The Effect of a Gap Nonlinearity on Recursive Parameter Identification Algorithms

by

Scott E. Schaffer

B.E. Hofstra University (1985)

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

Aeronautics and Astronautics

at the

Massachusetts Institute of Technology

May 1988

Signature of Author _____

Department of Aeronautics and Astronautics
May 6, 1988

Certified by _____

Prof. Andreas von Flotow
Thesis Supervisor, Department of Aeronautics

Accepted by _____

Prof. Harold Y. Wachman
Chairman, Department Graduate Comittee

Contents

List of Figures

Abstract

This work examines the performance of the Recursive Least Squares (RLS) Algorithm and a derivative, the Recursive Lattice Least Squares (RLLS) Algorithm in matching a linear model to a simple nonlinear system. The response of a single degree of freedom mass-spring-dashpot system to continuous forcing is simulated, and estimates for the modal parameters are obtained. Nonlinearity is introduced by allowing the restoring spring to slide without friction in a gap of specified width. Such a nonlinearity is of interest since it is a simple model of the effect of loose joints in a deployable spacecraft structure. The effect of the gap size on the algorithms is studied, and the noise level and sampling rate are adjusted to see how they influence these results. The RLS Algorithm is found to be the most reliable. Finally, the RLS algorithm is extended to include an estimate for gap width, and good results are obtained for the noise-free case. Additional conclusions are made on how to minimize the effect of such a gap nonlinearity on the identification process.

Acknowledgements

I am grateful for the support of the following people, who made my time studying at MIT so pleasant:

Professor Andy von Flotow, whose inciteful questions during our discussions of my work were always able to show any flaws in my reasoning.

Gary Blackwood, who helped to make me feel at home in the Facility for Experimental Structural Dynamics, and who showed me how to make full use of the resources at MIT.

Erwin and Barbara Schaffer. my parents, who are always interested in my work and give lots of encouragement.

The members of the MIT Ballroom Dance Club, who provided a wonderful alternative to studying seven days a week.

Notation

System Notation:

$\overline{A}$	average displacement amplitude
c	damping constant
k	spring stiffness
M	mass
u	input (force)
y	output (displacement)
y_0	location of boundary within gap
$y(0)$	initial displacement

Modal Parameters:

ω	natural frequency
ς	damping ratio
M	mass
δ	gap width

Algorithm Variables:

a, a^*	coefficients of outputs in ARMAX form (RLLS)
b, b^*	coefficients of inputs in ARMAX form (RLLS)

e errors in equations (RLS)

$\underline{e}$ forward eqn errors, used as regressors (RLLS)

G forward coefficient matrix $\left\{ \begin{array}{c} \underline{a}^T \\ \\ \underline{b}^T \end{array} \right\}$ (RLLS)

H backward coefficient matrix $\left\{ \begin{array}{c} \underline{a}^{*T} \\ \\ \underline{b}^{*T} \end{array} \right\}$ (RLLS)

H unknown coefficients of parameters (RLS)

K, K^* matrix of coefficients to regressors $\underline{r}$, $\underline{e}$ (RLLS)

$\underline{k}, \underline{k}^*$ Kalman gain factor (RLLS)

n number of steps in regression (RLLS)

P, P^* covariance matrix (RLLS)

$\underline{p}$ desired parameters (RLS)

q^{-1} backwards time shift (RLLS)

$\underline{r}$ backward eqn errors, used as regressors (RLLS)

z known quantities (RLS)

γ parameter for variance estimate (RLLS)

λ forgetting factor (1.0 for no forgetting) (RLLS)

(Note that starred variables indicate use in the backwards difference equation in the RLLS algorithm)

Method of Averaging Variables:

A The constant amplitude chosen for study

$A(t)$ displacement amplitude as a function of time

α_i small angle variables for use in computing boundaries

β change in phase angle as a function of time

$\phi(t)$ frequency as a function of time

Chapter 1

Introduction

1.1 Background

As man's experience with space has grown, the objects placed into orbit have become larger and more complicated. Current project designs will require an entirely new type of structure to provide the required surface area or volume. Large sensor arrays, wide mirrors for astronomy, and truss structures like the proposed space station all fall into this category of Large Space Structures (LSS). Several NASA experiments, including the MAST flight experiment and the COFS (Control of Flexible Structures) program, deal with this this type of structure.

Accurate modelling of large spacecraft structures presents many novel difficulties. It is impossible to conduct full size ground tests of the structure because it is so large, and any finite element model is therefore of limited accuracy. Perversely, the need for active control of the structural dynamics makes accurate modelling necessary. One solution to this problem is to perform model or parameter identification, which when applied to structural dynamics is known as modal analysis. This involves the use of a computer algorithm to transform the motion of the structure into parameters that can be used in a model for control purposes. Experimental modal analysis of large spacecraft structures presents unique challenges, however. Because a structure in space cannot be secured, it will have several rigid body modes. A further complication is that the structure

may require the application of control forces, even while parameter identification is in progress. Lastly, since it is being assembled on orbit, it may include relatively loose joint hardware, designed to enhance ease of assembly. These joints introduce important nonlinearity into the response. [11]

If modal analysis is used to analyze these spacecraft structures, it must have several important characteristics:

- Recursive. The algorithm must be capable of frequently updating the estimates of parameters, to allow the structure to be modelled even in the process of modification. More importantly, recursive algorithms require less data storage, since they only work with the most current information. This more than compensates for then disadvantage that recursive (or 'on-line') algorithms usually do not give as accurate results as batch (or 'off-line) algorithms.

- Allow Input Superposition. To prevent rigid body motion or excessive deflections during the tests, control forces would have to be applied while the test was conducted. Algorithms that require a specific forcing function (such as impulse or sinusoidal) could not be used.

- Robust. The algorithm must be able to perform in the presence of the nonlinearity introduced by the loose joint hardware, and give reliable estimates.

Recently, some time-domain parameter identification algorithms have been proposed which have the first two characteristics, but there has been little study of their performance in the face of nonlinearity. If these algorithms are to be used with confidence in a parameter identification process on a large spacecraft structure, this performance must be examined.

1.2 Thesis Outline

The most common type of nonlinearity found in the joint hardware of truss beams is a gap (deadband) nonlinearity. Therefore, in chapter 2, a simple model containing a gap nonlinearity is chosen and analyzed, and the equations of motion are developed. This nonlinear system will be used to test the identification algorithms.

Two Least-Squares based algorithms are chosen for study, a basic Recursive Least Squares (RLS) Algorithm, and a Recursive Lattice Least Squares (RLLS) Algorithm as formulated by Voss. These algorithms, along with other recursive algorithms, are clearly explained in a text by Ljung on recursive identification. [9] The basic theory of the chosen algorithms is explained in chapter 3. Chapter 4 then gives details on the implementation of the model and algorithms as part of the computer simulation, including the choice of forcing function and programming details.

The effect of the nonlinearity on the modal parameter estimates (natural frequency, damping ratio, and mass) are studied for both algorithms. The influence of noise, sampling rate, and (for the RLLS algorithm) the noise parameter γ are also examined. These are detailed in Chapter 5. A linear relationship is found between the size of the gap and the estimated parameters. Noise, sampling rate, and choice of γ are found to have a significant effect on these estimates. An attempt to improve the identification process is made by extending the RLS algorithm to include an estimate for the gap, and good results are obtained for the noise-free case. The results of this extension are shown in Chapter 6.

The RLS algorithm is found to be generally superior to the RLLS algorithm. Knowledge gained from studying the effect of the gap on the three algorithms leads to several conclusions of how to minimize the nonlinear effect on the identification process.

Chapter 2

The Nonlinear Model

2.1 Analytical Model

A truss structure assembled in space typically will have a static load-deflection curve as shown in Figure 2.1, indicating a gap (deadband) nonlinearity and slight hysteresis. In order to study the effect of this gap (without the hysteresis) on recursive parameter identification (PID) algorithms, a single degree of freedom mass-spring-dashpot model is chosen (figure 2.2).

The boundary of the system is allowed to move in a frictionless gap of arbitrary width, 2δ. This yields the following equations of motion:

$$M\ddot{y} + c\dot{y} + k(y - \delta) = u \tag{2.1}$$

$$M\ddot{y} + c\dot{y} + k(y + \delta) = u \tag{2.2}$$

$$M\ddot{y} = u \tag{2.3}$$

$$c(\dot{y} - \dot{y}_0) + k(y - y_0) = 0 \tag{2.4}$$

where (2.1) is for motion on the right side of the gap, (2.2) is for motion on the left side of the gap, and (2.3-2.4) is for motion inside the gap. The boundaries between the states occur when

$$c(\dot{y} - \dot{y}_0) + k(y - y_0) = 0$$

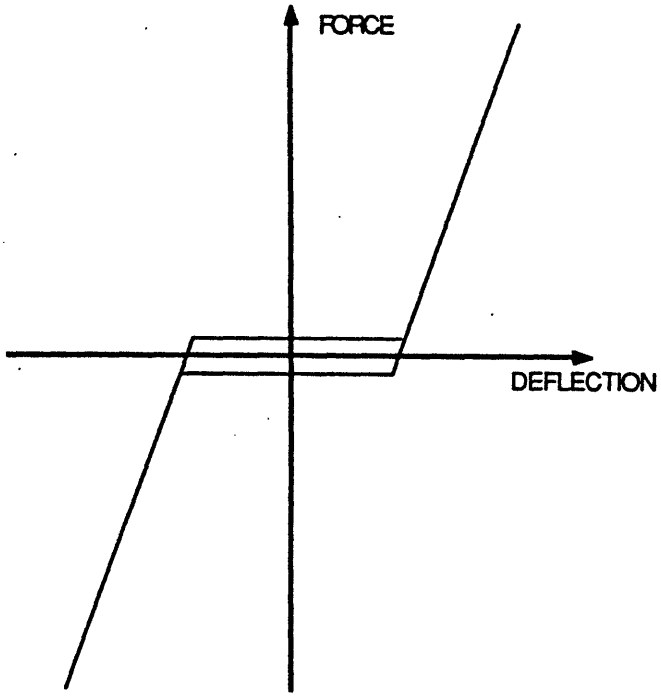


Figure 2.1: Typical Load–Displacement Curve for Truss Structure.

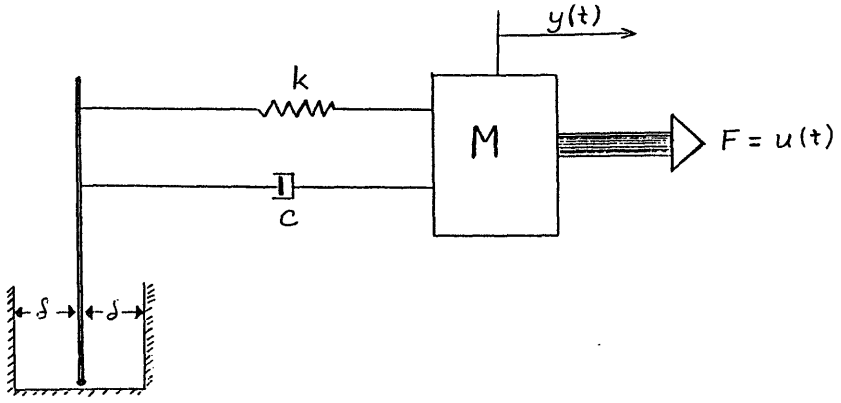


Figure 2.2: Mass-Spring-Dashpot Model for System with Gap Nonlinearity.

or

$$y = y_0 - \frac{c}{k}(\dot{y} - \dot{y}_0)$$

where the stop location $y_0 = \delta$ at the right boundary and $y_0 = -\delta$ at the left boundary. The sign of the $\dot{y}_0$ term indicates the direction the system is moving across the boundary condition. This model has desired the type of Load-Displacement curve.

Since the parameters of interest are the modal parameters of natural frequency (ω), damping ratio (ς), and modal mass (m), the equations of motion are rewritten using the identities $c = 2\varsigma\omega m$ and $k = m\omega^2$. This produces equations of the form

$$\ddot{y} + 2\varsigma\omega\dot{y} + \omega^2(y - \delta) = u/m \tag{2.5}$$

$$\ddot{y} + 2\varsigma\omega\dot{y} + \omega^2(y + \delta) = u/m \tag{2.6}$$

$$\ddot{y} = u/m \tag{2.7}$$

$$2\varsigma\omega(\dot{y} - \dot{y}_0) + \omega^2(y - y_0) = 0. \tag{2.8}$$

2.2 Adaptation for Simulation

In order to make these equations useful for computing a displacement history, they are changed to finite difference equations using the Central Difference Method. This method is chosen because of its simplicity, stability, and lack of numerical damping. This explicit integration method uses the following approximations for the derivatives:

$$\dot{y} \approx \frac{1}{2\Delta t}(y_{t+\Delta t} - y_{t-\Delta t})$$

$$\ddot{y} \approx \frac{1}{\Delta t^2}(y_{t+\Delta t} - 2y_t + y_{t-\Delta t})$$

So the equations of motion may be written explicitly to solve for $y_{t+\Delta t}$ in each of the cases. This gives the final solution for the motion which the computer can solve:

$$y_{t+\Delta t} = \frac{[2-(\omega\Delta t)^2]\, y_t + [-1+\varsigma\omega\Delta t]\, y_{t-\Delta t} + \frac{\Delta t^2}{m} u_t + (\omega\Delta t)^2\, \delta}{(1+\varsigma\omega\Delta t)} \tag{2.9}$$

$$y_{t+\Delta t} = \frac{[2-(\omega\Delta t)^2]\, y_t + [-1+\varsigma\omega\Delta t]\, y_{t-\Delta t} + \frac{\Delta t^2}{m} u_t - (\omega\Delta t)^2\, \delta}{(1+\varsigma\omega\Delta t)} \tag{2.10}$$

$$y_{t+\Delta t} = 2\, y_t - y_{t-\Delta t} + \frac{\Delta t^2}{m} u_t \tag{2.11}$$

$$y_{0\,t+\Delta t} = \frac{(\omega\Delta t)^2}{\varsigma\omega\Delta t}(y_t - y_{0\,t}) + (yt + \Delta t - y_{t-\Delta t} + y_{0\,t-\Delta t}) \tag{2.12}$$

Note that the first two equations differ from the linear system equation by the term

$$\frac{(\omega\Delta t)^2\, \delta}{(1+\varsigma\omega\Delta t)}$$

which may be thought of as a constant noise term, or as a disturbance force. The difference equations are used to calculate the displacement of the system at each time step $y_{t+\Delta t}$, based on the forcing and displacement history. This generates the pseudo-data which is used to test the parameter identification schemes.

In the actual algorithm, a slightly different formulation of the last equation was used. Letting the spring stretch $q = y - y_0$, $\dot{q} = \dot{y} - \dot{y_0}$, where $y_0(0) = -\delta$ and $y(0) = -\delta - \frac{c\dot{y}(0)}{k}$ for the mass moving right in the gap, then $q(0) = -\frac{c\dot{y}(0)}{k}$.

Since the equation of motion is

$$c(\dot{y} - \dot{y_0}) + k(y - y_0) = 0$$

then $\quad c\dot{q} + kq = 0.$

This yields $q(t_g) = q(0)\, e^{-(k/c)\, t_g}$, or

$$\begin{aligned} y_0(t_g) &= y(t_g) + \frac{c\dot{y}(0)}{k} e^{-(k/c)\, t_g} \\ y_0(t_g) &= y(t_g) + \frac{2\zeta}{\omega \Delta t} (y(t_0 - y(t_{-\Delta t})\, e^{-(\omega/2\zeta)\, t_g}. \end{aligned}$$

This shows directly the exponential nature of the relaxation of the spring/dashpot element within the gap. It does however make the assumption that the system returns to equilibrium before touchdown on the other side. In retrospect, working with difference equations directly might be preferable. The displacement is updated in the gap as

$$y_{t+\Delta t} = 2y_t - y_{t-\Delta t} + \frac{\Delta t^2}{M} u_t.$$

The boundary conditions occur when $y = y_0 - \frac{c}{k}(\dot{y} - \dot{y_0})$. Liftoff occurs from the left side when the mass was moving to the right, and $y_0 = -\delta$, $\dot{y_0} = 0$:

$$y_{t+\Delta t} \geq -\delta - \frac{2\zeta}{\omega \Delta t}(y_t - y_{t-\Delta t}).$$

Liftoff occurs from the right side when the mass was moving to the left, and $y_0 = +\delta$, $\dot{y_0} = 0$:

$$y_{t+\Delta t} \leq \delta - \frac{2\zeta}{\omega \Delta t}(y_t - y_{t-\Delta t}).$$

Initial conditions on the displacement are calculated by setting the initial displacement to $y(0)$ and the initial force and velocity to zero. This yields:

$$\frac{1}{\Delta t^2}(y(1) - 2y(0) + y(-1)) + \omega^2(y(0) + \delta) = \frac{u(0)}{M}$$

For $y(-1) \approx y(1)$ (at a minimum point),

$$y(1) = \frac{1}{2}\left[(2 - \omega^2 \Delta t^2)\, y(0) + (\frac{u(0)}{M} - \omega^2 \delta)\, \Delta t^2\right].$$

The actual algorithm is included in Appendix C.

Chapter 3

Identification Algorithms

3.1 Choice of Domain and Algorithms

Identification algorithms are designed to take the frequency response or time history of a system and from this information extract the parameters to fit a specific model. A flowchart of this process is shown in figure 3.1. These parameters can then be used to verify or update a model, improving the accuracy, to determine the effects of a structural modification, or to perform modal synthesis (determining the response of a structure based on the response of its components). Identification algorithms fall into two categories: frequency domain methods and time domain methods.

In the frequency domain, the equations of motion for a linear structural system are typically given as

$$[-\omega^2 M + j\omega C + K]\, y(j\omega) = u(j\omega)$$

or $y(j\omega) = H(j\omega)u(j\omega)$, where $H(j\omega)$ is the frequency response function (FRF) matrix. Fullekrug [4] classifies the frequency domain algorithms into two categories: Sine-Dwell methods, which measure the resonance of a structure directly at its natural frequencies, and FRF techniques which measure responses for closely spaced discrete frequencies and use matrix techniques to extract the parameters.

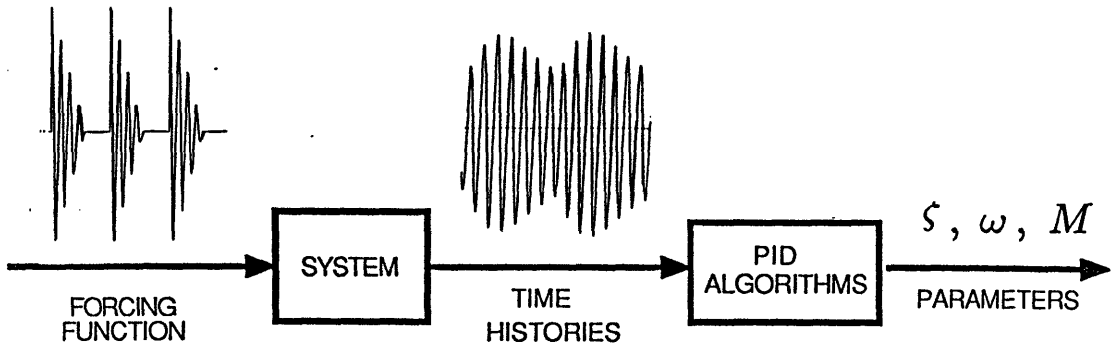


Figure 3.1: Parameter Identification Procedure

In the time domain, the equations of motion may be written as

$$M\ddot{y} + C\dot{y} + Ky = u(t).$$

Fullekrug divides the time domain algorithms into those based on free vibration, and those using forced vibration.

Because for this project a recursive algorithm allowing force superposition was desired, the algorithms was chosen from the time domain algorithms based on forced vibration. These include such Least-Squares based algorithms as the Eigensystem Realization Algorithm (ERA) and Autoregressive Moving Average (ARMA) methods.

Two algorithms were chosen for study, a simple Recursive Least Squares (RLS) algorithm, and a Recursive Lattice Least Squares (RLLS) algorithm. The latter was chosen primarily because of its use for identification in a concurrent experiment by Blackwood. [2]

3.2 The RLS Algorithm

The RLS algorithm is simple in concept, since it is just a slightly modified form of the basic Least Squares formulation. The idea behind a least squares solution is simple. Given a set of equations which linearly involve a set of parameters, one may write them in matrix form:

$$\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} = \begin{bmatrix} \Phi_{11} & \cdots & \Phi_{1m} \\ \Phi_{21} & \cdots & \Phi_{2m} \\ \vdots & \cdots & \vdots \\ \Phi_{n1} & \cdots & \Phi_{nm} \end{bmatrix} \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_m \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}$$

where

z_i = Results

$\underline{\Phi}_i^T$ = Coefficient vector

$\underline{\theta}$ = parameter vector

e_i = error.

If the number of equations exceeds the number of parameters (i.e. $n > m$), the estimated values for the parameters $\underline{\hat{\theta}}$ that produce the minimum sum of the squared errors $\sum e_i^2$ may be found from the equation

$$\underline{\hat{\theta}} = (\Phi^T \Phi)^{-1} \Phi^T \underline{z}.$$

This is referred to as the pseudoinverse. The relation of the pseudoinverse form to a kalman filter formulation is shown in Appendix B.

Since this algorithm is trying to fit the modal parameters of a linear system, $\ddot{y} + 2\varsigma\omega\dot{y} + \omega^2 y = u/M$, the governing equation is put in the form

$$\ddot{y}_i = \begin{bmatrix} -2\dot{y}_i & -y_i & u_i \end{bmatrix} \begin{pmatrix} \varsigma\omega \\ \omega^2 \\ 1/M \end{pmatrix}$$

and evaluated at a sequence of instants to generate a matrix equation. Given the pseudo-data points of u_i and y_i, the values of $\varsigma\omega$, ω^2, and $1/M$ are found that minimize the squared error. The derivatives of y are calculated using the central difference method. The algorithm is made recursive by evaluating the contributions to the 3x3 $\Phi^T\Phi$ and 3x1 $\Phi^T\underline{z}$ matrices at each time step and adding them to the existing values. The new matrices are used to calculate new values of $\underline{\hat{\theta}}$ for each $i > 3$.

3.3 The RLLS Algorithm

The RLLS algorithm is a linear, discrete time, state-space method. It is based on the ARMAX (Autoregressive Moving Average, Exogeneous inputs) form of the linear difference equation, which may be written as

$$y(t) = a_1 y(t-1) + b_1 u(t-1) + \cdots + a_n y(t-n) + b_n u(t-n) + b_0 u(t) + e_n(t)$$

This form is also known as an estimator form, since the output is a linear combination of previous inputs and outputs, and may be used to estimate future output values for a system, as was done in simulating the nonlinear system (see equations 2.9 to 2.12). The ARMAX form has been used for many years on linear systems with excellent results, and the technique is commonly known as Linear Regression.

The RLLS algorithm offers several advantages. Besides being recursive and allowing input superposition, it has fast convergence for small data variance, and it can handle

many degrees of freedom simultaneously. A disadvantage is that it requires strong excitation of the natural frequencies, and so has difficulty with a pseudorandom input.

The RLLS algorithm is unusual because it also makes use of a backwards difference ARMAX equation:

$$y(t-n) = a_0^* y(t) + b_0^* u(t) + \cdots + a_{n-1}^* y(t-n+1)$$
$$+ b_{n-1}^* u(t-n+1) + b_n^* u(t-n) + r_n(t).$$

The two equations then go through a change of basis to make the error terms the regressing coefficients. This derivation is included in Appendix A. The result is a pair of equations which form a lattice (see figure 3.2):

$$\begin{pmatrix} y(t) \\ u(t) \end{pmatrix} = \begin{bmatrix} K_0(t) & K_1(t) & \cdots & K_{n-1}(t) \end{bmatrix} \begin{pmatrix} \underline{r}_0(t-1) \\ \underline{r}_1(t-1) \\ \vdots \\ \underline{r}_{n-1}(t-1) \end{pmatrix}$$

$$\begin{pmatrix} y(t-n) \\ u(t-n) \end{pmatrix} = \begin{bmatrix} K_0^*(t-n+1) & \cdots & K_{n-1}^*(t) \end{bmatrix} \begin{pmatrix} \underline{e}_0(t-n+1) \\ \underline{e}_1(t-n+2) \\ \vdots \\ \underline{e}_{n-1}(t) \end{pmatrix}$$

where K_i and K_i^* are 2x2 Reflection Coefficient matrices and $\underline{e}_i$ and $\underline{r}_i$ are 2x1 error vectors used as regressors.

The Voss formulation of the algorithm makes use of two Kalman filters to solve for the values of K_i and K_i^*. This reduces the number of computations required. According to Ljung [9], this type of formulation produces assymptotic convergence of the estimates with a zero mean, normal distribution, covariance matrix for the errors e and r. He also

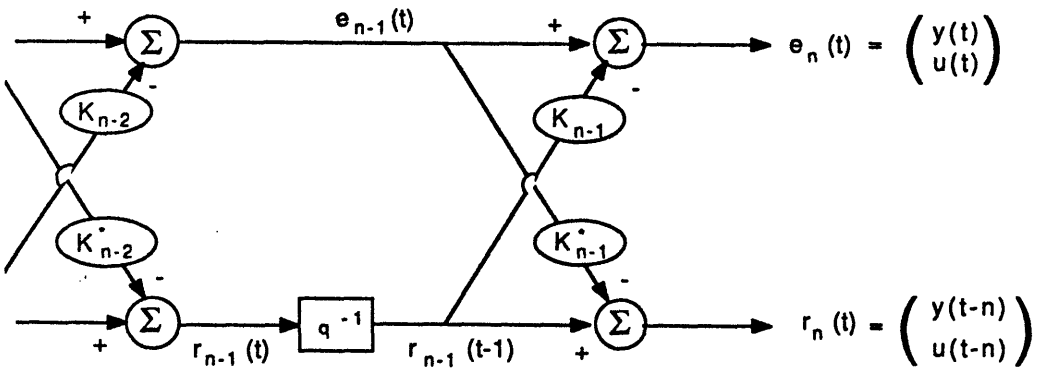


Figure 3.2: Lattice Effect of the Two Equations in the RLLS Algorithm.

mentions that while it works well using a sinusoidal input, it may be bad for white noise input. The Voss RLLS algorithm in algebraic form is included at the end of Appendix A.

Once the K_i and K_i^* have been determined, the desired modal parameters may be extracted by examining the equations of motion in estimator form as computed through the Central Difference Method:

$$\frac{1}{\Delta t^2}(y_{t+\Delta t} - 2y_t + y_{t-\Delta t}) + \frac{1}{\Delta t}(y_{t+\Delta t} - y_{t-\Delta t})\,(\varsigma\omega_0) + y_t\,(\omega_0^2) = u_t\,(\frac{1}{M}).$$

This yields the estimator form:

$$y_{t+\Delta t} = \left[\frac{2 - \omega_0^2\Delta t^2}{1 + \varsigma\omega_0\Delta t}\right] y_t + \left[\frac{\Delta t^2/M}{1 + \varsigma\omega_0\Delta t}\right] u_t + \left[\frac{-1 + \varsigma\omega_0\Delta t}{1 + \varsigma\omega_0\Delta t}\right] y_{t-\Delta t}.$$

The parameters may then be extracted, since

$$\begin{aligned} y_{t+\Delta t} &= G_1 y_t + G_2 u_t + G_3 y_{t-\Delta t} \\ \varsigma\omega\Delta t &= \frac{1 + G_3}{1 - G_3} \\ \omega_0 &= \frac{1}{\Delta t}\sqrt{(2 - G_1(1 + \varsigma\omega\Delta t)} \\ \varsigma &= \frac{\varsigma\omega\Delta t}{\omega\Delta t} \\ M &= \frac{\delta t^2/G_2}{1 + \varsigma\omega\Delta t}. \end{aligned}$$

The RLLS algorithm has been studied in several papers, mostly by Morf for linear models.[7,8] This algorithm has a number of advantages over the simpler RLS algorithm. Unlike the RLS algorithm, with which there is a penalty in accuracy for going beyond the true model order, the RLLS algorithm may be based on a model of arbitrary order. Additionally, all estimates provided by lower order models may be recovered, because only the new information provided by each increase in model order is used to change the estimates. The ability to overspecify the order enables an unknown system to be given

an arbitrary order (based on an overspecified number of degrees of freedom) and a good model obtained by seeing when the results cease to improve for a higher order. Another advantage for selecting the RLLS algorithm is that it it more computationally efficient for large models (n>5), since the computing cycle time is linear with the number of states, as opposed to a quadratic relationship for the RLS. Since our model system is small, this advantage will not be shown here.

A final incentive for choosing this algorithm for study was its use for parameter identification in a concurrent modal synthesis experiment being conducted at M.I.T. by Blackwood.[2] This experiment involves the analysis of a real vibrating jointed beam with a gap nonlinearity. The bending modes of the beam are excited by a shaker apparatus, and screws at the joints may be adjusted to introduce varying amounts of gap into the joint. The experimental beam is shown in Figure 3.3. The spectrum analyzer for this experiment, a Signology SP-20, uses the RLLS algorithm.

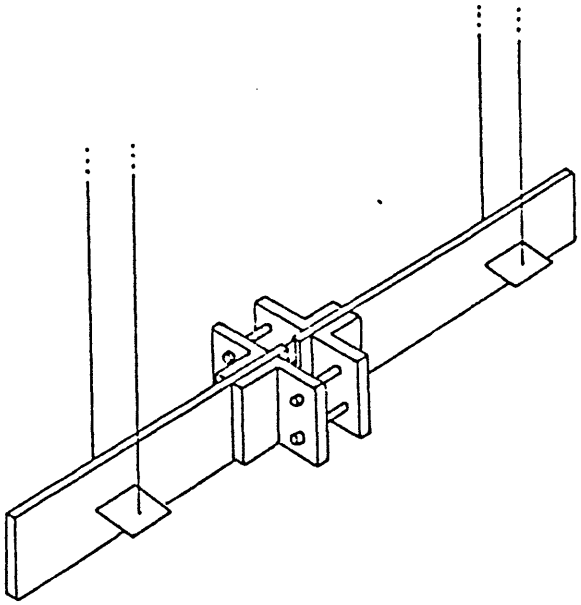


Figure 3.3: Component in the Blackwood experiment in a suspended test configuration.

Chapter 4

Organization of Project

4.1 Choice of Forcing Function

One of the most important criteria when choosing a forcing function is that it is 'persistently exciting', or excites all the modes that need to be identified. For his use of the RLLS algorithm, Morf generally assumes that input is white noise. [7] The first forcing function chosen to excite the system was therefore a random input. This function, however, did not excite the system enough to give good results, and the parameter estimates given by the parameter identification algorithms were different for each random function. Additionally, Soucy and Vigneron [11] recommend against use of random forcing as tending to eliminate the nonlinearity through averaging.

Therefore, the forcing function chosen for this work is a sinusoid in a linear envelope, repeated six times (See Figure 4.1). The basic function was determined by Voss [12] to be optimal for the RLLS frequency estimate, and it is repeated to give the mass and damping ratio estimates time to converge. Too short a time makes the algorithms provide meaningless results for larger gaps, and the RLLS algorithm becomes very sensitive to choice of γ. It was determined by trial that long term forcing, rather than free decay, was needed to give good estimates. This repetition gives better results than a single function with a longer envelope. This forcing function also gives a fairly constant displacement amplitude, which is important when analyzing the influence of a

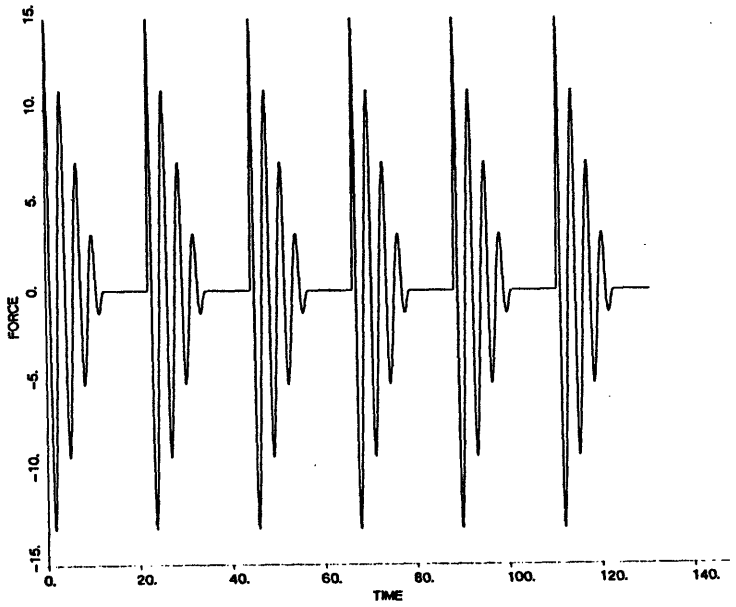


Figure 4.1: Forcing Function

gap nonlinearity. Ideally, the forcing frequency is chosen at the natural frequency of the system, but is chosen here at 95% of this value to simulate modelling error, or inexact response from actuators. Both the input(force) and output(displacement) at each time step were made available to the identification algorithms.

4.2 Implementation of Algorithms

The RLS algorithm is simple to implement. The 3x3 matrix is inverted analytically, allowing the contribution at each time step to be added directly, rather than having to invert the matrix at each time step. The algorithm will not converge exactly in 3 time steps, even for a linear noise-free system, because the Central Difference Method includes terms for y_t and $y_{t-\Delta t}$ but does not account for the force at the previous time step, $u_{t-\Delta t}$, that relates the two displacements.

The RLLS algorithm required considerably more time to implement. Major problems

were caused by numerical instability in the form of roundoff error. It was determined that a change in the 4th decimal place of the input would change the first decimal place of the parameters. The problem was eased by using a UDU^T formulation for the covariance matrix P, which factors the matrix into two triangular matrices and a diagonal matrix as suggested in references [9] and [12]. This formulation reduces the number of addition operations in the matrix multiplications, reducing the chance for roundoff error.

Another cause of delay was that the best sizes of the required values for the parameters P(0) and γ were ambiguously treated in the literature. P(0) is the initial value for the covariance matrix, which requires large values (see Appendix A). Values greater than $200000 \times I_{2x2}$ yielded negligible improvement, so this was chosen. The parameter γ is a scalar weighting factor used to account for noise in the data. More exactly, $\gamma = \lambda\beta$, where λ is the forgetting factor and β is the least squares weighting factor described in Ljung's book. [9] The specified initial value, $\gamma_0 = \lambda V$, is the forgetting factor times the nondimensionalized variance of the data noise. The forgetting factor is used to discount old data if the system is time varying. For our constant system, this parameter should be set to 1 for no forgetting. For a noise-free case, γ could be set to zero, but because of numerical errors, a value of .000001 was found to be suitable. Voss claims that the best results for noisy data occur when V is set to 100 times the true variance.

Because of the forcing function chosen, it was also found necessary to modify the way that the mass parameter was calculated, since a contribution from the force at time $t-\Delta t$ was identified. The force is changed from $u_t(1/M)$ to $u_t(1/M)(x) + u_{t-\Delta t}(1/M)(1-x)$ (since $u_t \approx u_{t-\Delta t}$). This splits the influencing force over the two time steps. In this case, since

$$G_2 = \frac{x\Delta t^2/M}{1+\varsigma\omega_0\Delta t} \qquad G_4 = \frac{(1-x)\,\Delta t^2/M}{1+\varsigma\omega_0\Delta t},$$

Then the mass is

$$M = \frac{\Delta t^2}{(1 + \varsigma\omega_0\Delta t)(G_2 + G_4)}.$$

This yielded the correct mass for the linear noise-free case.

To non-dimensionalize the size of the gap, it should be divided by the total path length that the displacement covers. This was found by writing a FORTRAN program to keep track of the peak displacements. This program, along with the system estimation and parameter identification algorithms, is included in Appendix C.

The simulation was organized into three steps. First the forcing function was computed at discrete time steps. This was then used as input into the system to be studied, and a file of the resulting displacements was created. To simulate a sampling effect, the displacements were recorded for only every other time step. Last of all, both the force and displacement histories were fed into the parameter identification algorithms. Two different versions of the output could be obtained: either showing the identified parameters at each time step, to check things like speed of convergence, or giving all parameters at only the last time step, to save computing time when running through a large number of cases. These final values (after 40 periods) are the ones used in the results section.

Chapter 5

Results

5.1 Addition of the Gap Nonlinearity

The specific system used for the simulations has the following parameters:

$k = 20$ N/m	$\omega = 2$/sec
$c = 0.4$ N-s/m	$\varsigma = .02$
$M = 5.0$ kg	$\delta =$ specified
$y(0) =$ -2 m	Period $= \pi$

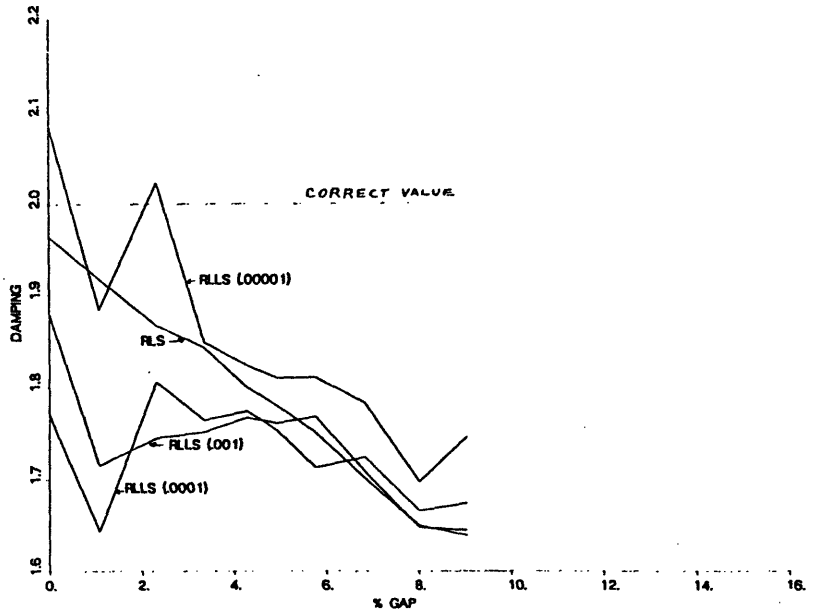
This is then forced by the function

$$u = 15\,(1 - .08t)\cos(1.9t)$$

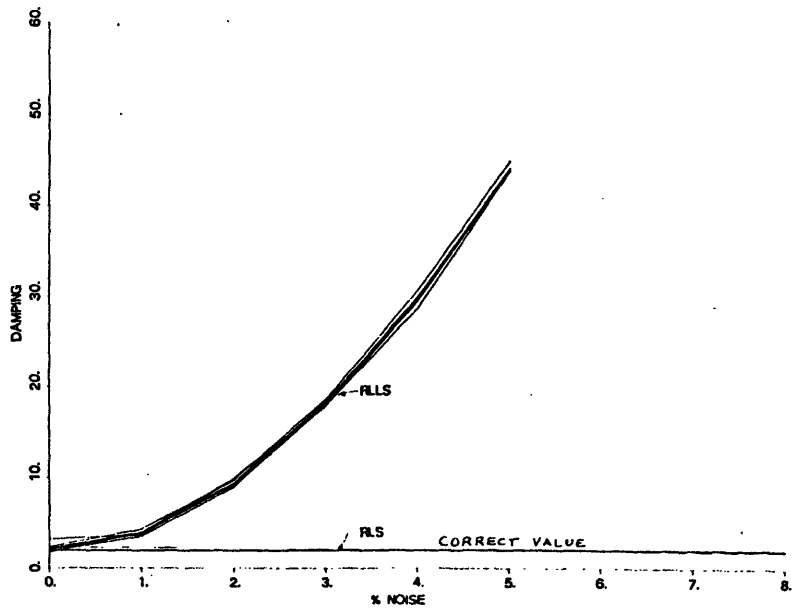
repeated every seven periods, for a total of 42 periods (total time 132 sec).

The standard sampling rate is 30 per period. It is found that at higher rates (such as 60 per period) the RLLS Algorithm proves to be very unstable with the parameter γ, even with the UDU^T and Kalman Filter formulations. The sampling rate chosen proved to be much more stable, probably because of smaller numerical roundoff errors. It should be mentioned that the RLS algorithm does not suffer from these numerical difficulties, and its accuracy improved with the increased sampling rate, but not enough to warrant using different rates for each algorithm.

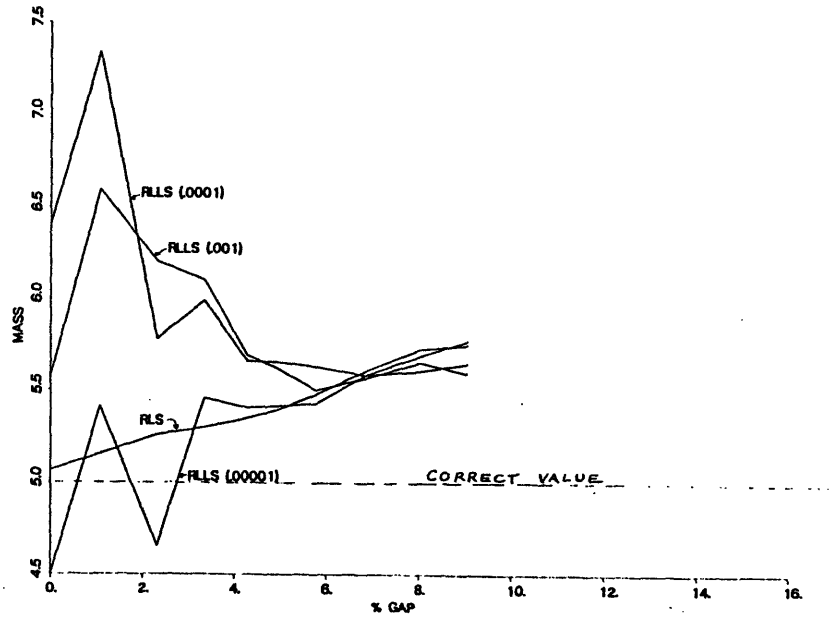
The influence of the gap nonlinearity is studied by use of the nondimensional ratio of $\delta/\overline{A}$ (gap size/average amplitude). This particular ratio is chosen because it determines the fraction of the displacement amplitude that passes through the gap. With this choice, when the gap size approaches zero or the amplitude becomes very large, the results approach those of the linear system. The final estimates for each of the three parameters are plotted versus $\delta/\overline{A}$ in figure 5.1.



a) Frequency Estimates



b) Damping Ratio Estimates



c) Mass Estimates

Figure 5.1: Influence of the Gap Nonlinearity on Frequency, Damping Ratio, and Mass Estimates.

The RLS algorithm shows a linear decrease in the estimates for the natural frequency and damping ratio, and a linear increase in the mass estimate as $\delta/\overline{A}$ increases. The RLLS algorithm gives the same results for frequency, since the forcing has been optimized for this estimate. The RLLS mass and damping estimates are highly dependent on the choice of γ for small gaps ($\delta/\overline{A} < 5\%$), but converge to the RLS estimates for larger gaps. Approximate linear equations for the RLS estimates are:

$$\begin{aligned} \omega_e &= \omega(.997 - .54\,\delta/\overline{A}) \\ \varsigma_e &= \varsigma(.980 - 1.88\,\delta/\overline{A}) \\ M_e &= M(1.011 + 1.54\,\delta/\overline{A}) \end{aligned}$$

These may be used to adjust a linear model to account for a gap. They also could be used to estimate the size of the gap, given a variation between the linear model and the estimated parameters, but surely there are better ways to do this.

At first it seemed that the RLLS estimates were more consistent with increasing $\delta/\overline{A}$ because the system was being forced more closely to its natural frequency (i.e. the natural frequency had dropped to 95% of the linear value because of the gap). However, changing the forcing frequency does not change the trend. Examination of the time traces (see figure 5.2) shows that the estimates for larger gaps converge more quickly. This faster convergence probably comes from what the algorithm sees as increased variance in the data relative to γ, the variance parameter, since it is noted that smaller values of γ (for the same gap size) also make the estimates converge more quickly (see figure 5.3). This will lead to an investigation into the effect of noise on the RLLS estimates.

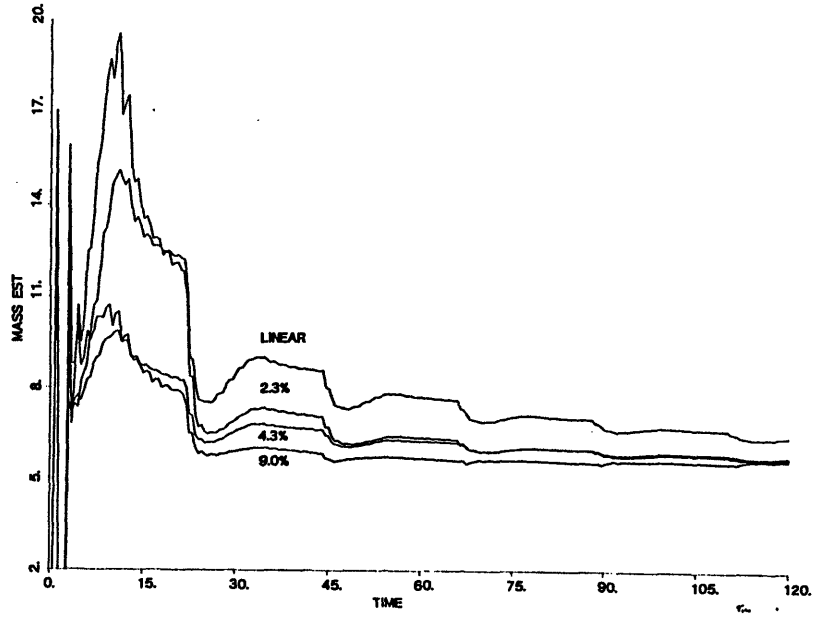


Figure 5.2: Effect of the Gap Nonlinearity on the Convergence of the Mass Estimates.

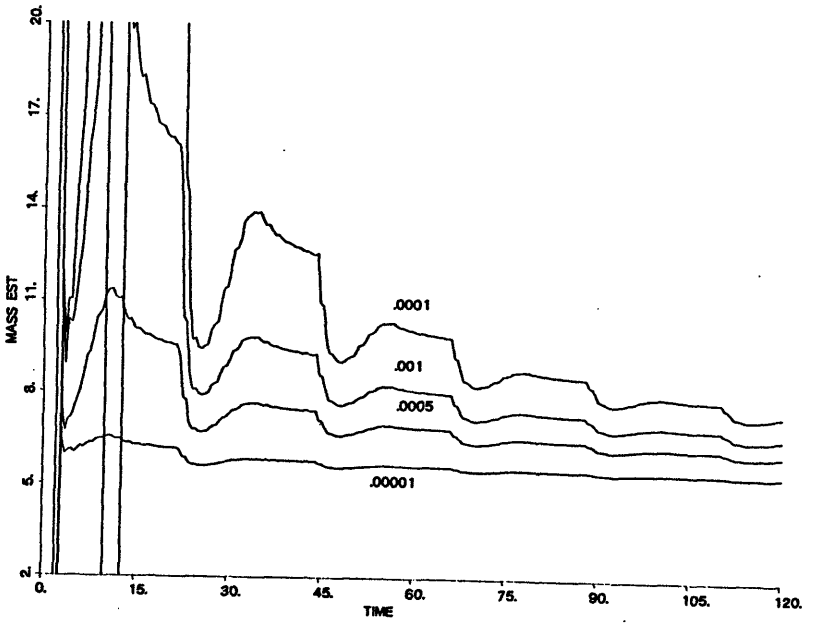


Figure 5.3: Effect of Noise Parameter Gamma on the Convergence of the Mass Estimates.

5.2 An Analytical Comparison

The effect of the gap nonlinearity on the natural frequency of the system may be estimated analytically using the method of averaging. The system is cast in the form

$$\ddot{y} + \omega_0^2 y = f(y, \dot{y}),$$

where

$$\begin{aligned} f(y,\dot{y}) &= -2\varsigma\omega_0\dot{y} + \omega_0^2\delta \\ f(y,\dot{y}) &= -2\varsigma\omega_0\dot{y} - \omega_0^2\delta \\ f(y,\dot{y}) &= \omega_0^2 y \end{aligned}$$

for the right side, left side, and inside of the gap (stipulating an unforced system).

A solution is assumed of the form

$$\begin{aligned} y(t) &= A(t)\cos\phi(t) \\ \phi(t) &= \omega_0 t + \beta(t) \\ \dot{y}(t) &= -A(t)\omega_0 \sin\phi(t). \end{aligned}$$

Then, in terms of ϕ,

$$\begin{aligned} f(A(t)\cos\phi(t), -A(t)\omega_0\sin\phi(t)) &= 2\varsigma\omega_0^2 A(t)\sin\phi(t) + \omega_0^2\delta \\ &= 2\varsigma\omega_0^2 A(t)\sin\phi(t) - \omega_0^2\delta \\ &= \omega_0^2 A(t)\cos\phi(t) \end{aligned}$$

for the right side, left side, and inside of the gap.

Assuming that $A(t)$ and $\phi(t)$ can be treated as constants over one period, the boundary conditions are calculated in terms of ϕ. For $\phi(t) = 0$, $y(t) = A(t)$ the maximum right

displacement. Using small angle assumptions, the following identities are determined:

$$\cos(\pi/2-\alpha_1)\approx\alpha_1 \qquad \cos(\pi/2+\alpha_2)\approx-\alpha_2$$

$$\cos(3\pi/2-\alpha_3)\approx-\alpha_3 \qquad \cos(3\pi/2+\alpha_4)\approx\alpha_4$$

$$\sin((\pi/2-\alpha_1)\approx 1 \qquad \sin(\pi/2+\alpha_2)\approx 1$$

$$\sin(3\pi/2-\alpha_3)\approx 1 \qquad \sin(3\pi/2+\alpha_4)\approx 1.$$

As the phase goes form 0 to 2π, each liftoff and landing condition is encountered. Since the liftoff and landing moving left occur at $\delta-\frac{2\zeta}{\omega_0}\dot{y}$ and $-\delta$:

$$A(t)\cos\phi(t) = y(t) = \delta-\frac{2\zeta}{\omega_0}[-A(t)\omega_0\sin\phi(t)]$$

$$\phi(t) = \pi/2-\frac{\delta+2\zeta A(t)}{A(t)}.$$

and

$$A(t)\cos\phi(t) = y(t) = -\delta$$

$$\phi(t) = \pi/2+\frac{\delta}{A(t)}.$$

Similarly, the liftoff and landing moving right occur at $-\delta-\frac{2\zeta}{\omega_0}\dot{y}$ and δ:

$$A(t)\cos\phi(t) = y(t) = -\delta-\frac{2\zeta}{\omega_0}[-A(t)\omega_0\sin\phi(t)]$$

$$\phi(t) = 3\pi/2+\frac{-\delta+2\zeta A(t)}{A(t)}.$$

and

$$A(t)\cos\phi(t) = y(t) = \delta$$

$$\phi(t) = 3\pi/2+\frac{\delta}{A(t)}.$$

The values for $\dot{A}(t)$ and $\dot{\beta}(t)$ may then be calculated from the following equations:

$$\dot{A}(t) = \frac{-1}{2\pi\omega_0}\int_0^{2\pi} f(A(t)\cos\phi(t), -A(t)\omega_0\sin\phi(t))\sin\phi(t)\ d\phi \tag{5.1}$$

$$\dot{\beta}(t) = \frac{-1}{2\pi\omega_0 A(t)}\int_0^{2\pi} f(A(t)\cos\phi(t), -A(t)\omega_0\sin\phi(t))\cos\phi(t)\ d\phi \tag{5.2}$$

If t is chosen at a point when $A(t) = A$, a constant, this yields

$$\begin{aligned}
\dot{\beta}(t) = \frac{-1}{2\pi\omega_0 A}(& \\
& \int_0^{\pi/2-\frac{\delta}{A(t)}+2\zeta} (2\zeta\omega_0^2 A(t)\sin\phi(t) + \omega_0^2\delta)\cos\phi(t)\ d\phi \\
+ & \int_{\pi/2-\frac{\delta}{A(t)}+2\zeta}^{\pi/2+\frac{\delta}{A(t)}} (\omega_0^2 A(t)\cos\phi(t))\cos\phi(t)\ d\phi \\
+ & \int_{\pi/2+\frac{\delta}{A(t)}}^{3\pi/2+\frac{-\delta}{A(t)}+2\zeta} (2\zeta\omega_0^2 A(t)\sin\phi(t) - \omega_0^2\delta)\cos\phi(t)\ d\phi \\
+ & \int_{3\pi/2+\frac{-\delta}{A(t)}+2\zeta}^{3\pi/2+\frac{\delta}{A(t)}} (\omega_0^2 A(t)\cos\phi(t))\cos\phi(t)\ d\phi \\
+ & \int_{3\pi/2+\frac{\delta}{A(t)}}^{2\pi} (2\zeta\omega_0^2 A(t)\sin\phi(t) + \omega_0^2\delta)\cos\phi(t)\ d\phi \quad).
\end{aligned}$$

The solution is then:

$$\dot{\beta} = -\frac{\omega_0\delta}{\pi A}$$

or (for $\beta(0) = 0$)

$$\beta = -\frac{\omega_0\delta}{\pi A}t.$$

The linear relation between natural frequency and gap size can then be shown by looking at the phase equation:

$$\begin{aligned}
\phi(t) &= \omega_0 t - \frac{\omega_0\delta}{\pi A}t \\
&= \omega_0(1 - \frac{1}{\pi}\frac{\delta}{A})t.
\end{aligned}$$

Thus $\Omega = \omega_0(1 - .318\frac{\delta}{A})$. This has a slightly flatter slope than that obtained from the algorithm, but still shows the linear relation.

5.3 The Effect of Noise on a Linear Model.

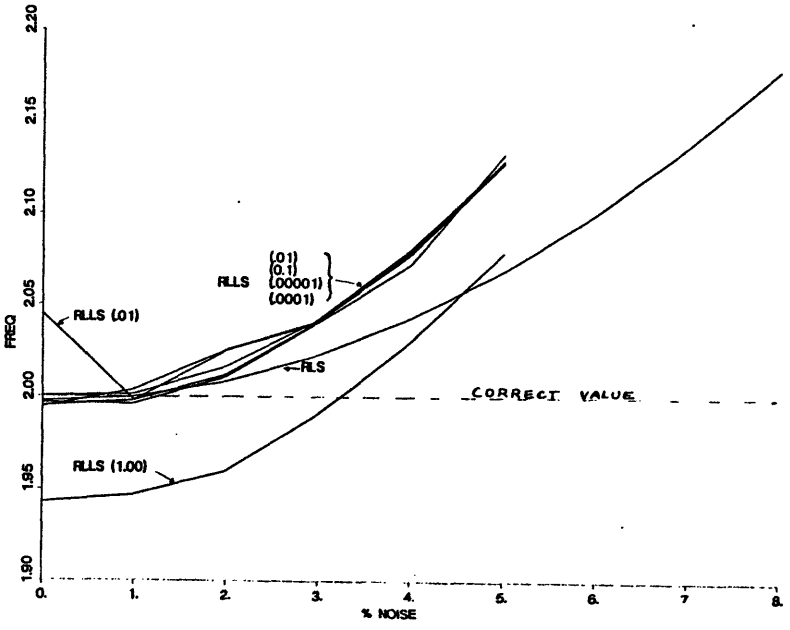
If the gap nonlinearity is indeed seen as adding noise to the system, a study of the effects of noise on the linear model is important. Noisy displacement data is produced by adding a random perturbation to the sensed displacement at each time step. The maximum amplitude of this noise is proportional to the average displacement amplitude, $\Delta y = N\overline{A}$. The proportionality constant N is referred to here as the percent noise. The noisy data is then passed to the two parameter identification algorithms.

This noise is found to have a very large impact, since the estimates of Least Squares-based algorithms have bias errors with correlated noise.[10,12] The modal parameter estimates are plotted versus percent noise in figure 5.4. The RLS algorithm shows quadratic increase in frequency and mass estimates, and quadratic decrease in the damping ratio estimate. The RLLS algorithm shows quadratic increase in frequency and damping estimates, and an inverse relationship for the mass estimate. For estimates to be within 10% of the true values, noise has to be under 5% for the RLS and under 1% for the RLLS algorithm. The mass estimate is the limiting factor in both cases. It is noted that the addition of noise also makes the algorithms converge slightly faster, and makes the RLLS algorithm less sensitive to γ. This is to be expected if the gap is treated as an increase in noise, verifying this hypothesis.

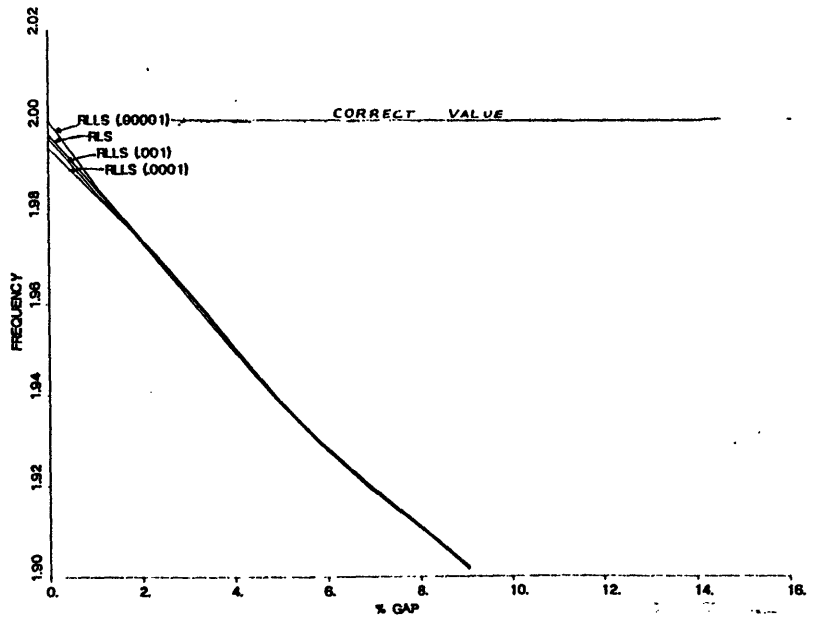
The two algorithms yield the same results when the parameter γ in the RLLS algorithm approaches zero (for the noise-free case). This agrees with theory, and is true as long as the initial estimates are zero.[3]

Adjustment of the parameter γ can be made to compensate for the increase in frequency due to noise, with very small values being best in the noise-free case and larger values needed as the noise is increased. Setting $\gamma = 0$ does not yield a convergent

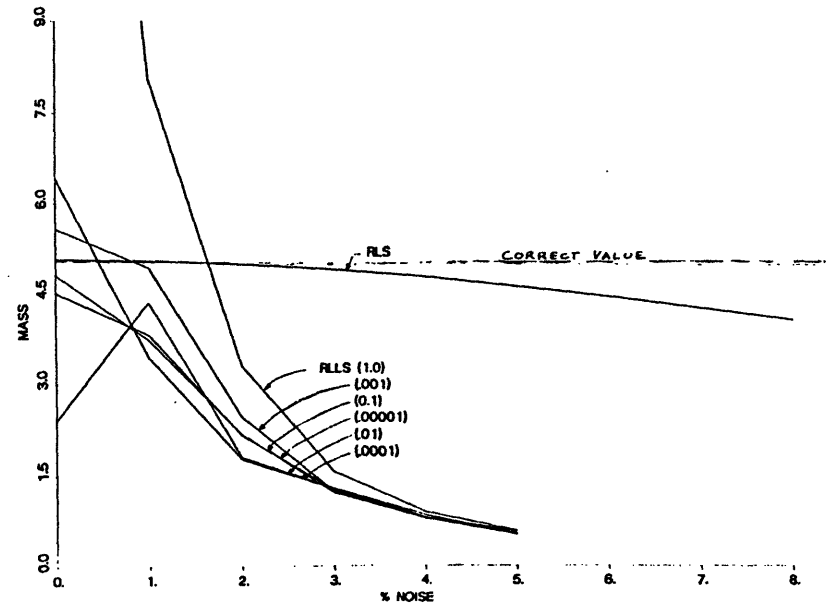
estimate in the noise-free case because of small numerical errors, so a value of $\gamma =$ .000001 is chosen. Voss' rule of thumb of 100 times the true variance seems to work only for percent noise under 1%, since it was found that for 1% noise the best value is .0001 and for 5% noise it is 4.0 . The choice of a larger value of γ allows more data to be processed before the estimates converge, allowing the algorithm to compensate for the noise. The parameter, however, is unable to compensate for the very strong changes in damping ratio and mass estimates with increasing noise. These estimates would have to be normalized somehow for the parameter to affect all the estimates. Without a priori knowledge of the modal parameters, this would be difficult.



a) Frequency Estimates



b) Damping Ratio Estimates

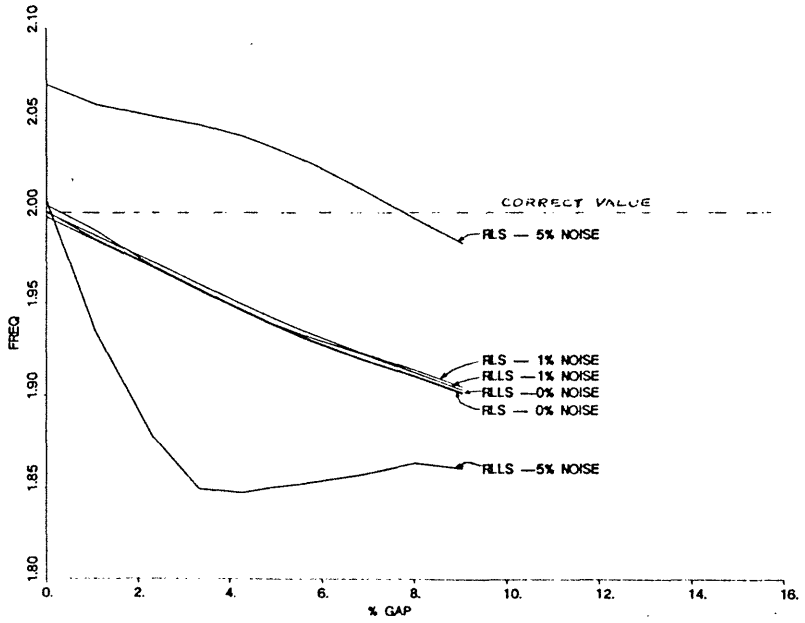


c) Mass Estimates

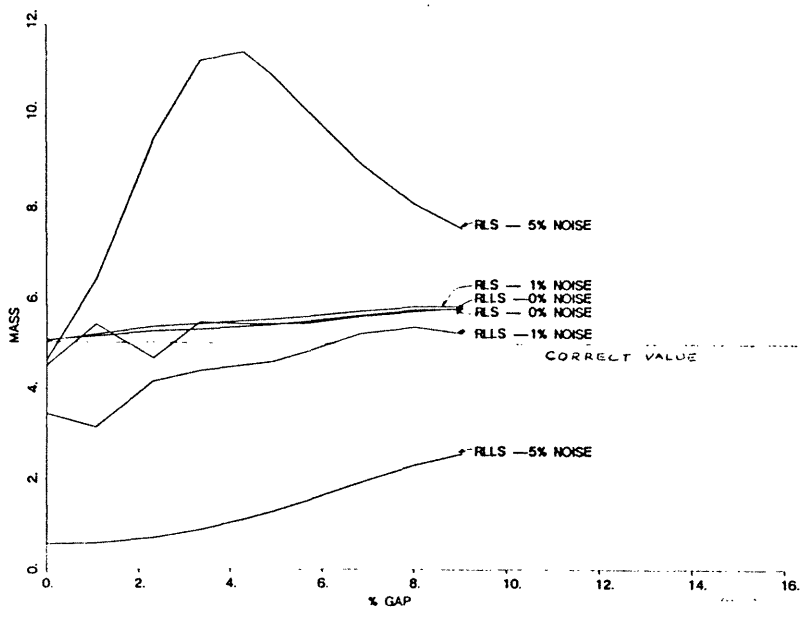
Figure 5.4: Effect of Noise on the Frequency, Damping Ratio, and Mass Estimates.

5.4 Combined Effects of Noise and Nonlinearity

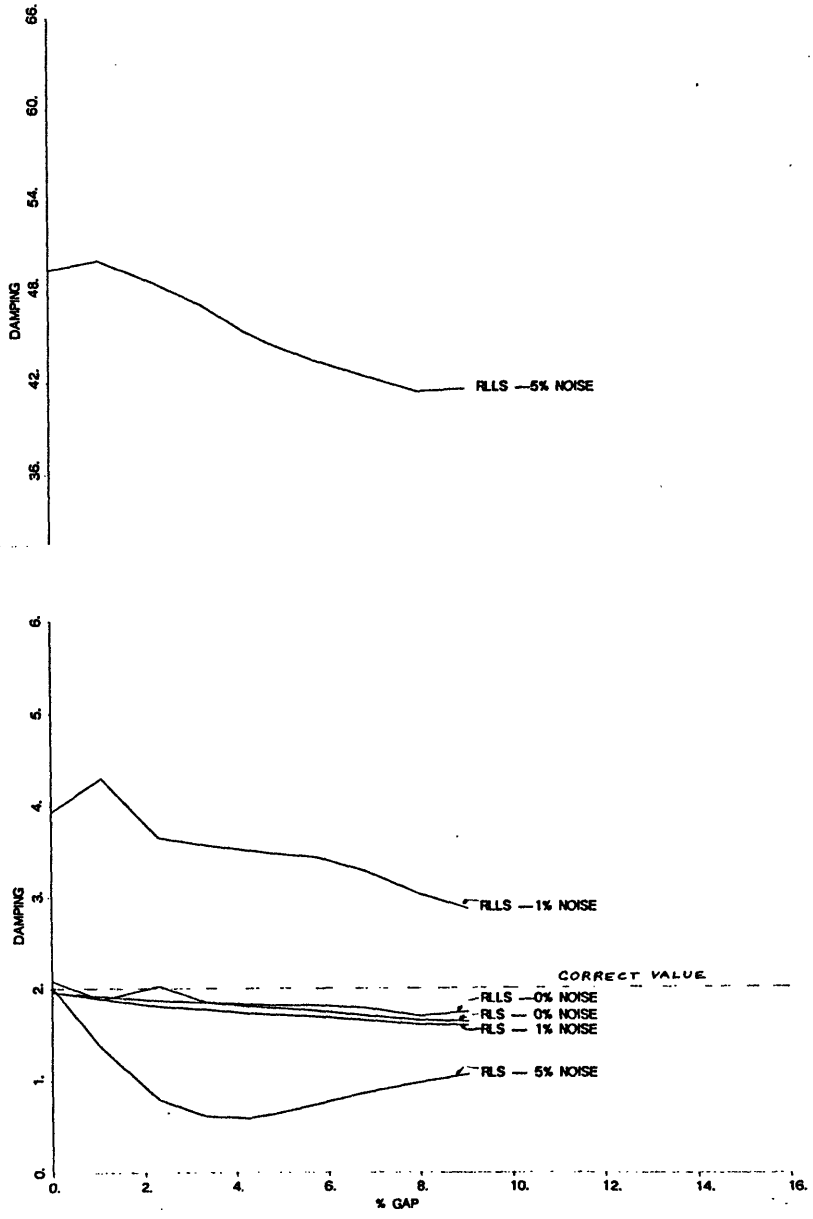
The prior sections summarize the effect of measurement noise and nonlinearity when each is present. The effects on the parameter estimates are, in part, in opposite directions. The next step is to determine the effect on the estimates when both noise and nonlinearity are present. It turns out that when noise is added to the nonlinear data, the curves for the variation of parameter estimates with gap are shifted accordingly, as shown in figure 5.5. Here, only the optimal value of γ for each noise level is used (.00001 for linear, .0001 at 1%, 4.0 at 5%). Low noise levels produce the same linear increase or decrease (higher noise levels have a greater deviation in shape, probably due to incomplete convergence). This shows that the effects due to the gap and the noise can be superimposed. A small amount of noise can completely offset the drop in frequency due to the gap.



a) Frequency Estimates



b) Mass Estimates



c) Damping Ratio Estimates

Figure 5.5: Effect of Noise on the Estimates for a System with a Gap Nonlinearity.

Chapter 6

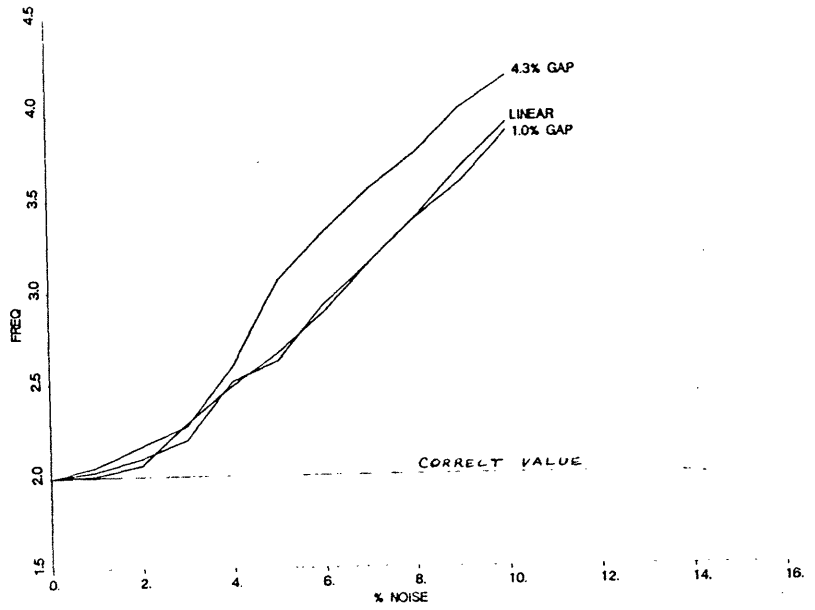
Extension of the RLS Algorithm for an Explicit Nonlinear Model

The RLS algorithm may be extended to include an estimate for the size of the gap itself. This type of extension will work for any linearity that adds extra terms to the equation of motion, such as a cubic or quadratic nonlinearity. The new matrix formulation is

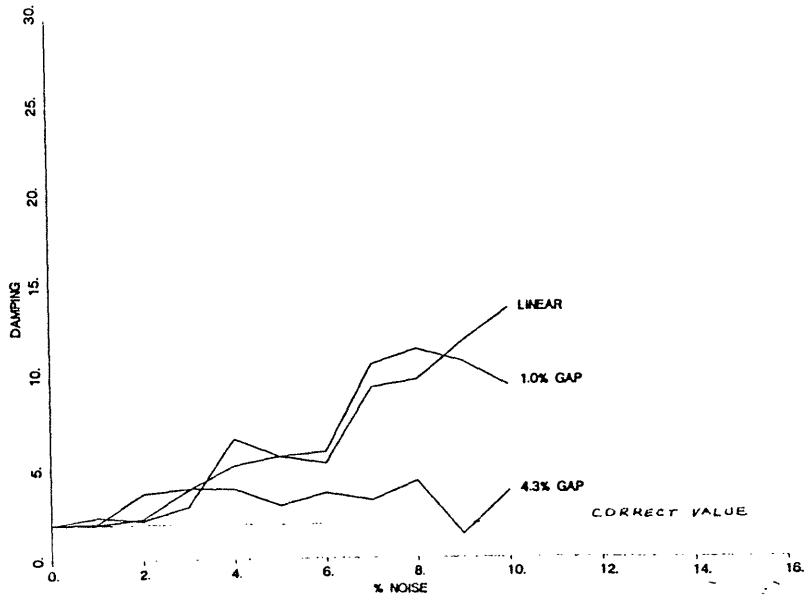
$$\ddot{y} = \begin{bmatrix} 2\dot{y} & y & u & \pm 1 \end{bmatrix} \begin{pmatrix} \varsigma\omega \\ \omega^2 \\ 1/M \\ \delta\omega^2 \end{pmatrix}.$$

This is solved as the RLS algorithm was, through use of the Central Difference Method to compute the derivatives, and extracting the parameter estimates through the pseudoinverse. The actual program used is included in Appendix C. As indicated in the equation above, it ignores the data within the region of the gap, $-\hat{\delta} \leq y \leq \hat{\delta}$.

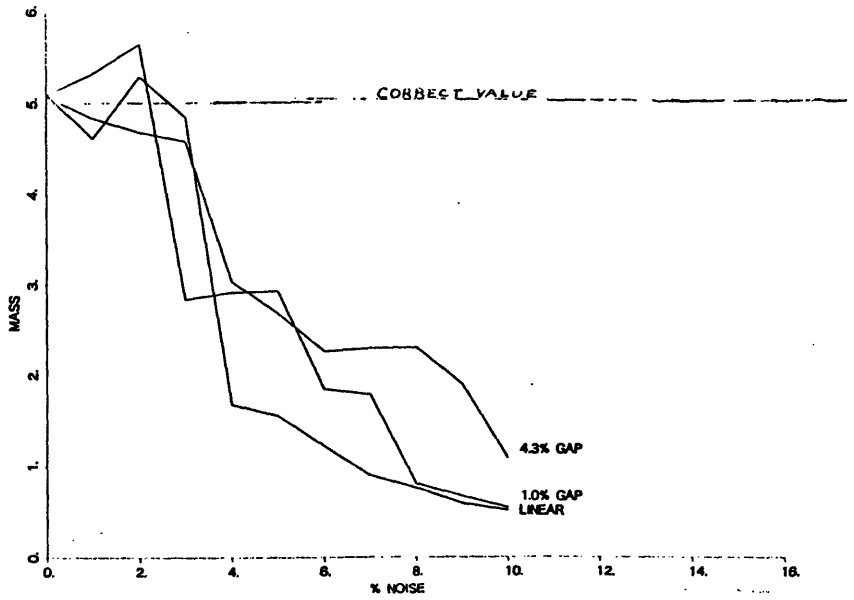
This algorithm produces very good results for the noise-free case, regardless of the size of the gap. In fact, the parameter estimates become slightly more accurate as the gap increases. In the presence of noise, however, (since this algorithm is also based on a Least Squares formulation) a strong bias is introduced. These results are shown in figure 6.1.



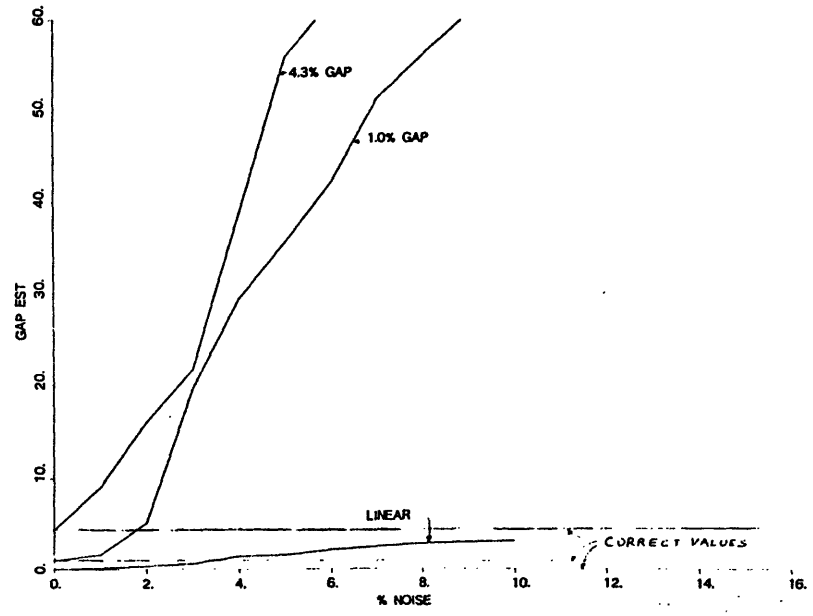
a) ERLS Frequency Estimates



b) ERLS Damping Ratio Estimates



c) ERLS Mass Estimates



d) ERLS Gap Estimates

Figure 6.1: Extended RLS Estimates for Frequency, Damping Ratio, Mass, and Gap for Three Gap Sizes.

Chapter 7

Conclusions and Recommended Future Work

7.1 Conclusions

A gap nonlinearity, such as would be found in an assembled truss structure, has a very definite effect on parameter identification algorithms. Increasing the gap size produces a linear decrease in frequency and damping ratio estimates, and a linear increase in the mass estimate for both the RLS and RLLS algorithms. The gap does not cause the algorithms to fail, because both simply fit the best linear model they can. Additionally, the gap causes faster convergence of the estimates, and makes the RLLS algorithm less sensitive to choice of the noise variance parameter γ.

The addition of noise to the data also produces faster convergence and less sensitivity to γ. Noise however causes very large bias errors in the estimates, and these are in the opposite direction to the change caused by a gap. These large biases are probably not seen in the results for the gap nonlinearity (figure 5.1) because the gap is seen only as a low frequency, constant amplitude disturbance. While the Central Difference Method used in the algorithms differentiates the noise, making it worse, the constant amplitude error from the gap differentiates to zero. Since bias errors for the gap are probably small, the drop in frequency and damping ratio noted with increasing gap should accurately reflect the average system parameters. Use of the method of averaging showed this same type of linear decrease in frequency.

The effect of 5% noise on the three algorithms is shown in Table 7.1. The optimal values of the variance parameter γ are used throughout. The frequency estimates of the ERLS algorithm in the presence of noise are consistently worse than those of the RLS and RLLS algorithms, but the damping and mass estimates are better than those from the RLLS algorithm, and become better than those from the RLS algorithm with higher gap size. The RLS and RLLS results are comparable in the magnitude of the frequency estimate error, but the RLS yields far better estimates for the other parameters. In general, for moderate noise levels, the RLS algorithm is the best choice of these three. For low noise levels and significant gap, the ERLS Algorithm would be preferable.

	LINEAR			1% GAP			4.3% GAP		
	$\Delta\omega$	$\Delta\varsigma$	ΔM	$\Delta\omega$	$\Delta\varsigma$	ΔM	$\Delta\omega$	$\Delta\varsigma$	ΔM
RLS	3.6	2.7	-9.0	3.8	-29.4	23.4	4.9	-67.3	113.2
RLLS	0.3	2695	-87.4	-2.5	2557	-89.2	-5.1	2390	-79.7
ERLS	31.5	185.1	-69.4	33.6	185.4	-42.3	53.2	48.5	-47.0

Table 7.1: **Comparison of the Effect of 5% Noise on the Algorithms. Given as a percent change from the noise-free value of the estimates.**

An additional reason that the RLS algorithm is preferable to the RLLS algorithm is that the RLLS algorithm is too sensitive to the noise variance weighting factor and to finite roundoff errors. Therefore, certain gaps sizes do not mix well with choices of γ, as was found for the choice of $\gamma = .0001$ at the 1% noise level with a 4.3% gap. Numbers around this unstable point yielded worthless answers as the algorithm could not converge. While a faster sampling rate improved the performance of the RLS

algorithm, it only aggravated the problems in the RLLS algorithm.

The noise problem can be dealt with in a number of ways. The easiest way is to use one or more low pass filters, to eliminate the noise before it enters the algorithms. Another common method is to overspecify the number of degrees of freedom in the RLLS model for the system, allowing the algorithm to fit the noise to high frequency modes. In this case, the RLLS algorithm may work better, since the system is easily overspecified and the improvement with each extra degree of freedom may be examined without running the algorithm again. There is an entire class of Least-Squares based algorithms that are designed to handle the bias due to noise by increasing the dimensions of the matrices to include the magnitude of the error due to noise. These are referred to as Extended Least Squares algorithms. Finally, it is probably better to use acceleration data and integrate for the other variables, thereby minimizing the noise effects rather than increasing them.

If a detailed model including the gap is desired, the ERLS algorithm as written in this thesis may be used, or another algorithm may be similarly modified. This gives very accurate results in the absence of noise. There are also a number of algorithms which use different approaches to get a value for the nonlinearity directly. The Kalman filter may be extended to include a nonlinearity. In the Bayesian form [9], the parameter is thought of as a random variable, and a solution is given as a probability density function for the states. Billings [1] expands the nonlinearity into sines and cosines and obtains a linear model. Others, such as the Threshold Adaptive Modelling Technique [5] or Volterra and Wiener Series use a complicated kernal formulation and are quite recent. They seem to show promise when used with complex nonlinearities like a gap.

Short of using these more complicated techniques, there are several ways that the

effect of the gap on the identification process can be minimized. It is important to use a continuous forcing function, since in free decay the size of the gap relative to the average displacement amplitude is always growing, leading to constantly greater distortion of the results. The frequency content should excite all the desired modes, and the amplitude of the displacement should be relatively constant. While it would be nice to force the displacement amplitude to be as large as possible (since this minimizes the numerical problem and the nondimensional gap and noise sizes), it is preferable that the displacement amplitude matches that which the system is expected to encounter during use. In this case, the linear nature of the frequency drop may be exploited in that only two or three amplitudes need to actually be tested, and a linear plot of frequency versus displacement amplitude can be drawn. This would allow a switch between models at different amplitudes for best results. The forcing needs to continue for a long enough time (here 42 periods was chosen, and this is probably a minimum) for the algorithm to converge.

It may also be noted that the higher the damping ratio, the more accurate the damping ratio estimate. This was found in a separate project, not discussed here. The effects of a gap can be eliminated entirely by loading the structure in compression (or tension), and never allowing the displacement to cross the region of the gap. This is sometimes done in practice.

With precautions, parameter identification of a system with a gap nonlinearity can be conducted, and very good results obtained.

7.2 Recommended future work

This study of the effect of a gap nonlinearity on parameter identification could be followed up in a number of ways.

1. Higher Degree of Freedom Model. A model consisting of two or more degrees of freedom, each containing a gap nonlinearity could be studied. This work could be related to the modal synthesis of the Blackwood experiment. It could be determined whether it is better to include the nonlinearity as a disturbance force $\mathcal{F} = k\delta$ instead of a change in displacement. Additionally, a better sense of how the gaps interact over the entire structure would be obtained.

2. Reliability With Other Nonlinearities. A model containing hysteresis, as well as a gap, or a model with a cubic nonlinearity included could be tested. This should give a better indication of how the algorithms would work in actual use.

3. Use on Real Structure. The algorithms need to be tested on a real nonlinear structure, to determine the accuracy of the estimates compared to those obtained by other test methods.

4. Study of Noise. Some of the ways to minimize the effect of noise on the algorithms could be studied, including the addition of noise modes or the use of an Extended Least Squares formulation of the algorithms. Further study on the effect of different types of noise could be done analytically, as Longman has done for the ERA algorithm.[10] The interaction of a Minimum Model Error (MME) state estimator [7] with noisy gap data would be another interesting line of work.

5. Extension of Studied Algorithms. The RLLS algorithm could be extended to include an estimate for the gap, as was done in this thesis for the RLS algorithm.

The Extended Least Squares form mentioned above could be used.

6. Study of Additional Algorithms. Study of the effectiveness on a gap nonlinear system of the ERA, or a nonlinear identifying algorithm could be undertaken.

Bibliography

[1] Billings, S. 'Identification of Nonlinear Systems – A Survey.' **Proceedings of IEEE**, vol. #127, part D (Nov 1980) p.272.

[2] Blackwood, G.H. and A.H. von Flotow,'Guidelines for Component Mode Synthesis of Structures with Sloppy Joints.' **29th AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics and Materials Conference**, #3 (April, 1988) p.1565.

[3] Brown, R.G., **Introduction to Random Signal Analysis and Kalman Filtering**. John Wiley & Sons, 1983.

[4] Fullekrug, U. 'Survey of Parameter Estimation Methods in Experimental Modal Analysis.' **5th International Modal Analysis Conference**, #1 (April 1987) p.460.

[5] Hsieh, S. and H. Lai, 'Threshold Adaptive Modeling for Drilling Dynamics Characterization and Process Monitoring.' **5th International Modal Analysis Conference**,#2 (April 1987) p.1586.

[6] Juang, J. and R. Pappa, 'An Eigensystem Realization Algorithm (ERA) for Modal Parameter Identification and Model Reduction).' **AIAA Journal of Guidance, Control, and Dynamics** vol.#8 (Sept-Oct 1985) p. 620.

[7] Mook, D. and J. Lew, 'A Combined ERA/MME Algorithm for Robust System Realization/Identification.' **29th AIAA/ ASME/ ASCE/ AHS/ ASC Structures, Structural Dynamics and Materials Conference**, #3 (April, 1988) p.1556.

[7] Lee, D., M. Morf, and B. Friedlander, 'RLS Ladder Estimation Algorithms.' **IEEE Transactions** ASSP 29, #3 (1981) p.627.

[8] Lee, D., M. Morf, and B. Friedlander, 'Recursive Ladder Algorithms in ARMA Modelling.' **IEEE Transactions** AC-27, #4 (1982) p.753.

[9] Ljung, L. and T. Soderstrom, **Theory and Practice of Recursive Identification.** The MIT Press, 1983.

[10] Longman, R.W., and J.Juang, 'A Variance Based Confidence Criterion for ERA Identified Modal Parameters.' Paper # AAS 87-454, AAS/AIAA Astrodynamics Specialist Conference, August, 1987.

[11] Soucy, Y. and F. Vigneron, 'Identification of Structural Properties of a Continuous Longeron Space Mast.' AIAA 84-0930, 1984.

[12] Voss, J. E., 'Real Time Identification of Large Space Structures.' MIT Ph.D. Thesis, February, 1987.

Appendix A

Derivation of the Voss RLLS Algorithm

A RLLS algorithm is based on the interaction of two linear difference equations in ARMAX (Auto-Regressive, Moving Average, Exogeneous inputs) form. The first is called the forward difference equation:

$$y(t) = a_1 y(t-1) + b_1 u(t-1) + \cdots + a_n y(t-n) + b_n u(t-n) + b_0 u(t) + e_n(t). \quad \text{(A.1)}$$

The terms $y(t)$ and $u(t)$ indicate outputs and inputs at time t respectively, so that the current output depends on a linear combination of the previous inputs and outputs. The variable n indicates the finite number of previous time steps, and is known as the order of regression. The term $e_n(t)$ is the error between the linear approximation of order n and the actual $y(t)$. This form of the equation is also known as a linear regression.

The second equation is known as the backward difference equation:

$$\begin{aligned} y(t-n) &= a_0^* y(t) + b_0^* u(t) + \cdots + a_{n-1}^* y(t-n+1) + b_{n-1}^* u(t-n+1) \\ &\quad + b_n^* u(t-n) + r_n(t). \end{aligned}$$

The equations are shown graphically in figure A.1 below. Since the format of each equation is similar, only the forward difference equation is treated for now.

The ARMAX equation may be written in a matrix form:

$$y(t) = \theta^T \phi + e_n(t) \quad \text{(A.2)}$$

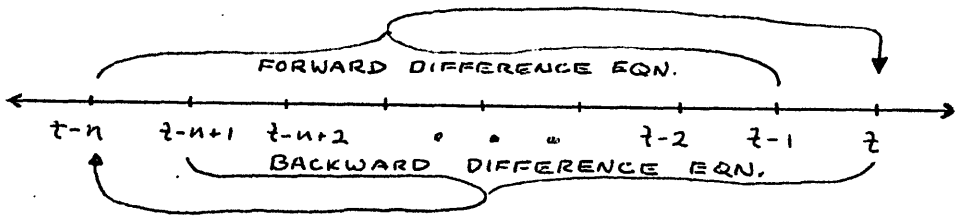


Figure A.1: A Graphical Interpretation of the RLLS Difference Equations.

$$y(t) = \begin{bmatrix} a_1 & b_1 & \cdots & a_n & b_n & b_0 \end{bmatrix} \begin{pmatrix} y(t-1) \\ u(t-1) \\ \vdots \\ y(t-n) \\ u(t-n) \\ u(t) \end{pmatrix} + e_n(t)$$

where θ is then the vector of unknown coefficients a and b and ϕ is a vector of the previous outputs and inputs. One way to solve this type of equation is through use of a Kalman filter, which computes the unknown θ based on the following state space representation:

$$\begin{aligned} \theta(t+1) &= \theta(t) \\ y(t) &= \varphi^T(t)\,\theta(t) + e(t). \end{aligned}$$

This shows explicitly that $\theta(t)$ is made up of values which are constant for all time t. Note that $\theta^T(t)\,\varphi(t) = \varphi^T(t)\,\theta(t)$ since both variables are vectors.

A more general state space equation is written as

$$\begin{aligned} x(t+1) &= F\,x(t) + G\,u(t) + w(t) \\ y(t) &= H\,x(t) + I\,u(t) + e(t) \end{aligned}$$

Here, $w(t)$ indicates changes in the states, and $e(t)$ indicates errors in the output.

This general state-space equation yields three iterative equations to solve for $\hat{x}(t) = E(\,x(t)|y(0),u(0),\ldots,y(t-n),u(t-n)\,)$ (an equation indicating that $\hat{x}(t)$ is an estimation based on the previous data),

$$\begin{aligned}
\hat{x}(t+1) &= F(t)\hat{x}(t) + \underline{k}(t)\,[y(t) - H(t)\hat{x}(t)] \\
\underline{k}(t) &= \frac{F(t)P(t)H^T(t)}{H(t)P(t)H^T(t) + r_2(t)} \\
P(t+1) &= F(t)P(t)F^T(t) + r_1(t) - \underline{k}(t)\,H(t)P(t)F^T(t).
\end{aligned}$$

This is known as a Kalman filter, $\underline{k}(t)$ is known as the Kalman gain. The parameters r_1 and r_2 come from the 2x2 matrix of variances, R, where $r_1 = E(w(t)w^T(t))$, $r_2 = E(e(t)e^T(t))$ and $r_{12} = E(w(t)e^T(t))$. Since the errors are independent, $r_{12} = 0$. $P(t)$ is then the diagonal covariance matrix (the inverse of R). The derivation of these equations may be found in *Ljung* [9].

For our specialized case, $F = 1$, $H = \varphi^T(t)$, $G = I = 0$, $x(t) = \theta(t)$, and the first error $r_1(t) = 0$. This yields the following three equations:

$$\begin{aligned}
\hat{\theta}(t+1) &= \hat{\theta}(t) + \underline{k}(t)\,[y(t) - \varphi^T(t)\hat{\theta}(t)] \\
\underline{k}(t) &= \frac{P(t)\varphi(t)}{\varphi^T(t)P(t)\varphi(t) + r_2(t)} \\
P(t+1) &= P(t) - \underline{k}(t)\,\varphi^T(t)P(t)
\end{aligned}$$

This formulation is equivalent to the least squares solution if the system is overdetermined, the state is assumed constant, and no prior knowledge of the state is assumed. Although this is surprising, given that the least squares solution is deterministic, and the Kalman filter is a stochastic system, this is proved in Appendix B. The Kalman filter has the advantage that the initial state can be assigned a value, perhaps from a model. This speeds convergence.

The advantage of the RLLS formulation is that the effect of changing the regression order n on the estimate is made explicit. This allows the number of degrees of freedom

in the model to be overspecified, and the increase in order examined to see when no further improvement occurs in the parameter estimates. Additionally, the estimates for all orders less than n may be extracted, as opposed to the normal Kalman filter, which would require recomputing all the coefficients. The resulting form has the shape of a ladder (or lattice) as was shown in Chapter 3.

In her thesis, Voss makes maximum use of the Lattice properties and of the power of the Kalman filter by using an individual Kalman filter for both the forward and backward equations. She also modifies the notation used, in order to improve the algorithm's performance. She expands the basic equation (A.1) to include the current input as well as output, and defines a new variable $\underline{x}(t)$, where $\underline{x}(t) = [\ \underline{y}^T(t)\ \underline{u}^T(t)\ ]^T$. This requires the assumption that $y(t)$ is not affected by $u(t)$, which simply means that there are no instantaneous effects on the system displacement. This yields the matrix equation

$$\underline{x}(t) = \begin{pmatrix} y(t) \\ u(t) \end{pmatrix}$$

$$= \begin{bmatrix} a_1 & b_1 & \cdots & a_n & b_n \\ c_1 & d_1 & \cdots & c_n & d_n \end{bmatrix} \begin{pmatrix} y(t-1) \\ u(t-1) \\ \vdots \\ y(t-n) \\ u(t-n) \end{pmatrix} + \begin{pmatrix} e_{n1}(t) \\ e_{n2}(t) \end{pmatrix}$$

There is a similar equation for the backward difference equation. This arrangement is made so that inputs and outputs may be paired at each time step. No attempt is really made to estimate the force $u(t)$, so the values of c and d are essentially meaningless. Note that although the algorithm is usable for multiple inputs and outputs, I will only be using it for the single-input, single-output case, so that $\underline{x}(t) = [\ y(t)\ u(t)\ ]^T$.

The two RLLS equations then become

$$\underline{x}^{(n)}(t) = \hat{G}^{(n)}(t)\,\underline{\phi}^{(n)}(t-1) + \underline{e}_n(t) \tag{A.3}$$

for the forward direction, and

$$\underline{x}^{(n)}(t-n) = \hat{H}^{(n)}(t)\,\underline{\phi}^{(n)}(t) + \underline{r}_n(t) \tag{A.4}$$

for the backward direction. The hat ($\hat{}$) indicates the variable being estimated. Note that G and H here are not the same as the coefficients in the general state space equation. Here the first row of $\hat{G}$ is $\hat{\theta}$. The composition of $\hat{H}$ is similar in size and meaning. The notation $\underline{x}^{(n)}(t)$ merely indicates that the recursion order is n, and has no effect on the value of $\underline{x}(t)$.

The vectors $\underline{\phi}^{(n)}(t-1)$ and $\underline{\phi}^{(n)}(t)$ are defined to be

$$\underline{\phi}^{(n)}(t-1) = \begin{pmatrix} \underline{x}(t-1) \\ \underline{x}(t-2) \\ \vdots \\ \underline{x}(t-n) \end{pmatrix} \quad \underline{\phi}^{(n)}(t) = \begin{pmatrix} \underline{x}(t) \\ \underline{x}(t-1) \\ \vdots \\ \underline{x}(t-n+1) \end{pmatrix}. \tag{A.5}$$

The superscript n continues to indicate the total time steps needed.

The equations A.3 and A.4 then undergo a change of basis, so that the errors $\underline{e}_n(t)$ and $\underline{r}_n(t)$ become the regressing terms, rather than $\underline{x}(t)$. Since

$$\underline{e}_n(t) = \underline{x}^{(n)}(t) - \hat{G}^{(n)}(t)\,\underline{\phi}^{(n)}(t-1) \tag{A.6}$$

$$\underline{r}_n(t) = \underline{x}^{(n)}(t-n) - \hat{H}^{(n)}(t)\,\underline{\phi}^{(n)}(t) \tag{A.7}$$

so that

$$\underline{r}_n(t-1) = \underline{x}^{(n)}(t-n-1) - \hat{H}^{(n)}(t-1)\,\underline{\phi}^{(n)}(t-1) \tag{A.8}$$

If $\hat{G}^{(0)}(t)$ and $\hat{H}^{(0)}(t)$ are set equal to a zero 2x2 matrix for all time t, then

$$\underline{e}_0(t) = \underline{x}^{(0)}(t) \quad \underline{r}_0(t) = \underline{x}^{(0)}(t)$$

which is simply the data taken at time t.

The equation A.8 may then be written in matrix form, which shows the change of basis:

$$\begin{pmatrix} \underline{r}_0(t-1) \\ \underline{r}_1(t-1) \\ \vdots \\ \underline{r}_{n-1}(t-1) \end{pmatrix} = \begin{bmatrix} I_{2x2} & 0 & \cdots & 0 \\ -\hat{H}^{(1)}(t-1) & I_{2x2} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \leftarrow -\hat{H}^{(n-1)}(t-1) \rightarrow & & & I_{2x2} \end{bmatrix} \begin{pmatrix} \underline{x}(t-1) \\ \underline{x}(t-2) \\ \vdots \\ \underline{x}(t-n) \end{pmatrix}$$

$$= T\underline{\phi}^{(n)}(t-1)$$

which makes use of the definition of $\underline{\phi}^{(n)}(t-1)$ in equation A.5, and also equation A.7.

From this, choose a matrix K partitioned in the form

$$K = \begin{bmatrix} K_0 & K_1 & \ldots & K_{n-1} \end{bmatrix}$$

where the K_i are the reflection coefficient matrices, which are square matrices of dimension equal to number of rows in $\underline{x}$ (2 for a SISO system). Define K to be

$$K = \hat{G}\,T^{-1}$$

so that in the new regressors (from equation A.3),

$$\begin{aligned} \underline{x}^{(n)}(t) &= (\hat{G}^{(n)}(t)\,T^{-1})(T\,\phi^{(n)}(t-1)) + \underline{e}_n(t) \\ &= \begin{bmatrix} K_0 & K_1 & \ldots & K_{n-1} \end{bmatrix} \begin{pmatrix} \underline{r}_0(t-1) \\ \underline{r}_1(t-1) \\ \vdots \\ \underline{r}_{n-1}(t-1) \end{pmatrix} + \underline{e}_n(t) \\ &= \sum K_{n-1}(t)\,\underline{r}_{n-1}(t-1) + \underline{e}_n(t). \end{aligned}$$

This equation is easier to work with, since only one regressor, $\underline{r}_n(t-1)$, is needed at a time, making it much more suitable for an algorithm. The covariance matrix P

undergoes a similar change of basis so that $P = T^T(t)\,P(t)\,T(t)$. The new equation in K is solved by a Kalman filter, and becomes (after a shift in time)

$$\begin{aligned} \underline{k} &= \frac{P_n(t-2)r_n(t-1)}{r_n^T(t-1)P_n(t-2)r_n(t-1)+\gamma_n(t)} \\ K_n(t) &= K_n(t-1)+[e_n(t)-K_n(t-1)r_n(t-1)]\,\underline{k}^T \\ P_n(t-1) &= \frac{1}{\lambda}[I-\underline{k}\,r_n^T(t-1)]\,P_n(t-2) \end{aligned}$$

If the parameter identification is working, the values of $\underline{e}_n(t)$ and $\underline{r}_n(t)$ will be minimized as $t \rightarrow \infty$ and these minimum values should decrease with the order of regression n.

A similar Kalman filter is provided for the backwards difference equation, using $\underline{e}(t)$ as the recursor, with all other variables starred. The values for the recursors as n is increased are found by several updating equations which relate the regressors from both sets of equations,$\underline{e}(t)$ and $\underline{r}(t)$. Since

$$\begin{aligned} \underline{x}(t) &= \sum K_{n-1}(t)\underline{r}_{n-1}(t-1)+\underline{e}_n(t) \\ &= \sum K_n(t)\underline{r}_n(t-1)+\underline{e}_{n+1}(t) \\ \underline{e}_{n+1}(t) &= \underline{e}_n(t)-K_n(t)\underline{r}_n(t-1). \end{aligned}$$

By similar derivation from the backward Kalman filter,

$$\underline{r}_{n+1}(t) = \underline{r}_n(t-1) - K_n^*(t)\underline{e}_n(t).$$

These last two equations clearly show the exchange of information between filters which give the lattice form shown in figure 3.2.

There is also an updating equation for the parameter γ, which is used to account for measurement noise. The initial value is $\gamma_0(t) = \lambda V$, the forgetting factor times the assumed variance in the measurements. The updating equation is derived in reference [9] from the definition of the least squares weighting factor $\beta_n(t)$, where $\gamma_n(t) = \lambda\beta_n(t)$,

$$\gamma_{n+1}(t) = \gamma_n(t) - \lambda r_n^T(t-1)P_n(t-1)r_n(t-1).$$

Finally, the algorithm calculates the unknown matrix $\hat{G}$ at each time step and for each value of n as follows. Given

$$\begin{aligned}\hat{x}^{(n+1)}(t) &= \sum \hat{K}_n(t)\, r_n(t-1) \\ &= \hat{G}^{(n)}(t)\, \phi^{(n)}(t-1) + \hat{K}_n(t)\, r_n(t-1)\end{aligned}$$

and that

$$\hat{x}^{(n+1)}(t) = \hat{G}^{(n+1)}(t)\, \phi^{(n+1)}(t-1)$$

then

$$\hat{G}^{(n+1)}(t)\, \phi^{(n+1)}(t-1) = \hat{G}^{(n)}(t)\, \phi^{(n)}(t-1) + \hat{K}_n(t) r_n(t-1).$$

From equation A.8,

$$\underline{r}_n(t-1) = \underline{x}^{(n)}(t-n-1) - \hat{H}^{(n)}(t-1)\, \underline{\phi}^{(n)}(t-1)$$

so this can be factored as

$$\hat{G}^{(n+1)}(t) \begin{pmatrix} \phi^{(n)}(t-1) \\ x(t-(n+1)) \end{pmatrix} =$$
$$\hat{G}^{(n)}(t)\, \phi^{(n)}(t-1) + \hat{K}_n(t)\, [\, -\hat{H}^{(n)}(t-1)\phi^{(n)}(t-1) + x(t-n-1)]$$

which yields

$$G^{(n+1)}(t) = [\ G^{(n)}(t) \quad 0\ ] + K_n(t)\, [\ -H^{(n)}(t-1) \quad I\ ].$$

This simple updating equation requires much less work than calculating $\hat{G} = K\,T$. The updating equation for the H matrix is similarly derived to be

$$H^{(n+1)}(t) = [\ 0 \quad H^{(n)}(t-1)\ ] + K_n^*(t)\, [\ I \quad -G^{(n)}(t)\ ].$$

The entire algorithm is included as an appendix. Derivation of the second Kalman filter for recursion in $\underline{e}_n(t)$ and K^* may be found in Appendix A of [12].

An RLLS algorithm then actually consists of two loops. The inner loop estimates the unknown parameters as n (the order of the recursion) increases. The outer loop reads the next data point as time increases and updates in time the two linear difference equations. The complete algorithm in algebraic form is shown in figure A.2.

Initialization: $K_n(0) = K_n^*(0) = 0$

$P_n(-1) = P_n^*(0) = \text{large } I \quad (\sim 100{,}000\, I)$ $\quad n=0 \to M-1$

$\gamma_o(t) = \lambda V$, where λ = forgetting factor

V = measurement noise variance

Each new measurement:

$$\underline{e}_o(t) = \underline{r}_o(t) = \left[\underline{y}^T(t)\ \underline{u}^T(t)\right]^T$$

For $n=0 \to M-1$:

$$\underline{k} = \frac{P_n(t-2)\ \underline{r}_n(t-1)}{\underline{r}_n^T(t-1)\ P_n(t-2)\ \underline{r}_n(t-1) + \gamma_n(t)}$$

$$K_n(t) = K_n(t-1) + \left[\underline{e}_n(t) - K_n(t-1)\ \underline{r}_n(t-1)\right]\underline{k}^T$$

$$P_n(t-1) = \frac{1}{\lambda}\left\{I - \underline{k}\ \underline{r}_n^T(t-1)\right\}P_n(t-2)$$

$$\underline{k}^* = \frac{P_n^*(t-1)\ \underline{e}_n(t)}{\underline{e}_n^T(t)\ P_n^*(t-1)\ \underline{e}_n(t) + \gamma_n(t)}$$

$$K_n^*(t) = K_n^*(t-1) + \left[\underline{r}_n(t-1) - K_n^*(t-1)\ \underline{e}_n(t)\right]\underline{k}^{*T}$$

$$P_n^*(t) = \frac{1}{\lambda}\left[I - \underline{k}^*\ \underline{e}_n^T(t)\right]\ P_n^*(t-1)$$

For $n \neq M-1$:

$$\underline{e}_{n+1}(t) = \underline{e}_n(t) - K_n(t)\ \underline{r}_n(t-1)$$

$$\underline{r}_{n+1}(t) = \underline{r}_n(t-1) - K_n^*(t)\ \underline{e}_n(t)$$

$$\gamma_{n+1}(t) = \gamma_n(t) - \lambda\ \underline{r}_n^T(t-1)\ P_n(t-1)\ \underline{r}_n(t-1)$$

To recover RLS coefficients ($\underline{x}(t) = G(t)\ \underline{\phi}(t-1)$):

$$G^{(n+1)}(t) = \left[G^{(n)}(t)\quad 0\right] + K_n(t)\left[-H^{(n)}(t-1)\quad I\right]$$

$$H^{(n+1)}(t) = \left[0\quad H^{(n)}(t-1)\right] + K_n^*(t)\left[I\quad -G^{(n)}(t)\right]$$

with: $G^{(1)}(t) = K_o(t)$ and $H^{(1)}(t) = K_o^*(t)$

Figure A.2: The Voss Formulation of the RLLS Algorithm.

Appendix B

Relationship Between the Pseudoinverse and Kalman Filter

This analysis is based on a discussion in reference [3].

The basic parameter identification equation is

$$y(t) = \varphi_t^T \Theta(t)$$

so that φ_t^T and $y(t)$ are the parameter coefficient vector and output, respectively. Given m time steps, the Kalman equations look as follows:

$$\begin{aligned} \hat{\Theta}(t+1) &= \hat{\Theta}(t) + \underline{k}(t)\,[Y(t) - \Phi(t)\hat{\Theta}(t)] \\ \underline{k}(t) &= \frac{P(t)\Phi^T(t)}{\Phi(t)P(t)\Phi^T(t) + R_2(t)} \\ P(t+1) &= P(t) + R_1(t) - \underline{k}(t)\,\Phi^T(t)P(t) \end{aligned}$$

Where

$$\Phi = \begin{pmatrix} \varphi_1^T \\ \varphi_2^T \\ \vdots \\ \varphi_m^T \end{pmatrix} \qquad Y = \begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(m) \end{pmatrix}$$

This system has the initial conditions $\hat{\Theta}(0) = 0$ if there are no coefficients determined initially. Since the variance of the data at $t = 0$ is zero, the values of the diagonal variance matrix are zero (i.e. $R_1(0) = 0$ and $R_2(0) = 0$). At time $t = 1$, there will be a

large variance in the data as the first data point is taken, so that $R = \rho \times I$ where ρ is a large number. Since the covariance matrix is P, where $P = R^{-1}$, $P(1) = 0$.

Starting at $t = 0$ yields

$$\begin{aligned} \hat{\Theta}(1) &= \underline{k}(0)[Y(0)] \\ \underline{k}(0) &= \frac{P(0)\Phi^T(0)}{\Phi(0)P(0)\Phi^T(0)} \\ 0 &= P(0) - \underline{k}(0)\Phi(0)P(0) \end{aligned}$$

If we set $P(0) = P_0$, $\Phi(0) = \Phi_0$ and $Y(0) = Y_0$,

$$\begin{aligned} P_0 &= P_0\Phi_0^T(\Phi_0 P_0 \Phi_0^T)^{-1}\Phi_0 P_0 \\ \Phi_0 P_0 &= \Phi_0 P_0 \Phi_0^T(\Phi_0 P_0 \Phi_0^T)^{-1}\Phi_0 P_0 \end{aligned}$$

Which is true for any value of P_0, so

$$\hat{\Theta}(1) = P_0\Phi_0^T(\Phi_0 P_0 \Phi_0^T)^{-1}Y_0$$

Which is equivalent to

$$\hat{\Theta}(1) = (\Phi_0^T P_0 \Phi_0)^{-1}\Phi_0^T P_0 Y_0$$

which is the formulation of the problem in a least squares solution with P_0 acting as the weighting matrix. For this reason, Ljung states that this matrix is the natural choice for a weighting factor. [9]

It is interesting that these two very different approaches, one deterministic and the other stochastic, each act to minimize the errors in an analogous way.

Appendix C

The Fortran Programs

C.1 The Nonlinear System Model

```
C     PROGRAM "SYSTEM"
C
      PROGRAM MSPR
      COMMON J,T,X1,X2,F2,DT,ROMEGA,RPSI,DISP
      COMMON RMASS,DELTA,BU,BL,CF
      OPEN(unit=14,file=file,status='old')
      OPEN(unit=15,file=file,status='old')
      OPEN(unit=16,file=file,status='new')
      OPEN(unit=17,file=file,status='new')
      OPEN(unit=50,file=file,status='old')
C       INITIALIZATION OF PARAMETERS
      romega=2.0
      rpsi=.02
      rmass=5.0
      READ(unit=50,fmt=*) delta
      disp=-2.0
      READ(unit=14,fmt=*) dt
      cf=2*rpsi/(romega*dt)
      READ(unit=15,fmt=*) dummy,f2
      coeff=.5*dt**2
      x1=disp+coeff*(f2/rmass-romega**2*(disp+delta))
      t=0
      j=0
```

```
      x2=disp
C
C CALCULATING COEFFICIENTS
      SOT=rpsi*romega*dt
      a=(2-(romega*dt)**2)/(1+SOT)
      b=(SOT-1)/(1+SOT)
      c=dt**2/rmass/(1+SOT)
      d=(delta*(romega*dt)**2)/(1+SOT)
C
C BOUNDARY CONDITIONS
      bu=delta
      bl=-delta
      vel=x2-x1
      IF(vel.GT.0) GOTO 60
C
C   ON RIGHT SIDE OF GAP
   20 IF(x2.LT.bu) GOTO 40
      disp=a*x2+b*x1+c*f2+d
      CALL DATA
      GOTO 20
C
C INSIDE GAP,MOVING LEFT
   40 tgap=0
      IF(rpsi.EQ.0) GOTO 42
   41 z=x2+rpsi*romega*(x2-x1)*exp(-romega/2/rpsi*tgap)
      GOTO 43
   42 z=0
   43 IF(z.LE.-delta) GOTO 60
      disp=2*x2-x1+c*f2
      CALL DATA
      tgap=tgap+dt
      GOTO 41
C
C ON LEFT SIDE OF GAP
   60 IF(x2.GT.bl) GOTO 80
      disp=a*x2+b*x1+c*f2-d
      CALL DATA
      GOTO 60
```

```
C
C INSIDE GAP,MOVING RIGHT
   80 tgap=0
      IF(rpsi.EQ.0) GOTO 82
   81 z=x2+rpsi*romega*(x2-x1)*exp(-romega/2/rpsi*tgap)
      GOTO 83
   82 z=0
   83 IF(z.GE.delta) goto 20
      disp=2*x2-x1+c*f2
      CALL DATA
      tgap=tgap+dt
      GOTO 81
  100 CONTINUE
      END
C
      SUBROUTINE DATA
      COMMON J,T,X1,X2,F2,DT,ROMEGA,RPSI,DISP
      COMMON RMASS,DELTA,BU,BL,CF
      j=j+1
      t=t+dt
C PRINTING ALTERNATING VALUES
      IF(j.NE.2) GOTO 400
      vel=x2-x1
C     WRITE(unit=16,fmt=*) t,X2
      WRITE(unit=17,fmt=*) f2,X2
      j=0
      IF(t.GT.131.) STOP
C GETTING NEXT DATA POINT
  400 x1=x2
      x2=disp
      READ(unit=15,fmt=*) dummy, f2
      bu=delta-cf*(x2-x1)
      bl=-delta-cf*(x2-x1)
      RETURN
  500 CONTINUE
      END
```

C.2 The RLS Algorithm

```
C PROGRAM "RLS"
      DIMENSION U(3), Y(3), A(6),HINV(6)
      DOUBLEPRECISION X1,X2,X3,DEN
      OPEN(UNIT=17,STATUS='OLD')
      OPEN(UNIT=18,STATUS='NEW')
      OPEN(UNIT=19,STATUS='NEW')
      OPEN(UNIT=20,STATUS='NEW')
      OPEN(UNIT=52,STATUS='OLD')
      K=0
      READ(UNIT=52,FMT=*) GAMMA,DT,NDATA
      SKIP=0
      A(1)=0
      A(2)=0
      A(3)=0
      A(4)=0
      A(5)=0
      A(6)=0
      FZ=0
      GZ=0
      HZ=0
C
      READ(UNIT=17,FMT=*) U(1),Y(1)
      READ(UNIT=17,FMT=*) U(2),Y(2)
      DO 200 N=1,NDATA
      READ(UNIT=17,FMT=*) U(3),Y(3)
C
      Z=(Y(3)-2*Y(2)+Y(1))/DT**2
      F=(Y(1)-Y(3))/DT/1000
      G=-Y(2)
      H=U(2)
      A(1)=A(1)+F**2
      A(2)=A(2)+F*G
      A(3)=A(3)+F*H
      A(4)=A(4)+G**2
      A(5)=A(5)+G*H
      A(6)=A(6)+H**2
```

```
      FZ=FZ+F*Z
      HZ=HZ+H*Z
      GZ=GZ+G*Z
C
      IF(K.LT.10) GOTO 190
      DEN=A(1)*(A(4)*A(6)-A(5)**2)+A(2)*(A(3)*A(5)-A(2)*A(6))
      DEN=DEN+A(3)*(A(2)*A(5)-A(3)*A(4))
      IF(DEN.EQ.0.) GOTO 190
      HINV(1)=(A(4)*A(6)-A(5)**2)/DEN
      HINV(2)=(A(3)*A(5)-A(2)*A(6))/DEN
      HINV(3)=(A(2)*A(5)-A(3)*A(4))/DEN
      HINV(4)=(A(1)*A(6)-A(3)**2)/DEN
      HINV(5)=(A(2)*A(3)-A(1)*A(5))/DEN
      HINV(6)=(A(1)*A(4)-A(2)**2)/DEN
      X1=HINV(1)*FZ+HINV(2)*GZ+HINV(3)*HZ
      X2=HINV(2)*FZ+HINV(4)*GZ+HINV(5)*HZ
      X3=HINV(3)*FZ+HINV(5)*GZ+HINV(6)*HZ
C
      SKIP=SKIP+1
      IF(SKIP.LT.5) GOTO 190
      T=DT*N
      ROMEGA=ABS(X2)**.5
      RPSI=X1/ROMEGA/1000
      IF(X3.EQ.0.) GOTO 190
      RMASS=1/X3
      WRITE(UNIT=18,FMT=*) T,ROMEGA
      SI=RPSI*100
      WRITE(UNIT=19,FMT=*) T,SI
      WRITE(UNIT=20,FMT=*) T,RMASS
      SKIP=0
C
  190 U(1)=U(2)
      U(2)=U(3)
      Y(1)=Y(2)
      Y(2)=Y(3)
      K=K+1
  200 CONTINUE
      END
```

C.3 The RLLS Algorithm

```
C PROGRAM "RLLS"
C
C INITIALIZATION
      DOUBLEPRECISION P(2,2,2),PSTAR(2,2,2),S1(2,2)
      DOUBLEPRECISION GMMA(2),A,B,C,AS,BS,CS
      DIMENSION HOLD(2,2),R(2,2),E(2,2)
      DIMENSION G1(2,2),H1(2,2)
      DIMENSION G2(2,2),H2(2,2)
      REAL K(2,2,2),KSTAR(2,2,2)
      REAL KBAR(2),KAST(2)
      DOUBLEPRECISION DEN
      OPEN(UNIT=29,STATUS='OLD')
      OPEN(UNIT=25,STATUS='OLD')
      OPEN(UNIT=26,STATUS='NEW')
      OPEN(UNIT=27,STATUS='NEW')
      OPEN(UNIT=52,STATUS='OLD')
C
      DO 30 I2=1,2
      DO 20 I=1,2
      DO 10 J=1,2
      K(I2,I,J)=0
      KSTAR(I2,I,J)=0
      P(I2,I,J)=0
      PSTAR(I2,I,J)=0
      S1(I,J)=0
   10 CONTINUE
      HOLD(2,I)=0
      P(I2,I,I)=100000
      PSTAR(I2,I,I)=100000
   20 CONTINUE
   30 CONTINUE
c
      READ(UNIT=52,FMT=*) GMMA(1),DT,ndata
      AMBDA=1.0
      KSKIP=0
C NOTE: GMMA=(L)AMBDA*VARIANCE
```

```
      READ(UNIT=29,FMT=*) HOLD(2,2),HOLD(2,1)
      READ(UNIT=29,FMT=*) R(1,2),R(1,1)
      DO 800 ITIME=1,NDATA
      READ(UNIT=29,FMT=*) E(1,2),E(1,1)
C RECURSION ORDER=2
      DO 400 N=1,2
C FACTORIZE P TO UDUt
      B=P(N,2,2)
      C=P(N,2,1)/B
      A=P(N,1,1)-P(N,2,1)**2/B
C CALCULATE kbar
      DEN=A*R(N,1)**2+B*(C*R(N,1)+R(N,2))**2+GMMA(N)
      IF(DEN.NE.0) GOTO 31
      DEN=.00000001
   31 KBAR(1)=(P(N,1,1)*R(N,1)+P(N,1,2)*R(N,2))/DEN
      KBAR(2)=(P(N,2,1)*R(N,1)+P(N,2,2)*R(N,2))/DEN
C CALCULATE K(t)
      T1=E(N,1)-(K(N,1,1)*R(N,1)+K(N,1,2)*R(N,2))
      T2=E(N,2)-(K(N,2,1)*R(N,1)+K(N,2,2)*R(N,2))
C LINE 50
      K(N,1,1)=K(N,1,1)+T1*KBAR(1)
      K(N,1,2)=K(N,1,2)+T1*KBAR(2)
      K(N,2,1)=K(N,2,1)+T2*KBAR(1)
      K(N,2,2)=K(N,2,2)+T2*KBAR(2)
C ' CALCULATE P(t-1)
      Z1=(P(N,1,1)*R(N,1)+P(N,1,2)*R(N,2))**2
      Z2=(P(N,1,1)*R(N,1)+P(N,1,2)*R(N,2))
      Z2=Z2*(P(N,1,2)*R(N,1)+P(N,2,2)*R(N,2))
      Z4=(P(N,1,2)*R(N,1)+P(N,2,2)*R(N,2))**2
      P(N,1,1)=(P(N,1,1)-Z1/DEN)/AMBDA
      P(N,1,2)=(P(N,1,2)-Z2/DEN)/AMBDA
      P(N,2,1)=P(N,1,2)
      P(N,2,2)=(P(N,2,2)-Z4/DEN)/AMBDA
C CALCULATE KAST
      BS=PSTAR(N,2,2)
      CS=PSTAR(N,2,1)/BS
      AS=PSTAR(N,1,1)-PSTAR(N,2,1)**2/BS
      DEN=AS*E(N,1)**2+BS*(CS*E(N,1)+E(N,2))**2+GMMA(N)
```

```
      IF(DEN.NE.0) GOTO 128
      DEN=.00000001
  128 KAST(1)=(PSTAR(N,1,1)*E(N,1)+PSTAR(N,1,2)*E(N,2))/DEN
      KAST(2)=(PSTAR(N,2,1)*E(N,1)+PSTAR(N,2,2)*E(N,2))/DEN
C CALCULATE KSTAR(t)
      T1=R(N,1)-(KSTAR(N,1,1)*E(N,1)+KSTAR(N,1,2)*E(N,2))
      T2=R(N,2)-(KSTAR(N,2,1)*E(N,1)+KSTAR(N,2,2)*E(N,2))
      KSTAR(N,1,1)=KSTAR(N,1,1)+T1*KAST(1)
      KSTAR(N,1,2)=KSTAR(N,1,2)+T1*KAST(2)
      KSTAR(N,2,1)=KSTAR(N,2,1)+T2*KAST(1)
      KSTAR(N,2,2)=KSTAR(N,2,2)+T2*KAST(2)
C CALCULATE PSTAR(t)
      Y1=(PSTAR(N,1,1)*E(N,1)+PSTAR(N,1,2)*E(N,2))**2
      Y2=(PSTAR(N,1,1)*E(N,1)+PSTAR(N,1,2)*E(N,2))
      Y2=Y2*(PSTAR(N,1,2)*E(N,1)+PSTAR(N,2,2)*E(N,2))
      Y4=(PSTAR(N,1,2)*E(N,1)+PSTAR(N,2,2)*E(N,2))**2
      PSTAR(N,1,1)=(PSTAR(N,1,1)-Y1/DEN)/AMBDA
      PSTAR(N,1,2)=(PSTAR(N,1,2)-Y2/DEN)/AMBDA
      PSTAR(N,2,1)=PSTAR(N,1,2)
      PSTAR(N,2,2)=(PSTAR(N,2,2)-Y4/DEN)/AMBDA
      IF(N.GT.1) GOTO 400
C UPDATE ORDER
      M=N+1
      DO 150 I=1,2
      DO 130 J=1,2
      G1(I,J)=K(1,I,J)
      H1(I,J)=KSTAR(1,I,J)
  130 CONTINUE
      E(M,I)=E(N,I)-(K(N,I,1)*R(N,1)+K(N,I,2)*R(N,2))
      R(M,I)=HOLD(M,I)
      HOLD(M,I)=R(N,I)-(KSTAR(N,I,1)*E(N,1)+KSTAR(N,I,2)*E(N,2))
  150 CONTINUE
C LINE 100
      B=P(N,2,2)
      C=P(N,2,1)/B
      A=P(N,1,1)-P(N,2,1)**2/B
      T3=A*R(N,1)**2+B*(C*R(N,1)+R(N,2))**2
      GMMA(M)=GMMA(N)-AMBDA*T3
```

```
C CLOSE LOOP FOR 2ND STAGE
  400 CONTINUE
C CALCULATE PARAMETER MATRICES
      DO 550 I=1,2
      DO 500 J=1,2
      G2(I,J)=G1(I,J)-(K(2,I,1)*S1(1,J)+K(2,I,2)*S1(2,J))
      H2(I,J)=KSTAR(2,I,J)
  500 CONTINUE
      DO 510 J=3,4
      L=J-2
      G2(I,J)=K(2,I,L)
      H2(I,J)=S1(I,L)-(KSTAR(2,I,1)*G1(1,L)+KSTAR(2,I,2)*G1(2,L))
  510 CONTINUE
  550 CONTINUE
C RECONSTRUCT PARAMETERS
C SOT=PSI*OMEGA*DT
      KSKIP = KSKIP+1
      IF(KSKIP.LT.5) GOTO 695
      SOT=(1+G2(1,3))/(1-G2(1,3))
      ROMEGA=ABS(2-G2(1,1)*(1+SOT))**.5/DT
      RPSI=SOT/ROMEGA/DT
      IF(G2(1,2).EQ.0) GOTO 695
      IF(SOT.EQ.-1) GOTO 695
      TME=DT*ITIME
      RMASS=(DT**2)/(G2(1,2)+G2(1,4))/(1+SOT)
C      WRITE(UNIT=25,FMT=*) TME,ROMEGA
      SI=RPSI*100
C      WRITE(UNIT=26,FMT=*) TME,SI
C      WRITE(UNIT=27,FMT=*) TME,RMASS
      KSKIP=0
C SHIFT ANY TIME PARAMETERS
  695 DO 710 I=1,2
      DO 700 J=1,2
      S1(I,J)=H1(I,J)
  700 CONTINUE
      R(1,I)=E(1,I)
  710 CONTINUE
C  NEW DATA POINT
```

```
800 CONTINUE
    WRITE(UNIT=25,FMT=*) ROMEGA,SI,RMASS
    END
```

C.4 The Extended RLS Algorithm

```
C PROGRAM "ERLS"
C
      DIMENSION A(10),B(10), U(3), Y(3)
      OPEN(UNIT=17,STATUS='OLD')
      OPEN(UNIT=21,STATUS='NEW')
      OPEN(UNIT=22,STATUS='NEW')
      OPEN(UNIT=23,STATUS='NEW')
      OPEN(UNIT=24,STATUS='NEW')
C
      NDATA=1251
      DT=.10
      BU=.1
      BL=-.1
      J=-1
C
      DO 50 I=1,10
      A(I)=0
   50 CONTINUE
      EZ=0
      FZ=0
      GZ=0
      HZ=0
      Q=0
      H=0
      I=0
      K=0
C
  100 READ(UNIT=17,FMT=*) U(1),Y(1)
      IF(Y(1).GT.BU) GOTO 110
      IF(Y(1).LT.BL) GOTO 120
      Q=Q+1
      GOTO 100
  110 I=-1
      GOTO 150
  120 I=1
  150 READ(UNIT=17,FMT=*) U(2),Y(2)
```

```
      IF(Y(2).GT.BU) GOTO 160
      IF(Y(2).LT.BL) GOTO 170
  155 CONTINUE
      Q=Q+2
      I=0
      GOTO 100
  160 IF(I.NE.-1) GOTO 155
      GOTO 200
  170 IF(I.NE.1) GOTO 155
  200 CONTINUE
      M=Q+3
      N=0
C
      DO 500 N=M,NDATA
      READ(UNIT=17,FMT=*) U(3),Y(3)
      IF(Y(3).GT.BU) GOTO 210
      IF(Y(3).LT.BL) GOTO 220
  205 CONTINUE
      Q=N
      I=0
      GOTO 100
  210 IF(I.NE.-1) GOTO 205
      GOTO 230
  220 IF(I.NE.1) GOTO 205
C
  230 Z=-(Y(3)-2*Y(2)+Y(1))/DT**2
      E=(Y(3)-Y(1))/DT/3
      F=Y(2)
      G=-U(2)/2.0
      H=I*2.0
  250 A(1)=A(1)+E**2
      A(2)=A(2)+E*F
      A(3)=A(3)+E*G
      A(4)=A(4)+E*H
      A(5)=A(5)+F**2
      A(6)=A(6)+F*G
      A(7)=A(7)+F*H
      A(8)=A(8)+G**2
```

```
      A(9)=A(9)+G*H
      A(10)=A(10)+H**2
      DO 260 L=1,10
      IF(A(L).NE.0)GOTO 260
      A(L)=1.E-10
  260 CONTINUE
C
      EZ=EZ+E*Z
      FZ=FZ+F*Z
      GZ=GZ+G*Z
      HZ=HZ+H*Z
C INVERTING HTH (4x4)
C USEFUL VARIABLES:
      C1=A(1)*A(5)-A(2)**2
      C2=A(2)*A(6)-A(3)*A(5)
      C3=A(3)*A(7)-A(4)*A(6)
      C4=A(6)*A(9)-A(7)*A(8)
      C5=A(8)*A(10)-A(9)**2
      C6=A(6)*A(10)-A(9)*A(7)
C
      D1=A(1)*A(6)-A(2)*A(3)
      D2=A(2)*A(7)-A(4)*A(5)
      D3=A(1)*A(7)-A(2)*A(4)
      D4=A(3)*A(9)-A(4)*A(8)
      D5=A(3)*A(10)-A(4)*A(9)
C COMPUTING INVERSE
      IF(K.LT.4) GOTO 280
      DEN=C1*C5-D1*C6+D3*C4+D5*C2-D4*D2+C3**2
      IF(DEN.EQ.0) GOTO 280
      B(1)=(A(5)*C5-A(6)*C6+A(7)*C4)/DEN
      B(2)=(A(3)*C6-A(2)*C5-A(4)*C4)/DEN
      B(3)=(A(10)*C2-A(9)*D2+A(7)*C3)/DEN
      B(4)=(A(8)*D2-A(9)*C2-A(6)*C3)/DEN
      B(5)=(A(1)*C5+A(4)*D4-A(3)*D5)/DEN
C
      B(6)=(A(2)*D5-A(1)*C6-A(4)*C3)/DEN
      B(7)=(A(1)*C4-A(2)*D4+A(3)*C3)/DEN
      B(8)=(A(10)*C1-A(7)*D3+A(4)*D2)/DEN
```

```
      B(9)=(A(7)*D1-A(4)*C2-A(9)*C1)/DEN
      B(10)=(A(8)*C1+A(3)*C2-A(6)*D1)/DEN
C CALCULATE VARIABLES
      X1=B(1)*EZ+B(2)*FZ+B(3)*GZ+B(4)*HZ
      X2=B(2)*EZ+B(5)*FZ+B(6)*GZ+B(7)*HZ
      X3=B(3)*EZ+B(6)*FZ+B(8)*GZ+B(9)*HZ
      X4=B(4)*EZ+B(7)*FZ+B(9)*GZ+B(10)*HZ
C
      X1=X1/3
      X3=X3/2.0
      X4=X4*2.0
      ROMEGA=ABS(X2)**.5
      RPSI=X1/ROMEGA
      RMASS=1/X3
      DELTA=(X4/X2)
C CALCULATE FOR NEW GAP SIZE
      CF=2*RPSI/ROMEGA
      IF (K.LT.200) GOTO 270
      VEL=(Y(2)-Y(1))/DT
      IF(VEL.GT.0.) GOTO 265
      BU=1.1*(DELTA-CF*VEL)
      BL=-DELTA
      GOTO 270
  265 BU=DELTA
      BL=1.1*(0-DELTA-CF*VEL)
C
  270 T=N*DT
      J=J+1
      IF(J.LT.10) GOTO 280
      WRITE(UNIT=21,FMT=*) T,ROMEGA
      SI=RPSI*100
      WRITE(UNIT=22,FMT=*) T,SI
      WRITE(UNIT=23,FMT=*) T,RMASS
      WRITE(UNIT=24,FMT=*) T,DELTA
      J=0
  280 K=K+1
C
  490 Y(1)=Y(2)
```

```
      Y(2)=Y(3)
      U(2)=U(3)
  500 CONTINUE
      END
```