IDENTIFICATION OF NONLINEAR SYSTEMS WITH A STATE-DEPENDENT ARX MODEL USING GENERALIZED LEAST SQUARES

A Dissertation

Presented to

The Faculty of the Department of Mechanical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in Mechanical Engineering

by:

Robert Warren Clark, Jr.

May 2007

IDENTIFICATION OF NONLINEAR SYSTEMS WITH A STATE-DEPENDENT ARX MODEL USING GENERALIZED LEAST SQUARES

[illegible]

Robert W. Clark, Jr.

Approved:

[illegible]

Chairman of the Committee
D. Zimmerman, Professor and
Associate Chairman,
Mechanical Engineering

Committee Members:

[illegible]

M. A. Franchek, Professor and Chairman,
Mechanical Engineering

[illegible]

K. Grigoriadis, Professor,
Mechanical Engineering

[illegible]

G. Song, Associate Professor,
Mechanical Engineering

[illegible]

Y. L. Mo, Professor,
Civil and Environmental Engineering

[illegible]

L. C. Witte, Associate Dean
Cullen College of Engineering

[illegible]

M. A. Franchek, Professor and Chairman,
Mechanical Engineering

Acknowledgements

I am grateful to the University of Houston's Cullen College of Engineering faculty for their dedication to higher education, especially to my advisor Dr. David Zimmerman. I am also indebted to Trina Johnson for her proactive assistance in ensuring my compliance with the University graduation policies. I would also like to acknowledge my employer's contribution to my educational endeavors by financing my tuition and fees. I also appreciate that Los Alamos National Laboratory in New Mexico made test data and reports available to the public for their eight degree-of-freedom test rig.

None of this effort would have been possible without the support of my family. During the pursuit of my Doctorate Degree, my wife Susanne gave birth to our daughter Annabelle. Susanne continued her fulltime career while caring for our child and gave me the opportunity to complete my degree. I dedicate this research and dissertation to the two most special people in my life: Susanne and Annabelle.

IDENTIFICATION OF NONLINEAR SYSTEMS WITH A STATE-DEPENDENT ARX MODEL USING GENERALIZED LEAST SQUARES

An Abstract of a

Dissertation

Presented to

The Faculty of the Department of Mechanical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in Mechanical Engineering

by:

Robert Warren Clark, Jr.

May 2007

Abstract

A single-input multiple-output nonlinear identification method is developed and applied to damage detection. The identification method is based on an Auto Regressive Exogenous stochastic model. The coefficients of the model are a function of the output states. The effects of noise on the input and outputs are minimized by utilizing Generalized Least Squares. The identification method is demonstrated on a three degree-of-freedom nonlinear numerical example. Data from tests performed at Los Alamos on a nonlinear eight degree-of-freedom system are used to demonstrate its performance in a damage detection application.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

The utility of system identification for health monitoring and damage detection has been documented in publications for numerous years [1,2]. Historically, this effort has focused on assuming the system being evaluated is linear. This assumption is adequate for many systems over the operational envelope of interest, an example being small amplitude oscillations of a welded structure.

This dissertation explores a method of identification for damage detection of nonlinear systems focused towards, but not limited to, structures and mechanisms. These nonlinearities can be soft, as in geometric stiffening, and hard [3], such as bilinear springs and stiffness dead-bands generated by cracks opening and closing. Furthermore, the goal is to develop a method that can be applied to a Single-Input/Multiple-Output (SI/MO) black-box system with both the input and outputs corrupted with noise.

Health monitoring is generally accomplished by analyzing the baseline system to establish a benchmark model and then periodically reanalyze the system to generate models to compare to the baseline. Changes in the model signify changes in the system, which may be caused by damage.

For linear systems, the first step of establishing a benchmark model typically involves characterizing the system with eigenvalues and eigenvectors (natural frequencies and mode shapes) by utilizing a modal parameter extraction method [4]. Since the concept of natural frequencies and mode shapes only apply to linear systems, or nonlinear systems linearized about an operating point, this approach is not suitable for a system with hard nonlinearities.

The nonlinear identification will be accomplished using a state-dependent ARX model. The state-dependent coefficients will be used for detecting damage. This will be accomplished by establishing the bounds of the coefficients for the undamaged system. Identifications performed at later periods will then be check against the established bounds to check for changes in the system.

Chapter 2 - Nonlinear System Identification Research

This chapter summarizes past research on nonlinear system identification and outlines the approach proposed within this dissertation.

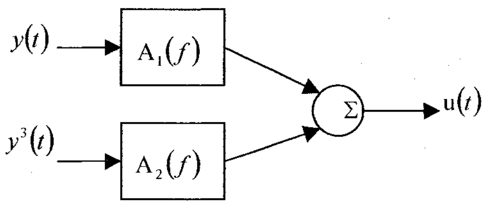
2.1 Past Research

There are applicable frequency-domain methods for the identification of nonlinear systems. One such method is a reverse identification technique by Bendat and is covered in Reference [5]. This method changes Single-Input/Single-Output (SI/SO) nonlinear models into a reverse Multiple-Input/Single-Output (MI/SO) linear models. The Duffing SI/SO nonlinear system (Equation 2-1) can be used as a simple example of this method

$$m\ddot{y}(t)+c\dot{y}(t)+ky(t)+k_3y^3(t)=u(t) \tag{2-1}$$

$$u(t)=\text{physical excitation}$$
$$y(t)=\text{physical response}.$$

By reversing the defined input and output, as well as treating the linear and cubic terms as separate inputs, Equation 2-1 can be represented as a two-input/single-output system (Figure 2-1).



$A_1(f)$ & $A_2(f)$ are frequency response functions

Figure 2-1. Duffing Reverse Two-Input/Single-Output Linear System

Bendat covers methods of separating the linear and nonlinear contributions from data and then generating the appropriate frequency response functions for reverse systems of the form in Figure 2-1. This method has limitations when applying it to SI/MO systems and to systems in which no prior knowledge exists. This is due to the method for multiple degree-of-freedom systems being a gray-box method, which is a system of equations derived based on physical interpretation. Frequency response functions are then determined using reverse identification to complete the model.

Feldman developed a method for nonlinear system identification using the unique properties of the Hilbert transform [6, 7]. The Hilbert transform of a real-valued function y(t) is defined by

$$\mathrm{H}[y(t)] = \tilde{y}(t) = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{y(\tau)}{t-\tau}d\tau \,. \qquad 2\text{-}2$$

In simple terms, the Hilbert transform of a signal is that signal shifted in phase by -π/2. It can be calculated by taking the Fourier transform of a signal, shifting the phase by -π/2 and then calculating the inverse Fourier transform. By using the original and the transformed signal, the instantaneous phase (ϕ), damped natural frequency (ω) and amplitude (Y) can be calculated using

$$\phi(t) = \tan^{-1}\left(\frac{y(t)}{\tilde{y}(t)}\right), \qquad 2\text{-}3$$

$$\omega(t) = \dot{\phi}(t) = \frac{d\phi(t)}{dt} \text{ and} \qquad 2\text{-}4$$

$$Y(t) = \sqrt{y(t)^2 + \tilde{y}(t)^2} \,. \qquad 2\text{-}5$$

Feldman shows methods of calculating the natural frequency and damping characteristics of an oscillatory system for both free and forced vibration.

To be able to apply Feldman's method to a multiple degree-of-freedom system, one must utilize the fact that when a frequency of excitation is close to one of the undamped natural frequencies, the mode shape is identical to the normal mode shape of the system. Therefore, the vibration of the system depends only on the parameters of this particular mode. This may not be possible for some systems and might mask the characteristic of the nonlinearities. Other observations obtained through application of this method show it works well for frequencies but not for damping.

Huang has a patented method for reducing a nonlinear/non-stationary time series into Intrinsic Mode Functions (IMFs), known as Empirical Mode Decomposition (EMD) [8]. It employs a series of de-trending passes, known as sifting, to produce the IMFs. Each IMF oscillates about a mean of zero with as many maximum and minimums as zero crossings. This causes each IMF to be "Hilbert friendly" and it makes no assumptions of the waveform. Huang then applies the Hilbert Transform to the IMFs and calculates the instantaneous frequencies as a function of time. This data is then displayed in a spectra graph format.

The sifting process is an effective filter with interesting characteristics. This is demonstrated by its ability to decompose the signal (Figure 2-2) generated by summing the following three waveforms:

Wave 1: Sine, amplitude of 2 and frequency of 0.16 Hz

Wave 2: Chirp, beginning at 0.1 Hz and ending at 1 Hz over 10 seconds

Wave 3: Triangular, amplitude of 0.25 and frequency of 25 Hz

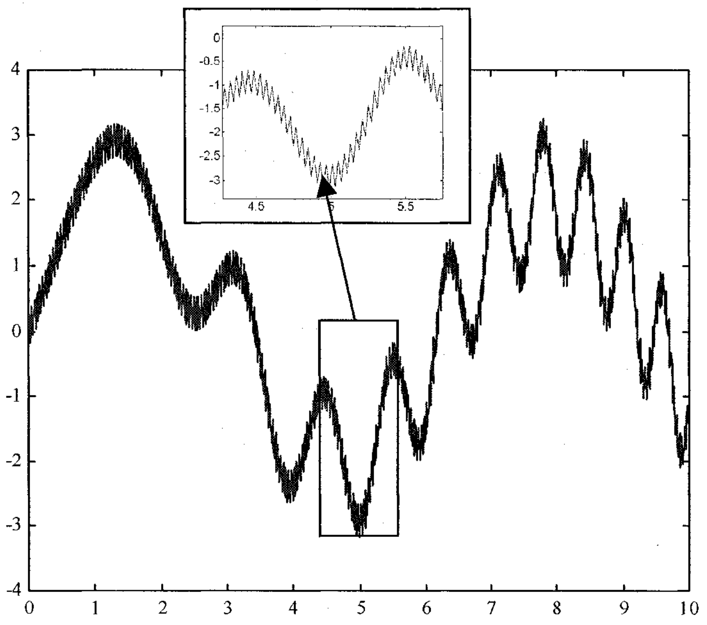


Figure 2-2. Example Case for Huang's Sifting Method

The summation of the three waveforms shown in Figure 2-2 was sifted with the algorithm described by Huang. Figure 2-3 shows the results from the sifting process. It can be seen that the method extracted the three original waveforms.

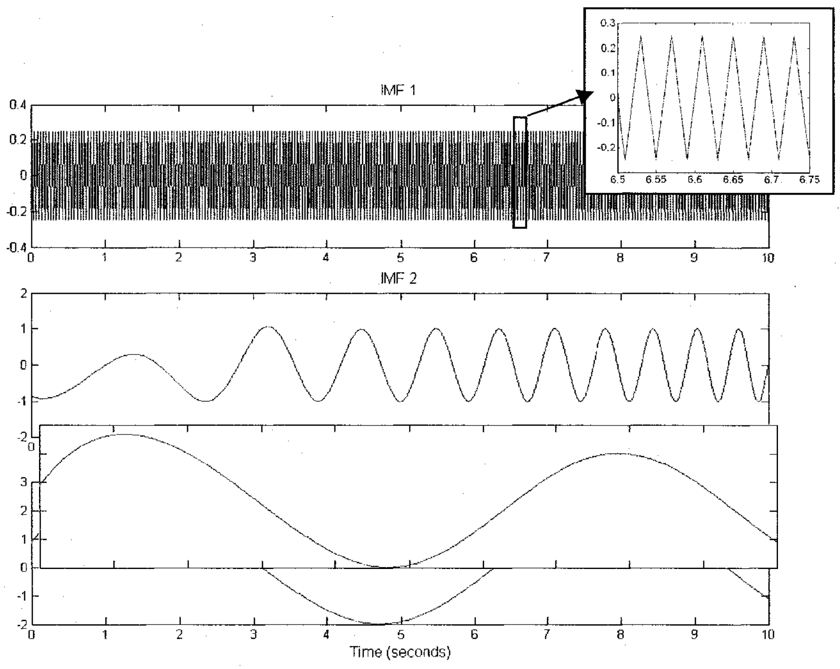


Figure 2-3. IMFs Identified using EMD from [8]

Although this method does not attempt to construct a model, it was hoped that it could be applied as a filter for Feldman's method to single out a modes's contribution. However, the de-trending process generates misleading results for some systems, which produces erroneous identification. For instance, one IMF will transition between two different phenomena confounding the data beyond usefulness.

There is another class of nonlinear identification methods that use stochastic models extracted from time-domain data. These methods are based on different forms of Auto-Regressive (AR) models and generate state-dependent coefficients,

$$y(n)=\sum_{i=1}^{P}a(i,y(n-1))y(n-i)+\sum_{j=0}^{Q}b(j,y(n-1))u(n-j)+e(n)\ . \qquad 2\text{-}6$$

Young [9] has developed a two-step process for generating this form of a model. The first involves performing Recursive Least Squares (RLS) to identify the coefficients of the model as a function of time, assuming the statistical properties of the signal are changing slow in relation to the rates the states are changing. By knowing the states and coefficients as a function of time, the relationship between the coefficients and the states can be estimated. By curve fitting this data, equations of the coefficients as a function of the states can be generated. Other researches use Young's process as the genesis of methods using state-dependent models, Toivonen [10] applies velocity-based linearization to form discrete state-space models that are state-dependent. Åkesson's [11] work is a continuation of the work performed by Toivonen, with more details on using neural networks to find the state-dependent parameters.

Techniques have been developed that calculate the state-dependent coefficients directly from the time-domain data. This is accomplished by expanding the regression vector coefficients using a general form of an orthogonal basis function,

$$a(i, y(n-1)) = \sum_{k=0}^{V} \alpha(i,k)\gamma_k(y(n-1)) \text{ and} \qquad 2\text{-}7$$

$$b(j, y(n-1)) = \sum_{k=0}^{V} \beta(j,k)\gamma_k(y(n-1)) \, , \qquad 2\text{-}8$$

where α and β are scalar weighting coefficients, and γ is a basis function. The error is then minimized by adjusting the coefficients of the basis vectors. For instance, Ljung [12] optimizes the weighting, dilation and phase of the basis function to minimize the error. Care must be taken during the optimization to ensure the solution is not a local minima.

Peng [13, 14] utilizes a neural network optimization method to find the centers, scale factors and coefficients of Radial-Basis Functions (RBF) to form the state-dependent coefficients of an ARX model. This method is applicable to smooth nonlinear systems.

Polynomial forms of the state-dependent coefficient are applicable to characterize smooth global nonlinear properties; similar to using RBF to expand the state-dependent coefficients. Billings [15] breaks the stochastic model down to multiple submodels; polynomial, wavelet and noise models. The polynomial model represents global properties, where as local properties are characterized with the wavelet representation.

Hu [16] also uses neural networks to find coefficients to state-dependent models. Hu's concern with neural networks not being user-friendly or easy-to-use for controller design and fault diagnosis, led him to a model consisting of two parts. The two parts are a macro-part, which is constructed with application specific knowledge and a kernel part, which uses nonlinear non-parametric models such as neural or neurofuzzy networks. It is not clear to this author how this addresses ease-of-use.

Gomez [17] presents a robust method of finding state-dependent coefficients of the form in Equations 2-7 and 2-8 using least squares that is not susceptible to local minima. However, consistency of estimates can be guaranteed only for noise free cases.

2.2 Proposed Approach

Of the methods reviewed in Section 2.1, the state-dependent ARX methods [9, 11 to 17] show the most promise in meeting the said objective. The associated pitfalls and overhead of performing identifications of multiple models removes Young [9], Billings [15] and Hu [16] from the list of desirable approaches. Being able to identify discontinuities such as dead-bands generated by cracks opening and closing excludes the methods only applicable to smooth nonlinear systems [13, 14]. The user-friendly and ease-of-use issues of neural networks addressed by Hu [16] eliminates [11, 12] from attractive approaches as well. Gomez's [17] technique lacks the ability to address corrupted signals, which excludes it as a possible approach. Therefore, a unique method will be developed to meet the objectives of this research.

The first step to meeting the objective will begin with the development of a process for generating SI/SO ARX models of nonlinear systems, including hard nonlinear characteristics. The nonlinear characteristics of the system will be captured by the coefficients being functions of the states. Noise will be applied to both the input and output signals. Once this method is developed, it will be expanded to SI/MO systems. The accuracy and sensitivity of the method will be evaluated to demonstrate its applicability to nonlinear damage detection.

Chapter 3 - Review of Linear Time-Invariant ARX Model System Identification

For familiarization, a brief description of the use of ARX models for linear time-invariant system identification follows, as well as an example. For a more thorough explanation, see Ljung [12] and Åström [18].

The general form of an ARX model is

$$y(n) = \sum_{i=1}^{P} a(i) y(n-i) + \sum_{j=0}^{Q} b(j) u(n-j) + e(n) . \qquad 3\text{-}1$$

The first summation represents the autoregressive portion of the model and the second the exogenous input. The output *{y(1), y(2),...,y(N)}* resulting from the input *{u(1), u(2),...,u(N)}* are discrete sampled data at evenly spaced intervals. The term *e* represents unknown white noise or error. The schematic representation is shown in Figure 3-1.

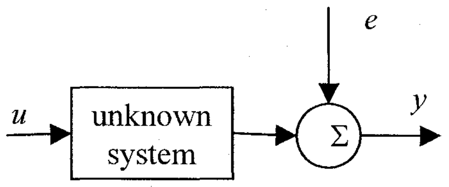


Figure 3-1. SI/SO Unknown System with Noise

The unknown parameters of Equation 3-1 can be rewritten as

$$\theta = \begin{bmatrix} a_1 & \dots & a_P & b_0 & \dots & b_Q \end{bmatrix}^T . \qquad 3\text{-}2$$

Organizing the output and input arrays

$$\varphi^T(k) = \begin{bmatrix} y(k-1) & \dots & y(k-P) & u(k) & \dots & u(k-Q) \end{bmatrix} \qquad 3\text{-}3$$

and

$$\Phi = \begin{bmatrix} \varphi^T(n) \\ \vdots \\ \varphi^T(N) \end{bmatrix} . \qquad 3\text{-}4$$

Equation 3-1 can be rewritten as

$$y(n) = \varphi(n)\theta + e(n) . \qquad 3\text{-}5$$

To estimate the coefficients, the equation error is minimized in the least-squares sense [19] using the function defined as

$$J(\theta) = \sum_{n=1}^{N} [y(n) - \varphi(n)\theta]^2 . \qquad 3\text{-}6$$

This is accomplished by following these steps (assuming $[\Phi^T \Phi]$ is nonsingular):

$$\Phi\theta = y , \qquad 3\text{-}7$$

$$\Phi^T \Phi\theta = \Phi^T y , \qquad 3\text{-}8$$

$$[\Phi^T \Phi]^{-1} [\Phi^T \Phi]\theta = [\Phi^T \Phi]^{-1} \Phi^T y \text{ and} \qquad 3\text{-}9$$

$$\theta = [\Phi^T \Phi]^{-1} \Phi^T y . \qquad 3\text{-}10$$

The effectiveness of this method will be demonstrated using the two Degree-of-Freedom (DOF) system shown in Figure 3-2 and the parameters listed in Table 3-1. Applying the force shown in Table 3-2 to mass 1, one obtains the displacement of mass 2 shown in Figure 3-3 (plot of applied force is also shown in figure for reference).

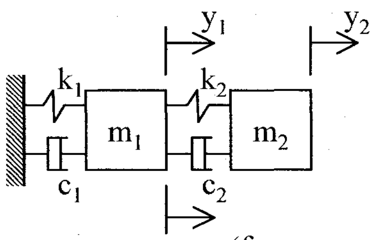


Figure 3-2. Linear Two DOF Spring-Mass-Damper System

Table 3-1. Linear Two DOF Spring-Mass-Damper System Parameters

Parameter	Value
m_1	1
k_1	40
c_1	0.6
m_2	1
k_2	100
c_2	5

Table 3-2. Force Applied to Linear Two DOF Spring-Mass-Damper

Time	Force
0	1
0.2	1
0.21	0
0.5	0
0.51	1
0.6	1
0.61	0
1	0

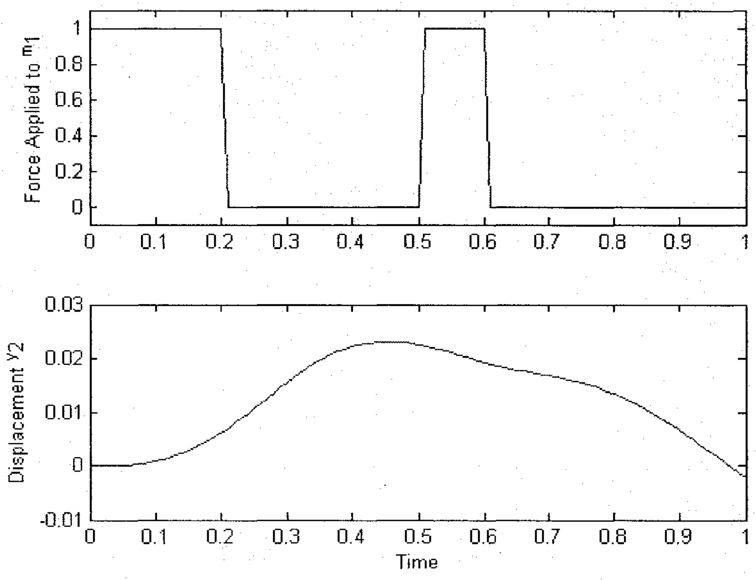


Figure 3-3. Linear Two DOF Example: Applied Force and Response

Using the data in Figure 3-3 (sampled with a delta-time of 0.01) in the ARX method outlined above, with model order consistent with P=4 and Q=3 in Equation 3-1, the coefficients in Table 3-3 were obtained. The MATLAB [20] m-file used to perform this is listed in Appendix A. To demonstrate the validity of the identified model, the response of mass 2 was synthesized. It can be seen in Figure 3-4 that the synthesized response matches the original response quite well.

Table 3-3. ARX Coefficients for Linear Two DOF Spring-Mass-Damper System

Coefficients	**Value**
a_1	3.85510
a_2	-5.59307
a_3	3.62042
a_4	-0.88249
b_0	1.6414E-08
b_1	8.5198E-07
b_2	2.8018E-06
b_3	-2.4013E-06

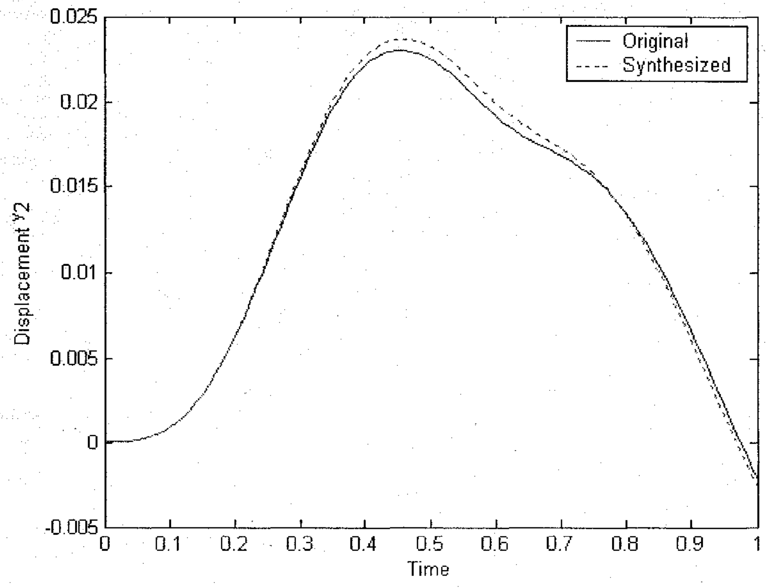


Figure 3-4. Linear Two DOF Example: Synthesized Response Compared to Original

To further illustrate the validity of the identified model, the response of the original and identified system was generated using a different input force than that used for performing the identification. For this, a sequence of random numbers between –0.5 and 0.5 generated at a time interval of 0.01 was used for the input force. Figure 3-5 shows a comparison of the results and again the identified system matches the original quite well.

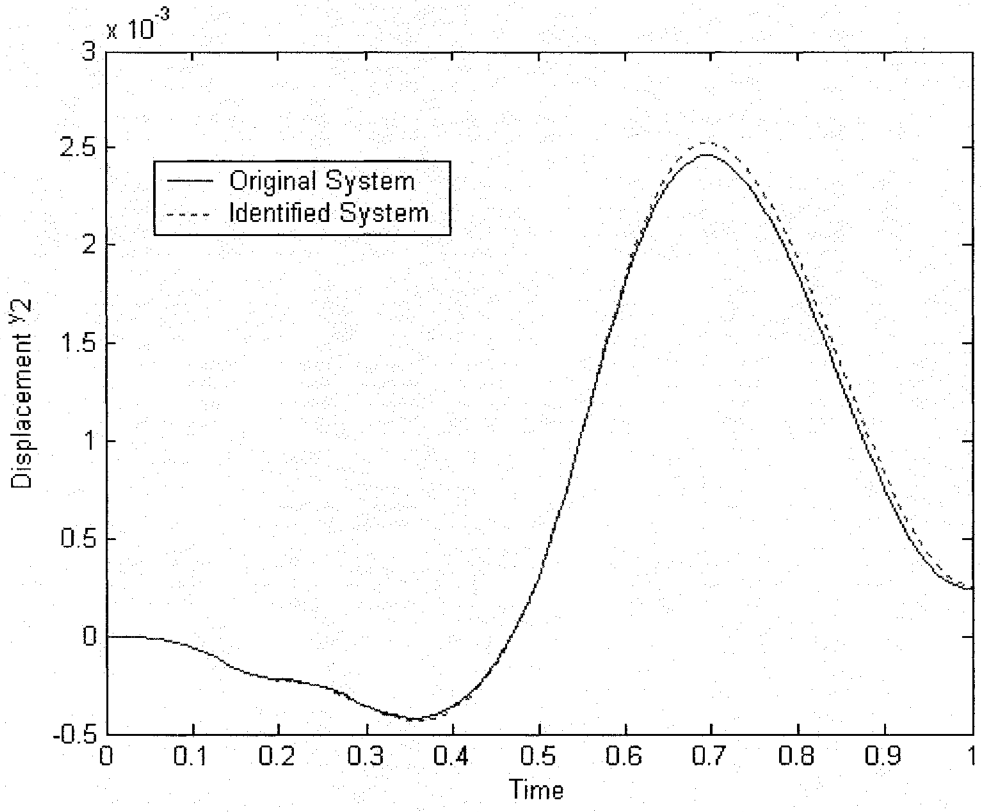


Figure 3-5. Linear Two DOF Example: Response of Original and Identified System to Different Forcing Function

Chapter 4 - Identification Method for Time-Invariant Nonlinear Systems

The utility and simplicity of system identification of a time-invariant linear system using the ARX approach was demonstrated in Chapter 3. Using solution tools such as least-squares contributes to the simplicity. Researchers have expanded this type of identification to time-variant systems by making the coefficients functions of time [21, 22, 23 & 24], while retaining the structure of the problem to use least-squares to find the coefficients. Other similar methods use solutions such as Newton-Raphson, which suffer from the pitfalls of local minima.

Using this type of methodology, except making the coefficients a function of the system's state, will be developed in the following chapters.

4.1 Nonlinear Single-Input/Single-Output

If one had insight into a system to the level of knowing how its characteristics varied with its states, the task of identifying it would be straightforward. For instance, if it were known that a Single-DOF (SDOF) system had a spring that behaved like a third-order polynomial, it would be included in the setup of the formulation. Unfortunately, one rarely has this luxury. So a method of forming a general nonlinear system is needed. Currently a popular way of approaching this is by utilizing basis functions, Zou, et al [22]. However, Zou applies this method to time-varying systems. What follows is an adaptation of Zou's method for time varying systems to systems that have time invariant nonlinear characteristics.

The ARX model modified to have coefficients as a function of the results from the previous time step is

$$y(n) = \sum_{i=1}^{P} a(i, y(n-1))y(n-i) + \sum_{j=0}^{Q} b(j, y(n-1))u(n-j) + e(n) \ . \tag{4-1}$$

Expanding the coefficient using a basis set results in

$$a(i, y(n-1)) = \sum_{k=0}^{V} \alpha(i,k)\gamma_k(y(n-1)) \text{ and} \tag{4-2}$$

$$b(j, y(n-1)) = \sum_{k=0}^{V} \beta(j,k)\gamma_k(y(n-1)) \ , \tag{4-3}$$

where α and β are scalar weighting coefficients, γ is a basis function, and e is error due to white noise. Figure 4-1 through Figure 4-3 show examples of basis sets for use in the Equations 4-2 and 4-3. The first two basis sets (Walsh functions and Block pulse) are well suited for modeling systems with hard nonlinear characteristics, where as Chebyshev Polynomials are more appropriate for smooth nonlinear systems. Chebyshev Polynomials have also proved useful for systems with excessive noise corruption. The MATLAB m-files used to generate the three basis sets are listed in Appendix B. Prior to use in Equations 4-2 and 4-3, the basis need to be mapped to the range of y in the data set being used (0 becomes the minimum and 1 the maximum value of y).

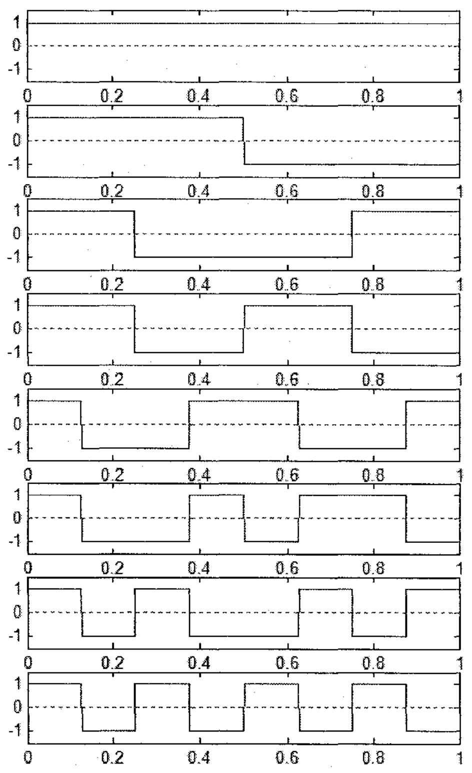


Figure 4-1. First Eight Walsh Functions

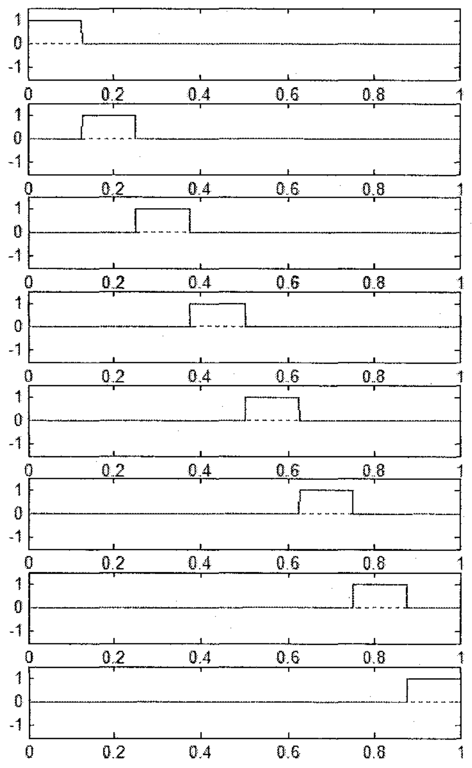


Figure 4-2. Series of Eight Block Pulses

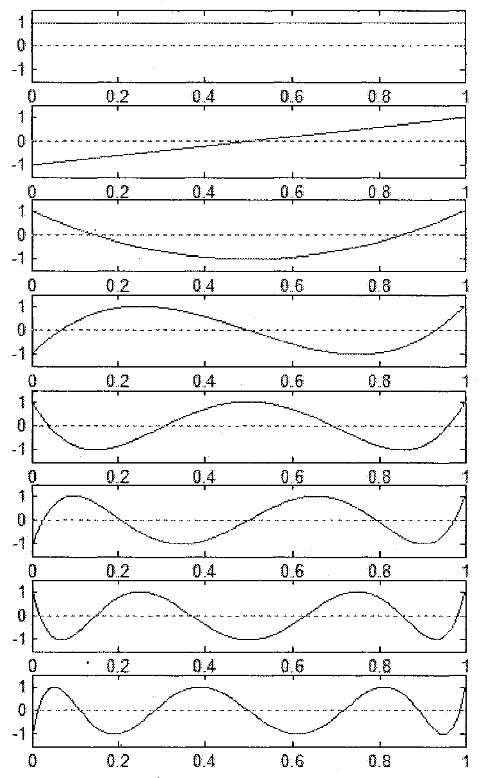


Figure 4-3. First Eight Chebyshev Polynomials of the First Kind

Substituting Equations 4-2 and 4-3 into Equation **Error! Reference source not found.** results in

$$y(n)=\sum_{i=1}^{P}\sum_{k=0}^{V}\alpha(i,k)\gamma_k(y(n-1))y(n-i)+\sum_{j=0}^{Q}\sum_{k=0}^{V}\beta(j,k)\gamma_k(y(n-1))u(n-j)+e(n)\ . \tag{4-4}$$

Equation 4-4 can be simplified by defining two variable definitions

$$y_k(n-i)=\gamma_k(y(n-1))y(n-i) \text{ and}$$
$$u_k(n-j)=\gamma_k(y(n-1))u(n-j)\ . \tag{4-5}$$

Using the above variables transforms Equation 4-4 to

$$y(n)=\sum_{i=1}^{P}\sum_{k=0}^{V}\alpha(i,k)y_k(n-i)+\sum_{j=0}^{Q}\sum_{k=0}^{V}\beta(j,k)u_k(n-j)+e(n)\ . \tag{4-6}$$

Writing out the product of the basis with the input and output arrays yields

$$M = [y_0(\mathbf{n}-1) \;\ldots\; y_V(\mathbf{n}-1) \;\; y_0(\mathbf{n}-2) \;\ldots\; y_V(\mathbf{n}-P) \\ u_0(\mathbf{n}) \;\ldots\; u_V(\mathbf{n}-1) \;\; u_0(\mathbf{n}-2) \;\ldots\; u_V(\mathbf{n}-Q)], \qquad 4\text{-}7$$

where $y_k(n-i) = \gamma_k(y(\mathbf{n}-1))y(\mathbf{n}-i)$
$= [\gamma_k(y(0))y(1-i) \;\; \gamma_k(y(1))y(2-i) \;\ldots$
$\gamma_k(y(N-1))y(N-i)]^T$,

$u_k(n-j) = \gamma_k(y(\mathbf{n}-1))u(\mathbf{n}-j)$
$= [\gamma_k(y(0))u(n-j) \;\; \gamma_k(y(1))y(2-j) \;\ldots$
$\gamma_k(y(N-1))y(N-i)]^T$

and $\mathbf{n}$ = 1 to N. The coefficients in Equation 4-6 can be arranged in a vector as

$$\Psi = [\alpha(1,0) \;\ldots\; \alpha(1,V) \;\; \alpha(2,0) \;\ldots\; \alpha(P,V) \\ \beta(0,0) \;\ldots\; \beta(0,V) \;\; \beta(1,0) \;\ldots\; \beta(Q,V)]^T . \qquad 4\text{-}8$$

Using the definitions shown in Equations 4-7 and 4-8 with Equation 4-6, one obtains a similar expression as shown in Equation 3-5,

$$y = M\Psi + e \; . \qquad 4\text{-}9$$

As with Equation 3-5, if $M^T M$ is nonsingular, the coefficients can be found using Equation 3-10. If each of the columns of M are not linear independent and therefore $M^T M$ is singular, a method must be applied to remove the columns of redundant information. In Zou's time-variant application, he compares column by column to check that each one is linear independent. A more efficient method of separating the linear independent vectors is by using QR factorization [19].

Once the coefficients are found for Equation 4-9, it is possible to construct the state-dependent coefficients for the ARX model (Equation **Error! Reference source not found.**). This can be accomplished by using Equations 4-2 and 4-3 for a and b, respectively. Due to the model only being valid for the range of y that was used in the identification process, it should not be used to predict behavior outside that range. In the event y does go slightly outside the limits, the coefficients should not be extrapolated, but held at the value at the limit.

Using the above outlined method, an algorithm can be written to generate a state-dependent Nonlinear ARX (NL-ARX) model for SI/SO systems. The following outlines the steps for such an algorithm. Unless otherwise noted, the referenced MATLAB [20] m-files are listed in Appendix C. The master file that follows the below steps is NLARX.m.

1. Choose model parameters: model order (P & Q), basis set(s) to be used, and number of vectors to be used in basis set (walsh.m cheby.m and/or block.m from Appendix B)
2. Construct y_k and u_k vectors (pool.m)
3. Find the linear independent vectors in y_k and u_k (linindi.m)
4. Construct M matrix from the linear independent y_k and u_k vectors
5. Solve for the α and β coefficients using Equation 3-10
6. Construct the state-dependent NL-ARX coefficients a and b (assemble.m)
7. Truncate the coefficients above and below range of identification (trim.m)
8. Reconstruct y to verify model (timeh.m)

4.1.1 Example Nonlinear Single-Input/Single-Output System

A SDOF system with a hard nonlinearity was used to demonstrate this method (Figure 4-4). The parameters used for this system are listed in Table 4-1. The forcing function was a summation of two sine waves

$$u = \sin(10.1 \cdot t) + \sin(3.2 \cdot t) \tag{4-10}$$

and is shown in Figure 4-5 along with the response of the mass.

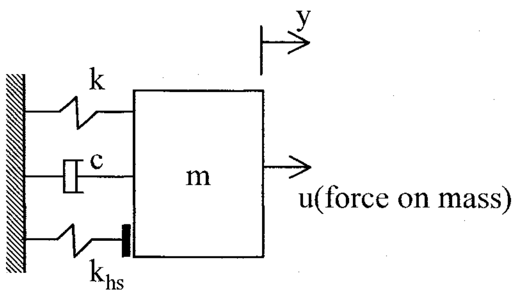


Figure 4-4. SDOF Nonlinear System with a Hard Stop

Table 4-1. SDOF Nonlinear System Parameters

Parameter	Value
m	1
k	20
c	0.5
k_{hs} (y < 0)	2000

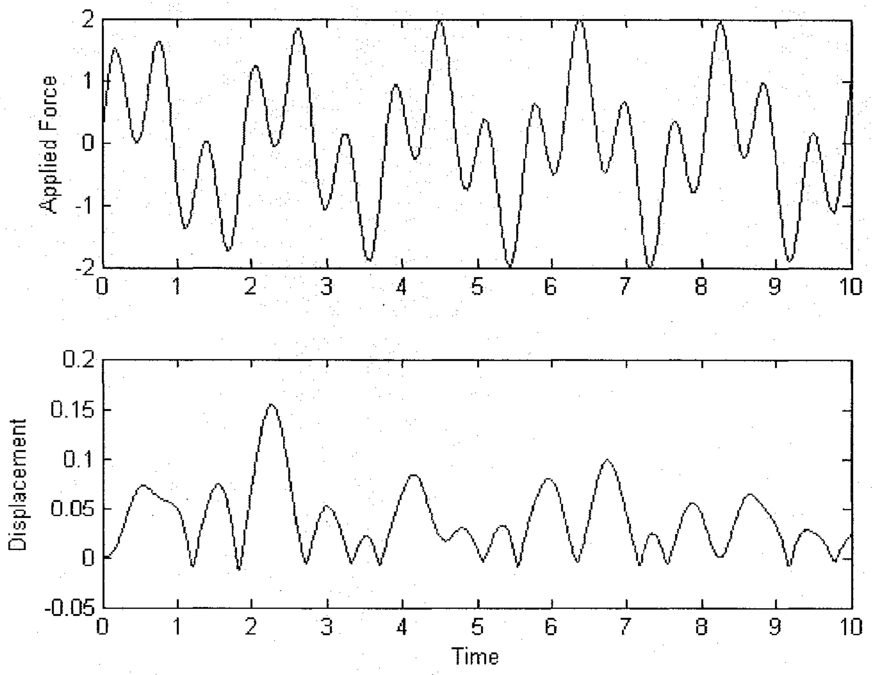


Figure 4-5. Nonlinear SDOF Example: Applied Force and Response

The NL-ARX method outlined above was applied to the data in Figure 4-5 (sampled with a delta-time of 0.01). The parameters used in the identification (Appendix C, NLARX.m) are listed in Table 4-2 and the resulting coefficients are plotted in Figure 4-6 and Figure 4-7 as a function of *y*. Walsh functions were chosen for the basis set due to their ability to duplicate discontinuities, which is demonstrated in Figure 4-6 and Figure 4-7 by the discontinuity in the coefficients at zero displacement resulting from the hard stop. The optimum number of Walsh functions was found by performing a series of identifications with a range of values of Walsh functions (e.g., 1 to 16). The Root-Sum-Square-Error (RSSE) of the synthesized output compared to the original output was calculated for each identified model. The results from the case with the lowest RSSE were then compared visually with the original output to confirm the validity of the fit.

Table 4-2. NL-ARX Parameters used in SDOF Nonlinear System Identification

Parameter	**Value**
P	2
Q	1
Number of Walsh Functions	14
Length of Block Vectors	1000

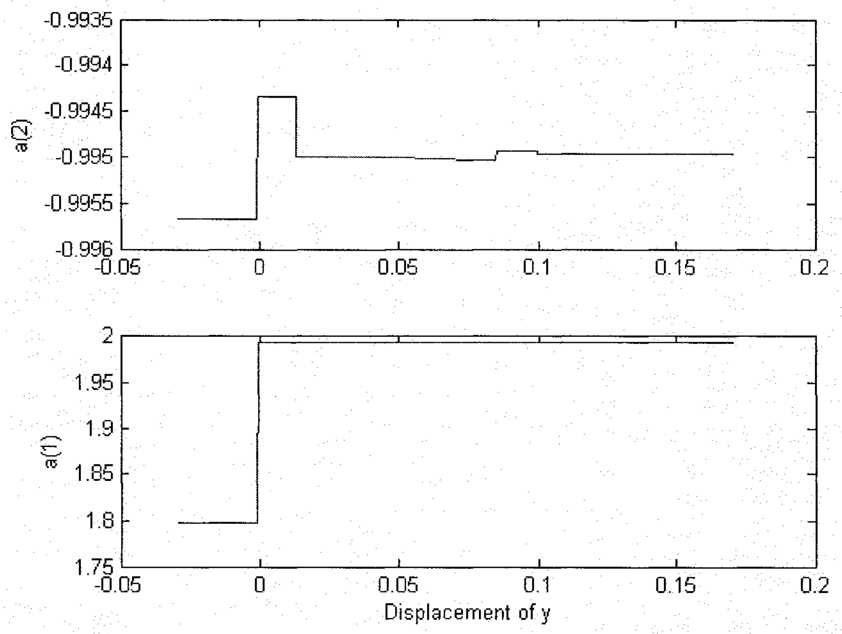


Figure 4-6. Nonlinear SDOF Example: NL-ARX "*a*" Coefficients as a Function of Displacement

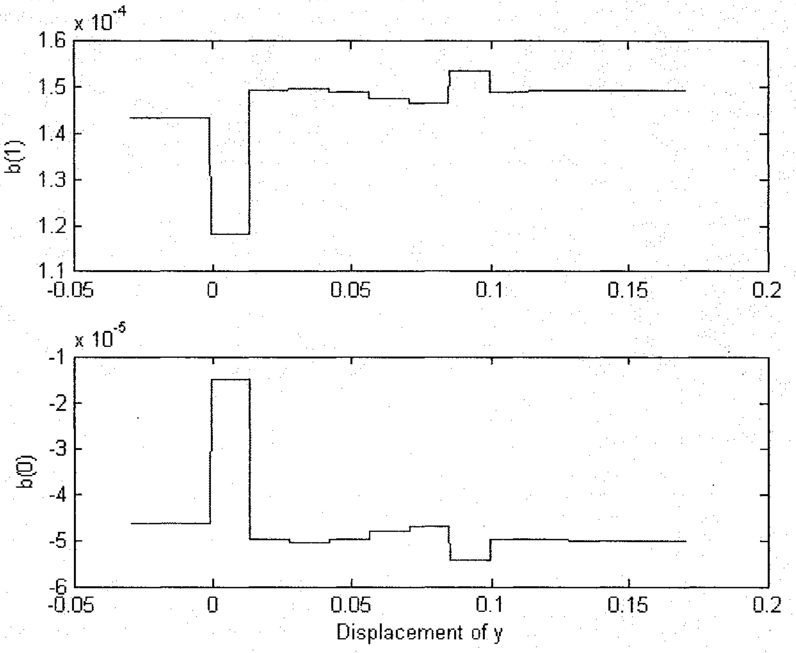


Figure 4-7. Nonlinear SDOF Example: NL-ARX "*b*" Coefficients as a Function of Displacement

As in Chapter 3, to demonstrate the validity of the identified model, the response was synthesized. It can be seen in Figure 4-8 that the synthesized response matches the original response very well.

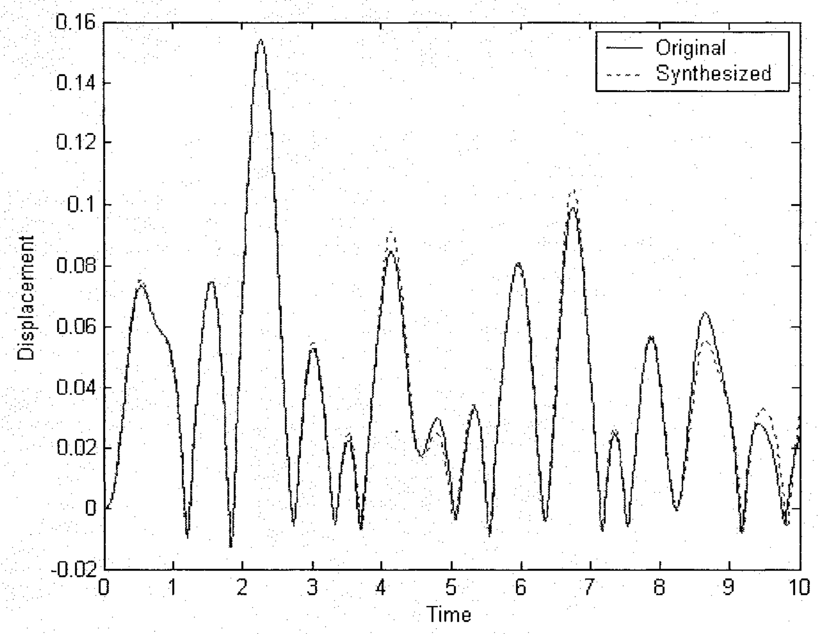


Figure 4-8. Nonlinear SDOF Example: Synthesized Response Compared to Original

As before, both the original system and the model constructed from the data were driven by an input other than the one used for the identification process, see Figure 4-9. These two outputs were than compared to further demonstrate the accuracy of the identified model. Again the match, shown in Figure 4-10, is close.

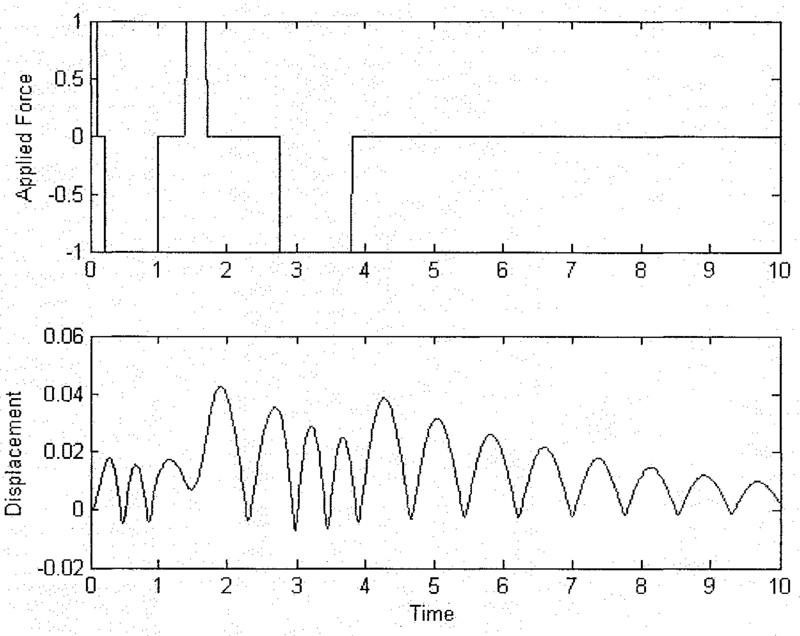


Figure 4-9. Nonlinear SDOF Example: New Forcing Function and Response of Original System

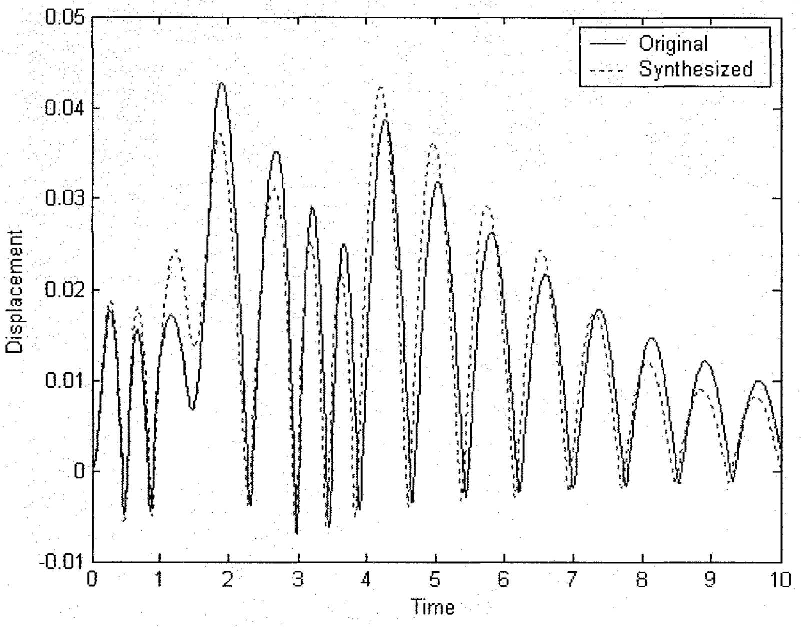


Figure 4-10. Nonlinear SDOF Example: Response of Original and Identified System to Different Forcing Function

The accuracy of the identification process can be greatly improved if data at a higher sampling rate are used. This will be demonstrated by repeating the previous example with a delta-time of 0.001 (the previous example used 0.01). The identification results in an even closer fit, which is nearly indistinguishable from the original system, shown in Figure 4-11.

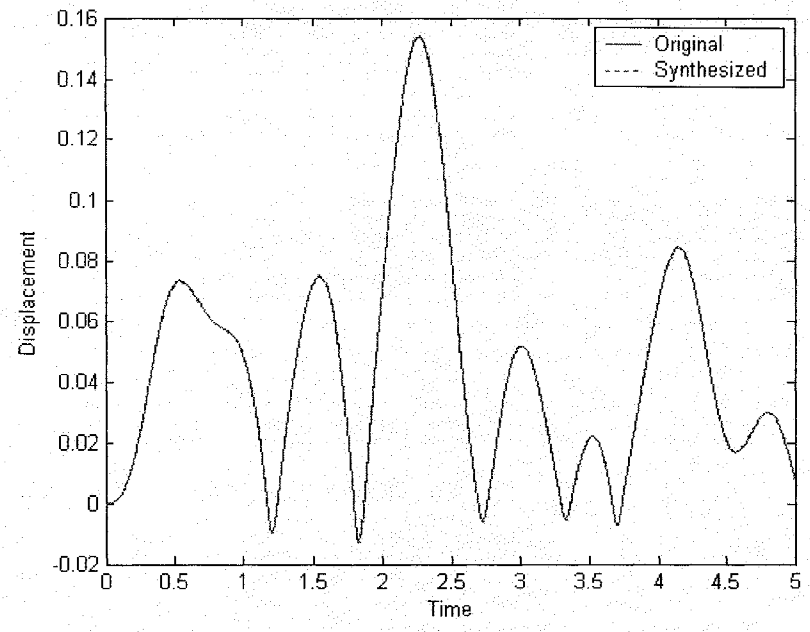


Figure 4-11. Nonlinear SDOF Example: Identification of System with 0.001 Sampling Period

The exercise of applying a different input than was used for the original identification to this system was repeated with the parameters identified with the smaller sampling period. This yielded much closer results as well, shown in Figure 4-12.

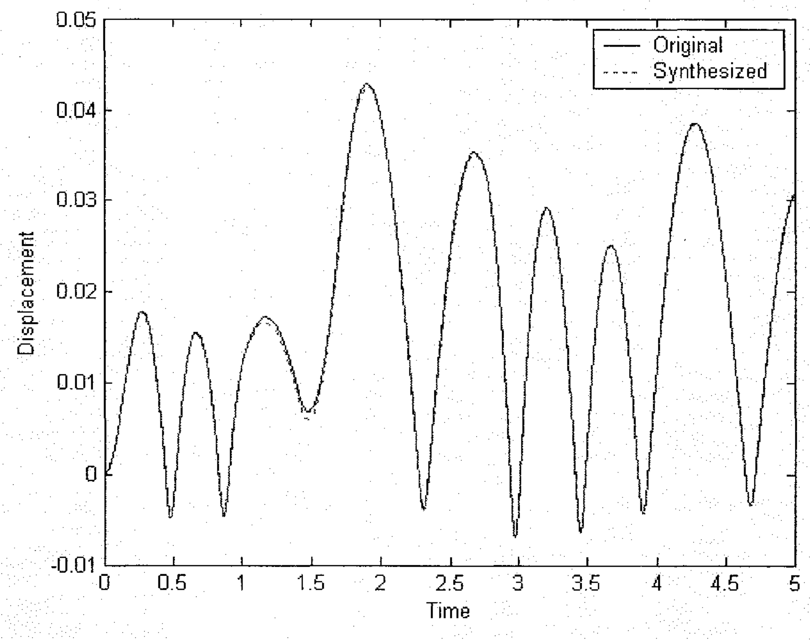


Figure 4-12. Nonlinear SDOF Example: Response of Original and Identified System to Different Forcing Function with 0.001 sampling period

The above exercise demonstrated how the accuracy of the identified model can be improved by using a smaller time-step. For the approximate eight and one-half cycles in Figure 4-12, the smaller time step equates to nearly 600 data samples in each oscillation. Although this level is achievable with today's data acquisition systems, it is excessive, so the remainder of the identification performed will be closer to 100 data samples in each oscillation. If increased accuracy is required when applying the methods addressed, a smaller time step is an option.

4.2 Nonlinear Single-Input/Single-Output with Corrupted Data

Up to this point only idealized input/output signals with no noise have been addressed, which is rarely the case in practice. Due to many different reasons, measurements are often corrupted. The method developed up to this point does not perform well when uncorrelated noise is superposed on the input and output signals. To illustrate this, simulated noise was used to corrupt the signals shown in Figure 4-5. Figure 4-13 schematically shows how this was performed. The noise is represented by η, and the subscript m represents the measured signal. The noise was constructed using a random signal with maximum and minimum values chosen to result in an 80 dB Signal-to-Noise Ratio (SNR), as calculated by [25]

$$\left(S/N\right) = 10\log_{10}\left(\sigma_s^2 / \sigma_n^2\right) . \qquad 4\text{-}11$$

Separate uncorrelated noise models were constructed for the input and output.

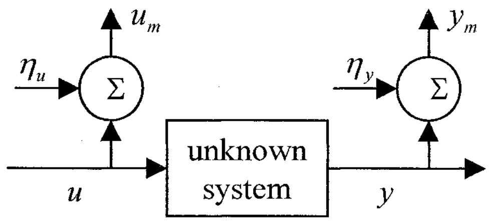


Figure 4-13. Illustration of Measurement Noise

The same parameters listed in Table 4-2 were used with the NL-ARX method developed and data corrupted as explained above. The identified parameters resulted in the synthesized output shown in Figure 4-14. Although the method successfully identified the higher stiffness for displacements less than zero, the general response does not match the data from the original system.

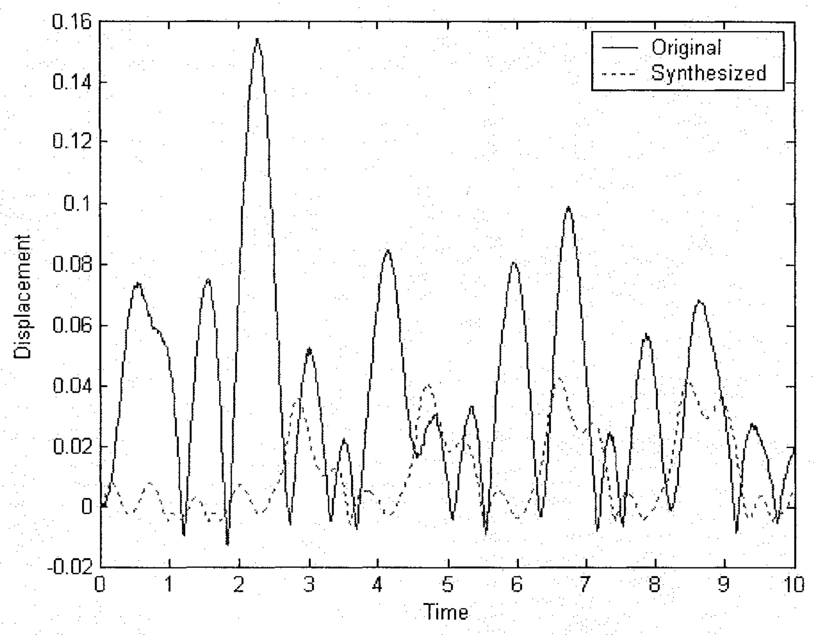


Figure 4-14. Nonlinear SDOF Example with Noise: Synthesized Response Compared to Original using NL-ARX

To reduce this effect, the ability to accommodate data with noise was needed. Therefore, an AR model of the noise was added to the algorithm. The identified noise model was then used to filter the input and output signals. This approach is referred to as Generalized Least Squares (GLS) [12, 26].

After the coefficients are found for Equation 4-9 using a least-squares method, the residuals can be found by solving

$$e = y - M\Psi . \qquad \text{4-12}$$

An AR model of the error can be constructed with the residuals,

$$e(n) = \sum_{l=1}^{R} c(l)e(n-l) . \qquad \text{4-13}$$

The coefficients for Equation 4-13 can be found using the method outlined in Equations 3-6 to 3-10. The coefficients found for Equation 4-13 can now be used as a filter for the input and output signals as

$$y_f(n) = \sum_{l=1}^{R} c(l) y(n-l) \text{ and} \qquad 4\text{-}14$$
$$u_f(n) = \sum_{l=1}^{R} c(l) u(n-l) \; .$$

The filtering influence of Equation 4-14 removes the noise content; however, it also linearly scales the data. The application of GLS in [26] is to linear systems, so this weighting phenomenon does not adversely affect the identification of the system. This is not the case when applying it to nonlinear systems. Therefore, a scale factor has to be applied to the filtered data. The scale factor is calculated by dividing the standard deviation (σ) of the original output data by that of the filtered,

$$scale = \frac{\sigma(y)}{\sigma(y_f)} \; . \qquad 4\text{-}15$$

Multiplying both the filtered input and output from Equation 4-14 by this scalar value restores the data to its original range,

$$y_f = scale * y_f \text{ and} \qquad 4\text{-}16$$
$$u_f = scale * u_f \; .$$

By recalculating the coefficients of Equation 4-9 using the filtered results from Equation 4-16, the accuracy of the fit is increased. Repeating the process until the solution converges further increases the accuracy of the fit. Convergence has been achieved when the results from the two most recent iterations generate the same residuals. Likeness of two residuals is calculated using

$$test = \frac{\left| e^T e_o - e_o^T e_o \right|}{e_o^T e_o} , \qquad 4\text{-}17$$

where the subscript "*o*" represents the residuals from the previous iteration step. In practice, the best results have been achieved by setting a tolerance on Equation 4-17 of less than 1.0×10^{-9}.

In addition to the identified AR filter, it is also beneficial to apply a pre-filter to both signals. The simple discrete filter

$$x_f^{n+1} = \left(x^{n+1} + x_f^n\right)/2 \qquad 4\text{-}18$$

has proven to be adequate in reducing the noise without masking the systems characteristics. The frequency response of the pre-filter is shown in Figure 4-15 for a sampling frequency of 100 Hz.

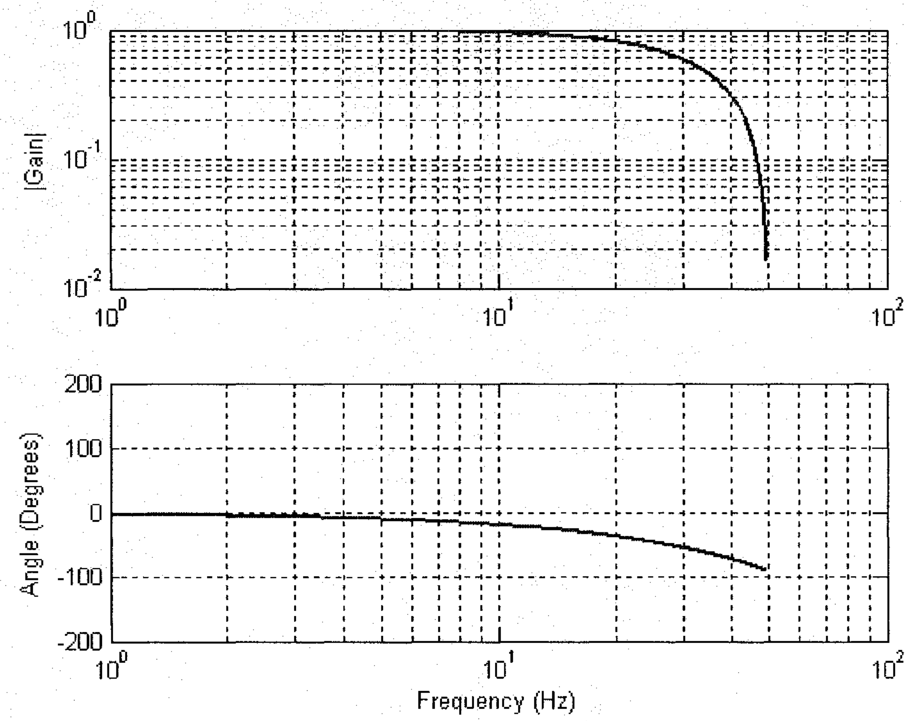


Figure 4-15. Bode Plot of Pre-Filter (sample frequency of 100 Hz)

The NL-ARX method outlined in Section 4.1 can be updated with both the pre-filter and GLS routine to generate a state-dependent NL-ARX-GLS model for SI/SO systems with corrupted data. The following outlines the steps for a NL-ARX-GLS algorithm. The main MATLAB routine (NLARXGLS.m) is contained in Appendix D.

1. Pre-filter the input and output signals with Equation 4-18
2. Choose model parameters: model order (P, Q & R), basis set(s) to be used, and number of vectors to be used in basis (walsh.m cheby.m and/or block.m from Appendix B)
3. Construct y_k and u_k vectors with the filtered data (pool.m)
4. Find the linear independent vectors in y_k and u_k (linindi.m)
5. Construct M matrix from the linear independent y_k and u_k vectors
6. Solve for the α and β coefficients using Equation 3-10
7. Find the residuals using Equation 4-12
8. Filter the input and output using GLS (gls.m, Appendix D)
9. Using Equation 4-17, check for convergence
10. If the algorithm has not converged and number of iterations is less than the maximum allowed, go to Step 3 and use the filtered data from Step 8 to create the vector pool, otherwise go to Step 11
11. Construct the state-dependent NL-ARX coefficients a and b (assemble.m)
12. Truncate the coefficients above and below range of identification (trim.m)
13. Reconstruct y to verify model (timeh.m)

4.2.1 Example Nonlinear Single-Input/Single-Output with Corrupted Data

To demonstrate the ability of the NL-ARX-GLS method, it was applied to the corrupt data from Section 4.2. The parameters used for the identification are listed in Table 4-3. Chebyshev Polynomials were used instead of Walsh functions because identifications performed during this research revealed they often work well with noisy data.

Table 4-3. NL-ARX-GLS Parameters used in SDOF Nonlinear System Identification with 80 dB Noise on Input and Output

Parameter	Value
P	2
Q	1
R	8
Number of Chebyshev Polynomials	19
Length of Block Vectors	1000
GLS Tolerance	1.0E-09
Maximum Iterations	100

As with previous examples, to demonstrate the validity of the identified model, the response was synthesized. It can be seen in Figure 4-16 that the synthesized response does not match as close as the clean data (Figure 4-8), but much better than the attempt of matching corrupted data with the NL-ARX method (Figure 4-14).

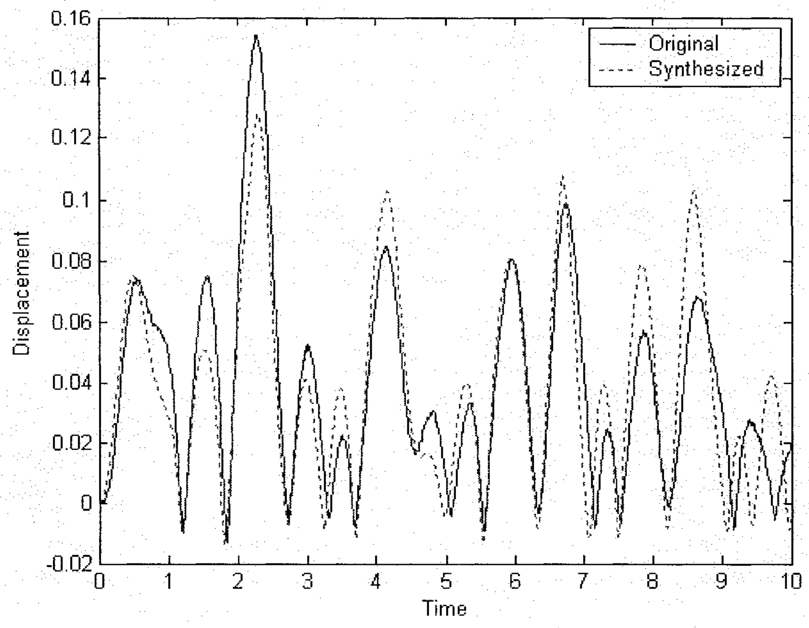


Figure 4-16. Nonlinear SDOF Example with Noise: Synthesized Response Compared to Original using NL-ARX-GLS

4.3 Nonlinear Single-Input/Multiple-Output with Corrupted Data

Unlike linear systems, some nonlinear systems dictate a level of spatial resolution of outputs to adequately perform system identification. For instance, if a multiple degree of freedom system contains a softening spring, a single input and output would not provide enough information to accurately identify the nonlinear characteristics; however, a measurement of both sides of the spring would. For this reason, a multiple output method more accurately captures the characteristic of nonlinear systems.

The method developed for the identification of SI/SO nonlinear systems with input/output signals corrupted with noise (NL-ARX-GLS) was expanded to SI/MO systems. This was accomplished by using Equation **Error! Reference source not found.** for each output and adding to it terms to represent nonlinear influences from other outputs. Coefficients, which are a function of the difference between outputs, are applied to the quantity resulting from differencing the output of interest with another output. It is recognized that this will not represent all possible nonlinear combinations, but since this research is focused on mechanical systems, it should suffice. For a more general form, the influence of each output as well as products can be added to the model.

With the above-mentioned conditions, the SI/SO model shown in Equation **Error! Reference source not found.** is transformed to

$$\begin{aligned} y_h(n) &= \sum_{i=1}^{P} a(i, y_h(n-1)) y_h(n-i) + \sum_{j=0}^{Q} b(j, y_h(n-1)) u(n-j) \\ &+ \sum_{l=1}^{m_{out}-1} \left[\sum_{i=1}^{P} a(l*P+i, y_h(n-i) - y_r(n-i))(y_h(n-i) - y_r(n-i)) \right] + e(n), \end{aligned} \tag{4-19}$$

where $r = \begin{cases} i & \text{if } l < i \\ i+1 & \text{if } l \geq i \end{cases}$.

The implementation of Equation 4-19 requires minor changes to the algorithm developed in Section 4.2. The algorithm flow remains generally the same; however, some of the routines do require modification. Table 4-4 shows a list of the routines for the SI/SO application that require modifications. The updated routines can be found in Appendix E.

Table 4-4. Routines Requiring Modification for SI/MO Identification

SI/SO Routine Name	SI/MO Routine Name
nlarxgls	mnlarxgls
pool	mpool
gls	mgls
assemble	massemble
trim	mtrim
timeh	mtimeh

Routine nlarxgls.m had to be modified to accommodate a reference channel, the channel of interest, as well as produce vectors of the reference channel differenced with the balance of the channels. The portion of the routine that generates the basis was also changed, to allow the different outputs to have different basis sets. Due to it being the main routine, the routines being called had to be updated from SI/SO to the SI/MO routine names.

The vector pool generation routine was modified to produce mpool.m by adding another dimension to the input and output matrices to accommodate multiple outputs. The order of the output and input fit, P and Q, were also vectorized to allow for different order models. The routines mgls.m, massemble.m, mtrim.m, and mtimeh.m are very similar to their SI/SO counterparts with an additional dimension to their outputs to represent the multiple outputs.

The best results have been achieved with the outlined algorithm by performing the identification on each output individually; using the other outputs measured data during the identification. The process is similar to the method used to find the optimum number of basis functions for SI/SO systems, with the number of basis functions for the output being identified, as well as for the coefficients being the variables. The number of basis functions for the input is kept the same as that of the output of interest.

4.3.1 Example of Nonlinear Single-Input/Multiple-Output System

To demonstrate the SI/MO NL-ARX-GLS identification algorithm, a three-DOF system with two nonlinearities was created (Figure 4-17). The two nonlinearities consist of a cubic spring (k_3) and a hard stop (k_{hs}); the values of the parameters are shown in Table 4-5. The system was excited with the input

$$u = 100 * \sin(10.1 \cdot t) + 100 * \sin(3.2 \cdot t) \quad . \tag{4-20}$$

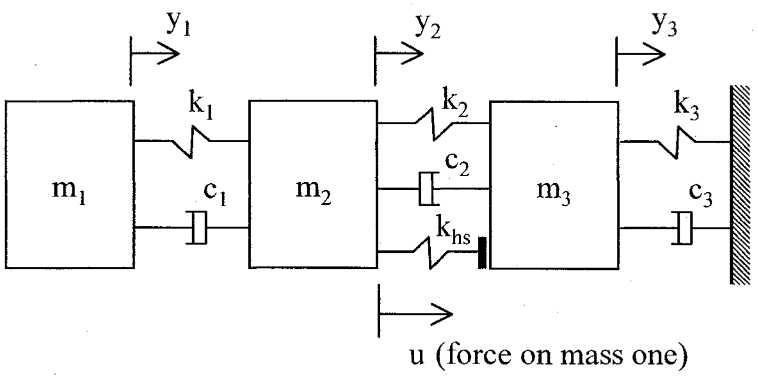


Figure 4-17. MDOF Nonlinear System with Single Input

Table 4-5. MDOF Nonlinear System Parameters

Parameter	Value
m_1	1
c_1	5
k_1	10
m_2	1
c_2	2
k_2	1
k_{hs} $(y_2 - y_3 > 0)$	2000
m_3	1
c_3	4
k_3 (cubic spring)	10

As with the example in Section 4.2, noise was added to the outputs and the input, all were on the order of 80 dB SNR.

The identification algorithm was first applied to y_1 with the parameters listed in Table 4-6. The first four Chebyshev functions were used for both the basis set for y_1 and $y_1 - y_2$. The coefficients for $y_1 - y_3$ were assumed to be constant. Figure 4-18 contains the results from the identified model. It should be noted that when creating the results shown in Figure 4-18, only y_1 was synthesized, the original values for y_2 and y_3 were used.

Table 4-6. SI/MO NL-ARX-GLS Parameters used to Identify Output 1

Parameter	**Value**
P	2
Q	1
R	8
Number of Chebyshev Polynomials in Basis for y_1	4
Number of Chebyshev Polynomials in Basis for $y_1 - y_2$	4
Coefficients for $y_1 - y_3$ Constant	N/A
Length of Basis Vectors	1000
GLS Tolerance	1.0E-09
Maximum Iterations	100

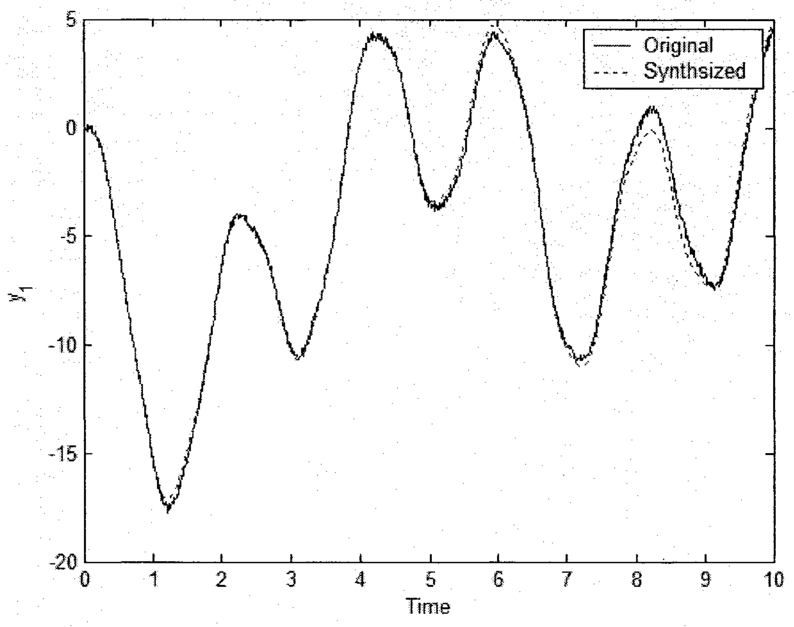


Figure 4-18. SI/MO Example with Noise: Synthesized Results Compared to Original for y_1 using the Original y_2 and y_3 Results

The identification algorithm was then applied to y_2 with the parameters listed in Table 4-7. The best model was identified using 14 Block Pulse functions for y_2, four Chebyshev Polynomials for $y_2 - y_1$ and 13 Block Pulse functions for the $y_2 - y_3$ basis sets. Figure 4-19 contains the results from the identified model and as with y_1, the results shown are for only y_2 being synthesized, the original values for y_1 and y_3 were used.

Table 4-7. SI/MO NL-ARX-GLS Parameters used to Identify Output 2

Parameter	**Value**
P	2
Q	1
R	8
Number of Block Pulse Functions in Basis for y_2	14
Number of Chebyshev Polynomials in Basis for $y_2 - y_1$	4
Number of Block Pulse Functions in Basis for $y_2 - y_3$	13
Length of Basis Vectors	1000
GLS Tolerance	1.0E-09
Maximum Iterations	100

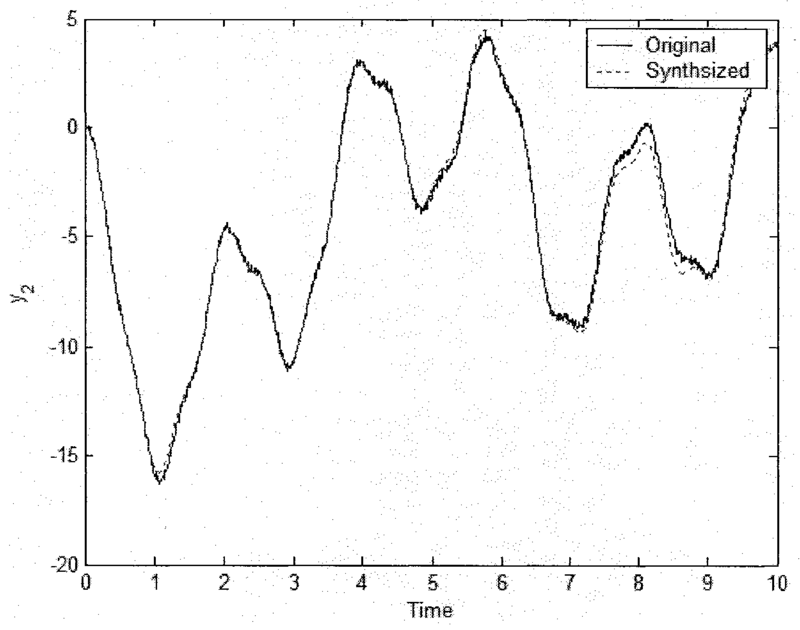


Figure 4-19. SI/MO Example with Noise: Synthesized Results Compared to Original for y_2 using the Original y_1 and y_3 Results

Finally, the identification algorithm was applied to y_3 with the parameters listed in Table 4-8. The best model was identified using four Chebyshev Polynomials for y_3, four Block Pulse Functions for $y_3 - y_1$ and 15 Block Pulse functions for the $y_3 - y_2$ basis sets. Figure 4-20 contains the results from the identified model and as with the other two outputs, the results shown are for only y_3 being synthesized.

Table 4-8. SI/MO NL-ARX-GLS Parameters used to Identify Output 3

Parameter	Value
P	2
Q	1
R	8
Number of Chebyshev Polynomials in Basis for y_3	4
Number of Block Pulse Functions in Basis for $y_3 - y_1$	4
Number of Block Pulse Functions in Basis for $y_3 - y_2$	15
Length of Basis Vectors	1000
GLS Tolerance	1.0E-09
Maximum Iterations	100

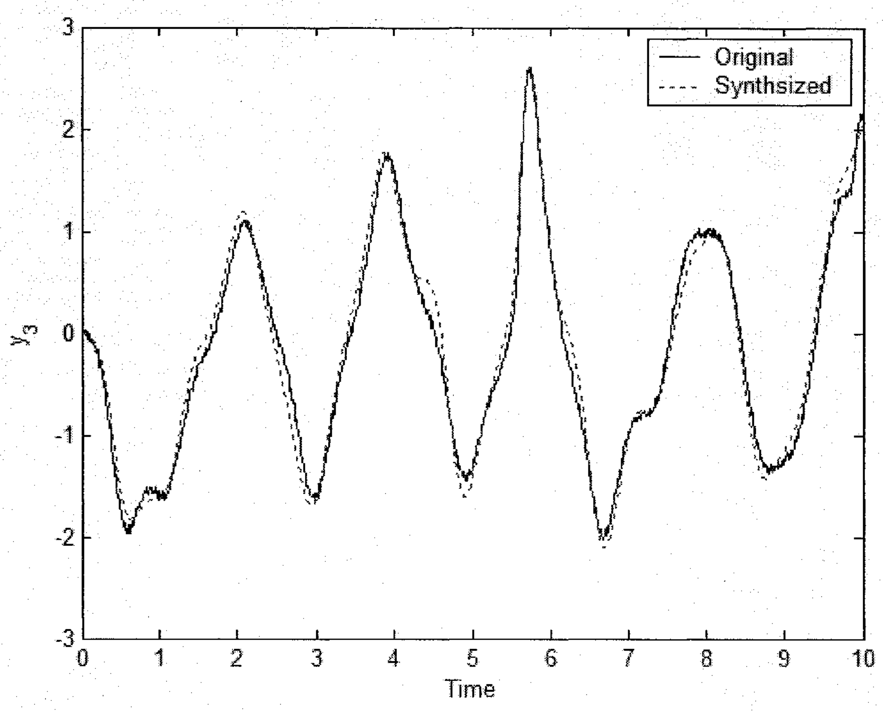


Figure 4-20. SI/MO Example with Noise: Synthesized Results Compared to Original for y_3 using the Original y_1 and y_2 Results

Figure 4-21 to Figure 4-32 contain plots of the coefficients found for all three outputs. As expected, the coefficients for y_1 (Figure 4-21 to Figure 4-23) are much more constant than those for the other outputs, due to the forces on y_1 having linear properties.

Applying the forcing function in Equation 4-20 to the identified model, Equation 4-19 with the coefficients shown in Figure 4-21 to Figure 4-32, outputs y_1, y_2 and y_3 were synthesized simultaneously. The synthesized results matched the original quite well, as can be seen in Figure 4-33.

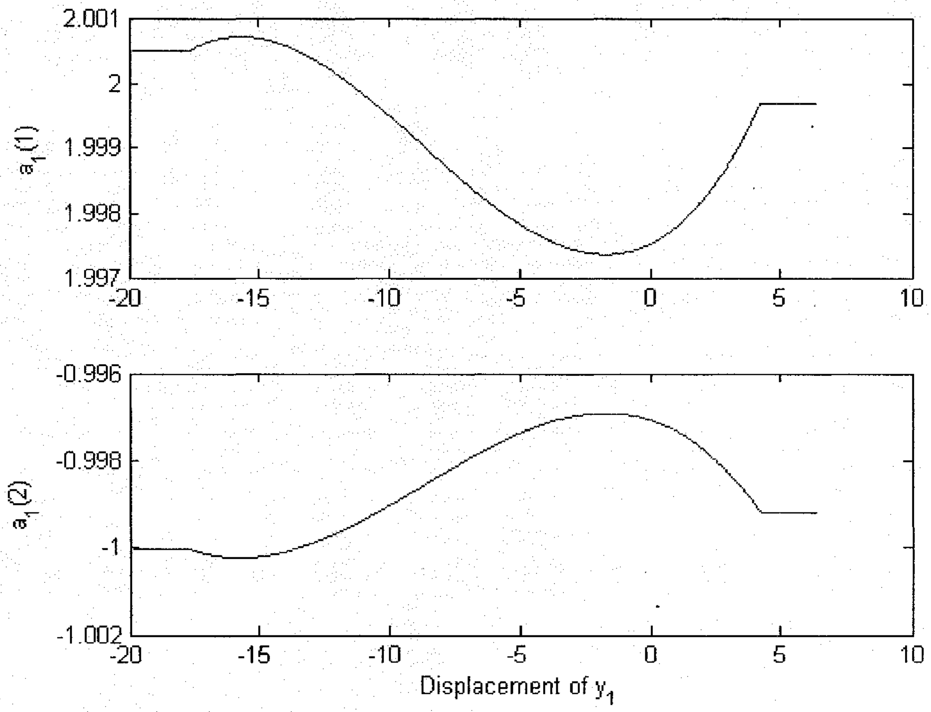


Figure 4-21. SI/MO Example with Noise: y_1 NL-ARX-GLS "*a*" Coefficients for y_1 Terms

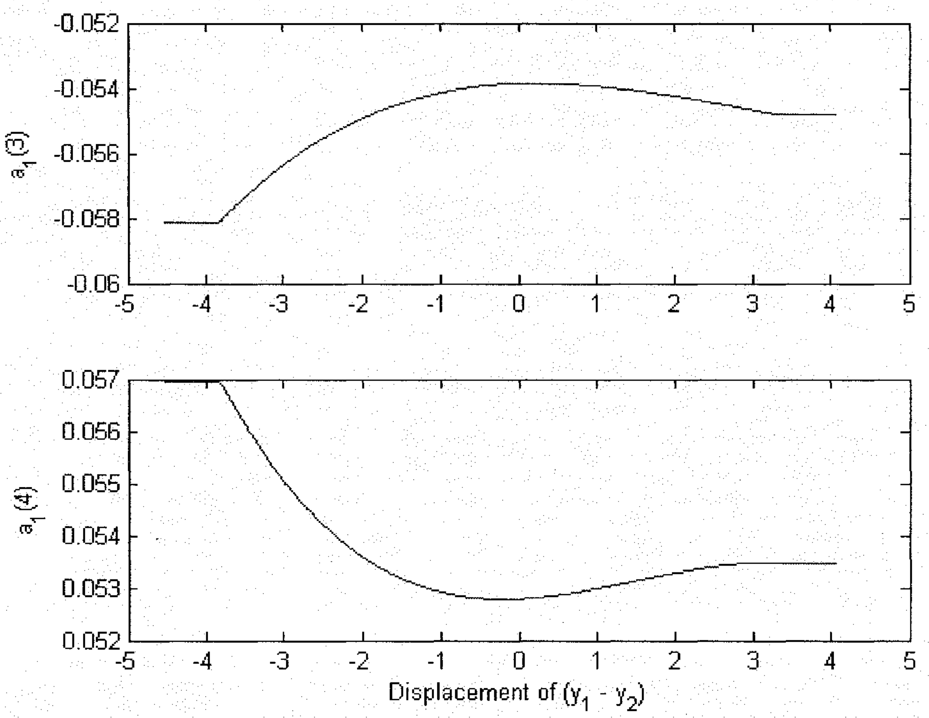


Figure 4-22. SI/MO Example with Noise: y_1 NL-ARX-GLS "*a*" Coefficients for $(y_1 - y_2)$ Terms

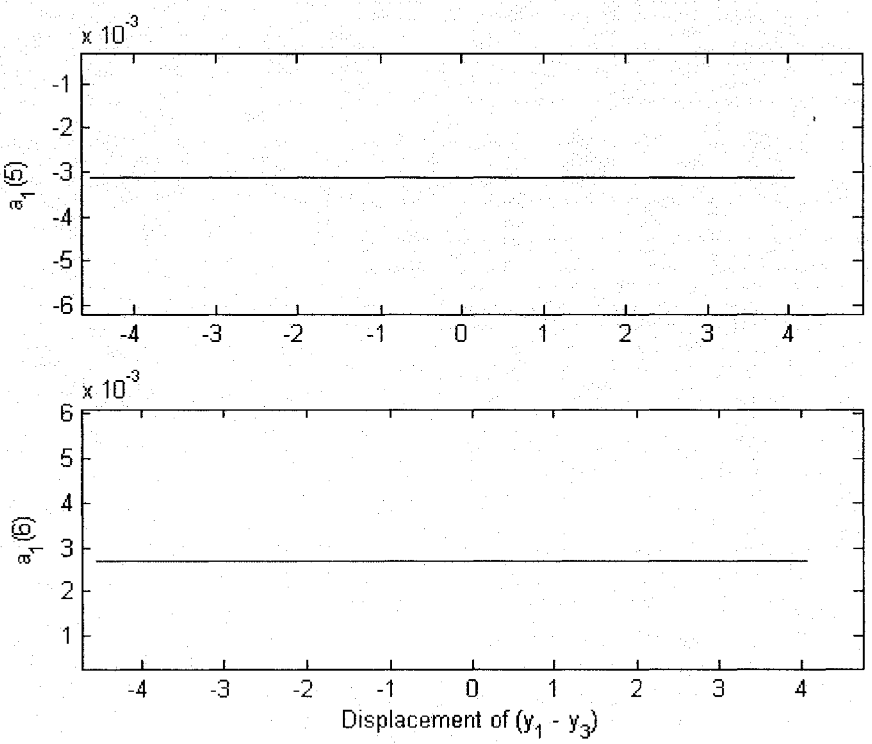


Figure 4-23. SI/MO Example with Noise: y_1 NL-ARX-GLS "*a*" Coefficients for $(y_1 - y_3)$ Terms

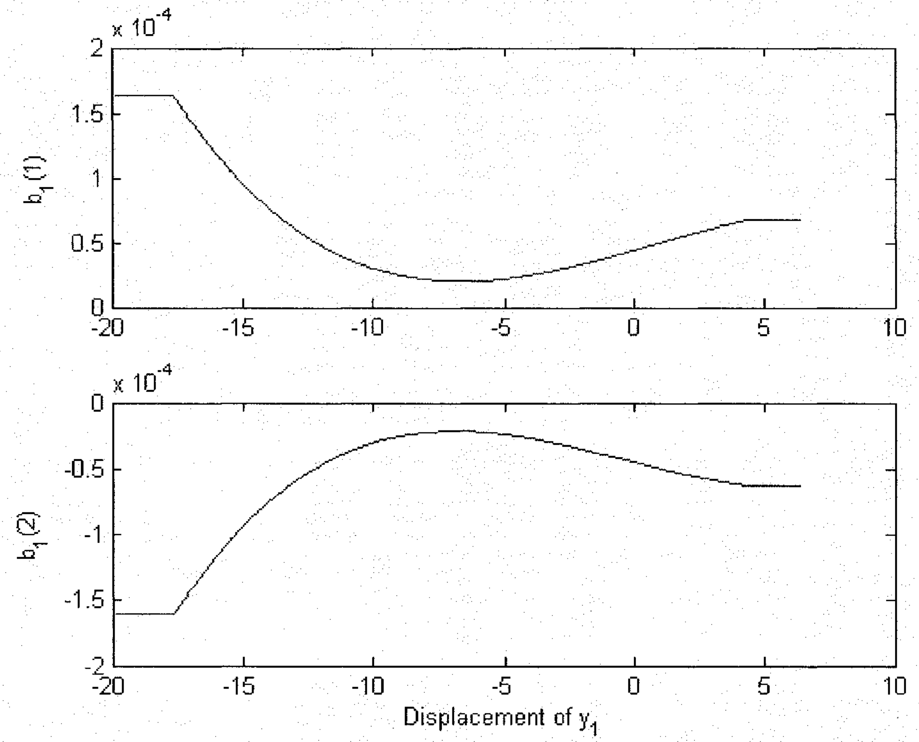


Figure 4-24. SI/MO Example with Noise: y_1 NL-ARX-GLS "*b*" Coefficients

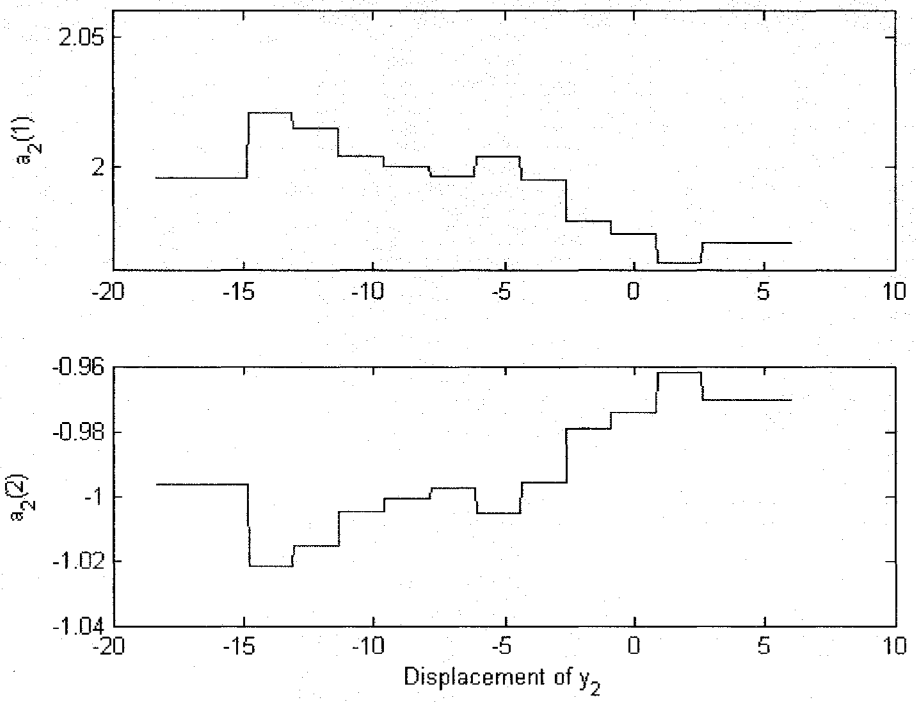


Figure 4-25. SI/MO Example with Noise: y_2 NL-ARX-GLS "*a*" Coefficients for y_2 Terms

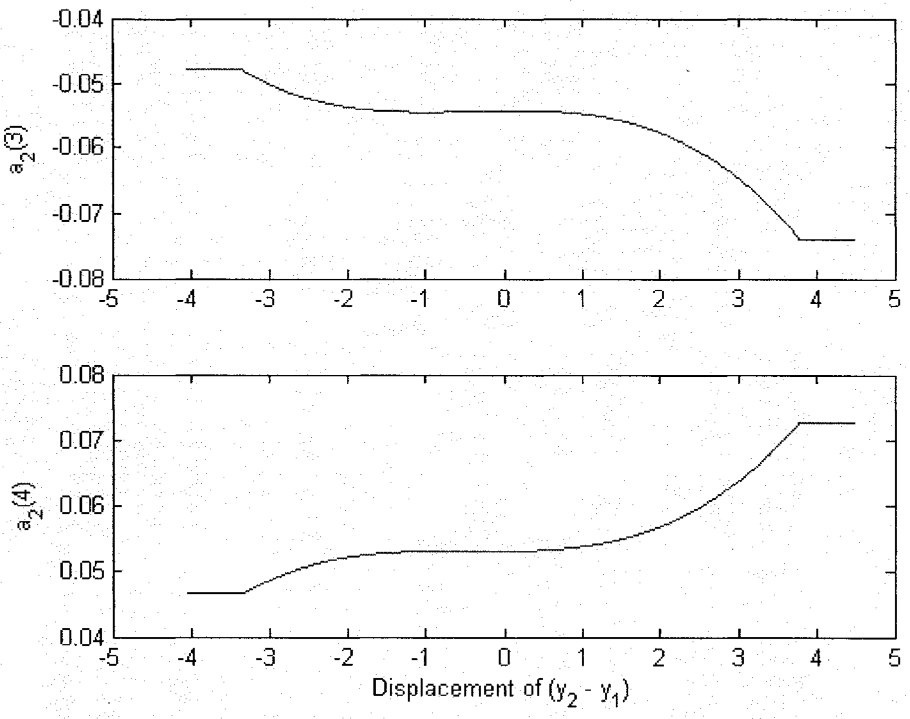


Figure 4-26. SI/MO Example with Noise: y_2 NL-ARX-GLS "*a*" Coefficients for $(y_2 - y_1)$ Terms

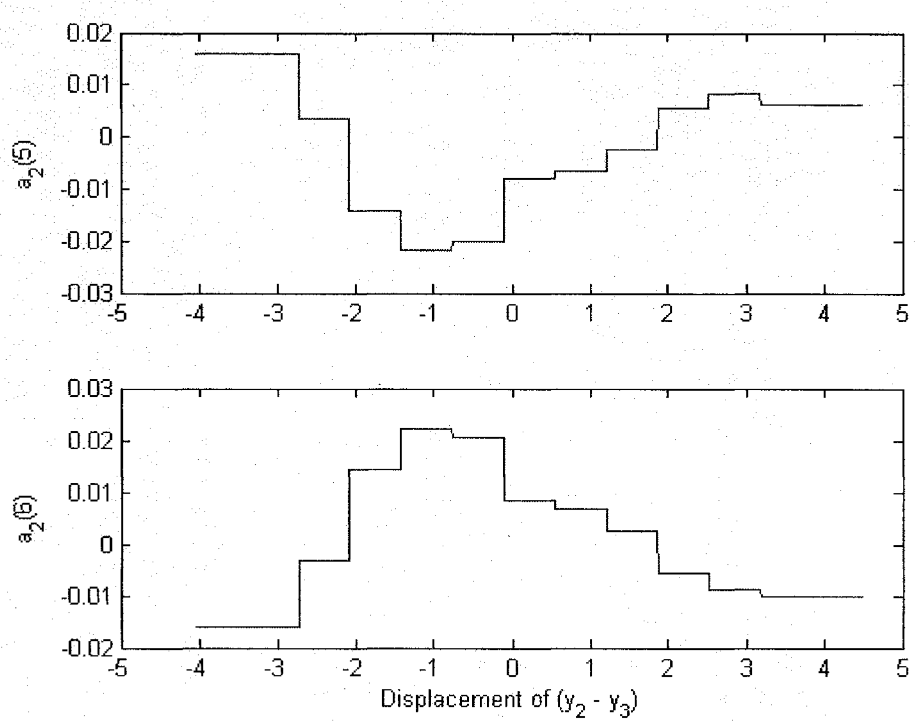


Figure 4-27. SI/MO Example with Noise: y_2 NL-ARX-GLS "*a*" Coefficients for $(y_2 - y_3)$ Terms

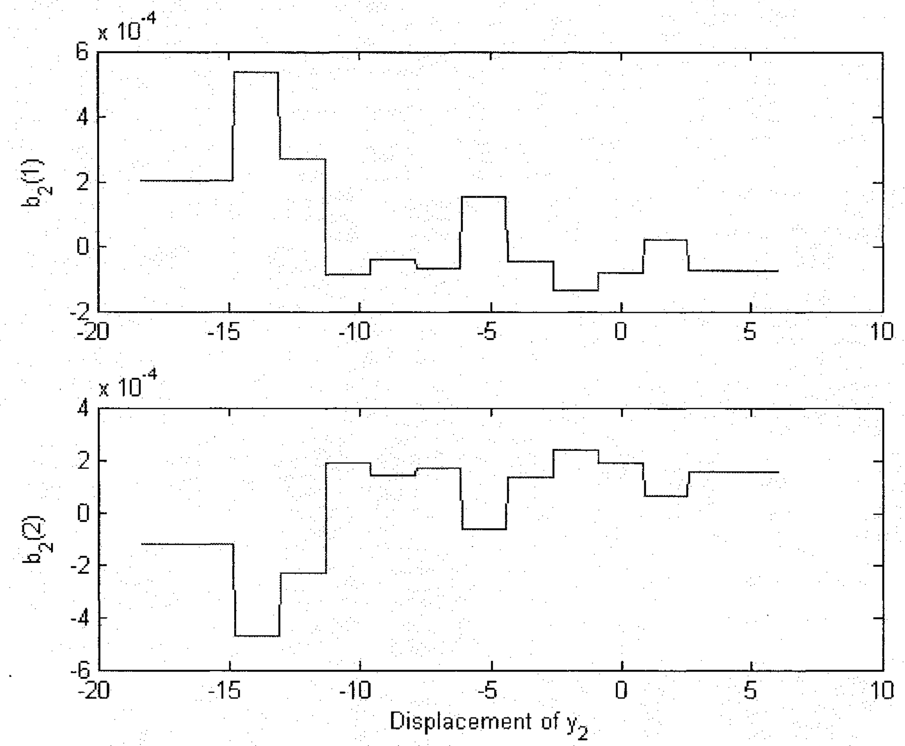


Figure 4-28. SI/MO Example with Noise: y_2 NL-ARX-GLS "*b*" Coefficients

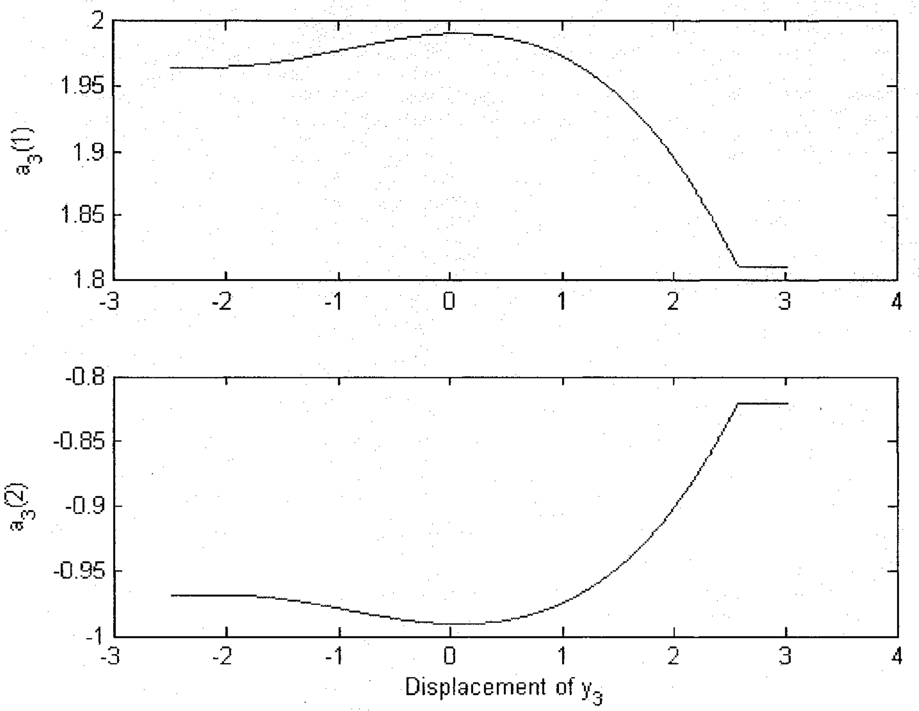


Figure 4-29. SI/MO Example with Noise: y_3 NL-ARX-GLS "*a*" Coefficients for y_3 Terms

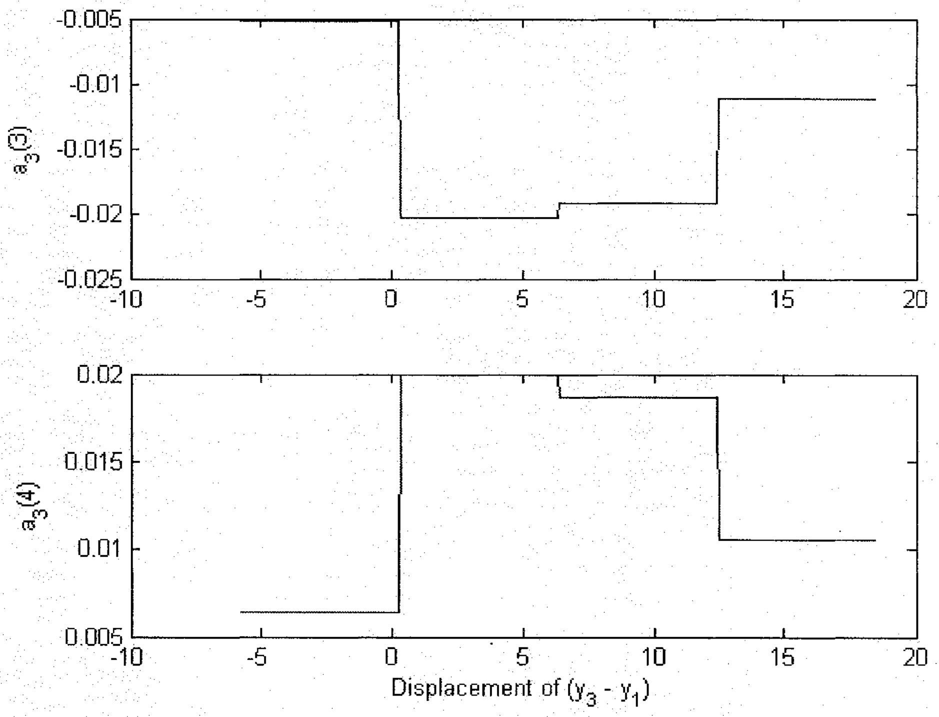


Figure 4-30. SI/MO Example with Noise: y_3 NL-ARX-GLS "*a*" Coefficients for ($y_3 - y_1$) Terms

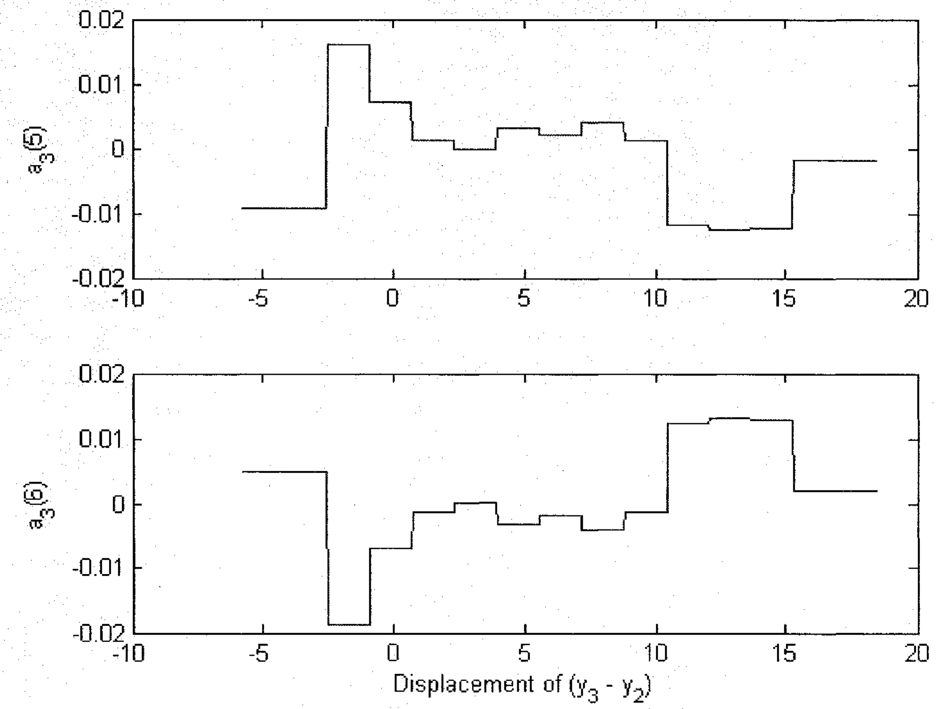


Figure 4-31. SI/MO Example with Noise: y_3 NL-ARX-GLS "*a*" Coefficients for ($y_3 - y_2$) Terms

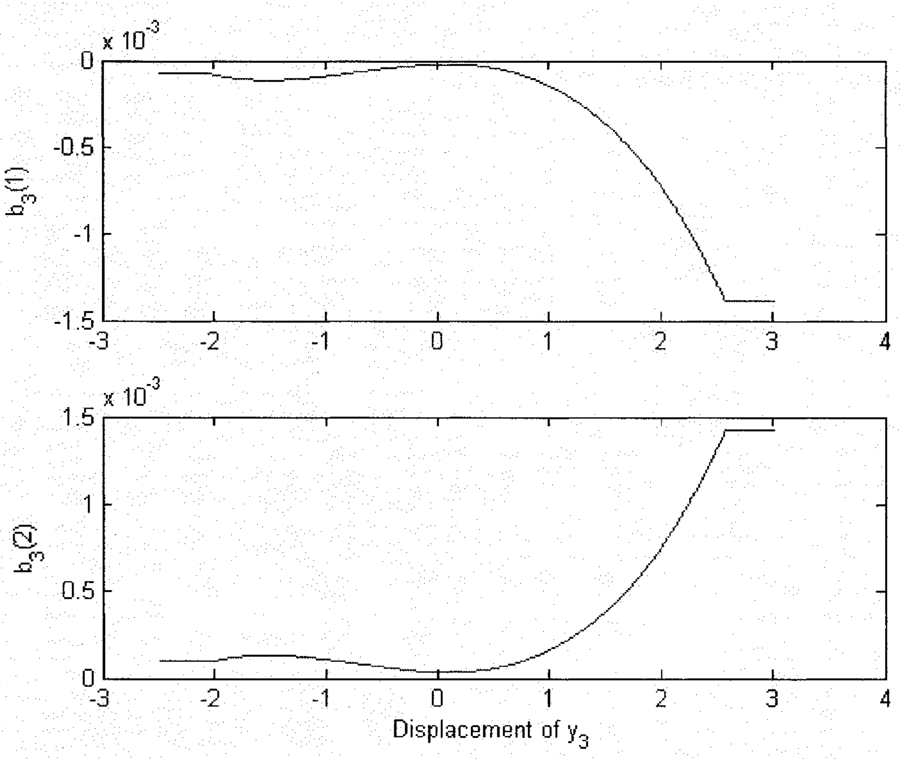


Figure 4-32. SI/MO Example with Noise: y_3 NL-ARX-GLS "*b*" Coefficients

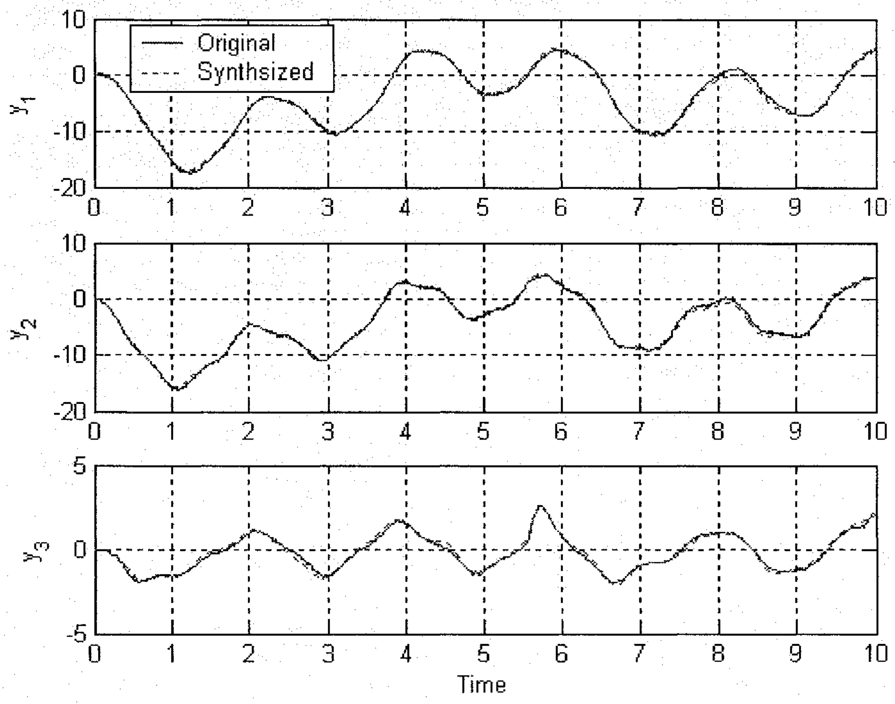


Figure 4-33. SI/MO Example with Noise: Full Synthesized Response Compared to Original using NL-ARX-GLS

To demonstrate the validity of the identified model, both the original system and the model constructed from the data were driven by an input other than the one used for the identification process, see Figure 4-34. Comparison of these results, shown in Figure 4-35, shows the accuracy of the identified model.

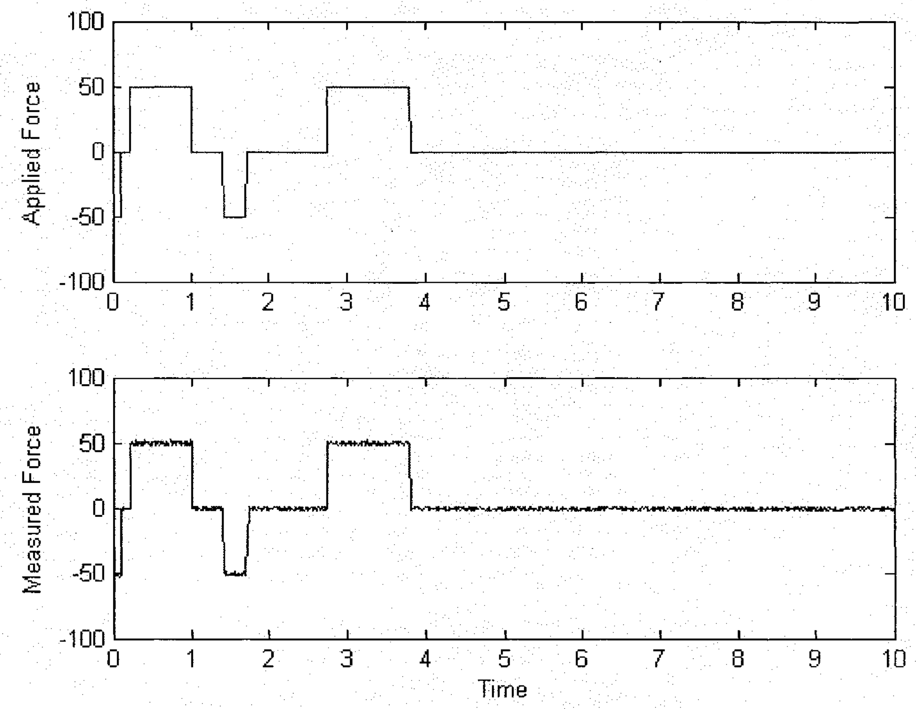


Figure 4-34. SI/MO Example with Noise: Measured and Applied Force to Identified Model

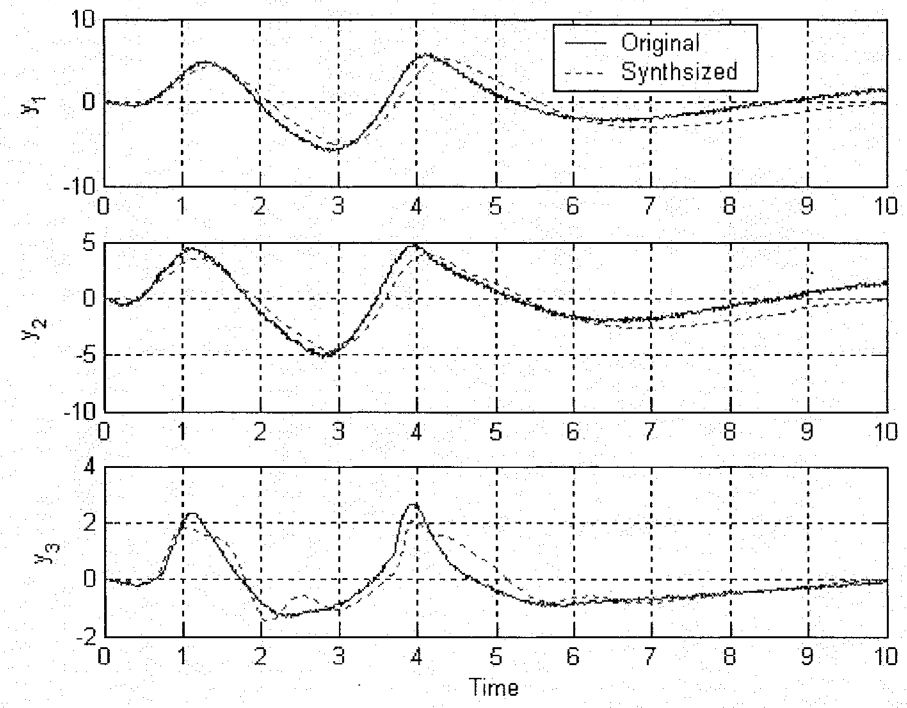


Figure 4-35. SI/MO Example with Noise: Response of Original and Identified System to Different Forcing Function

Chapter 5 - Application of NL-ARX-GLS Algorithm to Damage Detection

The ability of the developed SI/MO NL-ARX-GLS method to create accurate nonlinear models of systems has been shown. To complete the goal of this research, it was necessary to apply the method to damage detection of nonlinear systems.

Damage detection and isolation methods for nonlinear systems, such as that developed by Zhang [27], rely on the system being represented by differential-algebraic equations. However, state-dependent coefficients representing hard nonlinearities are not differential. Therefore, a novel comparison approach was developed.

The approach compares the coefficients of the undamaged model to those of models computed at later inspection times. Since variations of the coefficients occur even for the same system, a stochastic approach was taken. This involved performing multiple identifications on the undamaged model at different time windows using the same input and output orders. The mean and standard deviation of each coefficient as a function of its state define the undamaged model coefficient bounds.

Multiple identifications were then performed later at the inspection time. Mean values for the coefficients were calculated and compared to the statistical bounds of the undamaged model. Coefficients that are outside the bounds indicate areas of possible damage, the larger the deviation, the greater the possibility of damage. Due to the coefficients being a function of the system states, not only is the damaged output identified, but also the range of operation where the suspect damage occurs is predicted.

5.1.1 Example Nonlinear Single-Input/Multiple-Output Damage Detection

To demonstrate the damage detection capability, data from a series of tests performed on an eight degree-of-freedom system at the Los Alamos National Laboratory in New Mexico were used [28]. Digital data from the tests were obtained from [29].

The system was comprised of eight translating masses connected by springs, see Figure 5-1. Each mass was an aluminum disc with a Teflon lined center hole, with the mass attached to shaker having the mass of 559.3 grams and the remaining seven 419.4 grams. The masses slid on a highly polished steel rod that was lubricated to minimize friction. The masses were fastened together with coil springs with a stiffness of 56.7 kN/m. The system damping was low and caused primarily by Coulomb friction.

The excitation force was applied to m_1 and the acceleration responses of all masses were recorded during damage identification tests. The excitation during the testing was random. The specifications for the data acquisition are summarized in Table 5-1.

Table 5-1. Data Recording Parameters used for Eight-DOF System [28]

Time step	0.001953 sec.
Sampling rate	512 Hz
Time period	8 sec.
Number of data points	4096

The undamaged configuration of the system was when all the springs were identical with a linear spring constant. Placing a bumper between two adjacent masses so that the movement of one mass is limited relative to the other mass simulated nonlinear damage, see Figure 5-2. The level of damage was controlled by changing the amount of relative motion permitted before contact.

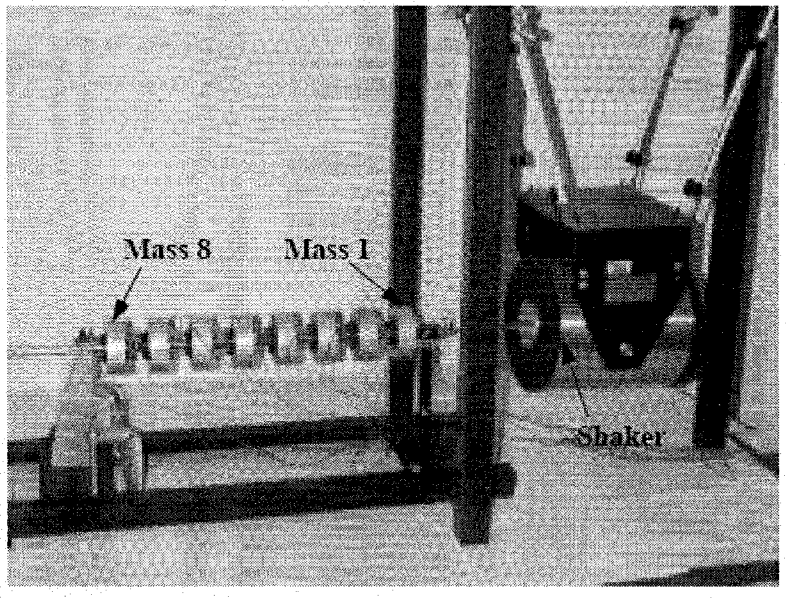


Figure 5-1. An Eight-DOF System Attached to a Shaker with Accelerometers Mounted on Each Mass [28]

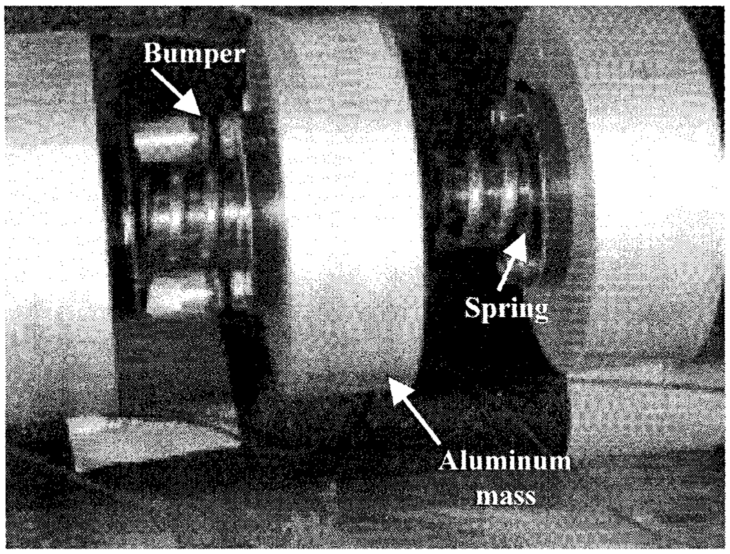


Figure 5-2. A Typical Bumper used to Simulate Nonlinear Damage [28]

Three test configurations were used in this study: Undamaged; Damage-1, bumper between masses 5 and 6 with 0.2 millimeter clearance and; Damage-2, bumper between masses 5 and 6 with 0.4 millimeter clearance. These were chosen to determine if the NL-ARX-GLS method could distinguish the difference between a linear and nonlinear system, as well as two similar nonlinear configurations. Data for five tests in each configuration at the same level of random excitation were used.

To extract an accurate nonlinear model, the displacement of each mass was needed, since the nonlinearity of the system was a function of displacement. The sampling frequency of the accelerometer was not high enough to accurately extract the displacement of each mass. So it was decided to proceed using the accelerometer data to investigate the robustness of the damage detection method.

To further exercise the ability of the SI/MO NL-ARX-GLS identification method, a purposely too high of order was chosen for the output fit, as well as nonlinearities were assumed between every output, not just adjacent masses. Table 5-2 contains the identification parameters used for every output channel.

Table 5-2. SI/MO NL-ARX-GLS Parameters used for Eight-DOF System

Parameter	**Value**
P	3
Q	1
R	8
Number of Chebyshev Polynomials in Basis for all channels and differences	8
Length of Basis Vectors	1000
GLS Tolerance	1.0E-09
Maximum Iterations	20

The identification was performed on all five tests for the three test configurations being studied. The mean and standard deviation of the Undamaged configuration was calculated; in general it would be desirable to have more than five averages to have a more statistically significant bound, but only five were available for each case. The mean of the Damage-1 configuration was also determined. The Damage-1 results were then compared to the bound of the Undamaged configuration.

Figure 5-3 contains the comparison of the coefficients for the difference between consecutive masses in Equation 4-19. The numerical values plotted correspond to how well the Damage-1 coefficients match that of Undamaged. A value of zero equates to the Damage-1 coefficient being within one standard deviation of the Undamaged model; one is between one and two standard deviations; two is between two and three standard deviations, and so on.

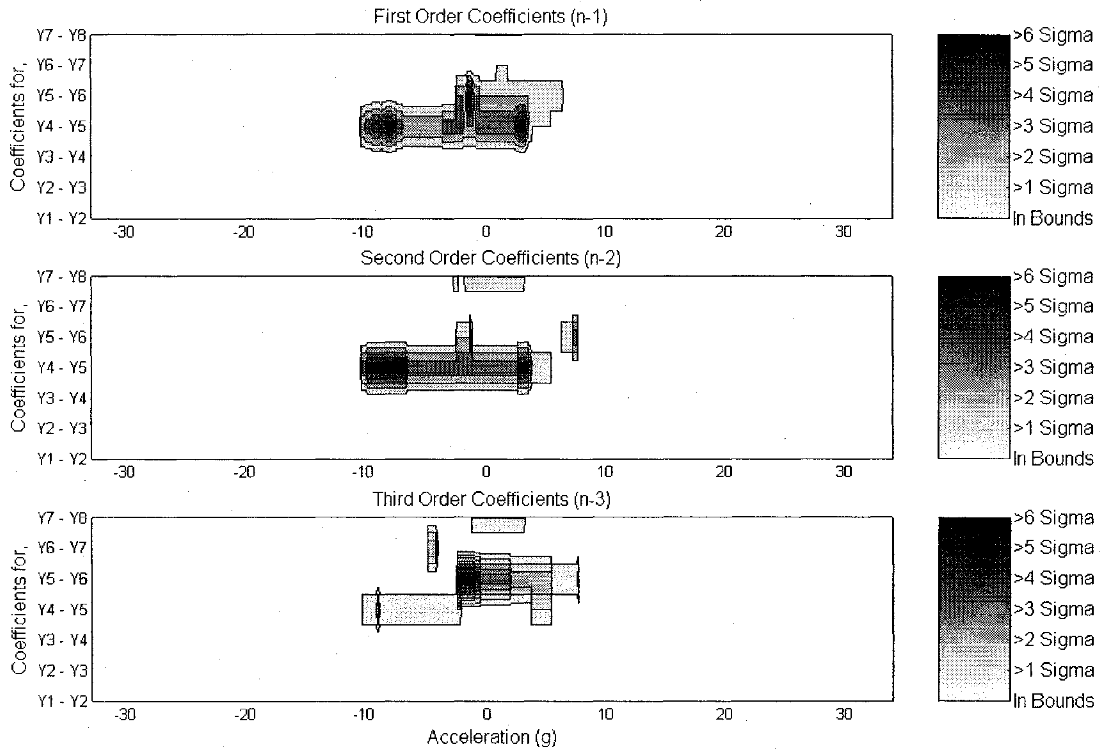


Figure 5-3. Damage-1 Compared to Undamaged Difference Coefficients

Figure 5-3 shows a consistent large difference for the coefficients to the y_5–y_6 term in Equation 4-19. It also shows differences for the y_4–y_5 coefficients. Using the data from Figure 5-3, one would have a clear indication that something has changed at or near output Channels 5 and 6, requiring a physical inspection.

Figure 5-3 shows that the SI/MO NL-ARX-GLS identification method is capable of identifying possible areas of damage for cases when the damage causes a linear system to become nonlinear. The next case will demonstrate the capabilities when a nonlinear system's parameters change.

The identified coefficients from the Damage-1 and Damage-2 tests were compared in the same way that the Undamaged and Damage-1 data were evaluated. Figure 5-4 reveals that the developed identification method recognized the change in the y_5–y_6 coefficients for Equation 4-19, indicating a possible area of damage.

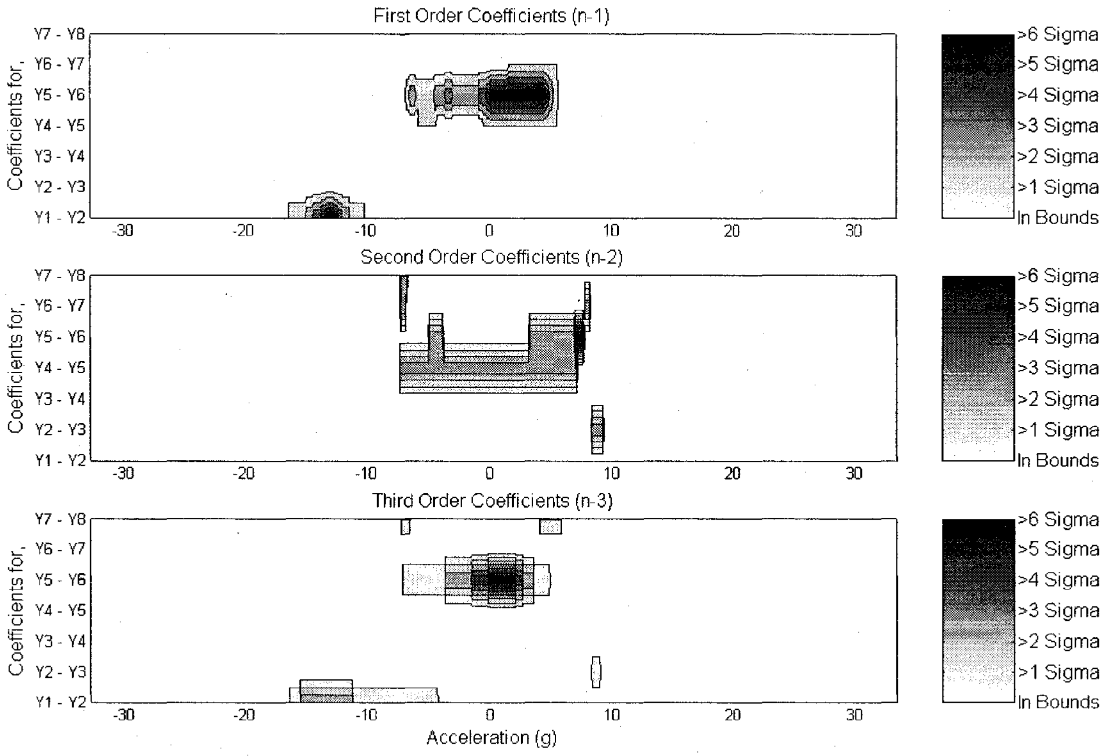


Figure 5-4. Damage-2 Compared to Damage-1 Difference Coefficients

As mention before, the data from the Los Alamos eight-DOF tests was not ideal. This is due to the nonlinearity being a function of displacement and the temporal resolution of the accelerometer data not being fine enough to extract displacement. To demonstrate how the method would perform with data more suited for identifying nonlinear behavior, which is a function of displacement, simulated data sets were generated. The model used for the simulation had parameters similar to that of the Los Alamos eight-DOF test rig. Five tests at similar configuration to Undamaged, Damage-1 and Damage-2 were generated using random forcing functions at levels producing similar accelerations to the Los Alamos tests. The displacement of each mass was recovered at 10,000 Hz. These simulated tests will be referred to as Undamaged-s, Damage-1s and Damage-2s.

The identification parameters were also changed to take advantage of the methods capability of identifying hard nonlinear characteristics. The earlier identification was performed with Chebyshev Polynomials, which is best for smooth nonlinear behavior; this was done to show the methods insensitivity to incorrect identification parameters. However, for the simulated case, the assumption is made that it is required to detect changes of the nonlinearity on the order of tenths of millimeters. Therefore, 50 block functions were used for the basis set. The order of the auto-regressive portion of the fit was lowered from three to two and the number of points in the basis vector was reduced from 1000 to 100. Using 1000 points would have added to the resolution, but combining the large number of points with 50 basis vectors would have led to excessive execution times. Table 5-3 contains a complete list of the identification parameters used.

Table 5-3. SI/MO NL-ARX-GLS Parameters used for Simulated Eight-DOF System

Parameter	**Value**
P	2
Q	1
R	8
Number of Block Function in Basis set for all channels and differences	50
Length of Basis Vectors	100
GLS Tolerance	1.0E-09
Maximum Iterations	20

A nominal amount of noise was added to the input and outputs of the system. An interesting characteristic of the method was discovered while performing the identification on the simulated data. The preliminary simulations were executed without synthesized noise, the data from these cases resulted erratic and unrealistic coefficients.

The source of the problem was determined to be that the M matrix (Equation 4-7) was near singular when uncorrelated noise was absent for systems with a larger number of outputs (approximately greater than five). The QR factorization and rank routines from MATLAB were generating a set of vectors that were not as linear independent as needed. Employing the slower, but more rigorous method used by Zou [19], it was possible to generate accurate coefficients with noise-free analytical data. Zou builds the M matrix vector by vector; for example, using $y_0(\boldsymbol{n}-1)$ to fit $y_1(\boldsymbol{n}-1)$ by least squares and calculate the residual of the fit. If the residual is larger than a preset threshold, then $y_1(\boldsymbol{n}-1)$ is considered to be linear independent of $y_0(\boldsymbol{n}-1)$ and the two are combined to begin the formation of the M matrix. The process is then repeated with the remaining vectors. Vectors are added to the *M* matrix if it is not possible to form a linear combination with the current *M* matrix using the residual as a test.

Although this method works, it is much slower than using QR factorization and greatly increases execution time for large systems. It is recommended to use the QR factorization in lieu of Zou's method. The requirement for noise should not be a problem when using actual test data, but should be kept in mind when applying it to idealized data. Although not utilized during the course of this research, a recursive least-squares method, such as that in Reference [30], could also be applied to find the coefficients.

Statistical evaluation of the SI/MO NL-ARX-GLS identification results successfully showed a changed in the coefficients near zero relative displacement when the nonlinearity was introduced to the system. Figure 5-5 shows a large change in coefficients between the Undamaged-s and Damage-1s configuration for y_5–y_6 near zero. It also incorrectly identifies y_4–y_5 as a possible damaged area. False identification between adjacent masses to the damaged area was also indicated in Figure 5-3 for the Undamaged to Damge-1 evaluation of the Los Alamos test data.

The statistical comparison of the simulated Damage-2s to Damage-1s configuration shows a clear indication of possible damage between masses 5 and 6, near zero displacement (Figure 5-6). No other coefficients show large differences.

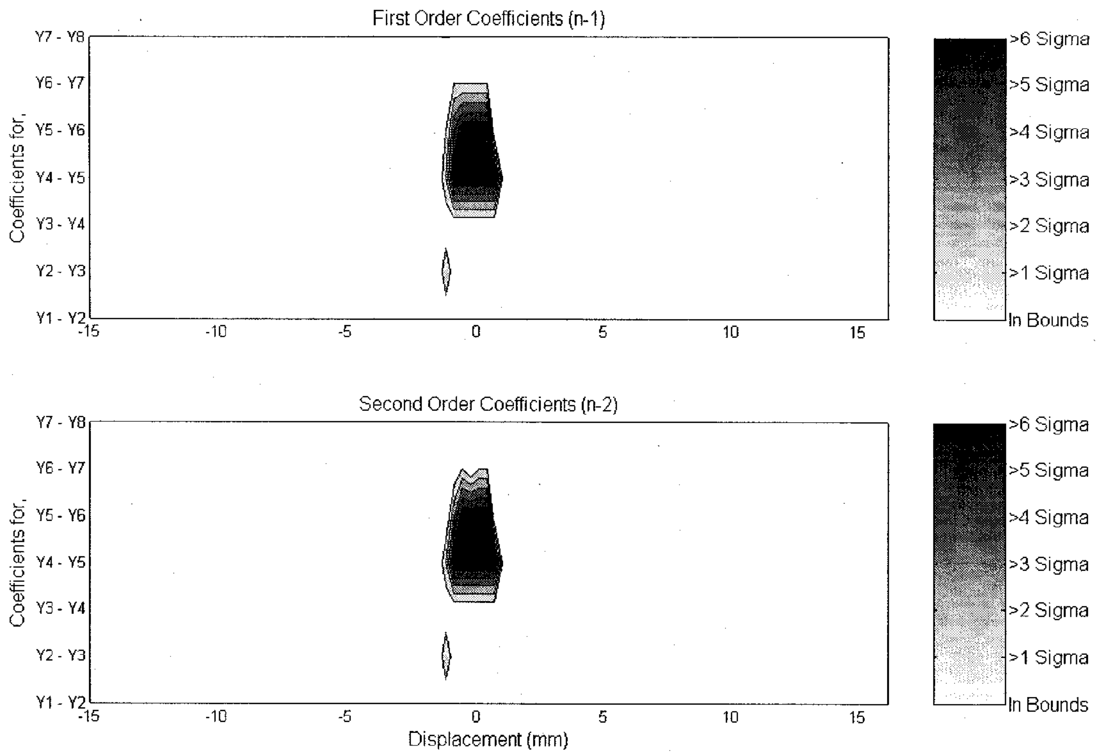


Figure 5-5. Simulated Damage-1s Compared to Undamaged-s Difference Coefficients

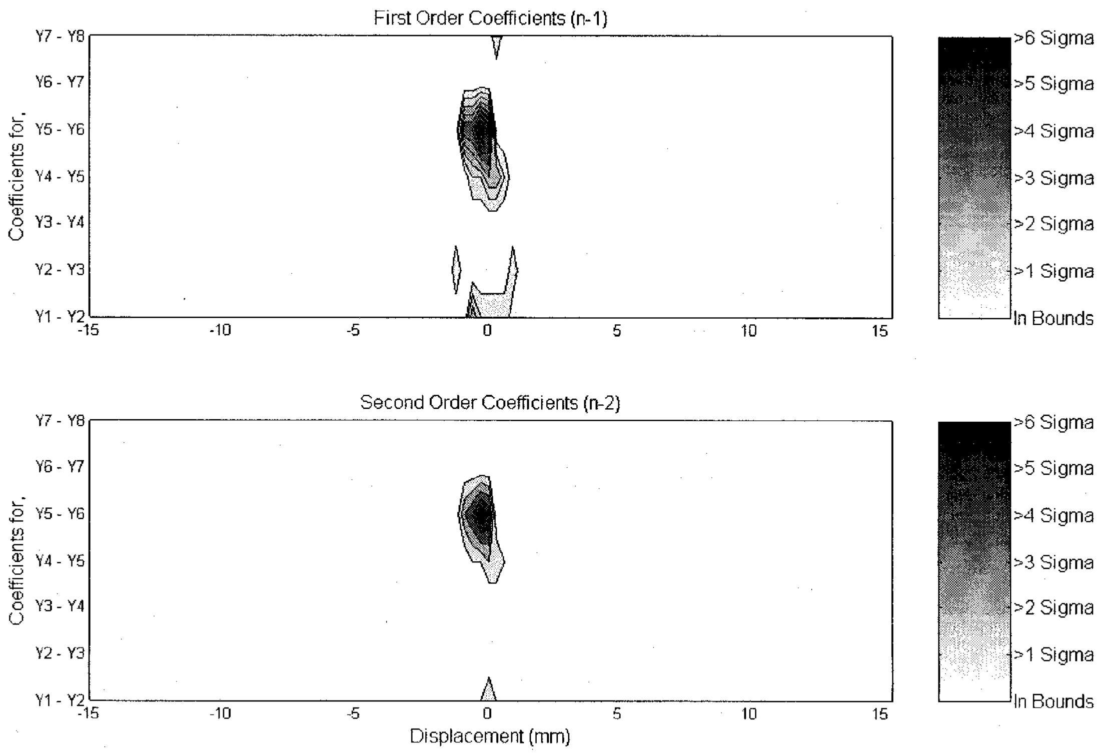


Figure 5-6. Simulated Damage-2s Compared to Damage-1s Difference Coefficients

Chapter 6 - Summary

This Chapter examines the conclusions that are drawn from the research covered, illustrates how to apply the developed method, as well as outlines future research that could bring increased performance and further utility to the SI/MO NL-ARX-GLS method developed.

6.1 Conclusions

A method for performing system identification of nonlinear single-input/multiple-output systems has been developed (SI/MO NL-ARX-GLS). The method utilizes basis sets to form a pool of linear independent vectors. Coefficients are found, using least squares, to form linear combinations of the vector pool to generate ARX coefficients that are a function of the states (outputs) of the system. Using a filter created with the residuals, the effects of noise on the input and outputs are reduced. The process is then repeated by recalculating the coefficients using the filtered input and outputs until the residuals converge. This is performed on each output independently.

The SI/MO NL-ARX-GLS identification method was demonstrated on a three degree-of-freedom nonlinear numerical example. The accuracy of the identified model was demonstrated by the close match of results from the truth model to the identified model using a different input than what was used for the identification. It was then shown how the coefficients from the identified model could be used for nonlinear damage detection, as well identification of a linear system that becomes nonlinear after being damaged. This was accomplished by using data from tests performed at Los Alamos on a nonlinear eight degree-of-freedom system. Not only did the method accurately identify the "damage" in the model, it did so using non-ideal data. Acceleration data was used, whereas the nonlinearity was a function of displacement; the model order was chosen too high and it was assumed that nonlinearities existed between every output. A numerical example, similar to the Los Alamos test rig, was used to show the methods capability of identifying the location of the damage in the operating range.

A SI/MO NL-ARX-GLS method was successfully developed and proven to be accurate in both producing models that represented the original system and finding locations of possible damage.

The benefits of the developed method are that it is capable of producing a state-dependent ARX model with discontinuous coefficients to reproduce the characteristics of a nonlinear system. This can be accomplished using corrupted input and output data. The results from the identification are then used for damage detection, applicable to both a linear system that becomes nonlinear after damage and a nonlinear system whose characteristics change after damage. The location and range of operation of the damage are predicted.

6.2 Application of NL-ARX-GLS

The nonlinear system identification method developed within this dissertation can be applied in a pure system identification role, or for damage detection. There are subtle differences in the identification process depending on the application. These differences are centered on how well the identified model reproduces the original results. In a system identification role, the focus is on how well the model reproduces the actual system. In a damage detection application, the ability to detect changes in the system at a given resolution is the goal. The following is a guide on how to apply the SI/MO NL-ARX-

GLS method in both roles. Although the method was developed for black-box applications, the subsequent text will assume some characteristics of the system are known.

Choosing the correct instrumentation, placement, and sample rate is key to how well the method will perform in both roles. The key contributor to this selection is the type and location of the expected nonlinearities, as well as the system's natural frequencies. The measurements should be, or be able to be converted into, the parameter the nonlinearity is a function of. For instance, a crack's behavior is a function of displacement; coulomb friction is a function of velocity. If one is using an accelerometer but needs displacement, the sample rate and accuracy need to be adequate for estimating the displacement. An estimate of the initial conditions is also required if the nonlinearity is a function of relative displacement or velocity of two measurements. The characteristics of the nonlinearity and the quality of the data due to noise both have to be considered as well when choosing the sample rate.

Accurately identifying a quickly changing nonlinearity, such as a hard stop, clearly drives the sample rate up. Less obviously, coping with noisy data drives the sample rate down. This can be explained by using the simple estimate for the derivative of a measurement

$$\frac{dy}{dt} = \frac{y_{n+1}^{a} - y_{n}^{a}}{\Delta t} + \frac{\delta}{\Delta t}, \tag{6-1}$$

where superscript "*a*" refers to actual value, Δt is the time step and δ is the noise contribution. Equation 6-1 shows that as the time step is increased, the noise has less effect for a given noise level. Of course, the time step has to be adequate for the highest frequency of interest.

The effect of sample rate on noise can be demonstrated using the system and forcing function from Section 4.1.1, without the hard stop. Figure 6-1 shows the Goodness of Fit [31] from ARX-GLS identification of simulations with different sample rates and SNR. The figure shows how larger time steps generate good fits for lower SNR as compared to identifications with smaller time steps. Therefore, if applying NL-ARX-GLS identification to a system that requires small time steps to accurately identify a nonlinearity, care must be taken in getting the cleanest possible measurements.

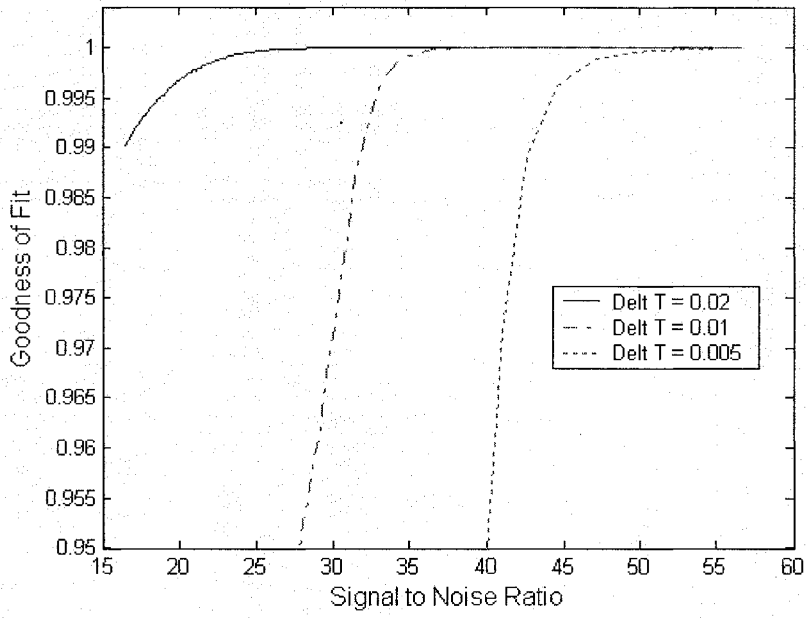


Figure 6-1. Interaction of Sample Rate and Noise on Goodness of Fit

If the goal of the identification is an accurate model for duplicating the system's behavior, an approach for finding the optimum identification parameters similar to that outlined in Section 4.1.1 should be followed. However, if damage detection is the purpose, the selection of the basis set and number of points in the basis vector is driven by the required detectable changes in the ARX coefficients. This was demonstrated with the numerical example in Section 5.1.1. The combination of the number of Walsh functions and the number of points describing the basis vectors yielded the capability of detecting changes of the nonlinearity on the order of tenths of millimeters. The ability of the model to duplicate the original system response is not necessary when performing damage detection.

The number and location of nonlinearities in the system being identified or the resolution of which to detect damage drive the placement of measurements. Measurements closely coupled to nonlinearities will generate the best characterization of the nonlinearity and more measurements will result in more refined estimates of damage location.

6.3 Recommended Future Research

Increased performance and further utility of the SI/MO NL-ARX-GLS method can be achieved by accomplishing the following task.

The method used for finding the optimum model order and basis sets was valid but time consuming. It involved performing a series of identifications using all possible combinations of the parameters being studied and used the set that generated the smallest Root-Sum-Square-Error of the output model being identified. The development of an optimization method for finding both the model order and basis set would both increase the speed and the accuracy of the identification, due it allowing a larger space to be searched. Execution time might also be improved by using a recursive least-squares method to find the coefficients to the basis functions.

The code written and the single processor used for this research did the identification for each output serially. Due to the SI/MO NL-ARX-GLS method performing identification on each output independently, they could be performed in parallel if the resources were available. Adapting the code to run on multiple processors would greatly speed up the identification process.

This research applied the developed identification method to damage detection of nonlinear system, yet the ability to create accurate models lends it to the development of control laws as well. The model could be linearized at discrete points throughout its operational envelope. The characteristics of the linearized versions of the model could then be used to develop control laws for the input to achieve the desired output performance.

References

1. Doebling, S.O, Farrar, C., Prime, M., and Shevitz, D., *Damage Identification and Health Monitoring of Structural and Mechanical Systems From Changes in Their Vibration Characteristics: A Literature Review*, Los Alamos National Laboratory Report LA-13070-MS (May 1996)
2. Sohn, H., Farrar, C., Hemez, F., Czarnecki, J., Shunk, D., Stinemates, D. and Nadler, B., *A Review of Structural Health Monitoring Literature: 1996-2001*, Los Alamos National Laboratory Report, LA-13976-MS, 2004
3. Slotine, J.J.E. and Weiping, L. (1991), *Applied Nonlinear Control*, Prentice Hall, New Jersey
4. Juang, J.N. and Papa, R.S. (1985), *An Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction*, J. Guidance, Vol. 8, No. 5, pp. 620-627
5. Bendat, J.S. (1998), *Nonlinear Systems Techniques and Applications*, John Wiley & Sons, New York
6. Feldman, M., *Non-Linear System Vibration Analysis Using Hilbert Transform – I. Free Vibration Analysis Method 'FREEVIB'*, Mechanical Systems and Signal Processing, 1994, 8(2), pp. 119-127
7. Feldman, M., *Non-Linear System Vibration Analysis Using Hilbert Transform – II. Forced Vibration Analysis Method 'FORCEVIB'*, Mechanical Systems and Signal Processing, 1994, 8(3), pp. 309-318

8. Huang, N.E., Shen, Z., Long, S. R., Wu, M.C., Shih, H.H., Zheng, Q., Yen, N., Tung, C.C., and Liu, H.H., *The Empirical Mode Decomposition and the Hilbert Spectrum for Nonlinear and Non-Stationary Time Series*, Proceedings Royal Society London, 1998, 454, pp. 903-995

9. Young, P., McKenna, P. and Bruun, J., *Identification of Non-Linear Stochastic System by State-Dependent Parameter Estimation*, International Journal of Control, 2001, Vol. 74, No. 18, pp. 1837-1857

10. Toivonen, H., *State-Dependent Parameter Models of Non-Linear Sampled-Data Systems: a Velocity-Based Linearization Approach*, International Journal of Control, 2003, Vol. 76, No. 18, pp. 1823-1832

11. Åkesson, B. and Toivonen, H., *State-Dependent Parameter Modeling and Identification of Stochastic Non-Linear Sampled-Data Systems*, Journal of Process Control, 16, 2006, pp. 877-886

12. Ljung, L., *System Identification – Theory for the User*, Prentice-Hall, New Jersey, 1999

13. Peng, H., Ozaki, T., Toyoda, Y. and Oda, K., *Nonlinear System Identification using Radial Basis Function-Based Signal-Dependent ARX Model*, 5th IFAC Symposium on Nonlinear Control Systems, July 2001, Saint-Petersburg, Russia, pp. 703-708

14. Peng, H., Ozaki, T., Toyoda, Y., Shioya, H., Nakano, K., Haggan-Ozaki, V., and Mori, M., *RBF-ARX Model-Based Nonlinear System Modeling and Predictive Control with Application to a NOx Decomposition Process*, Control Engineering Practice 12, 2004, 191-203

15. Billings, S. and Wei, H., *The Wavelet-NARMAX Representation: A Hybrid Model Structure Combining Polynomial Models with Multiresolution Wavelet Decompositions*, International Journal of Systems Science, Vol. 36, No. 3, February 2005, pp. 137-152

16. Hu, J., Kumamaru, K. and Hirasawa, K., *A Quasi-ARMAX Approach to Modeling of Non-Linear Systems*, International Journal of Control, 2001, Vol. 74, No. 18, pp. 1754-1766

17. Gómez, J. and Baeyens, E., *Identification of Nonlinear Systems using Orthonormal Bases*. In Proceedings of the IASTED International Conference on Intelligent Systems and Control, ISC 2001, Tampa, Florida, pp. 126-131, November 2001

18. Åström, K. and Björn W., *Computer-Controlled Systems: Theory and Design*, Prentice Hall, Upper Saddle River, NJ, 1997

19. Golub, G. and Van Loan, C., *Matrix Computations*, 3rd Edition, The Johns Hopkins University Press, Baltimore, MD, 1996

20. MATLAB, Simulink, Stateflow, Handle Graphics, and Real-time Workshop are registered trademarks, and TargetBox is a trademark of The MathWorks, Inc.

21. Lu, S., Ju, H. and Chon, K., *A New Algorithm for Linear and Nonlinear ARMA Model Parameter Estimation Using Affine Geometry*, IEEE Transitions on Biomedical Engeineering, Vol. 48, No. 10, October 2001

22. Zou, R., Wang, H. and Chon, K., *A Robust Time-Varying Identification Algorithm Using Basis Functions, Annals of Biomedical Engineering*, Vol. 31, pp. 840-853, 2003

23. Lu, S. and Chon, K., *Nonlinear Autoregressive and Nonlinear Autoregressive Moving Average Model Parameter Estimation by Minimizing Hypersurface Distance*, IEEE Transactions on Signal Processing, Vol. 51, No. 21, December 2003

24. Zou, R. and Chon, K., *Robust Algorithm for Estimation of Time-Varying Transfer Functions, IEEE Transactions on Biomedical Engineering*, Vol. 51, No. 2, February 2004

25. Bendat, J. and Piersol, A., *Random Data Analysis and Measurement Procedures*, 3rd Edition, Wiley Series in Probability and Statistics, John Wiley & Sons, Inc., 2000

26. Clarke, D., *Generalized Least Squares Estimation of Parameters of a Dynamic Model*, First IFAC Symposium on Identification in Automatic Control Systems, Prague, 1967

27. Zhang, Q., Basseville, M. and Benveniste, A., *Fault Detection and Isolation in Nonlinear Dynamic Systems, a Combined Input-Output and Local Approach*, Research Report IRISA, No. 1074, November 1996

28. Sohn, H., Worden, K. and Farrar, C., *Statistical Damage Classification under Changing Environmental and Operational Conditions*, Journal of Intelligent Materials Systems and Structures. LA-UR-02-1992

29. http://ext.lanl.gov/projects/damage_id/data.shtml

30. Phum, C. and Ogunfunmi, T., *On the Recursive Total Least-Squares,* IEEE 0-8186-7919-0/97, pp. 1989-1992, 1997

31. Walpole, R. and Myers, R., *Probability and Statistics for Engineers and Scientists*, 5th Edition, Macmillan Publishing Company, New York, 1993

Appendix A – Time-Invariant Linear ARX Model Identification Routine

```
function[a,b,ystar] = arx_simple(y,u,P,Q);
%
% [alpha,beta,ystar] = arx_simple(y,u,P,Q)
%
% finds coefficients for ARX model of the form:
% y(n) = a(1)*y(n-1) +...+ a(P)*y(n-P) + b(0)*u(n) +...+ b(Q)*u(n-Q)
%
% Also returns synthesis of y → ystar
%
% find length of data set
ndata = length(y);
% initialize phi matrix
phi = zeros(ndata,P+Q+1);
% populate phi matrix
for n = 1:ndata
   for idata = 1:P
      if n-idata > 0
         phi(n,idata) = y(n-idata);
      end
   end
   for jdata = 0:Q
      if n-jdata > 0
         phi(n,P+1+jdata) = u(n-jdata);
      end
   end
end
% find coefficients using least squares
theta = inv(phi'*phi)*phi'*y;
% divide coefficients in to "a" and "b"
a = theta(1:P);
b = theta(P+1:P+Q+1);
%
% synthesize response for comparison to original data
%
% initialize ystar vector
ystar = zeros(ndata,1);
% perform synthezsis using coefficeints found above
for n = 1:ndata
   for idata = 1:P
      if n-idata > 0
         ystar(n) = ystar(n) + a(idata)*ystar(n-idata);
      end
   end
   for jdata = 0:Q
```

```
        if n-jdata > 0
            ystar(n) = ystar(n) + b(jdata+1)*u(n-jdata);
        end
    end
end
```

Appendix B – Orthogonal Basis Generating Routines

```
function [wf] = walsh(nfun,ndata);
%
% [wf] = walsh(nfun,ndata)
%
% generates the first "nfun" walsh functions with "ndata" values
%
% ndata needs to be an even integer
%
wf = zeros(nfun,ndata);
wf(1,:) = ones(1,ndata);
for iodd = 2:2:nfun-1
   wf(iodd,1:ndata/2-1) = wf(iodd-1,1:ndata/2-1);
   wf(iodd,ndata/2:ndata) = -wf(iodd-1,ndata/2:ndata);
   ieven = iodd+1;
   wf(ieven,1:ndata/2) = wf((ieven-1)/2+1,1:2:ndata-1);
   wf(ieven,ndata/2+1:ndata) = wf((ieven-1)/2+1,1:2:ndata)*(-1).^((ieven-1)/2);
end
if nfun/2 == floor(nfun/2)
      wf(nfun,1:ndata/2-1) = wf(nfun-1,1:ndata/2-1);
      wf(nfun,ndata/2:ndata) = -wf(nfun-1,ndata/2:ndata);
end
*********************************************************************
function [cf] = cheby(nfun,ndata);
%
% [wf] = cheby(ndata)
%
% Generates the first "nfun" Chebyshev Polynomials of the first kind
% with "ndata" values
%
ymax = 1;
ymin = -1;
m = (ymax - ymin)/(ndata - 1);
b = ymin - m;

cf(1,:) = ones(1,ndata);
for idata = 1:ndata
   x = idata*m + b;
   for n = 1:nfun-1
      cf(n+1,idata) = 0;
      for k = 0:n/2
         cf(n+1,idata) = cf(n+1,idata) + n/2*(-1)^k*factorial(n-k-1)/factorial(k)/factorial(n-
2*k)*(2*x)^(n-2*k);
      end
   end
end
**********************************************************************
function [bf] = block(nfun,ndata);
%
%  [bf] = block(nfun,ndata)
%
%  generates "nfun" block pulses with "ndata" values
%
step = ndata/nfun;
bf = zeros(nfun,ndata);
for ifun = 1:nfun
   bf(ifun,round((ifun-1)*step)+1:round(ifun*step)) = 1;
end
```

Appendix C – State-dependent Nonlinear ARX Model Identification Routines

```
function [a,b,ys] = NLARX(y,u,P,Q,nfunb,nfunw,nfunc,nvec);
%
% [a,b,ys] = NLARX(y,u,P,Q,nfunb,nfunw,nfunc,nvec)
%
%  Nonlinear ARX model identification - SI/SO
%
% y = output
% u = input
% P = Order of output fit
% Q = Order of input fit
% nfunb = number of block functions
% nfunw = number of vectors in Walsh basis
% nfunc = number of vectors in Chebyshev basis
% nvec = length of basis vector
% a = output coefficients
% b = input coefficients
% ys = synthesis of y with IDed model

%  define block basis to be used
disp('Generating Block Basis')
[bkf] = block(nfunb,nvec);

%  define walsh basis to be used
disp('Generating Walsh Basis')
[wf] = walsh(nfunw,nvec);

%  define Chebyshev to be used
disp('Generating Chebyshev Basis')
[cf] = cheby(nfunc,nvec);

%  generate vector pool
disp('Generating Vector Pool')
[bf] = [bkf',wf',cf']';
[yk,uk,xy] = pool(y,u,bf,P,Q);

%  find linear independent set of vector pool
disp('Selecting Linear Independent Vectors from Pool')
[yindex,uindex] = linindi(yk,uk);

%  assemble new matrix pool
disp('Assembling LI Vectors')
W = [yk(:,yindex),uk(:,uindex)];

%  find ARX model coefficients
```

```
disp('Solving for Coefficients')
c = pinv(W)*y;

% assemble a and b coefficients
disp('Constucting a and b Coefficients')
[a,b] = assemble(bf,yindex,uindex,c);

% trim a and b coefficients outside of ID range
disp('Trim a and b Coefficients')
[a,b] = trim(a,b,xy,y);

% reproduce response with SV-ARX model
disp('Reconstucting Original Input')
[ys] = timeh(a,b,xy,u);function[yk,uk,xy] = pool(y,u,bf,P,Q);
**********************************************************************
function[yk,uk,xy] = pool(y,u,bf,P,Q);
%
% Generates vector pool for SV ARX
%
% [yk,uk,xy] = pool(y,u,bf,P,Q)
%
% y = system output
% u = system input
% P = order of autoregressive model
% Q = order of moving average model
% bf(nfun,nvec) = basis set of vectors
%            nfun - number of vectors
%            nvec = length of vectors
%

% determine size of basis
[nfun,nvec] = size(bf);

% map basis vector to range of y
ymax = max(y)+abs(max(y)-min(y))*.1;
ymin = min(y)-abs(max(y)-min(y))*.1;
slope = (ymax - ymin)/(nvec - 1);
inter = ymin - slope;
xy = [1:nvec]*slope + inter;

fun_mat = interp1(xy',bf',y');

% generate y pool vectors
yk = zeros(length(y),P*nfun);
for idata = 1:P
   for k = 1:nfun
```

```
        for n = 2:length(y)
           if n-idata > 0
              yk(n,(idata-1)*nfun+k) = fun_mat(n-1,k).*y(n-idata);
           end
        end
     end
  end

  % generate u pool vectors
  uk = zeros(length(y),Q*nfun);
  for jdata = 0:Q
     for k = 1:nfun
        for n = 2:length(y)
           if n-jdata > 0
              uk(n,(jdata)*nfun+k) = fun_mat(n-1,k).*u(n-jdata);
           end
        end
     end
  end
  **********************************************************************
  function[yindex,uindex] = linindi(yk,uk);
  %
  %  [yindex,uindex] = linindi(yk,uk)
  %
  %  This routine is used with the SVOPS method.  It retruns the indices of
  %  the linearly independent vectors in yk and uk.
  %
  %

  [nvec,nyk] = size(yk);
  [nvec,nuk] = size(uk);

  yindex = [];
  uindex = [];

  % estimate rank
  r = rank([yk uk]);

  % perform QR
  [Q,R,E] = qr([yk uk]);

  for isort = 1:r
     nfind = find(E(:,isort));
     if nfind <= nyk
        yindex = [yindex,nfind];
     else
```

```
        uindex = [uindex,nfind-nyk];
    end
end
*************************************************************************
function[a,b] = assemble(bf,yindex,uindex,c);
%
%  [alpha,beta] = assemble(bf,yindex,uindex,c)
%
[nfun,nvec] = size(bf);
a = zeros(max(ceil(yindex/nfun)),nvec);
b = zeros(max(ceil(uindex/nfun)),nvec);

for ydata = 1:length(yindex)
    idata = ceil(yindex(ydata)/nfun);
    k = yindex(ydata) - (idata-1)*nfun;
    a(idata,:) = a(idata,:) + c(ydata)*bf(k,:);
end

cshift = length(yindex);
for udata = 1:length(uindex)
    jdata = ceil(uindex(udata)/nfun);
    k = uindex(udata) - (jdata-1)*nfun;
    b(jdata,:) = b(jdata,:) + c(udata + cshift)*bf(k,:);
end
*************************************************************************
function[a,b] = trim(a,b,xy,y);
%
%   [at,bt] = trim(a,b,xy,y)
%
% This routine truncates the a and b coefficients to
% the value at the minimum and maximum value of y.
%
% a - output coefficients of SV-ARX model (and truncated)
% b - input coefficients of SV-ARX model (and truncated)
% xy - independent vector for coefficients
% y - output vector used to construct SV-ARX model
%
%
[P,nvec] = size(a);
[Q,nvec] = size(b);

maxy = max(y);
miny = min(y);

cutlow = find(xy<miny);
cuthigh = find(xy>maxy);
```

```
for iP = 1:P
   a(iP,cutlow) = a(iP,cutlow(end));
   a(iP,cuthigh) = a(iP,cuthigh(1));
end

for iQ = 1:Q
   b(iQ,cutlow) = b(iQ,cutlow(end));
   b(iQ,cuthigh) = b(iQ,cuthigh(1));
end
***********************************************************************
function [ys] = timeh(a,b,xy,u);
%
%    [ys] = timeh(a,b,xy,u)
%

[P,nvec]=size(a);
[Q,nvec]=size(b);
Q = Q-1;
ndata = length(u);
ys = zeros(ndata,1);

for n = 2:ndata
   for idata = 1:P
      if n-idata > 0
         test = isnan(interp1(xy,a(idata,:),ys(n-1),'linear','extrap'));
         if test > 0
            'NaN in timeh.m'
            n
            return
         end
         ys(n) = ys(n) + ys(n-idata)*interp1(xy,a(idata,:),ys(n-1),'linear','extrap');
      end
   end
   for jdata = 0:Q
      if n-jdata > 0
         ys(n) = ys(n) + u(n-jdata)*interp1(xy,b(jdata+1,:),ys(n-1),'linear','extrap');
      end
   end
end
```

Appendix D – Nonlinear ARX Generalized Least Squares Routines

```
function [a,b,xy,ys] = NLARXGLS(y,u,P,Q,R,nfunb,nfunw,nfunc,nvec,tol_gls,maxiter);
%
% [a,b,ys] = NLARXGLS(y,u,P,Q,R,nfunb,nfunw,nfunc,nvec,tol_gls,maxiter)
%
%  Nonlinear ARX-GLS model identification - SI/SO
%
% y = output
% u = input
% P = Order of output fit
% Q = Order of input fit
% R = Order of GLS filter
% nfunb = number of block functions
% nfunw = number of vectors in Walsh basis
% nfunc = number of vectors in Chebyshev basis
% nvec = length of basis vector
% a = output coefficients
% b = input coefficients
% ys = synthesis of y with IDed model
% tol_gls = generalized least squares error tolerance
% maxiter = maximum number of iterations

%  define block basis to be used
disp('Generating Block Basis')
[bkf] = block(nfunb,nvec);

%  define walsh basis to be used
disp('Generating Walsh Basis')
[wf] = walsh(nfunw,nvec);

%  define Chebyshev to be used
disp('Generating Chebyshev Basis')
[cf] = cheby(nfunc,nvec);

%  set initial filtered value to raw data
yf = y;
uf = u;

%  initialize residue vector
eo = ones(size(y));
test = 1;
iter = 0;

%  check for convergence and number of iterations
while test > tol_gls & iter < maxiter
```

```
iter = iter + 1;
disp('Iteration Number')
disp(iter)

% generate vector pool
disp('Generating Vector Pool')
[bf] = [bkf',wf',cf']';
[yk,uk,xy] = pool(yf,uf,bf,P,Q);

% find linear independent set of vector pool
disp('Selecting Linear Independent Vectors from Pool')
[yindex,uindex] = linindi(yk,uk);

% assemble new matrix pool
disp('Assembling LI Vectors')
W = [yk(:,yindex),uk(:,uindex)];

% find NLARX model coefficients
disp('Solving for Coefficients')
t = pinv(W)*yf;

% calculate residual
e = yf - W*t;

% find change in residue for convergence test
test = abs((e'*eo)-eo'*eo)/(eo'*eo);
rmse = sqrt(sum((e-eo).^2));
disp('Convergence Test')
disp(test)
eo = e;

% call generalized least squares routine
disp('Call GLS')
[yf,uf] = gls(y,yf,uf,e,R);

end

% assemble a and b coefficients
disp('Constucting a and b Coefficients')
[a,b] = assemble(bf,yindex,uindex,t);

% trim a and b coefficients outside of ID range
disp('Trim a and b Coefficients')
[a,b] = trim(a,b,xy,y);
```

```
% reproduce response with SV-ARX model
disp('Reconstucting Original Input')
[ys] = timeh(a,b,xy,u);
************************************************************************
function[yf,uf] = gls(y,yfo,ufo,e,R)
%
%  [yf,uf] = gls(y,yf,uf,e,R)
%
% Generalized Least Squares
% filters input and output based on AR model of error
%
% y - raw input
% yf - filtered output
% uf - filtered input
% e - solution residuals
% R - order of filter
%
clear BB
for n = 1:length(y)
   for ldata = 1:R
      if n-ldata > 0
         BB(n,ldata) = e(n-ldata);
      end
   end
end
% find AR noise model coefficients
c = -pinv(BB)*e;

yf = yfo;
uf = ufo;

for n = 1:length(y)
   for ldata = 1:R
      if n-ldata > 0
         yf(n) = yf(n) + c(ldata)*yfo(n-ldata);
         uf(n) = uf(n) + c(ldata)*ufo(n-ldata);
      end
   end
end
scale = sqrt(var(y)/var(yf));
yf = yf*scale;
uf = uf*scale;
```

Appendix E – Single-Input/Multiple-Output Nonlinear ARX – GLS Routines

```
% SI/MO Nonlinear ARS-GLS Parameters
% used in example in Section 4.3.1
%
nout = 3;         % number of outputs
nvec = 1000;      % length of basis vector
tol_gls = 1e-9;   % convergence tolerance for GLS
maxiter = 200;    % maximum iterations
% pre-filter input and outputs
y1f = prefilt(y1);
y2f = prefilt(y2);
y3f = prefilt(y3);
uf1 = prefilt(f1);
y=[y1f,y2f,y3f];
u=[uf1];
f=[f1];
time = [0:length(y1)-1]*.01;
% DOF 1 Parameters
iref = 1;
P(iref) = 2;       % Order of Output Model
Q(iref) = 1;       % Order of Input Model
R(iref) = 8;       % Order of Error Model
nfunw(1,iref) = 1;  % number of vectors in Walsh basis for y1
nfunw(2,iref) = 1;  % number of vectors in Walsh basis for y1-y2
nfunw(3,iref) = 1;  % number of vectors in Walsh basis for y1-y2
nfunc(1,iref) = 4;  % number of vectors in Chebyshev basis for y1
nfunc(2,iref) = 4;  % number of vectors in Chebyshev basis for y1-y2
nfunc(3,iref) = 1;  % number of vectors in Chebyshev basis for y1-y2
nfunb(1,iref) = 1;  % number of block pulse basis for y1
nfunb(2,iref) = 1;  % number of block pulse basis for y1-y2
nfunb(3,iref) = 1;  % number of block pulse basis for y1-y2
% Peform ID for DOF 1
[a1,b1,xy1,ys1] = mnlarxgls(y,u,P(iref),Q(iref),R(iref),nfunw(:,iref)...
          ,nfunc(:,iref),nfunb(:,iref),nvec,tol_gls,maxiter,iref);
% DOF 2 Parameters
iref = 2;
P(iref) = 2;       % Order of Output Model
Q(iref) = 1;       % Order of Input Model
R(iref) = 8;       % Order of Error Model
nfunw(1,iref) = 1;  % number of vectors in Walsh basis for y2
nfunw(2,iref) = 1;  % number of vectors in Walsh basis for y2-y1
nfunw(3,iref) = 1;  % number of vectors in Walsh basis for y2-y3
nfunc(1,iref) = 1;  % number of vectors in Chebyshev basis for y2
nfunc(2,iref) = 4;  % number of vectors in Chebyshev basis for y2-y1
nfunc(3,iref) = 1;  % number of vectors in Chebyshev basis for y2-y3
```

```
nfunb(1,iref) = 14;  % number of block pulse basis for y2
nfunb(2,iref) = 1;  % number of block pulse basis for y2-y1
nfunb(3,iref) = 13;  % number of block pulse basis for y2-y3
% Peform ID for DOF 2
[a2,b2,xy2,ys2] = mnlarxgls(y,u,P(iref),Q(iref),R(iref),nfunw(:,iref)...
            ,nfunc(:,iref),nfunb(:,iref),nvec,tol_gls,maxiter,iref);
% DOF 3 Parameters
iref = 3;
P(iref) = 2;        % Order of Output Model
Q(iref) = 1;        % Order of Input Model
R(iref) = 8;        % Order of Error Model
nfunw(1,iref) = 1;   % number of vectors in Walsh basis for y3
nfunw(2,iref) = 1;   % number of vectors in Walsh basis for y3-y1
nfunw(3,iref) = 1;   % number of vectors in Walsh basis for y3-y2
nfunc(1,iref) = 4;   % number of vectors in Chebyshev basis for y3
nfunc(2,iref) = 1;   % number of vectors in Chebyshev basis for y3-y1
nfunc(3,iref) = 1;   % number of vectors in Chebyshev basis for y3-y2
nfunb(1,iref) = 1;   % number of block pulse basis for y3
nfunb(2,iref) = 4;  % number of block pulse basis for y3-y1
nfunb(3,iref) = 15;  % number of block pulse basis for y3-y2
% Peform ID for DOF 3
[a3,b3,xy3,ys3] = mnlarxgls(y,u,P(iref),Q(iref),R(iref),nfunw(:,iref)...
            ,nfunc(:,iref),nfunb(:,iref),nvec,tol_gls,maxiter,iref);
% Synthesis original measure data
xy = [xy1 xy2 xy3];
a = [a1' a2' a3']';
b = [b1' b2' b3']';
[yy] = simotimeh(f,nout,P,Q,a,b,xy);
*********************************************************************
function[xf] = prefilt(x);
%
% Routine for pre-filtering data
%
%   [xf] = prefilt(x)
%
% x - input
% xf - output
%
ndata = length(x);
xf = zeros(size(x));
xf(1) = x(1);
for idata = 2:ndata;
   xf(idata) = (x(idata) + x(idata-1))/2;
end
*********************************************************************
function[a,b,xy,ys] = mnlarxgls(y,u,P,Q,R,nfunw,nfunc,nfunb,nvec,tol_gls,maxiter,iref)
```

```
%
% [a,b,ys] = NLARXGLS(y,u,P,Q,R,nfunw,nfunc,nfunb,nvec,tol_gls,maxiter,iref)
%
%  Nonlinear ARX-GLS model identification - SI/SO
%
% y = output
% u = input
% P = Order of output fit
% Q = Order of input fit
% R = Order of GLS filter
% nfunb = number of block functions
% nfunw = number of vectors in Walsh basis
% nfunc = number of vectors in Chebyshev basis
% nvec = length of basis vector
% a = output coefficients
% b = input coefficients
% ys = synthesis of y with IDed model
% tol_gls = generalized least squares error tolerance
% maxiter = maximum number of iterations
% iref = reference channel

% determine number of input/outputs and length of data
[ndata,nout] = size(y);
[ndata,nin] = size(u);

%  define block basis to be used
bf = zeros(max(nfunw)+max(nfunc)+max(nfunb)+1,nvec,nout);
for iout = 1:nout
   bf(1:nfunw(iout),1:nvec,iout) = walsh(nfunw(iout),nvec);
   bf(max(nfunw)+1:max(nfunw)+nfunc(iout),1:nvec,iout) = cheby(nfunc(iout),nvec);
   bf(max(nfunw)+max(nfunc)+1:max(nfunw)+max(nfunc)+nfunb(iout),1:nvec,iout)=...
            block(nfunb(iout),nvec);
end

%  set initial filtered value to raw data
uf = u;

%  populate yf with y(iref) in first column, out diff in rest of columns
iload = 2;
for iout = 1:nout;
   if iout == iref
      yf(:,1) = y(:,iref);
      maxy(1) = max(yf(:,1));
      miny(1) = min(yf(:,1));
   else
      yf(:,iload) = y(:,iref) - y(:,iout);
```

```
        maxy(iload) = max(yf(:,iload));
        miny(iload) = min(yf(:,iload));
        iload = iload + 1;
    end
end

% initialize residue vector
eo = ones(length(y),1);
test = 1;
iter = 0;

% check for convergence and number of iterations
while test > tol_gls & iter < maxiter
iter = iter + 1;
disp('Iteration Number')
disp(iter)

% generate vector pool
disp('Generating Vector Pool')
[yk,fk,xy] = mpool(yf,uf,bf,P,Q);

% find linear independent set of vector pool
disp('Selecting Linear Independent Vectors from Pool')
[yindex,uindex] = linindi(yk,fk);

% assemble new matrix pool
W = [yk(:,yindex),fk(:,uindex)];

% find NLARX model coefficients
disp('Find ARX Coefficients')
t = pinv(W)*yf(:,1);

% calculate residual
disp('Calculate Residuals')
e = yf(:,1) - W*t;

% find change in residue for convergence test
test = abs((e'*eo)-eo'*eo)/(eo'*eo);
disp('Convergence Test')
disp(test)
eo = e;

% call generalized least squares routine
disp('Call GLS')
[yf,uf] = mgls(y,yf,uf,e,R,iref);
```

```
% plot status of solution
iload = 2;
for iout = 1:nout;
   if iout == iref
      subplot(nout,1,1)
      plot(y(:,iref))
      hold on
      plot(yf(:,1),'r')
      legend raw filtered
      ylabel('reference')
      hold off
   else
      subplot(nout,1,iload)
      plot((y(:,iref) - y(:,iout)))
      hold on
      plot(yf(:,iload),'r')
      iload = iload + 1;
      hold off
   end
   pause(.1)
end

end

% assemble a and b coefficients
[a,b] = assemble(bf,yindex,uindex,t,P,Q,y,u);
% trim a and b coefficients outside of ID range
[a,b] = mtrim(a,b,xy,yf,u);
% reproduce response with SV-ARX model
disp('Reconstruction of y with SV-ARX Model')
[ys] = mtimeh(a,b,xy,u,y,iref,P,Q);
**********************************************************************
function[yk,uk,xy] = mpool(y,u,bf,P,Q);
%
% Generates vector pool for MIMO NL ARX GLS
%
% [yk,uk,xy] = pool(y,u,bf,P,Q);
%
% y = system output
% u = system input
% P = Order of output fit
% Q = Order of input fit
% bf(nfun,nvec) = basis set of vectors
%            nfun - number of vectors
%            nvec = length of vectors
%
```

```
% determine size of basis
[nfun,nvec,nout] = size(bf);

[ndata,nout] = size(y);
[ndata,nin] = size(u);

% map basis vector to range of y
for iout = 1:nout
   ymax = max(y(:,iout))+abs(max(y(:,iout))-min(y(:,iout)))*.1;
   ymin = min(y(:,iout))-abs(max(y(:,iout))-min(y(:,iout)))*.1;
   slope = (ymax - ymin)/(nvec - 1);
   inter = ymin - slope;
   xy(:,iout) = [1:nvec]'*slope + inter;
end

yk = [];

% generate AR pool vectors
for iout = 1:nout
   fun_mat = interp1(xy(:,iout)',bf(:,:,iout)',y(:,iout)');
   ykt = zeros(length(y(:,iout)),P*nfun);
   for idata = 1:P
      for k = 1:nfun
         for n = 1:length(y(:,iout))
            if n-idata > 0
               ykt(n,(idata-1)*nfun+k) = fun_mat(n-1,k).*y(n-idata,iout);
            end
         end
      end
   end
   yk = [yk,ykt];
end

uk = [];
fun_mat = interp1(xy(:,1)',bf(:,:,1)',y(:,1)');
% generate MA pool vectors
for iin = 1:nin
   ukt = zeros(length(y(:,1)),Q*nfun);
   for jdata = 0:Q
      for k = 1:nfun
         for n = 2:length(y)
            if n-jdata > 0
               ukt(n,(jdata)*nfun+k) = fun_mat(n-1,k).*u(n-jdata,iin);
            end
         end
```

```
    end
  end
  uk = [uk,ukt];
end
************************************************************************
function[yf,uf] = mgls(y,yfo,ufo,e,R,iref)
%
%  [yf,uf] = gls(y,yf,uf,e,R)
%
% Generalized Least Squares
% filters input and output based on AR model of error
%
% y - raw input
% yf - filtered output
% uf - filtered input
% e - solution residuals
% R - order of filter
%
[ndata,nout] = size(yfo);
[ndata,nin] = size(ufo);

clear BB
for n = 1:length(y)
  for ldata = 1:R
     if n-ldata > 0
        BB(n,ldata) = e(n-ldata);
     end
  end
end

% find AR noise model coefficients
c = -pinv(BB)*e;

yf = yfo;
uf = ufo;

for n = 1:length(y)
   for ldata = 1:R
      if n-ldata > 0
         for iout = 1:nout
            yf(n,iout) = yf(n,iout) + c(ldata)*yfo(n-ldata,iout);
         end
         for iin = 1:nin
            uf(n,iin) = uf(n,iin) + c(ldata)*ufo(n-ldata,iin);
         end
      end
   end
end
scale = sqrt(var(y(:,iref))/var(yf(:,1)));
yf = yf*scale;
uf = uf*scale;
**********************************************************************
function[a,b] = massemble(bf,yindex,uindex,t,P,Q,y,u);
%
%  [a,b] = massemble(bf,yindex,uindex,t,P,Q,y,u)
%
[nfun,nvec,nout] = size(bf);
[ndata,nout] = size(y);
[ndata,nin] = size(u);
a = zeros(P*nout,nvec);
b = zeros((Q+1)*nin,nvec);

for ydata = 1:length(yindex)
   iout = floor((yindex(ydata)-1)/nfun/P+1);
   idata = ceil(yindex(ydata)/nfun);
   k = yindex(ydata) - (idata-1)*nfun;
   a(idata,:) = a(idata,:) + t(ydata)*bf(k,:,iout);
end

tshift = length(yindex);
for udata = 1:length(uindex)
   jdata = ceil(uindex(udata)/nfun);
   k = uindex(udata) - (jdata-1)*nfun;
   b(jdata,:) = b(jdata,:) + t(udata + tshift)*bf(k,:,1);
end
**********************************************************************
function[a,b] = mtrim(a,b,xy,y,u);
%
%   [at,bt] = mtrim(a,b,xy,y)
%
% This routine truncates the a and b coefficients to
% the value at the minimum and maximum value of y.
%
% a - output coefficients of SV-ARX model (and truncated)
% b - input coefficients of SV-ARX model (and truncated)
% xy - independent vector for coefficients
% y - output vector used to construct SV-ARX model
%
%
[nvec,nout] = size(xy);
[ndata,nin] = size(u);
[P,nvec]=size(a);
```

```
P = P/nout;
[Q,nvec]=size(b);
Q = Q/nin;

for iout = 1:nout;
   maxy(iout) = max(y(:,iout));
   miny(iout) = min(y(:,iout));
end

for iout = 1:nout
   cutlow = find(xy(:,iout)<miny(iout));
   cuthigh = find(xy(:,iout)>maxy(iout));
   for iP = 1:P
      ia = (iout-1)*P + iP;
      a(ia,cutlow) = a(ia,cutlow(end));
      a(ia,cuthigh) = a(ia,cuthigh(1));
   end
   if iout == 1
      for iin = 1:nin
         for iQ = 1:Q
            ib = (iin-1)*Q + iQ;
            b(ib,cutlow) = b(ib,cutlow(end));
            b(ib,cuthigh) = b(ib,cuthigh(1));
         end
      end
   end
end
************************************************************************
function [ys] = mtimeh(a,b,xy,u,y,iref,P,Q);
%
% Recreates output time history
%
%  [ys] = mtimeh(a,b,xy,u,y,iref,P,Q);
%
[nvec,nout] = size(xy);
[ndata,nin] = size(u);
[Pd,nvec]=size(a);
ys = zeros(ndata,1);
yf = zeros(ndata,nout);
iload = 2;
for iout = 1:nout;
   if iout == iref
      yf(:,1) = y(:,iref);
   else
      yf(:,iload) = y(:,iout);
      iload = iload + 1;
```

```
  end
end

for n = 2:ndata
  for iout = 1:nout
    if iout == 1
      ydata = ys;
    else
      ydata = ys - yf(:,iout);
    end
    for idata = 1:P
      ia = (iout-1)*P + idata ;
      if n-idata > 0
        ys(n) = ys(n) + ydata(n-idata)*interp1(xy(:,iout),a(ia,:),...
          ydata(n-1),'linear','extrap');
      end
    end
  end
  for iin = 1:nin
    for jdata = 0:Q
      ib = (iin-1)*Q + jdata;
      if n-jdata > 0
        ys(n) = ys(n) + u(n-jdata,iin)*interp1(xy(:,iref),b(ib+1,:),...
          ys(n-1),'linear','extrap');
      end
    end
  end
end
************************************************************************
function [y] = simotimeh(u,nout,P,Q,a,b,xy);
%
%  [y] = simotimeh(u,nout,P,Q,a,b,xy);
%
%  u = input, vector (ndata)
%  nout = number of outputs
%  P = order of output fit for each output, vector (nout)
%  Q = order of input fit for each output, vector (nout)
%  a = output coefficients, [a_1 a_2 ... a_nout], (sum(P)*nout,nvec)
%  b = input coefficients, [b_1 b_2 ... b_nout], (
%
[ndata] = length(u);
y = zeros(ndata,nout);
for n = 2:ndata
        for iout = 1:nout
    xyshift = nout*(iout-1);
    if iout > 1
```

```
          ashift = sum(P(1:iout-1))*nout;
          bshift = sum(Q(1:iout-1))+iout-1;
        else
          ashift = 0;
          bshift = 0;
        end
        for j = 1:P(iout)
          if n-j > 0
            constant = interp1(xy(:,xyshift+1),a(ashift+j,:),y(n-1,iout),'linear','extrap');
            y(n,iout) = y(n,iout) + y(n-j,iout)*constant;
          end
        end
        ipass = 0;
        for pp = 1:nout
          if pp ~= iout
            ipass = ipass+1;
            for qq = 1:P(iout)
              if n-qq > 0
                ia = ipass*P(iout) + qq;
                dely = y(n-1,iout) - y(n-1,pp);
                constant=interp1(xy(:,xyshift+ipass+1),a(ashift+ia,:),dely,'linear','extrap');
                y(n,iout)=y(n,iout) + (y(n-qq,iout)-y(n-qq,pp))*constant;
              end
            end
          end
        end
        for ib = 0:Q(iout)
          if n-ib > 0
            constant = interp1(xy(:,xyshift+1),b(bshift+ib+1,:),y(n-1,iout),'linear','extrap');
            y(n,iout) = y(n,iout) + u(n-ib)*constant;
          end
        end
      end
    end
```