONTINUE
```

```
 50     CONTINUE
**********************************************************************
*****
*****    CALCULATING THE   GPAC = -S(K,J-K+1)/S(K,-K-J)
*****
**********************************************************************
      if (flag=g_pac) then
         DO 60 K = 1, KLAG
            DO 70 J = 0, JLAG
               IF (S(-K-J,K).EQ.0.)    S(-K-J,K) = 1E-2
               PHIKJ(J,K) = -S(J-K+1,K)/S(-K-J,K)
 70         CONTINUE
 60      CONTINUE

         prompt = "save GPAC array as:"
       origname = "gpac.dat"
         call putfil(filename,prompt,origname)
       if (filename<>' ') then
          open(1,file=filename,status='new')
          write(1,91)  (i,i=1,KLAG)
          do J=0,JLAG
               write(1,90)J, (PHIKJ(J,K),K=1,KLAG)
          enddo
          close(1)
       endif
       CALL plt_gsr(FLAG,ARRAY_PICT,PHIKJ,JLAG,KLAG,RECT)
      elseif(flag=s_array) then
         do j=-JLAG,JLAG
          do i=1,KLAG
             R(J+JLAG,I) = S(J-I+1,I)     !write shifted-S into R
          enddo
       enddo
       prompt = "save S-array as:"
       origname = "s_array.dat"
         call putfil(filename,prompt,origname)
       if (FILENAME<>' ') then
          open(1,file=filename,status='new')
          write(1,91)  (i,i=1,KLAG)
          do J=0,2*JLAG
               write(1,90)  J-JLAG,(R(J,I),I=1,KLAG)
            enddo
          close(1)
       endif
       CALL plt_gsr(FLAG,ARRAY_PICT,R,2*JLAG+1,KLAG,RECT)
     elseif (flag=r_array) then
     do j=-JLAG,JLAG
          do i=1,KLAG
             S(J+JLAG,I) = R(J-I+1,I)     !write shifted-R into S
          enddo
       enddo
       prompt = "save R-array as:"
       origname = "r_array.dat"
         call putfil(filename,prompt,origname)
       if (FILENAME<>' ') then
          open(1,file=filename,status='new')
          write(1,91)  (i,i=1,KLAG)
 91       format(9x,20(i2,7x))
          do J=0,2*JLAG
```

```
              write(1,90)J-JLAG,(S(J,I),I=1,KLAG)
90            format(1X,I3,1X,20(1X,F8.4))
            enddo
          close(1)
        endif
        CALL plt_gsr(FLAG,ARRAY_PICT,S,2*JLAG+1,KLAG,RECT)
      endif

      RETURN
      END
```

```
*********************************************************************
*
* THIS SUBROUTINE ESTIMATES THE PARAMETERS OF AN ARMA(P,Q) FILTER
* GIVEN THE ARMA SEQUENCE USING APPROXIMATE LIKELIHOOD ESTIMATION
* (i.e. I.C. OF Z'S ARE ZEROS). IT ALSO CALCULATES THE COVARIANCE
* AS WELL AS THE RESIDUALS (Ei) WHICH ARE OUTPUTTED TO A FILE FOR
* FURTHER ANALYSIS.
** THE RECURSIVELY CALCULATED PARAMETERS ARE RETURNED IN TH(N,P+Q).
* WHERE N, P, & Q ARE THE SEQUENCE LENGTH, AR ORDER, AND MA ORDER.
*
*********************************************************************
       SUBROUTINE MLE
      IMPLICIT NONE
      include hd:include files:utilities.inc

      INTEGER*4 toolbx
      integer as,ps            !array size and param vector size respec.
      parameter(as=4000,ps=15)
      REAL Z(-ps:as),THETA(ps),X(ps,0:as),A(ps,ps),EI(-ps:as),H(ps)
      REAL G(ps),ASTR(ps,ps),D(ps),GSTR(ps),HSTR(ps),EIPD(-ps:as)
      REAL THEOLD(ps),EOLD(-ps:as),ENEW(-ps:as),Z1(ps) !,TH(as,ps)
      REAL COVAR(ps,ps),SOLD,SNEW,RCOND,DET(2),VAR
      INTEGER IPVT(ps),T,P, Q,DUM,I,K,j,STATUS,EOF
      PARAMETER(EOF = -1)
      CHARACTER*64 FILENAME, PROMPT, ORIGNAME, char_mtrx(11)
      REAL PI, EPS, F2, DELTA, ed_fld_data(3)
      INTEGER PIMAX, ITERMAX, LDA, ITER,N,PD,WINDOW,HTE
      integer*2 rect(4),itemtype
      integer misc(2),proc,dd,mm,yy,seconds,hours,minutes,flength
      character temp*13,temp1*65, str255*256
      logical cnclflag,exist, ITERFLAG, PIFLAG
      integer*4 appllimit      ! address of current aplication limit
      integer*4 newlimit       ! new application limit
      parameter (appllimit=z'00000130')
       DATA PI,EPS,PIMAX,F2,DELTA/.01,.01,1000,5,.01/

      newlimit = LONG(appllimit)   ! get the current limit
      newlimit = newlimit-102400   !allocate an additional 100k of stack
      call toolbx(SETAPPLLIMIT,newlimit) !set the new application limit

       LDA=PS
       ITER =1
      ITERFLAG = .FALSE.
      PIFLAG   = .FALSE.


 3    proc=0                        !calling proc. is mle
      ed_fld_data(1) = 0.           !AR order (p)
      ed_fld_data(2) = 0.           !MA order (q)
      ed_fld_data(3) = 1000    !ITERMAX (max # of iterations)
      call dlog(6000,3,4,ed_fld_data,1,2,2,misc,
     +            cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) GOTO 300
      p = int(ed_fld_data(1))
      q  = int(ed_fld_data(2))
      if ((p=0).and.(q=0)) GOTO 300
      if ((p+q)>ps) then
         proc=11
```

```
         write(temp,92) ps
         char_mtrx(1)=''
         char_mtrx(2)='p + q should not exceed '//temp
         char_mtrx(3)=''
         char_mtrx(4)=''
         call dlog(515,0,0,ed_fld_data,1,0,2,misc,
     +               cnclflag,proc,char_mtrx)
          GOTO 3
      endif
      ITERMAX= int(ed_fld_data(3))
      prompt = 'Select an input file for Maximum Likelihood Estimate:'
      call getfil("TEXT",filename,prompt)
      if (filename=' ') GOTO 300
      call setcur(4)
*
* read data from 'filename' file
*
      open(unit=2, file=filename)
      i=1; status=0
      while ((status<>eof).and.(i<as))
         read(2,*,IOSTAT=status) dum,z(i)
         i=i+1
      repeat
      if (i>= as) then
         proc=11
         write(temp,92) as
         char_mtrx(1)=''
         char_mtrx(2)='Number of points should not exceed '//temp
         char_mtrx(3)='The first '//trim(temp)//' points will be used.'
         char_mtrx(4)=''
         call dlog(515,0,0,ed_fld_data,1,0,2,misc,
     +               cnclflag,proc,char_mtrx)
      endif
      n = i-2
      close(2)
C
C INITIALIZE PARAMETER VECTOR THETA
C
       DO  I = 1, P+Q
          THETA(I) = 0.
      ENDDO
      DO I=-PS, 0
         EI(I) = 0.
         EIPD(I) = 0.
         Z(I) = 0.
         EOLD(I) = 0.
         ENEW(I) = 0.
      ENDDO

      call toolbx(INITDIALOGS,0)
      window=toolbx(FRONTWINDOW)
      pd = toolbx(GETNEWDIALOG,6200,0,-1)
      call toolbx(SETPORT, pd)
      call toolbx(SHOWWINDOW,pd)
      write(temp1,88) iter
      call toolbx(GETDITEM,pd,1,itemtype,hte,rect)
      call toolbx(SETITEXT,hte,str255(temp1))
```

```
C
C CALCULATION OF X
C
  1    CALL ECAL(THETA,Z,P,Q,N,EI,0)
       DO  I = 1, P+Q
          CALL ECAL(THETA,Z,P,Q,N,EIPD,I)
          DO  T = 1, N
             X(I,T) = (EI(T) - EIPD(T))/DELTA
          ENDDO
       ENDDO
C
C CALCULATION OF A  ( A = X'*X )
C
        DO  I = 1, P+Q
           DO  J = 1, P+Q
              A(I,J) = 0.
              DO T=1, N
                 A(I,J)= A(I,J) + X(I,T)*X(J,T)
             ENDDO
          ENDDO
       ENDDO
C
C CALCULATION OF G ( G = X'*E0 )
C
        DO  I = 1, P+Q
           G(I) = 0.
           DO  T = 1, N
              G(I) = G(I) + X(I,T)*EI(T)
          ENDDO
       ENDDO
C
C CALCULATION OF D
C
        DO  I = 1, P+Q
           D(I) = SQRT(A(I,I))
       ENDDO
C
C NORMALIZE A & CALCULATE ASTR AND GSTR
C
 2      DO  I = 1, P+Q
           DO  J = 1, P+Q
              ASTR(I,J) = A(I,J)/(D(I)*D(J))
          ENDDO
           ASTR(I,I) = 1. + PI
           GSTR(I) = G(I) / D(I)
       ENDDO
C
C SOLVE FOR HSTR ( ASTR * HSTR = GSTR ) UTLIZING LINPAC SUBROUTINES
C
        CALL SGECO(ASTR,LDA,P+Q,IPVT,RCOND,Z1)
        CALL SGEDI(ASTR,LDA,P+Q,IPVT,DET,Z1,1)
        DO  I = 1, P+Q
           H(I) = 0.
           DO  J = 1, P+Q
              H(I) = H(I) + ASTR(I,J) * GSTR(J)/D(I)
          ENDDO
       ENDDO
C
```

```
C CALCULATE SOLD AND SNEW (SUM OF ERRORS SQUARED)
C
       DO I = 1, P+Q
          THEOLD(I) = THETA(I)
          THETA(I) = THETA(I) + H(I)/10
      ENDDO
      CALL ECAL(THETA,Z,P,Q,N,ENEW,0)
      CALL ECAL(THEOLD,Z,P,Q,N,EOLD,0)
       SOLD = 0.
       SNEW = 0.
       DO  T = 1, N
          SOLD = SOLD + EOLD(T)**2
          SNEW = SNEW + ENEW(T)**2
      ENDDO
      write(temp1,88) iter
 88   format('Iteration : ',I4)
      call toolbx(GETDITEM,pd,1,itemtype,hte,rect)
      call toolbx(SETITEXT,hte,str255(temp1))
      write(temp1,89) snew
 89   format('Noise sum of squares : ',f10.1)
      call toolbx(GETDITEM,pd,2,itemtype,hte,rect)
      call toolbx(SETITEXT,hte,str255(temp1))
C
C CALCULATION OF COVARIANCE MATRIX
C
       CALL SGECO(A,LDA,P+Q,IPVT,RCOND,Z1)
       CALL SGEDI(A,LDA,P+Q,IPVT,DET,Z1,1)
       DO  I = 1, P+Q
          DO  J = 1, P+Q
             COVAR(I,J) = SOLD*A(I,J)/FLOAT(N-P-Q)
        ENDDO
      ENDDO
C
C TEST FOR CONVERGENCE
C
       IF (SNEW.LT.SOLD)  THEN
          DUM = 0.
          DO I = 1, P+Q
             IF (ABS(H(I)).GT.EPS) THEN
                DUM = 1.
             ENDIF
          ENDDO
          IF (DUM.EQ.0.) THEN
C            DO I = 1, P+Q
C               WRITE(9,*) (COVAR(I,J),J=1,P+Q)
C           ENDDO
             GOTO 200                        ! DONE
          ELSE
             ITER = ITER +1
            IF (ITER.GT.ITERMAX) THEN
               ITER = ITER -1
               ITERFLAG=.TRUE.
              GOTO 200                       ! TERMINATE CALCULATION
            ENDIF
             PI = PI/F2
          ENDIF
       ELSE
          PI = PI * F2
```

```
            IF (PI.GT.PIMAX) THEN
               PIFLAG=.TRUE.
              GOTO 200                        ! TERMINATE CALCULATION
            ENDIF
            IF (ITER.GT.ITERMAX) THEN
               ITERFLAG=.TRUE.
              GOTO 200                        ! TERMINATE CALCULATION
            ENDIF
            GOTO 2
         ENDIF
         GOTO 1

*
* prompt the user for output files and save data to them
*
 200    call toolbx(DISPOSEDIALOG, pd)
        call toolbx(SETPORT, window)
        proc=12                         !calling proc. is doarma
        call date(mm,dd,yy)
        write(temp,90) mm,dd,yy
  90    format(I2,2('/',I2))
        char_mtrx(1) = temp                   ! date
        call time(seconds)
        hours = seconds/3600;    seconds= mod(seconds,3600)
        minutes = seconds/60;    seconds= mod(seconds,60)
        write(temp,94) hours,minutes,seconds
  94    format(I2,2(':',I2))
        char_mtrx(2) = temp                   ! time
        write(temp,91) snew
  91    format(f7.1)
        char_mtrx(3) = temp                   ! noise sum of squares
        write(temp,92) n-p-q
  92    format(I4)
        char_mtrx(4) = temp                   ! degrees of freedom
         var = 0
         do i = 1, n
           var  = ei(I)**2 + var
         enddo
        write(temp,103) sqrt(var/n)
103     format(G10.3)
        char_mtrx(5) = temp                   ! residuals STD
        write(temp,92) iter
        char_mtrx(6) = temp                   ! # of iterations
        char_mtrx(7) = ''
        do i=1, p
           write(temp,93) theta(i),sqrt(covar(i,i))
           char_mtrx(7) = trim(char_mtrx(7))//temp
                                 ! AR parameters and their STD
  93       format('(',f5.2,',',f5.2,')')
        enddo
        char_mtrx(8) = ''
        do i=p+1, q+p
           write(temp,93) theta(i),sqrt(covar(i,i))
           char_mtrx(8) = trim(char_mtrx(8))//temp
                                 ! MA parameters and their STD
        enddo
        if (iterflag) then
           char_mtrx(9) = '   No convergence. ITER exceeded max.'
```

```
      elseif ( piflag) then
         char_mtrx(9) = '    No convergence. PI exceeded max.'
      else
         char_mtrx(9) = '                   Parameters converged'
      endif
      char_mtrx(10) = 'residuals.dat'
      char_mtrx(11) = 'MLE.dat'
      call setcur(0)
      call dlog(6100,0,6,ed_fld_data,1,2,7,misc,
     +           cnclflag,proc,char_mtrx)
      if (.NOT.cnclflag) GOTO 300
      call setcur(4)
      if ((proc.and.B'001')=1) then ! proc is a flag to indicate
         filename=char_mtrx(10)           ! the selected options.
         flength = ichar(filename(1:1))
         filename = filename(2:flength+1)
         inquire (file=filename,exist=exist)
         if (exist) then
            prompt = "Save residuals as:"
            origname = filename
            call setcur(0)
            call putfil(filename,prompt,origname)
            call setcur(4)
            if (filename=' ')  GOTO 201
         endif
         open(1,file=filename,status='new')
         do i=1 ,n
            write(1,*) i-1,ei(i)
         enddo
         close(1)
      endif

201   if (( proc.and.B'010')=2) then
         filename=' '
         filename=char_mtrx(11)           ! the selected options.
         flength = ichar(filename(1:1))
         filename = filename(2:flength+1)
         inquire (file=filename,exist=exist)
         if (exist) then
            prompt = "Save MLE report as:"
            origname = filename
            call setcur(0)
            call putfil(filename,prompt,origname)
            call setcur(4)
            if (filename=' ')  GOTO 202
         endif
         open(1,file=filename,status='new')
         write(1,95) trim(char_mtrx(1)),trim(char_mtrx(9)),
     +               trim(char_mtrx(2))
95       format(4x,'Date: ',A,5x,A,5x,'Time: ',A/)
         write(1,96) char_mtrx(3),char_mtrx(4)
96       format(/,4x,'Noise sum of squares: ',A,T40,
     +                        'Degrees of freedom: ',A/)
         write(1,97) char_mtrx(5),char_mtrx(6)
97       format(4x,'Residuals standard error: ',A,T40,
     +          'Number of iterations: ',A/)
         write(1,100)
100      format(4x,'AR parameters (phi,std)')
```

```
        j=1
        while (j*5-4<=p)
           temp1=''
           do i=j*5-4, min(5*j,p)
             write(temp,98) theta(i),sqrt(covar(i,i))
              temp1 = trim(temp1)//temp
           enddo
           write(1,102) temp1
           j=j+1
        repeat
102     format(4x,A)
98      format(5('(',f5.2,',',f5.2,')', :)/)

        write(1,99)
99      format(/4x,'MA parameters (theta,std)')
        j=1
        while (j*5-4+p<=p+q)
           temp1=''
           do i=j*5-4+p, min(5*j+p,p+q)
             write(temp,98) theta(i),sqrt(covar(i,i))
             temp1 = trim(temp1)//temp
           enddo
           write(1,102) temp1
           j=j+1
        repeat
        close(1)
     endif
202  if ((proc.and.B'100')=4) then
        if ((proc.and.B'010')=2).AND.(filename<>' ') then
           call spool(filename,0,1)
        else
           filename = 'MacModel.scratch'
           open(1,file=filename,status='new')
        write(1,95) trim(char_mtrx(1)),trim(char_mtrx(9)),
    +                 trim(char_mtrx(2))
           write(1,96) char_mtrx(3),char_mtrx(4)
           write(1,97) char_mtrx(5),char_mtrx(6)
           write(1,100)

           j=1
           while (j*5-4<=p)
             temp1=''
             do i=j*5-4, min(5*j,p)
                write(temp,98) theta(i),sqrt(covar(i,i))
                temp1 = trim(temp1)//temp
             enddo
             write(1,102) temp1
             j=j+1
           repeat

           write(1,99)
           j=1
           while (j*5-4+p<=p+q)
             temp1=''
             do i=j*5-4+p, min(5*j+p,p+q)
                write(temp,98) theta(i),sqrt(covar(i,i))
                temp1 = trim(temp1)//temp
```

```
            enddo
            write(1,102) templ
            j=j+1
          repeat
          close(1)
          call spool(filename,0,1)
        endif
      endif
 300  call setcur(0)
      newlimit = LONG(appllimit)          ! get the current limit
      newlimit = newlimit+102400          ! allocate an additional 100k
of stack
      call toolbx(SETAPPLLIMIT,newlimit)  ! set the new application
limit

      RETURN
       END
*********************************************************************
*****
*****     SUBROUTINE TO CALCULATE THE RESIDUALS
*****
*********************************************************************

       SUBROUTINE ECAL(THETA,Z,P,Q,N,E,I)
      IMPLICIT NONE

      integer as,ps            !array size and param vector size respec.
      parameter(as=4000,ps=15)
       REAL THETA(ps), E(-ps:as), Z(-ps:as),DELTA,E1,E2
      PARAMETER(DELTA=.01)
       INTEGER P,Q,T,J,K,N,I

       IF (I.NE.0) THETA(I) = THETA(I) + DELTA
      DO T=1,N
         E1 = 0.
         DO  J =1, P
            IF ((T-J).GE.0.)  E1 = E1 + Z(T-J) * THETA(J)
         ENDDO
         E2 = 0.
         DO  K = 1,Q
            IF ((T-K).GE.0.) E2= E2 + E(T-K) * THETA(P+K)
         ENDDO
          E(T) = Z(T) - E1 + E2
      ENDDO
       IF (I.NE.0) THETA(I) = THETA(I) - DELTA
       RETURN
       END

*********************************************************************
*****
*****     SUBROUTINES FROM LINPAC TO CALCULATE THE INVERSE
*****
*********************************************************************

      subroutine sgedi(a,lda,n,ipvt,det,work,job)
      subroutine sgeco(a,lda,n,ipvt,rcond,z)
```

VITA

Ali Y. Chaaban

Candidate for the Degree of

Master of Science

Thesis: A SOFTWARE PACKAGE FOR SYSTEM IDENTIFICATION

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Tyre, Lebanon, June 15, 1963, the son of Youssef H. and Amal S. Chaaban.

Education: Graduated from Sidon National School, Sida, Lebanon, in May 1983; received Bachelor of Science Degree in Electrical Engineering from University of Tulsa, Tulsa, Oklahoma, in December 1987; completed requirements for the Master of Science degree at Oklahoma State University in December, 1991.

Professional Experience: Teaching Assistant, Department of Electrical Engineering, Tulsa University, January, 1988, to May 1988; Teaching Assistant, Department of Electrical Engineering, Oklahoma State University, September, 1988, to May 1990; Data Communication Specialist, Computer Center, Oklahoma State University, June, 1990, to August 1991.