AFIT/GA/ENY/93D-7

MODELING OF A LARGE UNDAMPED SPACE STRUCTURE USING TIME DOMAIN TECHNIQUES

THESIS

Anthony R. Nash
Captain, USAF

AFIT/GA/ENY/93D-7

AFIT/GA/ENY/93D-7

MODELING OF A LARGE UNDAMPED SPACE STRUCTURE USING TIME DOMAIN TECHNIQUES

THESIS

Presented to the
Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Astronautical Engineering

Anthony R. Nash
Captain, USAF

November 1993

Acknowledgments

The completion of this thesis would not have been possible without the assistance of several people. Dr. Bradley Liebst's direction helped to keep me focused on the goals of this project. Thanks also goes to Mr. Jay Anderson of the AFIT labs for his assistance with the PACOSS hardware. I also need to thank Captain Rich Cobb for his assistance and ideas throughout the experimental portion of this program. Finally, thanks go to Dr. Joseph Hollkamp who wrote the MATLAB script files which performed the actual ERA and BLS routines, and who helped me learn how to use them.

Anthony R. Nash

Table of Contents

List of Figures

List of Tables

List of Symbols

$\mathbf{A}$ -	State Space Plant Matrix
$\mathbf{A}_p$ -	Least Squares Discrete Summation Input Matrix
$\mathbf{B}$ -	State Space Input Matrix
$\mathbf{B}_p$ -	Least Squares Discrete Summation Output Matrix
$\mathbf{C}$ -	State Space Output Matrix
$\mathbf{C}_p$ -	Backward Least Squares Summation Input Matrix
$\mathbf{D}$ -	State Space Feedforward Matrix
$\mathbf{D}_p$ -	Backward Least Squares Summation Output Matrix
$\mathbf{D}_{sv}$ -	Diagonal Singular Value Matrix
$\mathbf{E}$ -	Block Identity Matrix
f_s -	System Sampling Frequency
$\mathbf{G}$ -	Damping Matrix
Γ -	Least Squares Data Matrix
$G(s)$ -	System Transfer Function
$\mathbf{H}(j\omega)$ -	Frequency Response
$\mathbf{H}_{rs}(k)$ -	Hankel Matrix
$\mathbf{K}$ -	Stiffness Matrix
$\mathbf{M}$ -	Mass Matrix
$\mathbf{P}$ -	Left-Hand Isometric Singular Value Decomposition Matrix
$\mathbf{Q}$ -	Right-Hand Isometric Singular Value Decomposition Matrix
$\bar{q}(t)$ -	Truth Model State Vector
$\boldsymbol{\theta}$ -	Least Squares Unknown Parameter Vector
$\bar{u}(t)$ -	Imput Vector
ω_d -	System Damped Natural Frequency
ω_n -	System Undamped Natural Frequency
$\bar{x}(k)$ -	Discrete Time State Vector
$\bar{x}(t)$ -	Continuous Time State Vector
$\bar{Y}(k)$ -	Markov Parameters (Impulse Response Output Vector)
$\bar{y}(k)$ -	Discrete Time Output Vector
$\bar{y}(t)$ -	Continuous Time Output Vector
z -	System Eigenvalue
ζ -	System Damping Factor

AFIT/GA/ENY/93D-7

Abstract

The primary objective of this research was to develop an accurate mathematical model in state space form for the Passive and Active Control of Space Structures (PACOSS) Dynamic Test Article (DTA), so that active control techniques can be effectively applied to the system in the future. To accomplish this goal, accurate system identification techniques had to be found which would solve a multi-input multi-output system. The system identification methods chosen, the Eigensystem Realization Algorithm (ERA) and a Backward Least Squares (BLS) technique, were compared to a truth model with a known state space system, to determine the accuracy and applicability of each method. With this completed, PACOSS DTA data was generated and a state space model was developed. Finally, Bode plots of the system model and the actual PACOSS system were compared to determine the accuracy of the models developed. The BLS method did not develop an adequate model. The ERA method developed an accurate *plant* matrix, but the *input* and *output* matrices were inaccurate, resulting in a good match of system poles, but a poor match of system zeros.

MODELING OF A LARGE UNDAMPED SPACE STRUCTURE USING TIME DOMAIN TECHNIQUES

I. Introduction

Future space systems are going to be much larger and require much more accurate pointing and vibration suppression systems. These systems will push the state-of-the-art in control system technology. But a control system cannot be designed without an accurate model of the system to be controlled. This leads to the need for system identification techniques which accurately model large space structures with multiple, low frequency, closely spaced modes (1:463). The Passive and Active Control of Space Structures (PACOSS) Dynamic Test Article (DTA) was designed to simulate a large undamped space structure (2:1). It will allow testing of system identification techniques and active control systems, on an accurate, but inexpensive, earth based test bed.

The objective of this thesis is to develop a state space model for the PACOSS DTA to enable the development of active control systems. This will be done using time domain techniques and will result in a evaluation of each technique's effectiveness in developing models of large space structures.

DTA Background

One of the most demanding tasks of a spacecraft control system is the vibration suppression of the spacecraft after maneuver or impact with space debris. In addition, accurate pointing of weapon or sensor platforms may require not only micro radian accuracy, but settling times under a second. For a large space structure (LSS) with flexible body appendages, this will require extremely advanced control systems. The PACOSS DTA has proven to be an effective earth based test bed to for researching the implementation of both passive and active vibration control techniques for LSS (2:3).

The PACOSS DTA was designed and built by Martin Marietta to simulate a large flexible spacecraft for the purpose of control system design. It has multiple, closely spaced, low frequency modes (most under 10 Hertz), eight inertial mass actuators with collocated accelerometers, and is isolated from the environment by an air bearing system (2:1). Physically, it is shaped like a primary and secondary mirror assembly with two solar panels (Figure 1).

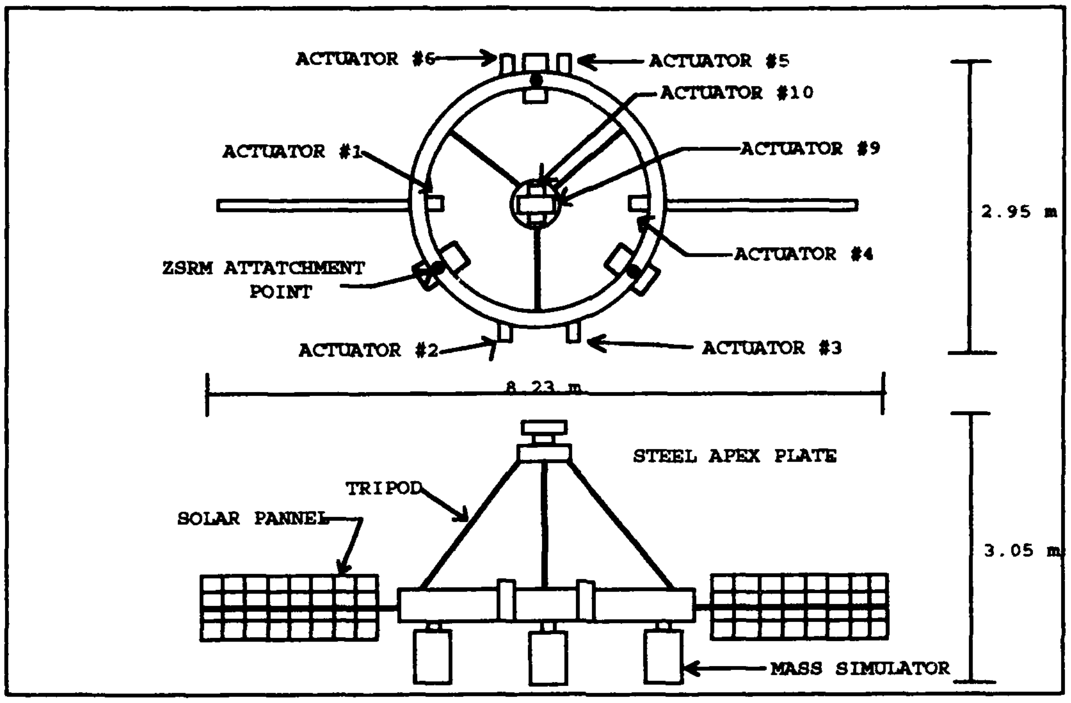


Figure 1: Diagram of the PACOSS DTA

This gives it the shape and modal characteristics of typical large space structures: closely spaced, low frequency modes, as well as low damping ratios.

System Identification Background

Before control systems can be designed, an accurate model of the system must be developed. Several previous attempts have been made to model the PACOSS DTA. Martin Marietta, the developers of PACOSS, used several types of finite element modeling (2). Specifically, they had difficulty identifying the modes in the 1 to 10 Hz range, and found that mode locations and damping factors varied by as much as 20 percent about the average depending upon the particular routine used. These difficulties were attributed to the closely spaced nature of the model, and to the difficulty in acquiring accurate data on such a complex system (11:1). This demonstrates the need to develop new, and more accurate system identification techniques for LSS.

In addition, AFIT students Captain Scott George used a modal analysis technique (7) and Lieutenant Chad Matheson used a frequency domain analysis technique (8) to identify the system modes. Although both were successful at identifying the system modes, Captain George did not attempt to develop a state space model, while the state space model developed by Lieutenant Matheson included only the collocated input/output pairs.

With both frequency domain and finite element techniques already applied to the PACOSS, one set of identification techniques which have not been attempted are the time domain methods. These involve curve fitting a transfer function to a set of actual input disturbance signals sent to the PACOSS DTA and their corresponding output signals received from the DTA. Specifically the Eigensystem Realization Algorithm (ERA) and the Backward Least Squares (BLS) method will be used to determine the system modes, from which the *plant* matrix (or A matrix) and *input* matrix (or B matrix) will be formed. A third method, called the Prediction Error Method (PEM) will be used in conjunction with the above methods to determine the *output* matrix (or C matrix).

The ERA method is based upon a singular value decomposition of the Hankel matrix formed from the Markov parameters. The Markov parameters are the system impulse responses (6:621). The Backward Least Squares method uses a reversed least squares technique to separate the system modes from the computational, or noise, modes (10).

The ERA and BLS methods are specifically designed to identify the system modes and mode shapes of large undamped space structures given time domain data. The ERA method was used by Juang and Pappa at NASA Langley Research Center to accurately identify 34 modes of the Galileo space craft (2:624). Also, Hollkamp was able to accurately identify 10 modes of the 12 meter truss located at Wright Laboratory, Wright-Patterson AFB, Ohio, using both the ERA and the BLS methods (10:553). Although no attempts were made to create state space models of the structures, both were very successful at identifying system modes, damping factors, mode shapes, and modal amplitude participation factors, on large undamped structures similar to the PACOSS DTA. Given the system modes and damping factors, the *plant* matrix can then be developed.

Finally, PEM is based upon an iterative least squares numerical technique to model the system parameters as well as the errors associated with the system output (4:75-6). A computer coded version of the Prediction Error Method can be found in the MATLAB System Identification toolbox (3:I-1).

II. Theory

The development of the equations used in this project are divided into four phases. First is the development of the discrete state space model of any system and how it applies to the PACOSS. This is followed by a development of the techniques which were used to identify the specific state space model of the PACOSS. These techniques were ERA, BLS and PEM. The final section is a development of the method used to combine these techniques in order to improve the accuracy of the system model.

Equations of Motion

The first step in the modeling of any dynamic system is to determine the equations of motion for that system. Any continuous system can be modeled as a lumped parameter spring-mass-damper system. For the PACOSS, with force inputs, the equation of motion is:

$$\mathbf{M}\ddot{\bar{q}}(t) + \mathbf{G}\dot{\bar{q}}(t) + \mathbf{K}\bar{q}(t) = \bar{F}(t) \tag{1}$$

where:

$\mathbf{M}$ -	Mass Matrix (n x n)
$\mathbf{G}$ -	Damping Matrix (n x n)
$\mathbf{K}$ -	Stiffness Matrix (n x n)
$\bar{q}(t)$ -	displacement vector (n x 1)
$\dot{\bar{q}}(t)$ -	velocity vector (n x 1)
$\ddot{\bar{q}}(t)$ -	acceleration vector (n x 1)
n -	number of degrees of freedom

This can then be expressed as the first order continuous system:

$$\begin{aligned} \dot{\bar{x}}(t) &= \mathbf{A}\bar{x}(t) + \mathbf{B}\bar{u}(t) \\ \bar{y}(t) &= \mathbf{C}\bar{x}(t) + \mathbf{D}\bar{u}(t) \end{aligned} \tag{2}$$

where:

$\bar{x}$ -	State Displacement Vector (2n x 1)
$\dot{\bar{x}}$ -	State Velocity Vector (2n x 1)
$\mathbf{A}$ -	State Transition Matrix (2n x 2n)
$\mathbf{B}$ -	State Input Matrix (2n x m)
$\mathbf{C}$ -	State Output Matrix (p x 2n)
$\mathbf{D}$ -	Feedforward Matrix (p x m)
n -	Number of Identified Modes
m -	Number of Inputs
p -	Number of Outputs

In discrete format, this equation becomes:

$$\begin{aligned} \bar{x}(k+1) &= \mathbf{A}\bar{x}(k) + \mathbf{B}\bar{u}(k) \\ \bar{y}(k) &= \mathbf{C}\bar{x}(k) + \mathbf{D}\bar{u}(k) \end{aligned} \tag{3}$$

where k is the discrete time increment variable.

The goal of system identification when applied to control system design is to identify the **A**, **B**, **C** and **D** matrices which exactly describe the system motion. However, for a continuous system, the number of states, n, is infinite. So, an approximation for a finite value of n, must be made, and the motion can therefore only be approximated. The required accuracy of the approximation is a key factor in the system identification process.

For a large flexible space structure like PACOSS, the primary concern is with the low frequency, non rigid body modes, primarily those under 10 Hz. So, 10 Hz will be the cut-off value for the determination of the number of modes.

Eigensystem Realization Algorithm

The Eigensystem Realization Algorithm was developed in 1985 as a method to determine the modal parameters of large flexible multi-input multi-output space structures. The ERA produces a minimum-order realization given dynamic test data (7:620). This algorithm is based upon the singular values of the Hankel matrix. To create the Hankel matrix, the Markov parameters must first be generated. The Markov parameters are defined in discrete time by:

$$\overline{Y}(k) = \mathbf{C}\mathbf{A}^{k-1}\mathbf{B} \tag{4}$$

The Hankel matrix is:

$$H_{rs}(k-1) = \begin{bmatrix} \overline{Y}(k) & \overline{Y}(k+t_1) & \cdots & \overline{Y}(k+t_{s-1}) \\ \overline{Y}(j_1+k) & \overline{Y}(j_1+k+t_1) & \cdots & \overline{Y}(j_1+k+t_{s-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \overline{Y}(j_{r-1}+k) & \overline{Y}(j_{r-1}+k+t_1) & \cdots & \overline{Y}(j_{r-1}+k+t_{s-1}) \end{bmatrix} \quad (5)$$

where $j_i(i=1,...,r-1)$ and $t_i(i=1,...,s-1)$ are arbitrary integers defined such that r is the number of modes to be determined (both system modes and computational, or noised derived, modes), and s is larger then r (generally 3-5 times the value of r).

The first step to compute the Hankel matrix is to determine the velocity impulse response given the transfer function generated from acceleration data. This transfer function is defined as:

$$G_a(s) = \frac{Y(s)}{U(s)} \quad (6)$$

where:

$$Y(s) = a_n s^n + a_{n-1} s^{n-1} + \ . \ . \ . + a_1 s^1 + a_0 \quad (7)$$

$$U(s) = b_n s^n + a_{n-1} s^{n-1} + \ . \ . \ . + b_1 s^1 + b_0 \quad (8)$$

which is a rational transfer function. However, in state space form, a rational transfer function will have a non-zero **D** matrix. The ERA method assumes a zero **D** matrix.

But, by integrating the transfer function (which is multiplying by 1/s in the Laplace domain), the transfer function becomes non-rational and the **D** matrix becomes zero. Numerically, this integration is:

$$H_v(j\omega) = \frac{H_a(j\omega)}{j\omega} \tag{9}$$

Then the impulse response, $h_v(t)$, is found by taking the inverse Fourier transform of the frequency response, $H_v(j\omega)$. From this data, the Markov parameters, and hence the Hankel matrix, can be formed.

Once the full order Hankel matrix is determined, the next step is to achieve a minimum realization by determining the order of $\mathbf{H}_{rs}(0)$. This is done by using the singular value decomposition, defined as:

$$\mathbf{H}_{rs}(0) = \mathbf{P}\mathbf{D}_{sv}\mathbf{Q}^T \tag{10}$$

where **D** is a diagonal matrix of the singular values of $\mathbf{H}_{rs}(0)$ in descending order, and **P** and **Q** are isometric matrices. The order of $\mathbf{H}_{rs}(0)$ is determined by the rank of **D**. Unfortunately, for a real system which has noise present, the singular values may never be zero. In this case, it is necessary to determine some cutoff value below which all the singular values are small relative to the rest of the singular values, and can be assumed zero. This then becomes the rank of $H_{rs}(0)$.

The minimum realization for the system is then:

$$x(k+1) = D_{sv}^{-1/2}P^{T}H_{rs}(1)QD_{sv}^{-1/2}x(k)+D_{sv}^{1/2}Q^{T}Eu \qquad (11)$$

$$y(k) = E^{T}PD_{sv}^{1/2}x(k) \qquad (12)$$

where $E^{T}=[I_p, 0_p, \ldots 0_p]$ (6:622).

The final step is to determine the validity of each of the modes, and remove the noise created (computational) modes, leaving only the actual (system) modes. This will result in a minimum order system of purely system modes. This order reduction is done by defining the parameter called the Extended Modal Amplitude Coherence (EMAC). This is a combining of the controllability and observability of each mode, presented as a percent of completely controllable and observable. The EMAC is a function of the singular values, the modal participation factors and the mode shapes (6:623) and is used to help separate the computational modes from the system modes. The primary assumption behind the EMAC is that only system modes will be controllable and observable. Therefore, the computational modes will have significantly lower values for the EMAC. This allows for quick and easy separation of the system and computational modes. Additionally, each system mode has a different level of controllability and observability, resulting in different value of the EMAC for each mode. Given a limit to the number of system modes which can be kept, this provides a quantitative measurement to determine which modes should be kept and which modes can be discarded.

Backwards Least Squares

The second method used for system identification was a multi-input, multi-output Backwards Least Squares technique developed by Hollkamp (10:549). The forward least squares method is based upon the fact that a time-series model can be expressed in the form:

$$\begin{aligned}\mathbf{y}(k) &= -\mathbf{A}_1\mathbf{y}(k-1)-\mathbf{A}_2\mathbf{y}(k-2)-\ldots-\mathbf{A}_p\mathbf{y}(k-p)\\ &\quad +\mathbf{B}_0\mathbf{u}(k)+\mathbf{B}_1\mathbf{u}(k-1)+\ldots+\mathbf{B}_p\mathbf{u}(k-p)\end{aligned} \tag{13}$$

where $\mathbf{y}(k)$ is the kth output response, $\mathbf{u}(k)$ is the kth input, p is the order of the model, the $\mathbf{A}_1$, $\mathbf{A}_2$, ..., $\mathbf{A}_n$ and $\mathbf{B}_1$, $\mathbf{B}_2$, ..., $\mathbf{B}_n$ matrices are the matrices of model parameters (10:549). Then, for an over determined system, the solution is:

$$Y = \Gamma\theta \tag{14}$$

where Y is the vector of output data, Γ is a matrix of output and input data, and θ is the vector of unknown parameters (10:550). For the forward least squares technique, the solution to this problem will be a system model, but it will not be minimum order. That is because θ will include both system (actual) and computational (error created) modes. Order reduction of the system can be simplified by performance of the Backward Least Squares technique (10:550).

Any time-series system can be written in a backwards manner of the form:

$$\mathbf{y}(k) = -\mathbf{C}_1\mathbf{y}(k+1)-\mathbf{C}_2\mathbf{y}(k+2)-\ldots-\mathbf{C}_p\mathbf{y}(k+p) \\ +\mathbf{D}_0\mathbf{u}(k)+\mathbf{D}_1\mathbf{u}(k+1)+\ldots+\mathbf{D}_p\mathbf{u}(k+p) \qquad (15)$$

where the $\mathbf{C}_1$, $\mathbf{C}_2$, ..., $\mathbf{C}_n$ and $\mathbf{D}_1$, $\mathbf{D}_2$, ..., $\mathbf{D}_n$ matrices are the backward parameter matrices. The solution for the unknown parameters, θ, is found the same as for the forward method, using equation 14. Unlike the forward least squares method, the backward method will force all the computational eigenvalues inside the unit-amplitude circle (10:550). This means that only the eigenvalues outside the unit-amplitude circle need be kept, as they are the system modes. In this way, a reduced order system is found. The conversion back to forward least squares (to create a stable system with eigenvalues less than one) is:

$$\mathbf{A}_i = \mathbf{C}_p^{-1}\mathbf{C}_{p-i} \qquad (16)$$

$$\mathbf{B}_i = \mathbf{C}_p^{-1}\mathbf{D}_{p-i} \qquad (17)$$

The new, reduced order $\mathbf{A}_i$ and $\mathbf{B}_i$ matrices can now be used to solve for the new reduce order system modes.

Prediction Error Method

The ERA and BLS methods were developed primarily to develop natural frequencies, damping factors, mode shape matrices and modal participation matrices, (e.g. a modal model of the system) not a state space model. Therefore, an additional technique had to be added to enable development of an accurate system in state space form. If one assumes that the ERA and BLS methods are able to obtain accurate **A** and **B** matrices, then the PEM has the ability to compute the **C** matrix given the **A** and **B** matrices.

The Prediction Error Method is found in the system identification toolbox created by Math Works Inc. for MATLAB (3). This method attempts to minimize the error gradient between the model generated output and the actual system output. Given a set of input-output data, and a model format, with a set of fixed parameters and a set of unknown parameters, this program attempts to vary the unknown parameters along the gradient direction to fit the model results to the known set of data. The accuracy and computation time for this routine vary with model size and the number of unknown parameters.

This technique uses an iterative Gauss-Newton algorithm to solve the equation:

$$\mathbf{A}_{PEM}(q)\bar{y}(t) = \frac{\mathbf{B}_{PEM}(q)}{\mathbf{F}_{PEM}(q)}\bar{u}(t-nk) + \frac{\mathbf{C}_{PEM}(q)}{\mathbf{D}_{PEM}(q)}\bar{e}(t) \qquad (18)$$

Where e(t) is the measurement error, u(t-nk) is the input data, y(t) is the output data and $\mathbf{A}_{PEM}(q)$, $\mathbf{B}_{PEM}(q)$, $\mathbf{C}_{PEM}(q)$, $\mathbf{D}_{PEM}(q)$, and $\mathbf{F}_{PEM}(q)$ are of the form:

$$\mathbf{A}_{PEM}(q) = \mathbf{I} + \mathbf{a}_1 q^{-1} + \ldots + \mathbf{a}_n q^{-n} \qquad (19)$$

This method solves for a particular

$$\theta = [a_1, \ldots a_n, b_1, \ldots b_n, c_1, \ldots c_n, d_1, \ldots d_n, f_1, \ldots f_n] \qquad (20)$$

which minimizes the square of the error between the output, $y_a(t)$, of the actual system and the output, $y_m(t)$, of the model (3:I-7) using a gradient minimization technique. The actual system is assumed be a constant coefficient system.

Method Comparison and Combination

The ERA and the BLS methods were the primary techniques used to compute the system parameters. In addition, since they are different methods which use different types of data to generate the same solution, their results were compared to help determine the validity of the final system model developed.

The *plant* matrix was created by generating the discrete eigenvalues using the equation:

$$z = \exp[(-\zeta\omega_n + j\omega_d)/f_s] \qquad (21)$$

where z is the eigenvalue, ζ is the damping factor, ω_n is the undamped natural frequency, ω_d is the damped natural frequency, and f_s is the sampling rate (2:622). The discrete eigenvalues then become the diagonals of the ***plant*** matrix in discrete time. Since this is a complex matrix, it must be transformed into the real 2x2 block diagonal form. This is then the real **A** matrix which can be used in control system design. The complex modal participation matrix is the continuous time form of the **B** matrix. This **B** matrix must be transformed to real, discrete time form, to be consistent with the **A** matrix.

That leaves only the **C** matrix as an unknown. PEM was used to compute an accurate **C** matrix. This technique becomes more effective the fewer the parameters there are to estimate, since PEM computes a gradient for each unknown parameter and searches along that direction in an attempt at minimization. So with the **A** and **B** matrices as known parameters, all that PEM needs to approximate is the **C** matrix. By approximating the PACOSS as a linear system, then each output is independent of the other outputs. Then, PACOSS can be approximated as multiple single output systems. The number of parameters PEM needs to estimate is then equal to a single row of the **C** matrix. So, PEM was performed 8 times for each of the 8 outputs, then recombined into a single **C** matrix of the form:

$$\mathbf{C} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_8 \end{bmatrix} \tag{22}$$

This reduced the number of parameters PEM needed to estimate, while providing an accurate **C** matrix of the system. The combined methods of ERA **A** and **B** determination and PEM **C** determination will be termed ERA/PEM in the remainder of the thesis. The combined method of BLS **A** and **B** determination and PEM **C** determination will be termed BLS/PEM in the remainder of the thesis.

III. Method Validation

Before attempting to identify the PACOSS DTA, the accuracy of the identification methods selected were validated on a known truth model. This required a spring-mass-damper system with known *plant*, *input* and *output* to be created. ERA/PEM and BLS/PEM were then applied to this known system and the models developed were compared to the truth system to determine the accuracy of each method.

Truth Model

To determine the accuracy and applicability of the system identification techniques selected, it was decided to apply them to a low order "truth" model first. This would allow a comparison of each model against known **A**, **B**, **C**, and **D** matrices. To accurately determine the applicability of the system identification techniques selected, the truth model must contain the same characteristics as the system to be identified, in this case, the PACOSS DTA. These characteristics include multi-input multi-output and closely spaced poles at or below 10 Hz with low damping (2:15). As a result, a spring-mass-damper system of the form of equation (1) was developed (Figure 2). This was a 2 input, 2 output system with masses of:

m1	=	0.61743	kg
m2	=	0.19957	kg
m3	=	1.6582	kg

damping of:

c1	=	0.031633	kg/sec
c2	=	0.32738	kg/sec
c3	=	0.32918	kg/sec
c4	=	0.019153	kg/sec

and spring constants of:

k1	=	52.058	N/m
k2	=	0.89183	N/m
k3	=	19.756	N/m
k4	=	178.08	N/m

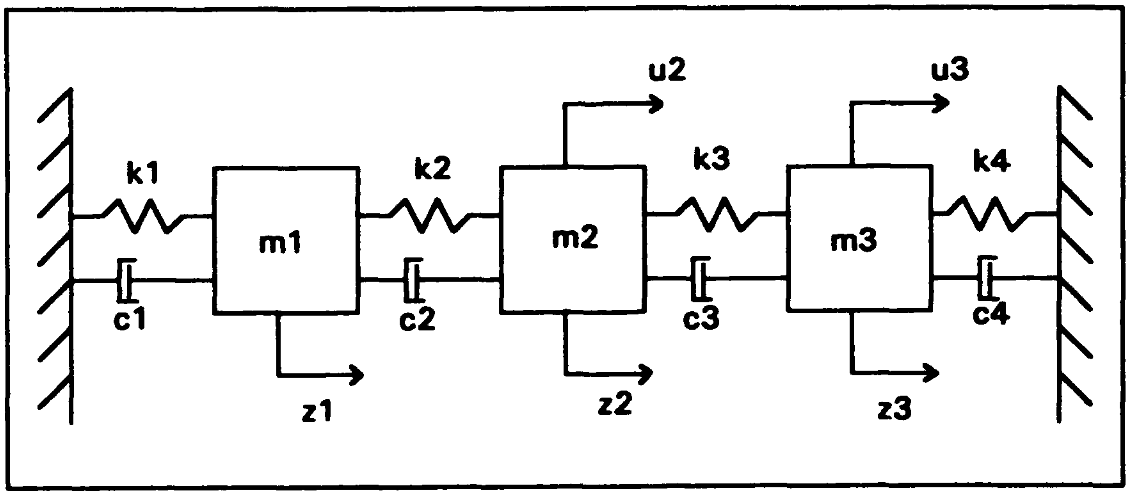


Figure 2: Truth Model
Physical System

The outputs were position measurements of each mass. They were combined into discrete time state space format

(Equation 3). The natural frequency and damping factors for a sampling rate of 14.2 samples per second are presented in Table 1.

Table 1: Truth Model Modes

ω_n (Hz)	ζ (%)
1.4414	2.0162
1.4537	9.9314
1.8817	8.0431

From which the following discrete time, state space system was developed:

$$\mathbf{A} = \begin{bmatrix} 0.7999 & 0.0005 & 0.0026 & 0.0639 & 0.0012 & 0.0001 \\ 0.0022 & 0.7766 & 0.1950 & 0.0036 & 0.0575 & 0.0084 \\ 0 & 0.0255 & 0.7250 & 0 & 0.0010 & 0.0630 \\ -5.4780 & -0.0279 & 0.1048 & 0.7647 & 0.0305 & 0.0045 \\ -0.0542 & -5.8419 & 4.6834 & 0.0944 & 0.5910 & 0.2881 \\ 0.0015 & 0.6453 & -7.4133 & 0.0017 & 0.0347 & 0.7134 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0.0001 & 0 \\ 0.0109 & 0.0001 \\ 0.0001 & 0.0014 \\ 0.0059 & 0.0001 \\ 0.2881 & 0.0051 \\ 0.0051 & 0.0.380 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Which are the **A**, **B**, and **C** matrices defined in Eq. (3). The **D** matrix is zero.

Truth Model Identification

The identification of a known truth model was the first step in analyzing the effectiveness of the system identification techniques chosen. For the ERA method, an impulse response of the truth model was generated. A zero mean Gaussian measurement error (RMS value of 0.1) was added to the output. The ERA method used a 20 column (10 modes) and 60 row (58 data points for 4.06 seconds of data) Hankel matrix. Only the 18 largest singular values of the Hankel matrix were retained, with the two remaining singular values discarded due to there relatively small value.

In a similar manner, the BLS method had a random signal of zero mean Gaussian noise which had an RMS value of 1, used as an input to the truth model. A measurement error of zero mean Gaussian noise with an RMS value of 0.01, was added to the output. The resulting natural frequencies and damping factors computed by each method are shown in Table 2 along with the actual natural frequencies and damping factors of the system.

From these results, it is evident that the ERA method was able to identify the actual modes of the system with a great deal of accuracy, despite having fairly high measurement errors (approximately 10% of the input). It was even able to separate and identify the closely spaced modes.

Table 2: Identified Truth Model Modes

Actual		ERA		BLS	
ω_n (Hz)	ζ (%)	ω_n (Hz)	ζ (%)	ω_n (Hz)	ζ (%)
1.4414	2.0162	1.4414	2.0162	1.4063	6.4529
1.4537	9.9314	1.4537	9.9314	-	-
1.8817	8.0431	1.8817	8.0431	1.9066	3.4481

The BLS technique did not identify the modes or damping factors as accurately as the ERA technique, despite having considerably lower measurement error noise. This is discouraging since the PACOSS DTA has several closely spaced poles, and a considerable amount of noise.

Next, the **A** and **B** matrices generated by each method were inserted into PEM to compute the **C** matrix. This gave a complete state space model of the truth system (Appendix A). Now, the time and frequency response outputs for each model were compared to those generated by the truth model (Appendix A). Figure 3 shows that the ERA/PEM method produces a virtually identical frequency response (dashed line) to the truth system (solid line) below 5 Hz, with some error entering into the system between 5 and 10 Hz. What is important to note is the accuracy with which the poles and zero were identified.

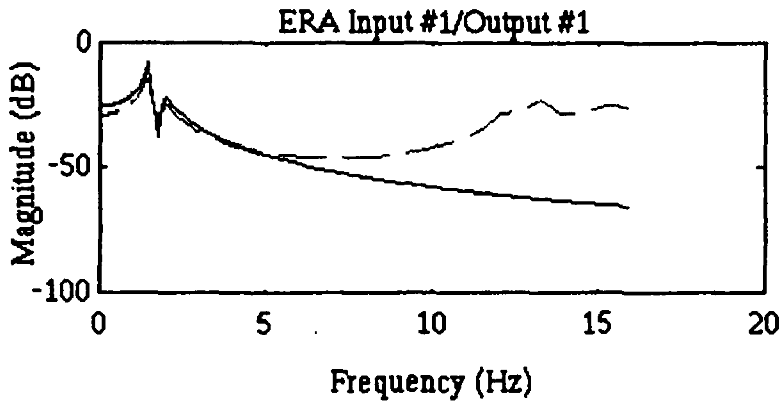


Figure 3: ERA/PEM Magnitude Plot

It is also important to note that the ERA method without PEM did not identify the zeros of the system correctly. As can be seen in Figure 4, the ERA method alone added a zero not found in the actual system. The PEM routine was needed to accurately identify the system zeros.

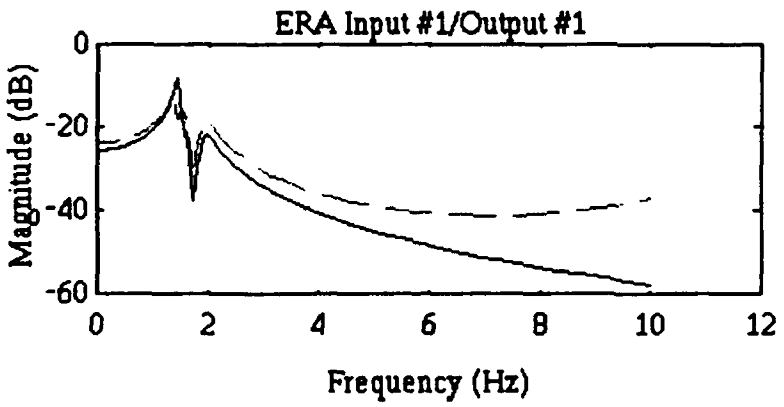


Figure 4: ERA Magnitude Plot

The results of the BLS/PEM method were not as encouraging. By examining the frequency response plots, it is evident that the BLS/PEM model identified two closely spaced poles as a single pole (Figure 5). The poles and zeros of the BLS model (dashed line) did not match up well with those of the true system (solid line), and the BLS model added a zero at 6 Hz, and failed to identify the pole at 1.4537 Hz. The resulting magnitude plot shows significant error. This was with only 1% measurement error.

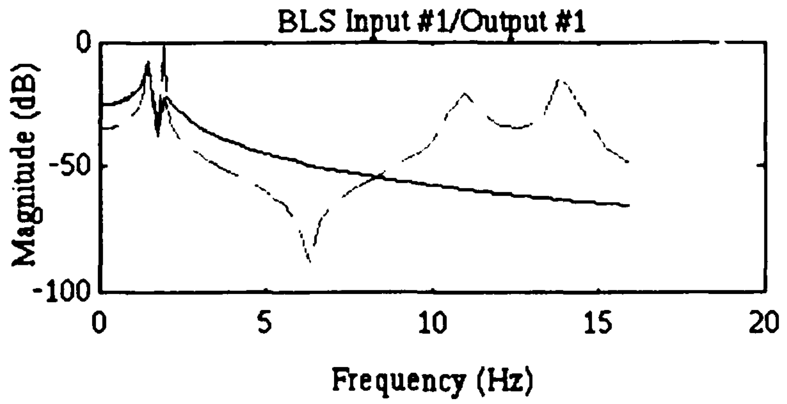


Figure 5: BLS/PEM Magnitude Plot

Appendix A contains the state space model developed by both ERA/PEM and BLS/PEM, as well as all 8 bode plots.

Once the system identification techniques had been tested on the truth model, the next step was to use these techniques to compute a state space model for the PACOSS DTA.

IV. Experimental Procedure

The identification of the system model for the PACOSS DTA was divided into three phases. The first phase was to evaluate different time domain system identification techniques to determine which was the most appropriate. The results of phase one are detailed in Chapter III. The second phase was to actually acquire the input and output data from the PACOSS system. While the third phase was to run the actual data through the chosen system identification techniques to determine the system model for the PACOSS DTA.

Data Generation

Once the accuracy of each identification technique was determined, the actual PACOSS input/output data was then generated. First, all equipment (in particular the accelerometers) had to be calibrated to identify any bias allowing the generation of accurate and consistent data. A Techtronix Digital Analyzer 2642 was used to generate the random input signals and to record the output from the PACOSS. It also computed and recorded frequency response functions for individual input/output pairs of the PACOSS DTA. One limitation of the Digital Analyzer is that it can only simultaneously record 2 channels of data. The PACOSS system is 8 input and 8 output, therefore this causes some difficulty in acquiring the data. Since the BLS and PEM techniques require knowing the input corresponding to each output, 1 input and 1 output (velocity) channel were recorded, requiring 64 separate data runs. This allowed the combination of all 64 data samples to be combined into a single 8 input 8 output model.

The next critical decision was to determine the sampling rate. Sampling rate is important because it determines the maximum frequency of the system which can be identified (4:378). In addition, a high sampling rate is necessary to distinguish between closely spaced poles. However, it is important to remember that the higher the sampling rate, the more data there is generated. This is an important trade-off since there are 64 input/output combinations for which data must be generated. In order to generate sufficient data yet maintain a Nyquist frequency in the 10 Hz region, the random input response data was acquired at 25.6 Hz for 20 seconds of data. The frequency response function plots were generated for data from 0 to 10 Hz, using the *Sweptsine* sub function on the 2642 Analyzer with a step size of 0.025 Hz collecting 401 points of data. This data was collected by Lieutenant Chad Matheson as part of his thesis, in 1992 (8).

Since the data collected by Lieutenant Matheson was a set of frequency response functions created from acceleration data, and the ERA method requires impulse response data, with output measurements of velocity, not acceleration, a transformation of the data had to be made. The first step was to integrate the frequency response function data. In the frequency domain, this becomes:

$$H_v(j\omega) = \frac{H_a(j\omega)}{j\omega} \qquad (23)$$

The primary problem encountered was to determine $H_v(j\omega)$ at $\omega = 0$ Hz (DC). Since most of the data below 1 Hz was not very accurate, and the modes below 1 Hz include primarily rigid body and pendulum modes which will not be controlled, it was decided to make $H_v(0) = 0$. This is the same method used by the MATH subroutine on the Techtronix Digital Analyzer 2642.

Once the frequency response functions with respect to velocity were created, the inverse fourier transform of the data, could then be taken, again using a MATLAB script file (see Appendix D). This script file used the IFFT function in MATLAB for 2048 points, and multiplied the results by the time scaling factor, which is 2048. This then provided the velocity impulse response which the ERA method requires.

The other set of data required was the random input data. This data was generated by sending a random input signal to the proof mass actuators to excite the DTA. The acceleration output signal thus generated was sent through the analog integrator located in the motor control units, and was recorded by the Analyzer. However, only the collocated velocity data was valid since the velocity signal for the non-collocated measurements was so small as to be indistinguishable from the ambient noise of the system. This was true only for velocity data gathered from the Analyzer. This is because although the accelerations are large, they occur for only a short time in each direction, resulting in small velocities. The accelerations for the collocated data are an order of magnitude larger than for non-collocated data, resulting in collocated velocity data which is distinguishable from the ambient noise of the system. For this reason, the data inserted into PEM to generate the **C** matrix only included the collocated rate data.

Model Generation

With the data generated, the next step is to identify the system model. The impulse response data was inserted into the ERA software while one set of random input responses (and corresponding input signals) were inserted into the BLS software. Both methods generated natural frequencies, damping factors, mode shape matrices and modal participation matrices. The *plant* matrix was generated from the natural frequencies and damping factors while the *input* matrix was generated by taking the discrete transformation of the modal participation matrix. These matrices were then transformed to real matrices (2x2 block diagonal form for the **A** matrix).

The only matrix left to identify was the *output* matrix. This was were the PEM method was employed. Another set of random input response data was inserted into the PEM routine (see Appendix D). The elements of the **A** and **B** matrices were fixed, while the PEM routine was allowed to determine the elements of each row of the **C** matrix (given the real form of the mode **C** matrix derived by the ERA or BLS procedure as an initial condition). By fixing the **A** and **B** matrices, the PEM routine was able to accurately determine the **C** matrix because of the relatively few numbers of variable parameters it had to determine.

The final step in the system identification process was to test the accuracy of the model. Since the numerous errors associated with time domain data make comparison in the time domain difficult, and because small errors in the frequency domain (such as not identifying one pole or zero) translate to large errors in the time domain, it was decided that only a comparison of the system transfer functions, and no comparison of time domain output would be made. A comparison of the frequency response of each model was compared to the frequency response functions of the DTA. The PACOSS system frequency response functions were the same velocity transfer functions which were used earlier in the generation of the impulse response data for the ERA. The results of the PACOSS identification are presented in Chapter V.

With this identification accomplished, there is a *plant*, *input*, and an *output* matrix describing the motion of the PACOSS DTA. This state space model can be used to design an active control system for the PACOSS. Additionally, there is a comparison of two system identification techniques for their accuracy in modeling large, multi-input multi-output, undamped space structures with multiple, closely spaced, low frequency modes.

V. PACOSS IDENTIFICATION

With the model identification techniques selected, and a validation of their ability to generate accurate state space models, the final step is to generate a state space model for the PACOSS DTA. Both the ERA/PEM and BLS/PEM methods were used, and the results were compared to the frequency response of the PACOSS DTA to determine the accuracy of model thus developed.

ERA Identification

Since the ERA method produced an extremely accurate model of the truth system, it was the first method employed to identify the PACOSS DTA. The impulse response data was inserted into the ERA/PEM software, using a Hankel matrix that had 90 columns (45 modes) and 400 rows (79 data points for 1.66 second of data). The singular value cutoff was at the 86th singular value. The **A** and **B** matrices thus created were inserted into PEM and a **C** matrix was computed. Appendix B contains the state space model, in discrete time form, developed by ERA/PEM.

As can be seen in Table 3 ERA/PEM identified 10 modes under the 10 Hz cutoff. These modes were closely space and they all had low damping ratios.

Table 3: ERA/PEM Identified
PACOSS Modes

ERA/PEM	
ω_n (HZ)	ζ (%)
1.5258	5.1503
2.8969	2.0358
3.9228	1.2975
4.7210	1.7325
4.9421	0.9926
6.9764	5.3497
7.2674	0.4309
9.1699	0.4153
9.2688	4.0237
9.6443	1.8441

In order to test the accuracy of these modes, a comparison was made to the experimentally obtained frequency response function plots from the PACOSS. Figure 6 is a comparison of the PACOSS frequency response plots (solid line) and the model developed using the **C** matrix generated by ERA only (dashed line). It can be noted that few zeros were identified and many of the poles were canceled out by the zeros that were identified.

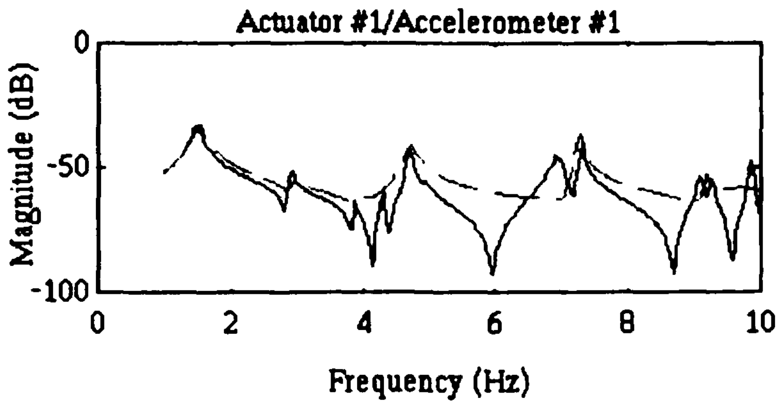


Figure 6: ERA Magnitude Plot for Actuator 1/Accelerometer 1

Figure 7, is a comparison of the PACOSS frequency response plots (solid line) and the model which results from the PEM developed C matrix (dashed line). This is a more accurate match with no poles canceled by zeros. Appendix C contains the graphs of all the actuator/accelerometer pair comparisons between each PACOSS frequency response plot and its corresponding ERA/PEM model frequency response plot.

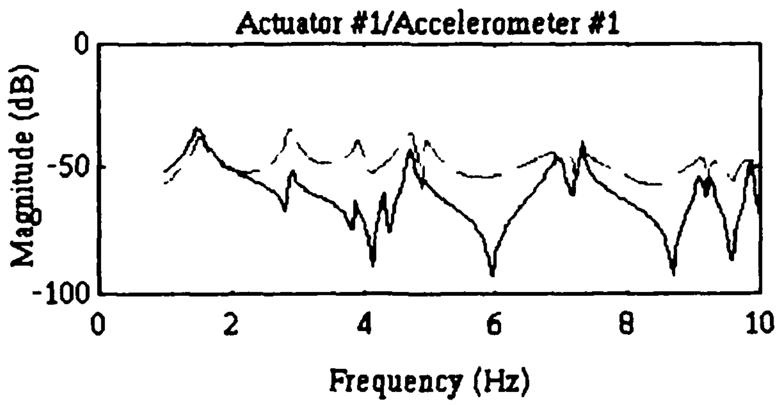


Figure 7: ERA/PEM Magnitude Plot for Actuator 1/Accelerometer 1

The ERA/PEM method shows a good correlation between 1 Hz and 10 Hz for the system poles, but not for the system zeros. The ERA/PEM method identified almost all the poles, and many of the zeros. Which specific poles and zeros it identifies accurately varies with the actuator/accelerometer pair being examined. As can be seen in Figure 7, for the actuator 1/accelerometer 1 pair, most of the poles are identified by ERA/PEM quite accurately, but 4 significant zeros are not identified. This is typical of the system model developed (see Appendix C for other actuator/accelerometer pairs).

What this signifies is that the **A** matrix for the system model is accurate and could be used in control system design. The **B** and **C** matrices, however, are not accurate. They do identify some zeros, but they miss several significant zeros. Overall, the ERA/PEM method does a good job of modeling a very complex system, but the model developed would not be accurate enough to enable control system design.

BLS Identification:

Although the BLS method was not as accurate as the ERA method in identifying the truth model, in particular when there was a large measurement noise, it was thought that BLS could help to confirm some of system modes developed by ERA. For this method, the set of random input data was used to develop the system model. As before, the **A** and **B** matrices were inserted into PEM to compute a C matrix and provide a complete state space model of the PACOSS DTA.

As can be seen in Table 4, the BLS method identified 10 modes. These modes were grouped around 1.5 Hz and 12 Hz, with most having damping ratios in excess of 30%.

Table 4: BLS/PEM Identified PACOSS DTA Modes

BLS/PEM	
ω_n (HZ)	ζ (%)
0.7202	81.3024
1.2406	58.2258
1.5165	62.8178
1.4941	59.9581
1.5001	53.0558
1.3518	30.8457
1.4993	40.9521
12.6124	3.5030
12.6761	1.6485
12.8435	8.2277

These modes are very different from the actual system response as seen in Figure 8, where the modes are spread out from 2 Hz to 10 Hz with damping ratios under 5%. As can be seen, there is almost no correlation between the BLS/PEM frequency response (dashed line) and the actual PACOSS DTA frequency response (solid line).

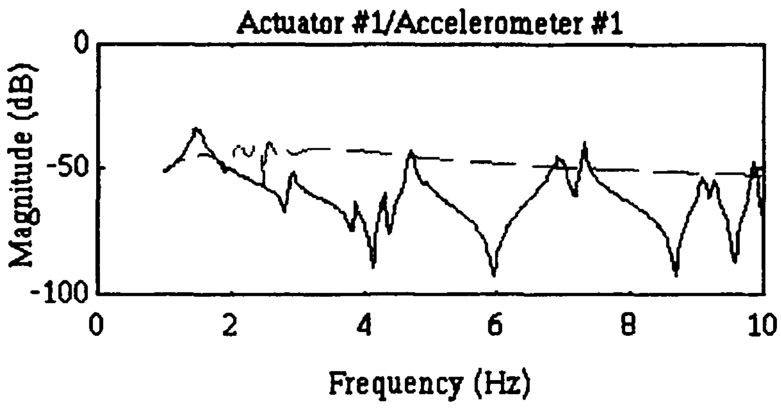


Figure 8: BLS Magnitude Plot for Actuator #1/Accelerometer #1

This lack of correlation leads to the conclusion that the state space model developed using the BLS/PEM method was not accurate enough to enable the development of a control system.

Sources of Error

The primary reason the model did not exactly match the PACOSS DTA in the frequency domain was because of the numerous errors inherent in the system identification process, in addition to the errors particular to this specific experiment. Inherent in the attempt to identify any real world system is the fact that any real world system is a continuous system with an infinite number of modes. However, a model is approximated by a system of discrete point masses and can only have a finite number of modes. The attempt is made to retain the modes which have the most effect on the system behavior. But, the fewer modes retained, the less accurate the model. The ERA/PEM method generated a model with 10 modes. This is far from an infinite number of modes, but most of the power is located in these modes.

Another set of errors inherent in any system identification process is the noise introduced into the response measurements. This noise can come from a number of sources and includes accelerometer noise and bias, errors measuring the inputs, air currents, modes introduced by the attachment points between the experiment and the environment, and rigid body or pendulum modes. These noises all very in magnitude, and are not always zero mean Gaussian. This makes them difficult to predict and remove from the data before processing. Since most system identification models attempt to account for zero mean Gaussian noise, noise which does not fit this description will effect the system model. The result is that system identification techniques model this non-zero mean non-Gaussian noise as additional system modes, therefore creating a model with more modes than the actual system.

For this particular experiment, there were several known sources of noise. The first, and probably the most significant, was accelerometer noise and bias. For velocity data, the accelerometer ambient noise level was on the order of 10 mVolts, with a steady state bias ranging from -100 mVolts to -400 mVolts. This was significant because the non collocated velocity data had velocity values on the order of 10 mVolts. This meant that none of the non collocated velocity measurements could be used because the data could not be distinguished from the noise. Additionally, the PACOSS air bearing isolation system has a known natural frequency of 0.6 Hz, which was accounted for, and an unknown number of additional frequencies which introduce noise into the output response. Additionally, there were low frequency pendulum modes which effected the data. For this reason, the data below 1 Hz was viewed with skepticism.

VI. Conclusions and Recommendations

The final result of this research is that the system identification methods used were unable to create a state space model with sufficient accuracy to enable the design of an active control system for the PACOSS DTA. They were, however, able to accurately identify the system modes and damping factors, from which the *plant* matrix was created.

Conclusions

The ERA method appears to be a accurate method for determining system modes. It accurately predicted the system poles, as is evident in the frequency domain plots in Appendix C. This is, after all, what it was designed to do. One reason this method had difficulty identifying all the modes at all the sensor locations was due the difficulty it had in identifying the system zeros. For the most part, ERA/PEM failed to identify the system zeros, resulting in a state space model with far fewer zeros than were found in the actual system. The result of this may be that at certain actuator/accelerometer pairs, the ERA/PEM method did not identify a zero, which in the actual system canceled a pole. The model frequency response would then show a pole where the PACOSS frequency response does not, due to pole-zero cancellation.

The inability of the ERA/PEM method to accurately identify all the system zeros was disappointing, but not unexpected, since the method was designed as a modal identification technique, not a state space model development technique. But, for control system design, the location of the zeros is as important as the location of the poles. For this reason, to develop a truly accurate active control system for the PACOSS, some other method will have to be used, instead of, or in conjunction with, the ERA method.

This thesis attempted to supplement the ERA's zeros finding capabilities with the PEM method. For the small truth model, this worked well. But for the larger, and more complex PACOSS system, this method did not work very well. PEM did identify several zeros, but it missed several others. Poor performance in the larger system is most likely due to the method PEM employs in finding zeros. It uses a gradient search to identify unknown parameters of the **A**, **B**, and/or **C** matrix. For this thesis, the parameters of the **C** matrix were identified one row at a time. This means PEM was trying to minimize the gradient for 20 unknown parameters, as opposed to the 6 unknown parameters in the truth model.

To help verify that this set of zeros was the best result PEM could achieve, the **C** matrix was held constant and the **B** matrix was identified one column at a time. The results were that the same set of zeros were identified and were not identified as before. Finally, when both B and C were inserted into PEM and allowed to vary, the results were worse. In this case, most of the zeros that were identified, were identified incorrectly. This is because in this case, a gradient search on 40 parameters was performed. This once again leads to the conclusion that another method of zero identification is needed.

The BLS method did not create an accurate state space model for either the truth model, or the PACOSS DTA. Even in the fairly simple 2 input/2 output, 3 mode truth model, it was only able to identify 2 of the modes, and those modes were not identified very accurately. Additionally, the BLS method demonstrated in the truth model a vulnerability to measurement errors of as small as 1% RMS. This, and the fact that it was unable to identify any of the PACOSS DTA modes accurately suggests that this method may not be suitable for identification of systems of the magnitude of the PACOSS.

Recommendations

The primary recommendation is to continue working with the ERA and other time domain methods to develop an accurate system model of the PACOSS. The system identification, literature as well as the truth model in this thesis, demonstrate that the ERA method is very capable of accurately identifying the system modes. However, for control system design, one needs the *input* and *output* matrices as well. The ERA method is not very accurate at predicting these matrices. Additional work should be done to identify a method that will identify the **A**, **B**, and **C** matrix independently, or in conjunction with ERA, but more accurately than PEM.

Bibliography

1. Rajaram, S. and Junkins, J.L., "Identification of Vibrating Flexible Structures", Journal of Guidance and Control Vol 8, Jul-Aug 1985, pp 463-465.

2. Gehling, R.N., Morgenthaler, D.R., Richards, K.E. Passive and Active Control of Space Structures: Final Report, November 1988 - April 1991. CDRL 14, Contract #F33615-82-C-3222. Denver, Co: Martin Marietta Astronautics Group, 5 June 1991.

3. Pro-Matlab User's Guide. The Math Works Inc. Sherborn, MA, 1987.

4. Ljung, Lennart. System Identification: Theory for the User. New Jersey: Prentice-Hall, Inc, 1987.

5. Franklin, Gene F. and Powell, David J., Digital Control of Dynamic Systems. Addison-Wesley Publishing Company: Reading, MA, 1980.

6. Juang, Jer-Nan, and Pappa, Richard S., "An Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction" Journal of Guidance and Control. Vol 8, Jul-Aug 1985, pp 620-627.

7. George, Scott E., A Modal Analysis and Modelling of a Lightly Damped Large Space Structure, MS thesis, AFIT/GA/ENY/92J-01. Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 1992.

8. Matheson, Chad T., Vibration Suppression in Large Flexible Space Structures using Active Control Techniques, MS thesis, AFIT/GAE/ENY/92D-13. Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1992.

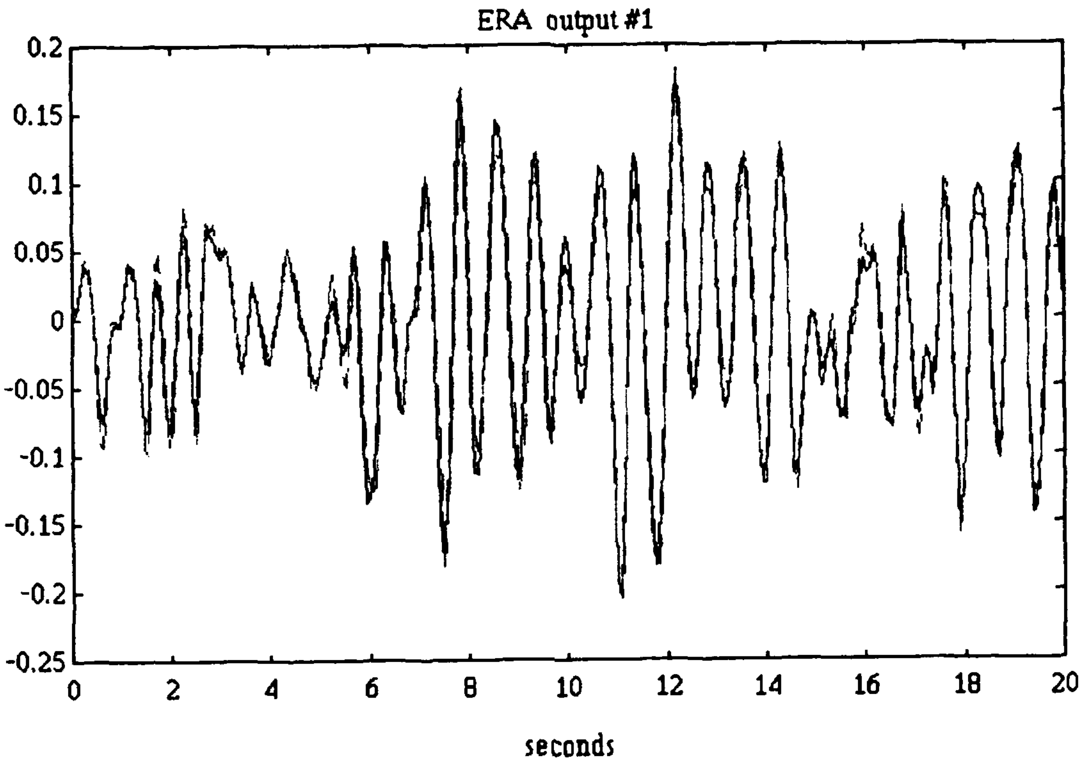
9. Craig, Roy R. Jr., Structural Dynamics: An Introduction to Comuter Methods. New York: John Wiley & Sons, 1981.

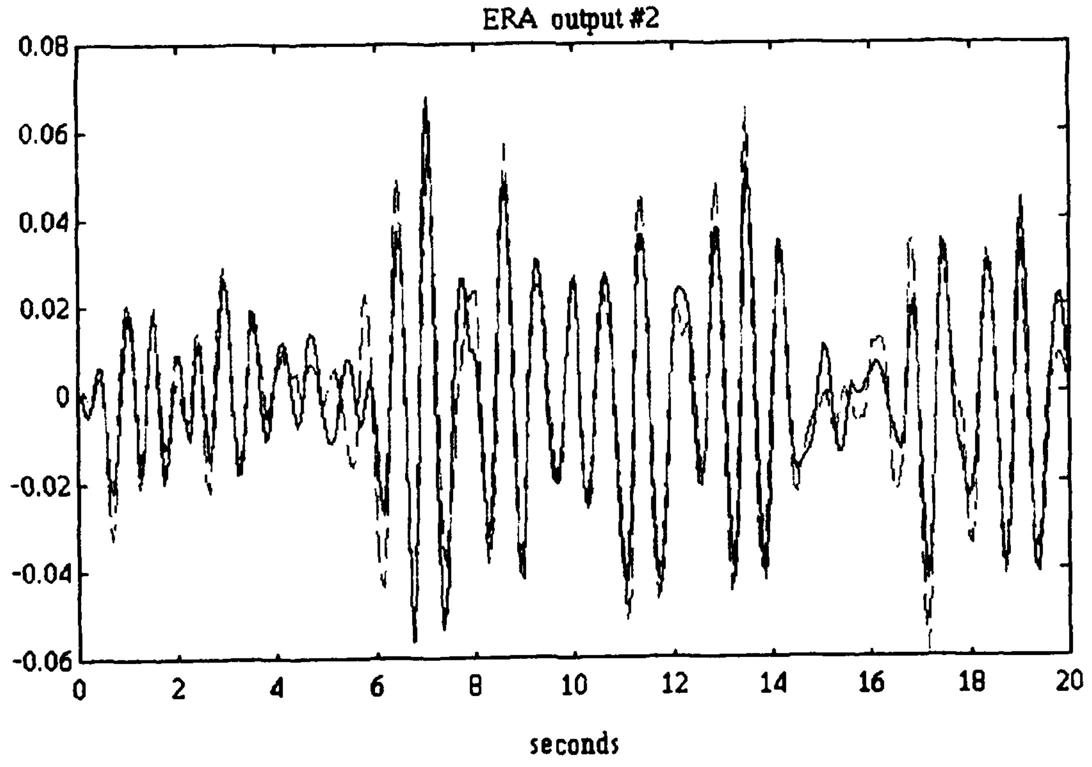
10. Hollkamp, Joseph J., "Multiple-Input, Multiple-Output Time-Series Models from Short Data Records" Journal of Guidance and Control. Vol 16, May-June 1993, pp 549-556.

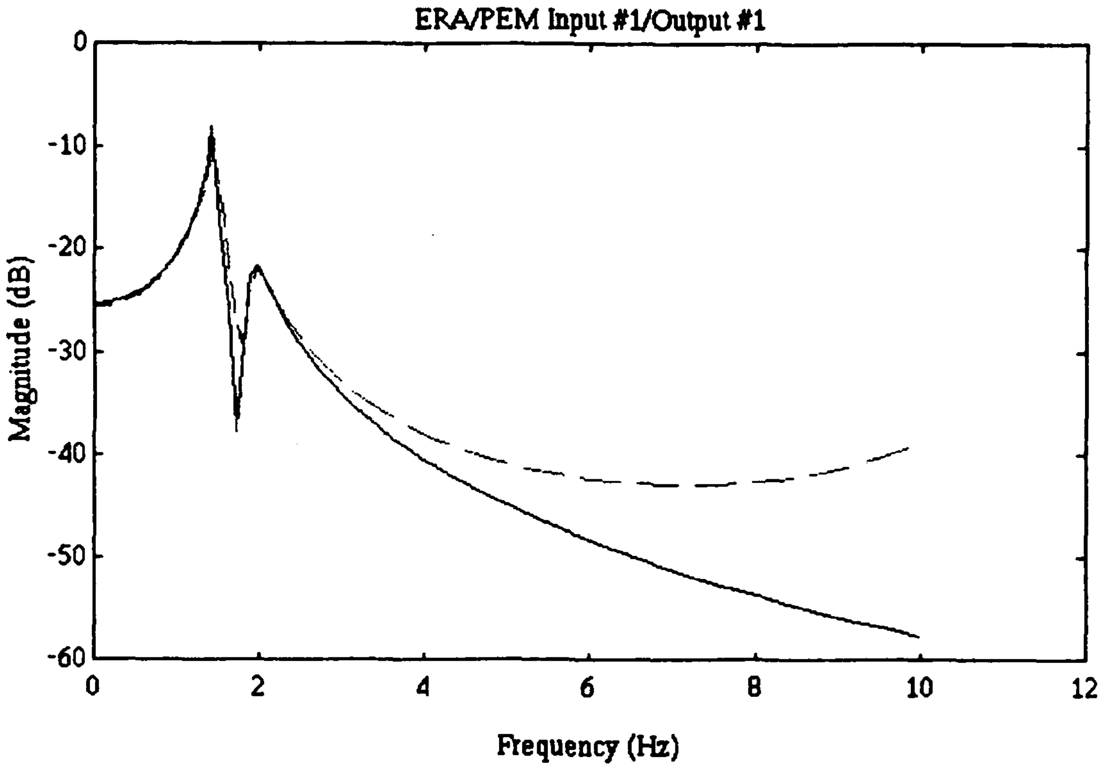
11. Gehling, R.N., Passive and Active Control of Space Structures: Volume 2 Test and Analysis Results, September 1987 - November 1988. CDRL 14, Contract #F33615-82-C-3222. Denver, Co: Martin Marietta Astronautics Group, 30 Sep 90.

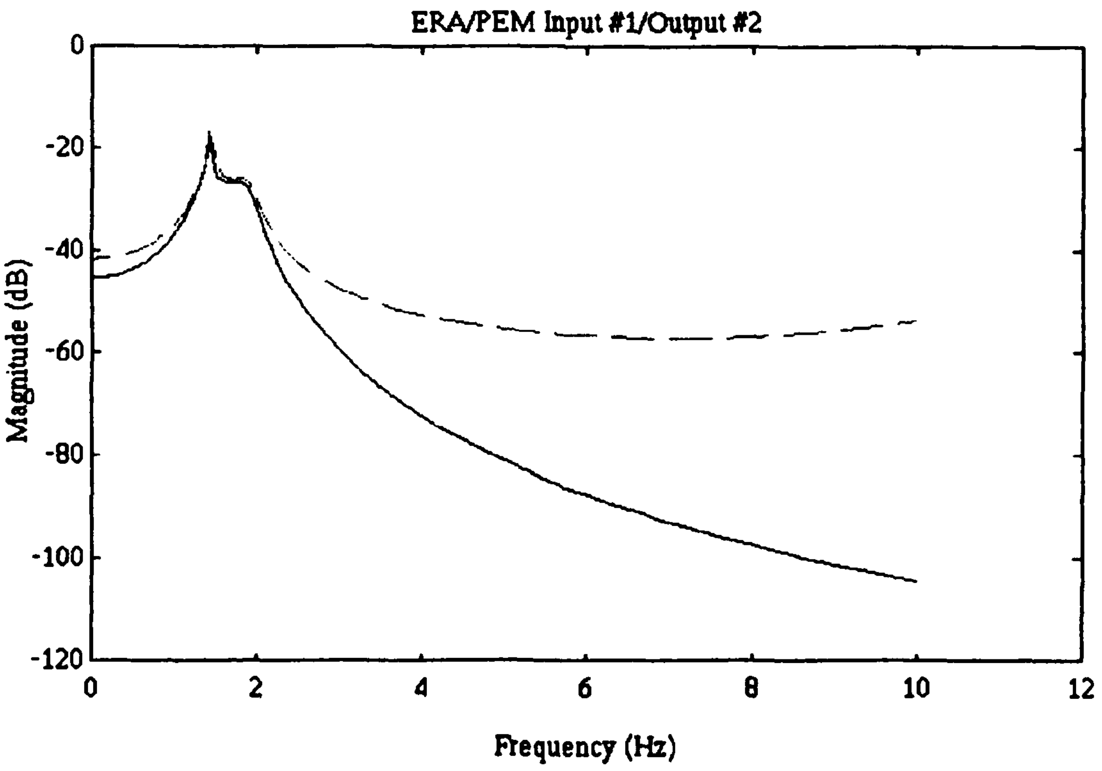
Appendix A. Truth Model

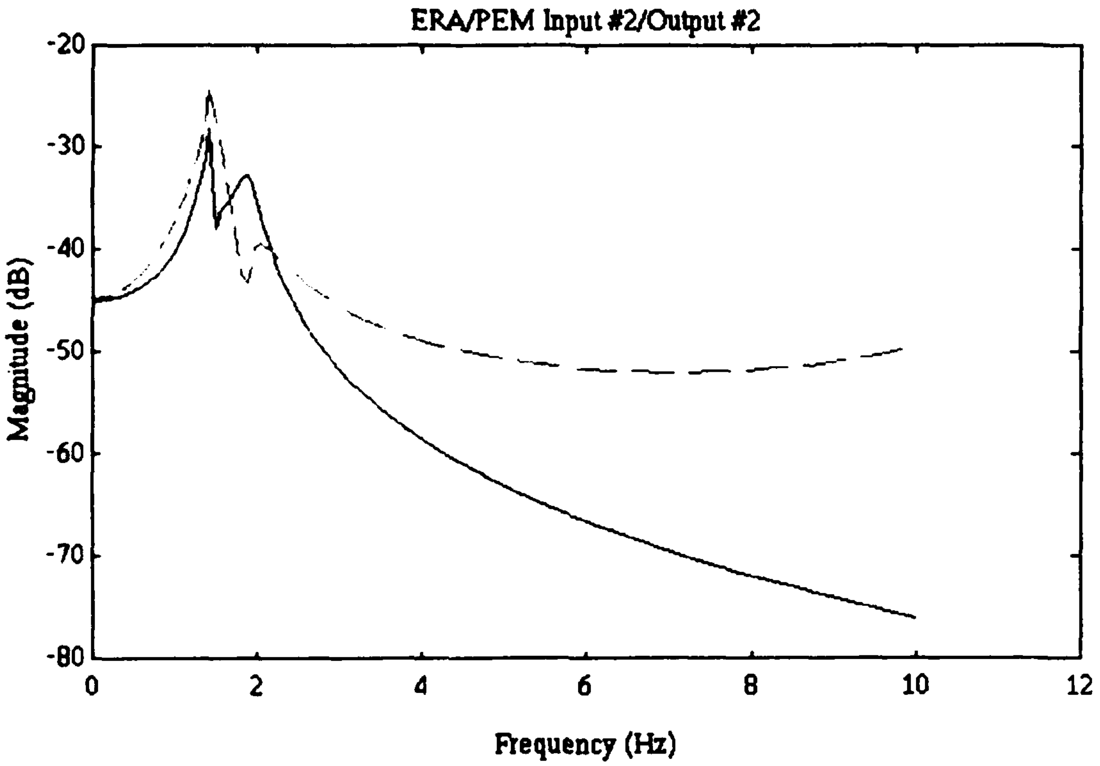
This appendix contains the time domain and bode plots for both the ERA/PEM and the BLS/PEM generated models for the truth system. On all graphs, the truth system is the solid line while the model is the dashed line.

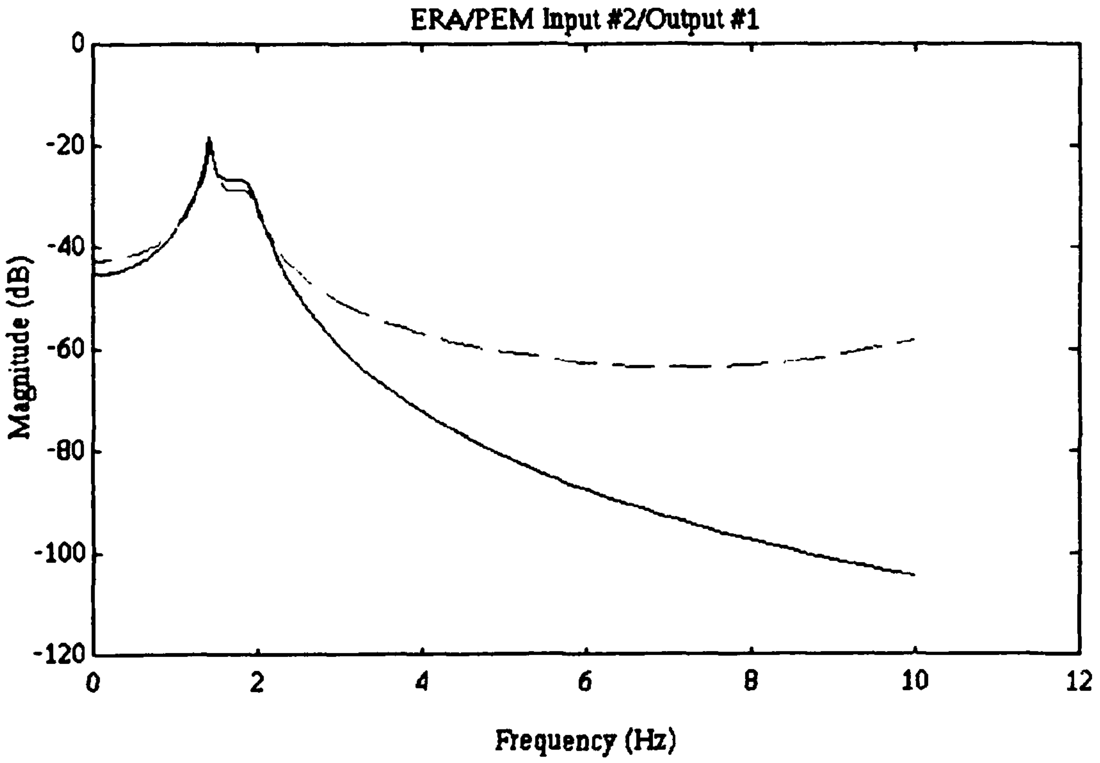

ERA output #1
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
2
4
6
8
10
12
14
16
18
20
seconds

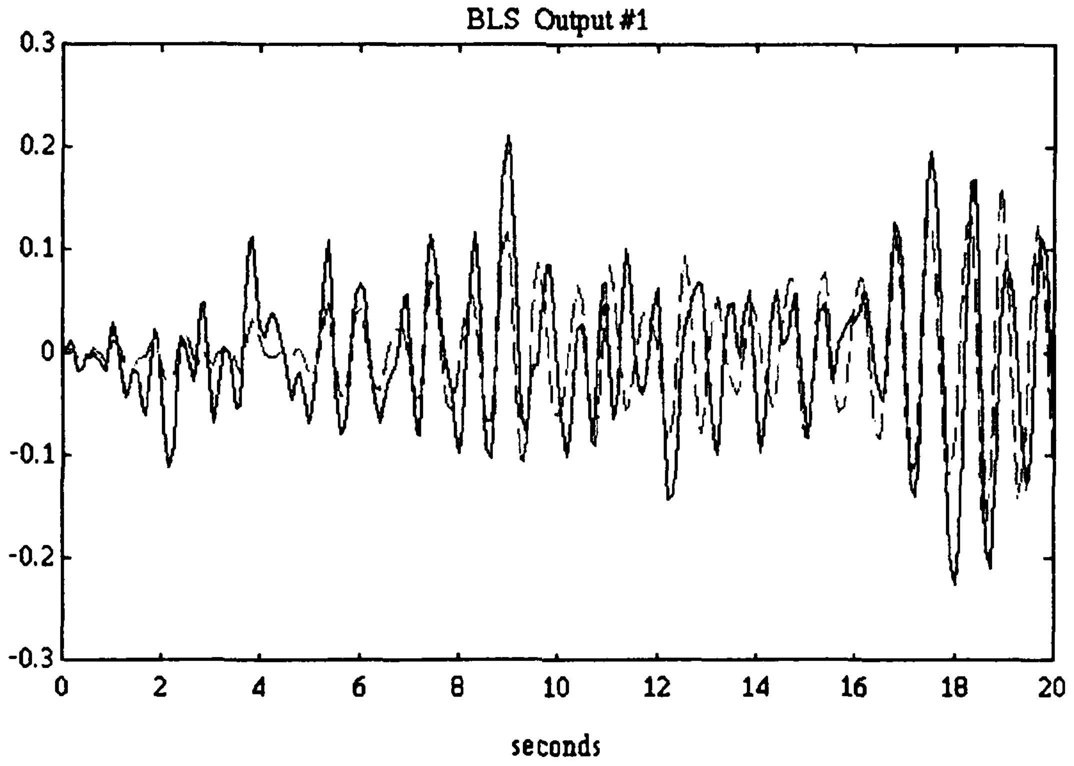

ERA output #2
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
0
2
4
6
8
10
12
14
16
18
20
seconds

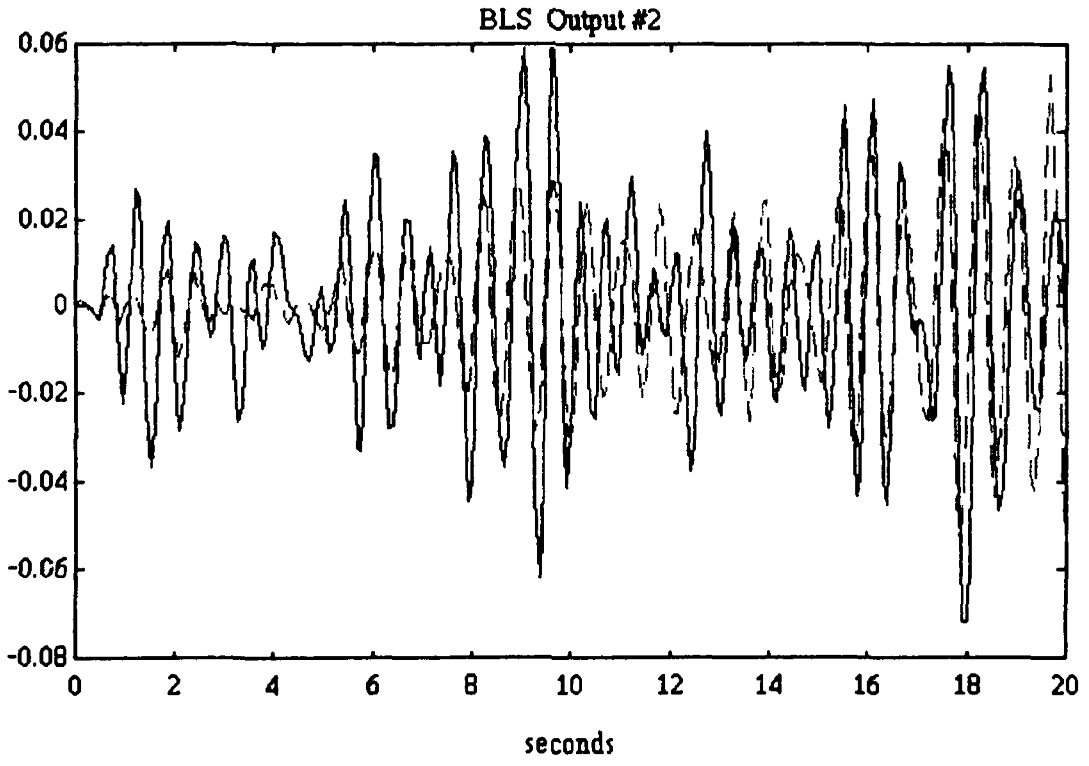

ERA/PEM Input #1/Output #1
0
-10
-20
-30
-40
-50
-60
Magnitude (dB)
0
2
4
6
8
10
12
Frequency (Hz)

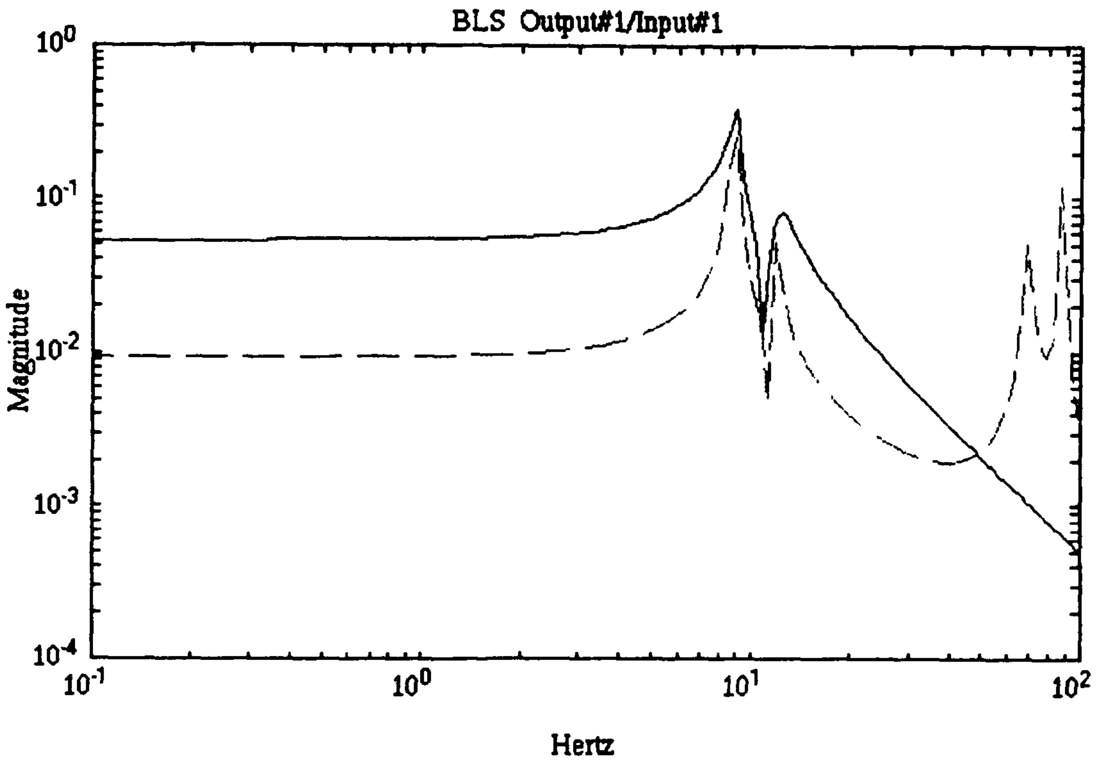

ERA/PEM Input #1/Output #2
0
-20
-40
-60
-80
-100
-120
Magnitude (dB)
0
2
4
6
8
10
12
Frequency (Hz)

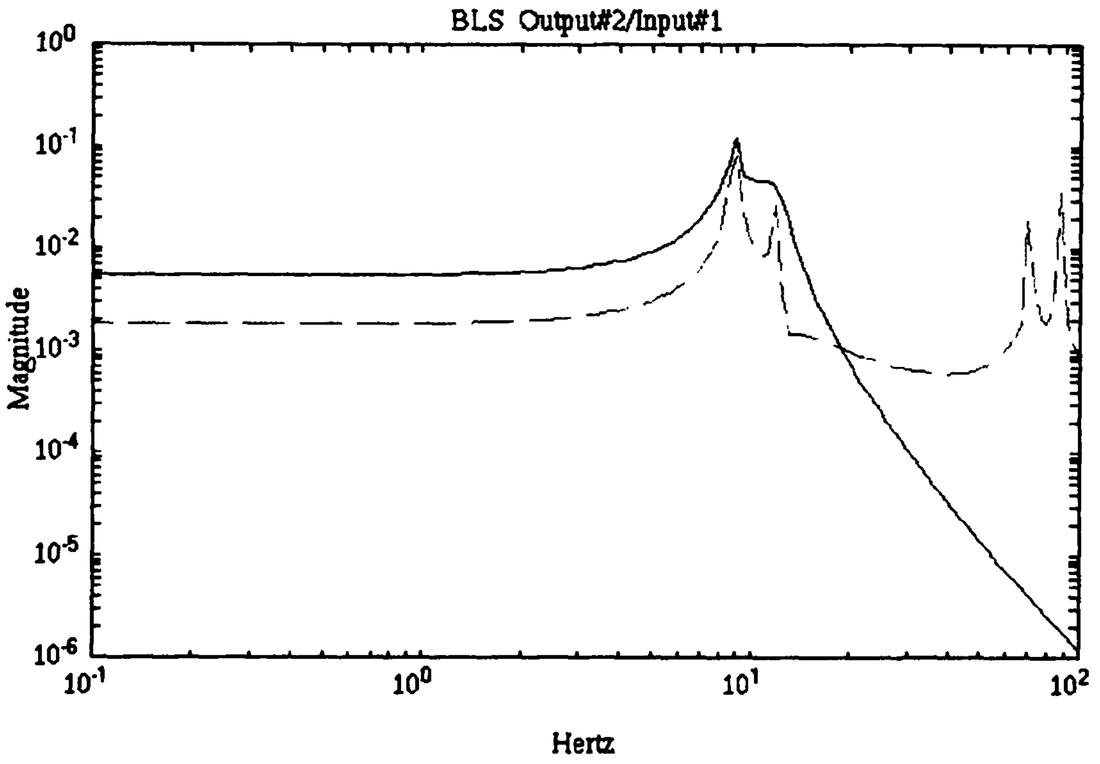

ERA/PEM Input #2/Output #2
-20
-30
-40
-50
-60
-70
-80
Magnitude (dB)
0
2
4
6
8
10
12
Frequency (Hz)

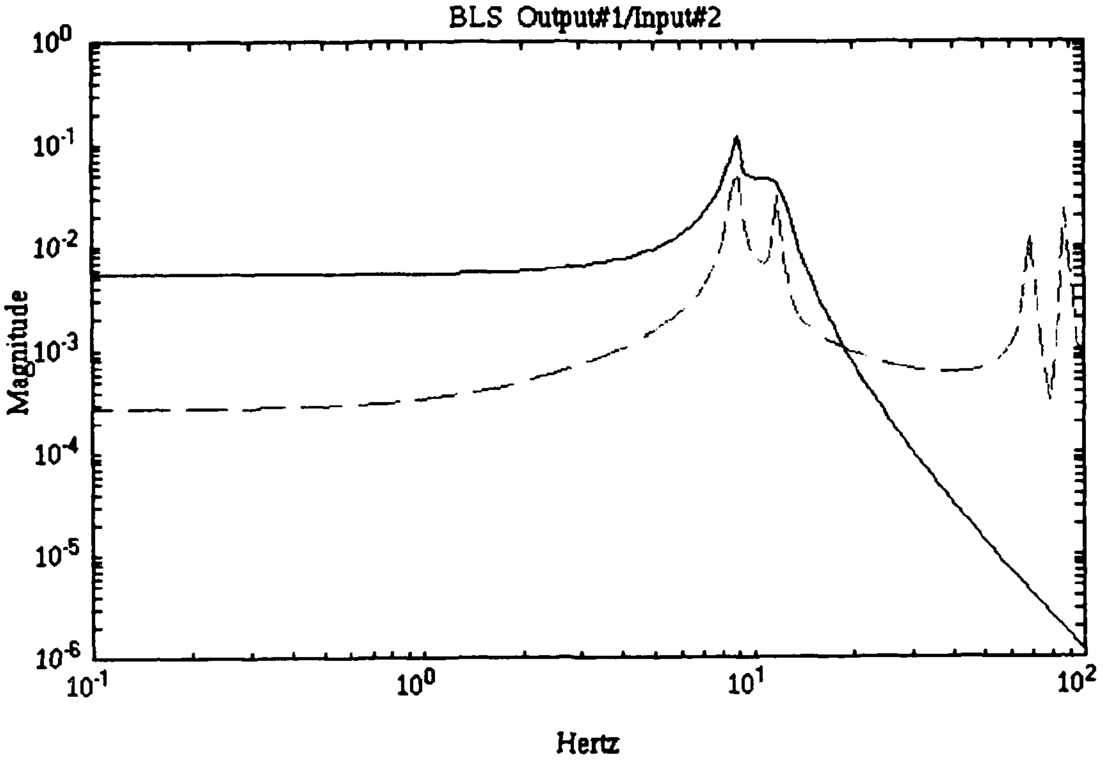

ERA/PEM Input #2/Output #1
0
-20
-40
-60
-80
-100
-120
Magnitude (dB)
0
2
4
6
8
10
12
Frequency (Hz)

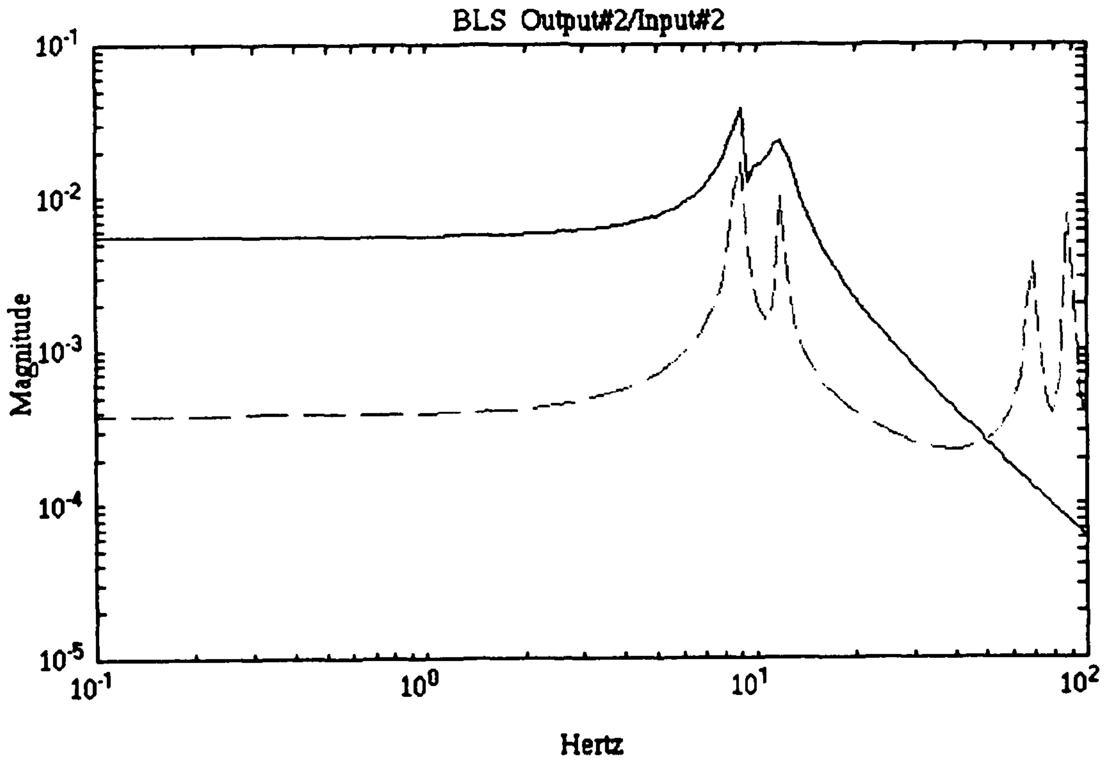

BLS Output #1
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
2
4
6
8
10
12
14
16
18
20
seconds


BLS Output #2
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
0
2
4
6
8
10
12
14
16
18
20
seconds


BLS Output#1/Input#1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
Magnitude
10^{-1}
10^0
10^1
10^2
Hertz


BLS Output#2/Input#1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
Magnitude
10^{-1}
10^0
10^1
10^2
Hertz


BLS Output#1/Input#2
Magnitude
Hertz
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-1}
10^{0}
10^{1}
10^{2}


BLS Output#2/Input#2
Magnitude
Hertz
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-1}
10^{0}
10^{1}
10^{2}

Appendix B. PACOSS State Space Model

The ERA/PEM method was able to generate a state space model for the PACOSS DTA. The model is in discrete block diagonal form, and contains real *plant*, *input*, and *output* matrices. The *plant* matrix is expressed in real, block diagonal form.

$$\mathbf{A} = \begin{bmatrix} 0.9731 & 0.1841 \\ -0.1841 & 0.9731 \end{bmatrix}, \begin{bmatrix} 0.9307 & 0.3455 \\ -0.3455 & 0.9307 \end{bmatrix}, \begin{bmatrix} 0.8808 & 0.4601 \\ -0.4601 & 0.8808 \end{bmatrix}, \begin{bmatrix} 0.8285 & 0.5419 \\ -0.5419 & 0.8285 \end{bmatrix},$$

$$\begin{bmatrix} 0.8167 & 0.5665 \\ -0.5665 & 0.8167 \end{bmatrix}, \begin{bmatrix} 0.6269 & 0.7207 \\ -0.7207 & 0.6269 \end{bmatrix}, \begin{bmatrix} 0.6256 & 0.7752 \\ -0.7752 & 0.6256 \end{bmatrix}, \begin{bmatrix} 0.4289 & 0.8982 \\ -0.8982 & 0.4289 \end{bmatrix},$$

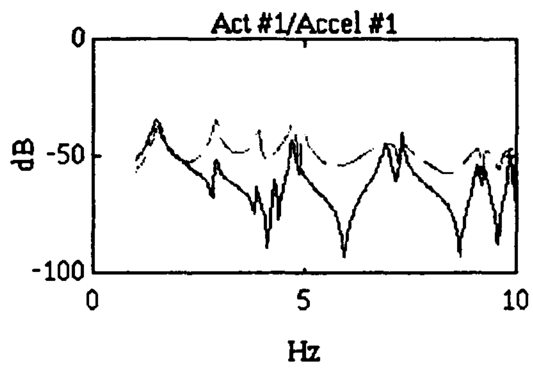
$$\begin{bmatrix} 0.4019 & 0.8666 \\ -0.8666 & 0.4019 \end{bmatrix}, \begin{bmatrix} 0.3697 & 0.9059 \\ -0.9059 & 0.3697 \end{bmatrix}$$

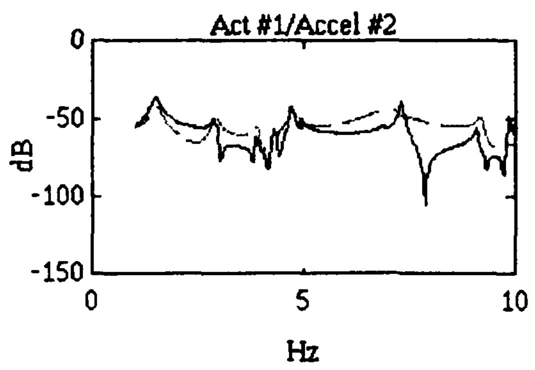
$$\mathbf{B} = \begin{bmatrix}
-0.1839 & -0.0858 & -0.0148 & 0.0166 & -0.0666 & -0.1587 & 0.0433 & 0.0256 \\
0.1422 & 0.0448 & -0.0389 & -0.1230 & -0.0963 & -0.1788 & 0.0253 & 0.0049 \\
-0.1659 & 0.2568 & 0.2017 & -0.1319 & 0.3584 & 0.4355 & -0.0207 & 0.2876 \\
0.1066 & -0.0856 & -0.1994 & 0.1220 & -0.4463 & -0.2985 & 0.0626 & -0.3602 \\
-0.0727 & -0.1196 & 0.1221 & 0.0924 & -0.1249 & 0.0462 & 0.4333 & -0.0051 \\
-0.0220 & -0.0166 & 0.0142 & -0.0172 & 0.0177 & 0.0620 & -0.1860 & 0.0365 \\
-0.0653 & -0.0619 & 0.0779 & 0.0954 & -0.0802 & -0.0389 & 0.0400 & -0.1021 \\
0.2431 & 0.2337 & -0.1498 & -0.1560 & -0.0935 & -0.1230 & 0.0213 & -0.1312 \\
-0.0396 & -0.1732 & -0.2710 & -0.1368 & 0.3347 & 0.2970 & -0.0419 & 0.4921 \\
-0.0290 & -0.0541 & -0.0089 & -0.0254 & 0.0581 & 0.0999 & 0.0021 & 0.0597 \\
-0.0827 & -0.0363 & -0.0561 & 0.1177 & 0.0739 & 0.0057 & -0.3922 & -0.0341 \\
0.1434 & -0.0283 & -0.0146 & -0.01478 & -0.1046 & -0.1193 & -0.1772 & 0.0006 \\
0.1438 & -0.2087 & -0.1740 & 0.1546 & -0.0977 & -0.1131 & -0.0175 & -0.0832 \\
-0.1213 & 0.1627 & 0.1766 & -0.1335 & 0.0942 & 0.0819 & 0.0124 & 0.0772 \\
0.0540 & 0.0120 & -0.0760 & 0.1462 & -0.0790 & -0.1703 & -0.1739 & -0.1717 \\
-0.0211 & -0.0118 & 0.0596 & -0.1210 & 0.0617 & 0.1249 & 0.2033 & 0.1472 \\
0.0741 & -0.0963 & 0.0577 & -0.0767 & -0.0329 & 0.0619 & 0.2585 & -0.0064 \\
0.0427 & -0.0596 & 0.0597 & -0.0784 & -0.0343 & 0.0859 & 0.2697 & 0.0146 \\
-0.0174 & 0.0079 & 0.0008 & -0.0244 & 0.0087 & 0.0452 & -0.0278 & 0.0556 \\
0.0202 & -0.0266 & 0.0041 & -0.0151 & 0.0123 & -0.0039 & 0.0164 & 0.0173
\end{bmatrix}$$

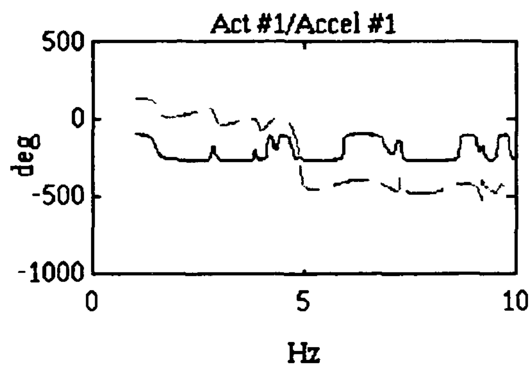
$$C^T = 0.01 * \begin{bmatrix} -0.0729 & -0.0351 & 0.3088 & 0.1679 & 0.1361 & 0.0579 & 0.6896 & 2.1689 \\ -0.0691 & -0.0585 & -0.2660 & -0.0256 & -0.1183 & -0.0431 & 0.4115 & -0.7177 \\ -0.1235 & 0.0305 & 0.0610 & -0.1052 & 0.0442 & 0.0643 & 0.0113 & 0.0941 \\ -0.0096 & 0.0026 & -0.0119 & 0.0144 & -0.0103 & 0.0008 & 0.3421 & 0.1181 \\ -0.01454 & -0.0289 & 0.0715 & 0.0839 & -0.0743 & 0.1312 & 0.0037 & -1.1872 \\ 0.0391 & 0.0134 & -0.0212 & -0.0662 & 0.0599 & 0.0081 & -0.0215 & 0.5127 \\ 0.0607 & 0.0369 & -0.0472 & -0.0275 & -0.2812 & -0.2208 & 0.4714 & 0.1425 \\ 0.1203 & 0.0372 & -0.1486 & -0.1580 & -0.0031 & -0.0186 & -0.4291 & 0.0504 \\ -0.0611 & -0.0241 & -0.0363 & -0.0613 & 0.0364 & 0.0363 & -0.3400 & -0.0427 \\ -0.3156 & -0.0214 & -0.0402 & -0.0914 & 0.0333 & 0.0257 & -0.0805 & -0.0072 \\ -0.0195 & -0.3803 & -0.7671 & 0.0800 & 0.0603 & -0.1987 & -0.1103 & 0.4800 \\ 0.3328 & -0.0176 & 0.0539 & -0.2555 & -0.4385 & -0.2449 & 0.0299 & -0.1354 \\ 0.0242 & -0.0050 & -0.0138 & 0.0193 & -0.0319 & -0.0217 & -0.1942 & -0.0144 \\ 0.0114 & -0.0022 & -0.0082 & 0.0112 & -0.0144 & -0.0083 & -0.0817 & -0.0103 \\ -0.0880 & -0.0679 & 0.0441 & -0.0235 & 0.0497 & 0.0134 & 0.0152 & -0.0162 \\ 0.0110 & 0.0546 & -0.0137 & 0.0090 & -0.0248 & -0.0106 & -0.0142 & 0.0718 \\ 0.6202 & -0.1401 & 0.4226 & -0.3691 & -1.0326 & 0.3141 & 0.1365 & -0.2453 \\ 0.2901 & -0.0494 & 0.4162 & -0.3517 & -0.6825 & 0.2415 & 0.0803 & -1.9738 \\ -0.4932 & 0.2255 & -4.4244 & 0.5451 & -1.4755 & 0.0061 & -0.3389 & 0.3047 \\ -0.6307 & 0.0718 & -0.5193 & -0.4821 & 0.5788 & 0.3518 & -0.5922 & 0.0118 \end{bmatrix}$$

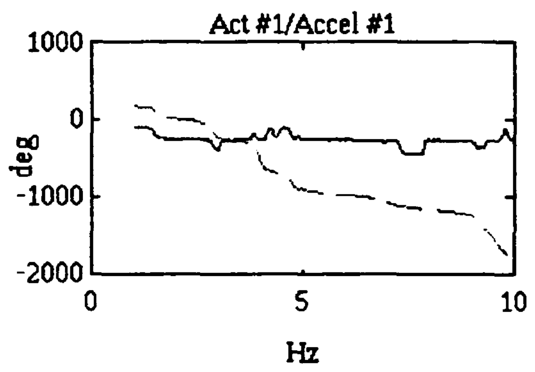
Appendix C. Bode Plots

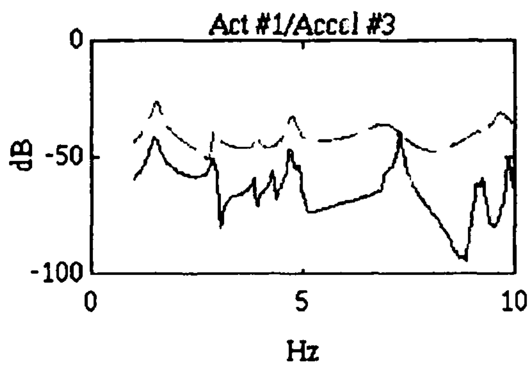
This appendix contains the bode plot comparissons of the actual PACOSS system and the ERA/PEM developed model. All plots are from 1 to 10 Hz, and are frequency response functions of velocity output to force input. In all plots, the dashed line represents the ERA/PEM model derived transfer function, while the solid line represents the actual PACOSS DTA derived frequency response function.

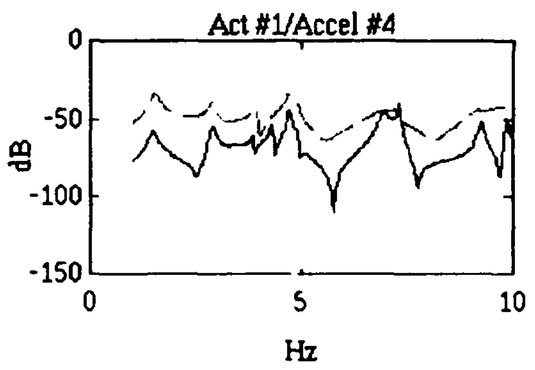

Act #1/Accel #1
0
-50
-100
dB
0
5
10
Hz

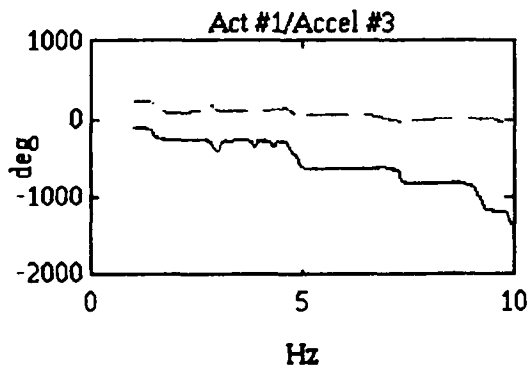

Act #1/Accel #2
0
-50
-100
-150
dB
0
5
10
Hz

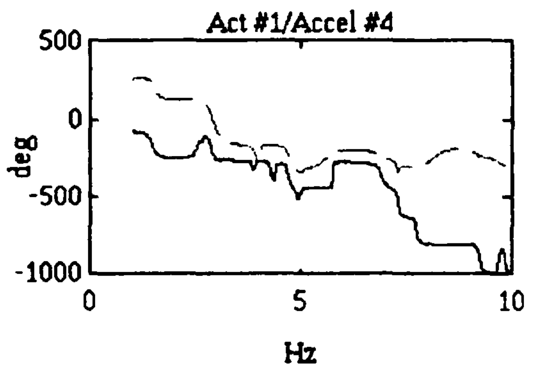

Act #1/Accel #1
500
0
-500
-1000
deg
0
5
10
Hz

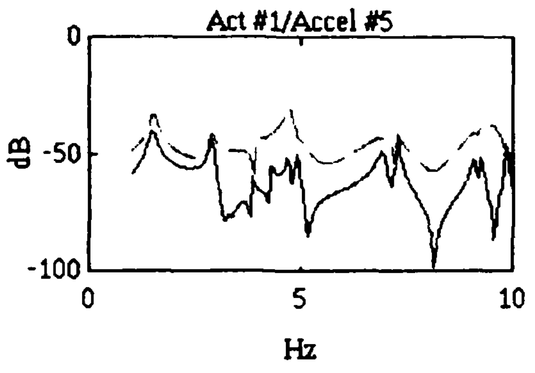

Act #1/Accel #1
1000
0
-1000
-2000
deg
0
5
10
Hz

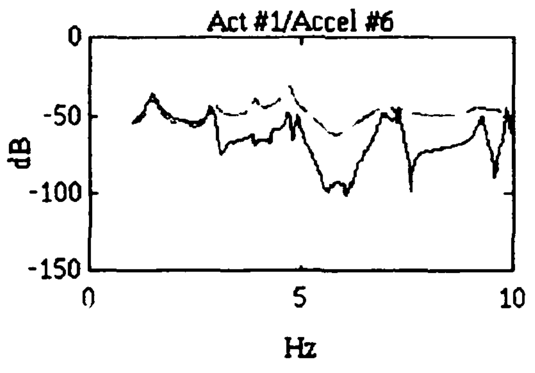

Act #1/Accel #3
0
-50
-100
dB
0
5
10
Hz

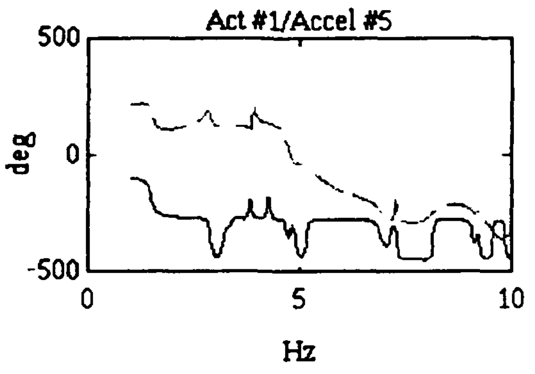

Act #1/Accel #4
0
-50
-100
-150
dB
0
5
10
Hz

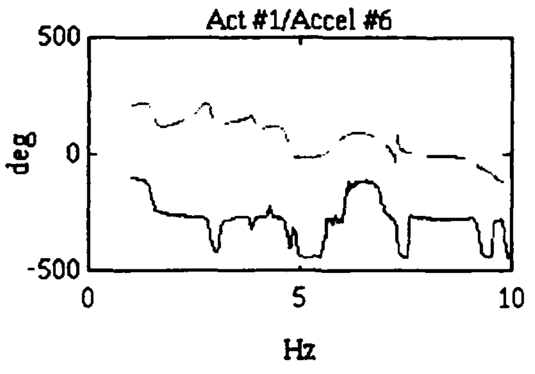

Act #1/Accel #3
1000
0
-1000
-2000
deg
0
5
10
Hz

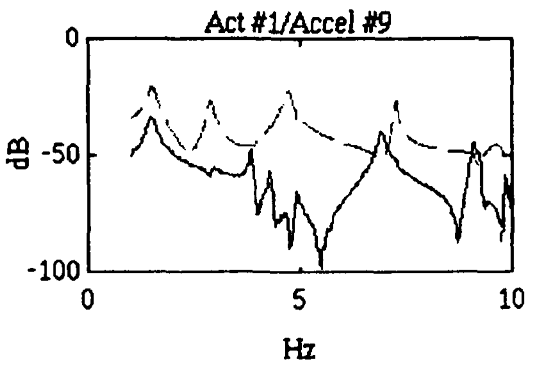

Act #1/Accel #4
500
0
-500
-1000
deg
0
5
10
Hz

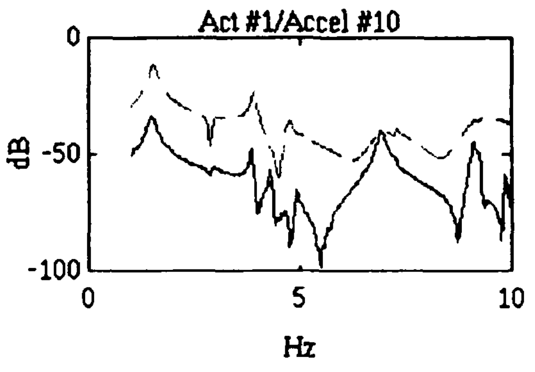

Act #1/Accel #5
dB
0
-50
-100
0
5
10
Hz

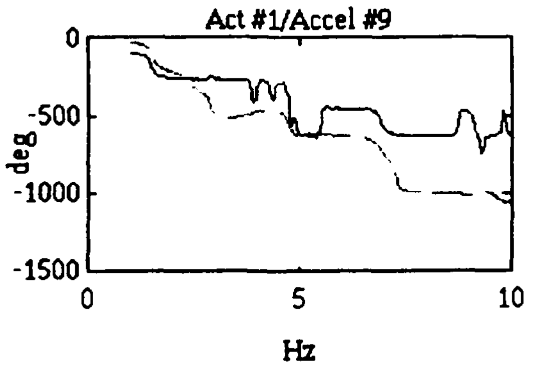

Act #1/Accel #6
dB
0
-50
-100
-150
0
5
10
Hz

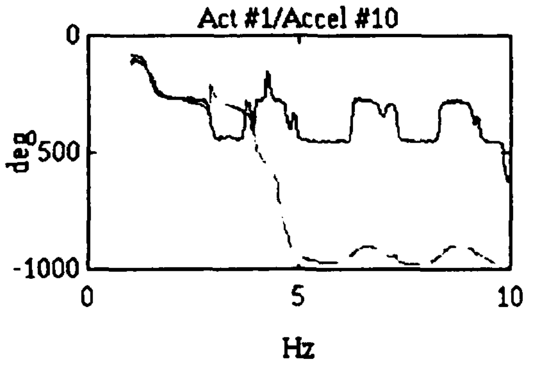

Act #1/Accel #5
deg
500
0
-500
0
5
10
Hz

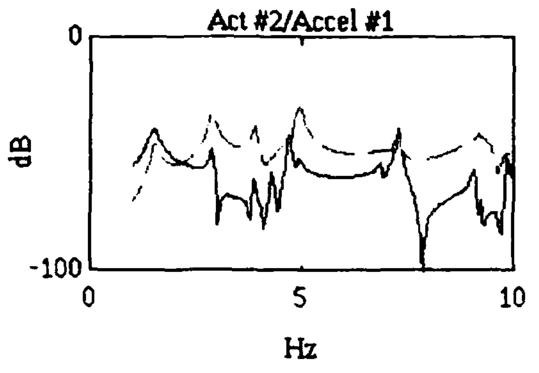

Act #1/Accel #6
deg
500
0
-500
0
5
10
Hz

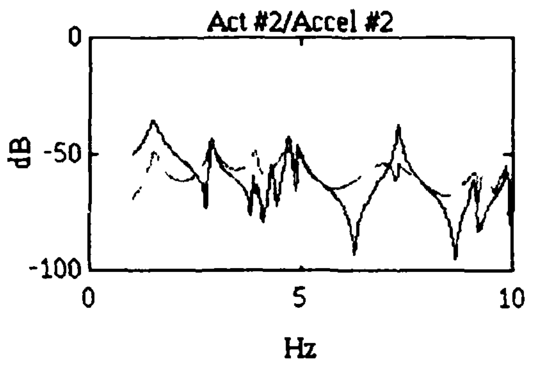

Act #1/Accel #9
dB
0
-50
-100
0
5
10
Hz

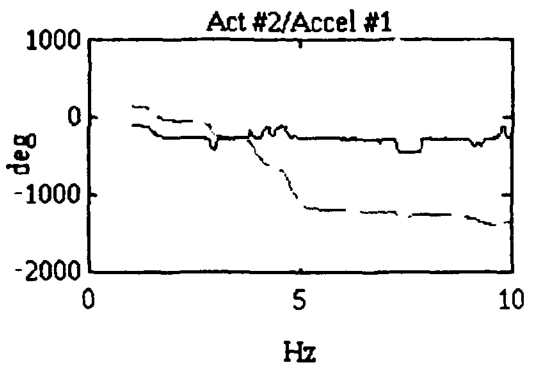

Act #1/Accel #10
dB
0
-50
-100
0
5
10
Hz

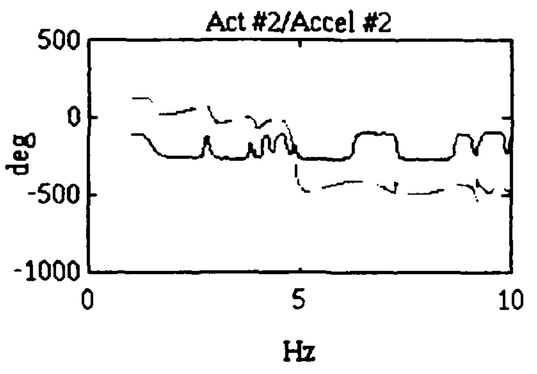

Act #1/Accel #9
deg
0
-500
-1000
-1500
0
5
10
Hz

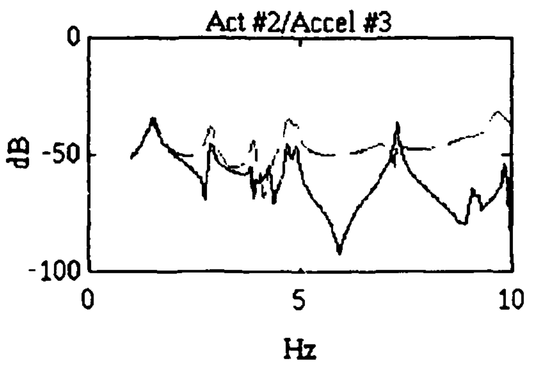

Act #1/Accel #10
deg
0
-500
-1000
0
5
10
Hz

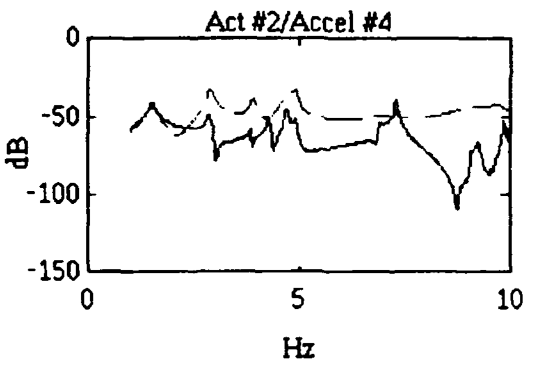

Act #2/Accel #1
dB
0
-100
0
5
10
Hz

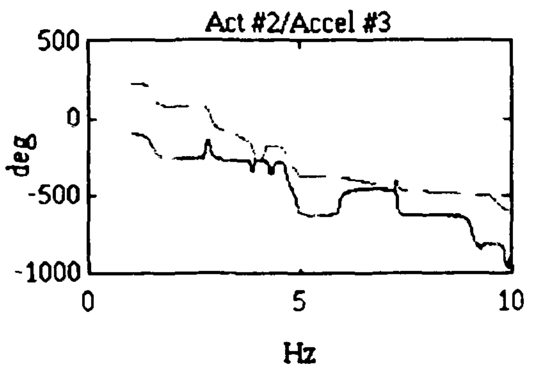

Act #2/Accel #2
dB
0
-50
-100
0
5
10
Hz

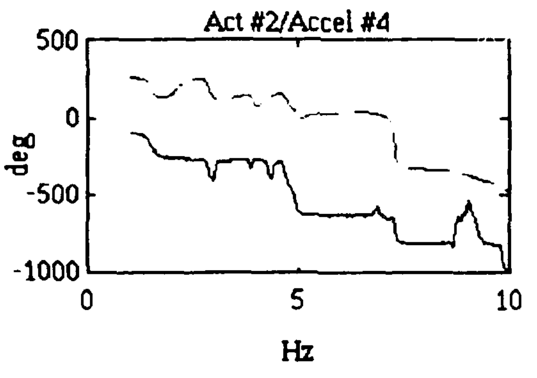

Act #2/Accel #1
deg
1000
0
-1000
-2000
0
5
10
Hz

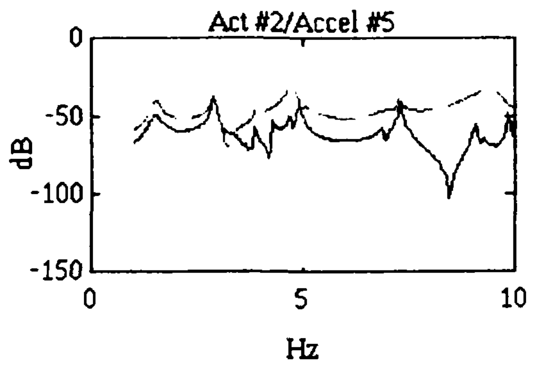
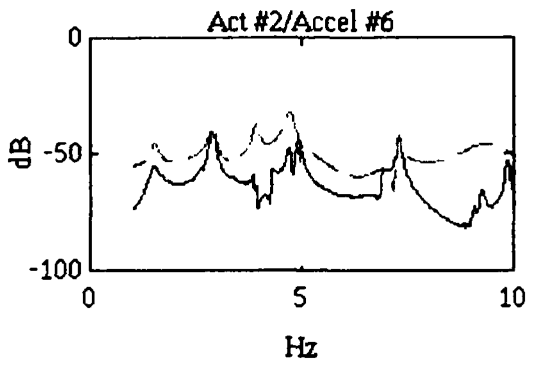
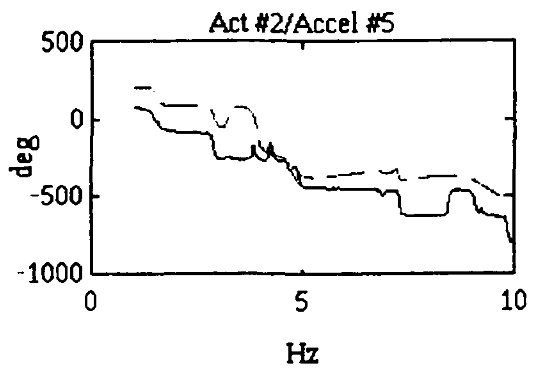
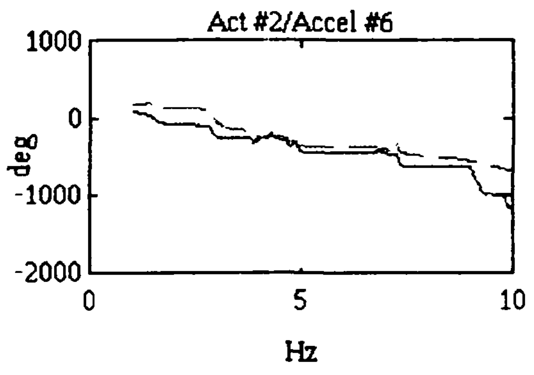
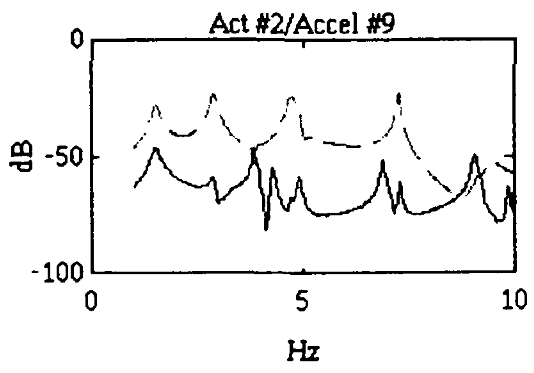
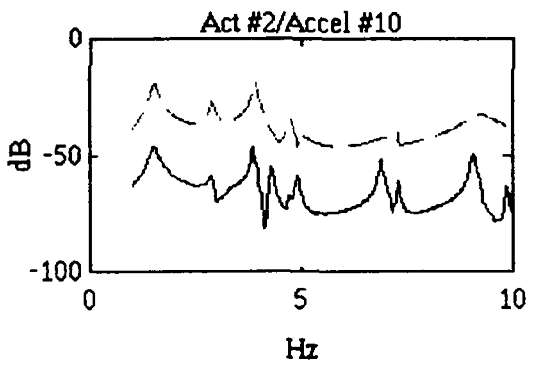
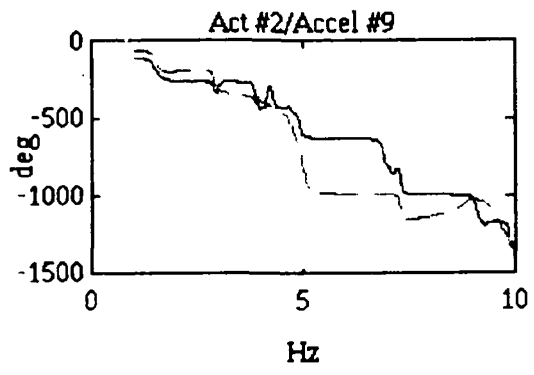
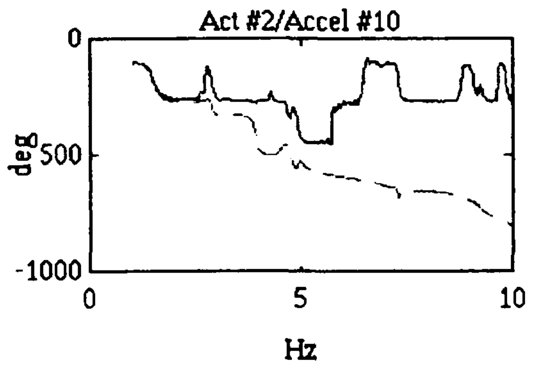

Act #2/Accel #2
deg
500
0
-500
-1000
0
5
10
Hz

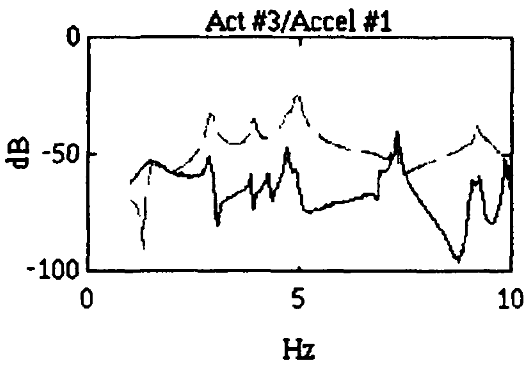

Act #2/Accel #3
dB
0
-50
-100
0
5
10
Hz

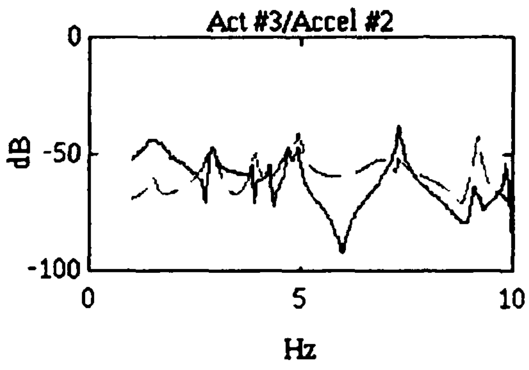

Act #2/Accel #4
dB
0
-50
-100
-150
0
5
10
Hz

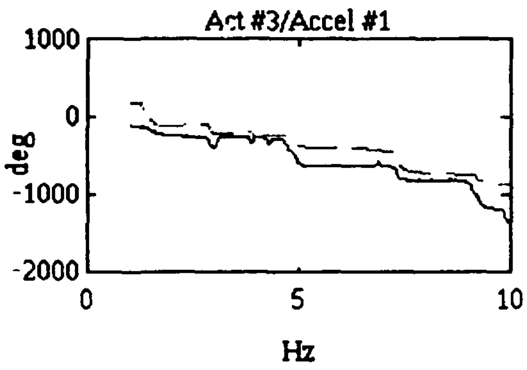

Act #2/Accel #3
deg
500
0
-500
-1000
0
5
10
Hz

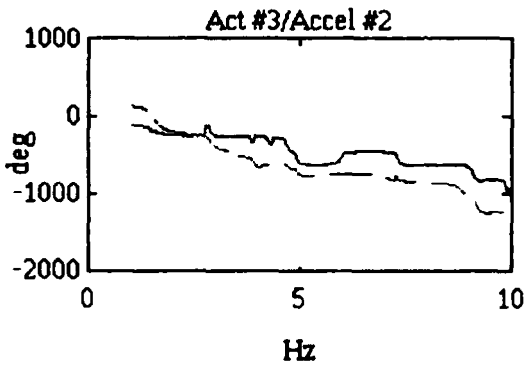

Act #2/Accel #4
deg
500
0
-500
-1000
0
5
10
Hz

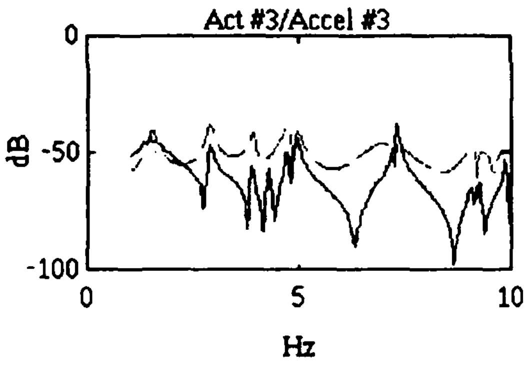

Act #2/Accel #5
0
-50
-100
-150
dB
0
5
10
Hz

Act #2/Accel #6
0
-50
-100
dB
0
5
10
Hz

Act #2/Accel #5
500
0
-500
-1000
deg
0
5
10
Hz

Act #2/Accel #6
1000
0
-1000
-2000
deg
0
5
10
Hz

Act #2/Accel #9
0
-50
-100
dB
0
5
10
Hz

Act #2/Accel #10
0
-50
-100
dB
0
5
10
Hz

Act #2/Accel #9
0
-500
-1000
-1500
deg
0
5
10
Hz

Act #2/Accel #10
0
500
-1000
deg
0
5
10
Hz

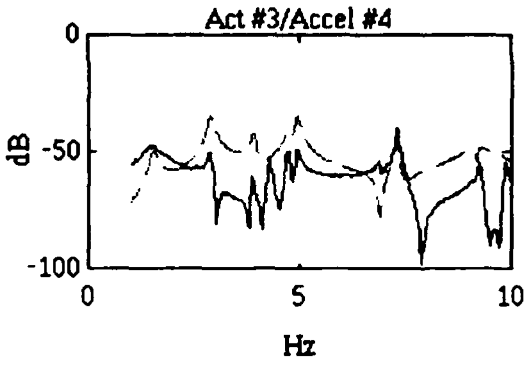

Act #3/Accel #1
dB
0
-50
-100
0
5
10
Hz

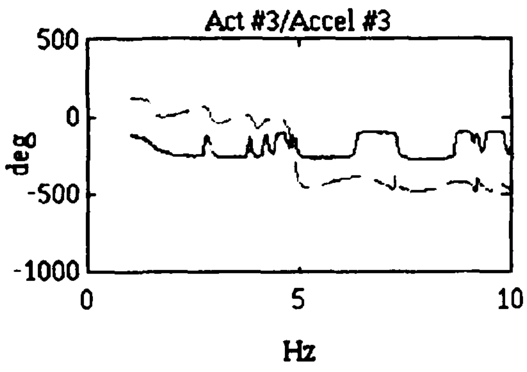

Act #3/Accel #2
dB
0
-50
-100
0
5
10
Hz

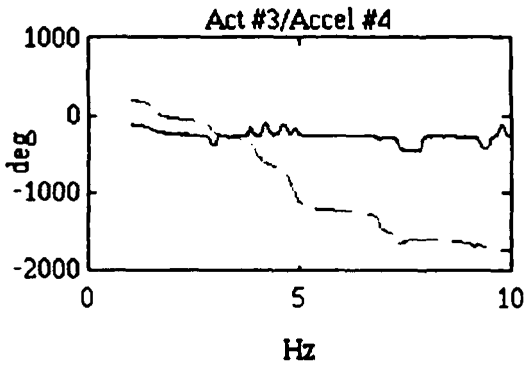

Act #3/Accel #1
deg
1000
0
-1000
-2000
0
5
10
Hz

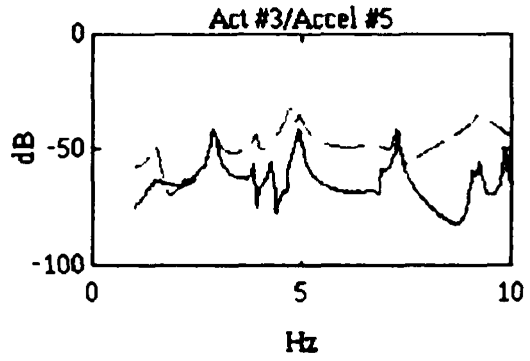

Act #3/Accel #2
deg
1000
0
-1000
-2000
0
5
10
Hz

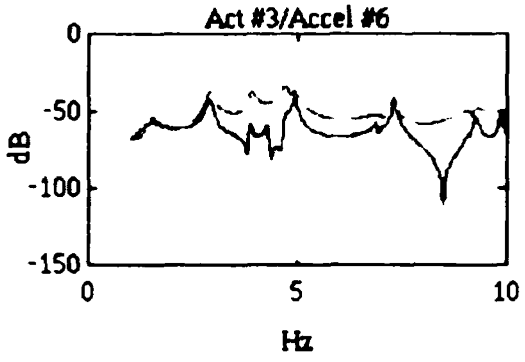

Act #3/Accel #3
dB
0
-50
-100
0
5
10
Hz

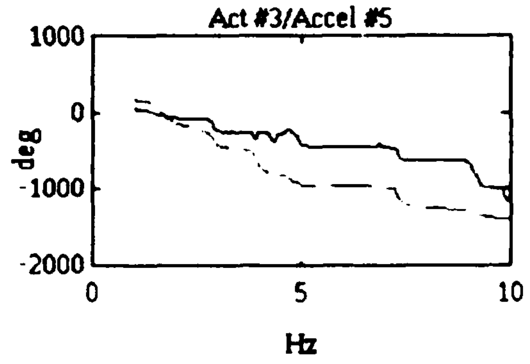

Act #3/Accel #4
dB
0
-50
-100
0
5
10
Hz

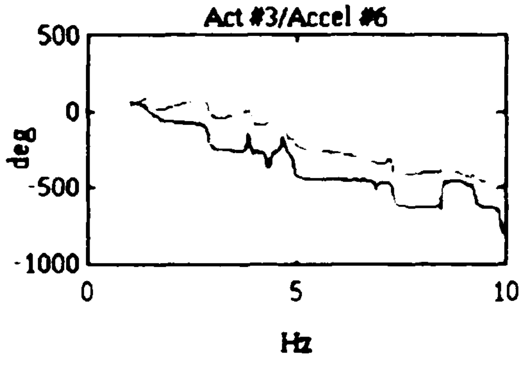

Act #3/Accel #3
deg
500
0
-500
-1000
0
5
10
Hz

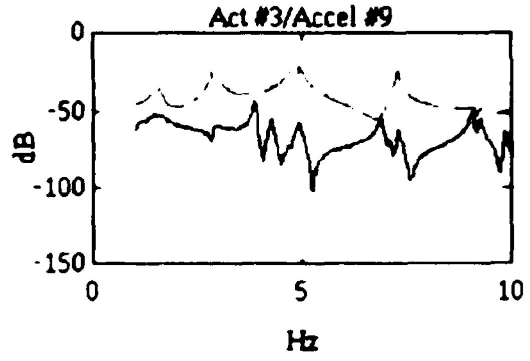

Act #3/Accel #4
deg
1000
0
-1000
-2000
0
5
10
Hz

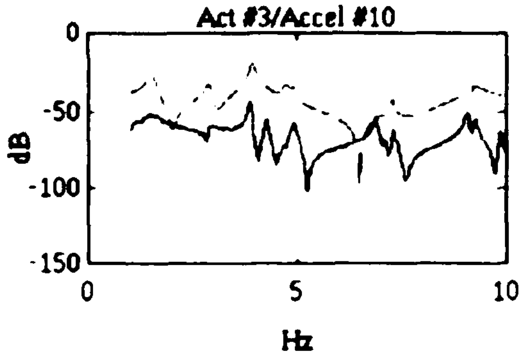

Act #3/Accel #5
0
-50
-100
dB
0
5
10
Hz

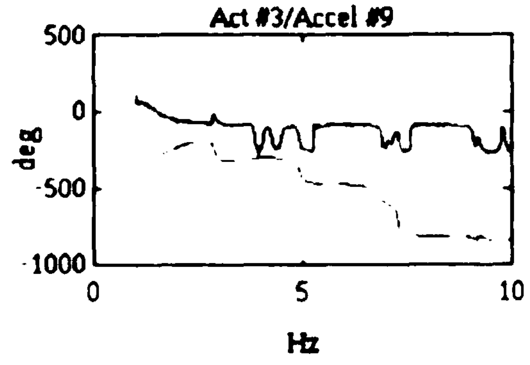

Act #3/Accel #6
0
-50
-100
-150
dB
0
5
10
Hz

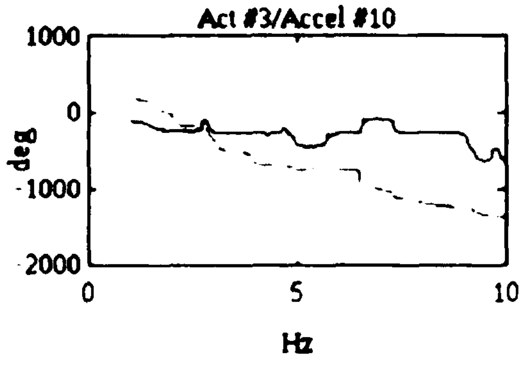

Act #3/Accel #5
1000
0
-1000
-2000
deg
0
5
10
Hz

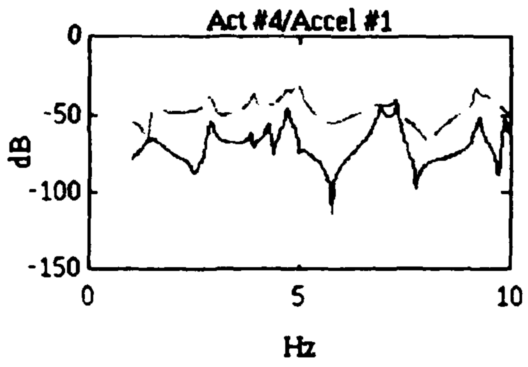

Act #3/Accel #6
500
0
-500
-1000
deg
0
5
10
Hz

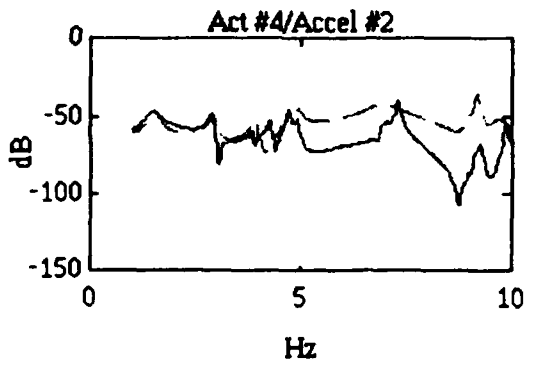

Act #3/Accel #9
0
-50
-100
-150
dB
0
5
10
Hz

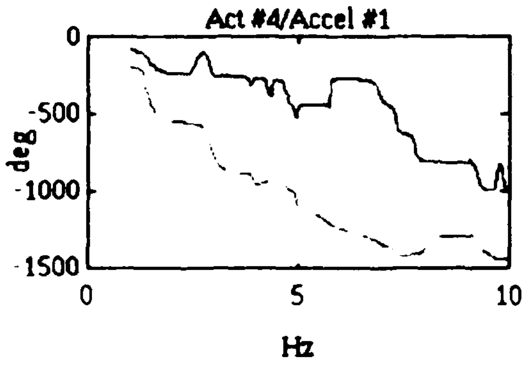

Act #3/Accel #10
0
-50
-100
-150
dB
0
5
10
Hz

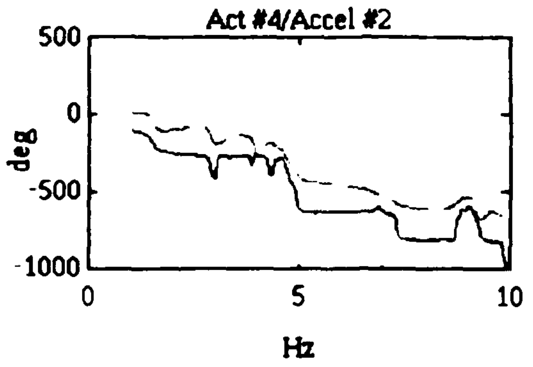

Act #3/Accel #9
500
0
-500
-1000
deg
0
5
10
Hz

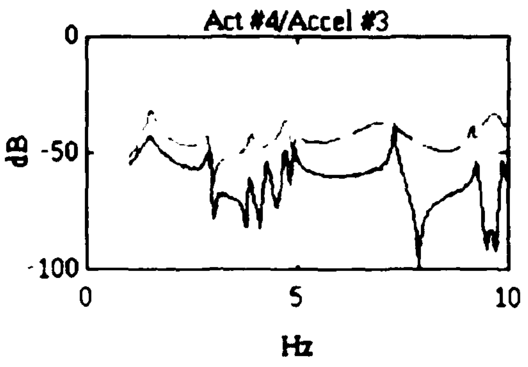

Act #3/Accel #10
1000
0
-1000
-2000
deg
0
5
10
Hz

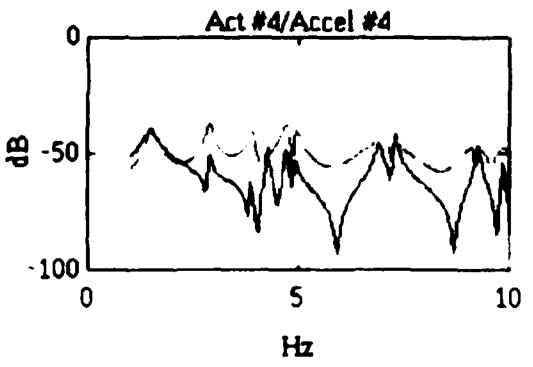

Act #4/Accel #1
0
-50
-100
-150
dB
0
5
10
Hz

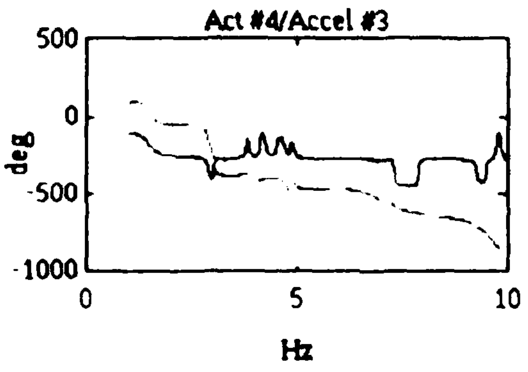

Act #4/Accel #2
0
-50
-100
-150
dB
0
5
10
Hz

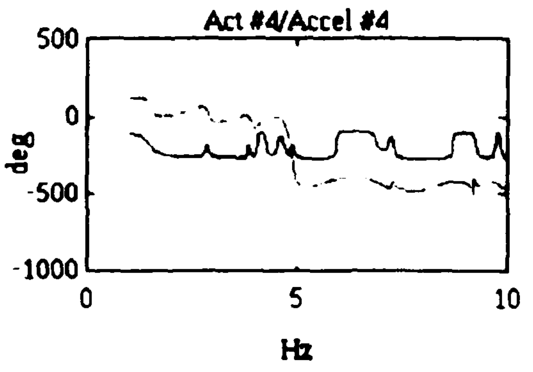

Act #4/Accel #1
0
-500
-1000
-1500
deg
0
5
10
Hz

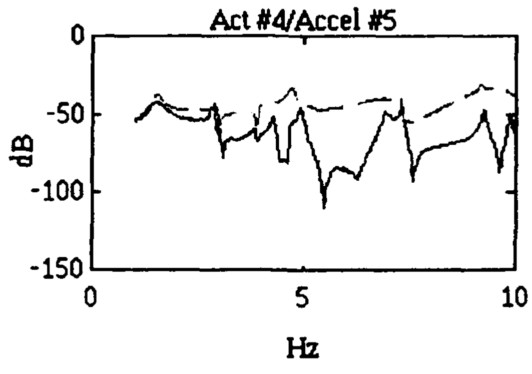

Act #4/Accel #2
500
0
-500
-1000
deg
0
5
10
Hz

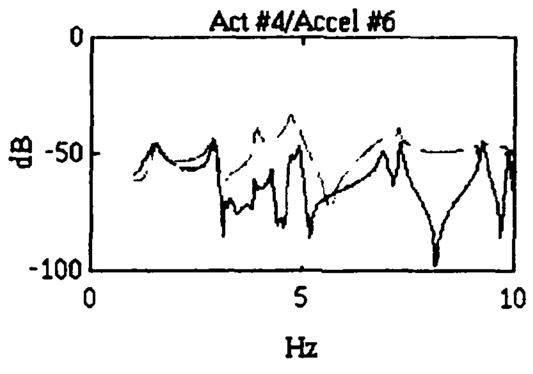

Act #4/Accel #3
0
-50
-100
dB
0
5
10
Hz

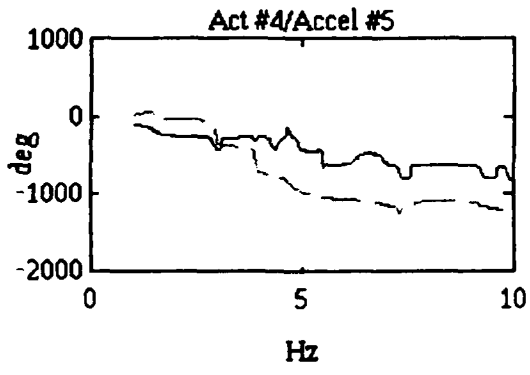

Act #4/Accel #4
0
-50
-100
dB
0
5
10
Hz

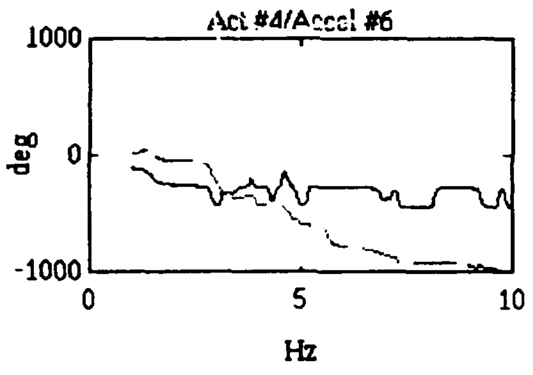

Act #4/Accel #3
500
0
-500
-1000
deg
0
5
10
Hz

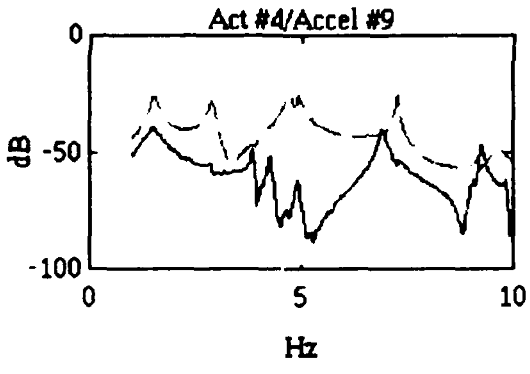

Act #4/Accel #4
500
0
-500
-1000
deg
0
5
10
Hz

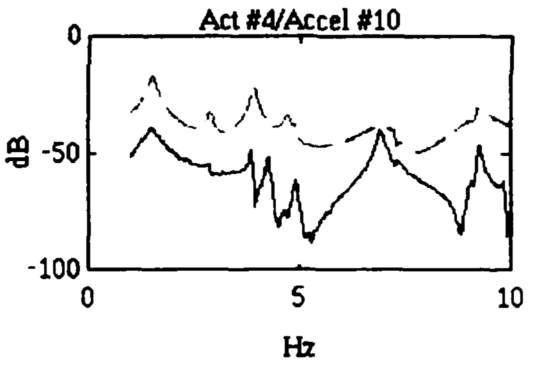

Act #4/Accel #5
0
-50
-100
-150
dB
0
5
10
Hz

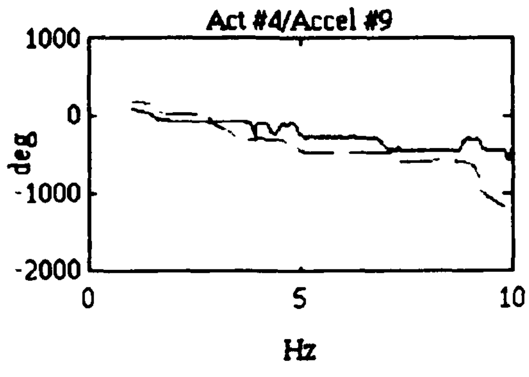

Act #4/Accel #6
0
-50
-100
dB
0
5
10
Hz

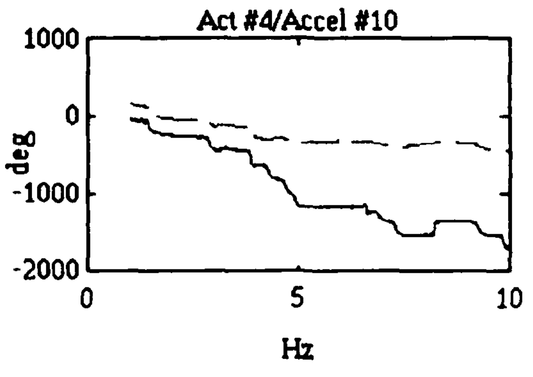

Act #4/Accel #5
1000
0
-1000
-2000
deg
0
5
10
Hz

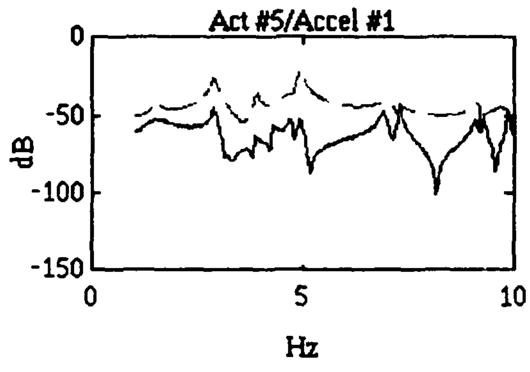

Act #4/Accel #6
1000
0
-1000
deg
0
5
10
Hz

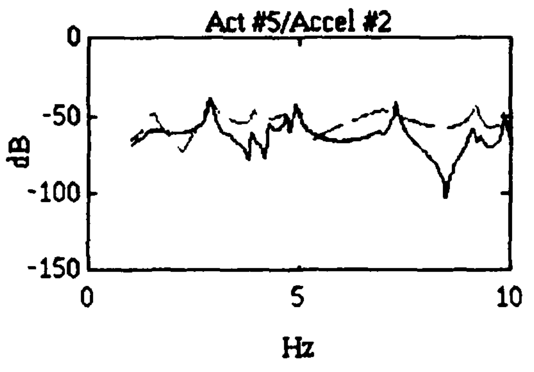

Act #4/Accel #9
0
-50
-100
dB
0
5
10
Hz

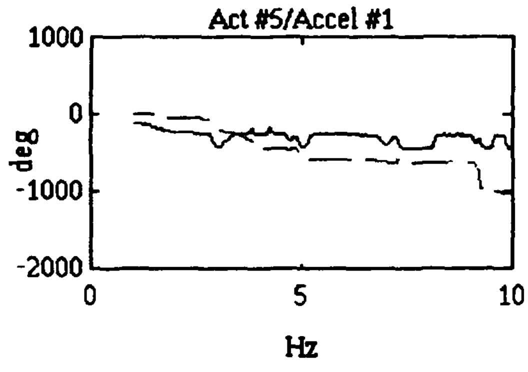

Act #4/Accel #10
0
-50
-100
dB
0
5
10
Hz

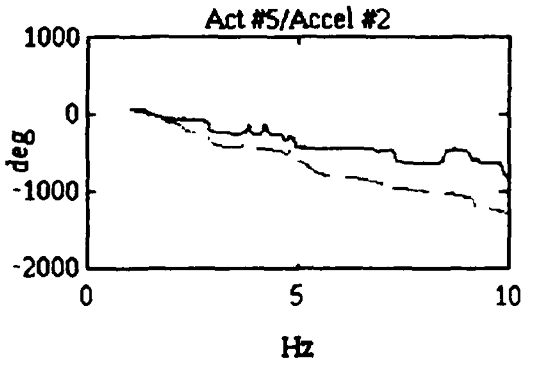

Act #4/Accel #9
1000
0
-1000
-2000
deg
0
5
10
Hz

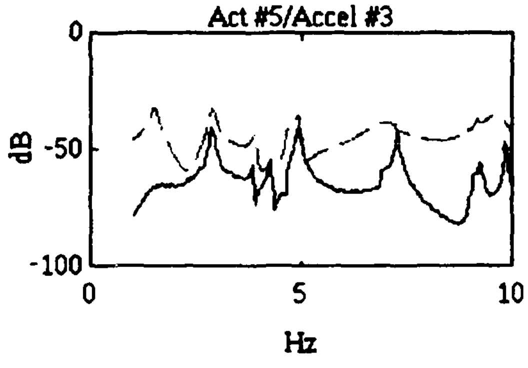

Act #4/Accel #10
1000
0
-1000
-2000
deg
0
5
10
Hz

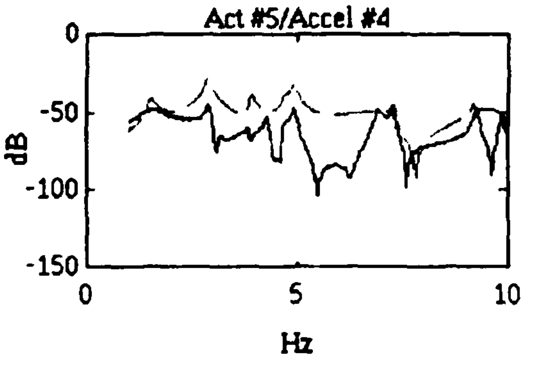

Act #5/Accel #1
0
-50
-100
-150
dB
0
5
10
Hz

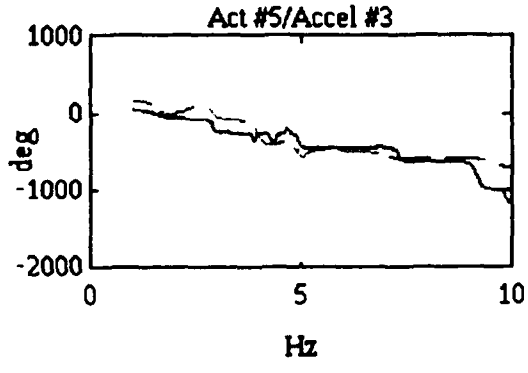

Act #5/Accel #2
0
-50
-100
-150
dB
0
5
10
Hz

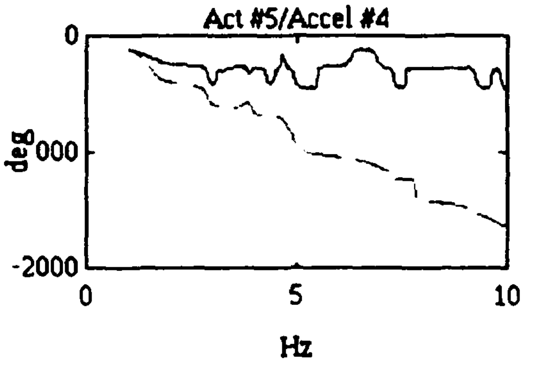

Act #5/Accel #1
1000
0
-1000
-2000
deg
0
5
10
Hz

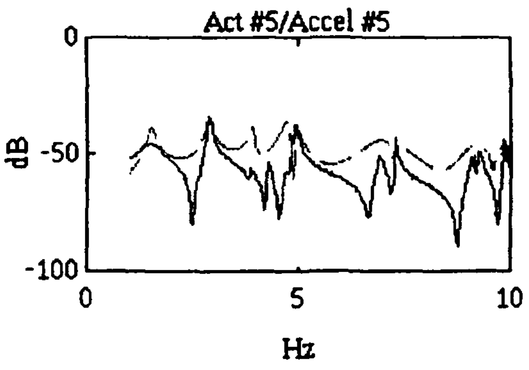

Act #5/Accel #2
1000
0
-1000
-2000
deg
0
5
10
Hz

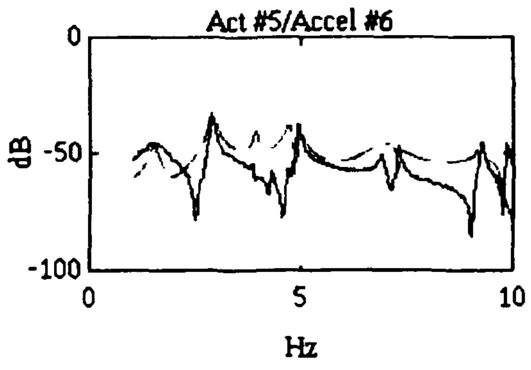

Act #5/Accel #3
0
-50
-100
dB
0
5
10
Hz

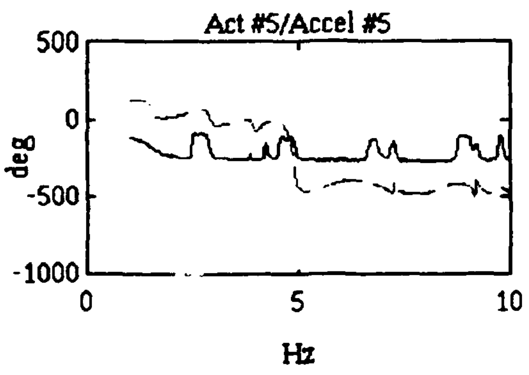

Act #5/Accel #4
0
-50
-100
-150
dB
0
5
10
Hz

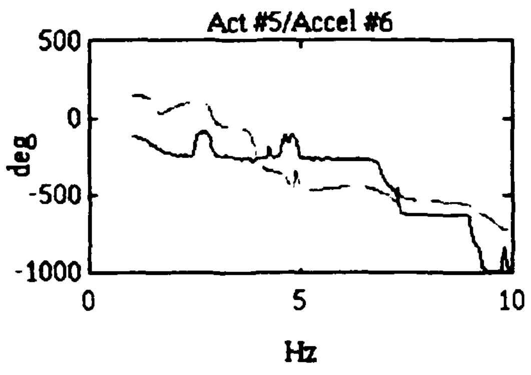

Act #5/Accel #3
1000
0
-1000
-2000
deg
0
5
10
Hz

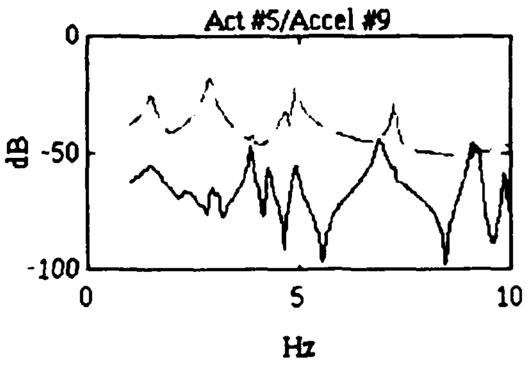

Act #5/Accel #4
0
000
-2000
deg
0
5
10
Hz

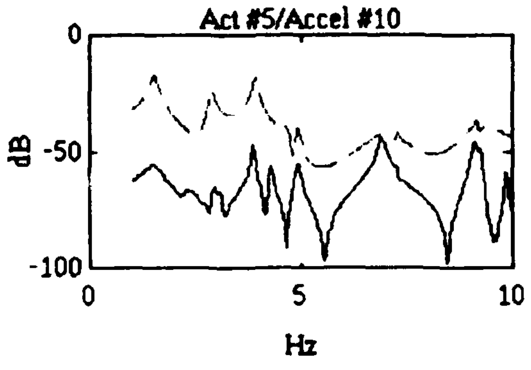

Act #5/Accel #5
dB
0
-50
-100
0
5
10
Hz

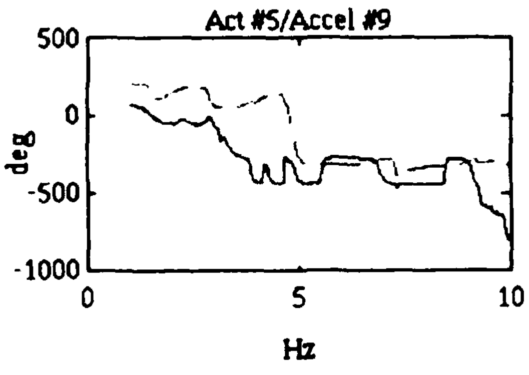

Act #5/Accel #6
dB
0
-50
-100
0
5
10
Hz

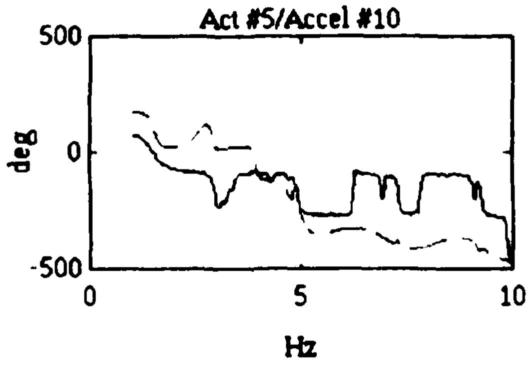

Act #5/Accel #5
deg
500
0
-500
-1000
0
5
10
Hz

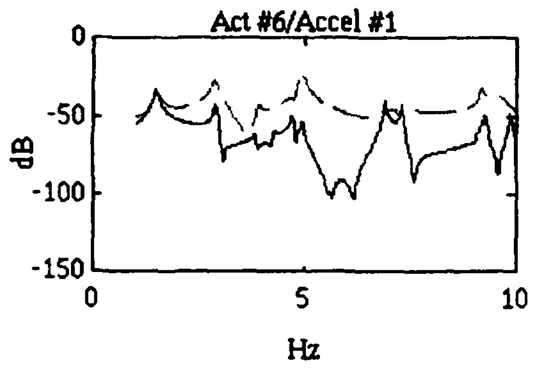

Act #5/Accel #6
deg
500
0
-500
-1000
0
5
10
Hz

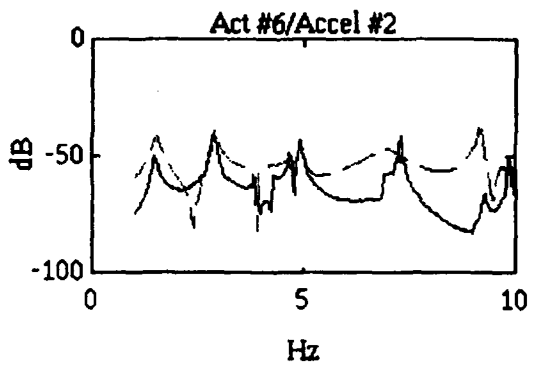

Act #5/Accel #9
dB
0
-50
-100
0
5
10
Hz

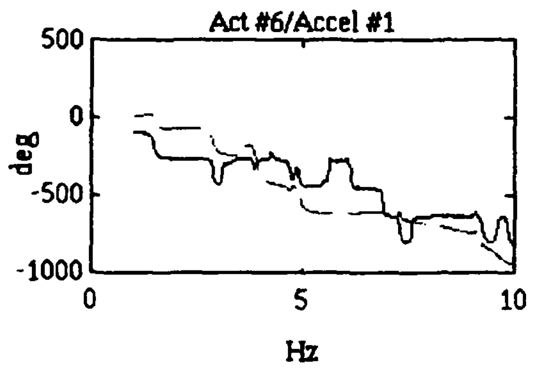

Act #5/Accel #10
dB
0
-50
-100
0
5
10
Hz

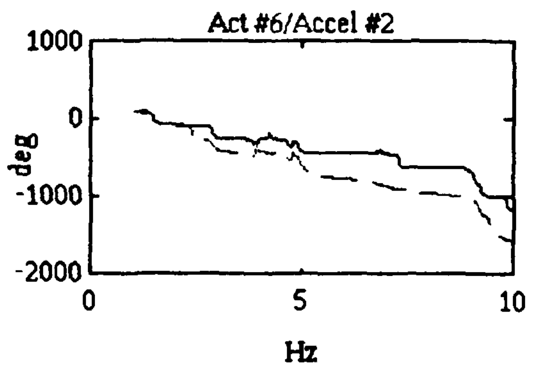

Act #5/Accel #9
deg
500
0
-500
-1000
0
5
10
Hz

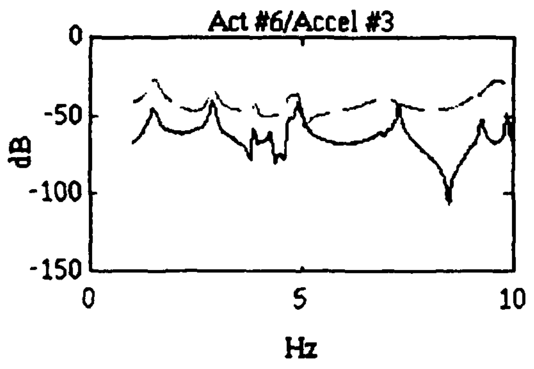

Act #5/Accel #10
deg
500
0
-500
0
5
10
Hz

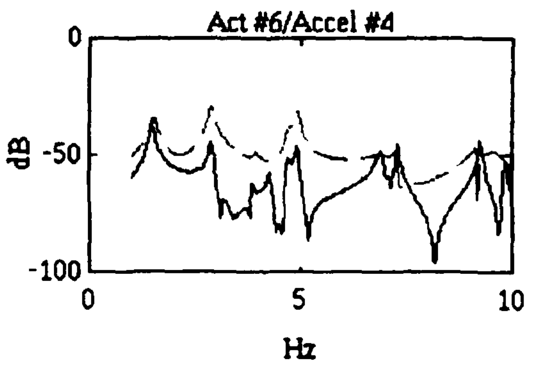

Act #6/Accel #1
0
-50
-100
-150
dB
0
5
10
Hz

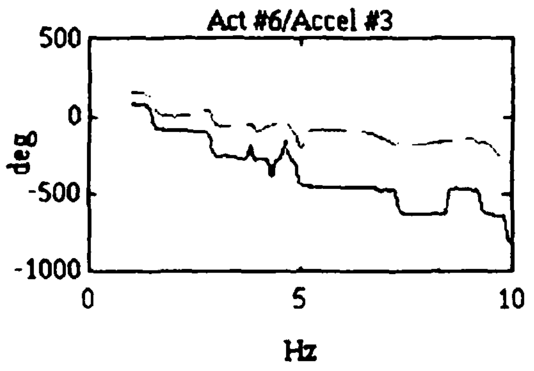

Act #6/Accel #2
0
-50
-100
dB
0
5
10
Hz

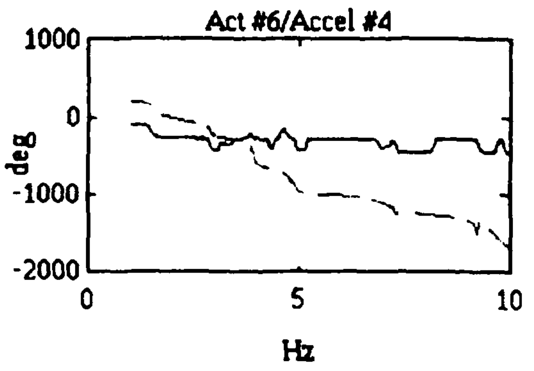

Act #6/Accel #1
500
0
-500
-1000
deg
0
5
10
Hz

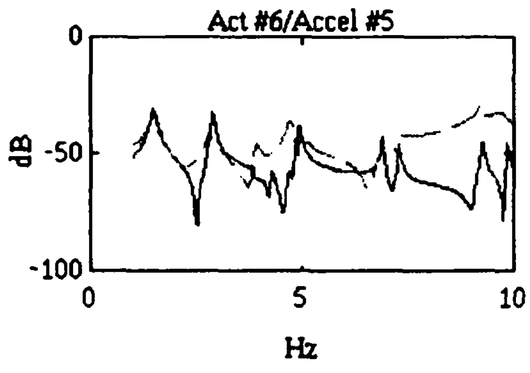

Act #6/Accel #2
1000
0
-1000
-2000
deg
0
5
10
Hz

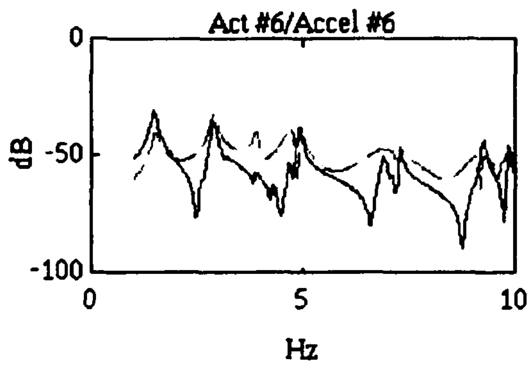

Act #6/Accel #3
0
-50
-100
-150
dB
0
5
10
Hz

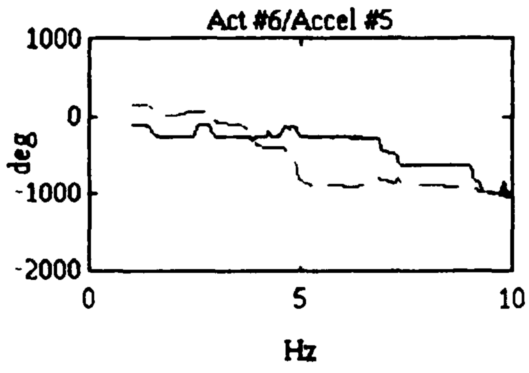

Act #6/Accel #4
0
-50
-100
dB
0
5
10
Hz

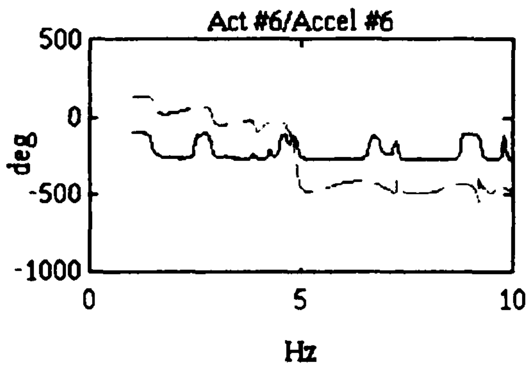

Act #6/Accel #3
500
0
-500
-1000
deg
0
5
10
Hz

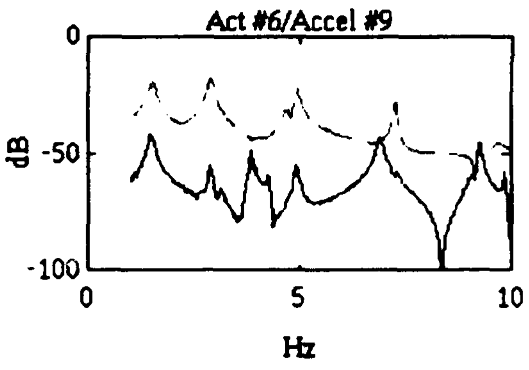

Act #6/Accel #4
1000
0
-1000
-2000
deg
0
5
10
Hz

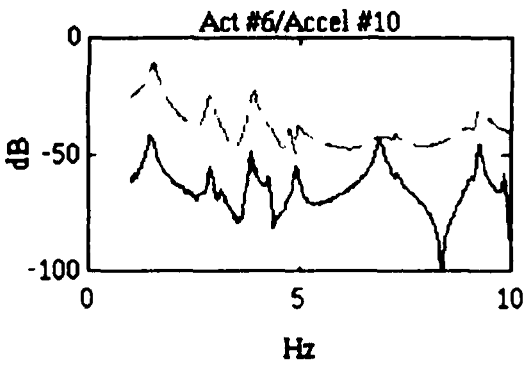

Act #6/Accel #5
dB
0
-50
-100
0
5
10
Hz

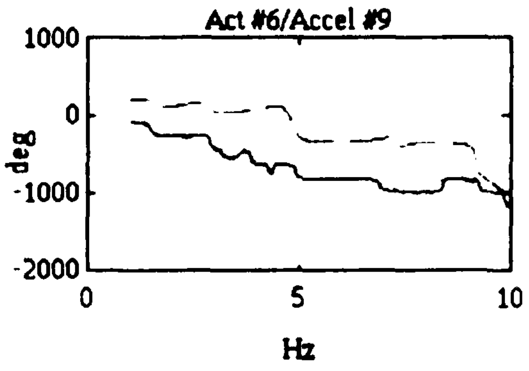

Act #6/Accel #6
dB
0
-50
-100
0
5
10
Hz

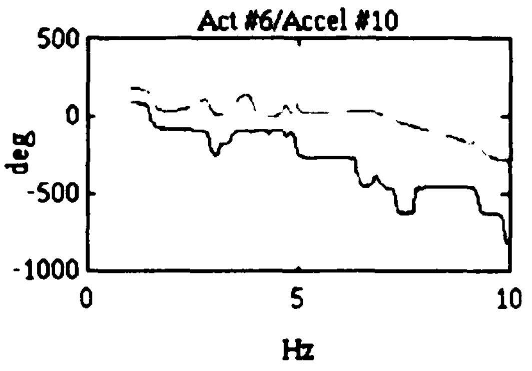

Act #6/Accel #5
deg
1000
0
-1000
-2000
0
5
10
Hz

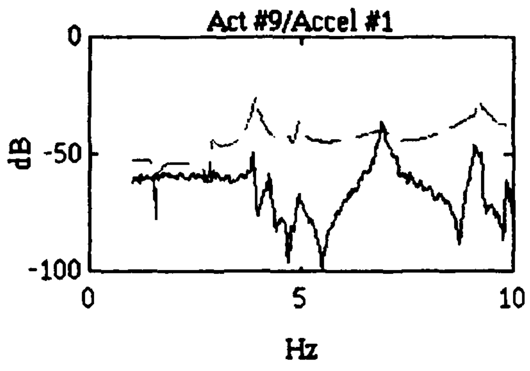
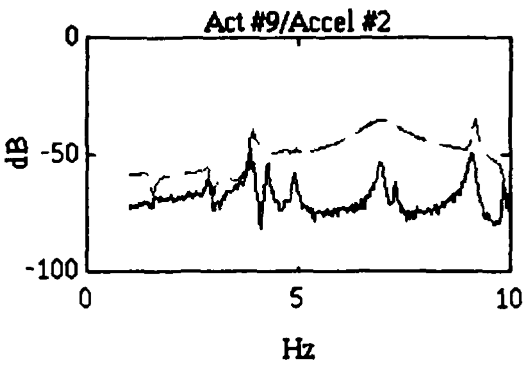
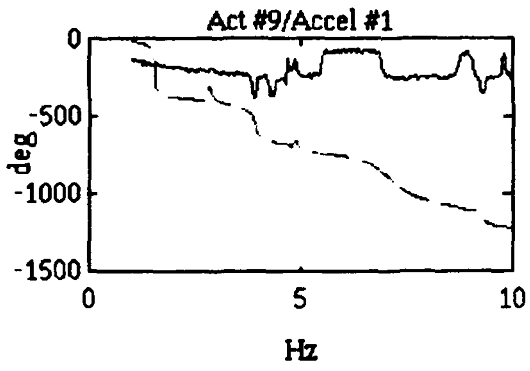
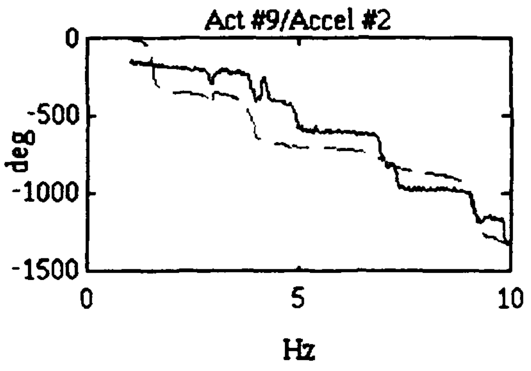
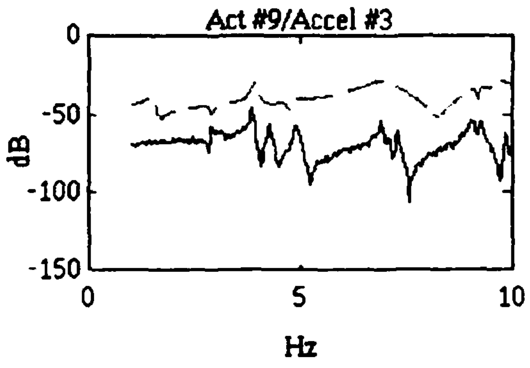
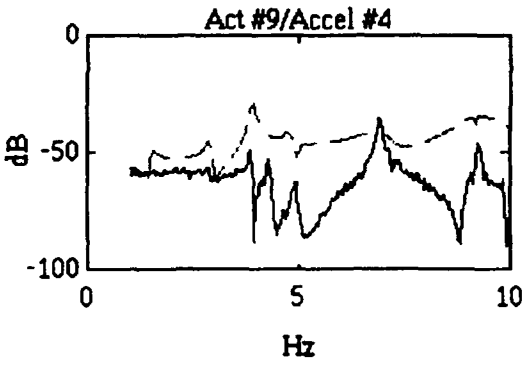
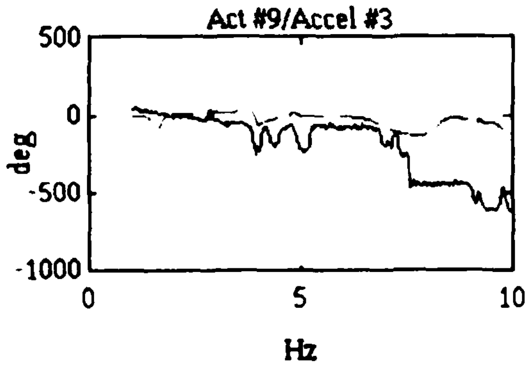
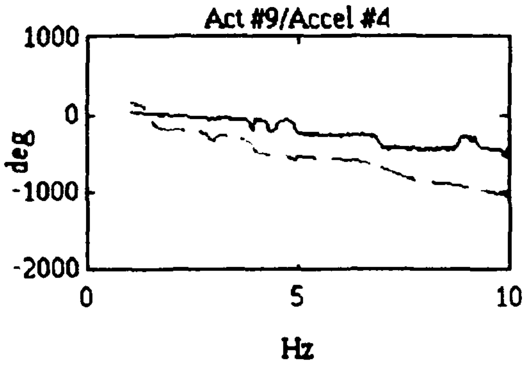

Act #6/Accel #6
deg
500
0
-500
-1000
0
5
10
Hz

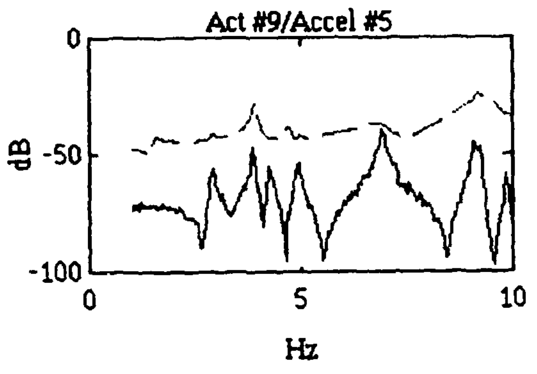

Act #6/Accel #9
dB
0
-50
-100
0
5
10
Hz

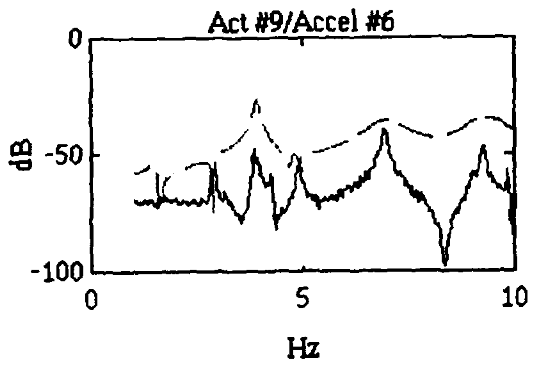

Act #6/Accel #10
dB
0
-50
-100
0
5
10
Hz

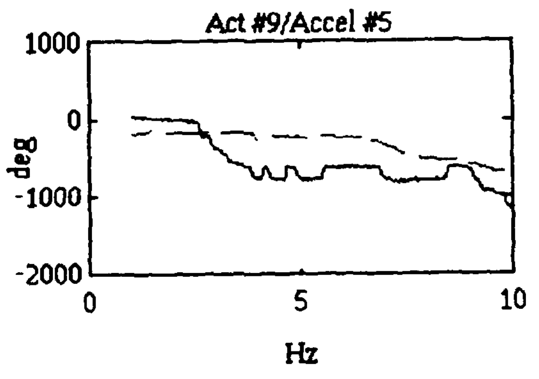

Act #6/Accel #9
deg
1000
0
-1000
-2000
0
5
10
Hz

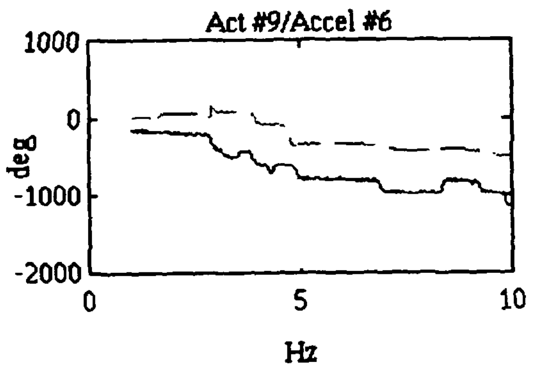

Act #6/Accel #10
deg
500
0
-500
-1000
0
5
10
Hz

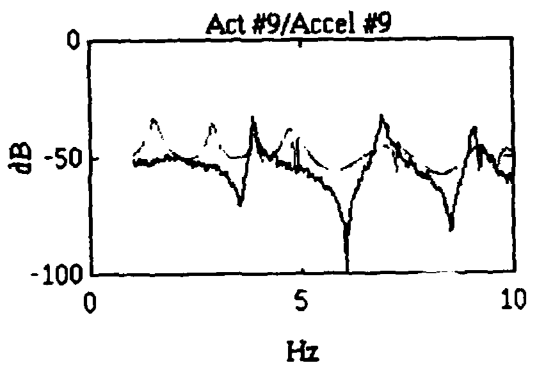

Act #9/Accel #1
dB
0
-50
-100
0
5
10
Hz

Act #9/Accel #2
dB
0
-50
-100
0
5
10
Hz

Act #9/Accel #1
deg
0
-500
-1000
-1500
0
5
10
Hz

Act #9/Accel #2
deg
0
-500
-1000
-1500
0
5
10
Hz

Act #9/Accel #3
dB
0
-50
-100
-150
0
5
10
Hz

Act #9/Accel #4
dB
0
-50
-100
0
5
10
Hz

Act #9/Accel #3
deg
500
0
-500
-1000
0
5
10
Hz

Act #9/Accel #4
deg
1000
0
-1000
-2000
0
5
10
Hz

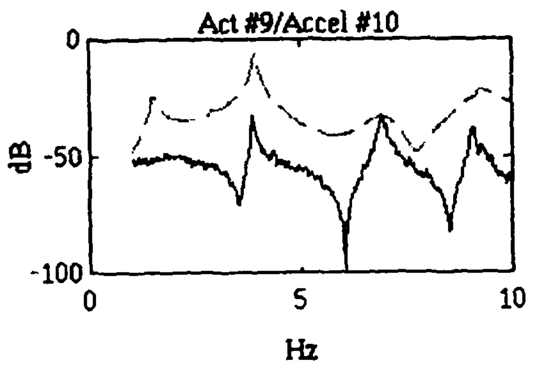

Act #9/Accel #5
dB
0
-50
-100
0
5
10
Hz

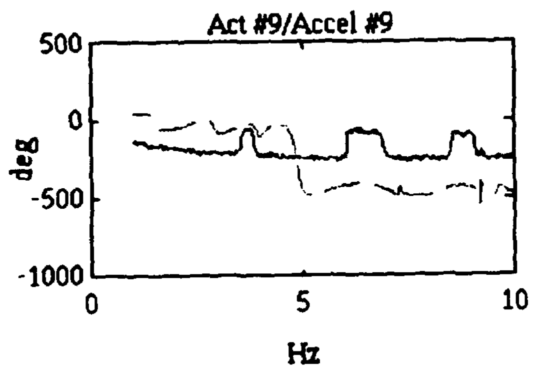

Act #9/Accel #6
dB
0
-50
-100
0
5
10
Hz

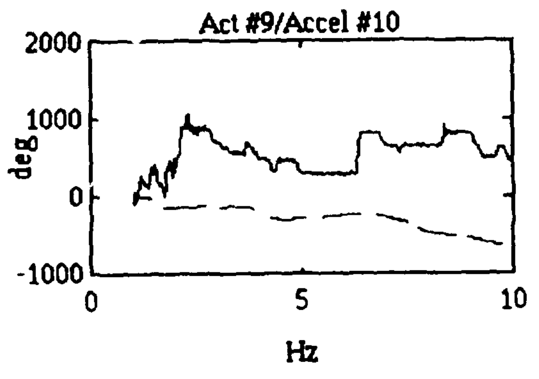

Act #9/Accel #5
deg
1000
0
-1000
-2000
0
5
10
Hz

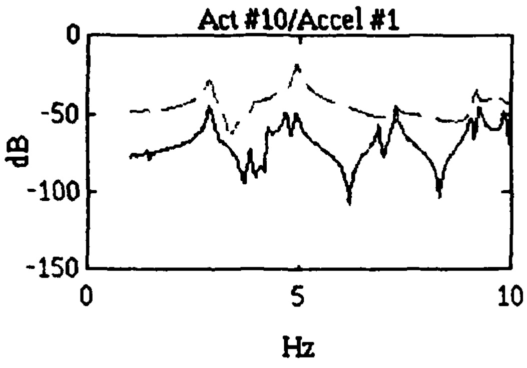

Act #9/Accel #6
deg
1000
0
-1000
-2000
0
5
10
Hz

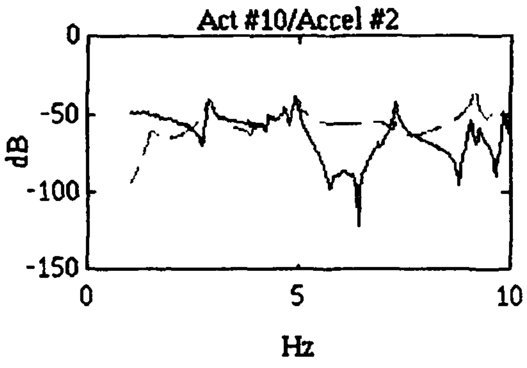

Act #9/Accel #9
dB
0
-50
-100
0
5
10
Hz

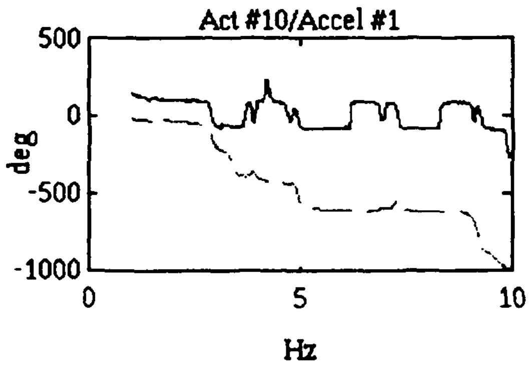

Act #9/Accel #10
dB
0
-50
-100
0
5
10
Hz

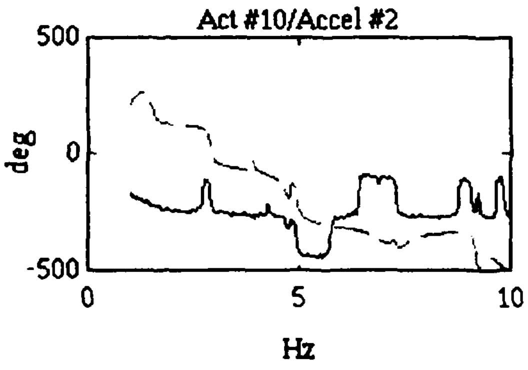

Act #9/Accel #9
deg
500
0
-500
-1000
0
5
10
Hz

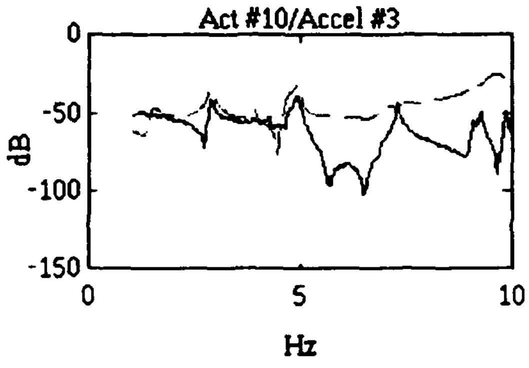

Act #9/Accel #10
deg
2000
1000
0
-1000
0
5
10
Hz

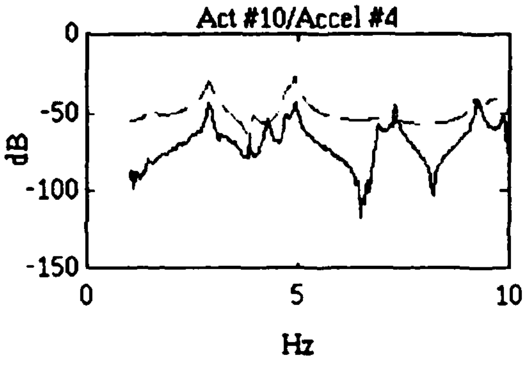

Act #10/Accel #1
0
-50
-100
-150
dB
0
5
10
Hz

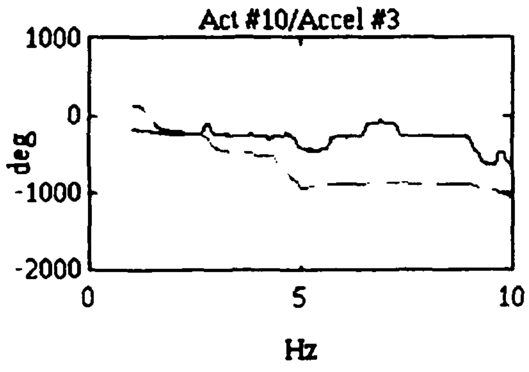

Act #10/Accel #2
0
-50
-100
-150
dB
0
5
10
Hz

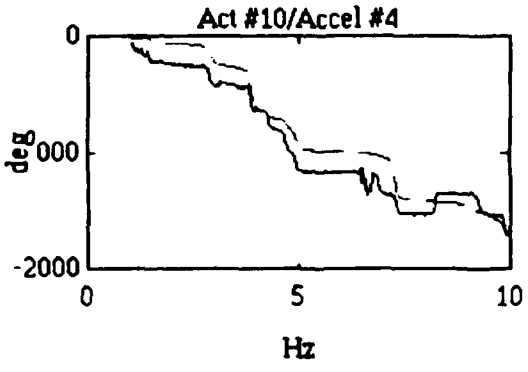

Act #10/Accel #1
500
0
-500
-1000
deg
0
5
10
Hz

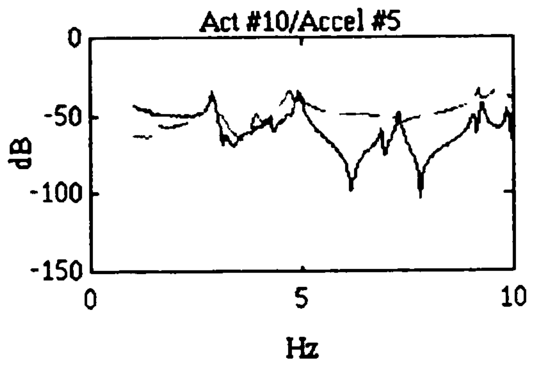

Act #10/Accel #2
500
0
-500
deg
0
5
10
Hz

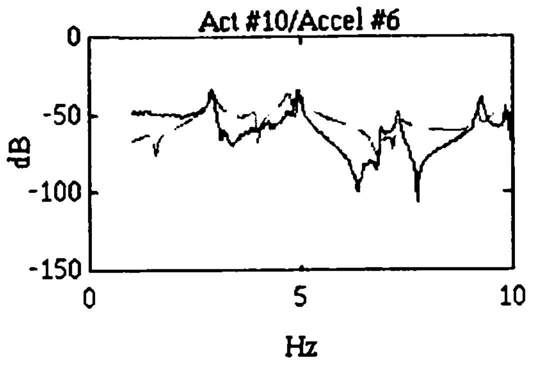

Act #10/Accel #3
0
-50
-100
-150
dB
0
5
10
Hz

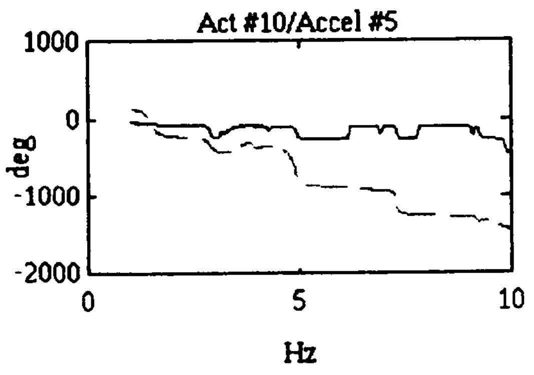

Act #10/Accel #4
0
-50
-100
-150
dB
0
5
10
Hz

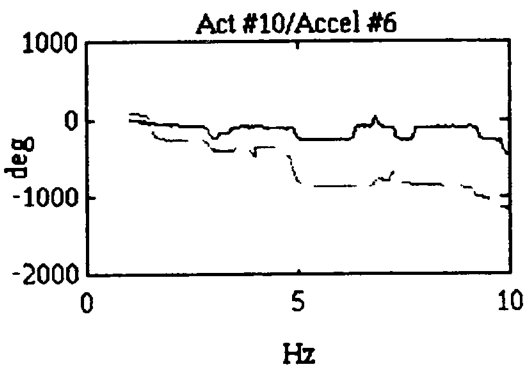

Act #10/Accel #3
1000
0
-1000
-2000
deg
0
5
10
Hz

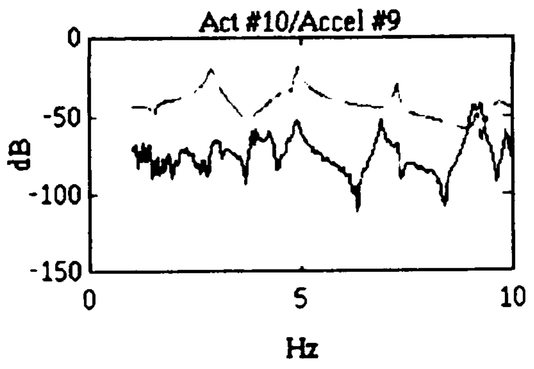

Act #10/Accel #4
0
000
-2000
deg
0
5
10
Hz

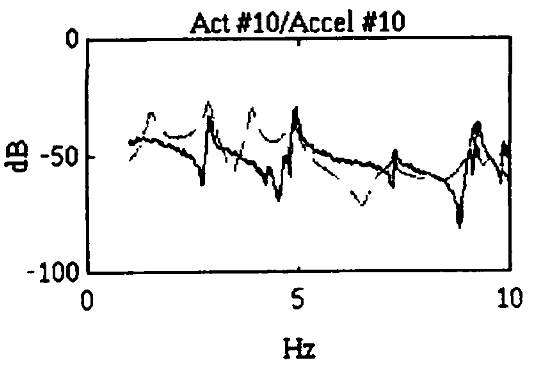

Act #10/Accel #5
dB
Hz

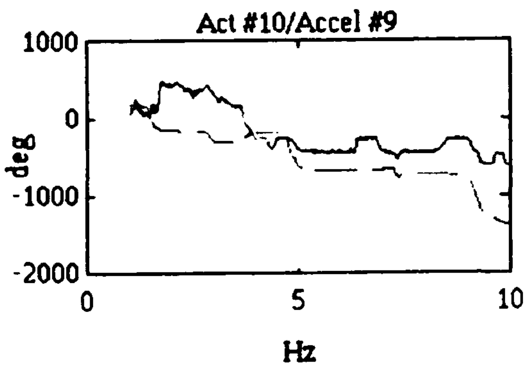

Act #10/Accel #6
dB
Hz

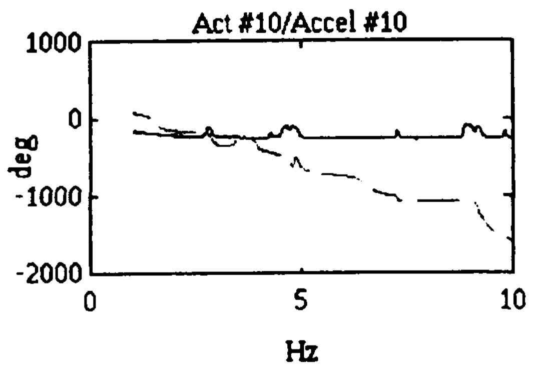

Act #10/Accel #5
deg
Hz


Act #10/Accel #6
deg
Hz


Act #10/Accel #9
dB
Hz


Act #10/Accel #10
dB
Hz


Act #10/Accel #9
deg
Hz


Act #10/Accel #10
deg
Hz

Appendix D. MATLAB Script Files

This appendix contains the 6 MATLAB script files used in this thesis. The first file was used to determine the accuracy of the ERA model by computing system modes and comparing the results to those of the truth system. The second file performed the same analysis using the BLS method to determine the system modes. The third file used the ERA method to compute the PACOSS system modes. The fourth file used the BLS method to compute the PACOSS system modes. The fifth computed the *output* matrix given the *plant* and *input* matrices from either the ERA method or the BLS method. The final file computed and compared the transfer functions from the PACOSS and the model developed by either the ERA or the BLS method.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%         eral
%    This file generates the truth information
%    for a 2 input - 2 output system with known
%             A, B, and C matrices, and then
%    identifies that system using the
%    ERA technique.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%    generate input
rand('normal');
T = 0.07;
t = [0:T:20];
A = 1;
     %  generate mesurement errors
E = 0.15;
for l = 1:length(t),
  e(l,1) = E*rand;
  e(l,2) = E/2*rand;
end;
     %  Define actual system TF
%         Truth Model
%  Mass coefficients
m1=0.61743; m2=0.19957; m3=1.6582;
%  Damping coefficients
c1=0.031633; c2=0.32738; c3=0.32918; c4=0.019153;
%  Stiffness coefficients
k1=52.058; k2=0.89183; k3=19.756; k4=178.08;
%  Inverse mass matrix
minv=diag([1/m1 1/m2 1/m3]);
%  Stiffness matrix
kk=[k1+k2 -k2 0;-k2 k3+k2 -k3;0 -k3 k3+k4];
%  Damping matrix
cc=[c1+c2 -c2 0;-c2 c3+c2 -c3;0 -c3 c3+c4];
%  The A,B,C,D matrices for the truth system
ag=[zeros(3,3) eye(3);-minv*kk -minv*cc];
bg=[0 0;0 0;0 0;minv*[0 0;1 0;0 1]];
cg=[0 1 0 0 0 0;0 0 1 0 0 0];
dg=zeros(2,2);
%  Convert to a TF
[numg1,deng1] = ss2tf(ag,bg,cg,dg,1);
[numg2,deng2] = ss2tf(ag,bg,cg,dg,2);
```

```
 [wn,z]=damp(ag);
 %  generate bode plot of truth system
 w=logspace(-1,2,150);
 [mag1,phas1]=bode(ag,bg,cg,dg,1,w);
 [mag2,phas2]=bode(ag,bg,cg,dg,2,w);
 %  Now change to discrete state space form
 [ad,bd,cd,dd] = c2dm(ag,bg,cg,dg,T,'zoh');
      %  generate truth output (impulse response for ERA)
 [y1,x] = impulse(ag,bg,cg,dg,1,t);
 [y2,x] = impulse(ag,bg,cg,dg,2,t);
   y1(1:length(t)-1,:) = y1(2:length(t),:);
   y2(1:length(t)-1,:) = y1(2:length(t),:);
      %  add measurement error to truth output signal
 for l = 1:length(t),
   y1(l,1) = y1(l,1) + e(l,1);
   y1(l,2) = y1(l,2) + e(l,2);
   y2(l,1) = y2(l,1) + e(l,1);
   y2(l,2) = y2(l,2) + e(l,2);
 end;
 y = y1+y2;
 y1 = y1';
 y2 = y2';
      %  now begin system identification
 ncols = 20;
 nrows = 60;
 inputs = 2;
 [Y] = weave(y1,y2);
 [fdk,zmk,shapesk,partfak,EMACk,sv,at,bt,ct] =
         erat(Y,1/T,ncols,nrows,inputs);
 cut = 10;
 dt = zeros(2,2);
      %  evaluate model
 wdm=2*pi*fdk;
 for l = 1:length(fdk),
   wnm(l)=wdm(l)/sqrt(1-zmk(l)/100*zmk(l)/100);
 end;
      %  generate random input for PEM
 for l = 1:length(t),
   u(l,1) = A*rand;
   u(l,2) = A*rand;
 end;
      %  generate truth output from random input
      %  for PEM
 [y,x] = lsim(ag,bg,cg,dg,u,t);
 y = y + e;
 [junk,ninput]=size(bt);
 [noutput,junk]=size(ct);
 dm=zeros(noutput,ninput);
 %  convert bm from continuous to discrete
 [atemp,Bm] = c2d(at,bt,T);
      %   now use PEM to improve C one row at
      %  a time to  determine zeros
 [rowa,cola] = size(Am);
 [rowb,colb] = size(Bm);
ai = at;
bi = bt;
ci = nan*ones(1,rowa);
di = zeros(1,colb);
ki = zeros(rowa,1);
x0i = zeros(rowa,1);
msi = modstruc(ai,bi,ci,di,ki,x0i);
parva1 = ct(1,:);
parva2 = ct(2,:);
lambdi = [];
thi1 = ms2th(msi,'d',parva1,lambdi,T);
thi2 = ms2th(msi,'d',parva2,lambdi,T);
index = [1:length(parva1)];
      %   perform system ID
th1 = pem([y(:,1) u],thi1,index,-1,1e-10,-1,-1,T);
th2 = pem([y(:,2) u],thi2,index,-1,1e-10,-1,-1,T);
      %  convert theta to SS format
[Am,Bm,cm1,dm,km1,x01] = th2ss(th1);
[Am,Bm,cm2,dm,km1,x01] = th2ss(th2);
      %  combine c matrices to get MIMO C matrix
Cm = [cm1;cm2];
Dm = dt;
for l = 1:length(t),
  u(l,1) = A*rand;
  u(l,2) = A*rand;
end;
[ya,x] = lsim(ag,bg,cg,dg,u,t);
[ym,x] = dlsim(Am,Bm,Cm,Dm,u);
%  generate bode plot of model
[magm1,phasm1]=dbode(Am,Bm,cm1,dm1,T,1,w);
[magm2,phasm2]=dbode(Am,Bm,cm1,dm1,T,2,w);
plot(t,ya(:,1),t,ym1(:,1));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%         tlst
%    This file generates the truth information
%    for a 2 input - 2 output system and then
%    identifies that system using a backward
%    batch least squares technique.  The results
%    are then run through PEM to determine the C
%    matrix
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%    generate input
rand('normal');
T = 0.08;
t = [0:T:20];
A = 1;
     %  generate measurment error
E = 0.01;
for l = 1:length(t)-1,
  e(l,1) = E*rand;
  e(l,2) = E/2*rand;
end;
     %  Define actual system TF
%         Truth Model
%  Mass coefficients
m1=0.61743; m2=0.19957; m3=1.6582;
%  Damping coefficients
c1=0.031633; c2=0.32738; c3=0.32918; c4=0.019153;
%  Stiffness coefficients
k1=52.058; k2=0.89183; k3=19.756; k4=178.08;
%  Inverse mass matrix
minv=diag([1/m1 1/m2 1/m3]);
%  Stiffness matrix
kk=[k1+k2 -k2 0;-k2 k3+k2 -k3;0 -k3 k3+k4];
%  Damping matrix
cc=[c1+c2 -c2 0;-c2 c3+c2 -c3;0 -c3 c3+c4];
%  The A,B,C,D matrices for the truth system
ag=[zeros(3,3) eye(3);-minv*kk -minv*cc];
bg=[0 0;0 0;0 0;minv*[0 0;1 0;0 1]];
cg=[0 1 0 0 0 0;0 0 1 0 0 0];
dg=zeros(2,2);
%  Convert to a TF
[numg1,deng1] = ss2tf(ag,bg,cg,dg,1);
```

```
[numg2,deng2] = ss2tf(ag,bg,cg,dg,2);
[wn,z]=damp(ag);
%  generate bode plot of truth system
w=logspace(-1,2,150);
[mag1,phas1]=bode(ag,bg,cg,dg,1,w);
[mag2,phas2]=bode(ag,bg,cg,dg,2,w);
%  Now change to discrete state space form
[ad,bd,cd,dd] = c2dm(ag,bg,cg,dg,T,'zoh');
     %  generate random inputs
for l = 1:length(t),
  u(l,1) = A*rand;
  u(l,2) = A*rand;
  e(l,1) = E*rand;
  e(l,2) = E/2*rand;
end;
     %  generate truth system response to
     %  random input
[y,x] = lsim(ag,bg,cg,dg,u,t);
%  add measurement error into the system
y(1:l,:) = y(1:l,:) + e(1:l,:);
%  now to identify the system
p = 6;     %  model order
[fd,zm,zpoles,shapes,partfac] = mimo(u,y,p,1/T);
%  find the undamped natural freqencies
wdm=2*pi*fd;
for l = 1:length(fd),
  wnm(l)=wdm(l)/sqrt(1-zm(l)/100*zm(l)/100);
end;
wnm = wnm';
%  create the state space model
n=length(fd);
nc=1;
for jh=1:n,
  wd1=2*pi*fd(jh);
  wn1=wd1/sqrt(1-zm(jh)/100*zm(jh)/100);
  z1(nc)=exp((-zm(jh)/100*wn1+j*wd1)*T);
  z1(nc+1)=conj(z1(nc));
  bm(nc,:)=partfac(jh,:);
  bm(nc+1,:)=conj(partfac(jh,:));
  cm(:,nc)=shapes(:,jh);
  cm(:,nc+1)=conj(shapes(:,jh));
  nc=nc+2;
end;
am=diag(z1);
[junk,ninput]=size(bm);
```

```
 [noutput,junk]=size(cm);
 dm=zeros(noutput,ninput);
 %  convert bm from continuous to discrete
 [atemp,Bm] = c2d(am,bm,T);
 %  convert am, Bm, and cm to real element matrices
 T1 = [1 1;j -j];
 T2 = T1;
 [r1,c1] = size(am);
 for k = 4:2:r1,
   [r,c] = size(T2);
   T2 = [T2 zeros(r,2);zeros(2,c) T1];
 end;
 Tinv = inv(T2);
 Am = T2*am*Tinv;
 Bm = T2*Bm;
 Cm = cm*Tinv;
 Dm = dm;
 %   now use PEM to improve C
 [rowa,cola] = size(Am);
 [rowb,colb] = size(Bm);
 [atemp,Bt] = c2d(Am,Bm,T);
 ai = Am;
 bi = Bt;
 ci = nan*ones(1,rowa);
 di = zeros(1,colb);
 ki = zeros(rowa,1);
 x0i = zeros(rowa,1);
 msi = modstruc(ai,bi,ci,di,ki,x0i);
 parva1 = Cm(1,:);
 parva2 = Cm(2,:);
 lambdi = [];
 thi1 = ms2th(msi,'d',parva1,lambdi,T);
 thi2 = ms2th(msi,'d',parva2,lambdi,T);
 index = [1:length(parva1)];
      %   perform system ID
 th1 = pem([y(:,1) u],thi1,index,-1,1e-10,-1,-1,T);
 th2 = pem([y(:,2) u],thi2,index,-1,1e-10,-1,-1,T);
      %  convert theta to SS format
 [Am,Bm,cm1,dm,km1,x01] = th2ss(th1);
 [Am,Bm,cm2,dm,km1,x01] = th2ss(th2);
      %  combine c matrices to get MIMO Cm matrix
 Cm = [cm1;cm2];
 %  generate bode plot of system model
 w=logspace(-1,2,150);
 [magm1,phasm1]=dbode(Am,Bt,Cm,Dm,T,1,w);
```

```
[magm2,phasm2]=dbode(Am,Bt,Cm,Dm,T,2,w);
%   compare results
for l = 1:length(t),
  u(l,1) = A*rand;
  u(l,2) = A*rand;
end;
[ya,x] = lsim(ag,bg,cg,dg,u,t);+
[ym,x] = dlsim(Am,Bm,Cm,Dm,u);
plot(t,ya(:,1),t,ym(:,1))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%          teral
%     This file computes the system model for
%     the PACOSS DTA using the ERA technique
%     It uses swept sine data to generate acceleration
%     transfer functions, then integrates those values
%     to obtain velocity TFs, and then takes
%     the inverse fourier transform to obtain
%     the impulse response. The impulse response
%     is then inserted into ERA to obtain the system
%     modes.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  load the response data and create the y vectors
               %  input #1
load a1m1
load a1m2
load a1m3
load a1m4
load a1m5
load a1m6
load a1m9
load a1m10
               %  input #2
load a2m1
load a2m2
load a2m3
load a2m4
load a2m5
load a2m6
load a2m9
load a2m10
               %  input #3
load a3m1
load a3m2
load a3m3
load a3m4
load a3m5
load a3m6
load a3m9
load a3m10
               %  input #4
```

```
load a4m1
load a4m2
load a4m3
load a4m4
load a4m5
load a4m6
load a4m9
load a4m10
          %  input #5
load a5m1
load a5m2
load a5m3
load a5m4
load a5m5
load a5m6
load a5m9
load a5m10
          %  input #6
load a6m1
load a6m2
load a6m3
load a6m4
load a6m5
load a6m6
load a6m9
load a6m10
          %  input #9
load a9m1
load a9m2
load a9m3
load a9m4
load a9m5
load a9m6
load a9m9
load a9m10
          %  input #10
load a10m1
load a10m2
load a10m3
load a10m4
load a10m5
load a10m6
load a10m9
load a10m10
%  define the frequency vector
```

```
freq = 0:0.025:10;
%  now the transfer function data must be recommbined into
%   complex numbers.
for l = 1:401,
     %  input #1
  a11(l)  = a1m1(l)  + j*a1m1(l+801);
  a12(l)  = a1m2(l)  + j*a1m2(l+801);
  a13(l)  = a1m3(l)  + j*a1m3(l+801);
  a14(l)  = a1m4(l)  + j*a1m4(l+801);
  a15(l)  = a1m5(l)  + j*a1m5(l+801);
  a16(l)  = a1m6(l)  + j*a1m6(l+801);
  a19(l)  = a1m9(l)  + j*a1m9(l+801);
  a110(l)  = a1m10(l)  + j*a1m10(l+801);
     %  input #2
  a21(l)  = a2m1(l)  + j*a2m1(l+801);
  a22(l)  = a2m2(l)  + j*a2m2(l+801);
  a23(l)  = a2m3(l)  + j*a2m3(l+801);
  a24(l)  = a2m4(l)  + j*a2m4(l+801);
  a25(l)  = a2m5(l)  + j*a2m5(l+801);
  a26(l)  = a2m6(l)  + j*a2m6(l+801);
  a29(l)  = a2m9(l)  + j*a2m9(l+801);
  a210(l)  = a2m10(l)  + j*a2m10(l+801);
     %  input #3
  a31(l)  = a3m1(l)  + j*a3m1(l+801);
  a32(l)  = a3m2(l)  + j*a3m2(l+801);
  a33(l)  = a3m3(l)  + j*a3m3(l+801);
  a34(l)  = a3m4(l)  + j*a3m4(l+801);
  a35(l)  = a3m5(l)  + j*a3m5(l+801);
  a36(l)  = a3m6(l)  + j*a3m6(l+801);
  a39(l)  = a3m9(l)  + j*a3m9(l+801);
  a310(l)  = a3m10(l)  + j*a3m10(l+801);
     %  input #4
  a41(l)  = a4m1(l)  + j*a4m1(l+801);
  a42(l)  = a4m2(l)  + j*a4m2(l+801);
  a43(l)  = a4m3(l)  + j*a4m3(l+801);
  a44(l)  = a4m4(l)  + j*a4m4(l+801);
  a45(l)  = a4m5(l)  + j*a4m5(l+801);
  a46(l)  = a4m6(l)  + j*a4m6(l+801);
  a49(l)  = a4m9(l)  + j*a4m9(l+801);
  a410(l)  = a4m10(l)  + j*a4m10(l+801);
     %  input #5
  a51(l)  = a5m1(l)  + j*a5m1(l+801);
  a52(l)  = a5m2(l)  + j*a5m2(l+801);
  a53(l)  = a5m3(l)  + j*a5m3(l+801);
  a54(l)  = a5m4(l)  + j*a5m4(l+801);
```

```
    a55(l)  = a5m5(l)  + j*a5m5(l+801);
    a56(l)  = a5m6(l)  + j*a5m6(l+801);
    a59(l)  = a5m9(l)  + j*a5m9(l+801);
    a510(l)  = a5m10(l)  + j*a5m10(l+801);
       %  input #6
    a61(l)  = a6m1(l)  + j*a6m1(l+801);
    a62(l)  = a6m2(l)  + j*a6m2(l+801);
    a63(l)  = a6m3(l)  + j*a6m3(l+801);
    a64(l)  = a6m4(l)  + j*a6m4(l+801);
    a65(l)  = a6m5(l)  + j*a6m5(l+801);
    a66(l)  = a6m6(l)  + j*a6m6(l+801);
    a69(l)  = a6m9(l)  + j*a6m9(l+801);
    a610(l)  = a6m10(l)  + j*a6m10(l+801);
       %  input #9
    a91(l)  = a9m1(l)  + j*a9m1(l+801);
    a92(l)  = a9m2(l)  + j*a9m2(l+801);
    a93(l)  = a9m3(l)  + j*a9m3(l+801);
    a94(l)  = a9m4(l)  + j*a9m4(l+801);
    a95(l)  = a9m5(l)  + j*a9m5(l+801);
    a96(l)  = a9m6(l)  + j*a9m6(l+801);
    a99(l)  = a9m9(l)  + j*a9m9(l+801);
    a910(l)  = a9m10(l)  + j*a9m10(l+801);
       %  input #10
    a101(l)  = a10m1(l)  + j*a10m1(l+801);
    a102(l)  = a10m2(l)  + j*a10m2(l+801);
    a103(l)  = a10m3(l)  + j*a10m3(l+801);
    a104(l)  = a10m4(l)  + j*a10m4(l+801);
    a105(l)  = a10m5(l)  + j*a10m5(l+801);
    a106(l)  = a10m6(l)  + j*a10m6(l+801);
    a109(l)  = a10m9(l)  + j*a10m9(l+801);
    a1010(l)  = a10m10(l)  + j*a10m10(l+801);
end;
       %  integrate to get velocity
%  set  0 Hz value to zero (no rigid body modes)
       %  input #1
v11(1)  = 0;v12(1)  = 0;v13(1)  = 0;
v14(1)  = 0;v15(1)  = 0;v16(1)  = 0;
v19(1)  = 0;v110(1)  = 0;
       %  input #2
v21(1)  = 0;v22(1)  = 0;v23(1)  = 0;
v24(1)  = 0;v25(1)  = 0;v26(1)  = 0;
v29(1)  = 0;v210(1)  = 0;
       %  input #3
v31(1)  = 0;v32(1)  = 0;v33(1)  = 0;
v34(1)  = 0;v35(1)  = 0;v36(1)  = 0;
```

```
v39(1) = 0;v310(1) = 0;
     % input #4
v41(1) = 0;v42(1) = 0;v43(1) = 0;
v44(1) = 0;v45(1) = 0;v46(1) = 0;
v49(1) = 0;v410(1) = 0;
     % input #5
v51(1) = 0;v52(1) = 0;v13(1) = 0;
v54(1) = 0;v55(1) = 0;v16(1) = 0;
v59(1) = 0;v510(1) = 0;
     % input #6
v61(1) = 0;v62(1) = 0;v13(1) = 0;
v64(1) = 0;v65(1) = 0;v16(1) = 0;
v69(1) = 0;v610(1) = 0;
     % input #9
v91(1) = 0;v92(1) = 0;v13(1) = 0;
v94(1) = 0;v95(1) = 0;v16(1) = 0;
v99(1) = 0;v910(1) = 0;
     % input #10
v101(1) = 0;v102(1) = 0;v13(1) = 0;
v104(1) = 0;v105(1) = 0;v16(1) = 0;
v109(1) = 0;v1010(1) = 0;
% divide by jw to integrate
for l = 3:401,
  s = j*2*pi*freq(l);
     % input #1
  v11(l,1) = a11(l)/s;
  v12(l,1) = a12(l)/s;
  v13(l,1) = a13(l)/s;
  v14(l,1) = a14(l)/s;
  v15(l,1) = a15(l)/s;
  v16(l,1) = a16(l)/s;
  v19(l,1) = a19(l)/s;
  v110(l,1) = a110(l)/s;
     % input #2
  v21(l,1) = a21(l)/s;
  v22(l,1) = a22(l)/s;
  v23(l,1) = a23(l)/s;
  v24(l,1) = a24(l)/s;
  v25(l,1) = a25(l)/s;
  v26(l,1) = a26(l)/s;
  v29(l,1) = a29(l)/s;
  v210(l,1) = a210(l)/s;
     % input #3
  v31(l,1) = a31(l)/s,
  v32(l,1) = a32(l)/s,
v33(1,1) = a33(1)/s;
v34(1,1) = a34(1)/s;
v35(1,1) = a35(1)/s;
v36(1,1) = a36(1)/s;
v39(1,1) = a39(1)/s;
v310(1,1) = a310(1)/s;
   % input #4
v41(1,1) = a41(1)/s;
v42(1,1) = a42(1)/s;
v43(1,1) = a43(1)/s;
v44(1,1) = a44(1)/s;
v45(1,1) = a45(1)/s;
v46(1,1) = a46(1)/s;
v49(1,1) = a49(1)/s;
v410(1,1) = a410(1)/s;
   % input #5
v51(1,1) = a51(1)/s;
v52(1,1) = a52(1)/s;
v53(1,1) = a53(1)/s;
v54(1,1) = a54(1)/s;
v55(1,1) = a55(1)/s;
v56(1,1) = a56(1)/s;
v59(1,1) = a59(1)/s;
v510(1,1) = a510(1)/s;
   % input #6
v61(1,1) = a61(1)/s;
v62(1,1) = a62(1)/s;
v63(1,1) = a63(1)/s;
v64(1,1) = a64(1)/s;
v65(1,1) = a65(1)/s;
v66(1,1) = a66(1)/s;
v69(1,1) = a69(1)/s;
v610(1,1) = a610(1)/s;
   % input #9
v91(1,1) = a91(1)/s;
v92(1,1) = a92(1)/s;
v93(1,1) = a93(1)/s;
v94(1,1) = a94(1)/s;
v95(1,1) = a95(1)/s;
v96(1,1) = a96(1)/s;
v99(1,1) = a99(1)/s;
v910(1,1) = a910(1)/s;
   % input #10
v101(1,1) = a101(1)/s;
v102(1,1) = a102(1)/s;
```

```
    v103(1,1) = a103(1)/s;
    v104(1,1) = a104(1)/s;
    v105(1,1) = a105(1)/s;
    v106(1,1) = a106(1)/s;
    v109(1,1) = a109(1)/s;
    v1010(1,1) = a1010(1)/s;
end;
%  find  0.025 Hz value (set to small number so it does not
%  interfere with results)
     %  input #1
v11(2,1) = v11(3,1);v12(2,1) = v12(3,1);v13(2,1) = v16(3,1);
v14(2,1) = v14(3,1);v15(2,1) = v15(3,1);v16(2,1) = v16(3,1);
v19(2,1) = v19(3,1);v110(2,1) = v110(3,1);
     %  input #2
v21(2,1) = v21(3,1);v22(2,1) = v22(3,1);v23(2,1) = v26(3,1);
v24(2,1) = v24(3,1);v25(2,1) = v25(3,1);v26(2,1) = v26(3,1);
v29(2,1) = v29(3,1);v210(2,1) = v210(3,1);
     %  input #3
v31(2,1) = v31(3,1);v32(2,1) = v32(3,1);v33(2,1) = v36(3,1);
v34(2,1) = v34(3,1);v35(2,1) = v35(3,1);v36(2,1) = v36(3,1);
v39(2,1) = v39(3,1);v310(2,1) = v310(3,1);
     %  input #4
v41(2,1) = v41(3,1);v42(2,1) = v42(3,1);v43(2,1) = v46(3,1);
v44(2,1) = v44(3,1);v45(2,1) = v45(3,1);v46(2,1) = v46(3,1);
v49(2,1) = v49(3,1);v410(2,1) = v410(3,1);
     %  input #5
v51(2,1) = v51(3,1);v52(2,1) = v52(3,1);v53(2,1) = v56(3,1);
v54(2,1) = v54(3,1);v55(2,1) = v55(3,1);v56(2,1) = v56(3,1);
v59(2,1) = v59(3,1);v510(2,1) = v510(3,1);
     %  input #6
v61(2,1) = v61(3,1);v62(2,1) = v62(3,1);v63(2,1) = v66(3,1);
v64(2,1) = v64(3,1);v65(2,1) = v65(3,1);v66(2,1) = v66(3,1);
v69(2,1) = v69(3,1);v610(2,1) = v610(3,1);
     %  input #9
v91(2,1) = v91(3,1);v92(2,1) = v92(3,1);v93(2,1) = v96(3,1);
v94(2,1) = v94(3,1);v95(2,1) = v95(3,1);v96(2,1) = v96(3,1);
v99(2,1) = v99(3,1);v910(2,1) = v910(3,1);
     %  input #10
v101(2,1) = v101(3,1);v102(2,1) = v102(3,1);v103(2,1) =
v106(3,1);
v104(2,1) = v104(3,1);v105(2,1) = v105(3,1);v106(2,1) =
v106(3,1);
v109(2,1) = v109(3,1);v1010(2,1) = v1010(3,1);
%   due to poor data, reduce influence of 2nd and 3rd point
for i = 2:3,
  v11(1,1) = 0.01*v11(1,1);v12(1,1) = 0.01*v12(1,1);v13(1,1)
= 0.01*v13(1,1);
  v14(1,1) = 0.01*v14(1,1);v15(1,1) = 0.01*v15(1,1);v16(1,1)
= 0.01*v16(1,1);
  v19(1,1) = 0.01*v19(1,1);v110(1,1) = 0.01*v110(1,1);
  v21(1,1) = 0.01*v21(1,1);v22(1,1) = 0.01*v22(1,1);v23(1,1)
= 0.01*v23(1,1);
  v24(1,1) = 0.01*v24(1,1);v25(1,1) = 0.01*v25(1,1);v26(1,1)
= 0.01*v26(1,1);
  v29(1,1) = 0.01*v29(1,1);v210(1,1) = 0.01*v210(1,1);
  v31(1,1) = 0.01*v31(1,1);v32(1,1) = 0.01*v32(1,1);v33(1,1)
= 0.01*v33(1,1);
  v34(1,1) = 0.01*v34(1,1);v35(1,1) = 0.01*v35(1,1);v36(1,1)
= 0.01*v36(1,1);
  v39(1,1) = 0.01*v39(1,1);v310(1,1) = 0.01*v310(1,1);
  v41(1,1) = 0.01*v41(1,1);v42(1,1) = 0.01*v42(1,1);v43(1,1)
= 0.01*v43(1,1);
  v44(1,1) = 0.01*v44(1,1);v45(1,1) = 0.01*v45(1,1);v46(1,1)
= 0.01*v46(1,1);
  v49(1,1) = 0.01*v49(1,1);v410(1,1) = 0.01*v410(1,1);
  v51(1,1) = 0.01*v51(1,1);v52(1,1) = 0.01*v52(1,1);v53(1,1)
= 0.01*v53(1,1);
  v54(1,1) = 0.01*v54(1,1);v55(1,1) = 0.01*v55(1,1);v56(1,1)
= 0.01*v56(1,1);
  v59(1,1) = 0.01*v59(1,1);v510(1,1) = 0.01*v510(1,1);
  v61(1,1) = 0.01*v61(1,1);v62(1,1) = 0.01*v62(1,1);v63(1,1)
= 0.01*v63(1,1);
  v64(1,1) = 0.01*v64(1,1);v65(1,1) = 0.01*v65(1,1);v66(1,1)
= 0.01*v66(1,1);
  v69(1,1) = 0.01*v69(1,1);v610(1,1) = 0.01*v610(1,1);
  v91(1,1) = 0.01*v91(1,1);v92(1,1) = 0.01*v92(1,1);v93(1,1)
= 0.01*v93(1,1);
  v94(1,1) = 0.01*v94(1,1);v95(1,1) = 0.01*v95(1,1);v96(1,1)
= 0.01*v96(1,1);
  v99(1,1) = 0.01*v99(1,1);v910(1,1) = 0.01*v910(1,1);
  v101(1,1) = 0.01*v101(1,1);v102(1,1) =
0.01*v102(1,1);v103(1,1) = 0.01*v103(1,1);
  v104(1,1) = 0.01*v104(1,1);v105(1,1) =
0.01*v105(1,1);v106(1,1) = 0.01*v106(1,1);
  v109(1,1) = 0.01*v109(1,1);v1010(1,1) = 0.01*v1010(1,1);
end;
%  save the velocity TF's
save tfv1.mat v11 v12 v13 v14 v15 v16 v19 v110
save tfv2.mat v21 v22 v23 v24 v25 v26 v29 v210
save tfv3.mat v31 v32 v33 v34 v35 v36 v39 v310
save tfv4.mat v41 v42 v43 v44 v45 v46 v49 v410
save tfv5.mat v51 v52 v53 v54 v55 v56 v59 v510
save tfv6.mat v61 v62 v63 v64 v65 v66 v69 v610
save tfv9.mat v91 v92 v93 v94 v95 v96 v99 v910
save tfv10.mat v101 v102 v103 v104 v105 v106 v109 v1010
%  take the inverse fourier transform to get the
%   impulse response (2048 point of data were gathered)
     %  input #1
y11 = 2048*real(ifft(v11,2048));
y12 = 2048*real(ifft(v12,2048));
y13 = 2048*real(ifft(v13,2048));
y14 = 2048*real(ifft(v14,2048));
y15 = 2048*real(ifft(v15,2048));
y16 = 2048*real(ifft(v16,2048));
y19 = 2048*real(ifft(v19,2048));
y110 = 2048*real(ifft(v110,2048));
     %  input #2
y21 = 2048*real(ifft(v21,2048));
y22 = 2048*real(ifft(v22,2048));
y23 = 2048*real(ifft(v23,2048));
y24 = 2048*real(ifft(v24,2048));
y25 = 2048*real(ifft(v25,2048));
y26 = 2048*real(ifft(v26,2048));
y29 = 2048*real(ifft(v29,2048));
y210 = 2048*real(ifft(v210,2048));
     %  input #3
y31 = 2048*real(ifft(v31,2048));
y32 = 2048*real(ifft(v32,2048));
y33 = 2048*real(ifft(v33,2048));
y34 = 2048*real(ifft(v34,2048));
y35 = 2048*real(ifft(v35,2048));
y36 = 2048*real(ifft(v36,2048));
y39 = 2048*real(ifft(v39,2048));
y310 = 2048*real(ifft(v310,2048));
     %  input #4
y41 = 2048*real(ifft(v41,2048));
y42 = 2048*real(ifft(v42,2048));
y43 = 2048*real(ifft(v43,2048));
y44 = 2048*real(ifft(v44,2048));
y45 = 2048*real(ifft(v45,2048));
y46 = 2048*real(ifft(v46,2048));
y49 = 2048*real(ifft(v49,2048));
y410 = 2048*real(ifft(v410,2048));
     %  input #5
y51 = 2048*real(ifft(v51,2048));
```

```
y52 = 2048*real(ifft(v52,2048));
y53 = 2048*real(ifft(v53,2048));
y54 = 2048*real(ifft(v54,2048));
y55 = 2048*real(ifft(v55,2048));
y56 = 2048*real(ifft(v56,2048));
y59 = 2048*real(ifft(v59,2048));
y510 = 2048*real(ifft(v510,2048));
     %  input #6
y61 = 2048*real(ifft(v61,2048));
y62 = 2048*real(ifft(v62,2048));
y63 = 2048*real(ifft(v63,2048));
y64 = 2048*real(ifft(v64,2048));
y65 = 2048*real(ifft(v65,2048));
y66 = 2048*real(ifft(v66,2048));
y69 = 2048*real(ifft(v69,2048));
y610 = 2048*real(ifft(v610,2048));
     %  input #9
y91 = 2048*real(ifft(v91,2048));
y92 = 2048*real(ifft(v92,2048));
y93 = 2048*real(ifft(v93,2048));
y94 = 2048*real(ifft(v94,2048));
y95 = 2048*real(ifft(v95,2048));
y96 = 2048*real(ifft(v96,2048));
y99 = 2048*real(ifft(v99,2048));
y910 = 2048*real(ifft(v910,2048));
     %  input #10
y101 = 2048*real(ifft(v101,2048));
y102 = 2048*real(ifft(v102,2048));
y103 = 2048*real(ifft(v103,2048));
y104 = 2048*real(ifft(v104,2048));
y105 = 2048*real(ifft(v105,2048));
y106 = 2048*real(ifft(v106,2048));
y109 = 2048*real(ifft(v109,2048));
y1010 = 2048*real(ifft(v1010,2048));
%  remove uneeded vectors to clear memory space
clear v11 v12 v13 v14 v15 v16 v19 v110
clear v21 v22 v23 v24 v25 v26 v29 v210
clear v31 v32 v33 v34 v35 v36 v39 v310
clear v41 v42 v43 v44 v45 v46 v49 v410
clear v51 v52 v53 v54 v55 v56 v59 v510
clear v61 v62 v63 v64 v65 v66 v69 v610
clear v91 v92 v93 v94 v95 v96 v99 v910
clear v101 v102 v103 v104 v105 v106 v109 v1010
%  combine the impulse response vectors into matrices
z1 = [y11 y12 y13 y14 y15 y16 y19 y110];
```

```
z2 = [y21 y22 y23 y24 y25 y26 y29 y210];
z3 = [y31 y32 y33 y34 y35 y36 y39 y310];
z4 = [y41 y42 y43 y44 y45 y46 y49 y410];
z5 = [y51 y52 y53 y54 y55 y56 y59 y510];
z6 = [y61 y62 y63 y64 y65 y66 y69 y610];
z9 = [y91 y92 y93 y94 y95 y96 y99 y910];
z10 = [y101 y102 y103 y104 y105 y106 y109 y1010];
% remove uneeded vectors to clear memory space
clear y11 y12 y13 y14 y15 y16 y19 y110
clear y21 y22 y23 y24 y25 y26 y29 y210
clear y31 y32 y33 y34 y35 y36 y39 y310
clear y41 y42 y43 y44 y45 y46 y49 y410
clear y51 y52 y53 y54 y55 y56 y59 y510
clear y61 y62 y63 y64 y65 y66 y69 y610
clear y91 y92 y93 y94 y95 y96 y09 y910
clear y101 y102 y103 y104 y105 y106 y109 y1010

% change from row to column form
y1 = z1';
y2 = z2';
y3 = z3';
y4 = z4';
y5 = z5';
y6 = z6';
y9 = z9';
y10 = z10';
     % begin system identification

ncols = 90;
nrows = 400;
inputs = 8;
% set sample rate
fs = 51.2;
[Y] = weave(y1,y2,y3,y4,y5,y6,y9,y10);
[fd,zm,shapes,partfac,EMAC,sv] =
era(Y,fs,ncols,nrows,inputs);
cut = 0.0;
% eliminate undesired modes
keep = input('how many modes will you keep ');
for l = 1:keep,
  k = input('keep mode #: ');
  fdk(l,1) = fd(k);
  zmk(l,1) = zm(k);
  shapesk(:,l) = shapes(:,k);
  partfak(l,:) = partfac(k,:);
  EMACk(1,1) = EMAC(k,1);
end;
%  evaluate model
wdm=2*pi*fdk;
for l = 1:length(fdk),
  wnm(l)=wdm(l)/sqrt(1-zmk(l)/100*zmk(l)/100);
end;
wnm = wnm/2/pi;
%  Create state space model (A,B,C, and D matrices)
[am,bm,cm,dm] =
erastate(fdk,zmk,shapesk,partfak,EMACk,fs,cut);
%  convert bm from continuous to discrete
[atemp,Bm] = c2d(am,bm,1/fs);
%  convert am,Bm, and cm to real element matrices only
T1 = [1 1;j -j];
T = T1;
[r1,c1] = size(am);
for k = 4:2:r1,
  [r,c] = size(T);
  T = [T zeros(r,2);zeros(2,c) T1];
end;
Tinv = inv(T);
A = T*am*Tinv;
B = T*Bm;
C = cm*Tinv;
D = dm;
save tabc.mat A B C D
save tresul.mat fd zm shapes partfac EMAC sv
save tkept.mat wnm fdk zmk shapesk partfak EMACk
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%         pera
%    This file computes the system model for
%    the PACOSS DTA using the ERA technique
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  load the bias for each accelerometer
load nom12;
bias(1) = mean(Timei3);
bias(2) = mean(Timei4);
load nom34;
bias(3) = mean(Timei3);
bias(4) = mean(Timei4);
load nom56;
bias(5) = mean(Timei3);
bias(6) = mean(Timei4);
load nom910;
bias(9) = mean(Timei3);
bias(10) = mean(Timei4);
%  load the response data and create the y vectors
          %  input #1
load ran11.mat;
u1(:,1) = Timei1;
y1(:,1) = Timei2;
load ran12.mat;
u1(:,2) = Timei1;
y1(:,2) = Timei2;
load ran13.mat;
u1(:,3) = Timei1;
y1(:,3) = Timei2;
load ran14.mat;
u1(:,4) = Timei1;
y1(:,4) = Timei2;
load ran15.mat;
u1(:,5) = Timei1;
y1(:,5) = Timei2;
load ran16.mat;
u1(:,6) = Timei1;
y1(:,6) = Timei2;
load ran19.mat;
u1(:,7) = Timei1;
y1(:,7) = Timei2;
```

```
%          input #2
load ran21.mat;
u2(:,1) = Timei1;
y2(:,1) = Timei2;
load ran22.mat;
u2(:,2) = Timei1;
y2(:,2) = Timei2;
load ran23.mat;
u2(:,3) = Timei1;
y2(:,3) = Timei2;
load ran24.mat;
u2(:,4) = Timei1;
y2(:,4) = Timei2;
load ran25.mat;
u2(:,5) = Timei1;
y2(:,5) = Timei2;
load ran26.mat;
u2(:,6) = Timei1;
y2(:,6) = Timei2;
load ran29.mat;
u2(:,7) = Timei1;
y2(:,7) = Timei2;
%          input #3
load ran31.mat;
u3(:,1) = Timei1;
y3(:,1) = Timei2;
load ran32.mat;
u3(:,2) = Timei1;
y3(:,2) = Timei2;
load ran33.mat;
u3(:,3) = Timei1;
y3(:,3) = Timei2;
load ran34.mat;
u3(:,4) = Timei1;
y3(:,4) = Timei2;
load ran35.mat;
u3(:,5) = Timei1;
y3(:,5) = Timei2;
load ran36.mat;
u3(:,6) = Timei1;
y3(:,6) = Timei2;
load ran39.mat;
u3(:,7) = Timei1;
y3(:,7) = Timei2;
%          input #4
```

```
load ran41.mat;
u4(:,1) = Timei1;
y4(:,1) = Timei2;
load ran42.mat;
u4(:,2) = Timei1;
y4(:,2) = Timei2;
load ran43.mat;
u4(:,3) = Timei1;
y4(:,3) = Timei2;
load ran44.mat;
u4(:,4) = Timei1;
y4(:,4) = Timei2;
load ran45.mat;
u4(:,5) = Timei1;
y4(:,5) = Timei2;
load ran46.mat;
u4(:,6) = Timei1;
y4(:,6) = Timei2;
load ran49.mat;
u4(:,7) = Timei1;
y4(:,7) = Timei2;
%          input #5
load ran51.mat;
u5(:,1) = Timei1;
y5(:,1) = Timei2;
load ran52.mat;
u5(:,2) = Timei1;
y5(:,2) = Timei2;
load ran53.mat;
u5(:,3) = Timei1;
y5(:,3) = Timei2;
load ran54.mat;
u5(:,4) = Timei1;
y5(:,4) = Timei2;
load ran55.mat;
u5(:,5) = Timei1;
y5(:,5) = Timei2;
load ran56.mat;
u5(:,6) = Timei1;
y5(:,6) = Timei2;
load ran59.mat;
u5(:,7) = Timei1;
y5(:,7) = Timei2;
%          input #6
load ran61.mat;
```

```
u6(:,1) = Timei1;
y6(:,1) = Timei2;
load ran62.mat;
u6(:,2) = Timei1;
y6(:,2) = Timei2;
load ran63.mat;
u6(:,3) = Timei1;
y6(:,3) = Timei2;
load ran64.mat;
u6(:,4) = Timei1;
y6(:,4) = Timei2;
load ran65.mat;
u6(:,5) = Timei1;
y6(:,5) = Timei2;
load ran66.mat;
u6(:,6) = Timei1;
y6(:,6) = Timei2;
load ran69.mat;
u6(:,7) = Timei1;
y6(:,7) = Timei2;
%          input #9
load ran91.mat;
u9(:,1) = Timei1;
y9(:,1) = Timei2;
load ran92.mat;
u9(:,2) = Timei1;
y9(:,2) = Timei2;
load ran93.mat;
u9(:,3) = Timei1;
y9(:,3) = Timei2;
load ran94.mat;
u9(:,4) = Timei1;
y9(:,4) = Timei2;
load ran95.mat;
u9(:,5) = Timei1;
y9(:,5) = Timei2;
load ran96.mat;
u9(:,6) = Timei1;
y9(:,6) = Timei2;
load ran99.mat;
u9(:,7) = Timei1;
y9(:,7) = Timei2;
%          input #10
load ran101.mat;
u10(:,1) = Timei1;
```

```
y10(:,1) = Timei2;
load ran102.mat;
u10(:,2) = Timei1;
y10(:,2) = Timei2;
load ran103.mat;
u10(:,3) = Timei1;
y10(:,3) = Timei2;
load ran104.mat;
u10(:,4) = Timei1;
y10(:,4) = Timei2;
load ran105.mat;
u10(:,5) = Timei1;
y10(:,5) = Timei2;
load ran106.mat;
u10(:,6) = Timei1;
y10(:,6) = Timei2;
load ran109.mat;
u10(:,7) = Timei1;
y10(:,7) = Timei2;
%  remove acclerometer bias form each measurment
[r,c] = size(y1);
for l = 1:r,
  for k = 1:7,
    y1(l,k) = y1(l,k) - bias(k);
    y2(l,k) = y2(l,k) - bias(k);
    y3(l,k) = y3(l,k) - bias(k);
    y4(l,k) = y4(l,k) - bias(k);
    y5(l,k) = y5(l,k) - bias(k);
    y6(l,k) = y6(l,k) - bias(k);
    y9(l,k) = y9(l,k) - bias(k);
    y10(l,k) = y10(l,k) - bias(k);
  end;
end;
%    combine the y and u values to get a MIMO system
Y = y1+y2+y3+y4+y5+y6+y9+y10;
U1 = [u1(:,1) u2(:,1) u3(:,1) u4(:,1) u5(:,1) u6(:,1)
u9(:,1) u10(:,1)];
U2 = [u1(:,2) u2(:,2) u3(:,2) u4(:,2) u5(:,2) u6(:,2)
u9(:,2) u10(:,2)];
U3 = [u1(:,3) u2(:,3) u3(:,3) u4(:,3) u5(:,3) u6(:,3)
u9(:,3) u10(:,3)];
U4 = [u1(:,4) u2(:,4) u3(:,4) u4(:,4) u5(:,4) u6(:,4)
u9(:,4) u10(:,4)];
U5 = [u1(:,5) u2(:,5) u3(:,5) u4(:,5) u5(:,5) u6(:,5)
u9(:,5) u10(:,5)];
```

```
U6 = [u1(:,6) u2(:,6) u3(:,6) u4(:,6) u5(:,6) u6(:,6)
u9(:,6) u10(:,6)];
U9 = [u1(:,7) u2(:,7) u3(:,7) u4(:,7) u5(:,7) u6(:,7)
u9(:,7) u10(:,7)];
U = U1+U2+U3+U4+U5+U6+U9;
     %  now begin system identification
p = 10;
%[fd1,zm1,shapes1,partfa1,EMAC1] =
mimo(U1,Y(:,1),p,Fsample);
%[fd2,zm2,shapes2,partfa2,EMAC2] =
mimo(U2,Y(:,2),p,Fsample);
%[fd3,zm3,shapes3,partfa3,EMAC3] =
mimo(U3,Y(:,3),p,Fsample);
%[fd4,zm4,shapes4,partfa4,EMAC4] =
mimo(U4,Y(:,4),p,Fsample);
%[fd5,zm5,shapes5,partfa5,EMAC5] =
mimo(U5,Y(:,5),p,Fsample);
%[fd6,zm6,shapes6,partfa6,EMAC6] =
mimo(U6,Y(:,6),p,Fsample);
%[fd9,zm9,shapes9,partfa9,EMAC9] =
mimo(U9,Y(:,7),p,Fsample);
[fd,zm,poles,shapes,partfac] = mimo(U,Y,p,Fsample);
cut = 0.001;
%  compute natural frequency
%  evaluate model
wdm=2*pi*fd;
for l = 1:length(fd),
  wnm(l)=wdm(l)/sqrt(1-zm(l)*zm(l));
end;
%  Create state space model A matrix
k = 1;
for l = 1:length(wnm),
  z(k)=exp((-zm(l)*wnm(l)+j*wdm(l))/Fsample);
  z(k+1)=conj(z(k));
  B(k,:)=partfac(l,:);
  B(k+1,:)=conj(B(k,:));
  cm(:,k)=shapes(:,l);
  cm(:,k+1)=conj(cm(:,k));
  k = k + 2;
end;
am = diag(z);
[r1,c1] = size(am);
[atemp,bm] = c2d(am,B,1/Fsample);
dm = zeros(7,8);
%  transform to real matrices
```

```
T1 = [1 1;j -j];
T = T1;
for k = 4:2:r1,
  [r,c] = size(T);
  T = [T zeros(r,2);zeros(2,c) T1];
end;
Tinv = inv(T);
A = T*am*Tinv;
B = T*bm;
C = cm*Tinv;
D = dm;
save tmabc.mat A B C D
save tmresul.mat  wnm fd zm shapes partfac
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%         tmdl
%
%    This file creates the model  C
%    matrix for the PACOSS.  The A
%    and B matrices are computed from
%    either the ERA or BLS methods.
%    This script file uses PEM to
%    approximate the C matrix one row
%    at a time.  It only uses the data from
%    the colocated actuator/accelerometer
%    pairs.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  input the PACOSS data needed to generate
%  the C matrix
%         Input #1
load tran11.mat;
u1(:,1)  = Timei4(45:512);
y1(:,1)  = Timei1(45:512);
%         input #2
load tran22.mat;
u2(:,2)  = Timei4(45:512);
y2(:,2)  = Timei1(45:512);
%         input #3
load tran33.mat;
u3(:,3)  = Timei4(45:512);
y3(:,3)  = Timei1(45:512);
%         input #4
load tran44.mat;
u4(:,4)  = Timei4(45:512);
y4(:,4)  = Timei1(45:512);
%         input #5
load tran55.mat
u5(:,5)  = Timei4(45:512);
y5(:,5)  = Timei1(45:512);
%         input #6
load tran66.mat;
u6(:,6)  = Timei4(45:512);
y6(:,6)  = Timei1(45:512);
%         input #9
load tran99.mat;
```

```
u9(:,7) = Timei4(45:512);
y9(:,7) = Timei1(45:512);
%          input #10
load tran1010.mat;
u10(:,8) = Timei4(45:512);
y10(:,8) = Timei1(45:512);
%  subtract out the accelerometer bias
%  the mean is used because the actual system
%  has no net motion, so the mean is zero
y1(:,1) = y1(:,1) - mean(y1(:,1));
y2(:,2) = y2(:,2) - mean(y2(:,2));
y3(:,3) = y3(:,3) - mean(y3(:,3));
y4(:,4) = y4(:,4) - mean(y4(:,4));
y5(:,5) = y5(:,5) - mean(y5(:,5));
y6(:,6) = y6(:,6) - mean(y6(:,6));
y9(:,7) = y9(:,7) - mean(y9(:,7));
y10(:,8) = y10(:,8) - mean(y10(:,8));
%   use PEM to estimate  C matrix
%  load the A, B, and C matrices
load tabc.mat
%load tmabc.mat
T = 1/25.6;
[rowa,cola] = size(A);
[rowb,colb] = size(B);
[rowc,colc] = size(C);
ai = A;
bi1 = B(:,1);
bi2 = B(:,2);
bi3 = B(:,3);
bi4 = B(:,4);
bi5 = B(:,5);
bi6 = B(:,6);
bi9 = B(:,7);
bi10 = B(:,8);
ci = nan*ones(1,rowa);
di = zeros(1,1);
ki = zeros(rowa,1);
x0i = zeros(rowa,1);
msi1 = modstruc(ai,bi1,ci,di,ki,x0i);
msi2 = modstruc(ai,bi2,ci,di,ki,x0i);
msi3 = modstruc(ai,bi3,ci,di,ki,x0i);
msi4 = modstruc(ai,bi4,ci,di,ki,x0i);
msi5 = modstruc(ai,bi5,ci,di,ki,x0i);
msi6 = modstruc(ai,bi6,ci,di,ki,x0i);
msi9 = modstruc(ai,bi9,ci,di,ki,x0i);
```

```
msi10 = modstruc(ai,bi10,ci,di,ki,x0i);
% initial guess for the C matrix
parva1 =  C(1,:);
parva2 =  C(2,:);
parva3 =  C(3,:);
parva4 =  C(4,:);
parva5 =  C(5,:);
parva6 =  C(6,:);
parva9 =  C(7,:);
parva10 = C(8,:);
lambdi = [];
thi1 = ms2th(msi1,'d',parva1,lambdi,T);
thi2 = ms2th(msi2,'d',parva2,lambdi,T);
thi3 = ms2th(msi3,'d',parva3,lambdi,T);
thi4 = ms2th(msi4,'d',parva4,lambdi,T);
thi5 = ms2th(msi5,'d',parva5,lambdi,T);
thi6 = ms2th(msi6,'d',parva6,lambdi,T);
thi9 = ms2th(msi9,'d',parva9,lambdi,T);
thi10 = ms2th(msi10,'d',parva10,lambdi,T);
index = [1:length(parva1)];

     %   perform system ID

th1 = pem([y1(:,1) u1(:,1)],thi1,index,-1,1e-10,-1,-1,T);
th2 = pem([y2(:,2) u2(:,2)],thi2,index,-1,1e-10,-1,-1,T);
th3 = pem([y3(:,3) u3(:,3)],thi3,index,-1,1e-10,-1,-1,T);
th4 = pem([y4(:,4) u4(:,4)],thi4,index,-1,1e-10,-1,-1,T);
th5 = pem([y5(:,5) u5(:,5)],thi5,index,-1,1e-10,-1,-1,T);
th6 = pem([y6(:,6) u6(:,6)],thi6,index,-1,1e-10,-1,-1,T);
th9 = pem([y9(:,7) u9(:,7)],thi9,index,-1,1e-10,-1,-1,T);
th10 = pem([y10(:,8) u10(:,8)],thi10,index,-1,1e-10,-1,-
1,T);
     %  convert theta to SS format
[am,bm1,cm1,dm,km1,x01] = th2ss(th1);
[am,bm2,cm2,dm,km1,x01] = th2ss(th2);
[am,bm3,cm3,dm,km1,x01] = th2ss(th3);
[am,bm4,cm4,dm,km1,x01] = th2ss(th4);
[am,bm5,cm5,dm,km1,x01] = th2ss(th5);
[am,bm6,cm6,dm,km1,x01] = th2ss(th6);
[am,bm9,cm9,dm,km1,x01] = th2ss(th9);
[am,bm10,cm10,dm,km1,x01] = th2ss(th10);
     %  combine c matrices to get MIMO C matrix
am = A;
bm = B;
cm = [cm1;cm2;cm3;cm4;cm5;cm6;cm9;cm10];
```

```
dm = D;
save tmodl.mat am bm cm dm
%save tmmodl.mat am bm cm dm
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%             tfcompl
%
%    This file compares the TF of the truth system
%    and the model system.  Magnitues plots are in
%    db, and phase plots are in degrees.  The TF's
%    are from 1 to 10 Hz
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   load the PACOSS TF data and
%   generate TF plots from PACOSS data
FreqV = 0:0.025:10;
%          input #1
load tfv1
  TF11 = 20*log10(abs(v11(41:401)));
  TF12 = 20*log10(abs(v12(41:401)));
  TF13 = 20*log10(abs(v13(41:401)));
  TF14 = 20*log10(abs(v14(41:401)));
  TF15 = 20*log10(abs(v15(41:401)));
  TF16 = 20*log10(abs(v16(41:401)));
  TF19 = 20*log10(abs(v19(41:401)));
  phase11 =
(180/pi)*unwrap(atan2(imag(v11(41:401)),real(v11(41:401))));
  phase12 =
(180/pi)*unwrap(atan2(imag(v12(41:401)),real(v12(41:401))));
  phase13 =
(180/pi)*unwrap(atan2(imag(v13(41:401)),real(v13(41:401))));
  phase14 =
(180/pi)*unwrap(atan2(imag(v14(41:401)),real(v14(41:401))));
  phase15 =
(180/pi)*unwrap(atan2(imag(v15(41:401)),real(v15(41:401))));
  phase16 =
(180/pi)*unwrap(atan2(imag(v16(41:401)),real(v16(41:401))));
  phase19 =
(180/pi)*unwrap(atan2(imag(v19(41:401)),real(v19(41:401))));

%          input #2
load tfv2
  TF21 = 20*log10(abs(v21(41:401)));
  TF22 = 20*log10(abs(v22(41:401)));
  TF23 = 20*log10(abs(v23(41:401)));
  TF24 = 20*log10(abs(v24(41:401)));
  TF25 = 20*log10(abs(v25(41:401)));
  TF26 = 20*log10(abs(v26(41:401)));
  TF29 = 20*log10(abs(v29(41:401)));
  phase21 =
(180/pi)*unwrap(atan2(imag(v21(41:401)),real(v21(41:401))));
  phase22 =
(180/pi)*unwrap(atan2(imag(v22(41:401)),real(v22(41:401))));
  phase23 =
(180/pi)*unwrap(atan2(imag(v23(41:401)),real(v23(41:401))))
  phase24 =
(180/pi)*unwrap(atan2(imag(v24(41:401)),real(v24(41:401))));
  phase25 =
(180/pi)*unwrap(atan2(imag(v25(41:401)),real(v25(41:401))));
  phase26 =
(180/pi)*unwrap(atan2(imag(v26(41:401)),real(v26(41:401))));
  phase29 =
(180/pi)*unwrap(atan2(imag(v29(41:401)),real(v29(41:401))));

%          input #3
load tfv3
  TF31 = 20*log10(abs(v31(41:401)));
  TF32 = 20*log10(abs(v32(41:401)));
  TF33 = 20*log10(abs(v33(41:401)));
  TF34 = 20*log10(abs(v34(41:401)));
  TF35 = 20*log10(abs(v35(41:401)));
  TF36 = 20*log10(abs(v36(41:401)));
  TF39 = 20*log10(abs(v39(41:401)));
  phase31 =
(180/pi)*unwrap(atan2(imag(v31(41:401)),real(v31(41:401))));
  phase32 =
(180/pi)*unwrap(atan2(imag(v32(41:401)),real(v32(41:401))));
  phase33 =
(180/pi)*unwrap(atan2(imag(v33(41:401)),real(v33(41:401))));
  phase34 =
(180/pi)*unwrap(atan2(imag(v34(41:401)),real(v34(41:401))));
  phase35 =
(180/pi)*unwrap(atan2(imag(v35(41:401)),real(v35(41:401))));
  phase36 =
(180/pi)*unwrap(atan2(imag(v36(41:401)),real(v36(41:401))));
  phase39 =
(180/pi)*unwrap(atan2(imag(v39(41:401)),real(v39(41:401))));
%          input #4
load tfv4
  TF41 = 20*log10(abs(v41(41:401)));
  TF42 = 20*log10(abs(v42(41:401)));
  TF43 = 20*log10(abs(v43(41:401)));
  TF44 = 20*log10(abs(v44(41:401)));
  TF45 = 20*log10(abs(v45(41:401)));
  TF46 = 20*log10(abs(v46(41:401)));
  TF49 = 20*log10(abs(v49(41:401)));
  phase41 =
(180/pi)*unwrap(atan2(imag(v41(41:401)),real(v41(41:401))));
  phase42 =
(180/pi)*unwrap(atan2(imag(v42(41:401)),real(v42(41:401))));
  phase43 =
(180/pi)*unwrap(atan2(imag(v43(41:401)),real(v43(41:401))));
  phase44 =
(180/pi)*unwrap(atan2(imag(v44(41:401)),real(v44(41:401))));
  phase45 =
(180/pi)*unwrap(atan2(imag(v45(41:401)),real(v45(41:401))));
  phase46 =
(180/pi)*unwrap(atan2(imag(v46(41:401)),real(v46(41:401))));
  phase49 =
(180/pi)*unwrap(atan2(imag(v49(41:401)),real(v49(41:401))));
%         input #5
load tfv5
  TF51 = 20*log10(abs(v51(41:401)));
  TF52 = 20*log10(abs(v52(41:401)));
  TF53 = 20*log10(abs(v53(41:401)));
  TF54 = 20*log10(abs(v54(41:401)));
  TF55 = 20*log10(abs(v55(41:401)));
  TF56 = 20*log10(abs(v56(41:401)));
  TF59 = 20*log10(abs(v59(41:401)));
  phase51 =
(180/pi)*unwrap(atan2(imag(v51(41:401)),real(v51(41:401))));
  phase52 =
(180/pi)*unwrap(atan2(imag(v52(41:401)),real(v52(41:401))));
  phase53 =
(180/pi)*unwrap(atan2(imag(v53(41:401)),real(v53(41:401))));
  phase54 =
(180/pi)*unwrap(atan2(imag(v54(41:401)),real(v54(41:401))));
  phase55 =
(180/pi)*unwrap(atan2(imag(v55(41:401)),real(v55(41:401))));
  phase56 =
(180/pi)*unwrap(atan2(imag(v56(41:401)),real(v56(41:401))));
  phase59 =
(180/pi)*unwrap(atan2(imag(v59(41:401)),real(v59(41:401))));
%         input #6
load tfv6
  TF61 = 20*log10(abs(v61(41:401)));
  TF62 = 20*log10(abs(v62(41:401)));
  TF63 = 20*log10(abs(v63(41:401)));
  TF64 = 20*log10(abs(v64(41:401)));
  TF65 = 20*log10(abs(v65(41:401)));
  TF66 = 20*log10(abs(v66(41:401)));
  TF69 = 20*log10(abs(v69(41:401)));
  phase61 =
(180/pi)*unwrap(atan2(imag(v61(41:401)),real(v61(41:401))));
  phase62 =
(180/pi)*unwrap(atan2(imag(v62(41:401)),real(v62(41:401))));
  phase63 =
(180/pi)*unwrap(atan2(imag(v63(41:401)),real(v63(41:401))));
  phase64 =
(180/pi)*unwrap(atan2(imag(v64(41:401)),real(v64(41:401))));
  phase65 =
(180/pi)*unwrap(atan2(imag(v65(41:401)),real(v65(41:401))));
  phase66 =
(180/pi)*unwrap(atan2(imag(v66(41:401)),real(v66(41:401))));
  phase69 =
(180/pi)*unwrap(atan2(imag(v69(41:401)),real(v69(41:401))));

%          input #9
load tfv9
  TF91 = 20*log10(abs(v91(41:401)));
  TF92 = 20*log10(abs(v92(41:401)));
  TF93 = 20*log10(abs(v93(41:401)));
  TF94 = 20*log10(abs(v94(41:401)));
  TF95 = 20*log10(abs(v95(41:401)));
  TF96 = 20*log10(abs(v96(41:401)));
  TF99 = 20*log10(abs(v99(41:401)));
  phase91 =
(180/pi)*unwrap(atan2(imag(v91(41:401)),real(v91(41:401))));
  phase92 =
(180/pi)*unwrap(atan2(imag(v92(41:401)),real(v92(41:401))));
  phase93 =
(180/pi)*unwrap(atan2(imag(v93(41:401)),real(v93(41:401))));
  phase94 =
(180/pi)*unwrap(atan2(imag(v94(41:401)),real(v94(41:401))));
  phase95 =
(180/pi)*unwrap(atan2(imag(v95(41:401)),real(v95(41:401))));
  phase96 =
(180/pi)*unwrap(atan2(imag(v96(41:401)),real(v96(41:401))));
  phase99 =
(180/pi)*unwrap(atan2(imag(v99(41:401)),real(v99(41:401))));
```

```
%         input #10
load tfv10
  TF101 = 20*log10(abs(v101(41:401)));
  TF102 = 20*log10(abs(v102(41:401)));
  TF103 = 20*log10(abs(v103(41:401)));
  TF104 = 20*log10(abs(v104(41:401)));
  TF105 = 20*log10(abs(v105(41:401)));
  TF106 = 20*log10(abs(v106(41:401)));
  TF109 = 20*log10(abs(v109(41:401)));
  TF1010 = 20*log10(abs(v1010(41:401)));
phase101=(180/pi)*unwrap(atan2(imag(v101(41:401)),real(v101(
41:401))));
phase102=(180/pi)*unwrap(atan2(imag(v102(41:401)),real(v102(
41:401))));
phase103=(180/pi)*unwrap(atan2(imag(v103(41:401)),real(v103(
41:401))));
phase104=(180/pi)*unwrap(atan2(imag(v104(41:401)),real(v104(
41:401))));
phase105=(180/pi)*unwrap(atan2(imag(v105(41:401)),real(v105(
41:401))));
phase106=(180/pi)*unwrap(atan2(imag(v106(41:401)),real(v106(
41:401))));
phase109=(180/pi)*unwrap(atan2(imag(v109(41:401)),real(v109(
41:401))));
phase1010=(180/pi)*unwrap(atan2(imag(v1010(41:401)),real(v10
10(41:401))));

%  load model A, B, C, and D matrices
load tmodl
%load tmmodl
%  generate bode plot of system model
wrad = 2*pi*FreqV;
%  reduce gain of system
cm = 0.01*cm;
[magm1,phas1]=dbode(am,bm,cm,dm,1/51.2,1,wrad);
[magm2,phas2]=dbode(am,bm,cm,dm,1/51.2,2,wrad);
[magm3,phas3]=dbode(am,bm,cm,dm,1/51.2,3,wrad);
[magm4,phas4]=dbode(am,bm,cm,dm,1/51.2,4,wrad);
[magm5,phas5]=dbode(am,bm,cm,dm,1/51.2,5,wrad);
[magm6,phas6]=dbode(am,bm,cm,dm,1/51.2,6,wrad);
[magm9,phas9]=dbode(am,bm,cm,dm,1/51.2,7,wrad);
[magm10,phas10]=dbode(am,bm,cm,dm,1/51.2,8,wrad);
%  compare truth and model
plot(FreqV(41:401),[TF11,20*log10(magm1(41:401,1))])
save tfinal.mat am bm cm dm
```

Vita

Anthony R. Nash was born on 24 Dec 1966 in Salem, OR. After graduating from Douglas McKay High School in Salem, OR in May 1985, he attended the United States Air Force Academy. He graduated in May 1989 with a Bachelor of Science degree in Astronautical Engineering. His first assignment was Deputy Program Manager for the Space Based Laser and then as the Deputy for System Engineering and Integration for the Air Force Strategic Defense Initiative Program Office at Los Angeles AFB, CA. In May 1992, he entered the School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio to pursue a Master of Science degree in Astronautical Engineering.

Permanent Address:

4566 Shawn Ct. N.E.

Salem, OR 97305

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE Dec 1993	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE MODELING OF A LARGE UNDAMPED SPACE STRUCTURE USING TIME DOMAIN TECHNIQUES	5. FUNDING NUMBERS
6. AUTHOR(S) ANTHONY R. NASH CAPT, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology WPAFB OH 45433-6583	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GA/ENY/93D-7
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Kevin Slimak OL-AC PL/VTS 9 Antares Rd. Edwards AFB, CA 93524-7620	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The first phase of this project was the selection of accurate time domain state space modeling techniques by comparing the models generated by various methods, against a known system. The techniques selected were the Eigensystem Realization Algorithm and the Backward Least Squares method. These modeling techniques were applied to the vibration data from the PACOSS dynamic test article. The resulting model transfer function was then compared to the frequency response function of the actual system, and was found to be an accurate match for the system poles, but not for the system zeros.

14. SUBJECT TERMS Large Space Structures, Dynamic Control, System Modes, Structural Vibration, Flexible Spacecraft, System Identification	15. NUMBER OF PAGES 114
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL