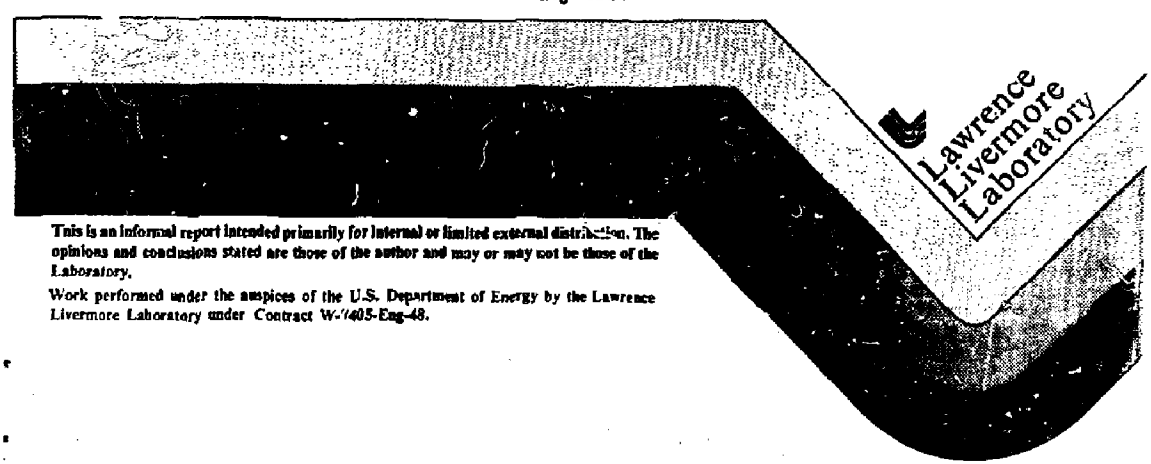
UCID- 18744

MXLKID: A Maximum Likelihood Parameter Identifier

D. T. Gavel

July 1980



MXLKID

A MAXIMUM LIKELIHOOD PARAMETER IDENTIFIER

D. T. GAVEL

FOREWORD

The MXLKID (MaXimum Parameter LiKelihood IDentifier) computer code was written and developed under the Lawrence Livermore National Laboratory (LLNL), Electrical Engineering Department Engineering Research Funds as part of the Signal Processing Project. Many LLNL projects are involved in experimental design and control which involves the fitting of nonlinear and linear dynamic models representing the physical phenomonology to noisy measurement data. MXLKID represents a state-of-the-art algorithm which is very general and powerful.

The generation of this code began after the development of a nonlinear dynamic estimator code called DYNEST[4]. MXLKID essentially is a set of sophisticated optimization algorithms looped around the DYNEST code. Many people have been responsible for the development of DYNEST and therefore have contributed implicitly to the development of MXLKID. We would especially like to acknowledge the initial structural coding and DYNEST conversion by D. Freeman under the direction of J. Candy.

Work continues on the development of MXLKID in the Signal Processing Project to be able to identify large numbers (>40) of unknown parameters from noisy data. This effort will enable potential users to identify parameter in nonlinear distributed (partial differated equation) systems.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

ABSTRACT

MXLKID (MaXimum LiKelihood IDentifier) is a computer program designed to identify unknown parameters in a nonlinear dynamic system. Using noisy measurement data from the system, the maximum likelihood identifier computes a likelihood function (LF). Identification of system parameters is accomplished by maximizing the LF with respect to the parameters.

In the main body of this report, we briefly summarize the maximum likelihood technique and give instructions and examples for running the MXLKID program. MXLKID is implemented in LRLTRAN on the CDC7600 computer at LLNL. We include a detailed mathematical description of the algorithm in the appendices.

ACKNOWLEDGMENTS

The author would like to gratefully acknowledge J. Candy for his help during the development of MXLKID and the writing of this report. Thanks are also due to H. Cabayan and J. Zicker for applying MXLKID to a real problem of interest, thus providing me the opportunity to test the code and also gain an understanding of the needs of potential users. Special recognition must be given to D. Freeman who wrote the initial version of this code and to D. Lager who helped with program debugging.

1.0 INTRODUCTION

This report introduces a new tool for dynamic system analysis to the signal processing software library at LLL, the MaXimum LiKelihood parameter IDentifier (MXLKID). MXLKID identifies unknown parameters of a dynamic system from noisy measurement data.

Dynamic Systems are fundamentally important in the work of research scientists and engineers. Systems involving Newton's laws of motion, electromagnetics, chemical kinetics, structural vibrations, and gas dynamics are familiar examples of dynamic systems. Even so-called static systems can be considered dynamic systems at rest. Thus dynamic system modeling and analysis is an important field of interest to researchers desiring to know more about the universe around them.

System identification is a key problem in systems analysis. For many scientific and engineering systems there is no universal theory explaining the system structure nor is there any direct method of measuring unknown system parameters. All that is available to an experimenter is noisy measurement data which may only be indirectly related to the system parameters of interest. MXLKID is a useful tool for system parameter identification in this environment. While MXLKID will not perform the entire task of determining the structure and order of a system, it will numerically estimate unknown parameters in a system equation, using the information inherent within the measurement data.

To use MXLKID, the user must specify the system dynamic equation(s) in terms of a set of unknown parameters, and give initial (a-priori) estimates for these parameter values. MXLKID then iteratively improves the parameter estimates until the maximum likelihood set of parameters is reached. Likelihood, explained in further detail in Appendix A, is a useful objective function in the search for optimal parameter estimates because it serves as a measure of model validity in terms of consistency with the measured data.

This report is intended to provide the user with enough information to run the MXLKID program successfully and to gain insight from the results. The user is assumed familiar with FORTRAN and the Octopus CDC7600 computers.

The report's main body is organized into four chapters. This introduction comprizes Chapter 1. In Chapter 2 we define the basic MXLKID problem and give an overview of the algorithm. Chapter 3 gives detailed instructions and examples for set up and execution of the MXLKID program. Chapter 4 is a summary of the report.

The appendices of this report contain additional detailed information for the serious user of the MXLKID program. Appendix A describes the algorithm in greater detail, and gives mathematical derivations for the basic formulas used. In Appendix B, we present the details of the statistical whiteness test which is used to independently verify MXLKID results.

2.0 Basic MXLKID Algorithm

In this section we describe the basic parameter estimation algorithm found in the MXLKID program. We define the mathematical problem to be solved, present the algorithm designed to solve it, and then discuss the actual implementation.

2.1 Problem Definition

The MXLKID is a parameter estimation/identification algorithm designed to solve the following problem:

Given a continuous nonlinear dynamical system in state space form,

$$\dot{x}_t = f(x_t, \theta_t) + g(u_t, \theta_t) + w_t$$

and discrete measurement system,

$$z_k = h(x_k, \theta_k) + v_k$$

where x, u, z, θ are the n-state, r-input, m-measurement, and p-parameter vectors; $f(\cdot)$, $g(\cdot)$, $h(\cdot)$ are the respective vector functions; and w_t, v_k are white Gaussian processes with respective covariances Q and R, find the best (maximum likelihood) estimate $\hat{\theta}$ of θ.

The algorithm designed to solve this problem uses a recursive state estimator (Kalman filter) to generate quantities required to

compute the likelihood function. The likelihood (or negative log-likelihood) function is maximized (minimized) to find the "best" set of unknown parameters θ. Some of the fundamental references for maximum likelihood identification are Gupta and Mehra [1], Kashyap [2], and Best [3]. We now discuss the basic algorithm.

2.2 Algorithm

MXLKID is the implementation of some standard optimization techniques using a recursive estimator (Kalman filter) to generate some of the required quantities. The optimization algorithms implemented are basically gradient search techniques using the Levenberg-Marquardt or Gauss-Newton methods. All the methods involve a technique similar to the one shown below:

Log Likelihood Function Calculation:

$$J(\underline{\theta}) = -1/2\, \ell n\ (2\pi) - 1/2 \sum_{i=1}^{N} \varepsilon^T(i,\hat{\underline{\theta}}_{OLD})\ (R^{\varepsilon}(i,\hat{\underline{\theta}}_{OLD}))^{-1}\ \varepsilon(i,\hat{\underline{\theta}}_{OLD}) + \ell n \left| R^{\varepsilon}(i,\hat{\underline{\theta}}_{OLD}) \right| \quad * \qquad (3)$$

Gradient Calculation:

$$\frac{\partial J}{\partial \underline{\theta}}^{T} = \left[\frac{\partial J(\theta)}{\partial \theta_{\ell}}\right] \approx \frac{J(\hat{\underline{\theta}}_{OLD} + \Delta\hat{\underline{\theta}}_{\ell}) - J(\hat{\underline{\theta}}_{OLD})}{\Delta\theta_{\ell}} \qquad \ell = 1, \ldots, p \qquad (4)$$

* The quantities $\varepsilon(i,\underline{\theta})$ and $R^{\varepsilon}(i,\hat{\underline{\theta}})$, in these equations are generated recursively by the Kalman filter.

Hessian Calculation:

$$\frac{\partial^2 J(\theta)}{\partial \underline{\theta}^2} := \left[\frac{\partial^2 J(\hat{\theta}_{OLD})}{\partial \hat{\theta}_j \, \partial \hat{\theta}_k}\right] = \sum_{i=1}^{N} \frac{\partial \varepsilon(i,\hat{\theta}_{OLD})}{\partial \hat{\theta}_j} (R_i^{\varepsilon})^{-1} \frac{\partial \varepsilon(i,\hat{\theta}_{OLD})}{\partial \theta_k} +$$

$$+ 1/2 \text{ tr} \left[(R_i^{\varepsilon})^{-1} \frac{\partial R_i^{\varepsilon}}{\partial \theta_j} (R_i^{\varepsilon})^{-1} \frac{\partial R_i^{\varepsilon}}{\partial \theta_k}\right]$$

$$+1/4 \text{ tr} \left[(R_i^{\varepsilon})^{-1} \frac{\partial R_i^{\varepsilon}}{\partial \theta_j}\right] \text{tr} \left[(R_i^{\varepsilon})^{-1} \frac{\partial R_i^{\varepsilon}}{\partial \theta_k}\right] \qquad (5)$$

Parameter Estimate Update:

$$\hat{\underline{\theta}}_{NEW} = \hat{\underline{\theta}}_{OLD} + \rho \left[\frac{\partial 2 J(\theta)}{\partial \underline{\theta}} + \beta D^2\right]^{-1} \frac{\partial J}{\partial \underline{\theta}}$$

Loop:

$\hat{\underline{\theta}}_{NEW} \rightarrow \underline{\theta}_{OLD}$

where

$J(\underline{\theta})$ is the scalar negative log-likelihood function

$\frac{\partial J}{\partial \underline{\theta}}$ is the p-gradient vector

$\frac{\partial^2 J(\theta)}{\partial \underline{\theta}^2}$ is the pxp Hessian matrix

$\underline{\theta}$ is the p-parameter vector

$\Delta\theta_{\ell}$ is the p-incremental parameter change vector (only ℓ^{th} element nonzero)

$\underline{\varepsilon}$ is the m-innovation vector

R^{ϵ} is the mxm innovation covariance matrix

D is the pxp diagonal Marquardt matrix (contains the square root of the diagonal elements of $\left(\frac{\partial^2 J(\theta)}{\partial \theta^2}\right)$

ρ is the scalar step adjustment

β is the Marquardt parameter used to weight the diagonal Marquardt Matrix

If the Gauss-Newton algorithm is selected, then β is set to zero, while in the Levenberg-Marquardt option it is variable. ρ is a step size parameter which is set to one (ρ=1) initially, but is varied during the algorithm to ensure a reduction in the negative log likelihood.

MXLKID consists essentially of these equations in an iterative loop with the recursive Kalman filter providing the quantities for equations (3) and (5). Note that the dynamic process, measurement and noise models of (1) and (2) are included in the Kalman filter formulation [4,5,6]. For the interested reader, detailed equations for both the minimization algorithms and the Kalman filter are presented in Appendix A.

2.3 Algorithm Implementation

In this section we discuss the implementation of the MXLKID algorithm. We first overview the major tasks and then describe (simply) the program flow.

The main task of MXLKID is to identify the set of parameters which maximizes the likelihood function. To compute likelihood, we use both measurement data and a model of the system (equations (1) and (2)) of interest. The system model, provided by the user in a set of user-definable subroutines, is written in terms of the parameters which need to be identified. This model serves as a reference in a statistical, model-based, signal processing scheme (Kalman filter). Processing the measurement data results in the necessary ingredients for computing the likelihood function according to equation (3).

The Likelihood is generally a nonlinear function of the unknown parameters because results of the signal processing are dependent on the system model. To maximize likelihood, one of many well known nonlinear programming algorithms can be employed. The user currently has a choice of two such algorithms within MXLKID, Gauss-Newton and Levenberg-Marquardt (see Appendix A, Section 3 for descriptions of these optimization methods). Both of these methods rely on computing the likelihood several times during the search for a maximum.

MXLKID implementation is depicted (simply) in Figure 2-1. Note that after the initial parameters are set, the log-likelihood function is calculated using the Kalman filter to produce the innovations data.* The log-likelihood Jacobian (partial derivative) is numerically calculated as are the Jacobians used in (5). All of this information is passed to the parameter estimator (gradient optimization algorithm). Once covergence is achieved, a

* The Kalman filter is basically used as a "whitening" filter in this application, i.e., correlated measurements are input and uncorrelated innovations are output.

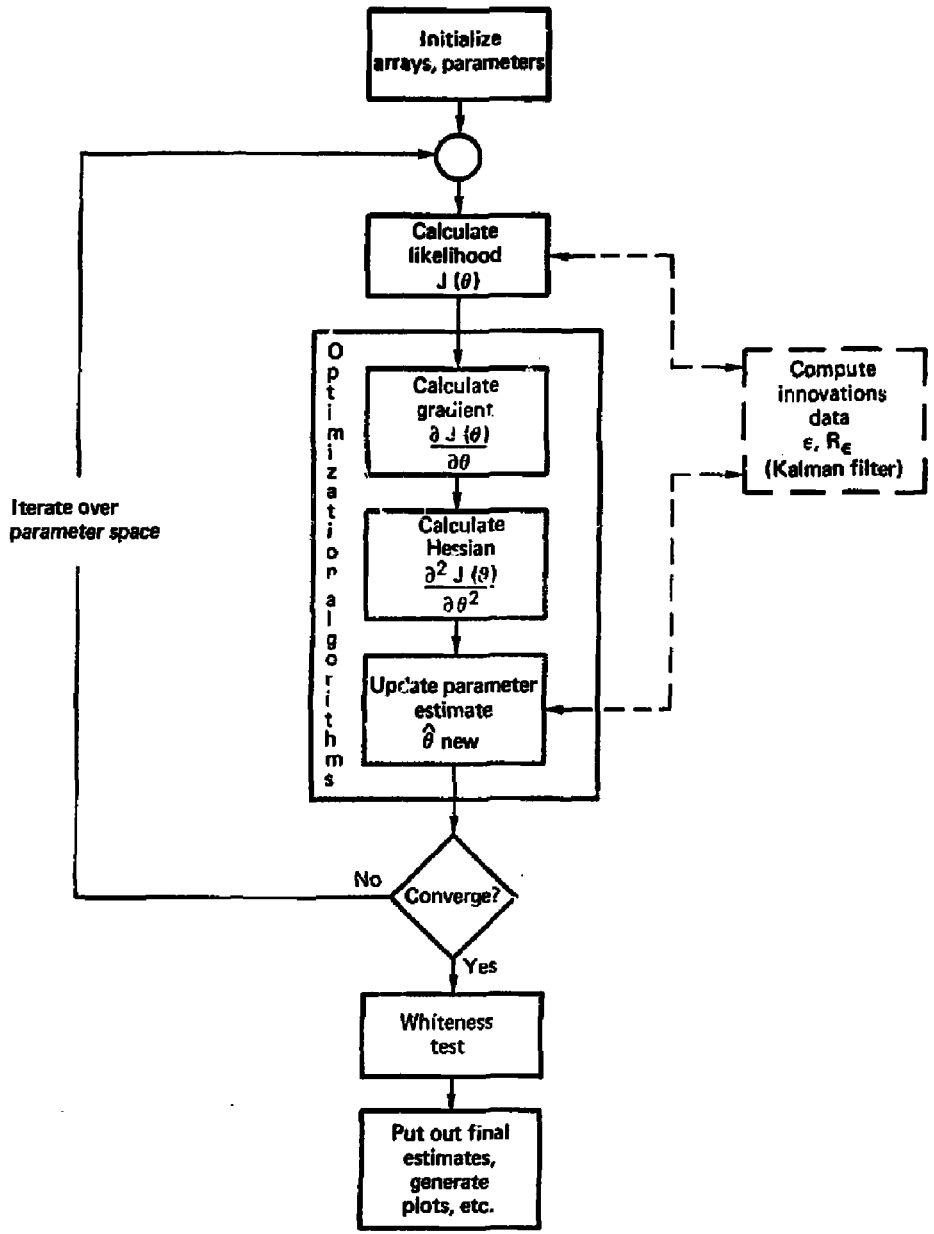


Figure 2-1. MXLKID Implementation

whiteness test is applied to the final innovations sequence as an independent verification of the identified model's validity.* Final results are in graphical and numerical form on output disk files.

2.4 Program Structure

The relationships of the major MXLKID subroutines are shown in Figure 2-2. We see that MAIN is the main drive for the program in which I/O assignments are made, parameters are initialized, and input files are read before control is passed to MINI, the subroutine controlling the numerical optimization process. MINI directs all the calculations necessary to step the parameter estimates toward a maximum likelihood calculation. An important calculation is the negative log-likelihood function (NLLF) evaluation, since the NLLF is the cost function which needs to be minimized. The NLLF is calculated in subroutine CFUNC using the formula give in equation (3).

Calculation of the NLLF requires that measurement data be processed through a Kalman filter (KF). The KF code is contained in subroutine FILTER, which is a block of code taken from the generalized extended Kalman filter program, DYNEST.[4] Each NLLF calculation requires a separate KF run over the data , so CFUNC, and hence FILTER, is called many times during the course of a search for a maximum likelihood solution.

The structure of the KF code is explained in the DYNEST manual, however, we show seven major modules in the organization chart under FILTER because their operation determines the behavior of

* The whiteness test is discussed in detail in Appendix B.

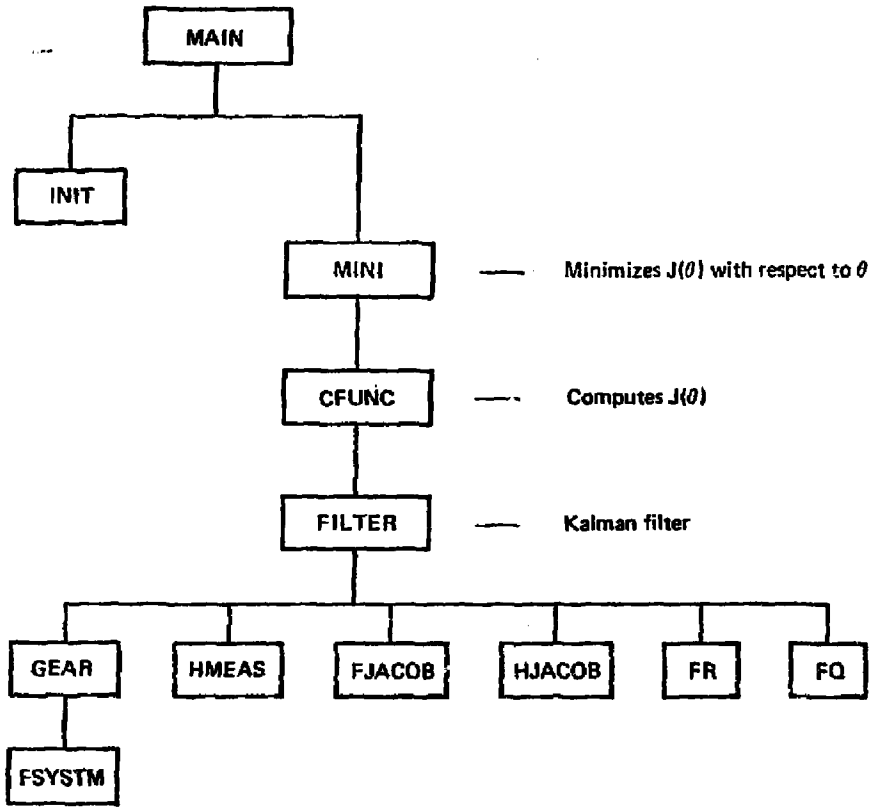


Figure 2-2. MXLK Program Subroutine Hierarchy

the dynamic system model. Six of the subroutines, FSYSTM, HMEAS, FJACOB, HJACOB, FR, and FQ must be modified by the user to reflect the operation of the dynamic system of interest. These modifications are explained, with examples, in Chapter 3. Essentially, GEAR is a numerical integrator (differential equation solver), FSYSTM provides the equations for system dynamics and HMEAS gives the equations describing the measurement process.

FJACOB and HJACOB are subroutines for analytic evaluation of system and measurement Jacobian matrices, respectively. (Details for programming the Jacobian evaluators are given in Chapter 3.) FQ and FR determine the statistics of system driving and measurements noises, respectively.

This concludes our brief overview of the MXLKID program. We now proceed to Chapter 3, where details in program implementation are given. Should the user find a need for a more in depth discussion of the maximum likelihood algorithm, Appendix A gives the derivations of the formulas used in MXLKID.

3.0 PROGRAM PREPARATION AND EXECUTION

In this chapter we show the MXLKID setup for a typical parameter identification problem. We discuss the fundamental code requirements, the user-definable subroutines and program execution. We also discuss a sample problem which is used throughout to show the user how to prepare and execute the MXLKID algorithm.

3.1 Program Requirements

The MXLKID program input/output requirements are shown in Figure 3-1. Two input files must be prepared by the user: a measurement file containing system measurement data, and a program control file, which contains all the various problem defining parameters, option selections, and initial conditions. Program output is sent to three places, the teletype terminal (optional), an output file, and a plot file. The teletype prints out parameter estimates as the MXLKID algorithm progresses iteratively to an optimal solution. The output file is filled with additional information, mostly the results from intermediate calculations. The plot file receives various data plots which can be useful later for problem diagnosis.

To prepare and run MXLKID for a particular problem, the following steps must be taken:

1. Prepare the INPUT file

2. Prepare the MEASUREMENT file

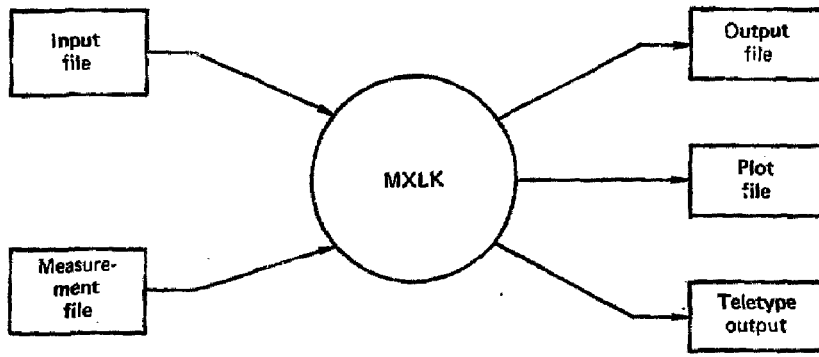


Figure 3-1. MXLK Input/Output Requirements

3. Prepare the problem dependent SUBROUTINES:

   FSYSTM
   HMEAS
   FJACOB
   HJACOB
   FR
   FQ

4. Execute MXLKID

5. Obtain OUTPUTS

The remainder of this chapter discuss the above steps in detail to obtain a MXLKID solution for a given problem. We first define the sample problem to be used in Section 3.2 and then follow the steps for problem solution. INPUT file requirements are discussed in Section 3.3 which includes the necessary MEASUREMENT file data. Section 3.4 discusses the preparation of the user-defined SUBROUTINES and then program execution is discussed in Section 3.5. The program OUTPUTS for the sample problem are presented in Section 3.6

3.2 The Sample Problem

A sample maximum likelihood identification problem is outlined in Table 3-1. This example is from the report by J. F. Best.[3]

The system is a two-pole resonator whose frequency and damping ratio are unknown. The waveform shown in Figure 3-2 is the driving function, which, for this problem, we assume is completely known. The measurement data available is the white noise corrupted signal shown in Figure 3-3.

TABLE 3.1

SAMPLE PROBLEM DEFINITION

System: (equation (1))

$$\dot{\underline{x}}(t) = \begin{bmatrix} 0 & 1 \\ -\theta_2^2 & -2\theta_1\theta_2 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 5\theta_3 \end{bmatrix} \underline{u}(t) + \underline{w}(t)$$

Measurement: (equation (2))

$$z_j = \begin{bmatrix} \theta_4 & 0 \end{bmatrix} \underline{x}(t_j) + v_j;\ j = 1, \ldots, K_T$$

where w is white Gaussian driving noise:

$$\text{cov}(w) = \begin{bmatrix} \theta_5^2 & 0 \\ 0 & \theta_5^2 \end{bmatrix} =: Q$$

u(t) is a known input (see Figure 3-2)

and v is white Gaussian measurement noise:

$$\text{cov}(v) = \theta_6^2 =: R$$

True Parameter Set:

$$\theta = [0.6,\ 3.14,\ 4.0,\ 2.0,\ 0.0,\ 0.1]^T$$

Initial Parameter Guesses:

$$\theta_1 = 0.1,\ \theta_2 = 9.0$$

(All other parameters are at their true values and assumed known for this example.)

Problem: Find the maximum likelihood estimate of the parameters, θ_1 and θ_2, given the set of measurement data $\{z_j;\ j = 1, \ldots, K_T\}$ (see Figure 3-3).

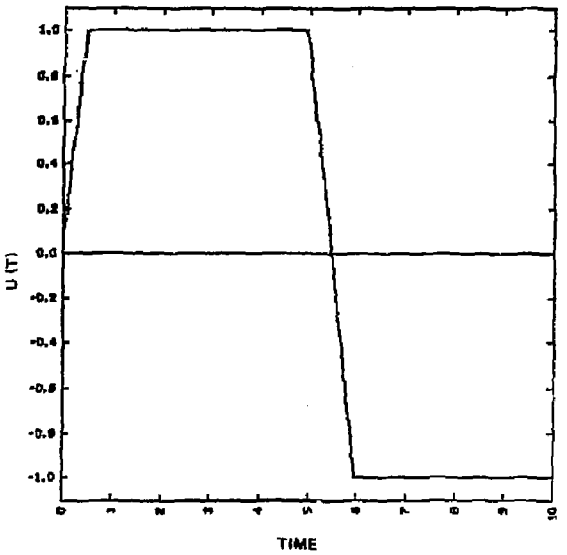


Figure 3-2 System Driving Function, u(t)

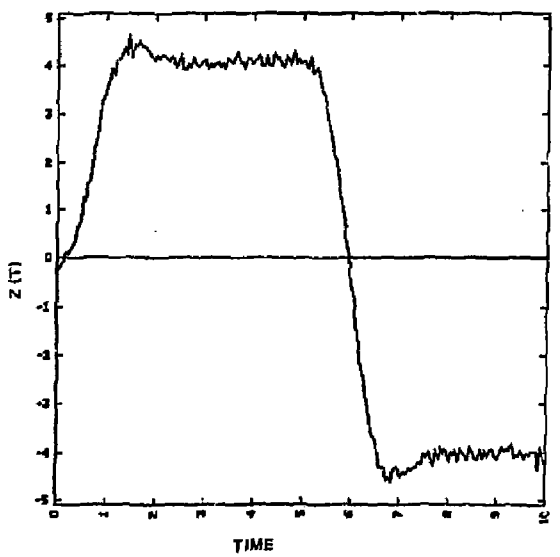


Figure 3-3. Simulated Measurements, z(t)

The data shown in Figure 3-3 was actually generated by a separate simulator program, which is designed to model the two pole resonator with the parameters set to their true values. The simulator adds Gaussian random noise to the simulated measurements.

We start the identification process with initial guesses for the unknown parameters (frequency and damping) of the system. (In this example the guesses are quite far from the true values.) In the next few sections we explain how to set up the MXLKID program for this problem, and how to interpret the results. The reader shall see that, for this example, MXLKID converges to the true parameter values very quickly.

3.3 Program Inputs

Two data sets must be prepared by the user for input to MXLKID: a measurement data file and an input file.

3.3.1 MEASUREMENT File

The MEASUREMENT file contains the system measurement data and the time values at each measurement point. Table 3-2 shows an example. The first data point is taken at time 0, and the measurement value is -2.965E-01. (Since the time value is read in at each measurement point, there is no need for the measurement points to be equally spaced in time.) The format for the measurement file is user-definable; it is read from the input file. For the example shown, the format is 2E12.3.

Table 3-2. Measurement File (First Ten Entries)

1	0.	-2.965E-01
2	5.000E-02	-1.974E-01
3	1.000E-01	-8.679E-02
4	1.500E-01	8.140E-02
5	2.000E-01	7.056E-02
6	2.500E-01	1.716E-01
7	3.000E-01	2.522E-01
8	3.500E-01	3.009E-01
9	4.000E-01	5.244E-01
10	4.500E-01	7.735E-01

Table 3-3. Input File

```
 1 2,1,                                   N,M
 2 50,.05,                                KT,DLT0
 3 0.,0.,                                 Q
 4 0.,                                    R
 5 9.0E-8,-3.4E-7,-3.4E-7,9.0E-6,         PI
 6 (2E12.3)                               MEAS FMT
 7 0.,0.,                                 X0
 8 6,5,1.E-6,                             MAXITR,MAXTRY,EVMIN
 9 1.,.01,1.E-7,0.,                       OPTN(MMF,BETH,BMIN)
10 0.1,9.0,4.0,2.0,0.,0.1,                THETA0
11 2,1,2,                                 KTH,IA
12 1,1,                                   NP,MP
13 0.,14.,0.,                             CRIT
```

3.3.2 INPUT File

The INPUT file contains information needed to help define the problem, assign initial conditions, select program options, etc. Table 3-3 shows an example of an input file. Elements of the input file are described in detail below.

N - Number of states in the system dynamic model, i.e., the number of first order differential equations which define the system dynamics.

M - The number of measurements taken at each discrete point in time. Dimension of the measurement vector, Z.

KT - Total number of discrete time points at which measurement data is available.

DLTO - The initial time gap between measurements.

Q - Diagonal elements of the driving noise covariance matrix.

R - Diagonal elements of the measurement noise covariance matrix. Note: Q and R are initialized to zero for the example problem because they are parameterized by THETA (5) and THETA (6) during program execution.

PI - Initial state error covariance matrix. The elements are listed by columns: PI (1,1), PI (2,1)...PI (N,1), PI (1,2)...etc.

MEAS FMT - Measurement file format.

XO - Estimate of the initial state of the dynamic system.

MAXITR - This is the maximum number of steps that the maximum likelihood algorithm will take in parameter space. MAXITR should be larger than the expected number of steps before convergence, i.e., it prevents the program from looping forever. The estimated maximum number of negative log likelihood function (NLLF) evaluations is MAXITR*KTH, where KTH is the number of parameters to be identified.

MAXTRY - The maximum number of line search attempts in a given search direction. One NLLF evaluation is required for each line search try.

EVMIN - Cutoff factor for discarding small eigenvalues during Hessian inversion. This causes the minimization algorithm to ignore step directions in which little improvement of the NLLF is likely. Eigenvalues less than EVMIN*(maximum eigenvalue) are ignored. 1E-6 is a good value.

OPTN - Option array.
OPTN(1) - MMF - Minimization method flag. MMF=1: Gauss Newton method. MMF=2: Marquardt Method.
OPTN(2) - BETH - Initial Marquardt Parameter (used only if MMF=2).
OPTN(3) - BMIN - Minimum allowable value for the Marquardt parameter (used only if MMF=2).

THETAO - Initial parameter list.

KTH - Number of parameters to be identified.

IA - Index Array. This array specifies the locations in THETAO which contain a parameter to be varied. All the other parameters in THETAO remain constant. IA contains KTH elements.

NP - Iteractive Mode. NP=0 implies no interaction is desired, NP=-1 requests interaction only upon algorithm convergence or when MAXITR or MAXTRY is exceeded. NP > 0 calls for, in addition to the cases above, interaction once every NP'th interation (beginning with the first).

MP - Innovations plot selector. A plot of the innovations (data residuals is made every MP iterations (beginning with the first). MP=0 turns off the innovations plotter except during the first and last iteration.

CRIT - Convergence Criteria. Convergence is established when:

1) The Euclidian norm (sum of squared elements) of the gradient vector is less than CRIT(1), or

2) All the parameters, for two successive iterations, agree in at least CRIT(2) significant digits, or

3) The cost function (NLLF) for two successive iterations changes by less than CRIT(3).

3.4 Program Subroutine Preparation

User definable subroutines were mentioned briefly in section 2.3. These subroutines are to be modified by the user such that they define the system model of interest. The subroutines are FSYSTM, HMEAS, FJACOB, HJACOB, FR, and FQ. They contain the dynamic system model, measurement model, system Jacobian, measurement Jacobian, measurement noise model, and driving noise model, respectively.

The subroutines are set up by the user in exactly the same manner as explained in the DYNEST manual ([4] pp. 20, 25), with the exception that the models may now be specified in terms of the unknown parameters which are to be identified by MXLKID. The parameters are passed to the user subroutines in the array THETA, which contains the parameters in exactly the same order as they were specified in the input file. MXLKID automatically handles the modification of THETA as the program steps toward an optimal solution.

3.4.1 FSYSTM

FSYSTM is a subroutine containing the differential equations describing the system:

$$\dot{\underline{x}} = f(\underline{x}, \underline{\theta}, \underline{u}) \qquad (3.4\text{-}1)$$

where $\underline{x}$ is a vector of n states (n is the order of the system), $\underline{\theta}$ is the set of unknown parameters, and $\underline{u}$ is an input disturbance vector, either known, or given in terms of unknown parameters.

The arrays $\underline{x}$, $\underline{\theta}$, and $\underline{u}$ are passed to FSYSTM. The user must insert the appropriate coding for f(.) as shown in the example in Figure 3-4.

3.4.2 HMEAS

HMEAS contains the ideal, no noise, measurement equation:

$$z = h(\underline{x}, \underline{\theta}) \qquad (3.4\text{-}2)$$

Figure 3.5 shows sample coding.

3.4.3 FJACOB

The Kalman filter code requires the following Jacobian matrix*:

$$F = \frac{\partial}{\partial \underline{x}} f(\underline{x}, \underline{\theta}, \underline{u}) \qquad (3.4\text{-}3)$$

* Refer to Table A.1 in Appendix A for a full set of equations for the Kalman filter algorithm.

```
      FUNCTION FSYSTM(NN,TT,X,XDOT)
C
C     THIS ROUTINE IS USED BY THE GEAR PACKAGE, AND IT COMPUTES THE
C     SYSTEM FUNCTION AND THE ERROR COVARIANCE PROPAGATION ARRAY.
C     CONTROL IS DIRECTED THROUGH THE COMMON BLOCK PDIRECT.
C     THIS IS A NECESSARY ROUTINE THAT WAS PROBLEM DEPENDENT CODING.
C     THIS ROUTINE IS PART OF THE GEAR INTEGRATION PACKAGE.
C
      USE LKDCOM

      DIMENSION TT(1),X(1),XDOT(1)
      DIMENSION XD(NS)

                                  .
                                  .
                                  .

      DO 8 II=1,NN
    8 XD(II)=X(II)
      AMLT=0.

  SK2 CONTINUE
C
C  EVALUATE THE DERIVATIVE OF THE STATE
C     ************************************************************
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *       YOUR CODING FOR THE SYSTEM MODEL GOES HERE.  AMLT*W2 *
C     *       MUST BE ADDED TO THE MODEL.  PUT THIS SUM INTO ARRAY *
C     *       XDOT AS SHOWN BELOW.                               *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *       SET THE MODEL PARAMETERS.  THIS COULD BE DONE IN   *
C     *       ROUTINE INIT AND THE VARIABLES PASSED TO THIS ROUTINE
C     *       THROUGH COMMON.                                    *
C     *                                                          *
C     *                                                          *
C     *
      IF(TT.LT.CB(1))U=TT/CB(1)
      IF((TT.GE.CB(1)).AND.(TT.LT.CB(2)))U=1.0
      IF((TT.GE.CB(2)).AND.(TT.LT.CB(3)))U=1.0
     &          -2.0*(TT-CB(2))/(CB(3)-CB(2))
      IF(TT.GE.CB(3))U=-1.0

      PU(K)=U

      XDOT(1)=XD(2)
      XDOT(2)=-(THETA(2)**2)*XD(1)
     &        -2*THETA(1)*THETA(2)*XD(2)
     &        +B*THETA(3)*U
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     *                                                          *
C     ************************************************************
      RETURN
```

Figure 3-4. Function FSYSTM

```
      FUNCTION HMEAS(HH,TT,X,HH)
C
C     THIS ROUTINE COMPUTES THE MEASUREMENT MODEL AS A FUNCTION OF THE
C     STATE VECTOR X AND THE TIME TT
C     THIS IS A NECESSARY ROUTINE WHICH IS PROBLEM DEPENDENT.
C
      USE LADCOM
C
      DIMENSION HH(1),X(1),TT(1)
C
C     SET THE MEASUREMENT MODEL.
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *       YOUR CODING FOR THE MEASUREMENT MODEL GOES HERE.         *
C     *       USE X FOR THE STATE, TT FOR THE TIME, AND PUT THE        *
C     *       MODEL INTO ARRAY H AS SHOWN BELOW.                       *
C     *                                                                *
C     *                                                                *
C     *                                                                *
      H(1)=THETA(4)*X(1)
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      HMEAS=0.
      RETURN
      END
```

Figure 3-5. Subroutine HMEAS

Although this derivative could be evaluated numerically using multiple calls to FSYSTM, (see the DYNEST manual if numerical Jacobians are desired) the user is given the opportunity to provide an analytic evaluation of the system Jacobian in subroutine FJACOB. The example is shown in Figure 3-6.

3.4.4 HJACOB

A similar Jacobian is needed for the measurement model:

$$H = \frac{\partial}{\partial \underline{x}} h(\underline{x}, \underline{\theta}) \qquad (3.4\text{-}4)$$

and sample coding is shown in Figure 3-7.

3.4.5 FQ

FQ sets up the covariance matrix for the input driving noise:

$$Q = E\left\{ ww^T \right\} \qquad (3.4\text{-}5)$$

Figure 3-8 shows the coding for the example problem. In the example problem, we modeled a system that had no driving noise, so we set Q equal to zero and held it there for the entire MXLKID run ($Q = \theta_5^2 = 0$).

3.4.6 FR

R is the covariance of the measurement noise:

$$R = E\left\{ vv^T \right\} \qquad (3.4\text{-}6)$$

The sample coding is shown in Figure 3-9.

```
      SUBROUTINE FJACOB(NN,TT)
C
C     THIS ROUTINE COMPUTES THE JACOBIAN OF THE SYSTEM MODEL.
C     THIS IS A NECESSARY ROUTINE, WHICH IS PROBLEM DEPENDENT.
      EXTERNAL FSYSTM
      LOGICAL MEAZ

      USE LIODCOM

      DIMENSION TT(1)
      DIMENSION XD(NS)

                                    .
                                    .
                                    .

C     EVALUATE THE JACOBIAN ANALYTICALLY
C     **********************************************************************
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *        YOUR CODING FOR THE SYSTEM JACOBIAN GOES HERE.              *
C     *        PUT THE JACOBIAN INTO ARRAY FJ AS SHOWN BELOW.              *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *        SET THE MODEL PARAMETERS.  THIS COULD BE DONE IN            *
C     *        ROUTINE INIT AND THE VARIABLES PASSED TO THIS ROUTINE       *
C     *        THROUGH COMMON.                                             *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
      FJ(1,1)=0.
      FJ(1,2)=1.0
      FJ(2,1)=-(THETA(2)**2)
      FJ(2,2)=-2*THETA(1)*THETA(2)
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     *                                                                    *
C     **********************************************************************
      GO TO 4

C
C     EVALUATE THE JACOBIAN NUMERICALLY
    3 CONTINUE
      MEAZ=.FALSE.
      CALL JACOB(FSYSTM,NN,NN,TT,XD,MEAZ)
    4 CONTINUE
      RETURN
      END
```

Figure 3-6. Subroutine FJACOB

```
      SUBROUTINE HJACOB(TT,X,HJAC)
C
C     THIS ROUTINE COMPUTES THE JACOBIAN OF THE MEASUREMENT MODEL
C     THIS JACOBIAN WILL BE AN N BY H MATRIX
C     THIS IS A NECESSARY ROUTINE WHICH IS PROBLEM DEPENDENT.
      EXTERNAL HMEAS
      LOGICAL MEAZ

      USE LKDCOM

      DIMENSION HJAC(NMR,1),X(1),TT(1)

      IF(KFJAC EQ.1) GO TO 1

C     SET THE MEASUREMENT JACOBIAN ANALYTICALLY
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *       YOUR CODING FOR THE MEASUREMENT JACOBIAN GOES HERE.    *
C     *       USE X FOR THE STATE, TT FOR THE TIME, AND PUT THE      *
C     *       JACOBIAN INTO ARRAY HJAC AS SHOWN BELOW.               *
C     *                                                              *
C     *                                                              *
C     *                                                              *
      HJAC(1,1)=THETA(4)
      HJAC(1,2)=0.
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     *                                                              *
C     ****************************************************************

      RETURN
C
C     CALCULATE THE MEASUREMENT JACOBIAN NUMERICALLY
    1 CONTINUE
      MEAZ=.TRUE.
      CALL JACOB(HMEAS,M,N,TT,X,MEAZ)
      RETURN
      END
C
C
```

Figure 3-7. Subroutine HJACOB

```
      SUBROUTINE FQ(AQ)
C
C     THIS ROUTINE COMPUTES THE CORRELATION ARRAY OF THE SYSTEM NOISE FUNCTION
C     THIS IS A NECESSARY ROUTINE WHICH COULD BE PROBLEM DEPENDENT.
C
      USE LKDCCM

      DIMENSION AQ(NMR,1)

      DO 1 I=1,N
      DO 2 J=1,N
      AQ(I,J)=0.
    2 CONTINUE
      AQ(I,I)=Q(I,I)
    1 CONTINUE
      Q(1,1)=THETA(5)*THETA(5)
      Q(2,2)=Q(1,1)
      RETURN
      END
```

Figure 3-8. Subroutine FQ

```
      SUBROUTINE FR(AR)
C
C     THIS ROUTINE COMPUTES THE CORRELATION OF THE MEASUREMENT NOISE
C     AR IS AN EXM ARRAY
C     THIS IS A NECESSARY ROUTINE WHICH COULD BE PROBLEM DEPENDENT.
C
      USE LKDCOM

      DIMENSION AR(NMR,1)

      DO 1 I=1,M
      DO 2 J=1,M
      AR(I,J)=0.
    2 CONTINUE
      AR(I,I)=RR(I,I)
    1 CONTINUE
      AR(1,1)=THETA(6)*THETA(6)
      CALL SCLM(-1,NMR,AR,M)
      CALL SCLM(-1,NMR,AR,M)
      RETURN
      END
```

Figure 3-9. Subroutine FR

3.5 Program Execution

Once the user has prepared the input files as explained in section 3.3.2, and set up the system desciptor subroutines (FSYSTM, HMEAS, FJACOB, HJACOB, FR AND FQ) as explained in section 3.4, the MXLKID program is ready to be compiled and run. The executable file is named XMXLKID. To start up the maximum likelihood identifier, the user types XMXLKID/t v. The program prompts for the name of the measurement, input, and output files. For the measurement and input files, give the names of the input files that were prepared earlier. The user can choose any name for the output file (up to ten alphanumeric characters), and output data from the program (described in the next section) will be stored in a disk file with that name.

The program now prompts for an 80-character run title. This title, along with time and date information, appears on the first line of the output file and at the bottom of each graph in the plot file. The graphs and printout can thereby be easily associated with a particular MXLKID run. Once the user has typed in a desired title, the program responds with the time and date.

3.6 Program Outputs

MXLKID sends program output to these places: the teletype terminal, an output file, and a plot file. The name of the output file is specified by the user when the program prompts for measurement, input, and output files as explained above. The plot file is named FX10$5MLID0.

3.6.1 TTY Printout

The amount of printout at the teletype depends on the choice of interactive mode (see the description of NP in

section 3.3). We provide here a description of what can be expected at the teletype if the program is in non-interactive mode. Then we provide a quick user's guide to the commands available under the interactive mode.

3.6.1.1 No Interaction

If no interaction is requested (NP=0) printout at the teletype looks like the example shown in Figure 3-10. (This printout is from the sample problem described in section 3.2). Note that each function evaluation results in a line of printout, including those evaluations required for numerical calculation of the gradient and line searching. When covergence is reached, the program types the total execution time in seconds and completes with "all done."

3.6.1.2 Interaction

When MXLKID is in the interactive mode (NP≠0), the user exercises a measure of control over program operation. If the maximum likelihood algorithm coverges, or if the algorithm detects a problem in converging*, control is handed over to the user. The program types a prompt ("]") and the user at that point can examine the state of the algorithm, change certain values if desired, then command that the algorithm continue iterating for better parameter

* Maximum number of iterations exceeded or maximum number of line search attempts exceeded. These maximum values are provided by the user in the input file. See section 3.3.

```
XMXLKID
INPUT;MEAS;OUTPUT FILES
.INLID OUTSIM SAMPOUT
 TYPE A LABEL FOR THIS RUN (UP TO 80 CHARACTERS)
MXLKID SAMPLE RUN FOR USER'S MANUAL
 THANK YOU    16:08:34R 05/13/80
IFN          F              THETAS
  1   21946.89808465      0.10000000       9.00000000
  2   21946.89807969      0.10000001       9.00000000
  3   21946.89936422      0.10000000       9.00000090
  4   49859.38187218      1.04264032     -12.79138543
  5     443.72356107      0.34970426       3.22735642
  6     443.72336119      0.34970429       3.22735642
  7     443.72332427      0.34970426       3.22735674
  8     -53.16623147      0.53794883       3.19922502
  9     -53.16626020      0.53794888       3.19922502
 10     -53.16606044      0.53794883       3.19922534
 11     -86.96043785      0.60954065       3.14366430

    27 061:38:00

ALL DONE
```

Figure 3-10. Sample TTY Printout—No Interaction

estimates, or halt execution. An additional interactive option is available to have the program prompt the user at the end of every iteration (NP=1) or every np iterations (NP > 0) if desired. A list of user commands is provided below:

HELP - Type out a brief (one line) explanation of each available command.

TYPE - Type: 1) the value of the negatiave log likelihood function (NLLF) at the current parameter values, 2) the current parameter values, and 3) the gradient (derivative of the NLLF with respect to the parameters).

BOUNDS - Type the parameter estimates and the 95% confidence bounds for these estimates.*

CONVERGE - Type out the results of the most recent covergence test along with the user specified convergence criteria.

MODIFY - Modify the convergence criteria. The program asks the user to specify which criterion to change (1, 2 or 3 corresponding to gradient norm, significant parameter digits, or

* Note: the confidence bound calculations are not accurate unless the parameter estimates are close to their true values, i.e., the algorithm is at or near convergence.

relative cost function change, respectively) and then asks for a new value.

PARAMETER - Modify the values of the parameters. The program asks the user to specify the parameter and provide a new value in a manner similar to the MODIFY command.

GO - Continue on from this point to iteratively improve parameter estimates. (Algorithm continues whether or not convergence or divergence had been previously detected.)

END - Stop any further parameter improvement attempts. Complete the plots, close the output files, and terminate the program.

Figure 3-11 shows a sample terminal session with full (NP=1) interaction. Note that typing the first letter of a command is sufficient for the interpreter to recognize the command.

3.6.2 Output File

The MXLKID program generates an output file in order to provide the user with a progress history of the attempts to iteratively improve the parameter estimates. The progress history can prove useful if, for example, the algorithm diverges. The output file can then help the user to diagnose the problem.

```
*MYLKID


INPUT;MEAS;OUTPUT FILES
.INLID OUTSIM SAMPOUT
 TYPE A LABEL FOR THIS RUN (UP TO 80 CHARACTERS)
MXLKID SAMPLE RUN FOR USER'S MANUAL
 THANK YOU    16:08:34R 05/13/80
IFN          F                THETAS
  1   21946.89808465        0.10000000        9.00000000
  2   21946.89807969        0.10000001        9.00000000
  3   21946.89936422        0.10000000        9.00000090
  4   49859.38187218        1.04264032      -12.79188543
  5     443.72356107        0.34970426        3.22735642
 ITERATION    1 COMPLETED
 ]
T
 THIS ITERATION: F=  4.4372E+02TH=  3.4970E-01  3.2274E+00
 LAST ITERATION: F0=  2.1947E+04TH0=  1.0000E-01  9.0000E+00
 GRADIENT= -4.9614E+02  1.4217E+03
 ]
E
 PARAMETER ESTIMATES AND 95% CONFIDENCE BOUNDS
    1.    3.4970E-01      +/-  6.8853E-02
    2.    3.2274E+00      +/-  2.4201E-01
 ]
C
 CONVERGENCE PARAMETER CALCULATIONS ARE AS FOLLOWS
 NORM OF THE GRADIENT=  2.2675E+06
 SIGNIFIGANT DIGIT CHANGE IN THE PARAMETERS= -7.6137E-01 DIGITS
 CHANGE IN THE COST FUNCTION= -2.1503E+04
 USER SPECIFIED CRITERIA FOR CONVERGENCE ARE
 GNORM:   0.                    SIGDIG:  1.4000E+01          DF:   0.
 ]
G
  6     443.72336119       0.34970429       3.22735642
  7     443.72332427       0.34970426       3.22735674
  8     -53.16623147       0.53794883       3.19922502
 ITERATION    2 COMPLETED
 ]
G
  9     -53.16626020       0.53794888       3.19922502
 10     -53.16606044       0.53794883       3.19922534
 11     -86.96043785       0.60954065       3.14366430
 ITERATION    3 COMPLETED
 ]
E

    27 061:38:00

 ALL DONE
```

Figure 3-11. Sample Session with Full (NP=1) Interaction

Figure 3-12 shows a partial listing of the output file for our sample problem. The header portion of the output file merely repeats information which was supplied by inputs to the program. After the first line of asterisks (line 46), an interation by iteration account of the progress history begins. Symbols used are explained in Tables 3-4 and 3-5.

Lines 46 through 81 are typical prinout for one iteration of the optimization algorithm. Included are the computed gradient vector (G), Hessian matrix (HESS), and step (parameter improvement) vector (STEP).

Lines 82-89 are prinout for a line search attempt. Notice that the computed STEP resulted in an increase, rather than decrease, in the NLLF (line 79). At that point the algorithm automatically shortened the step length (ρ in equation 1-4) and tried again. This time the NLLF decreased, so the algorithm continued normally. The process of modifying ρ until the NLLF is reduced is called a line search. Several line search attempts may be necessary during a particular iteration, however, in the example problem, one attempt was sufficient.

If the iteraction feature of the MXLKID code is utlized, any prinout at the teletype is alsc written (echoed) on the output file. Lines 90 through 102 show the interaction which occurred during iteration 1 of the sample problem.

After several iterations the algorithm will (hopefully) converge. The final printout inlcudes the converged, or at least the final, set of paramter estimates and their 95% confidence intervals, as determined by the algorithm.

```
 1
 2            MKLKID SAMPLE RUN FOR USER'S MANUAL
 3              13:23:59U 05/13/88
 4
 5            EXTENDED KALMAN FILTER RUN
 6
 7            THE NUMBER OF STATES IS:   2
 8            THE NUMBER OF MEASUREMENTS IS:   1
 9            SAMPLE COUNT=  50
10            THE INITIAL INTEGRATION STEP SIZE IS:  5.0000000E-02
11                 THE INITIAL PROCESS NOISE COVARIANCE MATRIX IS:
12             0.          0.
13             0.          0.
14                 THE INITIAL MEASUREMENT NOISE COVARIANCE MATRIX IS
15             0.
16                 THE INITIAL ERROR COVARIANCE MATRIX IS:
17             1.000E-06 0.
18             0.          1.000E-06
19
20 INITIAL CONDITION FILE -       INLID
21       MEASUREMENT FILE -      OUTSIM
22            OUTPUT FILE -     SAMPOUT
23
24                 THE INITIAL STATE VECTOR IS:
25            0.          0.
26                 MINIMIZATION ROUTINE PARAMETERS ARE:
27                      MAXITR=     6
28                      MAXTRY=     5
29                      EVMIN=  1.0000E-06
30                      OPTN=  1.0000E+00  1.0000E-02  1.0000E-07  0.
31                 INITIAL THETAS ARE
32                        1.000000E-01    9.000000E+00    4.000000E+00    2.000000E+00
33                        0.              1.000000E-01
34                   2 OF THE THETAS ARE TO BE VARIED, THEY ARE:
35                        1  2
36                 TTY INTERACTION OCCURS EVERY    1 ITERATIONS
37                 MINIMIZATION CONVERGENCE CRITERIA ARE AS FOLLOWS
38                 1. NORM OF THE GRADIENT: 0.
39                 2. SIGNIFICANT DIGIT CHANGE IN THE PARAMETERS: 1.4000E+01
40                 3. CHANGE IN THE COST FUNCTION: 0.
41                 THE P. " OPTION (MP) IS    1
42                 INNOVATIONS PLOTTED EVERY   1 ITERATIONS
43 F=  2.19468981E+04
44 TH
45        0.10000000       9.00000000
46 ********************************************************************************
47                        ITR#      1
48 F=  2.19468981E+04
49 TH
50        0.10000001       9.00000000
51 F=  2.19468994E+04
52 TH
53        0.10000000       9.00000090
54
55 G
56    1      -4.96139E+02 1.42175E+03
57
58 HES
59    1       8.12680E+02 1.23865E+01
60    2       1.23865E+01 6.57778E+01
61
62 NORM HESS
63    1       1.00000E+00 5.35736E-02
64    2       5.35736E-02 1.00000E+00
65
66 EVAL
67    1       9.46426E-01 1.05357E+00
68
69 EVEC
70    1       7.07107E-01 7.07107E-01
71    2      -7.07107E-01 7.07107E-01
72
```

Figure 3-12. Example Output File

```
 73  HES**-1
 74    1       1.23404E-03-2.32380E-04
 75    2      -2.32380E-04 1.52465E-02
 76
 77  STEP
 78    1       9.42640E-01-2.17919E+01
 79  F=  4.98593819E+04
 80  TH
 81         1.04264032     -12.79188543
 82  STEP TRY#   1
 83       DF=  2.79124838E+04      RHO= 1.00000E+00
 84  DFDRHO= -3.14502074E+04
 85  F=  4.43723561E+02
 86  TH
 87         0.34970426       3.22735642
 88  STEP TRY#   2
 89       DF= -2.15031745E+04      RHO= 2.64899E-01
 90  ITERATION   1 COMPLETED
 91  THIS ITERATION: F=  4.4372E+02TH=  3.4970E-01  3.2274E+00
 92  LAST ITERATION: F0=  2.1947E+04TH0=  1.0000E-01  9.0000E+00
 93  GRADIENT= -4.9614E+02  1.4217E+03
 94  PARAMETER ESTIMATES AND 95% CONFIDENCE BOUNDS
 95    1.   3.4970E-01      +/-  6.8853E-02
 96    2.   3.2274E+00      +/-  2.4201E-01
 97  CONVERGENCE PARAMETER CALCULATIONS ARE AS FOLLOWS
 98  NORM OF THE GRADIENT=  2.2675E+06
 99  SIGNIFICANT DIGIT CHANGE IN THE PARAMETERS= -7.6137E-01 DIGITS
100  CHANGE IN THE COST FUNCTION= -2.1503E+04
101  USER SPECIFIED CRITERIA FOR CONVERGENCE ARE
102  GNORM:  0.                    SIGDIG:  1.4000E+01          DF:  0.
103  ********************************************************************************
104                       ITR#     2
```

•
•
•

```
181  THETA=  6.0954E-01  3.1437E+00
182  STANDARD DEVIATION=  9.6789E-03  7.9765E-03
183          ****************************************
184          *  PARAMETER          95% ERROR BOUND    *
185          *    1.   6.0954E-01  +/-    1.8971E-02  *
186          *    2.   3.1437E+00  +/-    1.5634E-02  *
187          ****************************************
```

Figure 3-12. Example Output File (Continued)

Table 3-4 - Information in the Output File Corresponding to a Single Iteration of the Parameter Identifier.

Variable Name	Data	Comment
F	Cost Function (NLLF)	
TH	Parameter Values	
G	Gradient	
HES	Hessian	
EVAL	Eigenvalues of the Hessian	
EVEC	Eigenvectors of the Hessian	
HES**-1	Modified Inverse of the Hessian	The inverted hessian is modified to insure positive definiteness. The method of modification depends on the type of minimization algorithm used. See Appendix A, section 4 for details.
STEP	The computed parameter improvement step.	

Table 3-5 - Information in the Output File Corresponding to a Line Search Attempt.

Variable Name	Data
DF	Difference between the new and previous value of the NLLF.
DFDRHO	Computed derivative of NLLF with respect to step size.
RHO	Step size.

The confidence intervals provide the user with accuracy limits for the parameter estimates, i.e., one can say with 95% confidence that the parameter value is in the region of the estimate, plus or minus its confidence interval.

3.6.3 Plot File

The plot file contains a sequence of innovations plots, a whiteness test for the final set of innovations, a plot of the cost function versus iteration number, and a plot of the search path through parameter space. These plots are put into an FR8D file named FX105MLIDO.

Innovations are defined as data residuals, i.e., the difference between measured and estimated data. Figure 3-13 shows the sequence of innovations plots from the MXLKID run for our sample problem. At the user's option (MP in the input file), an innovations plot is made at the end of each iteration of the identifier algorithm. As the algorithm progresses, the system model tends to improve due to better parameter estimates and as a result, the innovations statistical characteristics change. The mean tends toward zero and the innovations at different time points become uncorrelated; i.e., the process approaches white noise. As the model becomes a more exact representation of the real system that produced the data, the innovations sequence becomes dominated by the random and uncorrelated measurement noise, and the correlating effects due to modeling error become negligible.

The innovations plot sequence can be examined by the user to determine if the identifier is operating properly. For example, if the identifier seems to converge to a set of

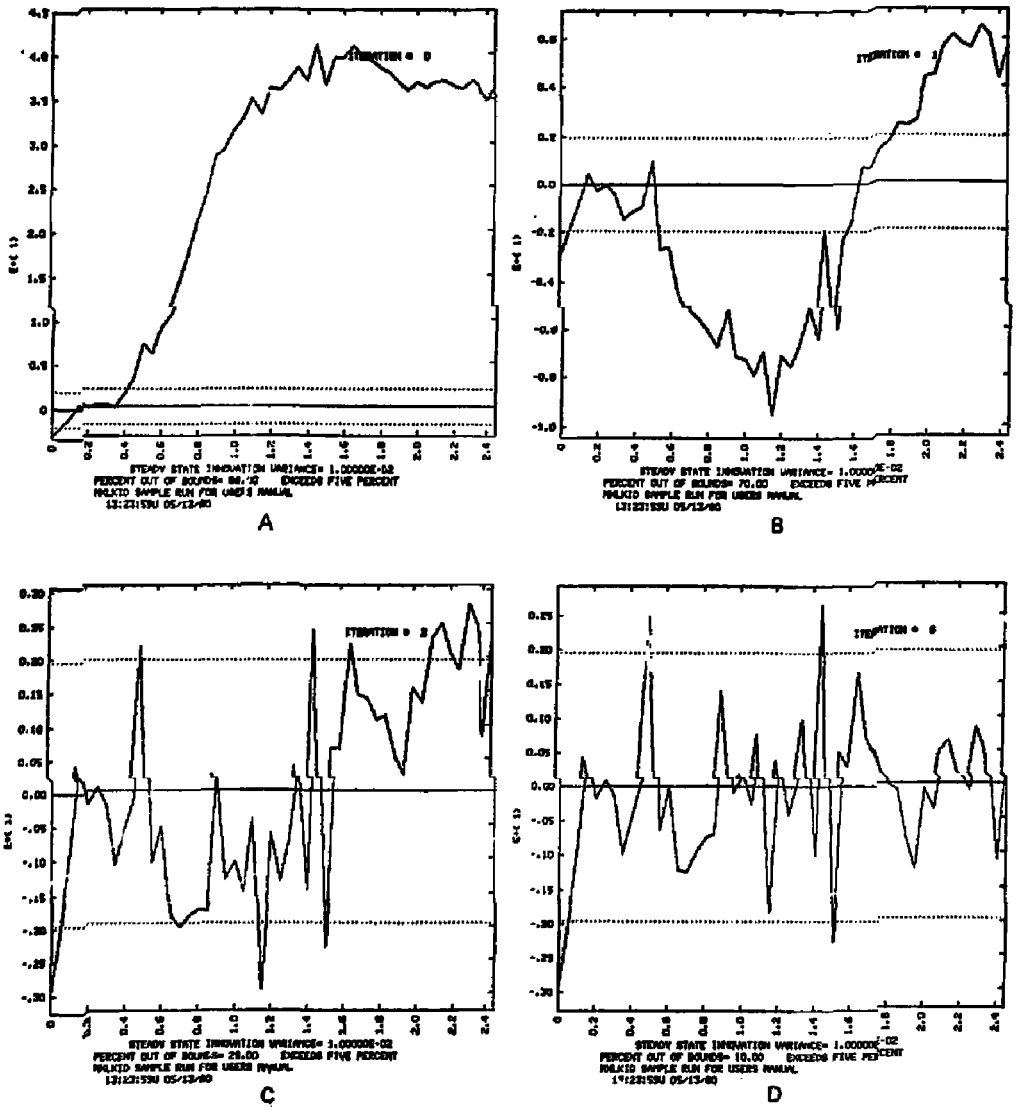


Figure 3-13. Sequince of Innovations Plots

parameter estimates, yet the final innovations are not "white" (uncorrelated), the convergence may be to a local minimum or saddle point.

MXLKID performs a statistical test on the final set of innovations to determine if the process is white. Figure 3-14 shows the results of the whiteness test for our sample problem. Basically, Figure 3-13 is a plot of the innovations sample autocorrelation function. We can conclude that our example process is white because none of the autocorrelations exceed the bounds indicated by the dashed lines ("percent out of bounds=0"). A full explanation of the whiteness test in included in Appendix B of this report.

The plot of NLLF as a function of iteration (Figure 3-15) shows graphically the improvement of the NLLF as the algorithm progresses. It should, of course, decrease monotonically if the algorithm is converging.

The parameter-space plot (Figure 3-16) shows the path of the identifier as it searches through parameter space for a minimum of the NLLF. The starting and ending points are indicated. With this plot, minimization algorithm problems such as hemstitching may be detected. Presently, the path plotter projects the path onto a $\theta_i \times \theta_j$ plane, where j=i+1 and i ranges from 1 to KTH-1 (KTH is the number of paramters). Future versions of MXLKID may include an option to select any two of the parameter to form a projection plane.

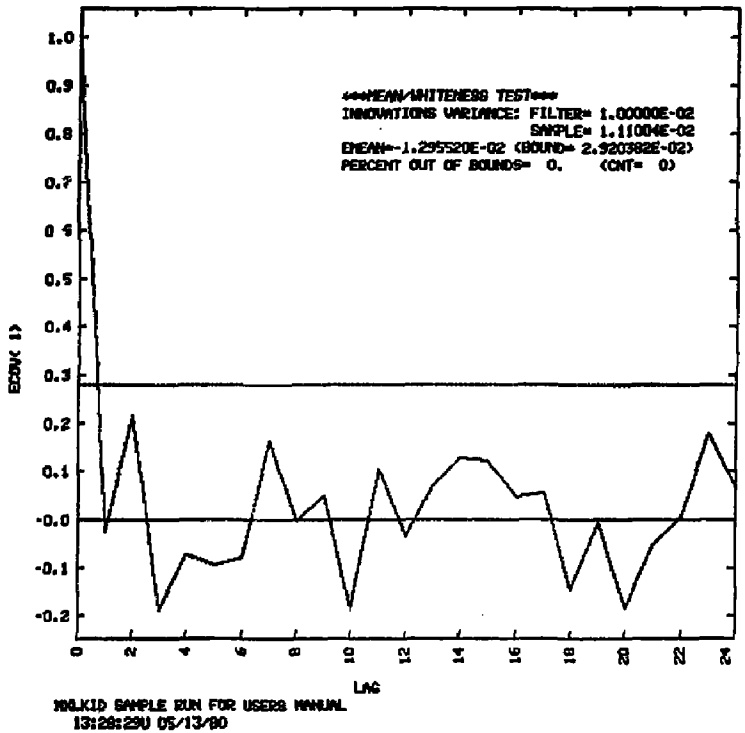


Figure 3-14. Innovations Whiteness Test

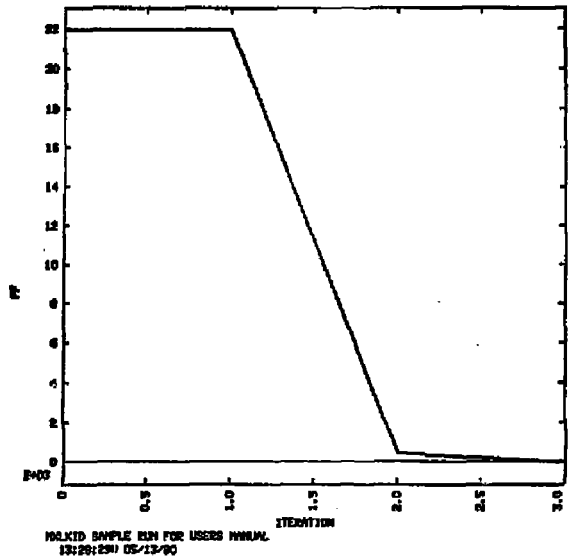


Figure 3-15 Negative Log Likelihood as a Function of Iteration

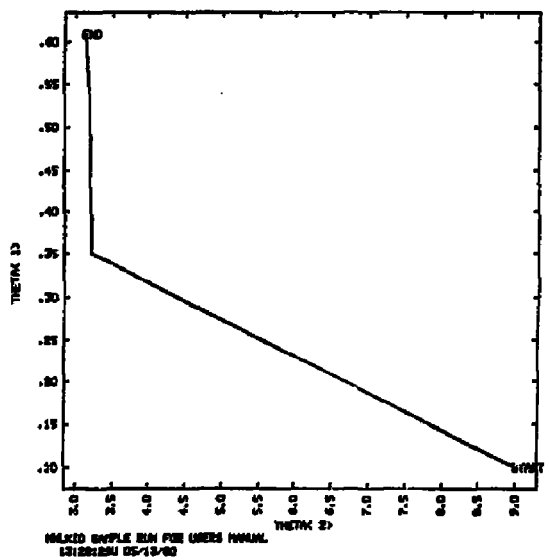


Figure 3-16. Path of Identification Algorithm in $\theta_1 \times \theta_2$

4.0 SUMMARY

In this report we have introduced the MXLKID parameter identification program as a tool for system analysis and identification. Since each iteration in MXLKID's nonlinear optimization process requires a Kalman filter pass over the set of measurement data, the program might be costly in terms of computer time, but as a system identification algorithm, it provides accurate parameter estimates and reliable confidence intervals in most cases.

A summary of MXLKID functions and requirements is shown in Table 4-1.

Table 4-1. Summary of MXLK Program Functions

```
*****************************************************************
*
*      THIS IS VERSION 1 OF MXLK
*      PROGRAMMED BY:  DON CAVEL,INFORMATION PROCESSING RESEARCH GROUP
*                      X2-8539,L-156
*                DATE:  MARCH 3, 1980
*
*****************************************************************

     MXLK -- PROGRAM FOR IDENTIFICATION OF SYSTEM PARAMETERS
             USING THE MAXIMUM LIKELIHOOD METHOD

    MXLK IS A METHOD FOR IDENTIFYING PARAMETERS CHARACTERIZING
    A DYNAMIC SYSTEM  MODELED BY EQUATIONS OF THE FORM:

              DX/DT=F(X,THETA,W,T)
                 Z=H(X,THETA,T)+V

            WHERE:
              W=N(0,Q), V=N(0,R), X(0)=N(XHAT(0),PI(0))
              THETA IS A SET OF UNKNOWN PARAMETERS TO BE IDENTIFIED
              Q,R,XHAT(0),AND PI(0) CAN BE FUNCTIONS OF THETA
              N(M,C) INDICATES NORMAL RANDOM VECTOR WITH MEAN, M AND
                     COVARIANCE, C

    USER INPUTS:
            THETA(0) - INITIAL PARAMETER ESTIMATES
                   R - COVARIANCE OF MEASUREMENT NOISE
                   Q - COVARIANCE OF RANDOM DRIVING NOISE
              PI(0) - COVARIANCE OF INITIAL STATE ERROR
               X(0) - INITIAL STATE
             FSYSTM - SYSTEM DYNAMICS SUBROUTINE (CONTAINS F(.))
              HMEAS - SYSTEM MEASUREMENT SUBROUTINE (CONTAINS H(.))
            CHOICE OF ALGORITHM FOR MINIMIZING THE
            NEGATIVE LOG LIKELIHOOD FUNCTION
                              A) GAUSS-NEWTON
                              B) LEVENBERG-MARQUARDT
            CONVERGENCE CRITERIA
                              A) NORM OF GRADIENT
                              B) RELATIVE CHANGE IN PARAMETERS
                              C) RELATIVE CHANGE IN COST FUNCTION

    OUTPUTS:
             THETA ESTIMATES
             THETA ESTIMATE VARIANCES
             CONVERGENCE CRITERION MET
             OUTPUT FILE CONTAINING A HISTORY OF:
                  THETAS
                  COST FUNCTION VALUES
                  GRADIENTS
                  HESSIANS
                  EIGENVALUES OF HESSIANS
                  STEP DIRECTIONS IN THETA SPACE
                  STEP SIZES AND HISTORY OF LINE SEARCH ATTEMPTS
                  MARQUARDT PARAMETERS
             PLOT FILE CONTAINING:
                  INNOVATIONS PLOT FOR EACH KALMAN FILTER RUN
                  PATH OF THE MINIMIZATION ALGORITHM IN THETA SPACE
                  COST FUNCTION VALUE VS. ITERATION NUMBER
    AVAILABILITY:
             MXLK IS AVAILABLE ON THE LLL CDC 7600 COMPUTER (OCTOPUS)
             IT IS WRITTEN IN LRLTRAN TO BE COMPILED BY THE CHAT COMPILER
             LOAD-TIME LIBRARIES USED: ORDERLIB, MASSLIB, STACKLIB, EE
             SUBROUTINE PACKAGES USED:
                       FILTER (KALMAN FILTER, FROM DYNEST4,
                               INFORMATION PROCESSING RESEARCH GROUP LIBRARY)
                       GEAR  (INTEGRATOR - NUMERICAL MATHEMATICS GROUP LIBRARY)
                       RS (EIGENVALUE PACKAGE - NUMERICAL MATHEMATICS GROUP LIBRARY)
```

References

[1.] N. K. Gupta and R. K. Mehra, "Computational Aspects of Maximum Likelihood Estimation and Reduction in Sensitivity Function Calculations," IEEE Transactions on Automatic Control, vol. AC-19, no. 6, pp. 774-783, December 1974.

[2.] R. L. Kashyap, "Maximum Likelihood Identification of Stochastic Linear Systems," IEEE Transactions on Automatic Control, vol. AC-15, no. 1, pp. 25-34, February 1970.

[3.] J. F. Best, Parameter Identification of Linear Systems Via the Maximum Likelihood Method, Technical Report TR-78-2, University of Connecticut School of Engineering, July 1978.

[4.] R. N. Castleton and J. V. Candy, DYNEST-A Dynamic Estimator Calculation Program, UCRL-52573, October 1978.

[5.] A. Gelb, Ed., Applied Optimal Estimation, M.I.T. Press, Cambridge, MA, 1974.

[5.] T. Kailath, Lectures on Linear Least-Squares Estimation, Springer Verlag, New York, 1976.

[7.] D. G. Luenberger, Introduction to Dynamic Systems, John Wiley and Sons, New York, 1979.

[8.] A. E. Bryson and Y. C. Ho, Applied Optimal Control, John Wiley and Sons, New York, 1975.

[9.] A. Papoulis, Probability, Random Variables, and Stochastic Processes, McGraw Hill, New York, 1965.

[10.] Y. Bard, "Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems," SIAM Journal of Numerical Analysis, vol. 7, no. 1, pp. 157-186, March 1970.

[11.] D. M. Himmelblau, Applied Nonlinear Programming, McGraw Hill, New York, 1972.

[12.] D. W. Marquardt, "An Algorithm for Least Squares Estimation of Nonlinear Parameters," SIAM Journal of Numerical Analysis, vol. 11, pp. 431-441, 1963.

[13.] H. L. Van Trees, Detection Estimation and Modulation Theory, Part I, John Wiley and Sons, New York, 1968.

[14.] R. K. Mehra and J. Peschon, "An Innovations Approach to Fault Detection and Diagnosis in Dynamic Systems," Automatica, 7, pp. 637-640, 1971.

APPENDIX A
MXLKID ALGORITHM DESCRIPTION

This appendix has been written to supplement the MXLKID users manual with more detailed background information on the program operation. We include all the basic formulas used in the maximum likelihood algorithm. The reader is assumed familiar with basic probability theory, linear algebra, and dynamic systems. A review of these areas is given in the first three chapters of Gelb [5].

The maximum likelihood parameter identification algorithm, as applied to linear dynamic systems, has been presented in papers by Gupta and Mehra [1] and Kashyap [2]. The MXLKID program is a relatively straightforward application of the algorithms presented in these papers, with the exception that MXLKID has been extended to cover non-linear, as well as linear, dynamic systems models.

Referring to the block diagram of the MXLKID algorithm (Figure A-1) we see that the main tasks of the algorithm are:

1. Use the Kalman filter to compute ε and R^e (innovations and innovations covariance).

2. Calculate the likelihood function, $J(\theta)$.

3. Update parameter estimates based on a gradient optimization algorithm.

4. Calculate a Hessian matrix and insure that it is positive definite (a subtask of the optimization algorithm).

5. Test the algorithm for convergence.

6. Apply a whiteness test on the final innovations.

The first five of the above tasks correspond to the five sections of this appendix. The whiteness test (task 6) is discussed in Appendix B.

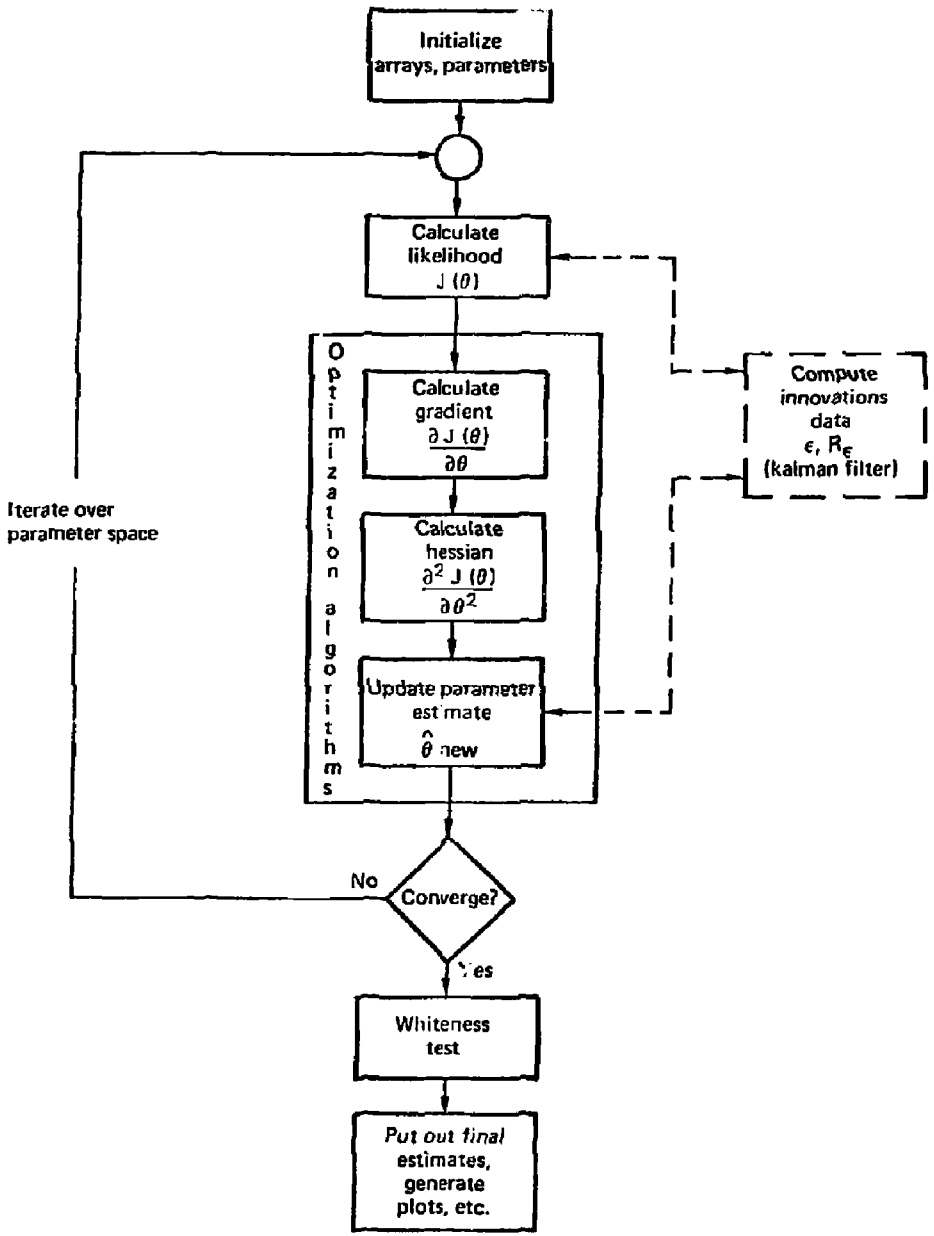


Figure A-1. MXLKID Implementation

A.1 Kalman Filter

In this section we present the Kalman filter as a tool for handling noisy data generated by a dynamic system. MXLKID uses the Kalman filter to generate an uncorrelated innovations sequence given the correlated measurement sequence, based on knowledge of an underlying dynamic model. Since the filter must be given a dynamic model of the system to work with, the optimization algorithm (discussed in section A.3) can essentially identify the system by trying out different models in the Kalman filter until an objective function called likelihood (see section A.2) is maximized. We start here by discussing dynamic system representations and presenting the Kalman filter algorithm.

A dynamic system can be modeled, mathematically, by ordinary differential equations (ODE). Within these ODE may be several parameters we wish to identify. Our approach is to first re-represent the ODE by transforming them to a set of simultaneous first order ODE. The resulting representation is the one most commonly used in modern systems analysis [5,7,8]. Simultaneous first order ODE are represented in vector-matrix notation as follows:

$$\dot{\underline{x}} = f(\underline{x}, \underline{\theta}, \underline{u}) \qquad \text{(A.1-1a)}$$

$$\underline{y} = h(\underline{x}, \underline{\theta}) \qquad \text{(A.1-1b)}$$

$\underline{y}$ represents the dependent variable in the original ODE (or variables, $\underline{y}$ may be a vector), $\underline{u}$ is the driving function and $\underline{\theta}$ is the set of unknown parmeters. Note that in the transformation from n^{th} order ODE to n simultaneous 1st order equations, a vector of n intermediate dependent variables, $\underline{x}$, is introduced. The vector $\underline{x}$ is called the *state*, and equation A.1-1 is called the state-space representation. Several state-space representations are possible for a given n^{th} order ODE. [7].

A mathematical representation of a dynamic system allows us to simulate that system on a digital computer. Given the initial state of the system at some time in the past, $\underline{x}(t_0)$, plus a history of all the inputs to the system since that time: $\{\underline{u}(t),\ t_0 \le t \le t_n\}$, and assuming we know the true set

of parameters; $\underline{\theta} = \underline{\theta}_{true}$, equation A.1-1 could be uniquely solved to determine $\underline{y}(t)$ for $t_0 \leq t \leq t_n$. In other words, we should ideally be able to simulate the true system outputs which would be measured in a laboratory test.

Of course, in practice, computer-model generated data never matches real data exactly. One source of mismatch is modeling error, that is, $\underline{\theta}$ is not known exactly or, more seriously, the order or non-linear structure of the system model is incorrect. Another source of mismatch between simulated and measured data is random noise. There is always some uncertainty in 1) the initial state, 2) the driving function, and 3) the measurement process. These noises are represented as follows:

$$\underline{x}(t_0) = \hat{\underline{x}}(t_0) - \tilde{\underline{x}}(t_0) \qquad \text{(A.1-2a)}$$

$$\underline{u} = \bar{\underline{u}} + \underline{w} \qquad \text{(A.1-2b)}$$

$$\underline{y} = h(\underline{x}, \underline{\theta}) \qquad \text{(A.1-2c)}$$

$$\underline{z} = \underline{y} + \underline{v} \qquad \text{(A.1-2d)}$$

where initial state error is represented by $\tilde{\underline{x}}(t_0)$ and estimate by $\hat{\underline{x}}(t_0)$; the driving function has a known part, $\bar{\underline{u}}$ and an unknown part $\underline{w}$; and the data measured in the laboratory is $\underline{z}$, and is corrupted by measurement noise, $\underline{v}$.

So, typically, the system identification problem is complicated. We have noisy data taken from an unknown system driven by uncertain inputs.

One tool available for handling noisy data is the Kalman filter[5,6] (KF) (Figure A-2). The KF is a statistical, model-based state estimator scheme. With the KF, knowledge of the system dynamics and statistics of the noise sources is used to reconstruct estimates of the true state of the system, $\left\{\hat{\underline{x}}(t,\underline{\theta}),\ t_0 \leq t \leq t_n\right\}$, and the true output, $\left\{\hat{\underline{y}}(t,\underline{\theta}),\ t_0 \leq t \leq t_n\right\}$. The reader unfamiliar with the KF can quickly skim over the algorithm outlined in Table A.1 or refer to Gelb[5]. Detailed knowledge of the KF is not necessary for running the MXLKID program.

Two outputs from the KF are required for computing the likelihood function (equations A.2-1 in the next section), 1) the innovations, ε_k, and 2) the innovations covariance, R^{ε}_k. Let us digress for a moment to discuss the role played by the innovations in the model identification process. Refer to the block diagram of the Kalman filter shown in Figure A.2. Note that the innovations, ε_k, are used in a feedback scheme to update state estimates in the model. If the filter model is valid, the innovations sequence is a statistically "white" and zero mean

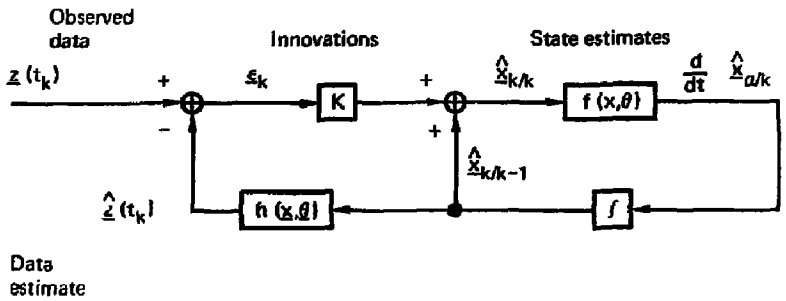


Figure A-2. Kalman Filter

TABLE A.1

SUMMARY OF CONTINUOUS-DISCRETE EXTENDED KALMAN FILTER

Prediction:

$$\hat{x}_{k+1|k} = \hat{x}_{k|k} + \int_{k}^{k+1} f(\hat{x}_{\alpha|k}, u_{\alpha})\, d\alpha$$

$$\tilde{\Pi}_{k+1|k} = \int_{k}^{k+1} \left[F(\hat{x}_{k|k})\tilde{\Pi}_{\alpha|k} + \tilde{\Pi}_{\alpha|k} F^{T}(\hat{x}_{k|k}) + Q_{\alpha} \right] d\alpha$$

Innovation:

$$\varepsilon_{k+1} = z_{k+1} - h(\hat{x}_{k+1|k})$$

$$R^{\varepsilon}_{k+1|k} = H(\hat{x}_{k+1|k})\tilde{\Pi}_{k+1|k} H^{T}(\hat{x}_{k+1|k}) + R_{k+1}$$

Correction:

$$K_{k+1} = \tilde{\Pi}_{k+1|k} H^{T}(\hat{x}_{k+1|k}) \left(R^{\varepsilon}_{k+1|k}\right)^{-1}$$

TABLE A.1 (Continued)

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}\varepsilon_{k+1}$$

$$\tilde{\Pi}_{k+1|k+1} = \left[I - K_{k+1}H(\hat{x}_{k+1|k})\right]\tilde{\Pi}_{k+1|k}\left[I - K_{k+1}H(\hat{x}_{k+1|k})\right]^T + K_{k+1}R_{k+1}K_{k+1}^T$$

where

$$F(\hat{x}_{k|k}) := \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{x}_{k|k}}, \quad H(\hat{x}_{k|k}) = \left.\frac{\partial (h(x))}{\partial x}\right|_{x=\hat{x}_{k|k}}$$

and

$$\tilde{\Pi}_{k+1|k} := \mathrm{Cov}(\tilde{x}_{k+1|k}) \text{ for } \tilde{x}_{k+1|k} := x_{k+1} - \hat{x}_{k+1|k}$$

$$x_0 \sim N(\hat{x}_{0|0}, \tilde{\Pi}_{0|0})$$

process, that is, uncorrelated from one time point to the next (Figure A.3). Innovations whiteness is useful as an independent statistical test to determine if the maximum likelihood algorithm has converged to a viable set of parameter estimates. For this reason, a plot of the innovations for each KF run is stored in an output file during MXLKID execution. After the algorithm has converged, a test of the innovations mean and whiteness is applied to assure the validity of the estimated model parameters. Details of this test are given in Appendix B.

Implementation of the KF within MXLKID demands that the user provide initial information concerning the statistics of process and measurement noise, and the dynamics of the system. (Detailed information on providing this information to the program is contained in Chapter 3 of this report.) The necessary starting ingredients include: R, the covariance of the measurement

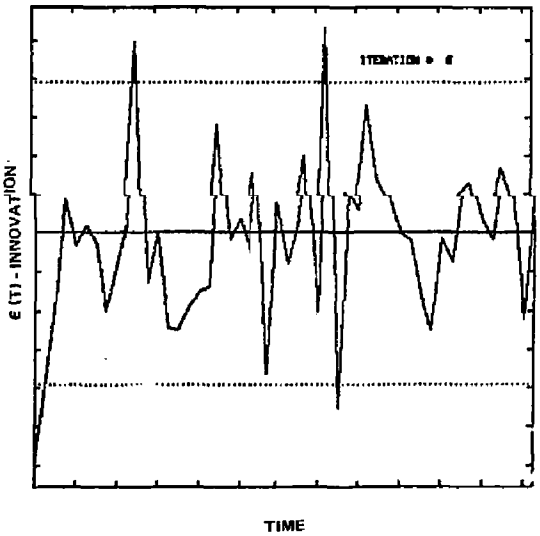


Figure A-3. Example of a White Innovations Sequence

noise, Q, the covariance of the random part of the driving function, $\tilde{\Pi}(t_0)$, the covariance of the initial condition error, plus a specification of system dynamics in a form similar to equations A.1-1a and b. Some or all of the above information may be expressed in terms of an unknown set of parameters, $\underline{\theta}$ which are to be identified by MXLKID. An initial "guess" for $\underline{\theta}$ must also be specified.

We now proceed to derive an expression for likelihood which uses the results of the Kalman filtering just described.

A.2 The Likelihood Function

Likelihood is a measure of the validity of a dynamic system model. The maximum likelihood algorithm is based on the assumption that the parameter vector is closest to the true set of system parameters when the likelihood function (LF) is at its maximum. We define the LF to be a conditional probability density[1] as follows:

$$L(\underline{\theta}) = p(Z(N)\ \underline{\theta}) = \prod_{k=0}^{N} \frac{1}{\sqrt{2\pi}\left|R^{\epsilon}(t_k,\underline{\theta})\right|^{1/2}} \cdot \exp\left\{-1/2\underline{\varepsilon}^{T}(t_k,\underline{\theta})\left(R^{\epsilon}(t_k,\underline{\theta})\right)^{-1}\underline{\varepsilon}(t_k,\underline{\theta})\right\} \qquad (A.2\text{-}1)$$

here, Z represents the entire observed data sequence $Z(N) = z(t_k)$; $k = 0, \ldots, N$ and $\underline{\theta}$ represents the set of unknown parameters which we wish to identify.

We choose a statistical measure of model validity because of the underlying statistical nature of the system identification process. Measurement error, random driving noise, and an uncertain initial state all contribute to randomness in the search for an ideal system model.

The equation for likelihood given above is an approximation, valid only insofar as we can assume a Gaussian PDF (probability density function) for Z(N) conditioned on $\underline{\theta}$. Among other things this assumes a linear or linearized system model.[9] In practice, however, many non-linear models have worked quite satisfactorily, the requirement being that the model be not too ill-behaved, so that linearized approximations can work to a certain extent.

Likelihood is computed by running the observed data through a Kalman filter to generate the innovations sequence, $\underline{\varepsilon}(t_i,\underline{\theta})$, and sequence of innovations

covariances, $R^{\epsilon}(t_i, \theta)$; i=1,... N. Note that, since the LF is conditioned upon $\underline{\theta}$, we must specify the parameter set before computing the LF. How can we specify such a $\underline{\theta}$, if initially it is unknown? Since the objective is to maximize $L(\underline{\theta})$ with respect to a choice of $\underline{\theta}$, we make an initial guess, and then, according to some numerical optimization scheme (we discuss optimization schemes in the next section) $\underline{\theta}$ is sequentially improved until the maximum of $L(\underline{\theta})$ is reached. This sequential alogrithm requires at least one function evaluation at each step along the way.

$L(\underline{\theta})$ can be maximized if we minimize its negative logarithm. We define a new function, the negative log-likelihood function (NLLF) as follows:

$$J(\underline{\theta}) = -\ell n\left[L(\underline{\theta})\right] = -1/2\,\ell n\ (2\pi) - 1/2 \sum_{i=1}^{N} \left\{ \underline{\varepsilon}^{T}(t_i, \underline{\theta}) \left(R^{\epsilon}(t_i, \underline{\theta})\right)^{-1} \underline{\varepsilon}(t_i, \underline{\theta}) + \ell n \left|R^{\epsilon}(t_i, \underline{\theta})\right| \right\} \qquad \text{(A.2-2)}$$

A minimum of the NLLF is a maximum of the LF because the logarithmic transformation is monotonic.

Having defined a suitable cost function (hereafter, when we mention the "cost function", we refer to the NLLF) all that remains to the maximum likelihood parameter identification scheme is the minimization of this cost function with respect to the unknown set of parameters. In the next section we discuss the numerical minimization techniques available in the MXLKID program.

A.3 Minimization Techniques

The main task of the maximum likelihood parameter identification algorithm is to find the parameter set, $\underline{\theta}$, which minimizes the cost function, $J(\underline{\theta})$. There exists a multitude of methods for numerically minimizing a function. Basically, they all follow an interative procedure resembling the following: [10]

1. Pick a starting $\underline{\theta}$.
2. Calculate $J(\underline{\theta})$(using the Kalman filter).
3. Using an algorithm of your choice, pick some new $\underline{\theta}$.
4. Calculate the new $J(\underline{\theta})$.
5. If the new $J(\underline{\theta})$ is less than the old $J(\underline{\theta})$ return to step 3 and continue until some convergence criterion is met.
6. If the new $J(\underline{\theta})$ is not less than the old $J(\underline{\theta})$, use some default algorithm to pick a better $\underline{\theta}$ and return to step 4, or give up.

The iterative minimization methods can be any of the many non-linear programming techniques developed over the years. [11] Of these, the gradient methods seem to offer the best success. In the MXLKID program two gradient methods are available: the Gauss-Newton algorithm and the Levenberg-Marquardt algorithm. In both these algorithms, the first derivative of the cost function (the gradient) is used to determine the direction of search for an improved $\underline{\theta}$:

$$\hat{\underline{\theta}}_{\ell+1} = \hat{\underline{\theta}}_{\ell} - \rho h \underline{g}_{\ell} \qquad \text{(A.3-1)}$$

where

$$\underline{g}_{\ell} = \frac{\partial J(\theta)}{\partial \underline{\theta}} \Bigg|_{\underline{\theta} = \underline{\theta}_{\ell}}$$

where h is a pxp (where p is the number of parameters) matrix, and ρ is a scalar and ℓ is the iteration index. (The selection of h and ρ are determined by the particular type of method used, as we shall show.) Note that the parameters are modified in a direction opposite the gradient (the direction may be modified somewhat by the h matrix), i.e., they are walked down the slope toward the minimum of $J(\underline{\theta})$. When the minimum is reached, $\underline{g}$ is (theoretically) a zero vector, and the algorithm has converged to solution for $\underline{\theta}$.

The gradient, $\underline{g}$, is calculated numerically in the MXLKID algorithm as follows:

$$g_i = \frac{J(\underline{\theta} + \Delta\underline{\theta}^{(i)}) - J(\underline{\theta})}{\Delta\theta_i} \; ; \qquad i = 1, \ldots, p \qquad \text{(A.3-2)}$$

where $\Delta\underline{\theta}^{(i)}$ is a p-vector whose only non-zero element is the i^{th} element, which is $\Delta\theta_i$. The Kalman filter is run p+1 times, once to calculate $J(\underline{\theta})$, and then p more times to calculate $J(\underline{\theta} + \Delta\underline{\theta}^{(i)})$ for i=1 to p. As would be expected, the gradient calculation is usually the most time consuming task in the MXLKID algorithm.

Let us now present the particular optimization algorithms programmed in MXLKID and discuss their individual characteristics.

A.3.1 Gauss-Newton

The Gauss-Newton method assumes the cost function surface to be, to a first approximate, a parabolic bowl (Figure A-4). This optimization method is designed to immediately send $\underline{\theta}$ to the lowest point within the bowl.

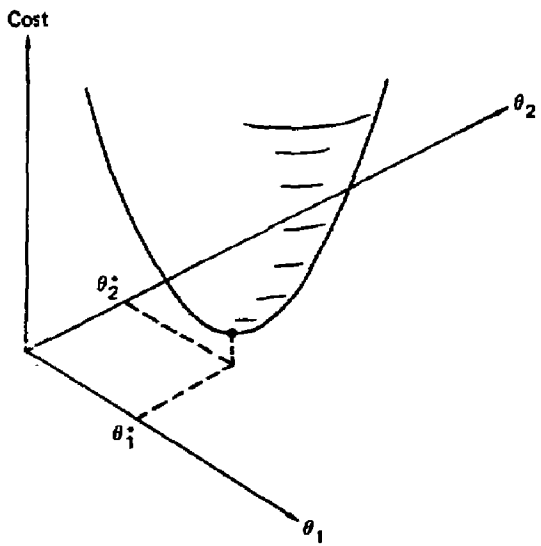


Figure A-4. Example of a Parabolic Approximation to a Cost Function Surface

We substitute the following terms into equation A.3-1:

$$\rho = 1 \qquad \text{(A.3-3)}$$

and $$h = H^{-1}$$

where $$H_{ij} = \frac{\partial^2 J(\theta)}{\partial\theta_i \partial\theta_j}$$

H, called the "Hessian" of the cost function, contains surface curvature information and must be computed at each $\underline{\theta}$. (Numerical Hessian calculation techniques are discussed in Section A-4.) Note from Figure A-5 that h modifies the step direction, $\vec{s}$, slightly away from the negative of the gradient, $-\vec{g}$, so that parameters are moved directly toward the minimum point.

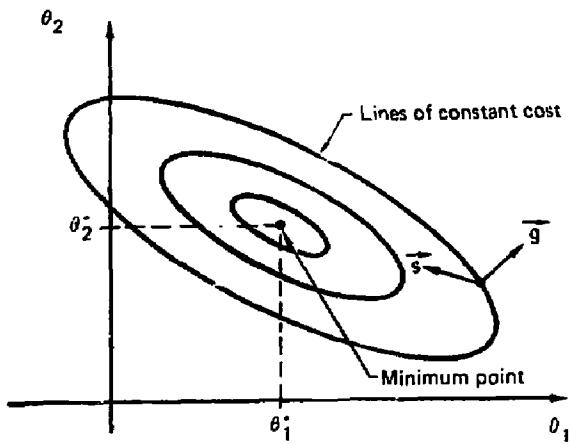


Figure A-5. Cost Function Contour Plot. $\vec{s}$ is the step direction, $\vec{g}$ is the gradient.

The Gauss-Newton algorithm generally provides fast convergence to the minimum of $J(\underline{\theta})$ so long as $\underline{\theta}$ starts out near that minimum. Problems arise however, when the cost function suface does not have a positive curvature, in which case the parabolic bowl approximation mentioned above cannot be applied. The Gauss-Newton algorithm may tend to take steps directly away from the minimum, and thus diverge, or the algorithm could coverge to a saddle point on the surface.

A.3.2 Levenberg-Marquardt

To introduce the Levenberg-Marquardt algorithm, we first discuss steepest descent methods.

If $\underline{\theta}$ lies far from the minimum, the Gauss-Newton approach may prove ineffective. Better results may be obtained if the parameters are forced to step in a direction directly opposite the gradient (i.e., along the line of "steepest descent"), ignoring the local surface curvature. In equation A.3-1, we set h equal to the identity matrix, and the step size, ρ, is arbitrary. (The step size must be modified when necessary to insure that the cost function decreases.)

Steepest descent methods provide rapid initial parameter improvement, avoiding the problems of the Gauss-Newton method, but convergence tends to be slow in the local vicinity of the cost function minimum. Ideally, at some point during the search, one should switch over to a Gauss-Newton algorithm to speed up final convergence.

The Levenberg-Marquardt (LM) algorithm[12] offers an alternative to explicitly switching the minimization method in the middle of the optimization procedure. This technique uses a sliding

parameter, β, which, when large, causes the algorithm to act similar to steepest descent, and when small, causes the algorithm to act similar to Gauss-Newton. β is adjusted according to the success achieved during previous iterations; the better the improvement in $J(\underline{\theta})$, the smaller β becomes.

The following is an outline of the LM algorithm as implemented in MXLKID:

1. Set β (Marquardt parameter) equal to some initial value, say 0.01.

2. Use equation A.3-1 with
   $\rho = 1$
   $h = (H + \beta D^2)^{-1}$ (A.3-4)
   where D^2 is a diagonal matrix containing the diagonal elements of H. (β may need to be modified here to force h to be positive definite. See section A.4.)

3. If the cost function, NLLF, is reduced, accept the new value for $\underline{\theta}$, reduce β by a factor of ten, and begin another interation (return to step 2). Otherwise, if the cost function increases, continue.

4. Check the angle between the negative gradient, $-\underline{g}$, and the step direction, $-h\underline{g}$. If this angle is greater than 45°, increase β by a factor of ten and return to step 2. (Increasing β brings the step direction more in line with the negative gradient). Otherwise, continue.

5. Search along the line defining the step direction (i.e., vary ρ) and accept the first $\underline{\theta}$ which reduces the cost function. (The search method used in MXLKID is a parabolic interpolation described in Bard [10].) Return to step 2 after reducing β by a factor of ten.

This concludes our general introduction to gradient minimization techniques. In the next section we discuss in more detail an important subtask of the minimization algorithms, calculation and inversion of the Hessian matrix.

A.4 Hessian Calculation and Inversion Techniques

In the previous section, we discussed methods of minimizing a non-linear function of a set of parameters, with respect to the parameters. The methods discussed, along with several other second order techniques, require the evaluation of a Hessian matrix, that is, a second derivative of the objective function with respect to the parameters. The Hessian provides information about the curvature of the objective function surface, and this information is useful to help determine both direction and step size in the search of a minimum.

For the maximum likelihood identification problem, direct numerical calculation of the Hessian would require p^2 Kalman filter runs, where p is the number of parameters. Fortunately, an approximation is possible which eliminates the need for so many KF runs. The approximation uses information generated during the numerical gradient calculation.

In this section we introduce the Hessian approximation technique used in MXLKID. We also discuss methods of forcing the Hessian matrix to be positive definite (a necessary condition in second order optimization methods) [10] and present the techniques used for inverting the matrix.

A.4.1 Hessian Approximation

The following fact is known from parameter estimation theory: [13]

$$E\left\{\frac{\partial^2 \ell n\, L(\theta)}{\partial\theta_i \partial\theta_j}\right\} = -E\left\{\frac{\partial \ell n L(\theta)}{\partial\theta_i}\;\frac{\partial \ell n L(\theta)}{\partial\theta_j}\right\} \qquad \text{(A.4-1)}$$

where $L(\underline{\theta})$ is the likelihood function as defined in section A.2, and repeated below:

$$L(\underline{\theta}) = p(Z(n)\,|\,\underline{\theta}) \qquad \text{(A.4-2)}$$

Likelihood is the probability density for an observed set of data Z(n) conditioned upon the set of parameters, $\underline{\theta}$. Using the following definitions:

$$J(\underline{\theta}) = -\ell n\, L(\underline{\theta}) \qquad \text{(A.4-3a)}$$

$$g_i(\underline{\theta}) = \frac{\partial}{\partial \theta_i} J(\underline{\theta}) \qquad \text{(A.4-3b)}$$

and

$$H_{ij}(\underline{\theta}) = \frac{\partial^2}{\partial \theta_i \partial \theta_j} J(\underline{\theta}) \qquad \text{(A.4-3c)}$$

We can simplify A.4-1 as follows:

$$E\left\{H_{ij}\right\} = E\left\{g_i g_j^T\right\} \qquad \text{(A.4-4)}$$

The above equation shows that gradient information alone can be used to calculate the Hessian. Gupta and Mehra [1] derive the following approximate formula for the Hessian:

$$H_{ij} = \sum_{k=1}^{N} \left(\frac{\partial \varepsilon_k}{\partial \theta_i}\right)^T (R_k^{\epsilon})^{-1} \frac{\partial \varepsilon_k}{\partial \theta_j} + 1/2\ \mathrm{tr}\left[(R_k^{\epsilon})^{-1} \frac{\partial R_k^{\epsilon}}{\partial \theta_i} (R_k^{\epsilon})^{-1} \frac{\partial R_k^{\epsilon}}{\partial \theta_j}\right] + 1/4\ \mathrm{tr}\left[(R_k^{\epsilon})^{-1} \frac{\partial R_k^{\epsilon}}{\partial \theta_i}\right] \mathrm{tr}\left[(R_k^{\epsilon})^{-1} \frac{\partial R_k^{\epsilon}}{\partial \theta_j}\right] \qquad \text{(A.4-5)}$$

where ϵ_k and R_k^ϵ are, respectively, the innovations and innovations covariances resulting from the model-based signal processing (Kalman filtering) of the measurement data. ϵ_k and R_k^ϵ are dependent on the parameter set, $\underline{\theta}$, because the system model in the Kalman filter is dependent on $\underline{\theta}$. The partial derivatives of ϵ_k and R_k^ϵ are computed numerically, concurrently with $\underline{g}$, as follows:

$$g_i := \frac{\partial J(\theta)}{\partial \theta_i} = \frac{J(\theta + \Delta\theta^{(i)}) - J(\theta)}{\Delta\theta_i} \qquad \text{(A.4-6)}$$

$$\frac{\partial \epsilon_k}{\partial \theta_i} = \frac{\epsilon_k(\theta + \Delta\theta^{(i)}) - \epsilon_k(\theta)}{\Delta\theta_i} \qquad \text{(A.4-7)}$$

$$\frac{\partial R_k^\epsilon}{\partial \theta_i} = \frac{R_k^\epsilon(\theta + \Delta\theta^{(i)}) - R_k^\epsilon(\theta)}{\partial\theta_i} \qquad \text{(A.4-8)}$$

$$(i = 1, \ldots, p)$$

where $\Delta\theta^{(i)}$ is a p-vector whose only non-zero element is the i^{th} element, which is $\Delta\theta_i$. Recall that the Kalman filter must be run p+1 times in order to calculate these partial derivatives.

A.4.2 Hessian Inversion Techniques

We stated in section A-3 that to insure that a gradient algorithm approaches a true minimum, the h matrix in equation A.3-1 must be positive definite. h, in the Gauss-Newton method (and in the Levenberg-Marquardt method as β approaches zero), would normally be equal to the inverse of the Hessian. However, if and only if the Hessian is positive definite, will h also be positive definite. We introduce here methods of modifying the Hessian inversion procedure so that h is guaranteed positive definite.

We first normalize the Hessian matrix, with respect to its diagonal elements:

$$H' = D^{-1}HD^{-1} \qquad \text{(A.4-6)}$$

where D is a diagonal matrix defined as follows:

$$D_{ii} = H_{ii}^{1/2} \quad ; \; D_{ij} = 0 \text{ if } i \neq j$$

If H is positive definite, then H' is also; therefore all of the eigenvalues of H' are greater than zero. If we decompose H' into its eigenvalues and eigenvectors, we can say:

$$H^{-1} = D^{-1}(H')^{-1}D^{-1}$$

$$= D^{-1}\left(\sum_{i=1}^{p} \underline{e}_i \underline{e}_i^T \lambda_i^{-1}\right) D^{-1} \qquad \text{(A.4-7)}$$

where λ_i is the i^{th} eigenvalue of H', and $\underline{e}_i$ is the eigenvector corresponding to the i^{th} eigenvalue.

Suppose some of the eigenvalues of H' are less than or numerically close to zero. (By numerically close, we mean relative to unity; since the matrix H' is normalized, the positive eigenvalues should typically be on the order of one.) What is done now depends on which minimization algorithm we are using. In the Gauss-Newton algorithm, eigenvalues less than some small value (say 10^{-6}) are not included in the summation shown in equation A.4-7.

$$h = D^{-1}\left(\sum_{\substack{\text{all } \lambda\text{'s} \\ >\lambda_{min}}} \underline{e}_i \underline{e}_i^T \lambda_i^{-1} \right) D^{-1} \qquad \text{(A.4-8)}$$

where $\lambda_{min} > 0$

In the Levenberg-Marquardt algorithm, the Marquardt parameter is added to each of the eigenvalues in equation A.4-7, provided the parameter is large enough to force all of the eigenvalues to be greater than zero. If the Marquardt parameter is not large enough, it is set equal to the absolute value of the smallest (negative) eigenvalue (plus some small number), so that it does become large enough.

$$h = D^{-1}\left[\sum_{i=1}^{p} \underline{e}_i \underline{e}_i^T (\lambda_i + \beta)^{-1} \right] D^{-1} \qquad \text{(A.4-9)}$$

where $\beta = \text{Max}\left[\beta, \beta_{min}, \left|\text{Min } \lambda_i;\ i=1, \ldots, p\right|\right]$

$\beta_{min} > 0$

In the next section, we discuss convergence criteria and parameter estimate accuracy.

A.5 Algorithm Convergence and Parameter Estimate Accuracy

Theoretically, a minimization algorithm has converged to a solution when the gradient vector becomes zero. Of course, with computer numerical accuracy, attaining an exact zero is not practical. Even convergence to near zero gradient may require an intolerably large number of tiny steps in theta space while the theta values may, all along, be within "acceptable" limits of accuracy. So the question of convergence may really be a question of the desired accuracy in the parameter estimates. On the other hand, due to a finite signal to noise ratio and limited amount of measurement data available, the desired parameter accuracy may not even be theoretically attainable.

We shall first discuss algorithm convergence criteria and then address the issue of parameter estimate accuracy. We can typically choose one of three types of convergence criteria for a minimization algorithm: 1) near zero gradient, 2) small change in the parameters or 3) small change in the cost function. Typically a user may wish to employ all three types of criteria to make sure the algorithm stops at some point. (MXLKID also includes a provision to detect divergence or extremely slow convergence by placing a user specified limit on the total number of iterations allowed.)

The MXLKID program has a provision for stopping the algorithm when any convergence criterion is achieved and user interaction is not in effect. If user interaction is in effect, the program will prompt the user, who must then specifically end the program by typing "END." The user (in interactive mode) also has the option of continuing the algorithm (or even changing the convergence criteria) as that point. Refer to section 3.6.1 for details of user interaction.

The three convergence tests are as follows:

1. Norm of the Gradient:

$$\|G\| =: \sum_{i=1}^{KTH} G_i^2 \qquad \text{(A.5-1)}$$

$$\|G\| < CRIT(1) \quad \Longrightarrow \text{Convergence}$$

2. Small Change in the Parameters:

Let $\hat{\theta}_{old}$ represent the old (previous iteration) parameter estimates, and $\hat{\theta}_{new}$ represent the current estimates. The significant change in the parameter estimate is computed by:

$$S = -\log_{10}\left[\max_{i=1,ith} |\hat{\theta}_{old}(i) - \hat{\theta}_{new}(i)|\right] \qquad \text{(A.5-2)}$$

S can be interpreted as the number of significant digits in the parameter estimate.

$$S > CRIT(2) \quad \Longrightarrow \text{convergence}$$

3. Small Change in the Cost Function:

$$|J(\hat{\theta}_{old}) - J(\hat{\theta}_{new})| < CRIT(3) \Longrightarrow \text{convergence}$$

Parameter estimate accuracy is the final topic of discussion. The maximum likelihood method of parameter identification has a distinct advantage of providing error bounds concurrently with the parameter estimates. This is due to the fact that the variance of the parameter estimates are related to the Hessian matrix in the following manner (according to the Cramer-Rao bound): [13]

$$E\left\{(\underline{\hat{\theta}} - \underline{\theta})\ (\underline{\hat{\theta}} - \underline{\theta})^T\right\} = H^{-1} \qquad \text{(A.5-3)}$$

Therefore

$$95\%\ \text{Confidence Interval} = 1.96\ (H^{-1})_{ii}^{1/2} \qquad \text{(A.5-4)}$$

The i^{th} diagonal element of the "information matrix" (inverse of the Hessian) is the variance of the i^{th} parameter estimate.* These confidence intervals are printed out at the end of MXLKID program execution and apply to the parameter estimates at the final iteration.

* Variance = standard deviation squared. 95% confidence interval = 1.96 times the standard deviation for Gaussian random variables. Please refer to Papoulis[9] or any standard text on probability for a discussion of these terms.

APPENDIX B

STATISTICAL TEST PACKAGE

NXLKID also includes a feature to test the statistical properties of the innovations sequence $\{\varepsilon_k\}$ -- a procedure used to indicate proper filter performance. It is well known [e.g., see Mehra [14]] that a necessary condition for optimum Kalman filter performance is that the innovations are a zero mean, independent (white) sequence. These properties are satisfied when the filter is properly tuned and the estimator model is satisfactory. In fact, testing properties of $\{\varepsilon_k\}$ enables the user to evaluate how well the *estimator model "matches" reality. These simple tests can also be used to aid* the designer in "tuning" the filter (adjusting Q, R, $\tilde{\pi}_0$) for satisfactory performance.

The first property evaluated is the independence or whiteness of $\{\varepsilon_k\}$. We assume that the filter has reached a statistical steady-state and that the innovations sequence is ergodic (time average equals ensemble average). The sample mean and sample autocorrelation are estimated using

$$\hat{\bar{\varepsilon}}(N) = \frac{1}{N}\sum_{i=1}^{N}\varepsilon(i) = \frac{1}{N}\begin{bmatrix}\sum_{i=1}^{N} \varepsilon_1(i)\\ \vdots \\ \sum_{i=1}^{N} \varepsilon_p(i)\end{bmatrix} \qquad (B\text{-}1)$$

and

$$\hat{R}^{\varepsilon}(\ell) = \frac{1}{N}\sum_{i=\ell+1}^{N}\left(\varepsilon(i) - \hat{\bar{\varepsilon}}(N)\right)\left(\varepsilon(i-\ell) - \hat{\bar{\varepsilon}}(N)\right)^{T} \qquad (B\text{-}2)^{*}$$

*Since the R^{ε} is the assumed covariance of a white sequence, we calculate only the diagonal covariances and assume the cross terms are null. Statistical tests can also be implemented to validate this assumption.

The normalized autocorrelation, $\hat{\rho}^{\epsilon}(\ell)$, is then calculated and tested. Asymptotically, for large N, it can be shown that $\hat{\rho}^{\epsilon}(\ell) \sim N(0, I/\sqrt{N})$ for $\ell \neq 0$; therefore, the 95% confidence limits are: (see Mehra [14] for details)

$$-1.96/\sqrt{N} \leq \hat{\rho}^{\epsilon}_{ii}(\ell) \leq 1.96/\sqrt{N} \quad (N \geq 30) \tag{B-3}$$

where

$$\hat{\rho}^{\epsilon}_{ii}(\ell) = \frac{\hat{R}^{\epsilon}_{ii}(\ell)}{\hat{R}^{\epsilon}_{ii}(0)}$$

Thus, if less than %5 of the $\hat{\rho}_{ii}$ exceeds the limits of (B-3), we are 95% confident that the innovations sequence is white. Tighter bounds could be constructed, however, for tuning purposes this test is satisfactory.

The second property evaluated is a test for zero mean. Since the sample mean is a linear combination of Gaussian random variables, it is distributed: $\hat{\epsilon}(\cdot) \sim N(0, R^{\epsilon}/N)$. Again the 95% confidence interval can be constructed using the sample covariance of (B-2), i.e.,

$$-1.96\sqrt{\hat{R}^{\epsilon}_{ii}(0)/N} \leq \hat{\bar{\epsilon}}(N) \leq 1.96\sqrt{\hat{R}^{\epsilon}_{ii}(0)/N} \quad (N > 30) \ . \tag{B-4}$$

In the sample problem of section 3.2, we performed these tests (see Fig. (B-1). We see that innovations are near zero mean and white. In the MXLKID algorithm, the whiteness test is performed on the set of innovations corresponding to the final set of parameter estimates.

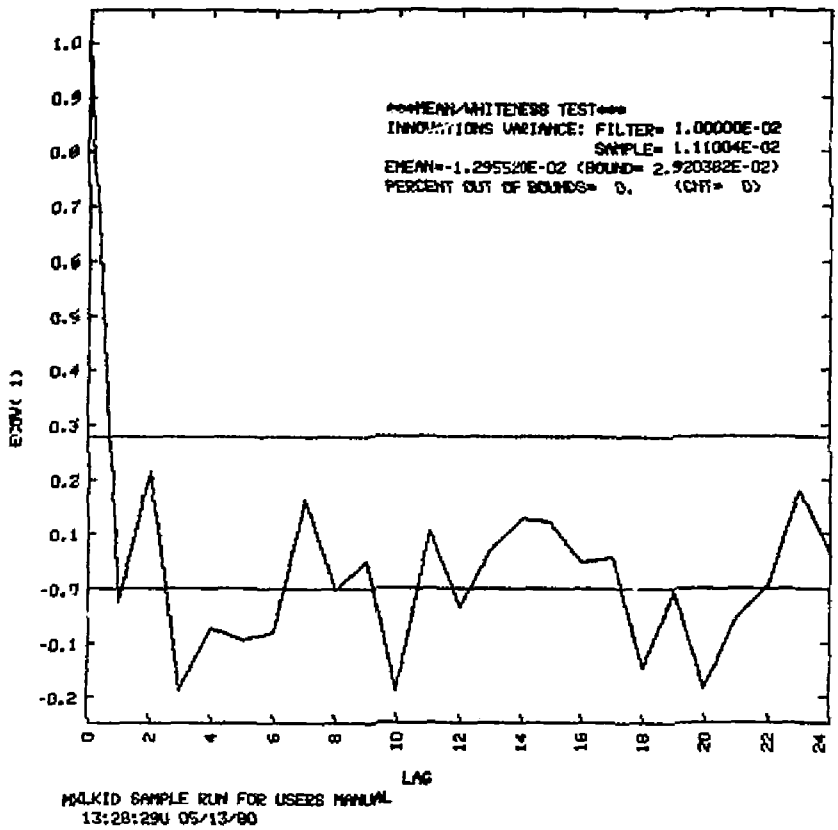


Figure B-1. Mean/Whiteness Test

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $3.50

Page Range	Domestic Price	Page Range	Domestic Price
001-025	$ 5.00	326-250	$18.00
026-050	6.00	351-375	19.00
051-075	7.00	376-400	20.00
076-100	8.00	401-425	21.00
101-125	9.00	426-450	22.00
126-150	10.00	451-475	23.00
151-175	11.00	476-500	24.00
176-200	12.00	501-525	25.00
201-225	13.00	526-550	26.00
226-250	14.00	551-575	27.00
251-275	15.00	576-600	28.00
276-300	16.00	601-up[1]	
301-325	17.00		

[1] Add 2.00 for each additional 25 page increment from 601 pages up.