NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ARTIFICIAL BOUNDARY CONDITIONS IN SENSITIVITY BASED FINITE ELEMENT MODEL UPDATING AND STRUCTURAL DAMAGE DETECTION

by

Constance R. S. Fernandez

December 2005

Thesis Advisor: Joshua H. Gordis

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2005	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Artificial Boundary Conditions in Sensitivity Based Finite Element Model Updating and Structural Damage Detection			**5. FUNDING NUMBERS**
6. AUTHOR(S) Constance R S Fernandez			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

A finite element (FE) model is a computational representation of a given structure. In order for the FE model to accurately predict structure response, the model is "updated" or improved. This thesis investigates the use of artificial boundary conditions in sensitivity-based model updating and damage detection. A comparative analysis was conducted on the accuracy of error identification and location with respect to the artificial boundary conditions imposed and the number of modes retained. Results are demonstrated with actual test data from a simple structure.

14. SUBJECT TERMS Finite Element Model Sensitivity Artificial Boundary Conditions			**15. NUMBER OF PAGES** 165
			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

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ARTIFICIAL BOUNDARY CONDITIONS IN SENSITIVITY BASED FINITE ELEMENT MODEL UPDATING AND STRUCTURAL DAMAGE DETECTION

Constance R. S. Fernandez
Lieutenant, United States Navy
B.S. Electrical Engineering, Illinois Institute of Technology, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2005

Author:

Constance R. S. Fernandez

Approved by:

Joshua H. Gordis
Thesis Advisor

Anthony J. Healey
Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A finite element (FE) model is a computational representation of a given structure. In order for the FE model to accurately predict structure response, the model is "updated" or improved. This thesis investigates the use of artificial boundary conditions in sensitivity-based model updating and damage detection. A comparative analysis was conducted on the accuracy of error identification and location with respect to the artificial boundary conditions imposed and the number of modes retained. Results are demonstrated with actual test data from a simple structure.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would to thank Professor Gordis for his unparalleled understanding and patience throughout this project.

I would also like to thank Elias Fernandez for his overwhelming support in my educational endeavors.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A finite element (FE) model is a computational model of a structure based on mathematical formulas and estimations. A FE model can be defined by a large number of physical and material parameters. These parameters which include but limited to dimensional properties of structural elements, moduli of elasticity, and densities are also called adjustable parameters because these properties can be adjusted to improve the accuracy of FE model's behavior with respect to the structure of which it is a model. In order to correctly predict structure response, it is necessary to have the most accurate FE model. The accuracy of a FE model is improved by a method called "updating." Updating involves measuring structural response, comparing the measured data to the FE response and making the appropriate adjustments to the FE model parameters.

As stated above, a FE model can be defined by a large number of parameters however; only a small number of parameters can be measured from the structure in a modal test. The measured parameters which define a structure are modal parameters, i.e. mode shapes and the natural frequencies of system. The disparity in number of measured modal parameters versus the number of adjustable parameters defines an undetermined problem. In other words, there are too many unknowns for an accurate solution to be found. Therefore, a procedure is needed to collect more data thus increase the amount of known quantities. Given that modal parameters which are measured in the lab are based on boundary conditions of the structure, one method of collecting more data is by applying different boundary conditions to the same structure and measuring the modal parameters. This procedure is effective but costly and time consuming.

A more efficient method is to identify additional and distinct modal parameters from the same modal test without need for physical modification of the structure or for more time in the lab. One method is that of applying Artificial Boundary Conditions (ABC) to the measured data. The term "artificial" indicates that no physical change in the boundary conditions has been made but the application results in additional modal parameters that correspond exactly to the modal parameters found when combinations of measured coordinates are constrained to ground. [Gordis 1999]

When performing a modal test, a choice is made either by ease of placement or importance of location as to which set of coordinates are equipped with transducers. In either case the resulting set of coordinates is the “analysis set” or “ASET.” ABC can only be applied to ASET coordinates. In a set of spatially incomplete frequency response function (FRF), the ABC are the boundary conditions that define an “omitted coordinate system (OCS)”, or “OSET.” The ABC method can yield several sets of modal parameters for a given experimental database due to the fact that a spatially incomplete FRF matrix is identically equal to the FRF matrix that is calculated from the exact dynamic reduction. [Gordis 1999]

Each of the ABC system modal parameters can be used to generate a sensitivity matrix, which is the link between the known modal parameters and unknown adjustable parameters of the FE model which are needed in model updating. In addition to providing a larger number of frequencies for a system using one measured database, the ABC can reduce the ill conditioning in the solution of the sensitivity equations by ensuring linear independence of equations. The application of ABC reduces the difficulties in determining which FE parameters are dissimilar to that of the actual model i.e. that are in error; therefore, improving error localization and FE model updating.

II. THEORY

The physical and material parameters, which define a finite element model, are categorized into three types: mass, damping, or stiffness parameters. These parameters are used in the equations of motion (EOM) (Eq 2.1), which define an *n* DOF system.

$$[M]\{\ddot{x}(t)\}+[C]\{\dot{x}(t)\}+[K]\{x(t)\}=\{f\} \tag{2.1}$$

Expanded (Eq 2.1) is written as

$$\begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{bmatrix}\begin{Bmatrix} \ddot{x}_1(t) \\ \\ \ddot{x}_n(t) \end{Bmatrix}+\begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}\begin{Bmatrix} \dot{x}_1(t) \\ \\ \dot{x}_n(t) \end{Bmatrix}+\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \ddots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix}\begin{Bmatrix} x_1(t) \\ \\ x_n(t) \end{Bmatrix}=\begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{2.2}$$

where $\{x\}$ is a vector of n coordinates needed to uniquely specify the configuration of the system at all instants of time and $\{f\}$ is a vector of excitation associated with each coordinate x. [M], [C] and [K] are the mass, damping, and stiffness matrices, respectively. All matrices are square and symmetric.

In a modal test only a limited number of coordinates or DOF can be measured. This small group of DOF is considered the "ASET" or analytical set and accelerometers are mounted on these coordinates. The remaining unmeasured DOF of the system are considered the "OSET" or omitted coordinate set.

A. OMITTED COORDINATE SET

Rewriting Eq (2.2) in the frequency domain for steady state harmonic excitation yields the steady state equation of dynamic equilibrium.

$$\left[\begin{bmatrix} K_{11} & \cdots & K_{1n} \\ \vdots & \ddots & \vdots \\ K_{n1} & \cdots & K_{nn} \end{bmatrix}-\Omega^2\begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{bmatrix}+j\Omega\begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}\right]+\begin{Bmatrix} x_1(t) \\ \\ x_n(t) \end{Bmatrix}=\begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{2.3}$$

where Ω (rad/sec) is the forcing frequency.

Since the total number of DOF, *n*, is the union of ASET and OSET, Eq (2.3) can be rewritten in portioned form as

$$\left[\begin{bmatrix} K_{aa} & K_{ao} \\ K_{oa} & K_{oo} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{aa} & M_{ao} \\ M_{oa} & M_{oo} \end{bmatrix} + j\Omega \begin{bmatrix} C_{aa} & C_{ao} \\ C_{oa} & C_{oo} \end{bmatrix}\right] \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{Bmatrix} f_a \\ f_o \end{Bmatrix} \tag{2.4}$$

where the subscript "a" refers to the ASET and "o" refers to the OSET.

For use with Artificial Boundary Conditions it is necessary to identify the relationship between OSET and ASET coordinates. For simplification in the development of required relationship, an undamped system where [C] = 0 is used and Eq (2.4) is rewritten as

$$\left[\begin{bmatrix} K_{aa} & K_{ao} \\ K_{oa} & K_{oo} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{aa} & M_{ao} \\ M_{oa} & M_{oo} \end{bmatrix}\right] \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{Bmatrix} f_a \\ f_o \end{Bmatrix} \tag{2.5}$$

A relationship between OSET and ASET coordinates implies that a solution for the OSET coordinates, $\{x_o\}$ must be developed in terms of ASET coordinates, $\{x_a\}$. If there is no excitation acting on the omitted coordinates then $\{f_o\} = \{0\}$ and Eq. (2.5) is rewritten as

$$\left[\begin{bmatrix} K_{aa} & K_{ao} \\ K_{oa} & K_{oo} \end{bmatrix} - \Omega^2 \begin{bmatrix} M_{aa} & M_{ao} \\ M_{oa} & M_{oo} \end{bmatrix}\right] \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{Bmatrix} f_a \\ 0 \end{Bmatrix} \tag{2.6}$$

When Eq (2.6) is mathematically rearranged two equations evolve. However, only one equation is needed to solve for OSET coordinates in terms of ASET coordinates and it is

$$[K_{oa}]\{x_a\} + [K_{oo}]\{x_o\} - \Omega^2 [[M_{oa}]\{x_a\} + [M_{oo}]\{x_o\}] = 0 \tag{2.7}$$

Grouping like terms and solving for x_o, Eq. (2.7) becomes

$$\{x_o\} = \left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]^{-1} \left[-K_{oo}^{-1} K_{oa} + \Omega^2 K_{oo}^{-1} M_{oa}\right] \{x_a\} \tag{2.8}$$

By definition, the bracketed inverse term is

$$\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]^{-1} = \frac{1}{Det\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right]} Adj\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right] \tag{2.9}$$

where Det[…] indicates the determinant and Adj[…] indicates the adjoint matrix. From Eq. (2.9), it is clear that the solution for $\{x_o\}$ in Eq. (2.8) does not exist at those frequencies, which satisfy

$$Det\left[I - \Omega^2 K_{oo}^{-1} M_{oo}\right] = 0 \tag{2.10}$$

Therefore, the relationship between the ASET and OSET does not exist at those same frequencies. By definition the frequencies which satisfy Eq. (2.10) are the eigenvalues of the system defined by $[K_{oo}]$ and $[M_{oo}]$. Since both the stiffness and mass matrices are composed solely of the OSET coordinates, the resulting system is the OSET system and the resulting frequencies are the OSET frequencies.

Given the fact that the characteristics, i.e. eigenvalues and eigenvectors, of a system are defined by the unrestrained DOF of said system, the OSET system is a system where the OSET coordinates are unrestrained and the ASET coordinates are fully constrained to ground.

Remembering that only a limited number of DOF responses are measured in a modal test it is necessary to understand how the spatially incomplete test data can be used in the identification of OSET frequencies. However, before OSET frequencies identification can be discussed it is important to understand how spatially incomplete data compares to spatially complete data and how it can be used in experiments.

Consider first the complete FRF matrix, which is *n* x *n* and then suppose that the description of the system is limited to only certain coordinates, thus ignoring what happens at the other coordinates. (Please note that ignoring coordinates is not the same as supposing that the other coordinates do not exist.) The resulting reduced model is now of the order N x N. It is clear that because the basic system has not been altered and that it still has the same number of DOF even though it was decided not to describe all of DOF, the elements which remain in the reduced FRF matrix are identical to the corresponding elements in the full *n* x *n* matrix. (Ewins 1982) In other words the reduced matrix is formed simply by extracting the elements of interest and leaving behind those to be ignored and still contains all modal information of a fully decribed matrix.

Given the before mentioned description of a reduced matrix and that the experimentally measured FRF matrix is based on a limited number of accelerometers i.e. a limited number of responses are measured and the rested are ignored, it is clear to see that the measured FRF matrix retains all the modal content of the orginial. Also given that fact that the measured FRF matrix implicity defines a dynamically reduced impedence model the reduction of the FE model is pursued in the same manner. (Gordis 1996)

The method of exact dynamic reduction and its use in the identification of OSET frequencies is discussed in the following section.

B. EXACT DYNAMIC REDUCTION AND FRF MATRICES

The complete or "full-order" model of a structure has a FRF matrix of infinite dimensions, which account for the infinite number of DOF, *n*, of the system.

$$[H] = \begin{bmatrix} H_{aa} & H_{ao} \\ H_{oa} & H_{oo} \end{bmatrix} \tag{2.11}$$

However, the measured FRF matrix is of finite dimension because only a limited set of transducers or accelerometers, which correspond to the ASET are used to measure the FRFs of the system. The resulting FRF matrix is a reduced order model of the system.

$$\left[\overline{H^x}\right] = [H_{aa}] \tag{2.12}$$

where the superscript *x* denotes an experimentally measured FRF and the overbar notation indicates a reduced model. $[H_{aa}]$ which is defined by Eq. (2.12) represents a structural dynamic model that has been reduced using exact dynamic reduction (Gordis 1996).

Given the identity [Z][H] = [I], $[H] = [Z]^{-1}$, $[Z] = [H]^{-1}$ where [Z] is the system impedance matrix a relationship can be found between the measured ASET data and the omitted OSET data of the system. In other words, $[H_{aa}]$ can be defined in terms of the OSET frequencies.

Impedance and FRF matrices are shown as partitioned matrices.

$$\begin{bmatrix} Z_{aa} & Z_{oa} \\ Z_{ao} & Z_{oo} \end{bmatrix} \begin{bmatrix} H_{aa} & H_{oa} \\ H_{ao} & H_{oo} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.13}$$

Four equations evolve from multiplying the impedance and FRF matrices. However, only two equations are needed for the development of the required ASET-OSET relationship.

$$\begin{aligned} Z_{aa}H_{aa} + Z_{ao}H_{oa} &= 1 \\ Z_{oa}H_{aa} + Z_{oo}H_{oa} &= 0 \end{aligned} \tag{2.14 a-b}$$

After rearranging and shifting sides the relationship between ASET coordinates and OSET coordinates is found to be

$$\left[H_{aa}\right] = \left[Z_{aa} - Z_{ao}Z_{oo}^{-1}Z_{oa}\right]^{-1} \tag{2.15}$$

where the impedance of the reduced order model is linearly independent of the impedance of the full order model (Gordis 1999).

Given that $\left[Z\right] = \left[K - \Omega^2 M - j\Omega C\right]$ or $\left[K - \Omega^2 M\right]$ when [C] = 0 and that the eigenvalues or natural frequencies of a system are defined when $\left[K - \Omega^2 M\right]$ = 0 then it is easy to see that the OSET system dynamics are present in Eq (2.15) in the term $[Z_{oo}]^{-1}$.

Using the applicable identity of [Z][H]=[I] and the fact that $[Z_{oo}]^{-1}$ is undefined when $\left[K_{oo} - \Omega^2 M_{oo}\right]$ = 0 or that every element in $[Z_{oo}]^{-1}$ is singular at the OSET frequencies, it is easily seen that the elements of $[H_{aa}]^{-1}$ will also be singular at the OSET natural frequencies (Gordis 1999). Graphically speaking a plot of $[H_{aa}]^{-1}$ versus frequency shows that the function peaks correspond to the OSET frequencies of the given system.

Given that a spatially incomplete FRF matrix implicitly defines a dynamically reduced impedance model and that from such a model the OSET frequencies are defined, it is concluded that a reduced model retains the modal content of the original model. This conclusion states that a reduced model created by retaining only rows and columns associated with the ASET coordinates from a full, *n* DOF, FE generated FRF matrix will fully represent the structure being tested.

C. DRIVING POINT FREQUENCY RESPONSE FUNCTION

Following a mathematical derivation of the driving point formula an example is presented to demonstrate a unique characteristic of driving point function leaving for the chapter how this characteristic is used in ABC application.

A driving point function or as Ewins refers "point mobility" is a function where the response coordinate and the excitation coordinate are identical. The transfer function or Ewins' "transfer mobility" is a function where the response and excitation coordinates are different. (Ewins 1982) The resultant FRF curve of a system develops from modal contribution of each function or simply stated a complete FRF curve of a system is a summation of all the individual driving point and transfer functions of the system.

Recalling $[Z]=\left[K-\Omega^2 M-j\Omega C\right]$ and the identity [Z] = [H]$^{-1}$ Eq. (2.3) is rewritten as

$$\begin{bmatrix} Z_{11} & \cdots & Z_{1n} \\ \vdots & \ddots & \vdots \\ Z_{n1} & \cdots & Z_{nn} \end{bmatrix} \begin{Bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{Bmatrix} = \begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \text{ or } \begin{Bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{Bmatrix} = \begin{bmatrix} H_{11} & \cdots & H_{1n} \\ \vdots & \ddots & \vdots \\ H_{n1} & \cdots & H_{nn} \end{bmatrix} \begin{Bmatrix} f_1 \\ \vdots \\ f_n \end{Bmatrix} \tag{2.16}$$

Eq (2.16) is rewritten in modal coordinates by applying $\{x\} = [\Phi]\{q\}$ where Φ is the mass normalized mode shape and q is the generalized coordinate

$$[Z][\Phi]\{q\}=\{f\} \tag{2.17}$$

Premultipling by $[\Phi]^T$ and expanding [Z] in terms of K,M,C yields

$$\left[[\Phi]^T[K][\Phi]-\Omega^2[\Phi]^T[M][\Phi]+j\Omega[\Phi]^T[C][\Phi]\right]\{q\}=[\Phi]^T\{\Im\} \tag{2.18}$$

Using orthogonality, $[\Phi]^T[M][\Phi]=1$ and assuming proportional damping, $\begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix}$

Eq (2.18) simplifies to

$$\left[\omega_i^2-\Omega^2+2j\zeta\Omega\omega_i\right]\{q\}=[\Phi]^T\{\Im\} \tag{2.19}$$

where ω_i is the natural frequency of the ***i***th mode, ζ is the damping ratio and the modal impedance matrix $\left[\omega_i^2-\Omega^2+2j\zeta\Omega\omega_i\right]$ is diagonal.

By premultipling by $[\Phi]$ and using $\{\Im\} = [\Phi]^T \{f\}$ Eq (2.19) is transformed back into physical coordinates resulting in

$$\{x\} = [\Phi] \left[\frac{1}{\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i} \right] [\Phi]^T \{f\} \tag{2.20}$$

Recalling Eq (2.16), $\{x\} = [H]\{f\}$

$$[H(\Omega)] = [\Phi] \left[\frac{1}{\omega_i^2 - \Omega^2 + 2j\zeta\Omega\omega_i} \right] [\Phi]^T \tag{2.21}$$

[H(Ω)] can also be written as

$$[H(\Omega)] = \sum_{k=1}^{\text{mod}es} \frac{\{\Phi^k\}\{\Phi^k\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k} \tag{2.22}$$

or any element,

$$[H_{ij}(\Omega)] = \sum_{k=1}^{\text{mod}es} \frac{\{\Phi_i^k\}\{\Phi_j^k\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k} \tag{2.23}$$

To demonstrate the characteristics of both a driving point and transfer function on the complete FRF an example of a 2 DOF system shown in Figure 1 is used.

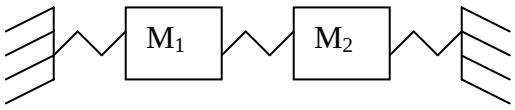

Figure 1. 2 DOF System

The matrices of this 2 DOF system are

Stiffness matrix: $[K] = \begin{bmatrix} K_1 + K_2 & -K_2 \\ -K_2 & K_2 + K_3 \end{bmatrix}$

Mass Matrix: $[M] = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}$

Setting $M_1 = M_2 = 1.0$ and $K_1 = K_2 = K_3 = 0.4$

Mode shapes: $\{\Phi^1\} = \begin{Bmatrix} -0.7071 \\ -0.7071 \end{Bmatrix}$ $\{\Phi^2\} = \begin{Bmatrix} -0.7071 \\ 0.7071 \end{Bmatrix}$

Natural frequencies: $\{\omega(rad/\sec)\} = \begin{Bmatrix} 0.6325 \\ 1.0954 \end{Bmatrix}$

Using Eq (2.23) $[H_{ij}(\Omega)] = \sum_{k=1}^{modes} \frac{\{\Phi_i^k\}\{\Phi_j^k\}^T}{\omega_k^2 - \Omega^2 + 2j\zeta\Omega\omega_k}$ and the above stated modal data for this undamped 2 DOF

The driving point function is

$$[H_{11}(\Omega)] = \frac{\{\Phi_1^1\}\{\Phi_1^1\}^T}{\omega_1^2 - \Omega^2} + \frac{\{\Phi_1^2\}\{\Phi_1^2\}^T}{\omega_2^2 - \Omega^2} = \frac{\{-0.7071\}\{-0.7071\}^T}{0.4 - \Omega^2} + \frac{\{-0.7071\}\{-0.7071\}^T}{1.2 - \Omega^2}$$

and the transfer function [H_{12}] is

$$[H_{12}(\Omega)] = \frac{\{\Phi_1^1\}\{\Phi_2^1\}^T}{\omega_1^2 - \Omega^2} + \frac{\{\Phi_1^2\}\{\Phi_2^2\}^T}{\omega_2^2 - \Omega^2} = \frac{\{-0.7071\}\{-0.7071\}^T}{0.4 - \Omega^2} + \frac{\{-0.7071\}\{0.7071\}^T}{1.2 - \Omega^2}$$

The obviously difference between the driving point and transfer function is the sign of the modal constant or numerator of the second mode. The individual modal curves shown in Figure 2 do not show this sign difference given that the curves are plotted on a logarithmic scale. However, the driving point and transfer functions which are summations of all modes and are also shown in Figure 2 and do show the sensitivity of sign changes in the modal constant.

Notice in the driving point function [H_{11}] in the upper plot of Figure 2 at frequencies below the first natural frequency, indicated by a shape peak or resonance, that both terms have the same sign and are thus additive, making the total FRF curve higher than either compoment. Due to the logarithmic scale the contribution of the second mode at these low frequencies is relatively insignificant. This observation of additive behavior can also be applied to higher frequencies, above the second natural frequency, where the total plot is slightly higher than that of the second mode alone. However, in the region between the natural frequencies where the model curves have

opposite signs, modal curves cross, the resultant is zero characteristized on a logarithmic plot as an anti-resonance, very similar a system resonance. A similar conclusion can be drawn between the additive and substractive characteristics of the individual modes in the transfer function shown in the lower plot of Figure 2. At very low and very high frequencies, the resultant FRF curve lies just below that of the nearest individual modal curve while in the region between the resonances, the two modal curves have the same signs and thus are additive making the magnitude of FRF curve at cross over is the sum of both modal curves. (Edwins 1982).

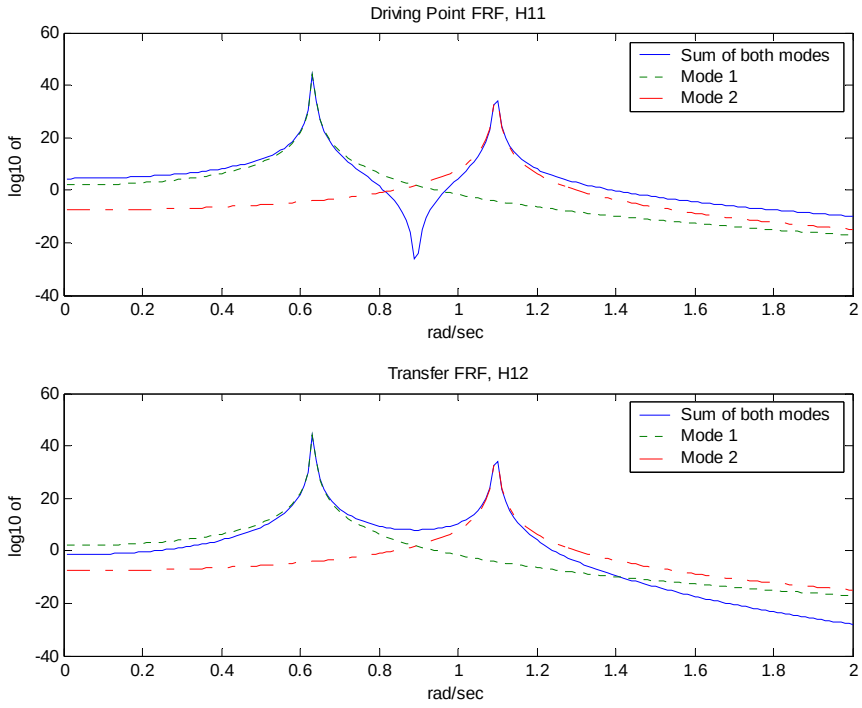

Figure 2. Plot of Driving Point [H_{11}] and Transfer Function [H_{12}] of 2 DOF System

The principles illustrated in this example can be extended to any number of degrees of freedom, thus demonstrating a fundamental rule that if two consecutive modes have the same sign for the modal constants, then there will be an anti-resonance at some frequency between the natural frequencies of those two modes, otherwise there will only be a minimum. From Eq 2.23 it can be seen that in a driving point function the modal

constant for every mode must be positive, Φ^2. Using this information and the fundamental rule it is concluded that for every resonance of a driving point function there must be anti-resonance without exception. The known existence of anti-resonance and how anti-resonances become resonances when H is inverted makes the driving point functions very important to the ABC application. The next chapter will illustrate how the anti-resonances found in the driving point function relates to the natural frequency of the structure with the same driving point DOF constrained to ground.

III. NATURAL FREQUENCIES OF ABC CONFIGURED SYSTEMS

As discussed in Chapter II a spatially incomplete FRF matrix obtained in a modal test contains the characteristics of the complete system and can be used to find the natural frequencies of a system whose ASET coordinates are pinned to ground. The pinning of the ASET is not a physical modification to the system but merely a mathematical manipulation of the measured data; therefore, the boundary conditions imposed on the ASET are artificial and as such are referred to as Artificial Boundary Conditions, ABC.

A. THE IMPORTANCE OF ABC FREQUENCIES IN MODEL UPDATING

It is essential to understand the importance ABC in model updating. Since the goal in updating a FE model is to have the greatest correlation of the dynamic behavior of the FE model and the actual structure, it is necessary to have an accurate comparison method of the systems. The best measure of correlation of dynamic behavior is how close the natural frequencies of the FE model are to the actual natural frequencies of the structure. As mentioned in the introduction FE models can be defined by a large number of adjustable parameters which can be referred to as design variables. Although an actual structure is composed of infinite number of DOF and thus has infinite number of modal parameters only a limited number of the structure's actual modal parameters, i.e. FRF, can be measured in the laboratory. The disparity in number of measured modal parameters versus the number of adjustable parameters defines an undetermined problem. Simply stated from an ordinary modal test there are far too many of the unknown variables and too few of the known variables for the solution to be accurate.

$$\begin{Bmatrix} \Delta\omega_1^2 \\ \vdots \\ \Delta\omega_k^2 \end{Bmatrix} = [T] \begin{Bmatrix} dV_1 \\ \vdots \\ dV_k \\ \vdots \\ dV_n \end{Bmatrix} \quad \text{where } k << n, \tag{3.1}$$

where $\Delta\omega^2 = \omega_x^2 - \omega_a^2$, ω is the modal parameter, ω_x = undamped natural frequency of the actual structure and ω_a = undamped natural frequency of the FE model, *dV* is the

adjustable parameters of the FE model and [T] is a matrix which relates the modal parameters to the adjustable parameters, referred to as a sensitivity matrix, and is discussed in great detail later (Gordis 1999).

As shown in Eq.(3.1) with $k<<n$ more modal parameters are needed for an accurate solution to be calculated. Since modal parameters are directly related to the boundary conditions of the structure, one method of measuring more modal parameters without introducing more adjustable parameters is to impose physical boundary conditions on the structure and measure new modal parameters. This method is time consuming, costly and frequently impossible. The method of Artificial Boundary Conditions is easy and can generate multiple sets of Artificial Boundaries from one set of measured data of a given structure reducing time and money. Therefore, the answer of "why study ABC?" is simple; it can generate more equations which lead to increased determinability of the system of equations and accuracy of error localization between the FE model and actual structure. For n ABC systems Eq. (3.1) would be

$$\begin{Bmatrix} \Delta\omega^2_{BASE} \\ \Delta\omega^2_{ABC1} \\ \vdots \\ \Delta\omega^2_{ABCn} \end{Bmatrix} = [T] \begin{Bmatrix} dV_1 \\ \vdots \\ dV_n \end{Bmatrix}$$

where $\Delta\omega^2_{BASE} = \begin{Bmatrix} \Delta\omega^2_1 \\ \vdots \\ \Delta\omega^2_k \end{Bmatrix}$ and $\Delta\omega^2_{ABC1} = \begin{Bmatrix} \Delta\omega^2_1 \\ \vdots \\ \Delta\omega^2_m \end{Bmatrix}$ $1<m<n$

Although the theory of ABC frequencies is mathematically sound only a few computer simulations have been run to evaluate its improvement of error localization. Since ABC could potentially play an important role in model updating a complete understanding of the subject is necessary. Chapter II only hinted at how ABC are imposed and how the additional OSET frequencies are found. The following examples will explain in detail the easy process of gathering OSET frequencies via ABC.

The first example shows the relationship between the driving point function and the natural frequencies of the same system with an ABC imposed on the driving point DOF.

B. ABC FREQUENCIES OF A GIVEN ASET ARE DEFINED BY THE CORRESPONDING DRIVING POINT ANTI-RESONANCES

It was proven in the previous chapter that although anti- resonances can exist in transfer function their existence is always guaranteed in driving point functions. In a graphical plot of a FRF matrix versus frequency the anti-resonances will be represented by negative peaks or valleys following each positive peak of the FRF as shown in a sample plot in Figure 3.

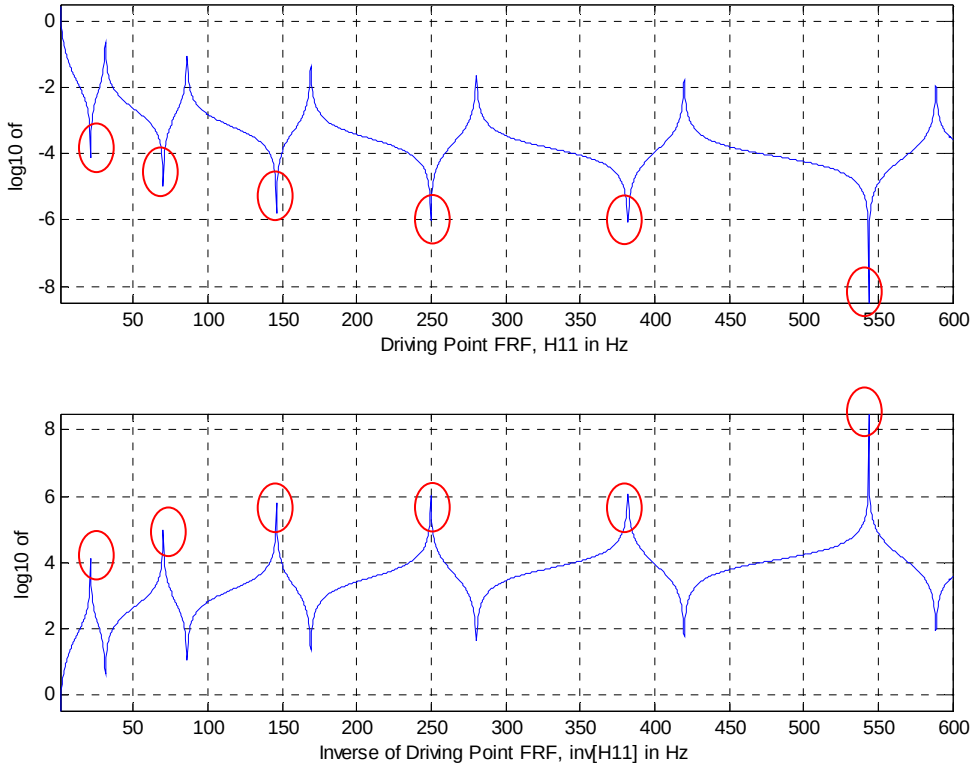

Figure 3. Plot of a Driving Point FRF,[H11] and inv [H11]

The anti-resonances of the system are marked with red circles in the top plot of Figure 3. It is easily seen in Figure 3 that the anti-resonance valleys of $[H_{11}]$ become the peaks of $[H_{11}]^{-1}$ which is plotted in bottom plot again marked with red circles. Recalling that the

elements of $\left[H_{aa}\right]^{-1}$ will be singular, at the frequencies which satisfy $\left[K_{oo}-\Omega^2 M_{oo}\right]=0$ and that singular elements are represented by peaks on a graphical plot of the function in question, the peaks of $\left[H_{aa}\right]^{-1}$ are always located at the natural frequencies of a system whose ASET coordinates are pinned.

To assist in understanding this conclusion a few example follow.

1. ABC Example using 2 DOF System

A simple 2 DOF system, masses (M1, M2) and springs (K1, K2) is shown in Figure 4 (Gordis 1999).

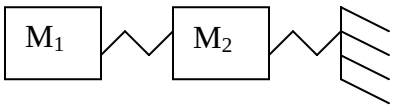

Figure 4. 2 DOF system

From Eq. 2.23, the undamped driving point FRF of any structure is given by

$$H_{ii}(\Omega)=\sum_{k=1}^{modes}\frac{(\Phi_i^k)^2}{\omega_k^2-\Omega^2} \tag{3.1}$$

where Φ_i is the mass normalized mode shape element, ω_k is the ***k***th natural frequency, and Ω is the forcing frequency. The frequency of the anti-resonance of $[H_{11}(\Omega)]$ is given by (Gordis 1999).

$$\Omega^2_{anti-res}=\frac{R_{11}^1\omega_2^2+R_{11}^2\omega_1^2}{R_{11}^1+R_{11}^2} \tag{3.2}$$

where the modal residue is $R_{ij}^{\ k}=\{\Phi(:,k)\}\{\Phi(:,k)\}$.

The matrices of 2 DOF system are

Stiffness matrix: $\left[K\right]=\begin{bmatrix}K_1 & -K_1\\ -K_1 & K_1+K_2\end{bmatrix}$

Mass Matrix: $\left[M\right]=\begin{bmatrix}M_1 & 0\\ 0 & M_2\end{bmatrix}$

Setting $M_1 = M_2 = 1.0$ and $K_1 = K_2 = 1.0$

Mode shapes: $\{\Phi^1\} = \begin{Bmatrix} -0.85065 \\ -0.52573 \end{Bmatrix}$ $\{\Phi^2\} \begin{Bmatrix} -0.52573 \\ 0.85065 \end{Bmatrix}$

Natural frequencies: $\{\omega(rad/\sec)\} = \begin{Bmatrix} 0.618 \\ 1.618 \end{Bmatrix}$

Using the above modal parameters and Eq (3.2), the resulting in a single anti-resonance is located at the frequency $\Omega_{\text{anti-res}} = \sqrt{2}$ rad/sec. The driving point FRF, [H11] is shown in Figure 5. The single anti-resonance is noticeable at $\sqrt{2}$ rad/sec.

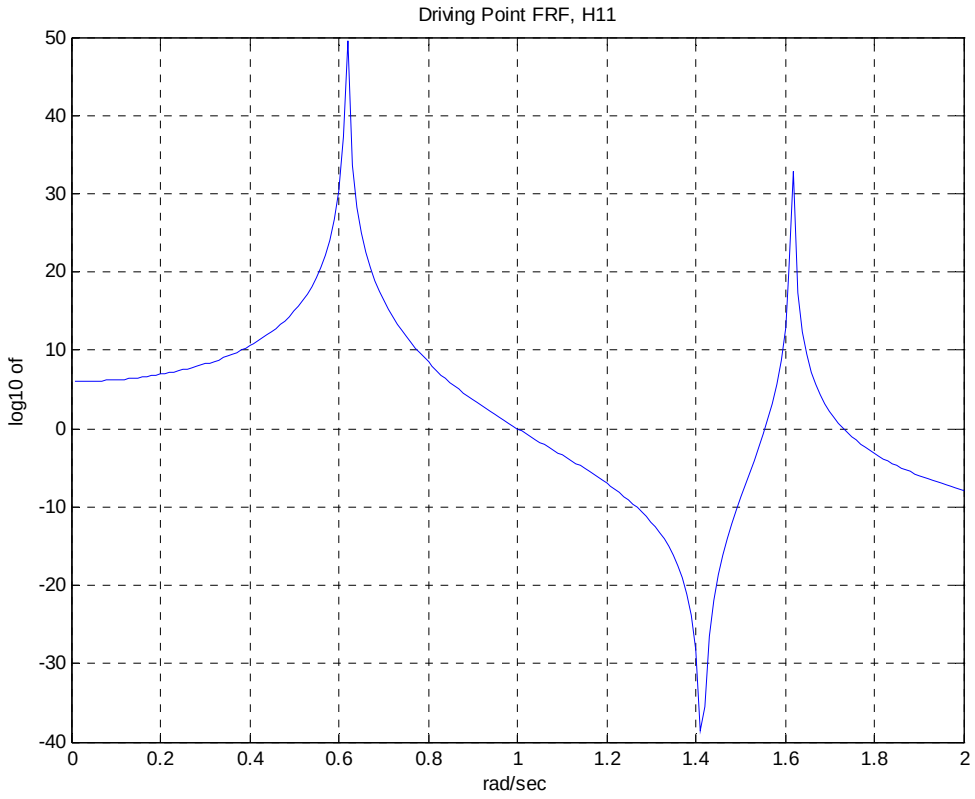

Figure 5. Driving Point FRF, H_{11}

Given that for this system $[H_{aa}] = [H_{11}]$, i.e. ASET = [1], calculations were done for the natural frequency of system in Figure 6, which is the same system as the one in Figure 4 but with ASET coordinate, DOF 1, pinned.

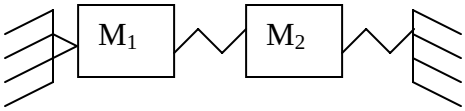

Figure 6. ASET, DOF #1 constrained to ground.

The single natural frequency of the system in Figure 6 is $\sqrt{2}$ rad/sec which is identically equal to anti-resonance frequency calculated for 2 DOF system in Figure 4.

$[H_{aa}\ (\Omega)]^{-1} = [H_{11}\ (\Omega)]^{-1}$ corresponding to system 2 and ASET = [1] was calculated using Eq (2.16) and plotted at each frequency. The impedance Z_{11} of system 2 is plotted the plot of driving point FRF of system 1 to show clearly that the anti-resonance of system 1 (Plot A) is equal to the singular frequency of system 2 (Plot B).

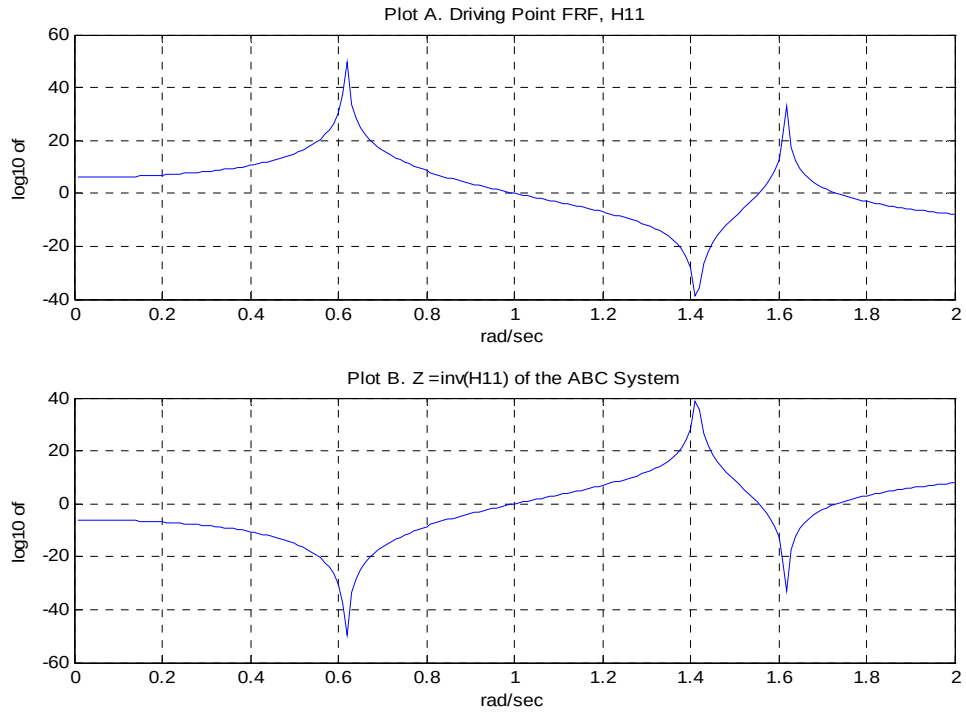

Figure 7. Plot A. Driving Point H11(Ω) of system 1 Plot B. $[H_{aa}(\Omega)]^{-1}$ of system 2

This example demonstrates nicely the relationship between the anti-resonance of the driving point FRF and natural frequency of system with the driving point DOF. Since confusion is possible with the use of only 2 DOF another example will be calculated using a multi-DOF system.

2. ABC Example using Multi - DOF System, Single Coordinate ASET.

A simple cantilever beam shown in Figure 8 is simply supported at one end. The beam has ten 2 node beam elements with a total of 20 DOF.

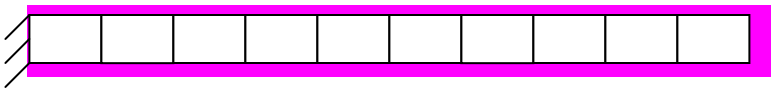

Figure 8. 10 Element Cantilever beam

The driving point function for the cantilever was calculated using Eq 2.23 in the same fashion as example 1. The natural frequencies of this system under 800 Hz are 4.9186, 30.826, 86.332, 169.29, 280.29, 419.91, 589.15, 789.3 and the respective anti-resonances calculated using Eq. (3.2) are 6.8697, 43.856, 124.28, 245.53, 406.42, 586.39, 704.69. The driving point FRF of DOF 3, i.e. $[H_{33}]$ was calculated using Eq (3.1) and plotted versus frequency. With ASET = 3, an ABC was applied to DOF 3, shown in Figure 9.

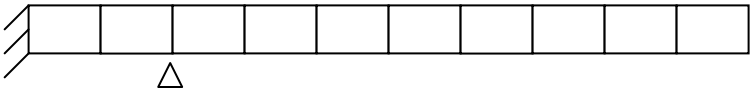

Figure 9. 10 Element Cantilever beam, DOF 3 Artificially pinned

and the natural frequencies were calculated using the reduced order [K], [M] and compared with the graphical plot of $[H_{33}]^{-1}$ which was calculated using Eq (2.16). The natural frequencies of the ABC system, DOF 3 pinned, under 800 Hz are 6.8697, 43.856, 124.28, 245.53, 406.42, 586.39, 704.69. There compare nicely with peaks of $[H_{33}]^{-1}$. The same as example the plots of $[H_{33}]$ and $[H_{33}]^{-1}$ are combined on Figure 10 for easier comparison of the location of peaks and anti-resonances.

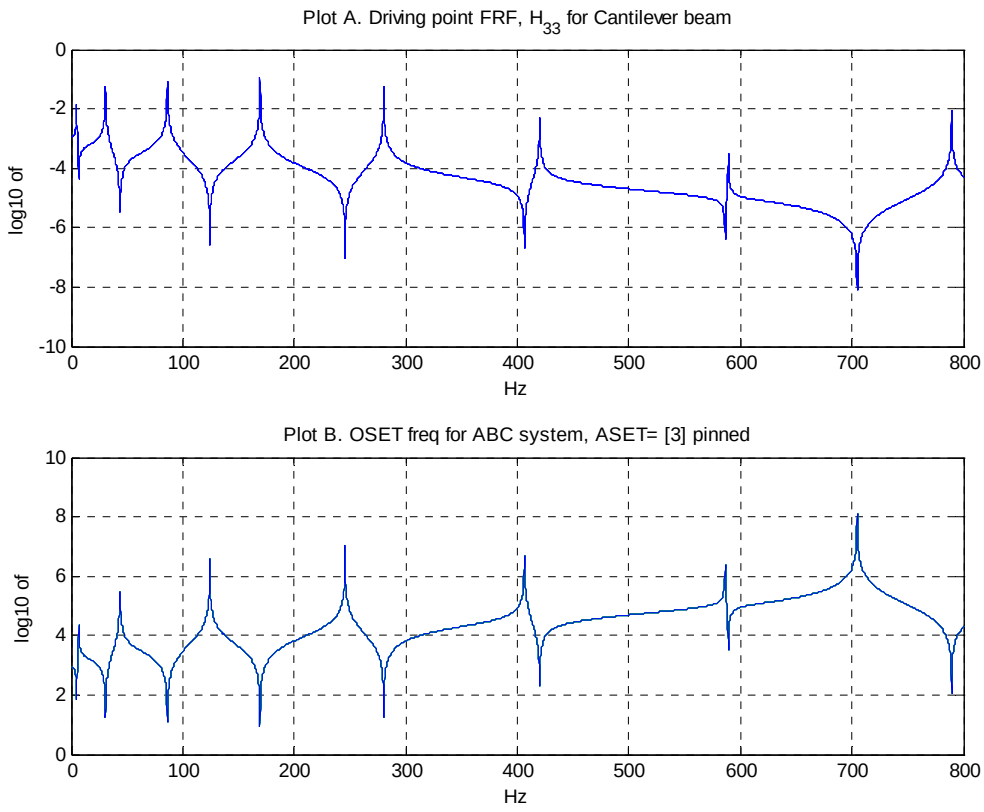

Figure 10. Plot A. Driving Point H_{33} (Ω) of cantilever beam Plot B. $[H_{33}(\Omega)]^{-1}$ of ABC system, DOF 3 Artificially pinned.

The relationship between the anti-resonance of the driving point FRF and the natural frequencies of the ABC system is again visible in the plots of H_{33} (Ω) and $[H_{33}(\Omega)]^{-1.}$ This example takes away any confusion about the frequency relationship and gives way to an example of a multiple coordinates ASET.

3. ABC Example using Multi - DOF System, Multiple Coordinate ASET

The same simply supported cantilever in Figure 8 is used for this example. However, five accelerometers have been added to the beam at translational DOF # 1,7,11,15,19, therefore the ASET = [1,7,11,15,19].

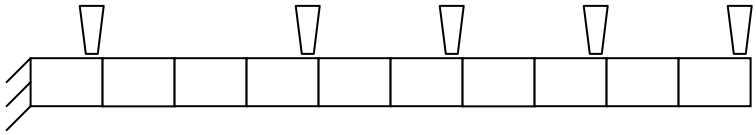

Figure 11. 10 Element cantilever beam, Accelerometers located at DOF # 1,7,11,15,19

If this was an actual experiment and each accelerometer was excited the measured FRF would be a spatially incomplete FRF matrix of 5x5 due to the length of ASET vector. In a MATLAB simulation of such an experiment the spatially incomplete FRF is calculated using the Eq (2.16) and by building a complete FRF matrix and retaining the columns and rows corresponding to ASET coordinates. The results of both [H] are identical and plotted versus frequency on arrange of 0-800Hz in Figure 12.

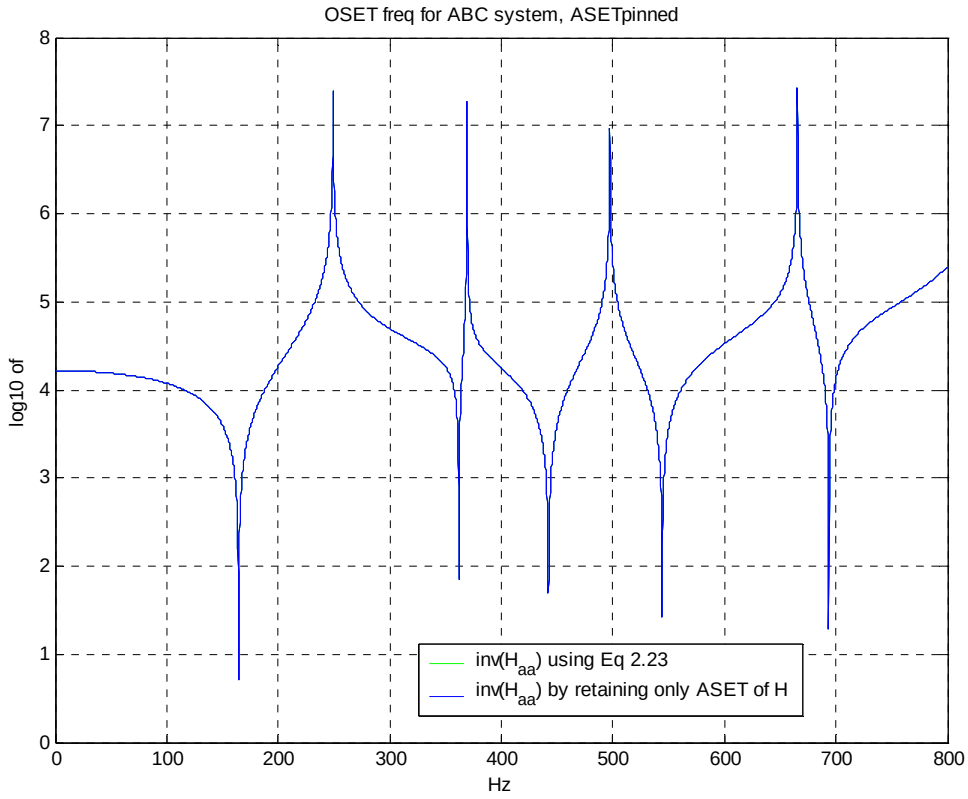

Figure 12. $[H_{aa}(\Omega)]^{-1}$ of 10 element cantilever beam with ASET = [1,7,11,15,19]

The peaks of $[H_{aa}]$ correspond to the natural frequencies of the system (249.17, 369.26, 497.32, 665.56 Hz) which were calculated from the reduced system [K] and [M]. The term "reduced" is used to indicate that the rows and columns corresponding to the ASET were removed from the respective matrices prior to the calculation of natural frequencies, since natural frequencies are calculated using only unrestrained DOF and the ASET DOF are fully restrained.

Although two methods were used in calculating $[H_{aa}(\Omega)]^{-1}$ Figure 12 only shows one function thus confirming that both methods are correct. Since in modal testing only FRF matrices are measured and the stiffness and mass matrices of the system are unknown it is impractical to use an equation which requires the knowledge of [K] and [M] to determine OSET frequencies. Therefore, the proof that retaining the ASET rows and columns from a given FRF matrix provides the same accurate information as the Eq (2.16) is very important information. In the next chapter, examples are presented to demonstrate the use of ABC frequencies in locating errors between the FE model and actual structure. The method for finding ABC frequencies will be that of retaining the ASET coordinates from a base FRF matrix and not by Eq. (2.16).

C. MULTIPLE ABC SYSTEMS AVAILABLE

All examples demonstrated so far have applied ABC to the complete ASET thus retaining all rows and columns of the measured FRF matrix implying that only one set of ABC frequencies can be obtained from a spatially incomplete FRF matrix measured in the laboratory. This is not true. In ABC application the term ASET evolves from just an analysis set to a set of potentially bounded coordinates. From a measured 6 x 6 FRF matrix there are over 36 different combinations of applicable artificial bounded coordinate sets, i.e. only one coordinate pinned or sets of pairs and so forth. The only limitation is the size of the original data set.

D. OBTAINING OSET FREQUENICES GRAPHICALLY

Since OSET frequencies are obtained from the graphical representation of $[H_{aa}(\Omega)]^{-1}$ a method of curve fitting is used to approximate the frequency of the peaks. Curve fitting techniques are used to extract frequency data from FRF plots and are not valid for impedance plots however because of the behavior of $[H_{aa}(\Omega)]^{-1}$ near the peak is similar to that of a FRF plot curving fitting techniques are valid for obtaining OSET frequencies from $[H_{aa}(\Omega)]^{-1}$ plots.

1. Theory of Curve Fitting

In order to understand the validity of the usage of curve fitting to find the OSET frequencies the following theory is provided.

The goal of data or curve fitting is to find a mathematical model by which a set of empirical data can be accurately described. These models depend on adjustable parameters. With the correct model and corresponding equations, one can determine what parameter values correspond to the data. In order to select an accurate curve fit model a good understanding of the underlying physics or properties of the system to be curve fit is needed. Once a model is picked, a rough assessment can be made by plotting it with the data. At least some agreement is needed between the data and the model curve before continuing.

To find the values of the model's parameters that yield the curve closest to the data points, a function that measures the closeness between the data and the model must be defined. This function depends on the method used to do the fitting, and the function is typically chosen as the sum of the squares of the vertical deviations from each data point to the curve. (The deviations are first squared and then added up to avoid cancellations between positive and negative values.) This approach is called the method of least squares. This method assumes that the measurement errors are independent and normally distributed with constant standard deviation. Once the correct function is found it is minimized to the smallest possible value. The parameters valves that minimize the function are the best-fitting parameters. In most engineering models the dependent variable depends on the parameters in a nonlinear way.

Nonlinear fitting usually can not use the system equation to solve for the minimizing parameters instead various iterative procedures are used. Nonlinear fitting always starts with an initial guess of the parameters values. User usually looks at the general shape of the model curve and compares it with the shapes of the data points. A good understanding of the selected model is important because the initial guess of the parameter values must make sense in the real world.

In the process of interpreting results the user must see whether the program can fit the model to the data that is whether the iteration converged to a set of values for the parameters. A look at the graphed data points and fitted curve can show of the curve is close to the data. If it is not, then the fit has failed, perhaps because the iterative

procedure did not minimize the parameters or the wrong model was chosen. If the model yields a physically meaningless result then the curve is wrong regardless if it seemingly does fit the data well. (Ledvij 2003)

2. Amplitude Fitting

After understanding the basic theory behind curve fitting this section will describe the specifics of the curve fitting performed in this experiment.

Rinawi and Clough developed a new non-iterative least squares method that weighs all the data points of a transfer function more uniformly and is more reliable and is easily programmed. Their method was applied successfully to identify the frequency and damping of a test structure during seismic simulation tests.

The procedure for this method which was used to identify the OSET frequencies from $[H_{aa}(\Omega)]^{-1}$ plots. Since the peak of $[H_{aa}(\Omega)]^{-1}$ was well separated it can be approximated by a single mode response as (Rinawi & Clough):

$$\ddot{y} + 2\omega_n \xi_n \dot{y} + \omega_n^2 y = P_n e^{i\Omega t} \tag{3.3}$$

In which ω_n, ξ_n are the structural frequencies for a particular mode. P_n is the participation factor for the mode. It is for this reason that the MATLAB program written for the identification of OSET frequencies evaluated only a small range of frequencies near the resonance peak. When the range was too large the OSET frequencies calculated had too great a variance to be considered accurate. Emphasis was placed on finding the most best usable range.

At a given input frequency Ω_k, the steady state amplitude of the response *y* is given by: $A_k = \dfrac{P_n}{\sqrt{(\omega_n^2 - \Omega_k^2)^2 + (2\omega_n \xi_n \Omega_k)^2}} = \dfrac{P_n}{D_k}$ (3.4)

The unknown parameters in the above equation are ω_n, ξ_n and P_n. It is importmant to note that the above equation is valid for a transfer function relating the input force to the output displacement otherwise the amplitude needs to be scaled by the appropriate power of Ω_k to transform the equation into the above form.

Eq (3.4) can be written as: (Notice the scale factor)

$$A_k^3 D_k^2 - A_k P_n^2 = 0 \tag{3.5}$$

Substituting D_k of Eq (3.5) results in

$$A_k^3 x_1 + A_k^3 \Omega_k^2 x_2 - A_k x_3 = -A_k^3 \Omega_k^4$$

where

$$\begin{aligned} x_1 &= \omega_n^4 \\ x_2 &= 4\xi_n^2 \omega_n^2 - 2\omega_n^2 \\ x_3 &= P_n^2 \end{aligned} \tag{3.6}$$

When this equation is solved over a range of frequencies Ω_k using least squares solution resulting the following equations used to compute modal parameters of the data set (Rinawi & Clough).

$$\begin{aligned} \omega_n &= (x_1)^{1/4} \\ \xi_n &= \sqrt{\frac{x_2}{4\omega_n^2} + \frac{1}{2}} \\ P_n &= \sqrt{x_3} \end{aligned} \tag{3.7}$$

During programming it was noticed the correct frequency was identified without identifying the correct damping ratio. Since frequencies were more important in this experiment the issue of incorrect damping ratio was not explored.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SENSITIVITY–BASED UPDATING WITH ARTIFICAL BOUNDARY CONDITIONS

A. SENSITIVITY MATRIX DEFINED

In addition to the problem of not being able to measure enough parameters to have a determined system of equations needed to properly update a FE model, there exists a problem with determining exactly what parameters are in error and in need of adjustment. Sensitivity-based updating is used to localize those parameters that require adjustment.

Simply stated the equation for sensitivity - based updating is

$$\left\{\Delta\omega^2\right\} = [T]\{\Delta DV\} \tag{4.1}$$

where $\left\{\Delta\omega^2\right\}$ is a vector of eigenvalues λ, ($\lambda = \omega^2$) errors. The errors are the difference between the experimental eigenvalues and the analytical eigenvalues $\left\{\omega_x{}^2 - \omega_a{}^2\right\}$, "*x*" represents the experimental data, "*a*" represents the analytical data. {DV} is the vector of design parameters, which are adjustable for the FE model. The vector represents location and quantity of change. [T] is the sensitivity matrix. The following section mathematically derives the sensitivity matrix used in this thesis.

B. SENSITIVITY MATRIX MATHEMATICALLY DERIVED

The sensitivity analysis used in this thesis is based on parametrizing the eigenvalue problem.

Consider a conservative *n* DOF system defined by

$$M(DV)\ddot{x}(t) + K(DV)x(t) = 0 \tag{4.2}$$

and the related eigenvalue problem

$$[K - \lambda_i M]\{\Phi_i\} = \{0\} \tag{4.3}$$

The design parameter *DV* represents the change in mass matrix [***M***] and/or the stiffness matrix [***K***]. It is shown that [***M***] and [***K***] are dependent on the parameter as are the eigenvalue λ_i , modal frequency squares and eigenvalue Φ_i, mode shape.

The derivation of the sensitivity matrices began with the differentiation of eigenvalue problem with respect to parameter *DV*,

$$\left[\frac{dK}{dDV}-\lambda_i\frac{dM}{dDV}-\frac{d\lambda_i}{dDV}M\right]\{\Phi_i\}+\left[K-\lambda_i M\right]\left\{\frac{d\Phi_i}{dDV}\right\}=\{0\} \tag{4.4}$$

Expand each term and premultiply by $\{\Phi_i\}^T$, for Eq (4.5)

$$\{\Phi_i\}^T\left[\frac{dK}{dDV}\right]\{\Phi_i\}-\lambda_i\{\Phi_i\}^T\left[\frac{dM}{dDV}\right]\{\Phi_i\}-\frac{d\lambda_i}{dDV}\{\Phi_i\}^T[M]\{\Phi_i\}+\{\Phi_i\}^T\left[K-\lambda_i M\right]\left\{\frac{d\Phi_i}{dDV}\right\}=\{0\}$$

Using the property $\{a\}^T[b]\{c\}=\{c\}^T[b]\{a\}$ the last element of Eq (4.5) can be written as

$$\left\{\frac{d\Phi_i}{dDV}\right\}^T\left[K-\lambda_i M\right]\{\Phi_i\} \tag{4.6}$$

Since $\left[K-\lambda_i M\right]\{\Phi_i\}=\{0\}$, the overall equation is reduced

$$\{\Phi_i\}^T\left[\frac{dK}{dDV}\right]\{\Phi_i\}-\lambda_i\{\Phi_i\}^T\left[\frac{dM}{dDV}\right]\{\Phi_i\}-\frac{d\lambda_i}{dDV}\{\Phi_i\}^T[M]\{\Phi_i\}=\{0\} \tag{4.7}$$

Using orthogonality, $[\Phi]^T[M][\Phi]$ = 1, Eq (4.7) is reduced

$$\{\Phi_i\}^T\left[\frac{dK}{dDV}\right]\{\Phi_i\}-\lambda_i\{\Phi_i\}^T\left[\frac{dM}{dDV}\right]\{\Phi_i\}-\frac{d\lambda_i}{dDV}=\{0\} \tag{4.8}$$

The terms of Eq(4.8) are recombined for the following,

$$\frac{d\lambda_i}{dDV}=\{\Phi_i\}^T\left[\frac{dK}{dDV}-\lambda_i\frac{dM}{dDV}\right]\{\Phi_i\} \tag{4.9}$$

Rearranging Eq 4.9 to solve for *dDV* in terms of known parameters, [K], [M], and λ

$$dDV=\frac{d\lambda_i}{\{\Phi_i\}^T\left[\frac{dK}{dDV}-\lambda_i\frac{dM}{dDV}\right]\{\Phi_i\}} \tag{4.10}$$

From Eq (4.10) it can be deduced that the associated sensitivities for [K] and [M] are as follows:

$$\text{Stiffness sensitivity} = \{\Phi_i\}^T \left[\frac{dK}{dDV} \right] \{\Phi_i\} \tag{4.11}$$

where $[dK] = [K_x - K_a]$ (4.11a)

and

$$\text{Mass sensitivity} = \{\Phi_i\}^T \left[-\lambda_i \frac{dM}{dDV} \right] \{\Phi_i\} \tag{4.12}$$

where $[dM] = [M_x - M_a]$ (4.12a)

C. SENSITIVITY MATRIX USED IN ERROR PREDICTIONS OF A SIMPLE CANTILEVER

A sensitivity matrix can be described in words as how one element is affected by a small change in another element. As part of validating the usage of ABC configured systems in FE model updating a MATLAB simulation was conducted using a cantilever beam. The beam represented the physical and material properties of the actual beam used in the experiment application which will be discussed in the following section. The beam was 42 inches in length, 1.5 inches wide and 0.5 inches in thick, density of 0.11 lbf/in^3 and elasticity modulus of 10 E6 lbf/sec^2-in and only 10 elements, yielding a total of 20 DOF after original boundary conditions were applied. The quantity of 10 elements was chosen for ease of comparison between an underdetermined system and a fully determined system of equations.

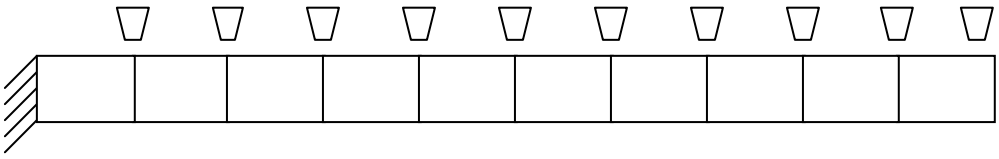

Figure 13. FE model of cantilever beam with Accelerometers at DOF [1,3,5,7,9,11,13,15,17,19]

A series of computer programs were written to perform the MATLAB simulation. All programs are provided Appendix B. In the simulation, two beams are created: Beam A (represents Analysis or FE Beam), Beam X (represents experimental beam). A known mass or EI is applied to Beam X on a specific element. Although the quantity and location of the "error" was known it was assumed not to be, thus a sensitivity matrix was developed to account for any error location.

Recalling Eq (4.12, 4.12a), a small change or perturbation of 1% mass was applied to each element one element at a time of the [M] without the error added. The perturbated [M] was considered [M_x], while FE model [M] was considered baseline or [M_a] for the purpose of calculating $\left[dM\right]=\left[M_x - M_a\right]$. Once [*dM*] was calculated the corresponding column of sensitivity matrix, [T] was calculated using Eq. (4.12). Each subsequent column of [T] was calculated using the same perturbation quantity but on a new corresponding element. The stiffness sensitivity matrix was calculated using the same procedure except for applying perturbation of 1% of EI to the stiffness matrix and using Eq. (4.11). For full details on the development of sensitivity matrix refer to BeamSensitivity_crs.m in Appendix B.

Once both of the sensitivity matrices [T(M)] & [T(K)] were developed of size *m* x *p* , where m is the number of modes and p is the number of elements, a complete [T] was assembled, size m x 2p, yielding changes in mass in the first 10 elements of {DV} and changes in EI in the last 10 elements of {DV} for the baseline configuration.

For each ABC system a new set of rows are created in $\left\{\Delta\omega^2\right\}$ and $\left[T\right]$ thus obtaining more set of equations in

$$\begin{Bmatrix} \Delta\omega^2_{BASE} \\ \Delta\omega^2_{ABC1} \\ \vdots \\ \Delta\omega^2_{ABCn} \end{Bmatrix} = \begin{pmatrix} T_{BASE} \\ T_{ABC1} \\ \vdots \\ T_{ABCn} \end{pmatrix} \begin{Bmatrix} dV_1 \\ \vdots \\ dV_n \end{Bmatrix} \tag{4.13}$$

It is desired to have a fully determined set of equations for the best prediction of error quantity and location. In the MATLAB simulation, DOF [1,3,5,7,9,11,13,15,17,19] were considered equipped with accelerometers and thus the only DOF available for

pinning. For a given error the simulation applies a pin at each accelerometer location, one at a time, and calculates the new natural frequencies and respective [T], resulting in 10 ABC configured $\left\{\Delta\omega_{ABC}{}^2\right\}$ and $\left[T_{ABC}\right]$ for each type and location of error.

A comparative analysis was conducted on the accuracy of error prediction with respect to error location and ABC used. The simulation compared mode shape differences or relative frequency errors between Beam A and Beam X, Norm of the columns and rows of the sensitivity matrix used in error prediction, the rank of [T] and figure of merit based on relative sum error. However, only the figure of merit gave any pseudo relationship between error location, ABC system used, and the accuracy of the error prediction.

The formula used for figure of merit (FOM) is

$$FOM = \frac{\left(\frac{DV_{cal}^{n}}{error}\right)^2}{\sum_{i=1}^{numelem}\left(\frac{\left\{DV_{cal}^{i}\right\}}{error}\right)^2} \tag{4.14}$$

where *error* is the actual % error added to beam, $\left\{\frac{DV_{cal}^{i}}{error}\right\}$ is the normalized predicted design variable vector of whole beam, $\left(\frac{DV_{cal}^{n}}{error}\right)$ is the normalized predicted design variable at actual location of error. All elements are squared to compensate for leakage defined as prediction of error on other elements and in order for the sum of all elemental predictions to be equal to the actual error. It was noticed that if the elemental predictions were simply summed the negative leakage canceld out the positive leakage yielding false accurancy of the overall prediction. Although, this formula was not as accurate as desired but it was sufficient in reducing the large quantity of calculations in order for the most accurate predictions to be further studied.

Even though twenty plots, each with 6 distinct scenarios, were generated to demonstrate the accuracy of error prediction with respect to error location and ABC system used, only four are shown in the following examples. The results of all 120

scenarios are located in Appendix A in the form of 6 FOM tables. The tables show all ABC configurations and the modes used in error prediction versus the type and location of error added to Beam X. The best representations of trends are exhibited by applying each type of change, mass or EI, to each end of the beam separately.

1. Error Prediction: Example 1

This example is of 10 % mass change applied to element 2 at the root end of the beam, node 11 pinned, shown in Figure 14.

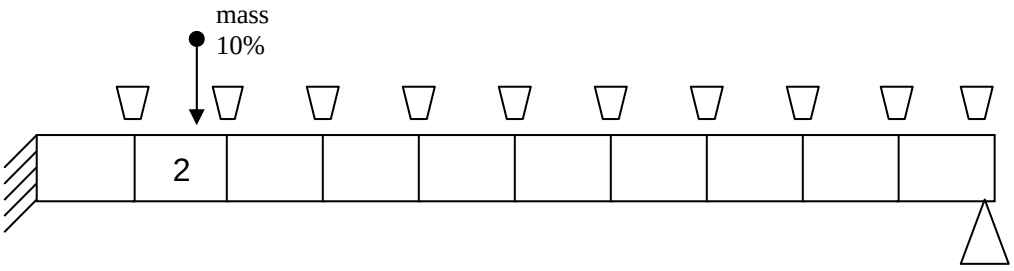

Figure 14. Diagram of Cantilever beam 10% change in mass applied to element 2, Node 11 pinned

In order to correctly read the next two bar graphs the following information is given. The yellow circle indicates the actual mass error in location and magnitude. The blue bars indicate the magnitude of the error for each element predicted using only the first 5 modes of the base system in development of the [T].

The graphs on the left represent the development of $\{\Delta\omega^2_{ABC}\}$ and $[T_{ABC}]$ using only 5 modes of the ABC system, shown is ABC 10, node 11 pinned. Thus the system of equations $\{\Delta\omega^2\} = [T]\{\Delta DV\}$ is underdetermined. The plots on the right represent the development of $\{\Delta\omega^2_{ABC}\}$ and $[T_{ABC}]$ using the first 5 modes of the BASE system, in addition to 5 modes of the ABC system. The top row of graphs represents the use of modes [1:5] of the ABC system; middle row, modes [6:10] and bottom row, modes [11:15]. For each subplot the condition number of [T] used and FOM of prediction are labeled.

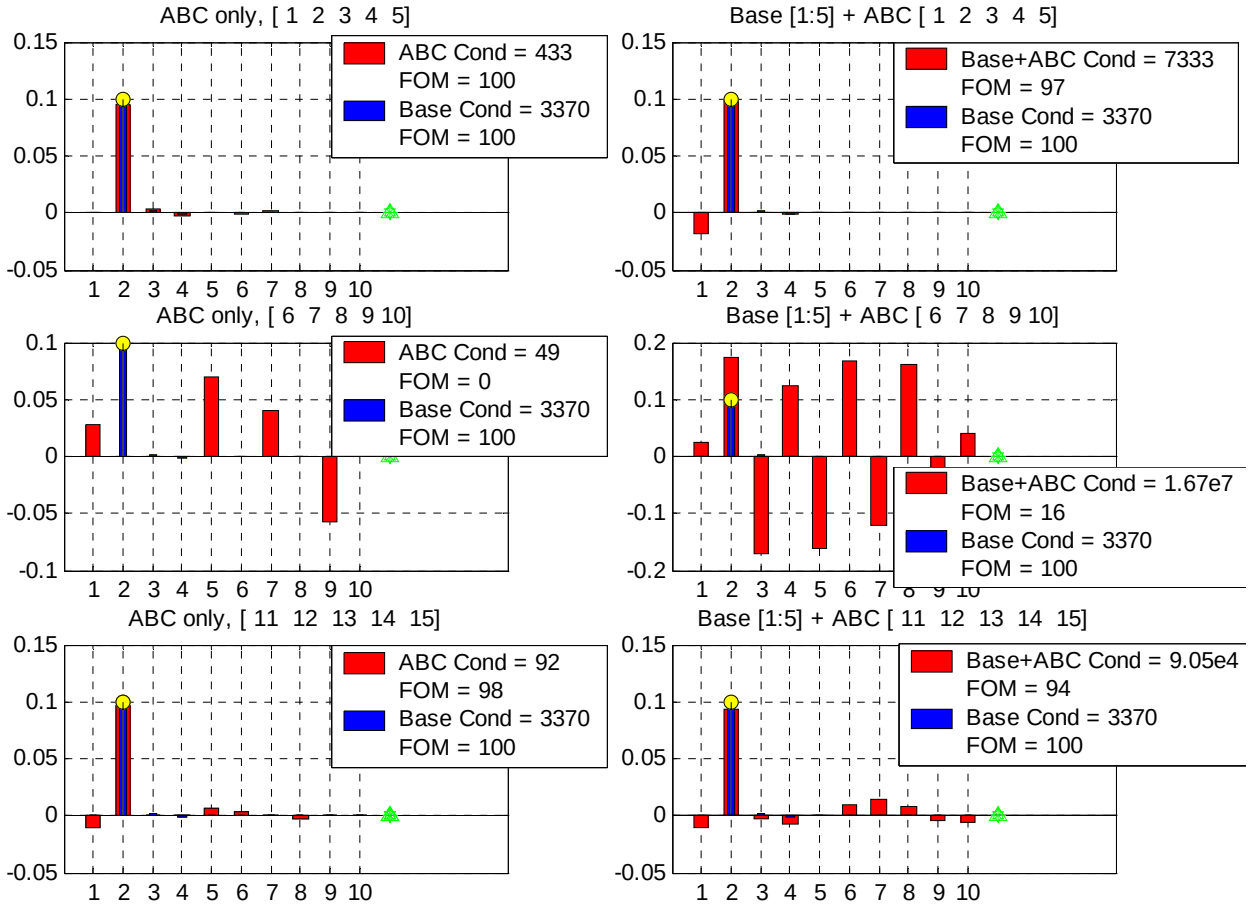

Figure 15. Plots of Error Prediction 10% change in mass applied to element 2, Node 11 pinned

In this case the error prediction using the first 5 modes of the baseline system provided a highly accurate result, FOM 100 with a [T] condition of 3370. Another highly accurate error prediction with a FOM of 100 was given by using the first 5 modes of the ABC system, node 11 pinned. However, the condition of this ABC system [T] was 433, suggesting nothing of a relationship between Cond (T) and accuracy of error prediction.

It was stated previously that the FOM formula was not completely accurate or sufficient. It was merely a method of reducing a large collection of data to a better handle collection for further study. This insufficiency is seen in a comparison between the error predictions using only the second 5 modes of ABC system, modes: 6-10 where the size of [T] is 5 by 10 and using the same 5 modes of the ABC system in combination with the

first 5 modes of the baseline system, where [T] is 10 by 10. The underdetermined of ABC modes alone has a FOM of 0 due to the fact that there is no error prediction on the correct element. The determined ABC + Base system has a FOM of 16 due to the fact that there is a small error prediction on the correct element. The FOM of these two system suggests that the ABC + Base system is the better error prediction even though the error prediction for the rest of beam is far worst than the beam error prediction of the underdetermined ABC only system. However, the FOMs a good job of precluding the respective system set-ups from further study, seeing that neither system is accurate enough to validiate further study.

2. Error Prediction: Example 2

This example is of 10 % mass change applied to element 8 at the free end of the beam, node 11 pinned, shown in Figure 16.

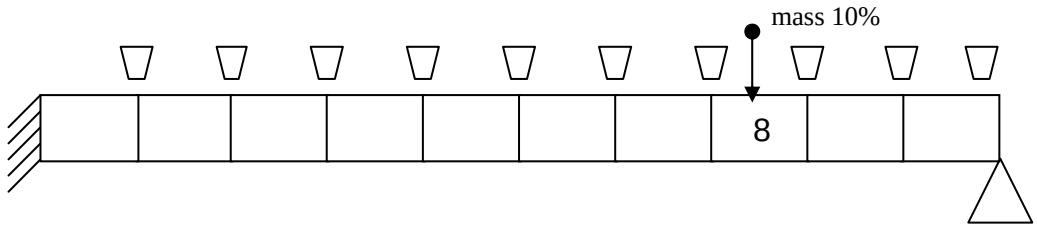

Figure 16. Diagram of Cantilever beam 10% change in mass applied to element 8, Node 11 pinned

The following error prediction graphs are displayed in the same fashion as those in Figure 15. The disparities between the graphs are based solely on the location of the error.

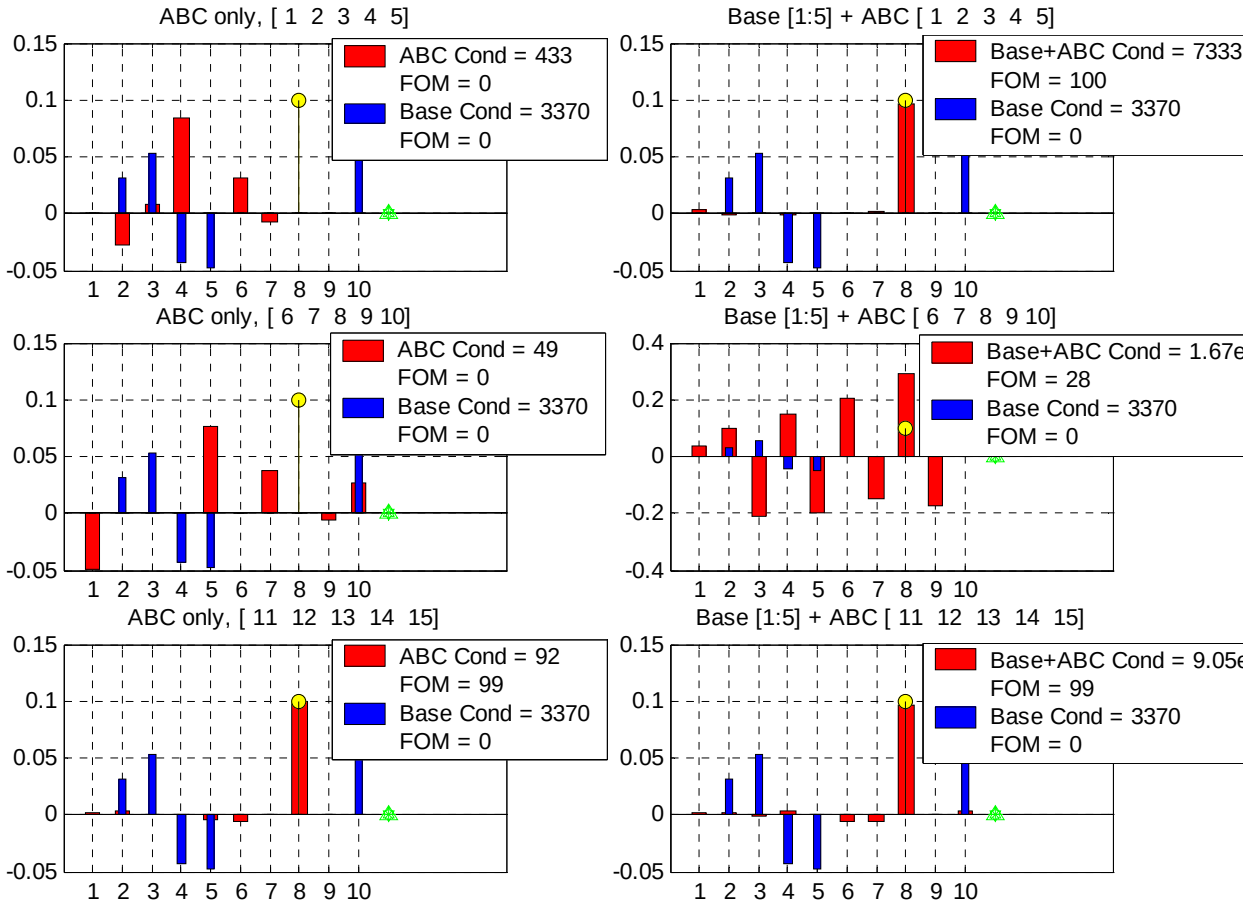

Figure 17. Error Prediction Plots for a 10% change in mass applied to element 8, Node 11 pinned

In this case the error prediction using the first 5 modes of the baseline system provided a highly inaccurate result, FOM 0 even though the condition [T] remained the same at 3370. The most accurate prediction was the system of ABC modes (1:5) + Base modes (1:5). This system had a FOM of 100 and a Cond ([T]) of 7333. The condition numbers of ABC (11:15) system, Cond(T) of 92 and ABC(11:15) +Base (1:5) system, Cond(T) = 9.05e5, suggest that a relationship between accuracy and condition of [T] does not exist.

3. Error Prediction: Example 3

This example is of 10 % EI change applied to element 2 at the root end of the beam, node 11 pinned, shown in Figure 18.

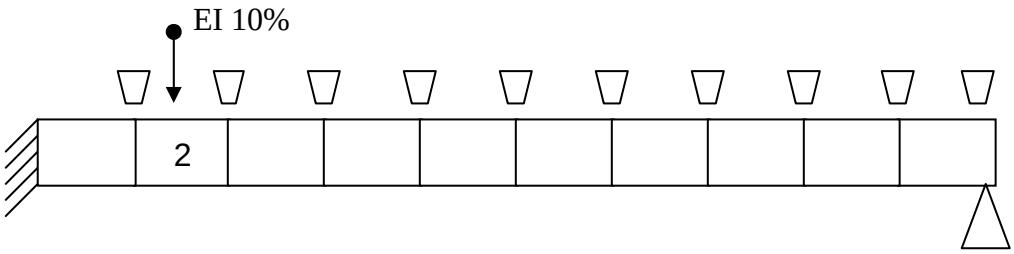

Figure 18. Diagram of Cantilever beam 10% change in EI applied to element 2, Node 11 pinned

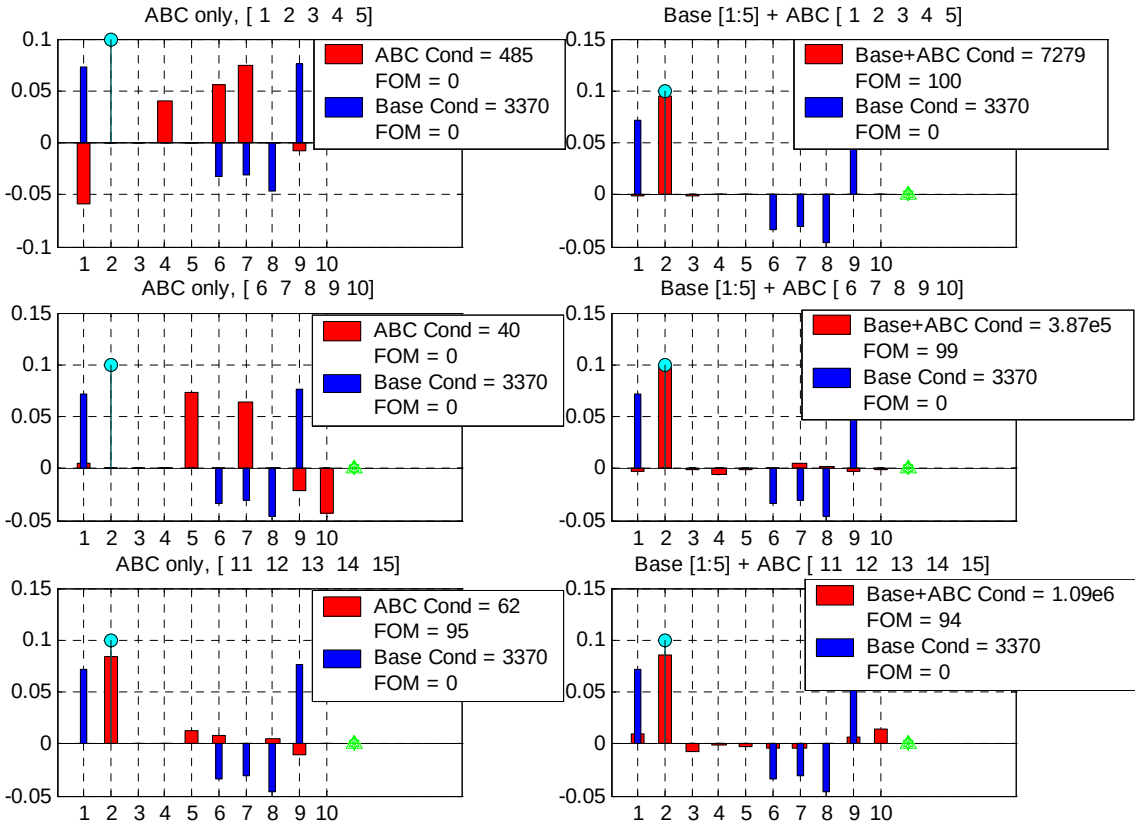

Figure 19. Error Prediction Plots for a 10% change in EI applied to element 2, Node 11 pinned

Similar to Example 2, in which a 10 % mass change added to an element at the free end of the beam, this situation yielded an highly inaccurate error prediction using the first 5 modes of the baseline system, FOM 0 even though the condition [T] remained the same at 3370. Also as in example 2, the most accurate prediction of this situation was the system of ABC modes (1:5) + Base modes (1:5). This system had a FOM of 100 and a Cond ([T]) of 7279. However, unlike the example 2, the current ABC (6:10) +Base (1:5) system, has an accurate error prediction with a FOM of 99.

4. Error Prediction: Example 4

This example is of 10 % EI change applied to element 8 at the free end of the beam, node 11 pinned, shown in Figure 20.

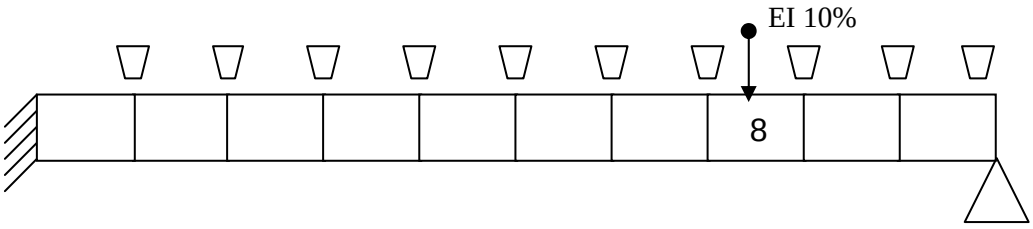

Figure 20. Diagram of Cantilever beam 10% change in EI applied to element 8, Node11 pinned

In this last example a similar trend is visible. The Base (1:5) system yields an accurate error prediction similar to that of example 1. However, ABC (1:5) + Base (1:5) and ABC (11:15) + Base (1:5) and ABC (11:15) systems all have FOM of 100 and prove to predict accurate error location and quantity without great leakage of error onto adjacent elements. Even ABC (6:10) + Base (1:5) system has a FOM and has only small leakage of error onto other beam elements.

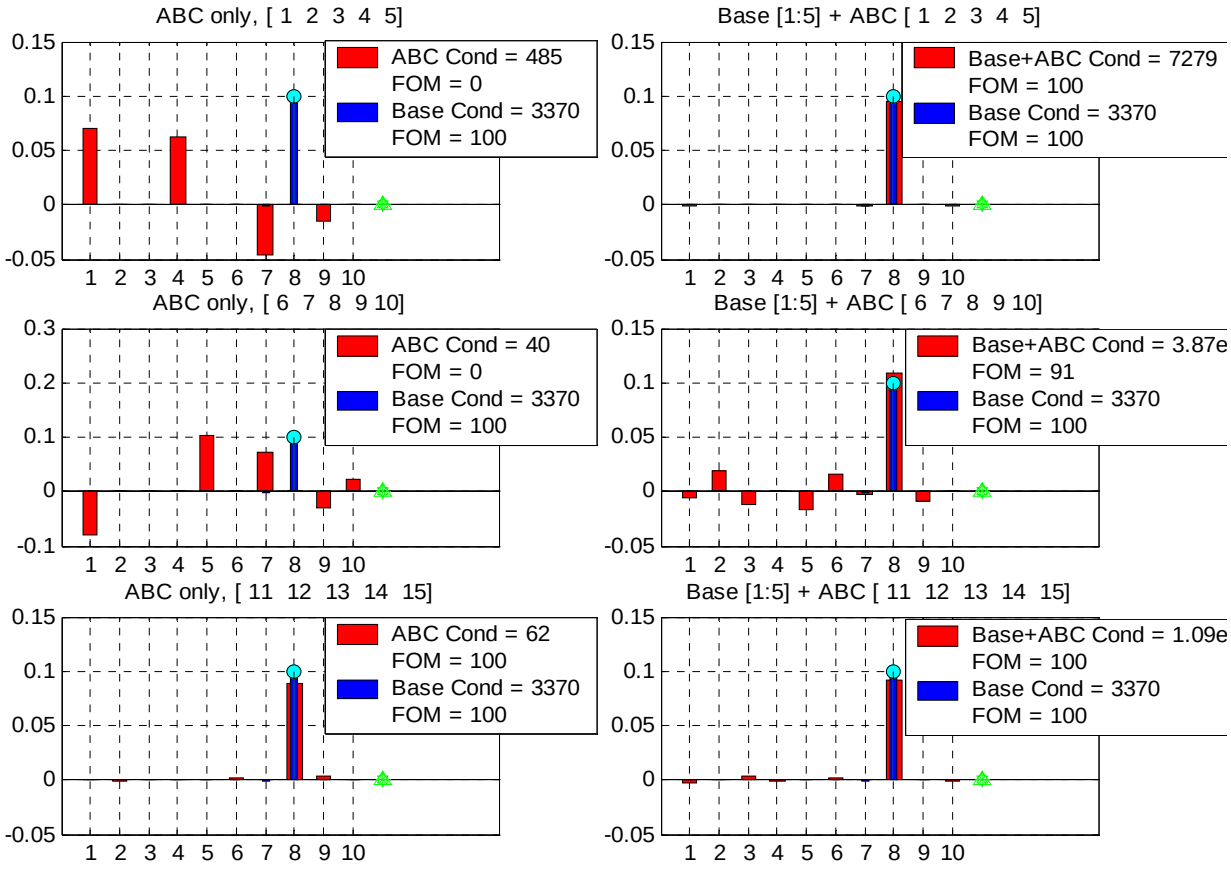

Figure 21. Error Prediction Plots for a 10% change in EI applied to element 8, Node11 pinned

V. EXPERIMENTAL APPLICATION

Previous section discussed in length the role of ABCs in error prediction and localization using a MATLAB simulation of a cantilever. This section explores the accuracy of OSET frequencies when ABCs are applied to experimentally measured spatially incomplete data. For comparison of data a FE model was created with the same dimensional and material properties of the cantilever beam used in the experiment. The FE model had 42 elements and 43 nodes for a total of 86 DOF before original boundary conditions (BC) were applied and reduced the DOF to 84.

A. CANTILEVER BEAM AND EQUIPMENT SETUP

A block of steel 18 inches in length, 8 inches wide and 2 inches thick was placed on a platform as a foundation. The cantilever beam made of T-6061 Aluminum, 48 inches in length, 1.5 inches wide and 0.5 inches thick, density of 0.11 lbf/in^3 and elasticity modulus of 10 E6 lbf/sec^2-in was then placed on top of foundation steel with 42 inches of length extending over the edge of the foundation steel. Two shorter length steel beams, each 6 inches long, 2 inches wide and 1 inch thick were used to access in the elimination platform generated modes. One short steel beam was placed above the Aluminum beam and another was placed beneath the platform. In order to represent a simply supported cantilever, C-clamps were used to clamp the Aluminum beam and associated steel pieces in place as shown in Figure 22.

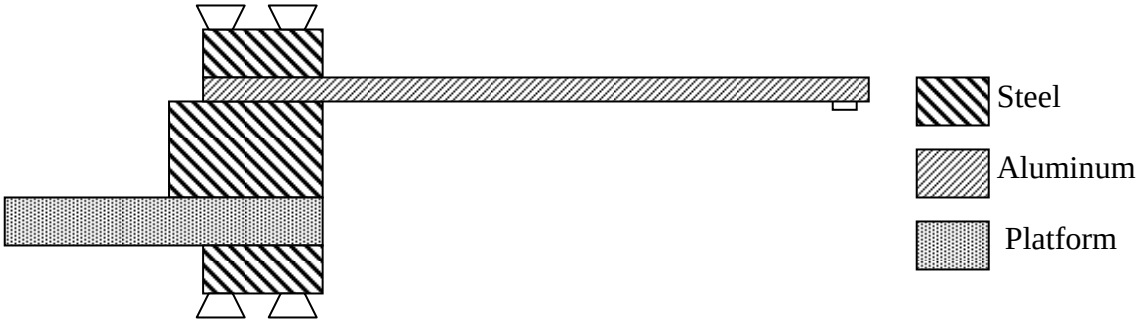

Figure 22. Diagram of Cantilever beam laboratory set-up

The beam consisted of 42 elements, each 1 inch in length; this corresponded to FE model element quantity and length. A Series 336 FLEXCEL ICP accelerometer was threaded into position at node 41 of the beam and wired into Channel 2 on a DACTRON Focus front end digital signal processor (DSP). An excitation was applied by a load cell, which was wired into Channel 1on the DSP. The accelerometer was calibrated and sensitivity adjustment applied in the set-up of RT Pro Focus 5.57 software.

B. DATA COLLECTION

Using DACTRON RT Pro Focus 5.57 software, FRFs were collected when the roving force was applied by the load cell at each node. Node 41 remained the reference as the load cell roved from one node to next allowing for the measurement of the response at each node. Due to this set-up only one column or row of the complete FRF matrix [H] was actually measured. The feasibility of measuring such a small quantity of data is explained in the next section.

1. A RT Pro Focus 5.57 software "Real-time" project was configured to measure 3200 spectral lines, 8192 points, with a delta T of 166.7μs over the frequency range 0-2400Hz. The frequency range of 0-2400 Hz was chosen because it covered the first 10 modes of system and signal resolution was sufficient for data acquisition. The excitation signal proved to be clean and thus no window was used for data measuring.

2. Channel set-up

	Channel 1 (Excitation)	Channel 2(Response)
Max Volts (mV):	0.1	0.3
Quantity:	Force	Accel.
EU:	lbf	gn
mv/EU:	9.48	101.3
Coupling:	ICP AC 7.0 Hz	ICP AC 7.0 Hz
Sensitivity Adjustment:	0	-0.3710

3. Trigger set-up

Source: Analog input
Run Mode: Manual Arm every frame
Input: Channel 1
Slope: Bi-polar
Level (%): 1, Level (V): 0
Pre/Post Points (-/+): -10
Pre/Post Time (-/+): -1.67μs

4. Average set-up:

Type: Linear
Domain: Frequency
Frames: 5 (Each node was excited 5 times and an average taken and saved.)
Accept/Reject: Manual Accept/Reject every frame.
(The user rejects double taps, under powered or overloaded signals.)

5. Modal Coordinate set-up:

Auto increment: ON
Rove: Excitation
Point increment: 1
Export: UFF text format, frequency response.

C. DATA ANAYLSIS

Since the procedure for picking OSET frequencies involved inverting the complete FRF matrix after ABC are applied, smooth FRF signals were ideal. Since the FRFs measured were only one column or row of the complete FRF matrix, the complete FRF matrix was developed by synthesizing the measured FRFs. The smoothing of the FRFs was achieved through curve fitting the measured FRF. ME'Scope'VES was used to analyze, curve fit and synthesize FRFs.

Once all of the FRFs are imported collectively as a "Data Block" into ME'Scope'VES ensure that each trace is labeled correctly in the DOF column. The label should read Roving: Reference, i.e. excitation at node 1 with reference at 41 is 1Z:41Z, Z indicates the axis of motion. In this experiment the Z-axis is the vertical axis.

Under "Modal Parameters" from the "Modes" drop down menu there are three steps used to curve fit and synthesize the data block.

Step 1 - Count peaks. With all traces selected, "Count peaks." Ensure the peak count is the correct quantity of natural frequencies in the frequency range measured, peaks = 10 in this experiment. Also ensure the correct location of peaks. To capture more or less peaks move the horizontal bar accordingly and recount. Once the correct quantity and location is counted proceed to step 2.

Step 2 – Frequencies and damping. The polynomial method was used globally to curve fit the data block. This method uses four extra polynomial terms to compensate for modes not measured. After clicking the "F&D" button a list of damped natural frequencies and % damping ratios was displayed. Check for accuracy of curve fit by selecting "Display, Fit Functions" from "Modes" drop down menu. The curve fit functions for measured FRFs are displayed in Figure 23.

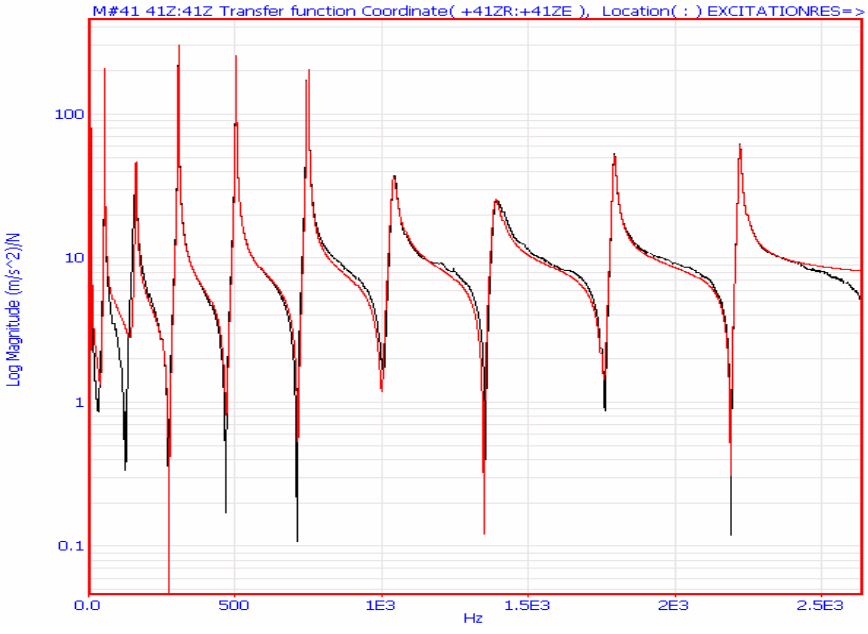

Figure 23. Driving Point Function (41Z:41Z)
Measured FRF = black, Curve Fit = red

Step 3 – Residues and Mode Shapes. With all traces still selected, create "Resides" using polynomial method and save corresponding mode shapes as a Shape Table file (*.shp), ensuring all residues are selected.

In order to save mode shapes in the proper format to be used with MATLAB simulation, 1) display the Shape Table file in "Co-Quad" which displays real and imaginary parts not magnitude and phase of the mode shapes and 2) "Save as ASCII, ASCII text file (.txt)." Attached MATLAB code, Hresidues.m explains in detail the final steps in the proper preparation of shape table file for use with MATLAB.

Once the mode shapes have been saved the synthesized FRFs can be displayed by selecting "Synthesize FRFs…" from the Data Block's "Modes" drop down menu or the Shape Table's "Tools" drop down. These synthesized FRFs are displayed in Figure 24.

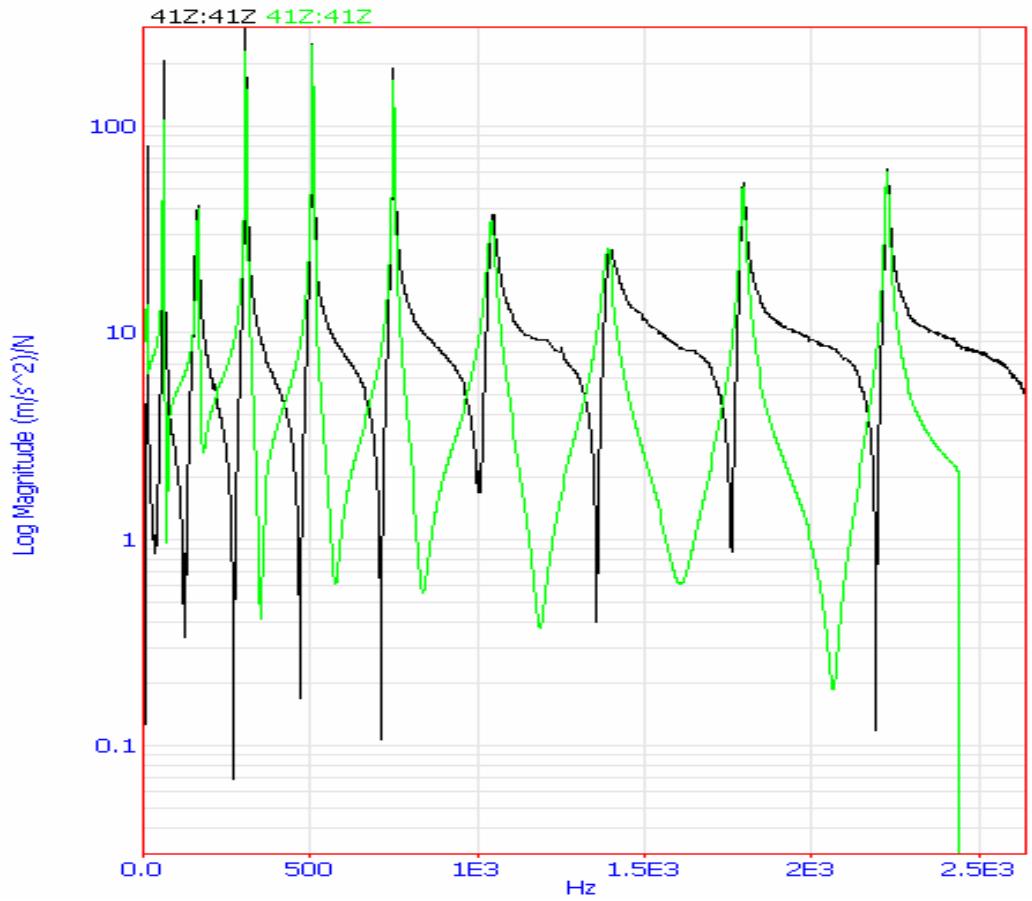

Figure 24. Driving Point Function (41Z:41Z) Measured FRF = black, Synthesized FRF = green

The disparity between the synthesized FRF and measured FRF in the locations of the anti-resonances was an area of great concern because of the importance of the anti-resonance with the inversion of FRF matrix when ABCs are applied. It was believed that the synthesized FRF matrix would not yield accurate OSET and more data would be required to complete the study of ABC in damage detection.

The discrepancy between the synthesized FRF and the curve fit FRF was due to the extra polynomial terms used in curve fit equation but not in the synthesis equation.

Synthesized FRF are necessary due to the fact that only one column of the complete FRF matrix was measured but a complete FRF matrix was needed. The following formula shows why only one row or column of the FRFs needs to be measured in order to completely represent the resonant vibration of a structure in terms of its modes. This formula is utilized by ME'Scope'VES in calculating the synthesized FRF Matrix whose driving point FRF is shown in Figure 24.

$$\left[H(\Omega)\right] = \sum_{k=1}^{modes} \left[\frac{\left[R(k)\right]}{j\Omega + p(k)} + \frac{\left[R^*(k)\right]}{j\Omega + p^*(k)} \right] \tag{5.1}$$

[H(Ω)] = FRF matrix (***n***x***n***)

Ω = forcing frequency

p(k) = pole location for the ***k***th mode = **-σ(k) + jω(k)**

σ(k) = modal damping for the ***k***th mode = $\omega(k)\zeta(k)/(1-\zeta(k)^2)^{½}$

ω(k) = damped natural frequency for the ***k***th mode

ζ(k) = percent of critical damping for the ***k***th mode

[R(k)] = Residue matrix for the ***k***th mode (***n***x***n***) = $A(k)\{\Phi(:,k)\}\{\Phi(:,k)\}^T$

{Φ(:,k)} = mode shape for the ***k***th mode (***n***-vector)

A(k) = scaling constant for the ***k***th mode

n – number of DOFs of the FRF model

***** = denotes complex conjugate

j = denotes imaginary axis in the complex plane

T = denotes transposed vector

Since mode shapes have unique internal relationships and not value, the scaling constant **A(k)** can always be chosen so that **A(k) =1.** With A(k) = 1, the equation simplifies so that,

$$[H(\Omega)] = \sum_{k=1}^{modes} \left[\frac{\left[\{\Phi(:,k)\}\{\Phi(:,k)\}^T \right]}{j\Omega + p(k)} + \frac{\left[\{\Phi^*(:,k)\}\{\Phi^*(:,k)\}^T \right]}{j\Omega + p^*(k)} \right] \qquad (5.2)$$

A simulation was conducted using the natural frequencies, damping ratios and residues generated by ME'Scope'VES. The plots generated from said simulation verified proper use of the above equation and allowed for the comparison to be completed using only MATLAB generated data.

A MATLAB simulation was used to find the effects of quantity of modes used in the calculation of the synthesized FRF matrix and the accuracy of the OSET frequencies when said synthesized FRF matrix was used in conjunction with ABC.

The FE model contained the given dimensions and material properties of the laboratory beam. The FE model was composed of 42 elements with 43 nodes (2 DOF per node for 86 DOF total before cantilever boundary conditions (BC) reduced the DOF to 84.) 42 elements corresponded to the locations of excitation on the beam and made for an easier comparison.

There are a few differences between the formula ME'Scope'VES used and the formula coded for MATLAB use.

1. Mode shapes {Φ(:,k)} is real, {Φ*(:,k)} = {Φ(:,k)}
2. **σ(k) = 0, p(k) = jω(k)** and **p*(k) = - jω(k)**
3. **A (k) = 1.** Scaling is wrong but the trend of the synthesized [H] remains the same.

$$[H(\Omega)] = \sum_{k=1}^{Modes} \left[\frac{\left[\{\Phi(:,k)\}\{\Phi(:,k)\}^T \right]}{j\Omega + j\omega(k)} + \frac{\left[\{\Phi(:,k)\}\{\Phi(:,k)\}^T \right]}{j\Omega - j\omega(k)} \right] \qquad (5.3)$$

The following three plots show the convergence of the anti-resonance with respect to the quantity of modes used in the development of synthesized FRF matrix in MATLAB.

Figure 25 is of the driving point function (41Z:41Z). Notice the shift of the anti-resonance to the left. This shift would indicate at first glance that more modes make the synthesized FRF less accurate since the anti-resonance of the measured FRF is to the right. This quick analysis of the plot is inaccurate and will be proved as such later. Before that discussion other FRF should be studies to see what, if any, difference exist in the synthesized FRF with respect to location on beam.

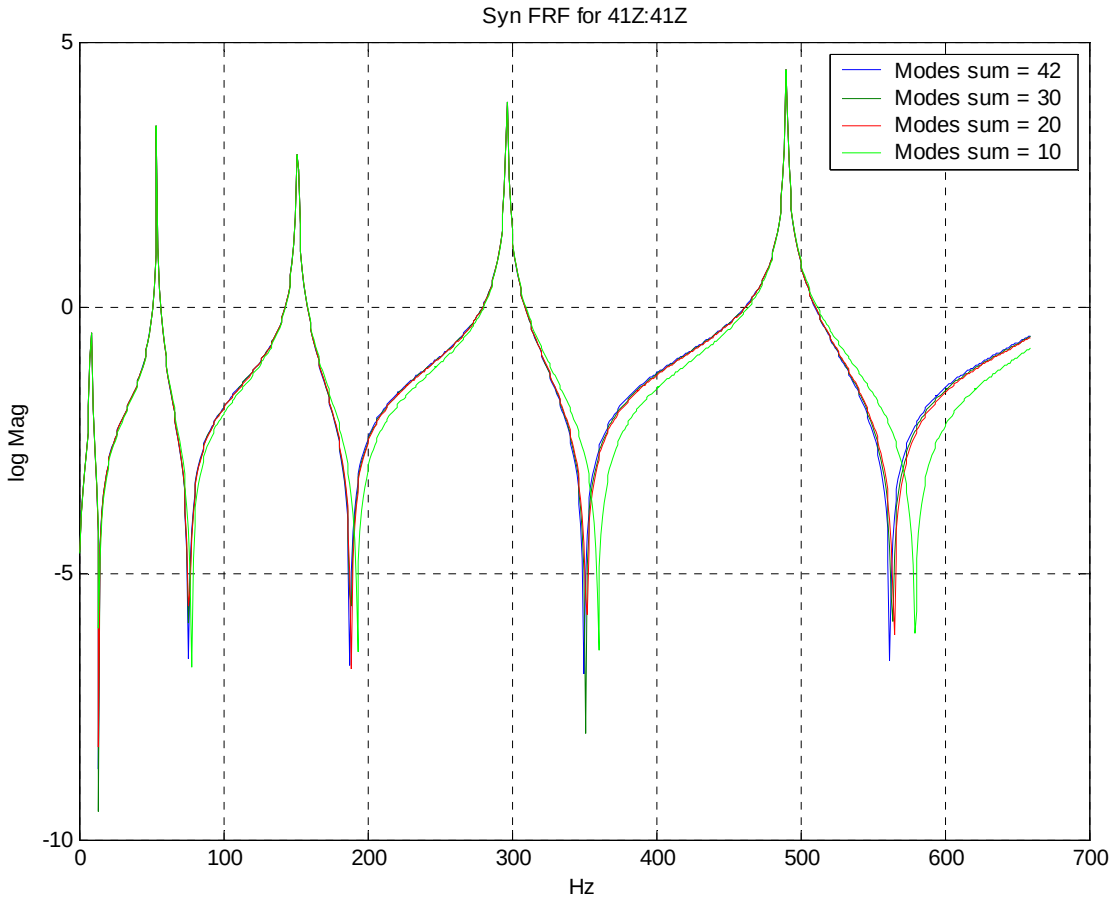

Figure 25. MATLAB Synthesized Driving Point Function (41Z:41Z)

Figure 26 shows the MATLAB synthesized FRF of 31Z:41Z. Notice the quick convergence of the anti-resonance locations. This FRFs only ten inches away from Driving point and the effects of the quantity of modes used are considerable less visible.

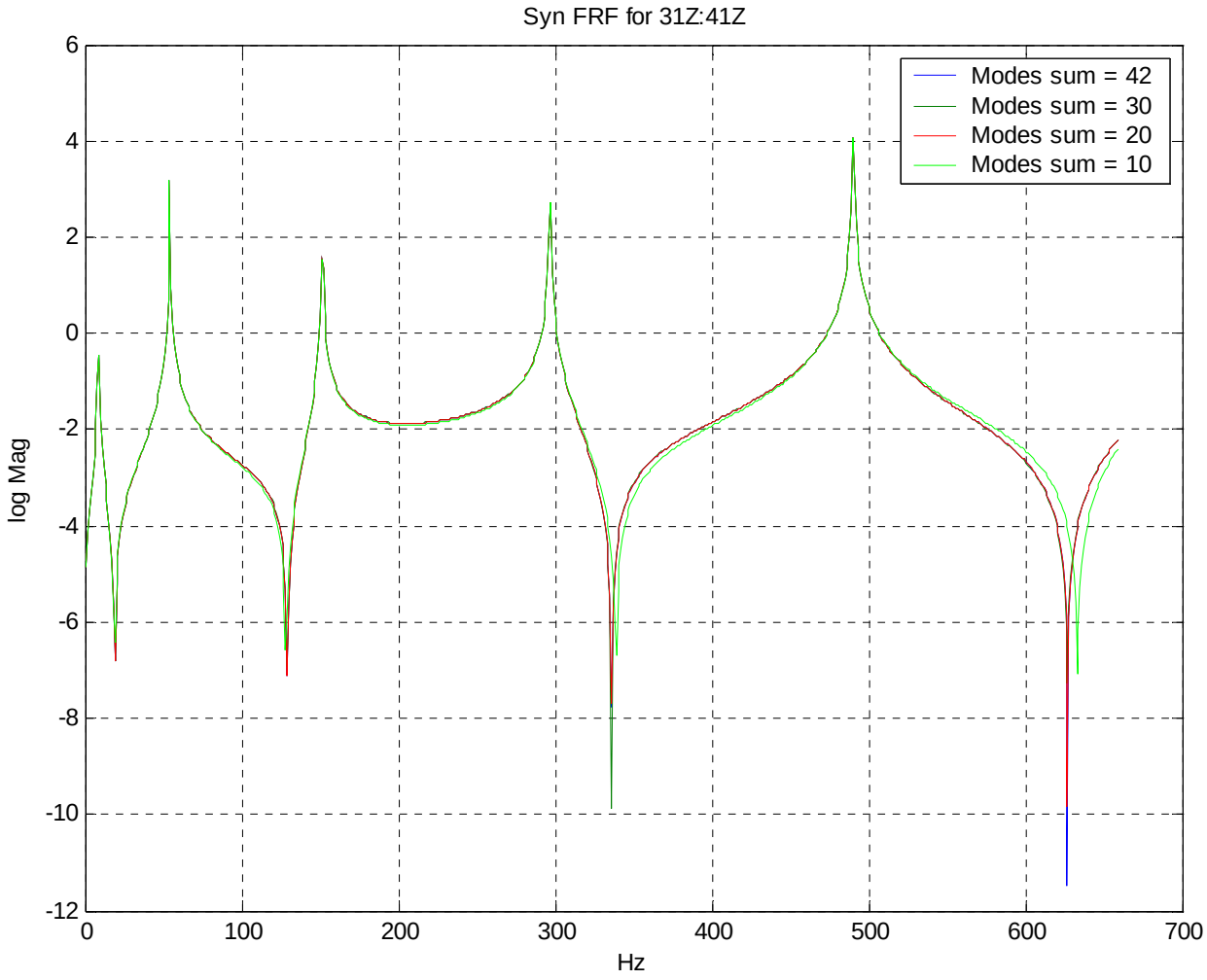

Figure 26. MATLAB Synthesized FRF (31Z:41Z)

In the last plot, Figure 27 which is the MATLAB synthesized FRF of 21Z:41Z, the plots of corresponding quantities of modes used in synthesis nearly lie top each other. This absence of convergence suggests that the effects of the quantity of modes used in synthesis become considerably less important as the distance from the driving point increases.

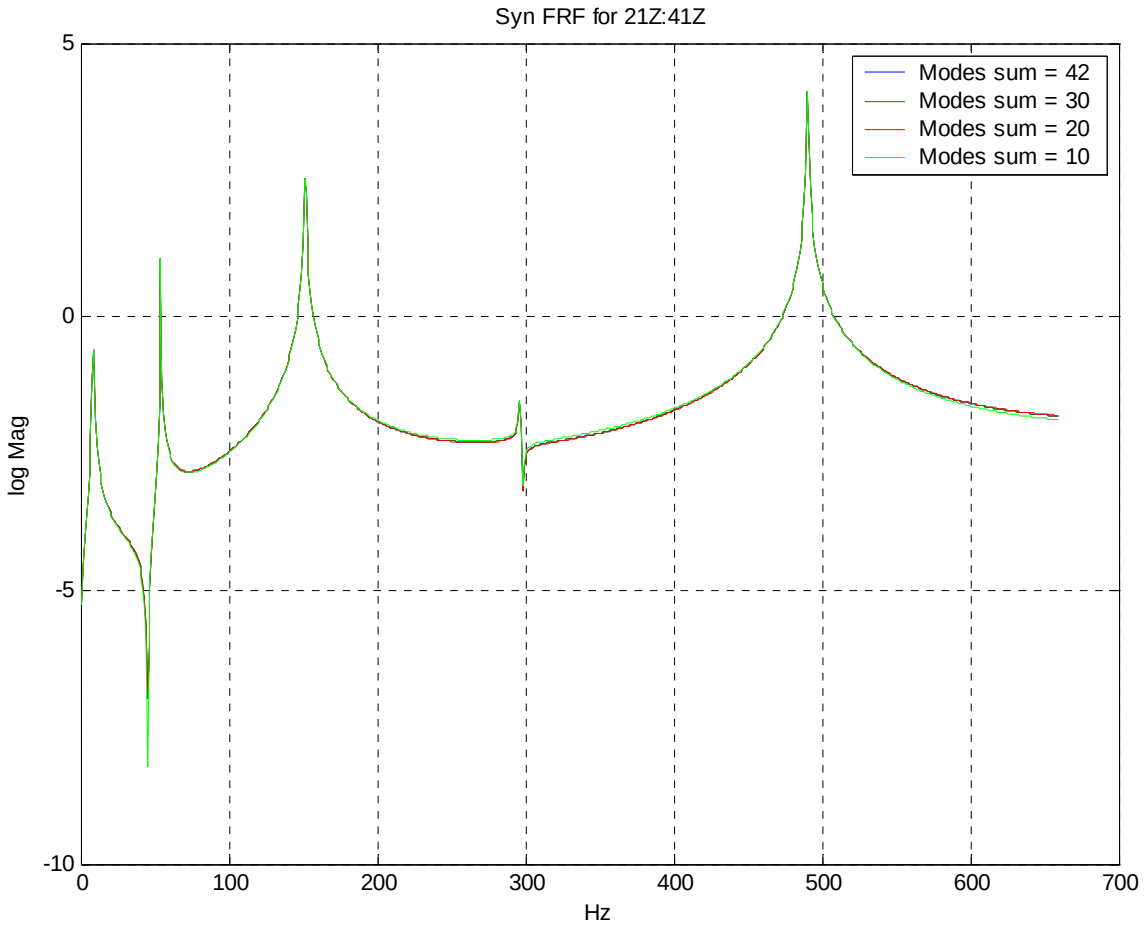

Figure 27. MATLAB Synthesized FRF (21Z:41Z)

An evaluation of the accuracy of the OSET frequencies when ABCs were applied was conducted using a MATLAB simulation. The simulation obtained OSET frequencies of the cantilever beam system:

1) The actual application of the boundary condition, a pin at node 41, to the [K] and [M] and solving for natural frequencies.

2) Synthesis of [H], keeping the pinned DOF, inverting [H_{aa}] and plotting the coordinating impedance matrix, [Z] over the range of 1-600Hz. The smaller range was used for great definition of plots.

Figure 28. shows the overlaying plots of inv [H_{aa}], each plot represents a different quantity of modes used in the synthesis of [H]. The vertical red lines capped with circles represent the actual frequencies of the system with node 41 pinned.

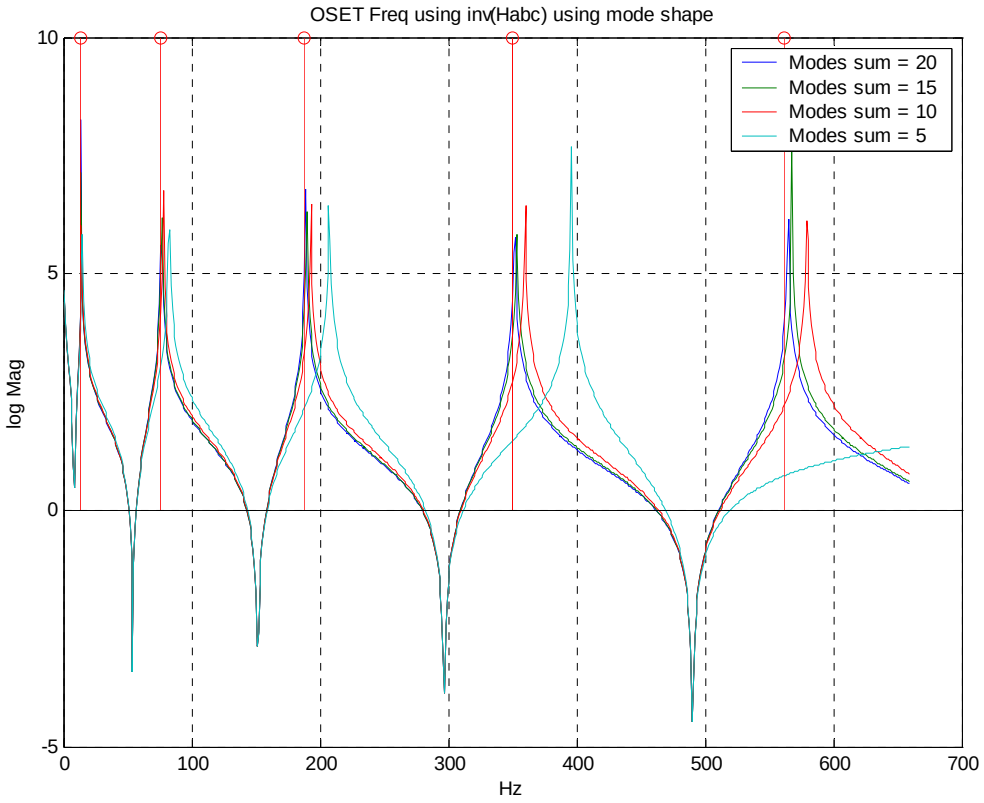

Figure 28. Comparison of OSET Frequencies using inv [Synthesized] and Actual Natural frequencies of the system, pin at node 41.

The convergence of the plots increases as the number of modes used in synthesis is increased thus the improving the accuracy of OSET frequencies. As seen in Figure 28 the higher natural frequencies require a higher quantity of modes to be used in the synthesis of [H]. This conclusion is expected; however, Figure 28 also verifies the use of synthesized FRFs in combination with ABCs which is contrary to the expected result from Figure 24, the comparison plot of the synthesized driving point FRF and the measured driving point FRF. However, synthesizing the complete FRF matrix in ME'Scope'VES was difficult due to a unit problem. If time would have permitted a complete FRF matrix using ME'Scope'VES residues would have been synthesized by a MATLAB program, Hresidues.m and an error localization would have been performed between the most accurate FE model and the measured data.

Since this comparison was not completed and it is recommended that such a comparison be run to validate ABC usage in combination with sensitivity based model updating with actual measured data.

VI. CONCLUSIONS AND RECOMMENDATIONS

Previous works had verified the use of artificial boundary conditions as a method of obtaining additional system frequencies from a single experimental database for improved error localization. The main objective of this thesis was to find a relationship between the accuracy of error location and the artificial boundary conditions used in frequency acquisition.

A. CONCLUSIONS

The following conclusions can be drawn from the analyses presented of ABC configured systems in combination with sensitivity updating:

1. The improvement of error prediction with respect to ABC is relative to the actual location of discrepancy between the model and actual structure.
    a. The underdetermined baseline system utilizing only the first 5 modes predicted with high accuracy if the error was a Mass error near the root end or an EI error near the free end.
    b. In general, the addition of ABC system frequencies improved the error prediction when error was a Mass error near the free end or an EI error near the root end.
2. An accurate method of evaluating accuracy of error prediction with respect to ABC configuration was not satisfactorily found.
    a. The relationship between the mode shapes of the Beam A and Beam X with ABC applied did not provide a good avenue of comparison of the accuracy of error prediction.
    b. A relationship between the condition of the sensitivity matrix and accuracy of error prediction was not found in this study.
    c. The FOM formula used to evaluate the 600 cases run in MATLAB simulation accounted for accuracy of location of error in spite of magnitude of the error.

The following conclusions can be drawn from the experimental data collection:

1. Synthesized data from Me'Scope'VES has an accuracy high enough to yield accurate OSET frequencies if enough modes are measured. However, synthesizing complete FRF matrix was difficult due to unit problem.

B. RECOMMENDATIONS

The following are areas of recommended improvement to continue with this field of study. The first three are recommendations for improving sensitivity-based error localization.

1. Apply two or more changes in varying locations and evaluate error localization with respect to ABC systems.
2. Build a sensitivity matrix by applying a perturbation to every other element and evaluate error localization.
3. Research more accurate methods of evaluating error localization.
4. Before data measuring conduct a MATLAB analysis to find the quantity of modes needed for convergence of anti-resonances. This would improve OSET frequency calculations and reduce time in laboratory.
5. Explore how to synthesis a complete FRF matrix using ME'Scope'VES.
6. Since a comparison between real data and FE data was not completed and it is recommended that such a comparison be run to validate ABC usage in combination with sensitivity based model updating with actual measured data.

APPENDIX A

The following tables for Figure of Merit (FOM) Charts for 10% change either in Mass or EI applied to elements 1:10, but only one change and element at a time. Each columns represents the elements the change is located. Each row represents system used to predict the error. The FOM formula is located in Chapter IV. FOMs of 100 are highlighted.

	Element where the MASS Error is located									
	1	2	3	4	5	6	7	8	9	10
BASE	69	28	65	0	6	74	21	29	40	72
ABC 1	100	89	94	92	93	94	96	96	98	97
ABC 2	97	90	92	79	80	89	92	95	94	99
ABC 3	46	29	30	18	34	16	14	3	33	19
ABC 4	33	19	85	82	53	71	52	27	93	49
ABC 5	37	13	0	7	5	58	75	11	29	34
ABC 6	98	98	95	98	99	99	98	98	97	97
ABC 7	98	97	99	98	98	99	98	87	98	89
ABC 8	36	50	70	3	59	61	18	51	44	55
ABC 9	75	90	92	74	72	97	46	43	25	14
ABC 10	92	95	99	90	90	99	89	72	82	99

Table 1. FOM of MASS Error Prediction using 10 modes of each system.

	Element where the EI Error is located									
	1	2	3	4	5	6	7	8	9	10
BASE	77	27	38	12	52	23	3	6	9	74
ABC 1	100	96	92	95	73	50	53	81	95	85
ABC 2	93	99	48	36	29	4	1	0	69	1
ABC 3	53	0	96	4	6	3	1	4	8	2
ABC 4	64	56	63	64	27	43	15	14	77	39
ABC 5	0	2	0	8	68	0	5	1	1	0
ABC 6	99	99	97	97	98	96	99	99	91	97
ABC 7	97	98	97	96	95	90	93	99	95	90
ABC 8	99	99	95	93	93	98	99	99	98	96
ABC 9	98	99	98	93	94	96	99	97	98	100
ABC 10	99	99	99	98	98	99	99	98	96	95

Table 2. FOM of EI Error Prediction using 10 modes of each system.

		Element where Mass Error is located.									
		1	2	3	4	5	6	7	8	9	10
	Modes										
	1:5	100	77	60	52	47	34	53	64	48	8
	6:10	100	94	86	37	25	1	9	19	98	33
ABC 1+Base (1:5)	11:15	4	3	1	8	12	14	36	1	19	0
	1:5	100	98	97	67	81	71	69	92	67	27
	6:10	16	5	85	80	85	83	43	84	82	84
ABC 2 +Base (1:5)	11:15	29	42	73	81	97	90	96	63	87	96
	1:5	100	0	4	7	5	11	4	18	7	1
	6:10	56	76	15	93	65	73	78	58	88	78
ABC 3 +Base (1:5)	11:15	87	78	91	98	95	99	94	100	98	99
	1:5	100	97	100	99	100	92	94	100	93	78
	6:10	7	69	85	53	43	23	43	2	65	9
ABC 4+Base (1:5)	11:15	92	71	54	66	93	89	98	83	86	97
	1:5	1	2	9	19	17	18	18	1	2	0
	6:10	34	39	12	66	0	52	1	5	52	24
ABC 5+Base (1:5)	11:15	92	66	23	14	88	95	12	0	15	96
	1:5	100	98	100	100	100	98	97	98	99	87
	6:10	72	90	53	98	72	82	72	84	49	80
ABC 6+Base (1:5)	11:15	93	96	61	93	97	99	93	83	59	99
	1:5	100	10	19	3	96	95	100	11	6	2
	6:10	48	90	89	88	19	56	72	23	7	23
ABC 7+Base (1:5)	11:15	87	72	72	72	99	51	63	81	47	64
	1:5	100	96	99	97	84	62	40	97	75	3
	6:10	64	66	28	1	59	37	97	94	5	6
ABC 8+Base (1:5)	11:15	97	99	99	100	95	92	98	99	11	64
	1:5	100	46	48	100	39	97	100	3	3	0
	6:10	83	97	92	78	64	92	98	89	93	92
ABC 9+Base (1:5)	11:15	100	99	60	90	84	88	99	75	83	10
	1:5	100	97	100	98	100	100	99	100	92	66
	6:10	4	16	32	5	19	11	78	28	1	17
ABC 10+Base (1:5)	11:15	97	94	97	98	100	98	98	99	95	81

Table 3. FOM of Mass Error Prediction using Base modes (1:5) and 5 modes from ABC system, (1:5), (6:10), or (11:15)

	Element where Mass Error is located.										
		1	2	3	4	5	6	7	8	9	10
	Modes										
	1:5	0	0	100	100	0	100	100	0	0	100
	6:10	0	97	0	98	97	93	0	0	0	99
ABC 1	11:15	96	0	98	99	99	99	0	0	0	0
	1:5	0	0	0	100	100	100	100	0	0	100
	6:10	0	76	99	97	0	0	0	0	98	99
ABC 2	11:15	80	71	97	0	0	0	0	97	91	0
	1:5	0	96	0	0	100	0	100	100	100	0
	6:10	0	0	85	97	99	96	0	0	0	97
ABC 3	11:15	0	83	91	0	0	0	99	100	99	0
	1:5	0	0	100	0	0	100	100	0	100	100
	6:10	0	97	0	96	99	99	0	0	0	97
ABC 4	11:15	0	0	81	83	97	0	0	0	98	99
	1:5	0	95	97	95	0	100	0	0	0	100
	6:10	57	0	87	0	0	76	92	0	0	94
ABC 5	11:15	0	0	96	97	98	0	0	98	98	0
	1:5	0	100	0	100	100	0	0	100	0	100
	6:10	98	0	0	0	99	99	98	99	0	0
ABC 6	11:15	0	0	0	100	100	0	0	98	95	100
	1:5	0	0	100	100	0	100	0	0	99	100
	6:10	97	0	0	0	96	99	97	85	0	0
ABC 7	11:15	0	0	100	99	0	100	0	0	97	99
	1:5	0	99	100	0	100	100	0	0	0	99
	6:10	99	99	0	0	0	0	96	99	0	83
ABC 8	11:15	99	100	99	0	0	94	0	0	0	31
	1:5	0	0	100	100	0	100	0	100	0	96
	6:10	97	0	0	0	94	0	97	0	96	98
ABC 9	11:15	0	100	98	99	0	0	100	0	0	74
	1:5	0	100	100	100	0	100	100	0	0	0
	6:10	98	0	0	0	96	0	99	0	98	100
ABC 10	11:15	99	98	0	0	100	96	0	99	0	0

Table 4. FOM of Mass Error Prediction using 5 modes from ABC system, (1:5), (6:10), or (11:15)

	Element where EI Error is located.										
		1	2	3	4	5	6	7	8	9	10
	Modes										
	1:5	5	98	100	100	97	96	93	100	88	100
	6:10	100	98	95	97	88	81	82	93	98	95
ABC 1 +BASE(1:5)	11:15	97	88	56	97	78	66	97	74	82	78
	1:5	91	100	98	99	100	100	99	100	96	100
	6:10	22	96	64	87	70	77	83	85	47	95
ABC 2+BASE(1:5)	11:15	37	94	97	74	86	81	91	91	81	91
	1:5	15	41	97	48	88	97	98	98	75	100
	6:10	8	13	49	97	24	1	2	64	38	5
ABC 3+BASE(1:5)	11:15	13	2	11	21	0	19	10	5	2	7
	1:5	94	99	99	100	100	100	100	100	99	100
	6:10	5	15	2	0	28	14	37	5	86	23
ABC 4+BASE(1:5)	11:15	70	0	11	92	99	76	98	29	86	96
	1:5	1	2	41	17	50	2	26	3	2	42
	6:10	6	82	0	34	63	48	7	1	11	43
ABC 5+BASE(1:5)	11:15	84	63	25	50	100	94	37	34	58	94
	1:5	99	99	99	100	100	96	98	87	89	100
	6:10	32	40	26	6	18	16	75	59	7	22
ABC 6+BASE(1:5)	11:15	100	94	95	99	98	99	100	68	81	99
	1:5	98	100	100	100	100	98	96	97	85	100
	6:10	76	87	59	17	24	70	81	99	78	35
ABC 7+BASE(1:5)	11:15	97	96	94	98	99	87	99	100	95	88
	1:5	100	93	100	94	97	98	90	100	90	100
	6:10	95	98	18	40	22	56	61	97	78	63
ABC 8+BASE(1:5)	11:15	92	97	86	99	92	72	95	89	98	9
	1:5	97	100	99	100	99	99	100	100	69	100
	6:10	98	99	96	90	84	90	91	96	98	100
ABC 9+BASE(1:5)	11:15	99	87	93	94	97	91	99	90	97	95
	1:5	100	100	100	100	100	100	100	100	100	100
	6:10	99	99	97	97	97	98	92	91	75	95
ABC 10+BASE(1:5)	11:15	98	94	94	96	100	95	95	100	98	97

Table 5. FOM of EI Error Prediction using Base modes (1:5) and 5 modes from ABC system, (1:5), (6:10), or (11:15)

		Element where EI Error is located.									
		1	2	3	4	5	6	7	8	9	10
	Modes										
	1:5	0	100	0	0	0	100	100	100	100	0
	6:10	0	99	97	0	97	0	0	0	98	98
ABC 1	11:15	0	94	96	99	0	0	99	0	0	100
	1:5	0	0	0	100	100	100	0	100	100	0
	6:10	90	0	0	98	0	96	0	0	98	98
ABC 2	11:15	85	0	0	0	98	0	0	98	82	97
	1:5	0	100	0	100	0	0	100	100	100	0
	6:10	83	0	0	0	99	94	0	93	0	98
ABC 3	11:15	98	88	0	0	99	0	99	100	0	0
	1:5	0	0	100	0	100	0	100	100	100	0
	6:10	83	93	0	0	0	97	0	89	0	98
ABC 4	11:15	0	0	71	100	0	0	0	96	99	100
	1:5	0	0	0	0	0	100	57	100	16	100
	6:10	70	84	0	97	0	0	0	68	0	56
ABC 5	11:15	0	0	92	92	0	0	0	92	95	99
	1:5	0	100	0	100	100	100	0	100	0	0
	6:10	99	0	98	0	99	0	0	0	99	96
ABC 6	11:15	0	99	0	100	100	0	0	97	0	100
	1:5	0	0	100	100	0	100	100	0	98	0
	6:10	98	0	97	0	98	99	0	0	0	81
ABC 7	11:15	0	0	100	99	0	99	0	0	96	100
	1:5	0	0	0	100	100	100	0	100	100	0
	6:10	99	99	0	0	92	98	0	0	90	0
ABC 8	11:15	0	99	97	0	95	90	0	0	0	40
	1:5	0	0	0	0	0	100	100	100	99	100
	6:10	98	0	97	0	88	97	0	0	0	100
ABC 9	11:15	100	99	0	0	0	0	100	97	0	100
	1:5	100	0	0	100	0	100	100	0	100	0
	6:10	99	0	0	0	95	0	99	0	98	100
ABC 10	11:15	0	95	0	0	100	93	0	100	98	0

Table 6. FOM of EI Error Prediction 5 modes from ABC system, (1:5), (6:10), or (11:15)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

ABCrunTHRU_crs.m

```
% This program calculates the condition number of the following
% sensitivity matrices used to calculate the DV (error prediction).
% 1) Base system only 5 modes (underdetermined)
% 2) ABC system 10 modes
% 3) Base system 5 modes + 5 modes from ABC system
% The last system is calculated 3 times.  Once for modes 1-5,
% another for modes 6-10, and again for modes 11-15.
%
% This program is called from Build2Beams.m.
%
% Written by Constance R S Fernandez, Spring 2004
% INPUTS
% --------
% icnt_oset
% T_sens_tot
% vect_lam_tot

% OUTPUTS
% -------
% aa
% dv_a,  dv_b,  dv_c
% intervel
% mode
% startmode
% ten
% dv_cal_ABC - matrix
% dv_calABCten - matrix
% dv_cal_BasePlus - matrix
% cond_ABC - matrix
% cond_ABCten - matrix
% cond_basePlus - matrix

% ----INITIALIZATION-------

abc = 0;
ten = 1;
intervel = 1;
int_abc = 1;
for aa = 1:icnt_oset +1 % number of conditions (base + ABC)
   a_c = 1; % reinitialize for each ABC system
```

```
   for mode = 1:3 % 3 sets of modes per ABC system (10 element beam)
      % (modes 1:5, 6:10, 11:15)
      startmode = abc + a_c;

      dv_a = [startmode: startmode+4]; % modes
      dv_c = [ten:ten+9]; % the first 10 modes of each ABC system
      % (modes 1-10 only)

      if dv_a == [1:.25*size(T_sens_tot,2)]; % if sensitivity matrix
         %has only 5 rows than the modes used are all 5, else modes
         %used are first five (base) and a set of selected 5 modes
         %of ABC system for a total of 10 modes.
         dv_b = [dv_a];
      else
         dv_b = [1:.25*size(T_sens_tot,2), dv_a];
      end
      %---Base System only---(underdetermined)

      % save DV calculates of as matrix for plotting
      dv_cal_ABC(:,intervel) = T_sens_tot(dv_a,:)\vect_lam_tot(dv_a);
      % condition number of Sensitivity matrix used in DV cal.
      cond_ABC(intervel,1) = cond(T_sens_tot(dv_a,:));

      % ---Base + 5 modes of ABC system ----
      dv_cal_BasePlus(:,intervel) = T_sens_tot(dv_b,:)\vect_lam_tot(dv_b);
      cond_basePlus(intervel, 1) = cond(T_sens_tot(dv_b,:));

      intervel = intervel + 1;
      a_c = a_c + 5; % five modes used at a time

   end % "mode" loop

   % ---ABC system only ----
      dv_cal_ABCten(:,int_abc) = T_sens_tot(dv_c,:)\vect_lam_tot(dv_c);
      cond_ABCten(int_abc,1) = cond(T_sens_tot(dv_c,:));

   int_abc = int_abc + 1;
   abc = abc + 19; % advances to next ABC system, must change number "19"
   % to reflect the number of DOF in beam.  This beam had 10 elements thus
   % 19 DOF.
   ten = ten + 19; % advances to next ABC system
end % "aa" loop
% ********************* END  ABCrunTHRU_crs.m  ************************
```

AddLumpMass.m

```
%  This script constructs a vector of lumped masses
%  which is added to the diagonal of the BeamX mass matrix.
%          Mass added to [mx] in Assemble2Beams.m
%
%  Written by Prof J.H. Gordis

% Inputs needed
% -----------------
% num_elements
% mx

% Outputs
% -----------------
% mass_diag
% mass_node
% mass_coord
% mass_DOF
% mx - updated

disp(' ');disp(' ');
disp(' ************************************************************')
disp(' ****          Lumped mass addition to beams         ****')
disp(' ****  Lumpmass DOFs defined for UNRESTRAINED beams  ****')
disp(' ************************************************************')

% initalize
if exist('mass_diag') == 0;  % define and apply lumped mass vector.

add_mass = 'n';
add_mass = input(' Add lumped masses to BeamX ? (y/n) ','s');

% Initialize vector to add to [mx] diagonal.

mass_diag = zeros(2*(num_elements+1),1);

  while add_mass == 'y';

    mass_node = input(' Node number for lumped mass ? ');

    mass_coord = input(' Translation or Rotation for lumped mass ? (t/r) ','s');

    if mass_coord == 't'; % Translational lumped mass
       mass_DOF = 2 * mass_node - 1;
    elseif mass_coord == 'r'; % rotational lumped mass
```

```
            mass_DOF = 2 * mass_node;
        end

        mass_diag(mass_DOF) = input(' Enter value of mass/inertia (in "lbf-sec^2/in" ');
   % puts lumped mass on correct DOF
        add_mass = input(' Add another lumped mass ?  (y/n) ','s');
   % can continue adding mass until 'n' is entered

  end;   % End while loop

end;   % End exist('mass_diag')

mx = mx + diag(mass_diag);   % Add lumped masses to [mx]:

% ********************  END ADDLUMPMASS.M  ************************
```

Assemble2Beams.m

```
% This program assembles the [K] and [M] matrices of 2 beams
% Written by Prof Gordis
% ~~~~~~~ ~~~~~~~~~~~~~~~~

% Inputs needed
% -----------------
% ndof
% num_elements
% element_length
% element_EI
% element_mass

% Programs needed:
% -----------------
% fbeamkm

% Outputs
% -----------------
% ka ma kx mx
% (all others cleared)


% Loop over the two beams:
% ~~~~ ~~~~ ~~~ ~~~ ~~~~~~
for icnt_beams = 1:2;

  k=zeros(ndof,ndof);
  m=zeros(ndof,ndof);

% Loop over the number of elements in the structure:
% ~~~~ ~~~~ ~~~ ~~~~~~~ ~~ ~~~~~~~~~ ~~ ~~~ ~~~~~~~~~~

  for elnum = 1:num_elements;

    dof1=2*connect(elnum,1)-1;
    dof2=dof1+1;
    dof3=2*connect(elnum,2)-1;
    dof4=dof3+1;

% ... set up beamel function call:

    l_temp = element_length;       % Using fixed element lengths
    ei_temp = element_EI(elnum,icnt_beams);
    m_temp = element_mass(elnum,icnt_beams);
```

```
   [kbeam,mbeam]=fbeamkm(l_temp,ei_temp,m_temp);

   k(dof1:dof2,dof1:dof2)=k(dof1:dof2,dof1:dof2)+kbeam(1:2,1:2);
   k(dof1:dof2,dof3:dof4)=k(dof1:dof2,dof3:dof4)+kbeam(1:2,3:4);
   k(dof3:dof4,dof1:dof2)=k(dof3:dof4,dof1:dof2)+kbeam(3:4,1:2);
   k(dof3:dof4,dof3:dof4)=k(dof3:dof4,dof3:dof4)+kbeam(3:4,3:4);

   m(dof1:dof2,dof1:dof2)=m(dof1:dof2,dof1:dof2)+mbeam(1:2,1:2);
   m(dof1:dof2,dof3:dof4)=m(dof1:dof2,dof3:dof4)+mbeam(1:2,3:4);
   m(dof3:dof4,dof1:dof2)=m(dof3:dof4,dof1:dof2)+mbeam(3:4,1:2);
   m(dof3:dof4,dof3:dof4)=m(dof3:dof4,dof3:dof4)+mbeam(3:4,3:4);

% end loop over the number of elements:
  end

% Reassign k and m to new variables and add lumped masses
% ~~~~~~~~ ~ ~~~ ~ ~~ ~~~ ~~~~~~~~~~ ~~~ ~~~ ~~~~~~ ~~~~~~

  if icnt_beams == 1;
       ka = k;ma = m;
  elseif icnt_beams == 2;
       kx = k;mx = m;
  end

% End icnt_beams loop:
end

clear dof1 dof2 dof3 dof4 l_temp ei_temp m_temp icnt_beams
clear k m kbeam mbeam elnum
%  ********************  END Assemble2Beams.m  ************************
```

AssembleSens_crs.m

```
%
% This program assembles the total sensitivity matrix, T_sens_tot and
% total lam vector,  vect_lam_tot and assembles the relative frequency
% error between the natural frequencies of Beam A and Beam X
% Written By Constance R S Fernandez, Spring 2004

% INPUTS
% -------
% vect_lamx_oset
% vect_lam_oset
% vect_lam
% T_sens_oset
% T_sens
%
% OUTPUTS
% -------
% vect_OSET
% vect_lam_tot
% T_sens_tot
% freq_OSET
% freq_OSETx
% rel_freqERROR

vect_OSET = vect_lamx_oset - vect_lam_oset;
% lamx from actual beam with error oset, lam from FE model oset
% Creating a vector of lam differences calculated (Lx-La)

if vect_OSET == 0;
   % when vector is empty at first, the total vector is equal to the
   % lam vector of Beam A, i.e. the first 19 values of vect_lam_tot are
   % the natural freq squared (rad^2/sec^2) of Beam A

   vect_lam_tot = vect_lam;
else
     vect_lam_tot = cat(1, vect_lam, vect_OSET);
end


if T_sens_oset == 0;

   T_sens_tot = T_sens;
else
   T_sens_tot = cat(1, T_sens, T_sens_oset);
end
```

```
freq_OSET = sqrt(abs(vect_OSET))/2/pi;
% Natural frequency vector of Beam A in Hz
freq_OSETx = sqrt(abs(vect_lamx_oset))/2/pi;
% Natural frequency vector of Beam X in Hz
rel_freqERROR = freq_OSET./freq_OSETx*100;
% Relative error between Beam A OSET natural freq and Beam X OSET natural Freq.

% ********************  END AssembleSens_crs.m  ***********************
```

BeamA_Prompt.m

```
% Written by Prof. Gordis

% Programs needed:
% -----------------
% BeamProperties_crs

% Outputs
% -----------------
% total_length
% num_elements
% nominal_EI
% nominal_area
% nominal_density
% ndof
% element_length
% element_EI
% element_area
% element_density
% element_mass


% _____________________________________________________________________
%
%                    Prompt User for BeamA Data
% _____________________________________________________________________


  disp(' ');disp(' ');
  disp(' Enter nominal physical properties for first beam')
  disp(' ~~~~~ ~~~~~~~~ ~~~~~~~~~ ~~~~~~~~~~~ ~~~ ~~~~~ ~~~~')

props_file = 'n';
props_file = input('  Run "BeamProperties.m" script to define nominal properties ? (y/n)
>> ','s');

if props_file == 'y';

  BeamProperties_crs;

else;

  total_length    = input('  Enter the total beam length in inches:  ');
  num_elements    = input('  Enter the number of elements:  ');
  nominal_EI      = input('  Enter the nominal EI value (lbf/in^2):  ');
  nominal_area    = input('  Enter the nominal cross section area (in^2):  ');
```

```
 nominal_density = input('  Enter the nominal weight density (lbf/in^3):  ');

end;


 for icnt_elements = 1:num_elements;
     connect(icnt_elements,1:2) = [icnt_elements,icnt_elements+1];
 end

 %ndof=length(connect)*2+2;   % this is unrestrained beam DOF
 ndof=num_elements*2+2; % CRS addition

 element_length = total_length/num_elements;
 element_EI(:,1) = nominal_EI * ones(num_elements,1);
 element_area(:,1) = nominal_area * ones(num_elements,1);
 element_density(:,1) = nominal_density * ones(num_elements,1);
 element_mass(:,1) = (element_density .* element_area * element_length)/386.4;


% Prompt to randomize the beam EI properties:
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% randomize = input('  Randomize the beam EI properties?  (y/n) >> ','s');
% if randomize == 'y';
%
%      load_rand_vec = 'b';
%       disp(' ');
%       disp('  To build a new random vector      - type "b" ');
%       disp('  To load an existing random_vector - type "l" ');
%       load_rand_vec = input(' Enter choice >> ','s');
%
%      if load_rand_vec == 'b';   % Build rand_vec
%          RandomizeProps;
%       else;                     % Load existing rand_vec from file
%          load rand_vec
%       end;
%
%
%       element_EI(:,1) = element_EI(:,1) .* (1 + rand_vec);
%
%
% end;

 % End Data input for 1st beam: Copy data for 2nd beam:
 % ~~~ ~~~~ ~~~~~ ~~~ ~~~ ~~~~  ~~~~ ~~~~ ~~~ ~~~ ~~~~~
  element_EI(:,2) = element_EI(:,1);
```

```
    element_area(:,2) = element_area(:,1);
    element_density(:,2) = element_density(:,1);
    element_mass(:,2) = element_mass(:,1);

clear props_file
 % ********************* END  BeamA_Prompt.m  ************************
```

BeamH4141.m

```
% This program plots the peak of driving point of Beam X  using impedence
% formula Z = [K] - omega^2*[M] +j*c*omega and omegas near driving point.
% Used for comparison of formula and natural frequencies calculations.

% Inputs
% -------
% kx_beamBC
% mx_beamBC

% Needed Programs
% ---------------
% fModes

% Outputs
% --------
% lam
% phi
% W
% iR
% f
% freq
% Zx_beam
% omega
% C
% hx_beam
% H4141

% ----Start Program ----
load testBEAM % saved test data file from Build2Beams_crs.m

c = 0.02; % damping ratio
iR = 0; % initialize loop counter
[lam,phi]=fModes(kx_beamBC,mx_beamBC);
freq = sqrt(lam)/2/pi;
disp('Natural freq in HZ')
f = freq(1:5)
for omega = [0:1.831050e-001:6.589949e+002]; % in Hz
        omega = omega*2*pi; % rad/sec
        iR = iR + 1;
        % loop counter% system impedence of multi DOF system
        Zx_beam= kx_beamBC - omega.^2 * mx_beamBC + j*c*omega;

        hx_beam = inv(Zx_beam);
        H4141(iR) = hx_beam(82,82);
```

```
    end
    w =  [0:1.831050e-001:6.589949e+002];
plot(w, log(abs(H4141))), grid on
axis tight
title ('Driving point H(41,41), FE model')
xlabel ('Hz')
ylabel ('log Magnitude')
% ********************  End BeamH4141.m  ************************
```

BeamH4141q.m

```
% Written by Constance Fernandez, Spring 2004
% This program creates and plots mode summation for H4141, H4131, H4121,
% H4111.

% Inputs
% -------
% kx_beamBC
% mx_beamBC

% Programs
% ---------
% fOset_from_Aset

% Outputs
% --------
% aset, oset
% k,m
% zeta
% lam, phi
% freq
% mm
% iR, i
% SUM
% omega
% H4141, H4131, H4121, H4111
% H41, H31, H21, H11
% w, ww

% ---Start Program -----

load testBEAM % FE generated data file
load HsynMEscope % measured data file

aset = [1:2:83];
oset = fOset_from_Aset(84, aset);
k = kx_beamBC;
m = mx_beamBC;

zeta = .02;

[lam,phi]=fModes(k,m);
freq = sqrt(lam)/2/pi;
for mm = 1:9;
```

```
   iR = 0;
   SUM = [42, 40, 35, 30, 25, 20, 15, 10, 5];

   for omega = [0:.5:2240]; % in Hz
      omega = omega;
      iR = iR + 1;
      H4141 = 0;
      H4131 = 0;
      H4121 = 0;
      H4111 = 0;

      for i = 1:SUM(mm)
         H4141 = ((phi(82,i)).*(phi(82,i)))/((freq(i)).^2 - omega^2) + H4141;
         H4131 = ((phi(82,i)).*(phi(62,i)))/((freq(i)).^2 - omega^2) + H4131;
         H4121 = ((phi(82,i)).*(phi(42,i)))/((freq(i)).^2 - omega^2) + H4121;
         H4111 = ((phi(82,i)).*(phi(22,i)))/((freq(i)).^2 - omega^2) + H4111;
         % eq from ABC examples

      end
      H41(iR,mm) =  H4141 ;
      H31(iR,mm) =  H4131 ;
      H21(iR,mm) =  H4121 ;
      H11(iR,mm) =  H4111 ;

   end
end
w = [0:.5:2240];
ww =  [0:1.831050e-001:6.589949e+002];

figure(1) % Mode summation for H4141
plot(w, log(abs(H41(:,1))),w, log(abs(H41(:,2))),w, log(abs(H41(:,3))),...
   w, log(abs(H41(:,4))), w, log(abs(H41(:,5))),w, log(abs(H41(:,6))),...
   w, log(abs(H41(:,7))),w, log(abs(H41(:,8))), w, log(abs(H41(:,9)))), grid on...
%     ww, (log (abs(HH))),'r')
axis tight
xlabel ('Hz')
ylabel ('log Mag')
title( 'Syn FRF for 41Z:41Z')
legend ('Modes sum = 42', 'Modes sum = 40', 'Modes sum = 35', 'Modes sum = 30', ...
   'Modes sum = 25', 'Modes sum = 20', 'Modes sum = 15', 'Modes sum = 10', ...
   'Modes sum = 5')%'MeScope Syn')

figure(2)% Mode summation for H4131
plot(w, log(abs(H31(:,1))),w, log(abs(H31(:,2))),w, log(abs(H31(:,3))),...
   w, log(abs(H31(:,4))), w, log(abs(H31(:,5))),w, log(abs(H31(:,6))),...
   w, log(abs(H31(:,7))),w, log(abs(H31(:,8))), w, log(abs(H31(:,9)))), grid on
```

```
axis tight
xlabel ('Hz')
ylabel ('log Mag')
title( 'Syn FRF for 31Z:41Z')
legend ('Modes sum = 42', 'Modes sum = 40', 'Modes sum = 35', 'Modes sum = 30', ...
   'Modes sum = 25', 'Modes sum = 20', 'Modes sum = 15', 'Modes sum = 10', ...
   'Modes sum = 5')

figure(3)% Mode summation for H4121
plot(w, log(abs(H21(:,1))),w, log(abs(H21(:,2))),w, log(abs(H21(:,3))),...
   w, log(abs(H21(:,4))), w, log(abs(H21(:,5))),w, log(abs(H21(:,6))),...
   w, log(abs(H21(:,7))),w, log(abs(H21(:,8))), w, log(abs(H21(:,9)))), grid on
axis tight
xlabel ('Hz')
ylabel ('log Mag')
title( 'Syn FRF for 21Z:41Z')
legend ('Modes sum = 42', 'Modes sum = 40', 'Modes sum = 35', 'Modes sum = 30', ...
   'Modes sum = 25', 'Modes sum = 20', 'Modes sum = 15', 'Modes sum = 10', ...
   'Modes sum = 5')


figure(4)% Mode summation for H4111
plot(w, log(abs(H11(:,1))),w, log(abs(H11(:,2))),w, log(abs(H11(:,3))),...
   w, log(abs(H11(:,4))), w, log(abs(H11(:,5))),w, log(abs(H11(:,6))),...
   w, log(abs(H11(:,7))),w, log(abs(H11(:,8))), w, log(abs(H11(:,9)))), grid on
axis tight
xlabel ('Hz')
ylabel ('log Mag')
title( 'Syn FRF for 11Z:41Z')
legend ('Modes sum = 42', 'Modes sum = 40', 'Modes sum = 35', 'Modes sum = 30', ...
   'Modes sum = 25', 'Modes sum = 20', 'Modes sum = 15', 'Modes sum = 10', ...
   'Modes sum = 5')

% ********************  END BeamH4141q.m  ************************
```

BeamProperties_crs.m

```
% This is the "props_file" to load nominal beam data.
% This program is called by BeamA_Prompt_crs to provide beam properties
% in order to build Beam A.

% Outputs
% -------
% depth
% width
% E
% rho
% total_length
% num_elements
% nominal_EI
% nominal_area
% nominal_density

% Following are actual measurements from experimental set-up cantilever
% beam.
        depth = .504;% in z-dir (inches)
        width = 1.506; % in y-dir (inches)
        E = 10e6;
   %E = 1.65e6; % lbf/sec^2-in (10e6-ksi)
   %(1bf/in^2 = 6894.76Pa)-> E(lbf/in^2) = ()Pa/6894.76
        rho =0.110460934; %0.098;% lbf/in^3

   % T6 temper alloys require a 35-ksi tensile strength, 30-ksi yield
   % strength and a 10e6-ksi elastic modulus. Alloy 6061-T6 has 1.0
   % pct magnesium, 0.6 pct silicon, 0.3 pct copper and 0.2 pct chromium.
   % It has a 45-ksi tensile strength and 35-ksi yield strength.1 The
   % machinability of aluminum alloys are high (300) compared to titanium
   % (40). Aluminum alloys can easily be bent and provide easy loading and
   % unloading of parts. Also, aluminum is a highly conductive metal
   % compared to titanium.

% all measurement of distance are in inches
        total_length    = 42;
        num_elements    = 10;
        nominal_EI      = (width * depth^3 / 12) * E;
        nominal_area    = depth * width;% in^2
        nominal_density = rho;% lbf/in^3

% ********************  END BeamProperties_crs.m  ************************
```

BeamSensitivity_crs.m

```
% Revision history:
% ~~~~~~~~ ~~~~~~~~
%
%  Ver. 1.0: 4/4/95  Basic frequency sensitivities
%  Updated: Spring 2004  Constance R S Fernandez

% ************************************************************************
%
% Program Description:
% ~~~~~~~ ~~~~~~~~~~~~
%
%  This program calculates mode frequency sensitivities as given by the
%  equation,
%
%        ¶w^2             ¶[k]      ¶[m]
%        --    =  {P}' * [  ---- - w^2 --- ] * {P}
%        ¶DV              ¶DV       ¶DV
%
%    where:  w  = natural frequency
%           {P} = associated mode shape
%            DV = design variable
%
%  The right side of the above equation is considered the addition of
%  the sensitivity matrix wrt EI and the sensitivity matrix wrt mass.
%
%  The program calculates the stiffness and mass matrix partials by finite
%  differences. That is, for example, the [k] matrix is assembled twice,
%  once in for the nominal beam parameters, and a second matrix is
%  assembled based on a small perturbation of element mass and/or EI.
%
%  This program makes use of the beam data created by the program
%  "Build2Beams.m."
%  1) Resets BeamX mass and EI data to be the same as BeamA data.
%  2) Enters a loop to create a sensitivity matrix (T)
%    In loop
%    a)  A small perturbation of mass is added to BeamX on ele.1.
%    b)  The mass matrix is assembled for this mass-perturbed beam,
%        and the mass matrix partial derivative is calculated as
%
%               ¶[m]       [m_perturb] - [m_baseline]
%               ---    =   -------------------------
%               ¶DV                ¶DV
%
%    c)  The first column of Sensitivity matrix is calculated using:
```

```
%
%          sens_mass = [phi_base]'*(-lam_base)*m_delta*[phi_base]
%
%    (Note: A column of T corresponds to the respective element on beam.)
%
%    d) T loop starts again but with the small change on element 2.
%    e) Difference calculated and second column of T is calculated.
%
%   3) loop continues until all columns of T are calculated, corresponding
%    to a small change added to the respective element.
%
%   4) The procedure is identical for stiffness sensitivity.
%     Except:
%     sens_EI = [phi_base]'*k_delta*[phi_base]
%
%   5)Combine the two T's into one complete sensitivity matrix :
%         T_sens = [sens_mass,sens_EI].
%    In words, the first set of columns (equal to number of elements)of
%    the combined matrix is the sensitivity mass wrt mass changes and
%    the last half is wrt EI changes.
%
% ********************************************************************

% Inputs Needed:
% --------------
% mass_lbls
% EI_lbls
% element_mass
% element_EI
% num_rbm
% num_elements
% lamx (experimently measured nat freq of beam)

% Programs called
% --------------
% Assemble2Beams_crs;
% BoundaryConditions_crs;
% fmodes
% fOset_from_Aset

% Outputs:
% --------------
% num_modes
% lam_base
% phi_base
% sens_mass
```

```
% sens_EI
% T_sens
% dv_cal
% freq_base
% vect_lam

%  Start code:
%  ~~~~~ ~~~~~
format long;


% ************************************************************************
% ************************* INITIALIZATION ***************************
% ************************************************************************

mass_change = 1; % Percent mass change for sensitivity calculation.
EI_change = 1;   % Percent EI change for sensitivity calculation.

element_mass_orig =  element_mass;  % Copy properties to retain them.
element_EI_orig   =  element_EI;

% Prompt for number of mode frequencies to generate sensitivities for:
%
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%num_modes = input(' Enter number of modes for sensitivity calculations>> ');
num_modes = 19; % 19 is maximum number of modes in a 10 element cantilever
            % beam.  This is hard coded for quicker run of simulation.

start_mode = num_rbm + 1;   % Skip the rigid body modes.

% ************************************************************************
% ************** MASS SENSITIVITY CALCULATION LOOP **************
% ************************************************************************

sens_mass = 0; % initialize Mass based sensitivity matrix

if mass_lbls ~= 0; % From Beam_XPrompt as user inputs

   for icnt_dv =  1:num_elements; % loop to create sensitivity matrix

      %   Resetting BeamX properties to BeamA properties
      element_mass(:,2)  =  element_mass(:,1);

      %   each element, one at a time will have a change in mass
      element_mass(icnt_dv,2) = element_mass(icnt_dv,2) * ...
```

```
      (1 + mass_change/100);

   Assemble2Beams_crs;   % Run script to assemble beams.

   BoundaryConditions_crs;  % Apply boundary conditions.

   [lam_base, phi_base] = fModes(ka,ma);

   % ma is the basebeam without changes,
   % mx is beam with small change in mass added (mass_change)

   %    Form mass derivative matrices:
   m_delta = (mx - ma)/(mass_change/100);
   % converts precent change into decimal amount


   %    Mode freq sens loop:
   end_mode = start_mode + (num_modes - 1);
   row_num = 0; % initialize loop
   for icnt_modes = start_mode:end_mode;
      row_num = row_num +1;
      sens_mass(row_num,icnt_dv) = phi_base(:,icnt_modes)' *...
         (-lam_base(icnt_modes) * m_delta ) *...
         phi_base(:,icnt_modes);
      % definition of mass sensitivity matrix
   end; % for "icnt_modes" inner loop

end;  % End "for icnt_dv" outer loop for sensitivity calculations

end;    % End "if mass_lbls ~= 0"


% ************************************************************************
% ************** EI SENSITIVITY CALCULATION LOOP ******************
% ************************************************************************

sens_EI = 0;  % initialize EI based sensitivity matrix
if EI_lbls ~= 0; % From Beam_XPrompt as user inputs

   for icnt_dv =  1:num_elements; % loop to create sensitivity matrix

      %   Resetting BeamX properties to BeamA properties
      element_EI(:,2)  =  element_EI(:,1);

      %   each element, one at a time will have a change in EI
      element_EI(icnt_dv,2) = element_EI(icnt_dv,2) * ...
```

```
      (1 + (EI_change/100) );

    Assemble2Beams_crs;   % Run script to assemble beams.

    BoundaryConditions_crs;  % Apply boundary conditions.

    [lam_base, phi_base] = fModes(ka,ma);


    % ka is the basebeam without changes,
    % kx is beam with small sensitivity added

    %    Form EI derivative matrices:
    k_delta = (kx - ka)/(EI_change/100);
    % converts precent change to decimal value

    %    Mode freq sens loop:
    end_mode = start_mode + (num_modes - 1);
    row_num = 0;

    for icnt_modes = start_mode:end_mode;
      row_num = row_num +1;
      sens_EI(row_num,icnt_dv) = phi_base(:,icnt_modes)' *...
        k_delta * phi_base(:,icnt_modes);
      %definition of EI sensitivity matrix
    end; % for "icnt_modes" inner loop

  end; % End "for icnt_dv" outer loop for sensitivity calculations

end;    % if EI_lbls = 0;


% Copy element EI and mass properties back into arrays:
element_EI   = element_EI_orig;
element_mass = element_mass_orig;

% cleans up workspace by clears all unimportant parameters
clear element_EI_orig element_mass_orig end_mode start_mode
clear row_num icnt_modes EI_change k_delta mass_change m_delta
clear ka kx ma mx icnt_dv

% Assembles complete sensitivity matrix
if sens_mass == 0& sens_EI ~=0;

  T_sens = sens_EI; % resultant sensitivity matrix is equal
  % to EI sensitivity matrix with only EI changes if no inputs
```

```
   % are given for mass changes

elseif sens_mass ~= 0 & sens_EI == 0;
   T_sens = sens_mass; % resultant sensitivity matrix is equal
   % to Mass sensitivity matrix with only mass changes if no inputs
   % are given for EI changes

else
   T_sens = cat(2, sens_mass,sens_EI);% else the resultant sensitivity
   %matrix is teh combination of mass sensitivity matrix in first set
   %of columns and EI sensitivity matrix in the last set of columns

end

freqx = sqrt(lamx)/2/pi;
freq_base = sqrt(lam_base)/2/pi;
% NOTE: % lamx = experiment measured natural freq of beam
vect_lam = (lamx(1:num_modes)-lam_base(1:num_modes));

% **********  END BeamSensitivity_crs.m  **********
```

BeamSensitivityOSET_crs.m

```
% Revision history:
% ~~~~~~~~ ~~~~~~~~
%
%  Ver. 1.0: 4/4/95  Basic frequency sensitivities
%  Updated: Spring 2004  Constance R S Fernandez to create resulting
%  sensitivity matrix using lam vector of multiple ABC systems

% ********************************************************************
%
% Program Description:
% ~~~~~~~ ~~~~~~~~~~~~
%
%  This program calculates mode frequency sensitivities as given by the
%  equation,
%
%        ¶w^2              ¶[k]      ¶[m]
%        --    =  {P}' * [  ---- - w^2 --- ] * {P}
%        ¶DV               ¶DV       ¶DV
%
%    where:  w  = natural frequency
%          {P} = associated mode shape
%           DV = design variable
%
%  The right side of the above equation is considered the addition of
%  the sensitivity matrix wrt mass and/or EI.
%
%  The program calculates the stiffness and mass matrix partials by finite
%  differences. That is, for example, the [k] matrix is assembled twice,
%  once in for the nominal beam parameters, and a second matrix is
%  assembled based on a small perturbation of element mass and/or EI.
%
%  This program makes use of the beam data created by the program
%  "Build2Beams.m."
%  1) Resets BeamX mass and EI data to be the same as BeamA data.
%  2) Enters a loop to create a sensitivity matrix (T)
%    In loop
%    a)  A small perturbation of mass is added to BeamX on ele.1.
%    b)  The mass matrix is assembled for this mass-perturbed beam,
%        and the mass matrix partial derivative is calculated as
%
%               ¶[m]       [m_perturb] - [m_baseline]
%               ---    =   ------------------------
%               ¶DV                ¶DV
%
```

```
%    c)  The first column of Sensitivity matrix is calculated using:
%
%          sens_mass = [phi_base]'*(-lam_base)*m_delta*[phi_base]
%
%    (Note: A column of T corresponds to the respective element on beam.)
%
%    d) T loop starts again but with the small change on element 2.
%    e) Difference calculated and second column of T is calculated.
%
%   3) loop continues until all columns of T are calculated, corresponding
%    to a small change added to the respective element.
%
%   4) The procedure is identical for stiffness sensitivity.
%     Except:
%     sens_EI = [phi_base]'*k_delta*[phi_base]
%
%   5)Combine the two T's into one complete sensitivity matrix :
%         T_sens = [sens_mass,sens_EI].
%    In other words, the first set of columns (equal to number of elements)of
%    the combined matrix is the sensitivity with respect to (wrt) mass
%    changes and the last half is wrt EI changes.
%
%
% ************************************************************************

% Inputs Needed:
% --------------
% mass_lbls
% EI_lbls
% element_mass
% element_EI
% num_rbm
% mx_beam
% kx_beam

% Programs called
% --------------
% Assemble2Beams_crs;
% BoundaryConditions_crs;
% fmodes
% fOset_from_Aset
% displacmentPlot_OSET

% Outputs:
% --------------
% num_modesO
```

```
% sens_massO, sens_EIO
% num_rbmOSET
% T_sens_oset
% dispX_tot, dispA_tot
% accel_plot
% oset_choice
% accelometer
% BC, BCose, BCOSET
% remaindof
% ASETtot, OSET, OSETtot
% inct_sens
% mass_change, EI_change
% phiXPLOT, phiAPLOT
% maO_base, ka0_base
% mxO, kxO
% plotmx, plotkx
% lamaOSET, phiaOSET
% lamxOSET, phizOSET
% lamxplot, phixplot
% faO
% m_deltaO, k_deltaO

%  Start code:
%  ~~~~~ ~~~~~

% ************************************************************************
% ************************* INITIALIZATION ***************************
% ************************************************************************
T_sens_oset = [];
vect_lam_oset = [];
dispX_tot  = [];
dispA_tot  = [];
inct_sens = 0;
icnt_oset = 0;
BCOSET = zeros(ndof, ndof);
OSETtot = zeros(ndof, ndof);
ASETtot = zeros(ndof, ndof);

oset_choice = 'n';
ndof % prints ndof to screen for user's reference
accelometer = [3 5 7 9 11 13 15 17 19 21]; % use this line for convenience
% in quicker calculation loops or use the next 2 lines.
%accelometer = input('On what nodes are the accel. located','s'); %requests
% user to input locations of accelometers
%accelometer = eval(['[',accelometer,']']); % converts string to vector of
% labels
```

```
% saved for plotting accelometers in correct position.
accel_plot = floor(accelometer/2)+1;
% graphical representation of accelometer locations

oset_choice = input('  Do you want to use ASET and OSET (y/n)? ','s');
% user can choose to use ASET and OSET calculations
while oset_choice ~= 'n';
   icnt_oset = icnt_oset +1;
   disp(' locations of Accel.') % displays "location of Accel" on screen
   accelometer % displays the vector of location of Accel for user's
   % reference in choosing which DOF are to be pinned
   disp(' ')
   disp('   Enter pinned DOF label(s)')
   disp('   Use MATLAB vector format> 1 3 5:7 9  ')
   pinned = input('   >> ','s');
   pinned = eval(['[',pinned,']']); % Converts string to vector of labels
   BC(icnt_oset,:) = [1,2,pinned]; % Boundry conditions for cantilever beam,
   % change line if different beam is used

   BCoset = fOset_from_Aset(ndof, BC(icnt_oset,:)); % gets OSET from ASET

   BCOSET(icnt_oset, 1:length(BCoset)) = BCoset; % copies OSET into another
   % vector to be used

   % loop to find remaining accelometers.
   for icnt = 1 : length(pinned);
      remain(icnt,:) = find(pinned(icnt) == accelometer);
   end

   remaindof = fOset_from_Aset((length(accelometer)), remain);
   % remaining unpinned DOF

   % remaining accelometers
   aset = accelometer(remaindof);
   ASETtot(icnt_oset, 1:length(aset)) = aset;

   % omitted oset, i.e. pinned accelometers
   OSET = fOset_from_Aset(ndof, aset);
   OSETtot(icnt_oset, 1:length(OSET)) = OSET; % copies OSET into another
   % vector to be used in======

   inct_sens = inct_sens+1;

   mass_change = 1;  % Percentage mass change for sensitivity calculation.
   EI_change = 1;    % Percentage EI change for sensitivity calculation.
```

```
   element_mass_orig =  element_mass;  % Copy properties over to retain them.
   element_EI_orig   =  element_EI;    %


   % Prompt for number of mode frequencies for which to generate sensitivities
   % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   % use the following three lines for user's input for number of modal
   % frequencies for which to generate sensitivity or use the fourth line
   % which uses the maximum number of modes for the defined system

   %disp ('max number of modes  ')
   %length(oset)
   %num_modesO = input('  Enter number of elastic modes for sensitivity calculations>>
');
   num_modesO = ndof - length(BC(icnt_oset,:));%max number of modes in system
   start_mode = num_rbm + 1;   % Skip the rigid body modes.

   % Initializes two vectors to be used in ploting program.
   phiXPLOT = zeros(ndof,num_modesO); %cantilever beam
   phiAPLOT = zeros(ndof,num_modesO); %cantilever beam


   % ***********************************************************************
   % ************** MASS SENSITIVITY CALCULATION LOOP **************
   % ***********************************************************************

   sens_massO = 0;
   if mass_lbls ~= 0; % from Beam_XPrompt as user inputs

      for icnt_dv =  1:num_elements;  % loop to create sensitivity matrix

         %   Resetting BeamX properties to BeamA properties.
         element_mass(:,2)  =  element_mass(:,1);

         %   each element, one at a time will have a change in mass
         element_mass(icnt_dv,2) = element_mass(icnt_dv,2) * ...
            (1 + mass_change/100);

         Assemble2Beams_crs;   % Run script to assemble beams.

         % Articial Boundry Conditions
         maO_base = ma(BCoset, BCoset);
         % new mass matrix of the system defined by OSET
         mxO = mx(BCoset, BCoset);
```

```
plotmx = mx_beam(BCoset, BCoset);
% resulting mass matrix with ABC used in plotting
plotkx = kx_beam(BCoset, BCoset);
% resulting stiffness matrix with ABC used in plotting

kaO_base = ka(BCoset, BCoset);
kxO = kx(BCoset, BCoset);

% lam (natural freq^2, rad^2/sec^2), phi (mode shapes)
[lamaOSET,phiaOSET] = fModes(kaO_base,maO_base);
% natural freq of new artifically bounded base system
[lamxOSET,phixOSET] = fModes(kxO,mxO);
% natural freq/ modes of new artifically bounded system with
% either EI or mass changes added similiar to orginial calculations
[lamxplot,phixplot] = fModes(plotkx,plotmx);
%resulting lam & phi of ABC system used in plotting

% Mode shapes
if pinned == 3 % the next DOF acts a little different
   % because of the location on beam thus it was easier
   % to program it seperately
   phiAPLOT(2:ndof-2, :) = phiaOSET(1:ndof-3, :);
   phiXPLOT(2:ndof-2, :) = phixplot(1:ndof-3, :);
else
   phiAPLOT(1:pinned-3, :) = phiaOSET(1:pinned-3, :);
   phiAPLOT(pinned-1:ndof-2,:) = phiaOSET(pinned-2:ndof-3,:);
   phiXPLOT(1:pinned-3, :) = phixplot(1:pinned-3, :);
   phiXPLOT(pinned-1:ndof-2,:) = phixplot(pinned-2:ndof-3,:);
end % "if pinned ==3"

num_rbmOSET = length(find(lamaOSET < 1));
% find number of rigid bodies in new ABC system
start_mode = num_rbmOSET + 1;   % Skip the rigid body modes.

faO = sqrt(lamaOSET)/(2*pi); %natural freq of the ABC in Hz

%    Form mass derivative matrices:
m_deltaO = (mxO - maO_base)/(mass_change/100);
% converts percent change to decimal value
% NOTE: mass_change needs to be the same change used in
% orginial mass sensitivity matrix calculation

%    Mode freq sens loop:
end_mode = start_mode + (num_modesO - 1);
row_num = 0;
```

```
      for icnt_modes = start_mode:end_mode;
         row_num = row_num +1;
         sens_massO(row_num,icnt_dv) = phiaOSET(:,icnt_modes)' *...
            (-lamaOSET(icnt_modes) * m_deltaO) *...
            phiaOSET(:,icnt_modes);
      end; % end "for icnt_modes" inner loop

   end;  % End "for icnt_dv" outer loop for sensitivity calculations

   displacmentPlot_OSET % calls a program

   % This program assembles the beam displacement vector for
   % sens_beam(dispX) and base beam(dispA) under ABC

   if  dispX_tot  == 0; % when the vector is empty
      dispX_tot = disp1; % displacement vector used in plotting
      dispA_tot = disp1a;
   else
      dispX_tot = cat(1,dispX_tot,disp1);
      dispA_tot = cat(1, dispA_tot,disp1a);
   end % end "if dispX_tot  == 0"

end;    % end "if mass_lbls = 0"


% ********************************************************************
% ************** EI SENSITIVITY CALCULATION LOOP *******************
% ********************************************************************
sens_EIO = 0;
if EI_lbls ~= 0;

   for icnt_dv =  1:num_elements; % loop to create sensitivity matrix

      %   Resetting BeamX properties to BeamA properties
      element_EI(:,2)  =  element_EI(:,1);

      %   each element, one at a time will have a change in EI
      element_EI(icnt_dv,2) = element_EI(icnt_dv,2) * ...
         (1 + (EI_change/100) );

      Assemble2Beams_crs;   % Run script to assemble beams.

      % Artifical Boundary Conditions
      maO_base = ma(BCoset, BCoset);
      mxO = mx(BCoset, BCoset);
```

```
plotmx = mx_beam(BCoset, BCoset);
plotkx = kx_beam(BCoset, BCoset);

kaO_base = ka(BCoset, BCoset);
kxO = kx(BCoset, BCoset);

% lam (natural freq^2, rad^2/sec^2), phi (mode shapes)
[lamaOSET,phiaOSET] = fModes(kaO_base,maO_base);
[lamxOSET,phixOSET] = fModes(kxO,mxO); %sens info
[lamxplot,phixplot] = fModes(plotkx,plotmx); %plot info


if pinned == 3
   phiAPLOT(2:ndof-2, :) = phiaOSET(1:ndof-3, :);
   phiXPLOT(2:ndof-2, :) = phixplot(1:ndof-3, :);
else
   phiAPLOT(1:pinned-3, :) = phiaOSET(1:pinned-3, :);
   phiAPLOT(pinned-1:ndof-2,:) = phiaOSET(pinned-2:ndof-3,:);
   phiXPLOT(1:pinned-3, :) = phixplot(1:pinned-3, :);
   phiXPLOT(pinned-1:ndof-2,:) = phixplot(pinned-2:ndof-3,:);
end

num_rbmOSET = length(find(lamaOSET < 1));
start_mode = num_rbmOSET + 1;   % Skip the rigid body modes.

faO = sqrt(lamaOSET)/(2*pi); %natural freq of the ABC, Hz
%    Form EI derivative matrices:
k_deltaO = (kxO - kaO_base)/(EI_change/100);   % in %/100

%    Mode freq sens loop:
end_mode = start_mode + (num_modesO - 1);
row_num = 0;

for icnt_modes = start_mode:end_mode;
   row_num = row_num +1;
   sens_EIO(row_num,icnt_dv) = phiaOSET(:,icnt_modes)' *...
      k_deltaO * phiaOSET(:,icnt_modes);
end; %end "for icnt_modes"


end; % End "for icnt_dv" outer loop for sensitivity calculations


displacmentPlot_OSET % calls a program

% This program assembles the beam displacement vector for
```

```
   % sens_beam(dispX) and base beam(dispA) under ABC

   if  dispX_tot  == 0;
      dispX_tot = disp1;
      dispA_tot = disp1a;
   else
      dispX_tot = cat(1,dispX_tot,disp1);
      dispA_tot = cat(1, dispA_tot,disp1a);
   end % end "if dispX_tot  == []"

end;    % end "if EI_lbls = []"


% Copy element EI and mass properties back into arrays:
element_EI   = element_EI_orig;
element_mass = element_mass_orig;

clear element_EI_orig element_mass_orig end_mode start_mode
clear row_num icnt_dv icnt_modes lbls num_EI_dv num_mass_dv

% assemble of total sensitivity matrix for ABC System
if sens_massO == 0 & sens_EIO ~=0;

   T_sensO = sens_EIO; % no changes in mass

elseif sens_massO ~= 0 & sens_EIO == 0;
   T_sensO = sens_massO; % no changes in EI

else
   T_sensO = cat(2, sens_massO,sens_EIO);
   % changes in both mass and EI

end %end "if sens_massO == 0 & sens_EIO ~=0"

% Builds complete sens matrix of all ABC systems
if  T_sens_oset == 0;% when matrix is originally empty

   T_sens_oset = T_sensO;

else % after the matrix has some values
   T_sens_oset = cat(1,T_sens_oset, T_sensO);
end %end "if  T_sens_oset == []"

lamaOSET = lamaOSET(find(lamaOSET > 1)); %skips Rigid body modes
vect_lamO = lamaOSET(1:num_modesO); % nat freq of base beam under ABC
```

```
   %builds a vector of natural freq of ABC systems
   if vect_lam_oset == 0;% when matrix is originally empty


      vect_lam_oset = vect_lamO;
   else% after the matrix has some values
      vect_lam_oset = cat(1, vect_lam_oset, vect_lamO);

   end %end "if vect_lam_oset == 0"
   disp(' ')
   oset_choice = input('  Another cycle of ASET and OSET (y/n)? ','s');
   % loop runs until "n" is inputed creating sens matrix & lam vector
   % for all ABC systems
   disp(' ')

end; % end "while oset_choice ~='n'"

% ********************  BeamSensitivityOSET_crs.m  ************************
```

BeamX_Prompt.m

```
% Written By Prof Gordis

% Inputs needed
% -----------------
% element_EI
% element_mass

% Outputs
% -----------------
% i_lbls
% change_mass, change_EI
% new_lbls
% updated element_mass in column 2
% updated element_EI in column 2
% mass_lbls - index for sensitivy matrix
% EI_lbls - index for sensitity matrix
% dv_EI
% dv_mass
% dv_tot

% _____________________________________________________________________
%
%                  Prompt User for BeamX Modification Data
% _____________________________________________________________________


disp(' ');disp(' ');
disp(' Modify nominal physical properties for second beam')
disp(' ~~~~~~ ~~~~~~~~ ~~~~~~~~~ ~~~~~~~~~~~ ~~~ ~~~~~~ ~~~~')

% Adjust mass values for second beam:
i_lbls = 0;

dv_mass =[];
mass_lbls = [];
change_mass = 'n';
change_mass = input('  Modify single/range element mass values (y/n)? ','s');
% user input
while change_mass ~= 'n';

   disp('   Enter element label(s) for mass modification')
   disp('   Use MATLAB vector format> 1 3 5:7 9  ')
   new_lbls = input('   >> ','s');
   new_lbls = eval(['[',new_lbls,']']); % Converts string to vector of labels
```

```
   i_lbls = i_lbls + 1;

   % CRS addition

   mass_lbls(i_lbls,1:length(new_lbls))= new_lbls; % index for sensitivity matrix

   disp('  Enter mass change for element range')
   mass_change = input('  Enter percentage mass change (+/- %) ');

   dv_mass(i_lbls,1) = mass_change/100; % vector of mass changes to second beam
(BeamX)

   element_mass(new_lbls,2) = element_mass(new_lbls,2)+...
      (mass_change/100) * element_mass(new_lbls,2);

   disp(' ')
   change_mass = input('  Modify another element mass value (y/n)? ','s');
   disp(' ')

end; % end while

% Adjust EI values for second beam:
i_lbls = 0;

change_EI = 'n';
dv_EI =[];
EI_lbls = [];

disp(' ')
change_EI = input('  Modify single/range element EI values (y/n)? ','s');
while change_EI ~= 'n';

   disp('   Enter element label(s) for EI modification')
   disp('   Use MATLAB vector format> 1 3 5:7 9  ')
   new_lbls = input('   >> ','s');
   new_lbls = eval(['[',new_lbls,']']); % Converts string to vector of labels

   i_lbls = i_lbls + 1;

   EI_lbls(i_lbls,1:length(new_lbls)) = new_lbls; % index for sensitivity matrix

   disp('  Enter EI change for element range')
   EI_change = input('  Enter percentage EI change (+/- %) ');

   dv_EI(i_lbls,1) = EI_change/100; % vector of EI changes on second Beam
```

```
   element_EI(new_lbls,2) = element_EI(new_lbls,2)+...
      (EI_change/100) * element_EI(new_lbls,2);

   disp(' ')
   change_EI = input('  Modify another element EI value (y/n)? ','s');
   disp(' ')
end; % end while

dv_tot = [dv_mass;dv_EI];
% vector of total changes to second beam (BeamX) but not location.

% End BeamX_Prompt.m
clear EI_change mass_change

% ********************  END BeamX_Prompt.m  ************************
```

BoundaryConditions_crs.m

```
% Written by Prof Gordis

% This script prompts the user boundary condition information
% The script creates a vector of DOF (with respect to the unrestrained
% structure) and then extracts the rows and columns of the complementary
% DOF.

%  Script defines vector "free_dof_set" containing
%  list of unrestrained dof.

% The boundary conditions are applied in this script.

% Inputs needed:
% -------------
% ndof
% ka, ma, kx, mx

% Outputs:
% -------------
% free_dof_set
% updated ka, ma, kx, mx
% icnt_dof
% add_dof
% bc_node
% bc_coord
% bc_DOF
% bc_boolean
% all_dofs
% restraint_switch
% ************************************************
% Start code:


if exist('free_dof_set')==0;    %  Build free_dof_set vector

   disp(' Select a boundary condition set:')
   disp('    (1) Clamped-free')
   disp('    (2) Clamped-Clamped')
   disp('    (3) Pinned-Pinned')
   disp('    (4) User-Defined')
   disp('    (5) Free-Free')

   BC_Choice = input(' >> Enter choice: ');
```

```
if BC_Choice == 1;      % Clamped-free _____________________________
   free_dof_set = [3:ndof];
   restraint_switch = 'y';

elseif BC_Choice == 2;  % Clamped-Clamped _________________________

   free_dof_set = [3:ndof-2];
   restraint_switch = 'y';

elseif BC_Choice == 3;  % Pinned-Pinned ___________________________

   free_dof_set = [2:ndof-2  ndof];
   restraint_switch = 'y';

elseif BC_Choice == 4;  % User-Defined ____________________________

   icnt_dof = 0;
   add_dof = 'y';
   while add_dof == 'y';

      bc_node = input(' Node number for restraint ? "0" to end: ');

      if bc_node == 0;
         break
      end;
      bc_coord = input(' Translation or Rotation ? (t/r) ','s');

      icnt_dof = icnt_dof + 1;
      if bc_coord == 't';
         bc_DOF(icnt_dof) = 2 * bc_node - 1;
      elseif bc_coord == 'r';
         bc_DOF(icnt_dof) = 2 * bc_node;
      end;   % End if-else block

   end;  % End while add_dof

   bc_boolean = ones(ndof,1);                % [1 1 1 ... icnt_dof]
   bc_boolean(bc_DOF) = zeros(length(bc_DOF),1);% Put zeros in restrained dof
   all_dofs = [1:ndof];                      % List of all dof
   free_dof_set = all_dofs(logical(bc_boolean));% Extract free dof
   restraint_switch = 'y';

elseif BC_Choice == 5; % Free-free beam _____________________________

   free_dof_set = [1:ndof];
   restraint_switch = 'n';
```

```
   end;                % End if-elseif choice block _____________________

end;       % End exist block

ka = ka(free_dof_set,free_dof_set);
ma = ma(free_dof_set,free_dof_set);
kx = kx(free_dof_set,free_dof_set);
mx = mx(free_dof_set,free_dof_set);

% ****************END BoundaryConditions_crs.m  ************************
```

Build2Beams_crs.m

```
clear
clc
% Revision history:
% ~~~~~~~~ ~~~~~~~~
%
%  Ver. 1.0: 9/22/94  Basic two beam assembly
%       2.0:          Added multi-element changes
%       2.1  3/28/95  Added read/write to file, rebuild capability
%       2.2  3/29/95  Added lumped mass additions
%            3/10/04  Added Sensitivity matrices, error prediction, plots

%*****************************************************************************
%
% Program Description:
% ~~~~~~~ ~~~~~~~~~~~~~
%
%  This program assembles the mass and stiffness matrices for 2 free-free
%  beams, referred to as "BeamA" (analysis) and "BeamX" (experimental). The
%  program can be run in several modes:
%
%  "Build" mode:
%   ~~~~~  ~~~~~
%  The user provides baseline data for BeamA, assumed to be a
%  homogeneous, uniform beam. Data provided:
%
%    (1) Beam length
%    (2) Number of elements
%    (3) Nominal EI
%    (4) Nominal cross-sectional area
%    (5) Nominal weight density
%
%  The program then prompts the user for instructions on how to modify
%  "BeamA" data to arrive at "BeamX" data. The user can modify element
%  masses, and/or element EI values. The modification can be applied to
%  either a single element, or range of elements, e.g.
%
%     Modify single/range element mass values (y/n)?  y
%
%  If "y" is entered, the user enters the number of the element for mass
%  adjustment:
%
%       Enter element label(s) for mass modification:  1
%       Use MATLAB vector format> 1 3 5:7 9
%
```

```
%              Enter percentage mass change (+/- %)
%
% The user is prompted to modify another element or range of elements:
%
%      Modify another element mass value (y/n)?  y
%
%  This process continues until the user enters an "n" for no change.
%  This entire process can then be repeated for EI adjustment.
%
%  The program saves the beam definition data in a binary (.mat) file
%  "beamdata" at the end of execution.
%
%  The program can also be run in "Read" mode by entering an "r" at
%  the initial prompt.
%
%
% Script Execution Path:
% ~~~~~~ ~~~~~~~~~~ ~~~~~
%
%
%
%    Build2Beams_crs.m         -- User executes this program.
%    BeamA_Prompt_crs.m        -- Prompts User for BeamA nominal beam data
%    BeamX_Prompt_crs.m        -- Prompts User for BeamX modification beam data
%    Assemble2Beams_crs.m      -- Called by Build2Beams, builds [ka] [ma] [kx]
%                                 [mx], plots freqs.
%    Lumpmass_crs.m            -- Prompts User for BeamX lumped mass addition
%    BoundaryConditions_crs.m  -- Prompts user for B.C.'s and applies them.
%    PlotBeamModes_crs.m       -- Calculate beam modes and plot frequencies
%
%    BeamSensitivity_crs.m     -- Calculate sensitivity matrix T-sens
%    BeamSensitivityOSET_crs.m -- Calculate sensitivity matrix using ABC
%    recorded_H_crs.m          -- Calculates the nat. freq of BeamX with ABC applied
%    AssembleSens_crs.m        -- Assembles the sens matrices and calculates errors.
%    ABCrunTHRU.m              -- Calculates the DV and cond number of matrix used
%    Saves data to "beamdata.mat"

%
%  Start code:
%  ~~~~~ ~~~~~
% ************************************************************************


disp('   Building 2 beams from scratch...')

BeamA_Prompt_crs;       %    Prompt for BeamA Data: run prompt script
```

```
BeamX_Prompt_crs;       %   Prompt for BeamX Modification Data:
Assemble2Beams_crs;     %   Run script to assemble mass and stiffness matrices
AddLumpmass_crs;        %   BeamX lumped mass vector construction and
%                           application

kx_beam = kx;  % saves the Beam X matrices without BC to be used later
mx_beam = mx;

BoundaryConditions_crs; %   Prompt for, and apply boundary conditions

kx_beamBC = kx;  % saves the Beam X matrices with BC to be used later
mx_beamBC = mx;
ka_beamBC = ka;  % saves the Beam A matrices with BC to be used later
ma_beamBC = ma;

PlotBeamModes_crs       %  Calculate beam modes and plot frequencies

BeamSensitivity_crs;    %  Calculate sensitivity matrix T-sens
BeamSensitivityOSET_crs;%  Calculate sensitivity matrix using ABC


recorded_H_crs;    % Calulates the nat. freq of BeamX with ABC applied
AssembleSens_crs;  % Assembles the sens matrices and calculates errors.
ABCrunTHRU_crs;    % Calculates the DV and cond number of matrix used
FOM_crs;           %Calculates the Figure of Merit for each prediction
plottingBARS_crs;      % Bar plots of predicted DV vs. true error



%                   Save Defining Parameters for Beams and plots
disp(' ...saving beam data to file')
save beamdata.mat


disp(' Build2Beams end.')
% _____________________

% ********************  END Build2Beams_crs.m  **********************
```

displacementPlot_crs.m

```
% Written By Constance Fernandez Spring 2004
% Program plots mode shapes (phi, lam) phi vs nodal position of beam. This
% program plots the displacement of BeamX and BeamA with actual location
% of errors used for visual comparison.

% Inputs
% --------
% kx_beamBC, ka_beamBC
% num_elements
% phix_plot, phia_plot
% EI_lbls, mass_lbls
% dv_mass, dv_EI

% Outputs
% ---------
% displ_plot
% displa_plot
% ypos
% jj, g

disp1_plot = zeros(.5*size(kx_beamBC,1),2*num_elements);
%initialize disp vector and provides the first zero of the vector.
disp1a_plot = zeros(.5*size(ka_beamBC,1),2*num_elements);
%initialize disp vector and provides the first zero of the vector.

for jj = 1:.5*size(ka_beamBC,1);
   disp1_plot(jj+1,:) = phix_plot(2*jj-1,:); % every other phi to give displacement at
sequential nodes
   disp1a_plot(jj+1,:) = phia_plot(2*jj-1,:);
end

% This loop normalizes the modes shapes to the tip modal displacement.
for g = 1:2*num_elements-1
   disp1_plot(:,g) = disp1_plot(:,g)/disp1_plot(num_elements+1,g);
   disp1a_plot(:,g) = disp1a_plot(:,g)/disp1a_plot(num_elements+1,g);
end

ypos = [1:1:num_elements+1]; % position in y direction along beam used
% to plot displacements at locations

% -----PLOTTING -------

figure(1) % plot displacements along BeamA and BeamX used in comparison
```

```
plot(ypos, disp1_plot(:,1),'-d', ypos, disp1_plot(:,2),'-s',ypos, disp1_plot(:,3),'-.x',ypos, ...
   disp1_plot(:,4),':+', ypos, disp1_plot(:,5),'--^',ypos, disp1a_plot(:,1),'-d', ypos, ...
   disp1a_plot(:,2),'-s',ypos, disp1a_plot(:,3),'-x',ypos, disp1a_plot(:,4),'-+', ypos, ...
   disp1a_plot(:,5),'-^'), grid on, hold on

legend('Mode 1 X','Mode 2 X','Mode 3 X','Mode 4 X','Mode 5 X','Mode 1 A',...
   'Mode 2 A','Mode 3 A','Mode 4 A','Mode 5 A', 4);

% plot stem in position of mass change and/or EI change
% mass change plotted in yellow, EI change plotted in cyan
if EI_lbls ~=[] & mass_lbls ~=[]% used if mass and EI errors were added
        stem(mass_lbls+.5, dv_mass,'y','filled'); hold on; stem(mass_lbls+.5, dv_mass,'k')
        EIplot = EI_lbls+10;hold on % mass error is plotted first then EI
        stem(EI_lbls+.5, dv_EI,'c','filled');hold on; stem(EI_lbls+.5, dv_EI,'k')
        % plots the location of inputted EI error

      elseif  mass_lbls ~=[] & EI_lbls ==[]% used with only mass error
        stem(mass_lbls+.5, dv_mass,'y','filled');hold on; stem(mass_lbls+.5, dv_mass,'k')

      else % used with only EI error
        stem(EI_lbls+.5, dv_EI,'c','filled');hold on; stem(EI_lbls+.5, dv_EI,'k')

end %"if EI_lbls ~=[] & mass_lbls ~=[]"  loop

% ----Plotting------

xlabel('position on Beam')
title('First five mode shape: Beam with error(X) vs Base Beam(A)')

% The following lines are used to input actual amount of error in legend *
% NOTE: problem with the display of digits

% Legend commands - used to print actual change in legend
% -----------------
%  legend('First','Second','Third','Fourth','Fifth',sprintf('%dper  -  Mass  ',  dv_mass*100),
sprintf('%dper - EI',dv_EI*100));
% dv = int2tr(dv_EI*100);
% dvT = sprintf('%sper - EI', dv);
% legend('First','Second','Third','Fourth','Fifth', dvT);


% ********************  END displacementPlot_crs.m  ************************
```

displacementPlot_OSET.m

```
% Written by Constance Fernandez Spring 2004

% This program plots the mode shapes (phi, lam) phi vs nodal position
% of beam when ABC are applied.

% Inputs
% -------
% plotkx
% kaO_base
% num_modesO
% phiXPLOT
% phiAPLOT
% pinned
% num_elements

% Outputs
% --------
%disp1, displa
%jj, g
% ypos

%------
disp1 = zeros(ceil(.5*size(plotkx,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.
disp1a = zeros(ceil(.5*size(kaO_base,1)),num_modesO);
%initilize disp vector and provides the first zero of the vector.

for jj = 1:ceil(.5*size(plotkx,1));
    disp1(jj+1,:) = phiXPLOT(2*jj-1,1:num_modesO);
    % every other phi to give displacement at sequential nodes
    disp1a(jj+1,:) = phiAPLOT(2*jj-1,1:num_modesO);
end

% This loop normalizes the modes shapes to the tip modal displacement.
if pinned == 21 % tip pinned is a special case,  no new calculations are needed
   disp1(:,:) = disp1(:,:);
   disp1a(:,:) = disp1a(:,:);
else
for g = 1:num_modesO
   disp1(:,g) = disp1(:,g)/disp1(num_elements+1,g);
   disp1a(:,g) = disp1a(:,g)/disp1a(num_elements+1,g);
end
end
```

```
ypos = [1:1:num_elements+1]; % Location of nodes used in plotting

% if mass_lbls ~= [];
%
%    for kk = 1:size(mass_lbls,1);
%       ff =0;
%       for JJ = 1:length(find(mass_lbls(kk,:)>0));
%          ff = ff+1;
%          posm(kk, 2*JJ-1) = mass_lbls(kk, ff);
%          posm(kk, 2*JJ) = mass_lbls(kk,ff)+1;
%       end
%    end
%
%    if kk == 1
%       posM = posm;
%
%    else
%
%       for uu = 1:kk-1;
%          posM = cat(2, posm(uu,:), posm(uu+1,:));
%       end
%
%    end
%    posM = sort(posM(find(posM>0)));
%    m = .5*ones(size(posM))+icnt_oset;
% end
%
% if EI_lbls ~= [];
%    for kk = 1:size(EI_lbls,1);
%       ff =0;
%       for JJ = 1:length(find(EI_lbls(kk,:)>0));
%          ff = ff+1;
%          pose(kk, 2*JJ-1) = EI_lbls(kk, ff);
%          pose(kk, 2*JJ) = EI_lbls(kk,ff)+1;
%       end
%    end
%
%    if kk == 1
%       posE= pose;
%
%    else
%
%       for uu = 1:kk-1;
%          posE = cat(2, pose(uu,:), pose(uu+1,:));
%       end
%    end
```

```
%   posE = sort(posE(find(posE>0)));
%   e = -.5*ones(size(posE))+icnt_oset;
% end

%pos = [mass_lbls, mass_lbls+1];

% figure(icnt_oset+10)
% subplot(2,1,1)
%
% % for ii = 1:num_modesO
% %     plot(ypos, disp1a(:,ii));
% %     hold on
% % end
% % plot(posM, m,'x' );grid on %posE, e
%
%
% plot(ypos, disp1(:,1),'-d', ypos, disp1(:,2),'-s',ypos, disp1(:,3),'-.x',...
%     ypos, disp1(:,4),':+', ypos, disp1(:,5),'--^',ypos, ...
%     disp1a(:,1),'-d', ypos, disp1a(:,2),'-s',ypos, disp1a(:,3),'-x',...
%     ypos, disp1a(:,4),'-+', ypos, disp1a(:,5),'-^'), grid on
%
% hold on
% plot(accel_plot, icnt_oset, 'kp',floor(pinned/2)+1,icnt_oset,'rs')
% % hold on
% % plot(posM, m,'-kV',posE, e,'-k>');grid on % posE, e
% % %
%
% %legend('accelerometer', 'pinned')
% %plot(floor(pinned/2)+1,icnt_oset,'rs')
% legend('First','Second','Third','Fourth','Fifth','First A','Second A',...
%     'Third A','Fourth A','Fifth A','accel')
% title(sprintf('pinned accel at Node # %d', floor(pinned/2)+1));
%
%
% **********  END displacementPlot_OSET.m  **********
```

fbeamkm.m

```
% function [kbeam,mbeam]=fbeamkm(l,ei,m)
% Provided by Prof Gordis

function [kbeam,mbeam]=fbeamkm(l,ei,m)
%
%
% This function returns the stiffness and mass matrices for
% a simple 2-node beam element.
%
% Note: m = rho * area * length = total element mass
%
% Reference: R.D. Cook, Concepts and Applications of F.E. Analysis

% Outputs
% ------
% kbeam, mbeam, i, j


kbeam=zeros(4,4);
mbeam=zeros(4,4);
%
kbeam(1,1)=12.0;
kbeam(1,2)=6.0*l;
kbeam(1,3)=-12.0;
kbeam(1,4)=6.0*l;
kbeam(2,2)=4.0*l^2;
kbeam(2,3)=-6.0*l;
kbeam(2,4)=2.0*l^2;
kbeam(3,3)=12.0;
kbeam(3,4)=-6.0*l;
kbeam(4,4)=4.0*l^2;
%
mbeam(1,1)=156.0;
mbeam(1,2)=22.0*l;
mbeam(1,3)=54.0;
mbeam(1,4)=-13.0*l;
mbeam(2,2)=4.0*l^2;
mbeam(2,3)=13.0*l;
mbeam(2,4)=-3.0*l^2;
mbeam(3,3)=156.0;
mbeam(3,4)=-22.0*l;
mbeam(4,4)=4.0*l^2;
%
for i=1:4;
```

```
   for j=i:4;
      kbeam(j,i)=kbeam(i,j);
      mbeam(j,i)=mbeam(i,j);
   end
end
%
kbeam=(ei/l^3)*kbeam;
mbeam=(m/420.0)*mbeam;
%
% end function beamkm

% ********************  END fbeamkm.m  ************************
```

fFRF.m

```
function [FRF] = fFRF(Wn, Zeta, phi, freq);
%  Provided by Prof Gordis
%
%       Usage: [FRF] = fFRF(Wn, Zeta, phi, freq);
%
%       Creates matrix whose ROWS are the FRF constructed from
%       modal parameters passed into the function.
%
%       Function uses all modes passed in, and will generate all FRF for unique
%       (symmetric) input-output pairs for all rows in [phi].
%       FRF are stored in "symmetric column storage" (See fSymmetricStore.m)
%
%       Wn:             Vector of natural frequencies (rad/sec)
%                       If Wn(i) < 0.1, rigid body mode is assumed.
%
%       Zeta:   Scalar, damping ratio applied to all modes
%
%       phi:    Mass normalized modal matrix.
%               Num rows = number of coordinates
%               Num cols = number of modes to be used.
%
%       freq:   Frequency (Hz). Row vector of sampling points for FRF evaluation.
%
% Inputs
% -------
% wn
% zeta
% phi
% freq

% Outputs
% --------
% ndof
% nmodes
% nsymcol
% FRF
% isymols
% omega
% j, irows, icols, imodes
% modeFRF

% _____________________________________________________________________
```

```
ndof                    = size(phi,1);
nmodes                  = size(phi,2);
nsymcol                 = ndof * (ndof + 1) / 2;                % Number of columns
FRF                             = zeros(nsymcol,length(freq));% Initialize matrix
modeFRF                         = zeros(1,length(freq));
isymcols                = 0;    % Will end up being nsymcol

omega = 2 * pi * freq;
j = sqrt(-1);

for irows = 1 : ndof;
        for icols = irows : ndof;
                isymcols = isymcols + 1;

        for imodes = 1 : nmodes;

                        if abs(Wn(imodes)) > 0.1;                % Then elastic mode

                                modeFRF = ((Wn(imodes)^2 - omega.^2) +
2*j*Zeta*Wn(imodes)*omega).^(-1);

                        elseif abs(Wn(imodes)) <= 0.1;          % Rigid body mode

                                modeFRF = -omega.^(-2);

                        end;    % End if abs(wn)

                        FRF(isymcols,:) = FRF(isymcols,:) +...
                                phi(irows,imodes) * phi(icols,imodes) * modeFRF;

                end;            % End icnt_modes

        end;            % End icnt_col_dof

end;            % End icnt_row_dof

% ********************  END fFRF.m  ************************
```

fModes.m

```
function [lam,phi]=fmodes(k,m,num_to_print);
% Provided by Prof Gordis
% This program prints to the screen natural modes of system (phi).
%
%       Usage: [lam,phi]=fmodes(k,m,num_to_print)
%
%   This function can be used with 1 to 3 arguments, as follows:
%
%       [lam,phi]=fmodes(a)   : Produces modes of [a] with no print of freqs in Hz.
%       [lam,phi]=fmodes(a,i)          : Produces modes of [a] with print of "i" freqs in
Hz.
%       [lam,phi]=fmodes(k,m)          : Produces modes of [m\k] with no print of freqs in
Hz.
%
%
%
%       This function returns a vector containing eigenvalues (rad/sec)^2
%       and a matrix containing the mass normalized mode shapes.
%       The mode information is sorted by frequency in ascending order.
%       If num_to_print > 0; tabular listing of num_to_print freqs in Hz is printed.
%       If num_to_print <= 0, no print.

% Inputs
% ------
% v, index, m, k

% Programs
% --------
% fNormalize

% Outputs
% --------
% phi
% num_to_print
% error
% e

%       ------------------------------------------------------------------------

if nargin == 1;                                                                 %
        [A] w/ no print request for freqs in Hz.

        %   v(1,:) = 1 normalization
        [v,d]=eig(k);
```

```
        [temp,indices] = sort(abs(diag(d)));
        lam = diag(d);
        lam = lam(indices);
        [phi]=fNormalize(v(:,indices), 'one');
        num_to_print = 0;

elseif nargin == 2 & size(m,1) == 1;            %       [A] w/ print request for freqs in Hz.

        %   v(1,:) = 1 normalization
        [v,d]=eig(k);
        [temp,indices] = sort(abs(diag(d)));
        lam = diag(d);
        lam = lam(indices);
        [phi]=fNormalize(v(:,indices), 'one');
        num_to_print = m;

elseif nargin == 2 & size(m,1) > 1;             %       [k],[m] w/ no print request for freqs in Hz.

        %   mass normalization
        [v,d]=eig(m\k);
        [lam,index]=sort(abs(diag(d)));
        [phi]=fNormalize(v(:,index),'mass',m);
        num_to_print = 0;

elseif nargin == 3 & size(k,1) > 1 & size(m,1) > 1;  %       [k],[m] w/ print request for freqs in Hz.

        %   mass normalization
        [v,d]=eig(m\k);
        [lam,index]=sort(abs(diag(d)));
        [phi]=fNormalize(v(:,index),'mass',m);

else

        num_to_print = -1;
        error('Error from fModes.m: Check input arguments.')

end

if num_to_print > length(k);
        num_to_print = length(k);
end

if nargin < 3 & rem(length(k),2)==0 & k(1:length(k)/2,1:length(k)/2) == zeros(length(k)/2,length(k)/2); % Have [A] matrix
```

```
        e = 1;              % Eigenvalues are wn
else
        e = 0.5;    % Eigenvalues are wn^2
end



if num_to_print > 0;

        disp('  '),disp('  ')
        disp('~~~~~~~~~~~~~~')
        disp('Freqs in Hz.:')
        disp((lam(1:num_to_print).^e)/2/pi)
        disp('~~~~~~~~~~~~~~')

end

% ********************  END fModes.m  ************************
```

fNormalize.m

```
function [phi] = fNormalize(phi,method,m);
%
% Usage: [phi] = fNormalize(phi,method,m);
%
%       phi: matrix whose columns are to be (independently) normalized.
%       method: String variable. The following choices are available:
%
%                       'mass'                  Mass normalization
%                       'inf'                   Infinity normalization
%                       'one'                   First element = 1
%                       'length'        Length = 1
%
%       m: matrix used for normalization, i.e. phi'*m*phi = eye
%
% _____________________________________________
%
        switch method

                case 'mass'                     % Mass normalization

%                       disp('mass normalization')
                        phi = phi * diag(sqrt(diag((phi' * m * phi).^(-1))));

                case 'inf'                      % Infinity normalization
%                       disp('inf normalization')
                        for icnt_cols = 1:size(phi,2);
                                phi(:,icnt_cols) = phi(:,icnt_cols)/norm(phi(:,icnt_cols),inf);
                        end

                case 'one'                      % First element = 1
%                       disp('one normalization')
                        phi = phi * diag((phi(1,:).^(-1))');

                case 'length'           % Length = 1
%                       disp('length normalization')
                        for icnt_cols = 1:size(phi,2);
                                phi(:,icnt_cols)                                        =
phi(:,icnt_cols)./norm(phi(:,icnt_cols),'fro');
                        end

        end

% ********************  END fNormalize.m  ************************
```

FOM_crs.m

```
%  This program calculates the Figure of Merit for the error predictions
%  calculated using the following sensitivity matrices
%     1) Base system only 5 modes (underdetermined)
%     2) ABC system 10 modes
%     3) Base system 5 modes + 5 modes from ABC system
% The last system is calculated 3 times.  Once for modes 1-5,
% another for modes 6-10, and again for modes 11-15.

% Written by Constance R S Fernandez, Spring 2004

% Inputs
% -------
% dv_tot
% dv_cal_ABC
% num_elements
% EI_lbls

% Outputs
% -------
% error
% x, xx, ix, is
% FOM2, FOM
% ABC5_norm, ABCten_norm
% ABCten_sq
% FOM_ABCten, FOM_ABC5
% ABC5_sq, PLUS_sq
% ABC5_sumNorm, PLUS_sumNorm
% FOM_PLUS,
% FOM_ABC5per, FOM_ABC10per, FOM_ABC_PLUSper
% ---------------------------------------------

error = sum(dv_tot); % total error added to beam(known quantity)
x = abs (dv_cal_ABC); % converts all errors to positive errors not to
% give false results of balancing out

x = sum(x,1)'; % sum of all errors
xx = x/error;% sum of errors divided by known error
for ix = 1:33
   FOM2 (ix) = abs((xx(ix)-1/xx(ix)));
end
FOM = (1-FOM2)*100;

%---Base System only---(underdetermined)
% ---ABC system only (only 5 modes)----
```

```
ABC5_norm = (dv_cal_ABC)/error; % normalized predicted error

% ---ABC system only using 10 modes instead of 5 modes ----

ABCten_norm = (dv_cal_ABCten)/error; % normalized predicted error

for ix = 1:size(dv_cal_ABCten,2);% Number of ABC system repeat for each ABC
system

   for is = 1:num_elements % repeat for each element

      ABCten_sq(is,1) = ABCten_norm(is, ix)^2; %square of each error in new vector
   end

   ABCten_sumNorm = sum(ABCten_sq); % sum of squared errors
   FOM_ABCten(ix,1) = (ABCten_norm(EI_lbls, ix).^2)/ABCten_sumNorm; % FOM
for each ABC system
end

% ---Base + 5 modes of ABC system ----

PLUS_norm = (dv_cal_BasePlus)/error; % normalized predicted error

% loop for relative error
for ix = 1:size(dv_cal_BasePlus,2);

      for is = 1:num_elements
         ABC5_sq(is,1) = ABC5_norm(is, ix)^2;

         PLUS_sq(is,1) = PLUS_norm(is, ix)^2;

      end
      ABC5_sumNorm = sum(ABC5_sq);
      FOM_ABC5(ix,1) = (ABC5_norm(EI_lbls, ix).^2)/ABC5_sumNorm;
      PLUS_sumNorm = sum(PLUS_sq);
      FOM_PLUS(ix,1) = (PLUS_norm(EI_lbls, ix).^2)/PLUS_sumNorm;
end

% converts the relative error to scale 1-100, 100 being the best prediction.
FOM_ABC5per = FOM_ABC5*100; % underdetermied
FOM_ABC10per = FOM_ABCten*100;% ABC system only using 10 modes
FOM_PLUSper = FOM_PLUS*100;% Base+5 modes of ABC system

% ********************   END FOM_crs.m  **********************
```

fOset_from_Aset.m

```
function [oset] = fOset_from_Aset(ndof,aset);
%
%  Usage: [oset] = fOset_from_Aset(ndof,aset);
%
% This function determines the complementary subset "oset"
% from a set [1:1:ndof] and the subset aset = [x x x ...].
%
%  ndof: Total number of DOF. Set is labeled "nset".
%  aset: Retained DOF (proper subset of [1:1:ndof])
%  oset: aset U oset = n
%
% Provided by Prof Gordis
% _____________________________________________________________

nset = [1:ndof];

for icnt = 1 : length(aset);
    indices(icnt) = find(nset == aset(icnt));
end

bool = ones(size(nset));
bool(indices) = zeros(size(indices));
oset = nset(find(bool>0));

% ********************  fOset_from_Aset.m  ************************
```

fSDOFCurveFit.m

```
function [Hfit,lam] = fSDOFCurveFit(fit_freqs, h_to_fit);

% This function performs a least squares curve fit
% of the magnitude of H(½). It identifies
% natural frequency, damping ratio, and participation factor.
%
% Ref: "Improved amplitude fitting for frequency and damping"
%  by A. M. Rinawi and R. W. Clough
% Proceedings of the 10th IMAC,  Vol.1, p.25.
%
% Usage:
%    Hamps is a vector of complex FRF values in form a + bj
%    lofreq: lower frequency limit in Hz.
%    hifreq: upper frequency limit in Hz.
%    deltafreq: frequency step in Hz.
%
% Provided by Prof Gordis
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% fit_freqs = freqPlot; % input fit_freqs as Hz in row vector form [a:b:c];

% h_to_fit = H11'; %input h_to_fit as column vector

fit_freqs = fit_freqs * 2 * pi;

% Assemble the linear system [T] {x} = {y} as per above Ref.

T=zeros(3,3);
T(1,1) =  sum(h_to_fit.^6);
T(1,2) =  sum((h_to_fit.^6).*(fit_freqs.^2));
T(1,3) = -sum(h_to_fit.^4);
T(2,2) =  sum((h_to_fit.^6).*(fit_freqs.^4));
T(2,3) = -sum((h_to_fit.^4).*(fit_freqs.^2));
T(3,3) =  sum(h_to_fit.^2);

for ii = 1:3;
   for jj = ii:3;
      T(jj,ii) = T(ii,jj);
   end
end

y = zeros(3,1);
y(1) = -sum((h_to_fit.^6).*(fit_freqs.^4));
y(2) = -sum((h_to_fit.^6).*(fit_freqs.^6));
```

```
y(3) =  sum((h_to_fit.^4).*(fit_freqs.^4));


x = T\y;

wn =(x(1)^(1/4));
zeta = sqrt((x(2)/(4*sqrt(x(1))) + (1/2)));
Pn = sqrt(x(3));
%
wn_vec = ones(size(fit_freqs)) * wn;
zeta_vec = ones(size(fit_freqs)) * zeta;

Hfit=Pn* ((wn_vec.^2-fit_freqs.^2).^2+...
   (2*wn_vec.*zeta_vec.*fit_freqs).^2).^(-1/2);

lam=wn^2;

sprintf('Identified Natural Frequency (Hz): %g', wn/2/pi)
% sprintf('Identified Damping Ratio (Non-Dimens.): %g', zeta)
% fit_freqs = fit_freqs/2/pi;

%plot(fit_freqs,log10(abs(Hfit)),'-.o');grid on

% **********************  fSDOFCurveFit.m  ************************
```

fSpringMass2.m

```
function [k,m]=fSpringMass2(springs,mass,BC);
%
%       Usage: function [k,m]=fSpringMass2(springs,mass,BC)
%
%       This function script assembles the stiffness [k] and mass
%       [m] matrices for an assemblage of springs.
%
%
%       A linear chain of springs and masses is assumed.
%               The number of springs is defined by the length of the vector 'springs'
%               and their values by the elements of 'springs.'
%
%               The number of masses is defined by the length of the vector 'mass'
%               and their values by the elements of 'mass'.
%               NOTE:The number of masses must equal to the final number of active
%                               DOF (i.e. after BC's applied).
%
%       Boundary conditions are specified by the vector 'BC.' This vector
%       contains the DOF numbers which are to be restrained.
%
%       For example, to build the following system:
%
%                   .01       .02       .015
%           |--////--[m]--////--[m]--////--[m]
%               5       6       3.4
%
%       springs = [5 6 3.4];
%       mass    = [.01 .02 .015];
%       BC              = [1]
%
%
% _____________________________________________________________________
%
%                   BEGIN SCRIPT
%                   ~~~~~ ~~~~~~
%

if length(mass) == (length(springs)+1) - length(BC);

        k               = zeros(length(springs)+1,length(springs)+1);
        m               = zeros(length(mass));

%       assemble stiffness matrix:
```

```
        rows = [0 1];
        for ispring = 1 : length(springs);

                rows = rows + 1;

                addthis = [springs(ispring) -springs(ispring);-springs(ispring)
springs(ispring)];
                k(rows,rows) = k(rows,rows) + addthis;
        end

        if ~isempty(BC);
                keep = fOset_from_Aset(length(springs)+1,BC);
                k = k(keep,keep);
        end

%       assemble mass matrix:

        m = diag(mass);

else

        disp('Error in fSpringmass2. Check # masses, springs, and BC"s.')
        return

end
% ********************  END fSpringMass2.m  ************************
```

HresiduesL.m

```
% Building FRF (H matrix) from residues in Shape files saved in ME Scope
% Preparing ME Scope file for Matlab:
% 1) In ME Scope export Shape Table file in format "Spreadsheet Shape
% Table(*.TXT). This file is tab delimited.
% 2) Delete header information.  The Matlab command "dlmread" does not read
% text.
% Note on the command "dlmread":
%    DLMREAD  reads numeric data from the ASCII delimited file.
%    Use '\t' to specify a tab.

%          RESULT= DLMREAD(FILENAME,DELIMITER,RANGE) reads the range
specified
%    by RANGE = [R1 C1 R2 C2] where (R1,C1) is the upper-left corner of
%    the data to be read and (R2,C2) is the lower-right corner. R and C are
%    zero-based so that R=0 and C=0 specifies the first value in the file.

% 3) Keep Damped Natural Frequencies in the first row.
% 4) Keep Damping Ratios in second row.
% 5) Delete the following or its equivalent for each row:
%   "GPO:PO"          "1Z:41Z[1]"   "(m/s^2)/N-sec",
% keeping only the numberic data in the remaining rows and columns.

% Written by Constance R S Fernandez, Spring 2004
%
% Inputs
% -------
% Hloop
% Raset
% lamABCtot

% Program called
% --------------
% fOset_from_Aset
% fSDOFCurveFit

% Outputs
% --------
% Rndof, RASET, HRoset
% m_inct, modes
% hh, dd
% HZ, DR
% sigma, pole, poleS
% u, uu, uj, uuj, U, UJ, U_vect, U_vectS
% iR, W, H, k, wp, kp
```

```
% HHABC, ZABC, ZOSET, HH
% FRFpeak, PP
% peakstart, peakdata, peakend, peakPlot
% Hfit_lamOSET, vect_LAMOSET

% --------
% The following code is written for ten modes and can be easily editted to
% handle more or less modes.
modes = 10; % number of modes to be used

Rndof = 42; %number of DOF recorded
RASET = Raset(Hloop, :); %location of new pinned BC
HRoset = fOset_from_Aset(Rndof,Raset); %unrestrained DOF

% "for" loop to read .txt file and build required vectors
for m_inct = 1:modes;
   % range to be used in reading .txt file, cooresponds to Natural Freq in Hz
   hh = [0 0 0 (modes-1)];
   % range to be used in reading .txt file, cooresponds to Damping Ratio
   dd = [1 0 1 (modes-1)];
   Hz = dlmread('shapeall.txt','\t',hh); % freq in Hz (vector)
   DR = dlmread('shapeall.txt','\t',dd); % damping ratio (vector)
   DR = DR/100; % Converts Damping Ratio from percent

   sigma(m_inct)= (Hz(m_inct)* DR(m_inct))/(sqrt(1-(DR(m_inct))^2)); % damping
coefficient (scaler)
   pole(m_inct) = -sigma(m_inct) + i*Hz(m_inct); %pole location (scaler)
   poleS(m_inct) = -sigma(m_inct) - i*Hz(m_inct);% pole conjugate (scaler)

   u = 2*m_inct-2; % corresponds to real part of respective mode shape
   uu = [2 u 43 u]; % range to be used in reading .txt file, real part
   uj= 2*m_inct-1;% % corresponds to imag part of respective mode shape
   uuj = [2 uj 43 uj]; % range to be used in reading .txt file, imag part

   U = dlmread('shapeall.txt','\t',uu); % real part of u-vector
   UJ = dlmread('shapeall.txt','\t',uuj); % imag part of u-vector

   U_vect(:,m_inct) = U + UJ*i; % mode shape vector, saved WRT to mode
   U_vectS(:,m_inct) = U - UJ*i;% complex conjugate of mode shape vector

end % "for" loop, m_inct = 1:modes

iR = 0;
for w = 0:1.831050e-001:6.589949e+002 % freq used in data collection
   iR = iR +1; % counter
   H = 0; % initalizes H for each freq
```

```
   for k = 1:modes; % summation for all modes used

      H = (U_vect(:,k)*(U_vect(:,k))')/(j*w-pole(k))+...
         (U_vectS(:,k)*(U_vectS(:,k))')/(j*w-poleS(k))+ H;
   end %"for k = 1:modes" loop
   % ------ABC applied     --------

   HHABC = H (Raset,Raset);
   % inverting to get Z, used in plotting peaks
   ZABC = inv(HHABC);
   % saves Z for each increment of frequency
   ZOSET(iR) = ZABC(1,1);

   HH(iR) = H(41,41); % driving point function NOTE: specific to this experiment
end % w = ... " loop

% plotting

w =  [0:1.831050e-001:6.589949e+002];
plot(w, log(abs(ZOSET))), grid on
axis tight
title ('OSET Freq inv(H) using mode shapes from MeScopeVES')
xlabel ('Hz')
ylabel ('log Magnitude')
figure(2)
plot(w,log(abs(HH))), grid on

%---------------
%---Peak gathering loop, curve fit program----
%---------------

% ----INITIALIZATION---
for FRFpeak = 1:modes;

   pp = pp+1; % index in natural frequency vector
   iRpeak=0; % initizes the index used in loop

   %sprintf('Mode %d',FRFpeak)% displays which mode the following is requested
   peakstart = lamABCtot(FRFpeak)-10; % input('Enter starting omega (Hz) : ');
   peakdelta = .5; %Hz input('Enter delta omega for this peak (Hz): ');
   peakend = lamABCtot(FRFpeak)+10; %input ('Enter ending omega (Hz): ');
   peakPlot = [peakstart : peakdelta : peakend]; % for plotting in Hz

   %---------------
   %-"for" loop to calculate the driving point and FRF of the remaining DOF
   %  of reduced range of peak, same as previous calculation except range
```

```
   %  is smaller which is needed for the curve fit program to be called.
   %---------------

   for wp = peakPlot % freq used in data collection
      iRpeak = iRpeak +1; % counter
      Hpeak = 0; % initalizes H for each freq
      for kp = 1:modes; % summation for all modes used

         Hpeak = (U_vect(:,kp)*(U_vect(:,kp))')/(j*wp-pole(kp))+...
            (U_vectS(:,kp)*(U_vectS(:,kp))')/(j*wp-poleS(kp))+ H;
      end % "kp = 1:modes" loop

      % ------ABC applied     --------

      HABCpeak = Hpeak(Raset,Raset);
      % inverting to get Z, used in plotting peaks
      ZABCpeak = inv(HABCpeak);
      % saves Z for each increment of frequency
      ZOSETpeak(iR) = ZABCpeak(1,1);

   end %  "for wp=peakPlot" loop

   [Hfit,lamOSET]= fSDOFCurveFit(peakPlot,ZOSETpeak);

   vect_LAMOSET(pp,1) = lamOSET; % saves nat freq found in fSDOFCurveFit
   % in vector form

   clear ZOSETpeak Hpeak HABCpeak ZABCpeak iRpeak

end % "FRFpeak for loop"
%vect_lamx_oset = vect_LAMOSET; % use with curve fit program

% vector of natural frequencies of ABC systems
% ********************  END HresiduesL.m  ************************
```

Hs.m

```
% This program plot Syn FRF for 41Z:41Z. Uses the following
% formula:  H = (res/(2*j*(s-peak)))+(res/(2*j*(s-peakS)))+H;
% This program is has aset and oset hard coded.

% Written by Constance R S Fernandez, Spring 2004

%load testBEAM
%load HsynMEscope

% Inputs
% --------
% kx_beamBC
% mx_beamBC

% Programs called
% ---------------
% fmodes

% Outputs
% --------
% k, m
% ndof, aset, oset
% lam, phi
% freq
% mm, iR
% SUM
% omega
% H, Haa, Zaa, Zaa11
% i, res, peak, S, peakS

k = kx_beamBC; % stiffness matrix with BC
m = mx_beamBC;% mass matrix with BC

% zeta = .02;
ndof = [1:1:84];
aset = [81]; % contrained set
oset = [1:1:80,82,83,84]; % uncontrained set

k = k(oset,oset);
m= m(oset,oset);

[lam,phi]=fModes(k,m);
```

```
freq = sqrt(lam)/2/pi;

for mm = 1:4;

   iR = 0;
   SUM = [20, 15, 10, 5];

   for omega = [0:1:2240]; % in Hz

      iR = iR + 1;
      H=0;
      for i = 1:SUM(mm)
         res = phi(:,i)*phi(:,i)';
         peak = j*freq(i);
         s = j*omega;
         peakS = -peak;
         H = (res/(2*j*(s-peak)))+(res/(2*j*(s-peakS)))+H;
      end % "for i=1:SUM (mm)" loop
 Haa = H(aset,aset);
 Zaa = inv(Haa);

      Zaa11(iR,mm) = Zaa(1,1);

   end %"for omega= [0:1:2240]" loop
end% for "mm  = 1:4" loop

% plotting

w = [0:1:2240];
figure(1)
plot(w, log(abs(Zaa11(:,1))),w, log(abs(Zaa11(:,2))),w, log(abs(Zaa11(:,3))),...
  w, log(abs(Zaa11(:,4))));

% , w, log(abs(Hs(:,5))),w, log(abs(Hs(:,6))),...
%    w, log(abs(Hs(:,7))),w, log(abs(Hs(:,8))), w, log(abs(Hs(:,9)))), grid on...

axis tight
xlabel ('Hz')
ylabel ('log Mag')
title( 'Syn FRF for 41Z:41Z')
legend ('Modes sum = 20', 'Modes sum = 15', 'Modes sum = 10', ...
   'Modes sum = 5')
hold on, grid on
x = ones(8,1)*8;
stem(freq(1:8),x,'b')
% ********************  END Hs.m  ************************
```

Htrial.m

```
%This  program was written orginially to run as a loop to find the modes of
%H with respect to each ABC system.  Since the program fSDOFCurveFit works
%when only one peak is used, this program plotted the H using the formula
%      Z = kx_beam - omega.^2 * mx_beam + j*c*omega;
%      h = inv(Z);
%      H = h(ASET, ASET); % reducing H is only ASET rows and columns
%      habc = H(HOSET, HOSET); % reducing H according to ABC
%      zabc= inv(habc); % inverse as defined as driving point

% Then the driving is plotted for visual to user.  The user is then asked
% to enter peak omega values and fSDOFCurveFit program is called to pick
% peak frequency value and build a vector of natural frequencies of system.
%  This loop is repeated for all ABC systems.  However, when the loop is
%  run there was a problem that the programmer could not correct.  Instead new
%  lines were written:

   % kxOSET = kx_beam(Hoset, Hoset);
   % mxOSET = mx_beam(Hoset, Hoset);
   % [LAMOSET,PHIOSET]=fmodes(kxOSET,mxOSET);
% Since the program saw direct correlation between the FE calculated i.e.
% lam and phi calculated from program "fmodes" and those calculated slowly
% and outside of complex programming loop by fSDOFCurveFit, programmer decided
% to use lam and phi calculated by fModes as FE values for experiment. Old
% code line are still shown in program as foundation for future versions of
% this program.
% Written by Constance R S Fernandez, Spring 2004

% Inputs
% -------
% icnt_oset
% OSETtot
% oset
% kx_beam, mx_beam

% Program called
% ---------
% fModes

% Outputs
% -------
% lamOSET
% vect_lamx_oset
% Hloop
% H_inct
```

```
% Hoset
% kxOSET
% mxOSET
% LAMOSET
% PHIOSET

% Initalization
lamOSET = [];
vect_lamx_oset = [];

for Hloop = 1:icnt_oset;% loop repeats for number os ABC system defined

   H_inct = size(find(OSETtot(Hloop, :)>0),2);
   % counts how many places in a given row of OSETtot are not zero
   Hoset = OSETtot(Hloop, 1:H_inct);
   % creates a new vector with just non-zero values

   OSETtot(icnt_oset, 1:length(oset)) = oset;

   %     H_inct = size(find(OMITset(Hloop, :)>0),2);
   %     Hoset = OMITset(Hloop, 1:H_inct);
   %     ASET = [1 3 5 7 9 11 13 15 17 19];
   %     HOSET = fOset_from_Aset(10,Hoset);
   %     StartOmega =1;
   %     DeltaOmega = 2;
   %     EndOmega = 2000;
   %     freqPlot = [StartOmega : DeltaOmega : EndOmega]; % used for plotting
   %
   %     Zabc11 = zeros(length(freqPlot), 1);

   %     iR=0; % initizes the index used in loop
   %     c = 0;

   kxOSET = kx_beam(Hoset, Hoset);% stiffness matrix with only unrestained DOF wrt
ABC system
   mxOSET = mx_beam(Hoset, Hoset);% mass matrix ...

   [LAMOSET,PHIOSET]=fmodes(kxOSET,mxOSET);

   %"for" loop to calculate the driving point and FRF of the remaining set of DOF
   %     for omega = freqPlot  %rad/sec
   %        omega = omega*2*pi;
   %        iR = iR + 1; % loop counter
   %
   %        Z = kx_beam - omega.^2 * mx_beam + j*c*omega;
   %
```

```
%       h = inv(Z);
%       H = h(ASET, ASET);
%       habc = H(HOSET, HOSET);
%       zabc= inv(habc);
%       Zabc11(iR) = zabc(1,1);
%    end
%
%    absmax = max(Zabc11);
%    absmin = min(Zabc11);
%    avg = (absmax+absmin)/2;
%    counter = 0;
%    for i = 1:length(freqPlot)
%       if Zabc11(i) >= avg
%          counter = counter+1;
%          modepeaks(counter) = freqPlot(i); % in Hz
%       else
%       end
%    end

%    figure (Hloop+3)
%
%    plot(freqPlot,log10(abs(Zabc11)),'g');grid on
%    hold on
%
%    title('inv(H) complete spectrum')
% %     for peak = 1:num_modesO;
% %
% %         sprintf('Mode %d',peak)
% %         peaklam(peak,Hloop)= input('Enter peak lamda: ');
% %     end
% % %
%    pp = 0;
% %
%    for peak = 1:num_modesO;
%       pp = pp+1;
%       sprintf('Mode %d',peak)
%       peakstart = input('Enter starting omega  : ');
%       peakdelta = input('Enter delta omega for this peak : ');
%       peakend = input ('Enter ending omega : ');
%       peakPlot = [peakstart : peakdelta : peakend];
%
%       iRpeak=0; % initizes the index used in loop
%       c = 0;
%       clear Zpeak hpeak habcpeak zabcpeak Zabc11peak
%
%       %"for" loop to calculate the driving point and FRF of the remaining set of DOF
```

```
%       for omega = peakPlot  % in Hz
%          omega = omega*2*pi;
%          iRpeak = iRpeak + 1; % loop counter
%          Zpeak = kx_beam - omega.^2 * mx_beam + j*c*omega;
%
%          hpeak = inv(Zpeak);
%          habcpeak = hpeak(Hoset, Hoset);
%          zabcpeak= inv(habcpeak);
%          Zabc11peak(iRpeak) = zabcpeak(1,1);
%       end
%       figure (2)
%
%       plot(peakPlot,log10(abs(Zabc11peak)),'g');grid on
%       hold on
%
%       title('inv(H) one peak')
%       [Hfit,lamOSET]= fSDOFCurveFit(peakPlot,Zabc11peak);%555

%       vect_LAMOSET(pp) = lamOSET;
%clear Zpeak hpeak habcpeak zabcpeak Zabc11peak iRpeak

%    end
%if lamOSET == [];

if vect_lamx_oset == [];

   vect_lamx_oset = LAMOSET(1:5);

else
   vect_lamx_oset  = cat(1,vect_lamx_oset,LAMOSET(1:5));
end % "if vect_lamx_oset == []"
end % "for Hloop=1:icnt_oset" loop
%end
% ********************  END Htrial.m  ***********************
```

normRUNthru_crs.m

```
% This program finds the NORM of the columns of sensitivity matrix and the
% NORM of the rows of the inverse of the sensitivity matrix.  This was used
% to find a correlation of the good prediction to the ABC system used.  It
% also plots the information in helpful graphes.
%
% This program was written for a system of 19 natural freq. Set of 5 modes
% were used in each ABC system, i.e., modes 1-5, modes 6-10,or modes 11-15.
% This accounts for the 3 sets of modes per condition as listed below in
% the "for" loop modeN = 1:3. This program compares the use of the first 5
% modes of the base system and one set of five modes of the ABC  to that of
% of just 10 modes of the ABC.  Notice that the sensitivity is a 10x10
% square matrix indicating that only mass or EI changes, not both were
% made.

% This program is not part of Build2Beams_crs.m program.  It is run
% separately.

% Written by Constance R S Fernandez, Spring 2004

% Inputs
% ------
% cond_basePlus
% icnt_oset
% T_sens_tot
% EI_lbls, mass_lbls
% dv_mass, dv_EI, dv_cal_ABCten

% Outputs
% --------
% BASE
% BASET
% abcN, countN, a_cN
% modeN, startmodeN, startmodeNT
% bb, t, tINV, cc, T, TINV, vv, tt
% modelabelNORM
% norm_vectT, norm_vecTABC, norm_vecTinvABC
% normC
% baseN, baseABCN, abc_conN, abc_conTN
% baseABCNten, abc_conNten, abc_conTNten

% ----Start program----

BASE = int2str(cond_basePlus(1)); % cond no. of the base line system
BASET = sprintf('Base[1:5] Cond = %s', BASE); % used for plotting
```

```
% ======Initialization=====%

abcN = 0;
countN =0;

% =======Calculations of NORM vectors======%
for count = 1:icnt_oset +1 % number of conditions (base + ABC)
   a_cN = 1;

   for modeN = 1:3 % 3 sets of modes per boundry condition
      startmodeN = abcN + a_cN; % the beginning mode number of each set

      % indicates the use of the first 5 modes of base system + 5 modes
      % of ABC system
      bb = [1:5, startmodeN: startmodeN+4];

      % labeling of modes for plotting
      modelabelNORM = int2str(a_cN:a_cN+4);

      a_cN = a_cN+5;    %advances to the next set of modes

      % base system plus 5 modes of ABC
      t = T_sens_tot(bb,:); % 10x10 matrix
      tINV = inv(T_sens_tot(bb,:));% 10x10 matrix

      % first 10 modes of ABC solo
      startmodeNT = abcN+1; % the beginning mode number of each set
      cc = [startmodeNT: startmodeNT+9];

      T = T_sens_tot(cc,:);% 10x10 matrix
      TINV = inv(T_sens_tot(cc,:));% 10x10 matrix

      % for loop for NORM of columns and rows of inv(sens matrix)
      for vv = 1:10 % 10 rows, 10 columns
         % base + ABC system
         norm_vecT(vv,countN+modeN) = norm(t(:,vv)); % columns
         norm_vecTinv(vv,countN+modeN) = norm(tINV(vv,:)); % rows
         % for 10 modes ABC solo
         norm_vecTABC(vv,countN+modeN) = norm(T(:,vv)); % columns
         norm_vecTinvABC(vv,countN+modeN) = norm(TINV(vv,:)); % rows

      end % vv loop
   end % ModeN loop
   abcN = abcN+19; % advances to the next ABC system
   countN = countN+3; % counts up each set of ABC
```

```
end % count loop
normC=4; % initialize for plots

%==========PLOTTING==========%

for tt=1:10 % figures (30-40) plots 6 graphes per figure
   figure(tt+30)
   subplot(3,2,1)
   bar(norm_vecT(:,normC))
   title ('Norm col Tsens, ABC Modes [1:5]');

   subplot(3,2,3)
   bar(norm_vecTinv(:,normC))

   title ('Norm row TsensINV, ABC Modes [1:5]');

   subplot(3,2,2)
   bar(norm_vecTABC(:,normC))
   title ('Norm col Tsens, ABC Modes [1:10]')
   subplot(3,2,4)
   bar(norm_vecTinvABC(:,normC))
   title ('Norm row TsensINV, ABC Modes [1:10]')

   subplot(3,2,5)
   % plotting error prediction
   % using [1:5] modes of ABC system + [1:5] modes of Base system;
   baseABCN = bar(dv_cal_BasePlus(:,normC),.5,'r');hold on

   % plotting error prediction using [1:5] modes of Base system;
   baseN = bar(dv_cal_ABC(:,1),.25,'b');


   abc_conN = int2str(cond_basePlus(normC)); % cond no. for legend
   abc_conTN = sprintf('Base[1:5]+ABC[1:5] Cond = %s', abc_conN);

   grid on
   legend([baseABCN,baseN],BASET,abc_conTN), hold on


   % plotting actual error
   if EI_lbls ~=[] & mass_lbls ~=[]
      stem(mass_lbls, dv_mass,'y','filled'); hold on;
      stem(mass_lbls, dv_mass,'k')

      EIplot = EI_lbls+10;hold on
      stem(EI_lbls, dv_EI,'c','filled');hold on;
```

```
   stem(EI_lbls, dv_EI,'k')

elseif  mass_lbls ~=[] & EI_lbls ==[]
   stem(mass_lbls, dv_mass,'y','filled');hold on;
   stem(mass_lbls, dv_mass,'k')

else
   stem(EI_lbls, dv_EI,'c','filled');hold on;
   stem(EI_lbls, dv_EI,'k')

end % if EI_lbls ~=[] & mass_lbls ~=[]

title (sprintf('Error, Base [1:5] + ABC [1:5], pinned NODE # %d', (tt+1)))

subplot(3,2,6)
% plotting error prediction using [1:10] modes of ABC system;
baseABCNten = bar(dv_cal_ABCten(:,normC));hold on

abc_conNten = int2str(cond_ABCten(normC));
abc_conTNten = sprintf('ABC[1:10] Cond = %s', abc_conNten);

grid on
legend([baseABCNten],abc_conTNten), hold on

% plotting actual error
if EI_lbls ~=[] & mass_lbls ~=[]
   stem(mass_lbls, dv_mass,'y','filled'); hold on;
   stem(mass_lbls, dv_mass,'k')

   EIplot = EI_lbls+10;hold on
   stem(EI_lbls, dv_EI,'c','filled');hold on;
   stem(EI_lbls, dv_EI,'k')

elseif  mass_lbls ~=[] & EI_lbls ==[]
   stem(mass_lbls, dv_mass,'y','filled');hold on;
   stem(mass_lbls, dv_mass,'k')

else
   stem(EI_lbls, dv_EI,'c','filled');hold on;
   stem(EI_lbls, dv_EI,'k')

end % EI_lbls ~=[] & mass_lbls ~=[]

title (sprintf('Error, ABC Modes [1:10], pinned NODE # %d', (tt+1)))

 normC = normC+3; % advances to the next ABC system
```

```
end % tt = 1:10 for plotting graphes

% ********************  END normRUNthru_crs.m  ************************
```

peakmodeloop.m

```
% This program was written to find graphically the peak of FRF of ABC
% system.  It calls the FSDOFCurveFit program to find peak of FRF.
% Written by Constance R S Fernandez, Spring 2004

for peak = 1:3;

      sprintf('Mode %d',peak)
      peakstart = input('Enter starting omega  : ');
      peakdelta = input('Enter delta omega for this peak : ');
      peakend = input ('Enter ending omega : ');
      peakPlot = [peakstart : peakdelta : peakend];

      iRpeak=0; % initizes the index used in loop
      c = 0;
      clear Zpeak hpeak habcpeak zabcpeak Zabc11peak

      %"for" loop to calculate the driving point and FRF of the remaining set of DOF
      for omega = peakPlot  % in Hz
         omega = omega*2*pi;
         iRpeak = iRpeak + 1; % loop counter
         Zpeak = kx_beam - omega.^2 * mx_beam + j*c*omega;

         hpeak = inv(Zpeak);
         habcpeak = hpeak(Hoset, Hoset);
         zabcpeak= inv(habcpeak);
         Zabc11peak(iRpeak) = zabcpeak(1,1);
      end
      figure (2)

      plot(peakPlot,log10(abs(Zabc11peak)),'g');grid on
      hold on

      title('inv(H) one peak')
      [Hfit,lamOSET]= fSDOFCurveFit(peakPlot,Zabc11peak);%555

      %clear Zpeak hpeak habcpeak zabcpeak Zabc11peak iRpeak

   end

% ********************  END peakmodeloop.m  ************************
```

PlotBeamModes_crs.m

```
% Calculates natural frequencies

% Provided by Prof Gordis

% Inputs needed:
% -----------------
% ka, ma, mx, kx

% Programs needed:
% -----------------
% fModes

% Outputs:
% -----------------
% lama, phia, lamx, phix (without rigid body modes)
% num_rbm
% phia_plot, phix_plot


disp('  ');
disp(' Calculating modes for each beam...plot frequency comparison')


% Get modes of each beam:

[lama,phia]=fModes(ka,ma);
[lamx,phix]=fModes(kx,mx);

%used to plot the mode shapes org BC before ABC
phia_plot = phia;
phix_plot = phix;

% Set any rigid body mode freqs to zero:

  num_rbm = length(find(lama < 1));

  sprintf('Number of Rigid Body Modes Found: %2i', num_rbm)

    disp( ' Removing rigid body mode frequencies from vectors...')
    lama = lama(find(lama > 1));
    lamx = lamx(find(lamx > 1));

% ********************  END PlotBeamModes_crs.m  ************************
```

plottingBARS_crs.m

```
% To be used with Build2Beams.m and
%
% This program plots 9 graphes per figure.  The first columns of 3 graphes
% are the mode shapes of the ABC system used in error prediction.  The next
% column of 3 graphes are the error prediction using only 5 modes of ABC
% system. The last column of 3 graphes are the error predictions using the
% first 5 modes of base system plus 5 modes of the ABC system.  The row
% represent modes 1-5, middle row: modes 6-10, last row : modes 11-15. Each
% of the error prediction graphes also have the base only prediction  and
% the actual error plotted for easy reference.
%
% Written by Constance R S Fernandez, Spring 2004

% Inputs
% --------
% cond_basePlus
% FOM_ABC5per, FOM_PLUSper
% icnt_oset
% modeshape
% rel_freqERROR
% ypos
% EI_lbls, mass_lbls
% dv_mass, dv_EI

% Outputs
% --------
% BASE, BASET
% FOMBASE, FOMABC, FOMPLUS
% intervelp
% ER, barp, shape, error, a_cp, modep, ap,
% modelabelp, FOMABClabelp, FOMPLUSlabelp
% abc_con, abc_conT
% ABC, base
% plus_con, plus_conT
% base, plus, EI_plot


BASE = int2str(cond_basePlus(1));
FOMBASE = int2str(FOM_ABC5per(1));
BASET = sprintf('Base Cond = %s, FOM = %s', BASE, FOMBASE);


intervelp = 3;
modeshape = 1;
```

```
ER = 1;
for barp = 1:icnt_oset

   figure(barp+10) % figures 11-20
   format bank
   shape = [modeshape:modeshape+10];
   error = round(rel_freqERROR(ER:ER+15)*100)/100;
   a_cp = 1;
   for modep = 1:3 %3 sets of modes per boundry condition

      ap = [a_cp: a_cp+4]; %modes
%        REL_error1 = int2str(error(a_cp));
%        Errorlabelp = sprintf('Rel error = %s', REL_error1);
%        REL_error2 = int2str(error(a_cp+1));
%         Errorlabelp = sprintf('Rel error = %s', REL_error2);
%        REL_error3 = int2str(error(a_cp+2));
%         Errorlabelp = sprintf('Rel error = %s', REL_error3);
%        REL_error4 = int2str(error(a_cp+3));
%         Errorlabelp = sprintf('Rel error = %s', REL_error4);
%        REL_error5 = int2str(error(a_cp+4));
%         Errorlabelp = sprintf('Rel error = %s', REL_error5);

      modelabelp = int2str(a_cp:a_cp+4);
      FOMABC = int2str(FOM_ABC5per(intervelp+modep));
      FOMABClabelp = sprintf('System FOM = %s', FOMABC);

      FOMPLUS = int2str(FOM_PLUSper(intervelp+modep));
      FOMPLUSlabelp = sprintf('System FOM = %s', FOMPLUS);


      % ======================================================%
      % =====mode shape or beam X and beam A with ABC=================%
      % ======================================================%

      figure(barp+50)
      subplot(3,1,modep)
      plot(ypos, dispA_tot(shape,a_cp),'k-o', ypos, ...
         dispA_tot(shape,a_cp+1),'g-s', ypos, ...
         dispA_tot(shape,a_cp+2),'b-d', ypos, ...
         dispA_tot(shape,a_cp+3),'r-x', ypos, ...
         dispA_tot(shape,a_cp+4),'m-*', ypos, ...
         dispX_tot(shape,a_cp),  'r--o', ypos, ...
         dispX_tot(shape,a_cp+1),'b--s', ypos, ...
         dispX_tot(shape,a_cp+2),'m--d', ypos, ...
         dispX_tot(shape,a_cp+3),'c--x', ypos, ...
         dispX_tot(shape,a_cp+4),'k--*'), grid on...
```

```
    legend(sprintf('Rel Freq Error = %d', error(a_cp)),...
       sprintf('Rel Freq Error = %d', error(a_cp+1)),...
       sprintf('Rel Freq Error = %d', error(a_cp+2)),...
       sprintf('Rel Freq Error = %d', error(a_cp+3)),...
       sprintf('Rel Freq Error = %d', error(a_cp+4)));

%       legend(sprintf('Bm X, Md %d', a_cp'), sprintf('Bm X, Md %d', a_cp+1),...
%          sprintf('Bm X, Md %d', a_cp+2), sprintf('Bm X, Md %d',a_cp+3),...
%          sprintf('Bm X, Md %d', a_cp+4), sprintf('Base, Md %d', a_cp), ...
%          sprintf('Base, Md %d', a_cp+1), sprintf('Base, Md %d', a_cp+2), ...
%          sprintf('Base, Md %d', a_cp+3), sprintf('Base, Md %d', a_cp+4))


    title(sprintf('Modes [ %s]',modelabelp))
    axis tight

    %=========================================================%
    % ===== bar graphes of error solution using only ABC==========%
    %=========================================================%

    figure(barp+10) % figures 11-20
    subplot(3,2,2*modep-1)

    abc_con = int2str(cond_ABC(intervelp+modep));
    abc_conT = sprintf('ABC Cond = %s, FOM = %s', abc_con, FOMABC);

    ABC = bar(dv_cal_ABC(:,intervelp+modep),.5,'r'); hold on
    base = bar(dv_cal_ABC(:,1),.25,'b');hold off%on % base first 5 modes
    %FOMabc = bar(1,0); hold off
    grid on
    %legend([ABC,base,FOMabc],plus_conT,BASET,FOMABClabelp), hold on
    % grid on
    legend([ABC,base],abc_conT,BASET), hold on

    title(sprintf('ABC only, [ %s]', modelabelp));



    if EI_lbls ~=0 & mass_lbls ~=0
       % plots actual error
       stem(mass_lbls, dv_mass,'y','filled'); hold on;
       stem(mass_lbls, dv_mass,'k'), hold on;

       EIplot = EI_lbls+10; hold on % last half of plot
```

```
    stem(EIplot, dv_EI,'c','filled');hold on;
    stem(EIplot, dv_EI,'k'); hold on
    % plots the green triangle which indicates pinned node
    plot(barp+1,0,'g^',barp+1,0,'gh',barp+1,0,'g*',...
       barp+11,0,'g^',barp+11,0,'gh',barp+11,0,'g*'  )

elseif  mass_lbls ~=0 %&EI_lbls =0
    % plots actual error
    stem(mass_lbls, dv_mass,'y','filled'); hold on
    stem(mass_lbls, dv_mass,'k'); hold on
    % plots the green triangle which indicates pinned node
    plot(barp+1,0,'g^',barp+1,0,'gh',barp+1,0,'g*')

else
    % plots actual error
    stem(EI_lbls, dv_EI,'c','filled');hold on;
    stem(EI_lbls, dv_EI,'k'); hold on
    % plots the green triangle which indicates pinned node
    plot(barp+1,0,'g^',barp+1,0,'gh',barp+1,0,'g*')

end % EI_lbls ...

% ========================================================%
% =========bar graphes of error solution using ABC + base========%
% ========================================================%

subplot(3,2,2*modep)


plus_con = int2str(cond_basePlus(intervelp+modep));% for legend
plus_conT = sprintf('Base+ABC Cond = %s FOM = %s', plus_con, FOMPLUS);
% for legend

plus = bar(dv_cal_BasePlus(:,intervelp+modep),.5,'r'); hold on
base = bar(dv_cal_ABC(:,1),.25,'b'); hold on % base first 5 modes
%FOMplus = bar(1,0); hold off
grid on
legend([plus,base],plus_conT,BASET), hold on
%     legend([plus,base,FOMplus],plus_conT,BASET,FOMPLUSlabelp), hold on

if EI_lbls ~=0 & mass_lbls ~=0
    % plots actual error
    stem(mass_lbls, dv_mass,'y','filled'); hold on;
    stem(mass_lbls, dv_mass,'k'); hold on
    EIplot = EI_lbls+10; hold on % last half of plot
    stem(EIplot, dv_EI,'c','filled');hold on;
```

```
        stem(EIplot, dv_EI,'k');hold on
        % plots the green triangle which indicates pinned node
        plot(barp+1,0,'g^',barp+1,0,'gh',barp+1,0,'g*',...
           barp+11,0,'g^',barp+11,0,'gh',barp+11,0,'g*'  )

      elseif  mass_lbls ~=0 %&EI_lbls =0
        % plots actual error
        stem(mass_lbls, dv_mass,'y','filled');hold on;
        stem(mass_lbls, dv_mass,'k');hold on
        % plots the green triangle which indicates pinned node
        plot(barp+1,0,'g^',barp+1,0,'gh',barp+1,0,'g*')
      else
        % plots actual error
        stem(EI_lbls, dv_EI,'c','filled');hold on;
        stem(EI_lbls, dv_EI,'k'), hold on
        % plots the green triangle which indicates pinned node
        plot(barp+1,0,'g^',barp+1,0,'gh',barp+1,0,'g*')

      end % if EI_lbls ...

      title(sprintf('Base [1:5] + ABC [ %s]', modelabelp));
      %       title(sprintf('Base + ABC, pinned at NODE# %d',barp +1))
      a_cp= a_cp +5;
   end  % modep loop

   modeshape = modeshape + 11; % advances
   intervelp = intervelp +3; % advances to the next ABC system
   ER = ER+19;
end % barp loop

% ********************  END plottingBARS_crs.m  ************************
```

recorded_H_crs.m

```
% Written by Constance R S Fernandez, Spring 2004

% This program is used to find the natural frequencies of the ABC Systems
%
% The natural frequencies of each ABC system were calculated by keeping
% the unrestrained DOF from the stiffness and mass matrices of Beam X
% because natural freq of system coorespond to the unrestained DOF of the
% system, oset.
% This method was used for ease of use in multiple computer runs.
%
% Another method of getting ABC system:
%
% 1) Build the impedence matrix Z
%    a) using a range of frequencies
%    b) stiffness and mass matrices of the experimental Beam X saved
%       in the program Build2Beams.m
% 2) H = inv(Z)
% 3) Only the pinned rows and columns of H are kept to yield
%    natural frequenices of the ABC systems
% 4) Z = inv(H reduced)
% 5) Plot the new Z,
%    (peaks coorespond to ABC system natural frequencies)
% 6) Use curve fitting program to estimate the freq of peak
%
% Method of getting ABC with recorded data.
% 1) H recorded is saved as spreadsheets
% 2) Only the pinned rows and columns of H are kept.
% 3) Plot the remaining H,
%    (peaks coorespond to ABC system natural frequencies)
% 4) Invert H to get Z
% 5) Use curve fitting program to estimate the freq of peak

% Inputs
% -------
% icnt_oset
% BCOSET
% BC
% kx_beam, mx_beam

% Programs Called
% ---------------
% fModes

% Outputs
% -------
```

```
% Hloop
% Ho_inct
% Hoset
% kABC,mABC

% ----INITIALIZATION---

lamOSET = [];
vect_lamx_oset = [];

%=============================================================
% loop for ABC natural frequencies (Method one)

for Hloop = 1:icnt_oset; % for all ABC systems

   Ho_inct = size(find(BCOSET(Hloop, :)>0),2);
   Hoset = BCOSET(Hloop, 1:Ho_inct); % free DOF

   HAset = BC(Hloop,:); % pinned DOF

   kABC = kx_beam(Hoset, Hoset); %stiffness matrix of (no BC)
   mABC = mx_beam(Hoset, Hoset); %mass matrix of unrestained DOF (no BC)

   % lam (natural freq^2, rad^2/sec^2), phi (mode shapes)
   [lamABC,phiABC]=fModes(kABC,mABC);

   lamABCtot = lamABC; % renames natural freq of ABC system

   %
%=============================================================
   %    %----------------------METHOD 2----------------------------------
   %
%=============================================================
   %
   %    % ----INITIALIZATION---
   %
   %    StartOmega = 1; % in Hz
   %    DeltaOmega = .5; % in Hz
   %    EndOmega = 1000; % in Hz
   %
   %    freqPlot = [StartOmega : DeltaOmega : EndOmega]; % plotting in Hz
   %    iR=0; % loop index
   %    c = 0.02; % damping ratio
   %    Zabc11 = zeros(length(freqPlot), 1);
   %    %---------------
   %    %-"for" loop to calculate the driving point and FRF of the remaining DOF
```

```
%    %---------------
%    for omega = StartOmega : DeltaOmega : EndOmega; % in Hz
%       omega = omega*2*pi; % rad/sec
%       iR = iR + 1;
%       % loop counter% system impedence of multi DOF system
%       Zx_beam= kx_beam - omega.^2 * mx_beam + j*c*omega;
%
%       hx_beam = inv(Zx_beam);
%       % dynamically reduced, spatically incomplete FRF
%       habc = hx_beam(HAset, HAset);
%       % inverting to get Z, used in plotting peaks
%       zhabc = inv(habc);
%       % saves Z for each increment of frequency
%       zhabc11(iR) = zhabc(1,1);
%
%    end % omega loop
%
%    %---------------
%    %---PLOTTING----
%    %---------------
%
%    figure(Hloop + 300) % plots the complete impedence matrix Z so user can
%    %see peaks and be able to keep peak range when prompted
%
%    plot(freqPlot,log10(abs(zhabc11)),'b');
%    xlabel('inv(Hx_beam11) of the ABC System using inv Zx_beam, (in HZ)')
%    ylabel('log10 of')
%    hold on
%
%    %---------------
%    %---Peak gathering loop, curve fit program----
%    %---------------
%
%    % ----INITIALIZATION---
%    pp = pp+1; % index in natural frequency vector
%    iRpeak=0; % initizes the index used in loop
%
%    sprintf('Mode %d',peak)% displays which mode the following is requested
%    peakstart = input('Enter starting omega (Hz) : ');
%    peakdelta = input('Enter delta omega for this peak (Hz): ');
%    peakend = input ('Enter ending omega (Hz): ');
%    peakPlot = [peakstart : peakdelta : peakend]; % for plotting in Hz
%
%    %---------------
%    %-"for" loop to calculate the driving point and FRF of the remaining DOF
%    %  of reduced range of peak, same as previous calculation except range
```

```
   %     %  is smaller which is needed for the curve fit program to be called.
   %     %---------------
   %
   %     for omega = peakPlot  % in Hz
   %        omega = omega*2*pi; % in rad/sec
   %        iRpeak = iRpeak + 1; % loop counter
   %        Zpeak = kx_beam - omega.^2 * mx_beam + j*c*omega;
   %
   %        hpeak = inv(Zpeak);
   %        habcpeak = hpeak(HAset, HAset);
   %        zabcpeak= inv(habcpeak);
   %        Zabc11peak(iRpeak) = zabcpeak(1,1);
   %     end %  "for omega loop"
   %
   %     [Hfit,lamOSET]= fSDOFCurveFit(peakPlot,Zabc11peak);
   %
   %     vect_LAMOSET(pp,1) = lamOSET; % saves nat freq found in fSDOFCurveFit
   %     % in vector form
   %
   %     clear Zpeak hpeak habcpeak zabcpeak Zabc11peak iRpeak



   %vect_lamx_oset = vect_LAMOSET; % use with curve fit program

   % vector of natural frequencies of ABC systems
   if vect_lamx_oset == 0;

      vect_lamx_oset = lamABCtot;
   else
      vect_lamx_oset  = cat(1,vect_lamx_oset,lamABCtot);
   end % "if"

end % Hloop "for"

clear peakPlot peakstart  peakdelta  peakend pp
clear freqPlot StartOmega DeltaOmega EndOmega]
clear Hloop Ho_inct Hoset HAset
clear Zx_beam hx_beam habc zhabc zhabc11 iR

% ********************  END recorded_H_crs.m  ************************
```

LIST OF REFERENCES

Berman, A. & Flannelly, W. G. (1971, August) Theory of Incomplete Models of Dynamic Structures. *AIAA Journal*. *9* (8), 1481 -1486

Brillhart R.D., Chism, T.L, Freed, A.M., & Hunt, D. (1990 Jan 29-Feb 1) Modal Test and Correlation of Commercial Titan Dual Payload Carrier, *For presentation at the 8th International Modal Analysis Conference in Kissimmee, FL*

Craig, R. R. (1981, August 5) *Structural Dynamics: An Introduction to Computer Methods.* New York: John Wiley and Sons.

Dascotte, E. Applications of Finite Element Model Tuning Using Experimental Modal Data. (June 1991) *Sound and Vibration*

Ewins, D. J. (1984). *Modal Testing: Theory and Practice*. England: Research Studies Press LTD.

Flanigan, C. C. Correlation of Finite Element Models Using Mode Shape Design Sensitivity.

Gordis, J. H. (1993) Spatial Frequency Domain Updating of Linear, Structural Dynamic Models

Gordis, J. H. (1993, April) A Frequency Domain Theory for Structural Identification. *Journal of American Helicopter Society*. 25-3

Gordis, J. H. (1994, February 21). Structural Synthesis in the Frequency Domain: A General Formulation. *Shock and Vibration. 1*(5) 461-471

Gordis, J. H. (1996, July). Omitted Coordinate Systems and Artificial Constraints in Spatially Incomplete Identification. *Modal Analysis: the International Journal of Analytical and Experimental Modal Analysis. 11*(1), 83-95

Gordis, J. H. (1999) Artificial Boundary Conditions for Model Updating and Damage Detention. *Mechanical Systems and Signal Processing, 13* (3), 437-448

Ledvij, M. (2003, April/May) Curve Fitting Made Easy. *The Industrial Physicist. 9*(2), 24-27

Nelson, R. B. (1976, September). Simplified Calculation of Eigenvector Derivatives. *AIAA Journal*, *14* (9), 1201- 1205

Rinawi, A. M. & Clough R.W. (1992) Improved Amplitude Fitting for Frequency and Damping Estimation. *Proceedings of the 10th International Modal Analysis Conference.* Vol 1, 25-29

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California