CHAPTER 15

SISO STATE SPACE MATLAB MODEL FROM ANSYS MODEL

15.1 Introduction

This chapter will develop a SISO state space MATLAB model from an ANSYS cantilever beam model. The cantilever is admittedly a trivial example, but like the tdof model used in the first part of the book, will serve as a good model to develop a fundamental understanding of the process. As we are going through the simple cantilever example we should be thinking about applying the process to a model of an actual device, for example a complete model of a disk drive, with hundreds of thousands of nodes and up to hundreds of modes in the frequency range of interest. Our objective for the model will be to provide the smallest MATLAB state space model that accurately represents the pertinent dynamics.

The model cantilever is shown in Figure 15.1. It is a 2mm wide by 0.075mm thick by 20mm long steel beam. The coordinate system is indicated on the figure. A z direction force is applied at the midpoint of the beam and z displacement at the tip is the output. Only x-z plane motion is allowed; all other degrees of freedom are constrained.

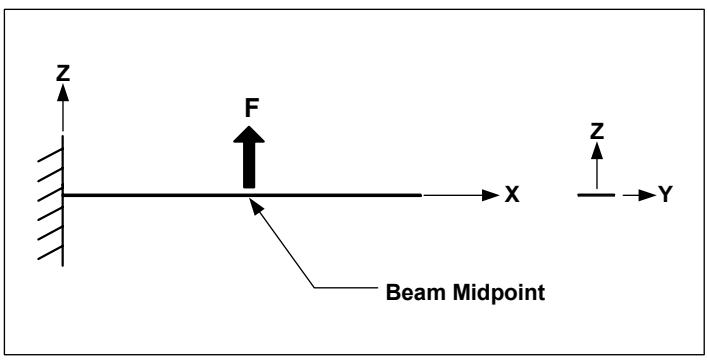

Figure 15.1: Cantilever beam with forcing function at midpoint.

We will begin by analyzing the major issues all finite element analysts face when setting up a model: defining the number of elements to use and calculating the effects of Guyan reduction, if used. We will analyze the cantilever with different numbers of elements. We will also analyze with and

without Guyan reduction and compare the resulting resonant frequencies with theoretical results. Knowing the frequency range of interest for the model, typically defined by servo bandwidth considerations, we will define a model (number of elements) that accurately predicts eigenvalues in the range of interest. In this theoretical example we have the luxury of knowing the exact values for the eigenvalues. However, in real life problems, we know that a finite element model is accurate only if we build another model with finer resolution and compare results, and/or have good experimental mode shape data with which to compare.

While Guyan reduction prior to conducting an eigenvalue analysis has been in the main replaced by the Block Lanczos eigenvalue extraction method, Guyan reduction will be presented because it is still used in creating "superelements" for large models (which are then solved using Block Lanczos) and is also used in correlating finite element and experimental model models.

For some problems, the time to perform frequency response calculations using Block Lanczos is of the same order of time as the eigenvalue extraction, which makes using MATLAB for state space frequency response models an efficient adjunct to ANSYS. We will review how to have MATLAB build a state space model given only the eigenvalues and required eigenvector information (eigenvector entries for all modes for only input and output degrees of freedom). This technique will be used for all following models, in conjunction with various mode elimination/truncation techniques.

The problem to be solved in this chapter is: Determine the smallest state space model which accurately constructs the frequency response characteristics through a given frequency range. We will assume for our problem that the servo system requires all significant modes through 20khz be included. The servo system will apply inputs in the z direction at the node located at the mid-length of the cantilever, with z direction displacement of the tip being the output.

The first step in defining the smallest model is to define the eigenvector elements for all modes for only the input and output degrees of freedom. The second step is to analyze the modal contributions of all the modes and sort them to define which ones have the greatest contribution.

One method for reducing the size of a modal model is to simply truncate the higher frequency modes. If this truncation is performed without understanding the contributions of each of the modes to the response, several problems could arise. One problem is that a high frequency mode that could alias to a lower frequency in a sampled servo system may be missed. Another hazard with arbitrarily truncating higher frequency modes is that a mode with a significant

dc gain contribution may be eliminated, adversely affecting the model. Typically the contributions of modes decrease as their frequencies increase; however, this is not always the case. In Chapter 16 we will see a cantilever model with an additional tip mass and a tip spring all mounted on a "shaker" base. It is used as an example of how excluding a specific higher frequency mode can result in a model with less than desired accuracy.

15.2 ANSYS Eigenvalue Extraction Methods

ANSYS has a number of different eigenvalue extraction techniques, but for most problems only two methods are commonly used. The first method, Block Lanczos, is the fastest and calculates all the eigenvalues or eigenvalues in a specific frequency range. Most practical models require knowledge of the modes from dc through a specified higher frequency.

The second method, Reduced, performs a Guyan reduction on the model to reduce its size, then calculates all the eigenvalues for the reduced model. All of the "master" degree of freedom eigenvector components are available immediately for use. Obtaining eigenvector components for the reduced degrees of freedom requires an additional calculation step in ANSYS.

For very large models, Block Lanczos has shown to be significantly faster than the Reduced method. If MATLAB state space models are used to calculate frequency responses using Block Lanczos results the total time to get model results can be quite satisfactory. Typically, the Reduced method is used only for small- to medium-size problems.

15.3 Cantilever Model, ANSYS Code cantbeam_ss.inp, MATLAB Code cantbeam_ss_freq.m

The ANSYS code **cantbeam_ss.inp,** listed in Section 15.7, is designed to allow the user to easily change the number of elements "num_elem" as well as the eigenvalue extraction technique "eigext."

The model was run for 2, 4, 6, 8, 10, 12, 16, 32 and 64 elements for both eigenvalue extraction methods. The Lanczos method resulted in twice the number of eigenvalues as the Reduced method because both translations and rotations are degrees of freedom for Lanczos, while the Reduced method has the rotations reduced out.

For those interested, the MATLAB code **cantbeam_ss_freq.m** plots the results of the ANSYS runs along with the theoretical frequencies for up to the first 16 modes (Chang 1969).

Figures 15.2 and 15.3 show the percentage frequency differences between the first 10 modes of the ANSYS Block Lanczos and Reduced runs and the theoretical prediction.

The maximum frequency difference for the Block Lanczos method is 2% and for the Reduced method it is 5%. For the frequency range of interest in our problem, 20 khz, the maximum frequency errors are 1% and 3%, which is deemed satisfactory. We will use the 10-element model with the Reduced method for the rest of the chapter. Real life models will have greater deviations because they have imperfect geometry, joints and connections to ground which are difficult to model accurately, and variations in material and mass properties.

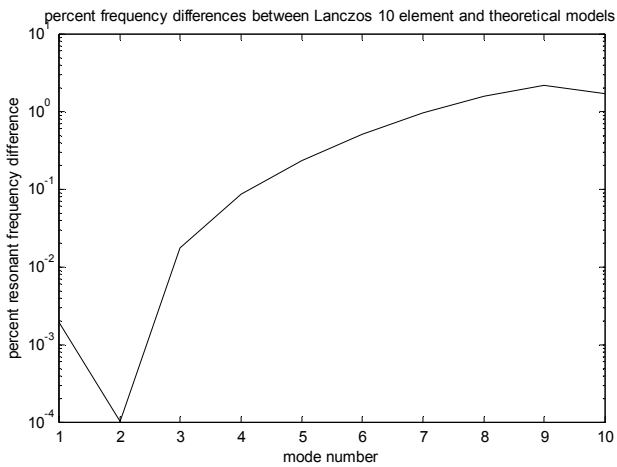

Figure 15.2: Percent resonant frequency differences between 10-element Block Lanczos ANSYS model and theoretical versus mode number.

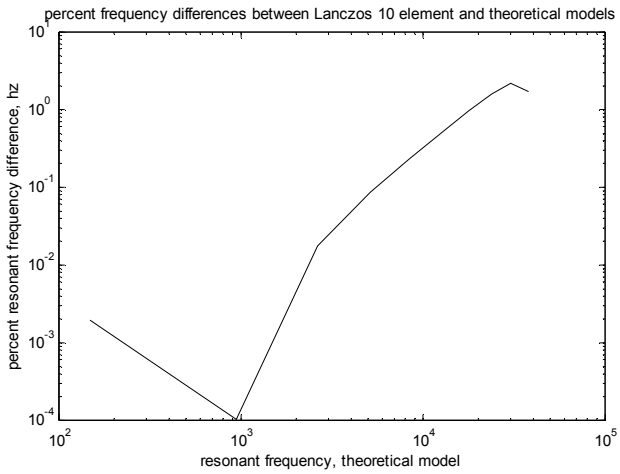

Figure 15.3: Percent resonant frequency differences between 10-element Block Lanczos ANSYS model and theoretical versus frequency.

15.4 ANSYS 10-element Model Eigenvalue/Eigenvector Summary

***** INDEX OF DATA SETS ON RESULTS FILE *****

SET	TIME/FREQ	LOAD STEP	SUBSTEP	CUMULATIVE
1	149.20	1	1	1
2	935.05	1	2	2
3	2619.0	1	3	3
4	5138.4	1	4	4
5	8521.2	1	5	5
6	12820.	1	6	6
7	18152.	1	7	7
8	24677.	1	8	8
9	32229.	1	9	9
10	39191.	1	10	10

Table 15.1: Frequency listing from cantbeam10red.frq file – frequencies for all 10 modes, hz.

In Table 15.2 we can see the eigenvector listing for the first two modes from the edited cantbeam10red.eig file, which contains information for all nodes for all 10 modes. As discussed in Section 7.4.2, ANSYS normalizes eigenvectors with respect to mass by default. Since our problem has input applied at the middle node (node 7), and output at the tip node (node 11), only those two nodes are required for the MATLAB model. We can choose to use ANSYS to output only the eigenvectors for nodes 7 and 11 or we can input the complete

modal matrix below in MATLAB and choose the appropriate rows of data within MATLAB.

```
SET COMMAND GOT LOAD STEP=     1  SUBSTEP=     1  CUMULATIVE ITERATION=
1
  TIME/FREQUENCY=  149.20
 TITLE= cantbeam, 10, red

 PRINT DOF  NODAL SOLUTION PER NODE

  ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=     1  SUBSTEP=     1
  FREQ=    149.20     LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN GLOBAL COORDINATES

  NODE       UX          UY          UZ          ROTX        ROTY        ROTZ
     1     0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
     2     0.0000      0.0000      6.9217      0.0000     -6.7553      0.0000
     3     0.0000      0.0000      26.357      0.0000     -12.514      0.0000
     4     0.0000      0.0000      56.320      0.0000     -17.287      0.0000
     5     0.0000      0.0000      94.863      0.0000     -21.099      0.0000
     6     0.0000      0.0000      140.11      0.0000     -23.997      0.0000
     7     0.0000      0.0000      190.29      0.0000     -26.054      0.0000
     8     0.0000      0.0000      243.83      0.0000     -27.371      0.0000
     9     0.0000      0.0000      299.37      0.0000     -28.085      0.0000
    10     0.0000      0.0000      355.87      0.0000     -28.358      0.0000
    11     0.0000      0.0000      412.66      0.0000     -28.407      0.0000

 MAXIMUM ABSOLUTE VALUES
 NODE        0           0           11          0           11          0
 VALUE     0.0000      0.0000      412.66      0.0000     -28.407      0.0000

 *ENDDO  INDEX= I

  ***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

 LOAD STEP=     1  SUBSTEP=     2
  FREQ=    935.05     LOAD CASE=   0

 THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN GLOBAL COORDINATES

 NODE       UX          UY          UZ          ROTX        ROTY        ROTZ
     1     0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
     2     0.0000      0.0000     -38.227      0.0000      34.605      0.0000
     3     0.0000      0.0000     -124.24      0.0000      47.942      0.0000
     4     0.0000      0.0000     -217.13      0.0000      41.980      0.0000
     5     0.0000      0.0000     -282.06      0.0000      20.864      0.0000
     6     0.0000      0.0000     -294.52      0.0000     -9.3483      0.0000
     7     0.0000      0.0000     -243.27      0.0000     -41.660      0.0000
     8     0.0000      0.0000     -130.84      0.0000     -69.535      0.0000
     9     0.0000      0.0000      28.911      0.0000     -88.467      0.0000
    10     0.0000      0.0000      216.16      0.0000     -97.088      0.0000
```

11	0.0000	0.0000	**412.70**	0.0000	-98.864	0.0000
MAXIMUM ABSOLUTE VALUES						
NODE	0	0	11	0	11	0
VALUE	0.0000	0.0000	412.70	0.0000	-98.864	0.0000

Table 15.2: Eigenvector listing for first two modes from the edited cantbeam10red.eig file.

15.5 Modal Matrix

The ANSYS output file cantbeam10red.eig can be sorted for only the UZ component for all 10 modes and put into a modal matrix form using **ext56uz.m** (see Appendix 1 for usage), as shown in Table 15.3. Each of the 10 columns in Table 5.3 represents the eigenvector for that mode, normalized with respect to mass. Compare the first two columns below with the bold "UZ" entries in the eigenvector listings in Table 15.2.

Columns 1 through 7

0	**0**	0	0	0	0	0
6.9217	**-38.2270**	94.1860	-159.3800	223.8100	-279.2100	320.1800
26.3570	**-124.2400**	249.6400	-311.9600	274.3700	-141.0000	-47.3120
56.3200	**-217.1300**	312.2800	-179.4100	-88.5050	283.0200	-246.1700
94.8630	**-282.0600**	217.1400	130.8000	-289.9000	45.1810	273.9500
140.1100	**-294.5200**	**8.0768**	292.6500	**1.1237**	-298.4100	**-1.2392**
190.2900	**-243.2700**	-195.6800	134.8400	291.6800	48.3890	-272.6900
243.8300	**-130.8400**	-271.4700	-164.7400	93.0350	284.4600	248.4900
299.3700	**28.9110**	-162.9800	-266.0000	-250.3900	-130.1600	52.2360
355.8700	**216.1600**	94.5080	-20.9260	-121.0900	-202.6200	-264.3300
412.6600	**412.7000**	413.1400	414.9000	419.7700	430.4900	449.0700

Columns 8 through 10

0	0	0
-341.5400	326.0200	223.4200
230.8300	-338.9500	-276.3500
-4.9143	283.7200	323.0000
-225.7900	-156.5300	-350.7100
320.6200	**-9.8888**	357.9400
-222.7800	173.8500	-344.2300
-8.8232	-294.1600	310.4200
237.8700	342.2200	-259.2800
-302.0500	-291.4900	187.8300
467.0400	439.1200	-282.9400

Table 15.3: Eigenvectors for UZ component of cantbeam10red.eig file.

The 11 rows represent the normalized displacements for the 11 nodes, starting with node 1 at the built-in end and node 11 at the tip. Editing the modal

matrix to use only the required degrees of freedom (nodes 7 and 11) will take place in MATLAB.

15.6 MATLAB State Space Model from ANSYS Eigenvalue Run – cantbeam_ss_modred.m

In this section we will create a MATLAB state space model using the eigenvalue and eigenvector results from the previous ANSYS run. We discussed in Section 7.9 how to decrease the size of the model by including only degrees of freedom actually used in the particular frequency response or time domain calculations. The new material deals with **how to rank the relative importance of the contributions of each of the individual modes**. In this chapter, we will use a **ranking of dc gains** of individual modes to select the modes to be used.

Once the modes are ranked, the most important can be selected for use, with modes with lower dc gains (typically, but not always, the higher frequency modes) eliminated from the model. When these modes are eliminated from the model their dc gain contributions are not included in the overall dc gain, so there is error in the low frequency gain. In order to eliminate this error, the MATLAB function "modred" is introduced and the theory behind the code is discussed. Using "modred" is analogous to using Guyan reduction to reduce some less important degrees of freedom, in that assumptions are made about some modes being more important than others. This allows reducing the size of the problem to that of the "important" modes, while adjusting the overall dc gain to account for the dc gains of the eliminated modes.

We will find that the simple cantilever beam used for an example in this chapter is not very sensitive to the elimination of higher frequency modes. Including a few modes is sufficient for creating a state space model with good accuracy for both frequency response and step response. Whether "modred" is used is not critical for this example. However, we will see that the example in the next chapter is extremely sensitive to dc gain, and will serve as a good model of the benefits of selecting modes to be eliminated judiciously or by using "modred."

Once the model is created, we will solve for frequency response and step response using various combinations of truncating and sorting modes.

The MATLAB code **cantbeam_ss_modred.m** will be discussed and listed in detail in the following sections.

15.6.1 Input

The code in this section asks the user to define how many elements will be used for the analysis. ANSYS runs have been made for 2, 4, 6, 8, 10, 12, 16, 32 and 64 elements. The ANSYS eigenvector results for each have been stripped out of the ANSYS format and put into frequency vector, "freqvec," and modal matrix, "evr," form and stored as MATLAB .mat files.

```
%       cantbeam_ss_modred.m

        clear all;

        hold off;

        clf;

%       load the .mat file cantbeamXXred, containing evr - the modal matrix, freqvec -
%       the frequency vector and node_numbers - the vector of node numbers for the modal
%       matrix

        model = menu('choose which finite element model to use ... ', ....
                                        '2 beam elements', ...
                                        '4 beam elements', ...
                                        '6 beam elements', ...
                                        '8 beam elements', ...
                                        '10 beam elements', ...
                                        '12 beam elements', ...
                                        '16 beam elements', ...
                                        '32 beam elements', ...
                                        '64 beam elements');

        if  model == 1
                load cantbeam2red;
        elseif  model == 2
                load cantbeam4red;
        elseif  model == 3
                load cantbeam6red;
        elseif  model == 4
                load cantbeam8red;
        elseif  model == 5
                load cantbeam10red;
        elseif  model == 6
                load cantbeam12red;
        elseif  model == 7
                load cantbeam16red;
        elseif  model == 8
                load cantbeam32red;
        elseif  model == 9
                load cantbeam64red;
        end
```

15.6.2 Defining Degrees of Freedom and Number of Modes

The code below checks the size of the modal matrix, where the number of rows indicates how many degrees of freedom are used and the number of columns indicates the number of modes. Since all of the models have an even number of elements, there is always a node at the midpoint of the beam and it is possible to define which row of the modal matrix corresponds to that middle node. The modal matrix row which corresponds to the tip is the last degree of freedom in the matrix. The code also defines a new variable, "xn," the normalized modal matrix.

```
%       define the number of degrees of freedom and number of modes from size
%       of modal matrix

        [numdof,num_modes_total] = size(evr);

%       define rows for middle and  tip nodes

        mid_node_row = 0.5*(numdof-1)+1;

        tip_node_row = numdof;

        xn = evr;
```

15.6.3 Sorting Modes by dc Gain and Peak Gain, Selecting Modes Used

The next step in creating the model is to sort modes of vibration so that only the most important modes are kept. We will discuss in this section two methods of sorting, one which is applicable for models with the same value of damping for all modes, $\zeta_i = \zeta$ = constant ("uniform" damping), and another which is applicable for models with different damping values for each mode ("non-uniform" damping).

Repeating from (8.54a,b) the general equation for the overall transfer function of undamped and damped systems:

$$\frac{z_j}{F_k} = \sum_{i=1}^{m} \frac{z_{nji} z_{nki}}{s^2 + \omega_i^2}$$

$$\frac{z_j}{F_k} = \sum_{i=1}^{m} \frac{z_{nji} z_{nki}}{s^2 + 2\zeta_i \omega_i s + \omega_i^2} \qquad (15.1a,b)$$

This equation shows that in general every transfer function is made up of additive combinations of single degree of freedom systems, with each system

having its residue determined by the appropriate input/output eigenvector entries, $z_{nji} z_{nki}$, and with resonant frequency defined by the eigenvalue, ω_i. Substituting $s = j\omega = j0 = 0$ to obtain the i^{th} mode frequency response at dc, the **dc gain**, which is the same for the undamped and damped cases is:

$$\frac{z_{ji}}{F_{ki}} = \frac{z_{nji} z_{nki}}{\omega_i^2}, \tag{15.2}$$

where $z_{nji} z_{nki}$ is the product of the jth (output) row and kth (force applied) row terms of the ith eigenvector divided by the square of the eigenvalue for the ith mode.

At resonance, the **peak gain** amplitude of each mode is given by substituting $s = j\omega_i, s^2 = -\omega_i^2$ into (15.1b):

$$\begin{aligned}
\frac{z_{ji}}{F_{ki}} &= \frac{z_{nji} z_{nki}}{s^2 + 2\zeta_i \omega_i s + \omega_i^2} \\
&= \frac{z_{nji} z_{nki}}{-\omega_i^2 + 2\zeta_i \omega_i^2 \, j + \omega_i^2} \\
&= \frac{z_{nji} z_{nki}}{2\zeta_i \omega_i^2 \, j} \\
&= \frac{-j \, z_{nji} z_{nki}}{2\zeta_i \omega_i^2} \\
&= \frac{-j}{2\zeta_i} \left(\frac{z_{nji} z_{nki}}{\omega_i^2} \right) \\
&= \frac{-j}{2\zeta_i} (\text{dc gain})
\end{aligned} \tag{15.3}$$

Comparing (15.2) and (15.3) it is evident that the relationship between the dc gain and peak gain for a mode is that the dc gain term is divided by 2ζ and multiplied by "$-j$," which gives a -90° phase shift at resonance. Since ζ values for mechanical structures are typically small, a few percent of critical damping, 2ζ is a small number, which serves to amplify the response by virtue of the division, thus the resonant "peak" in the response.

If the same value of ζ is used for all modes, then all the dc gain terms are divided by the same 2ζ terms and the relative amplitudes of the dc gains and

peak gains are the same, so there is no difference between sorting a uniform damping model using dc gain or peak gain.

However, if the modes have different damping the relationship between the dc gain and peak gain for all the modes is not a constant $1/2\zeta$ value and peak gain must be used to rank modes for importance. In this case, the MATLAB damping parameter "zeta" would not be a scalar but would be a vector with entries corresponding to damping in each mode.

We will use dc gain to rank the relative importance of the modes until Chapter 18, where a technique named "balanced reduction" will be introduced. The code shown below, and throughout the book, is easily modified to sort for peak gain instead of dc gain using (15.3) instead of (15.2) and entering a vector of damping values instead of a scalar.

The code below carries out the calculation of the dc gain and sorts from smallest to largest, keeping track of the new column locations in "index_sort." It then uses the "fliplr" command to list them from largest to smallest, so that the first mode has the highest dc gain. Various plots are then shown to indicate the relative importance of each mode. After plotting the dc gains, the user is asked to define the number of modes to be used in the frequency response, from 1 to all the available modes.

```
%   calculate the dc amplitude of the displacement of each mode by
%   multiplying the forcing function row of the eigenvector by the output row

            omega2 = (2*pi*freqvec)'.^2;     %  convert to radians and square

            dc_gain = abs(xn(mid_node_row,:).*xn(tip_node_row,:))./omega2;

            [dc_gain_sort,index_sort] = sort(dc_gain);

            dc_gain_sort = fliplr(dc_gain_sort);

            index_sort = fliplr(index_sort)

            dc_gain_nosort = dc_gain;

            index_orig = 1:num_modes_total;

            semilogy(index_orig,freqvec,'k-');
            title('frequency versus mode number')
            xlabel('mode number')
            ylabel('frequency, hz')
            grid
            pause

            semilogy(index_orig,dc_gain_nosort,'k-')
            title('dc value of each mode contribution versus mode number')
```

```
xlabel('mode number')
ylabel('dc value')
grid off
pause

loglog(freqvec,dc_gain_nosort,'k-')
title('dc value of each mode contribution versus frequency')
xlabel('frequency, hz')
ylabel('dc value')
grid off
pause

semilogy(index_orig,dc_gain_sort,'k-')
title('sorted dc value of each mode versus number of modes included')
xlabel('modes included')
ylabel('sorted dc value')
grid off
pause

num_modes_used = input(['enter how many modes to include …
                , ',num2str(num_modes_total),' default, max ...  ']);

 if (isempty(num_modes_used))
          num_modes_used = num_modes_total;
end
```

The first step in any finite element analysis is to understand the resonant frequencies of the model and how they relate to the frequency range of interest for the problem at hand.

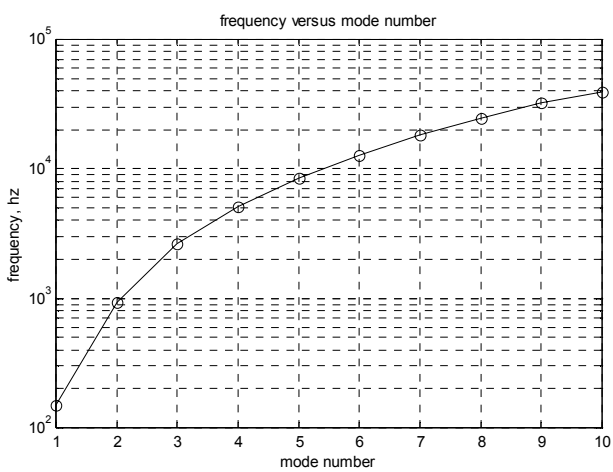

Figure 15.4: Resonant frequency versus mode number.

Figure 15.4 shows that modes 8, 9 and 10 have frequencies higher than the required 20 khz required by the problem, so our model should be adequate.

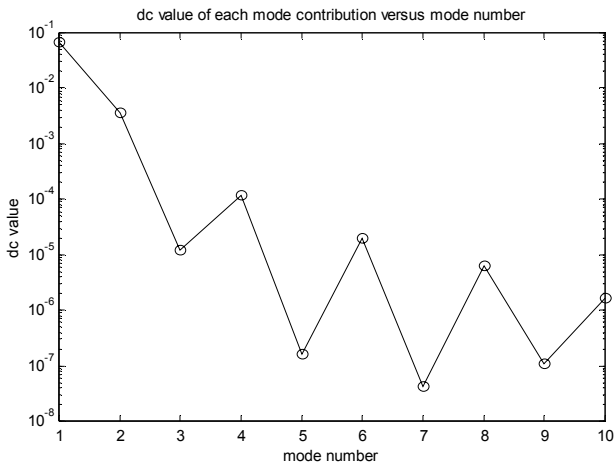

Figure 15.5: dc value of each mode contribution versus mode number.

Figure 15.5 shows the dc gain values for all the modes plotted versus mode number. It is interesting that the low values for modes 3, 5, 7 and 9 correspond to small values of the midpoint node elements of the respective eigenvectors (see the bold highlighted entries in columns 3, 5, 7 and 9 in Table 15.3). This means that the midpoint is nearly a "node" for those modes. Again, a "node" for a mode refers not to the number of the end point of the element but a location along the beam where the displacement is zero for a particular mode of vibration.

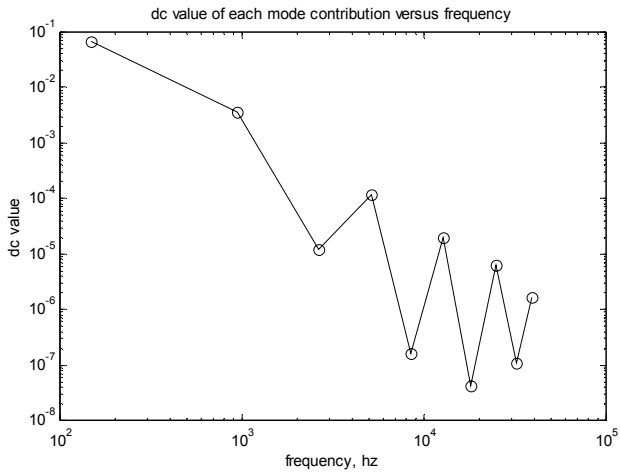

Figure 15.6: dc Value of each mode contribution versus resonant frequency.

Figure 15.6 shows dc gain versus frequency of the mode. Note that there is a general trend for lower gains as frequency increases. This is not always the case, as we shall see in Chapter 16.

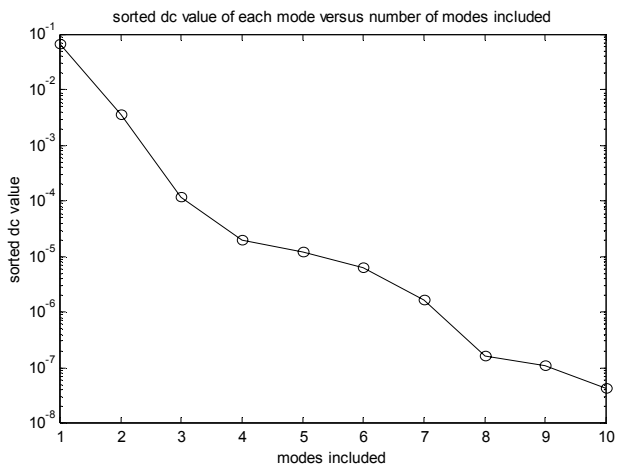

Figure 15.7: Sorted dc value of each mode versus number of modes included.

Figure 15.7 shows the sorted values for the dc gains, from largest to smallest. The list of mode numbers after sorting is given by "index_sort" below. The ordering can be seen in the dc value versus mode number plot in Figure 15.5.

```
index_sort =    1    2    4    6    3    8   10    5    9    7
```

15.6.4 Damping, Defining Reduced Frequencies and Modal Matrices

The section below asks for the damping value and whether to use the original ordering of modes or the modes sorted by dc gain. At this point, three different sets of modal matrices and eigenvalue vectors will be defined. The first set uses all the modes and frequencies and keeps them in their original, unsorted order. This set will be used to calculate frequency and step responses of the non-reduced model for comparison. The second set uses only the "num_modes_used" number of modes and keeps them in their original, unsorted order. This set will be used to see the effects of a simple truncation of higher frequency modes without sorting or ranking. The third set again uses the "num_modes_used" number of modes but includes only the modes with the highest dc gains. We will calculate frequency response and transient response results for both of the reduced cases and compare results with the "all modes included" case. The two reduced models are denoted with the "_nosort" and "_sort" suffixes throughout the code. We will see that because the dc gain values for this model generally decrease with frequency, the sorted and unsorted models will give almost the same results. The example in the next chapter, however, will not have this property.

```
        zeta = input('enter value for damping, .02 is 2% of critical (default) ... ');

        if (isempty(zeta))
            zeta = .02;
        end

%       all modes included model, use original order

        xnnew = xn(:,(1:num_modes_total));

        freqnew = freqvec((1:num_modes_total));

%       reduced, no sorting, just use the first num_modes_used modes in xnnew_nosort

        xnnew_nosort = xn(:,1:num_modes_used);

        freqnew_nosort = freqvec(1:num_modes_used);

%       reduced, sorting, use the first num_modes_used sorted modes in xnnew_sort

        xnnew_sort = xn(:,index_sort(1:num_modes_used));

        freqnew_sort = freqvec(index_sort(1:num_modes_used));
```

15.6.5 Setting up System Matrix "a"

The section below sets up three state space system "a" matrices. Since we know the form of the modal form state space equation from Chapter 10, it can be built automatically. The general form is given by (15.4). The system matrix is made up of eigenvalue and damping terms for each mode, and each mode is a 2x2 submatrix along the diagonal.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{15.4}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dots \\ \dots \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & \dots \\ -\omega_1^2 & -2\zeta_1\omega_1 & 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & 1 & \dots & \dots \\ 0 & 0 & -\omega_2^2 & -2\zeta_2\omega_2 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \dots \\ \dots \end{bmatrix} + \begin{bmatrix} 0 \\ F_{p1} \\ 0 \\ F_{p2} \\ \dots \\ \dots \end{bmatrix} u \tag{15.5}$$

The first system matrix, "a," is for the full, non-reduced system and includes all the modes in their original order. The second is "a_nosort" and has the reduced size with the original ordering of modes. The third is "a_sort" and has the reduced number of modes with dc gain ordering.

```
%       define variables for all modes included system matrix, a
        w = freqnew*2*pi;               %       frequencies in rad/sec
        w2 = w.^2;
        zw = 2*zeta*w;
%       define variables for reduced, nosorted system matrix, a_nosort
        w_nosort = freqnew_nosort*2*pi;         %       frequencies in rad/sec
        w2_nosort = w_nosort.^2;
        zw_nosort = 2*zeta*w_nosort;
%       define variables for reduced, sorted system matrix, a_sort
        w_sort = freqnew_sort*2*pi;             %       frequencies in rad/sec
        w2_sort = w_sort.^2;
        zw_sort = 2*zeta*w_sort;
%       define size of system matrix
```

```
asize = 2*num_modes_total;

asize_red = 2*num_modes_used;

disp('   ');
disp('   ');
disp(['size of system matrix a is ',num2str(asize)]);
disp(['size of reduced system matrix a is ',num2str(asize_red)]);

%       setup all modes included "a" matrix, system matrix

a = zeros(asize);

for  col = 2:2:asize

row = col-1;

a(row,col) = 1;

end

for  col = 1:2:asize

row = col+1;

a(row,col) = -w2((col+1)/2);

end

for  col = 2:2:asize

row = col;

a(row,col) = -zw(col/2);

end

%       setup reduced, nosorted "a_nosort" matrix, system matrix

a_nosort = zeros(asize_red);

for  col = 2:2:asize_red

row = col-1;

a_nosort(row,col) = 1;

end

for  col = 1:2:asize_red

row = col+1;

a_nosort(row,col) = -w2_nosort((col+1)/2);
```

```
            end
            for  col = 2:2:asize_red
            row = col;
            a_nosort(row,col) = -zw_nosort(col/2);
            end
%           setup reduced, sorted "a_sort" matrix, system matrix
            a_sort = zeros(asize_red);
            for  col = 2:2:asize_red
            row = col-1;
            a_sort(row,col) = 1;
            end
            for  col = 1:2:asize_red
            row = col+1;
            a_sort(row,col) = -w2_sort((col+1)/2);
            end
            for  col = 2:2:asize_red
            row = col;
            a_sort(row,col) = -zw_sort(col/2);
            end
```

15.6.6 Setting up Input Matrix "b"

As with the system matrix above, here we will set up three different input matrices, "b," "b_nosort" and "b_sort." We begin with the force vector in physical coordinates, with "numdof" rows. The rows are all zeros except for the "mid_node_row," which has a value of 1.0 mN. The force vector in principal coordinates is obtained by premultiplying by the transpose of the modal matrix. The state space form of the force vector in principal coordinates is the "numdof x 1" force vector in principal coordinates padded with zeros to create the same number of rows as states.

```
%       setup input matrix b, state space forcing function in principal coordinates

%       f_physical is the vector of physical force
%       zeros at each output DOF and input force at the input DOF

        f_physical = zeros(numdof,1);     %          start out with zeros

        f_physical(mid_node_row) = 1.0;           %   input force at node 6, midpoint node

%       f_principal is the vector of forces in principal coordinates

        f_principal = xnnew'*f_physical;

%       b is the vector of forces in principal coordinates, state space form

        b = zeros(2*num_modes_total,1);

        for  cnt = 1:num_modes_total

                b(2*cnt) = f_principal(cnt);

        end

%       f_principal_nosort is the vector of forces in principal coordinates

        f_principal_nosort = xnnew_nosort'*f_physical;

%       b_nosort is the vector of forces in principal coordinates, state space form

        b_nosort = zeros(2*num_modes_used,1);

        for  cnt = 1:num_modes_used

                b_nosort(2*cnt) = f_principal_nosort(cnt);

        end

%       f_principal_sort is the vector of forces in principal coordinates

        f_principal_sort = xnnew_sort'*f_physical;

%       b_sort is the vector of forces in principal coordinates, state space form

        b_sort = zeros(2*num_modes_used,1);

        for  cnt = 1:num_modes_used

                b_sort(2*cnt) = f_principal_sort(cnt);

        end
```

15.6.7 Setting up Output Matrix "c" and Direct Transmission Matrix "d"

The output matrices below, "c," "c_nosort" and "c_sort," are separated into displacement and velocity matrices, "cdisp" and "cvel," so that they can be premultiplied by the appropriate modal matrix to obtain vectors of displacements and velocities in physical coordinates. With the defined output displacement and velocity matrices, all displacement and velocity degrees of freedom in physical coordinates are available for plotting or further analysis. Since there is no direct feedthrough on this model, the "d" matrix is zero.

```
%       setup cdisp and cvel, padded xn matrices to give the displacement and velocity
%       vectors in physical coordinates
%       cdisp and cvel each have numdof rows and alternating columns consisting of columns
%       of xnnew and zeros to give total columns equal to the number of states

%       all modes included cdisp and cvel

        for  col = 1:2:2*length(freqnew)

        for  row = 1:numdof

        cdisp(row,col) = xnnew(row,ceil(col/2));

        cvel(row,col) = 0;

        end

        end

        for  col = 2:2:2*length(freqnew)

        for  row = 1:numdof

        cdisp(row,col) = 0;

              cvel(row,col) = xnnew(row,col/2);

        end

        end

%       reduced, nosorted cdisp and cvel

        for  col = 1:2:2*length(freqnew_nosort)

        for  row = 1:numdof

        cdisp_nosort(row,col) = xnnew_nosort(row,ceil(col/2));

        cvel_nosort(row,col) = 0;
```

```
            end
            end
            for  col = 2:2:2*length(freqnew_nosort)
            for  row = 1:numdof
            cdisp_nosort(row,col) = 0;
                  cvel_nosort(row,col) = xnnew_nosort(row,col/2);
            end
            end
%           reduced, sorted cdisp and cvel
            for  col = 1:2:2*length(freqnew_sort)
            for  row = 1:numdof
            cdisp_sort(row,col) = xnnew_sort(row,ceil(col/2));
            cvel_sort(row,col) = 0;
            end
            end
            for  col = 2:2:2*length(freqnew_sort)
            for  row = 1:numdof
            cdisp_sort(row,col) = 0;
                  cvel_sort(row,col) = xnnew_sort(row,col/2);
            end
            end
%           define output
            d = [0];  %
```

15.6.8 Frequency Range, "ss" Setup, Bode Calculations

The first part of this section defines the frequency range to be used for the frequency responses, logarithmically spaced frequency vectors in units of hz and rad/sec. Three "ss" state space systems are defined for the displacement of the tip of the beam, the non-reduced system, the "nosort" and the "sort." Since "cdisp" contains information about all the degrees of freedom, they are all available for output by defining the appropriate row. The "bode" command is used to calculate the magnitude and phase vectors over the defined frequency range, and the magnitudes are converted to db.

```
%       define frequency vector for frequency responses

        freqlo = 10;

        freqhi = 100000;

        flo=log10(freqlo) ;
        fhi=log10(freqhi) ;

        f=logspace(flo,fhi,200) ;
        frad=f*2*pi ;

%       take transfer functions, outputting the midpoint and tip node rows of the displacement
%       vector cdisp

%       define displacement state space system with the "ss" command

        sysdisptip = ss(a,b,cdisp(tip_node_row,:),d);

%       defined reduced systems using num_modes_used nosort modes

        sysdisptip_nosort = ss(a_nosort,b_nosort,cdisp_nosort(tip_node_row,:),d);

%       define reduced systems using num_modes_used sorted modes

        sysdisptip_sort = ss(a_sort,b_sort,cdisp_sort(tip_node_row,:),d);

%       use "bode" command to generate magnitude/phase vectors

        [magdisptip,phsdisptip]=bode(sysdisptip,frad)  ;

        [magdisptip_nosort,phsdisptip_nosort]=bode(sysdisptip_nosort,frad)  ;

        [magdisptip_sort,phsdisptip_sort]=bode(sysdisptip_sort,frad)  ;

%       convert magnitude to db

        magdisptipdb = 20*log10(magdisptip);

        magdisptipdb_nosort = 20*log10(magdisptip_nosort);
```

```
magdisptipdb_sort = 20*log10(magdisptip_sort);
```

15.6.9 Full Model - Plotting Frequency Response, Step Response

This section plots the frequency response for tip displacement due to a unit force at the beam midpoint. It then overlays the contribution of each individual mode to the overall response. Since the "a" matrix consists of 2x2 submatrices along the diagonal, all we have to do to get the contribution of each individual mode is to pull out successive 2x2 individual mode system matrices. Similarly, we take the appropriate rows and columns of "b" and "cdisp" for each mode. Because of the systematic form of the matrices, MATLAB can generate the individual mode matrices automatically. To facilitate comparison with the dc gain values calculated for all the modes (and used in their sorting), an "o" is plotted along the left-hand axis for each individual mode. Because the magnitude axis is in db units, the individual contributions cannot be combined graphically like with a linear magnitude axis as shown in Chapter 6. Nevertheless, using the overlaid plots to get a mental image of the combining modes is valuable.

For the unit force step response, a time vector, "t" and input vector "u" are defined for use with the MATLAB function "lsim."

```
%       start plotting

        if  num_modes_used == num_modes_total

%       plot all modes included response

        semilogx(f,magdisptipdb(1,:),'k.-')
        title(['cantilever tip displacement for mid-length force, all ', …
                num2str(num_modes_used),' modes included'])
        xlabel('Frequency, hz')
        ylabel('Magnitude, db mm')
        grid off
        pause

        hold on

        max_modes_plot = num_modes_total;

        for  pcnt = 1:max_modes_plot

                index = 2*pcnt;

                amode = a(index-1:index,index-1:index);

                bmode = b(index-1:index);
```

```
                    cmode = cdisp(numdof,index-1:index);

                    dmode = [0];

                    sysdisptip_mode = ss(amode,bmode,cmode,dmode);

                    [magdisptip_mode,phsdisptip_mode]=bode(sysdisptip_mode,frad)  ;

                    magdisptip_modedb = 20*log10(magdisptip_mode);

                    semilogx(f,magdisptip_modedb(1,:),'k-')

          end

          dc_gain_freq = freqlo*ones(size(freqnew));

          semilogx(dc_gain_freq(1:num_modes_used),20*log10(dc_gain
                    (1:num_modes_used)),'ko:')

          pause

          hold off

%         now use lsim to calculate step response to a unit force

          ttotal = 0.1;

          t = linspace(0,ttotal,200);

          u = ones(size(t));

          [disptip,ts] = lsim(sysdisptip,u,t);

          plot(ts,disptip,'k-')
          title(['tip disp for mid-length step force, all ',num2str(num_modes_used), …
                    ' modes included'])
          xlabel('time, sec')
          ylabel('displacement, mm')
          grid off
          pause
```

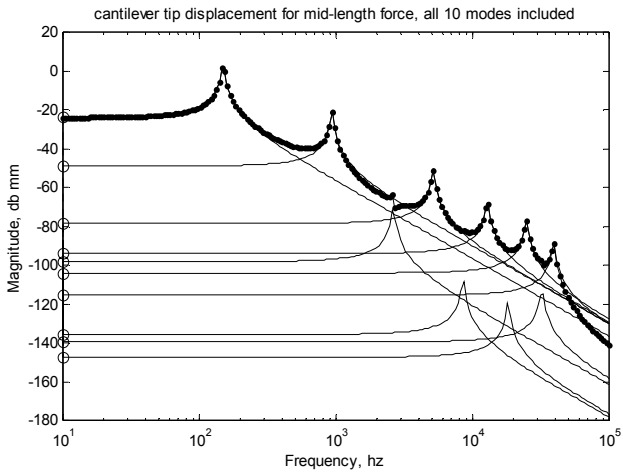

Figure 15.8: Cantilever tip displacement for mid-length force, all 10 modes included.

Figure 15.8 shows the overall frequency response with the overlaid sdof responses of all the individual modes for the 10-element model using all 10 available modes. The "o's" at the 10 hz frequency indicate the values of dc gain for each mode. Note that the fifth, seventh and ninth modes have such low gains that their resonant peaks are barely visible on the overall response. The third mode has a higher gain, as indicated by the small pole/zero combination between the second and fourth modes.

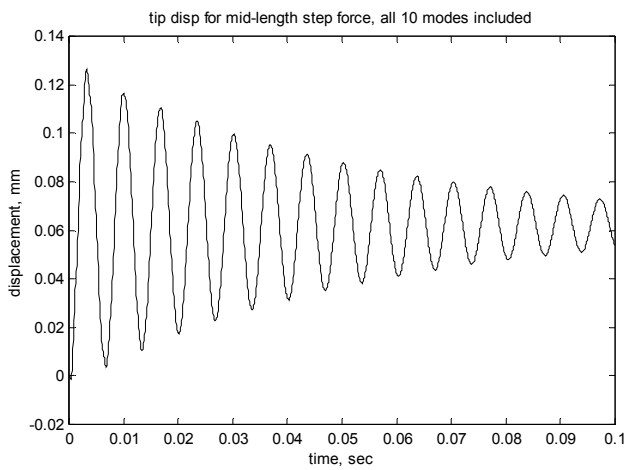

Figure 15.9: Cantilever tip displacement for mid-length force, all 10 modes included.

Figure 15.9 depicts the response of the beam tip due to a 1mN step force at the midpoint. We will be comparing the different modal truncation methods with this overall response.

15.6.10 Reduced Models – Plotting Frequency Response, Step Response

The following section of code does the same thing for the reduced unsorted and sorted models as the last section did for the full model. In all the plots, the full model results are overlaid with the reduced model results to show the differences. In the examples that follow, we will use four modes in the reduced models. The reader is encouraged to run the code using different numbers of reduced modes to see the effects on both frequency and time domain responses.

```
else

%       plot unsorted modal truncation

        semilogx(f,magdisptipdb(1,:),'k-',f,magdisptipdb_nosort(1,:),'k.-')
        title(['unsorted modal truncation:  cantilever tip displacement for mid- …
                length force, first ',num2str(num_modes_used),' modes included'])
        legend('all modes','unsorted partial modes',3)

        dcgain_error_percent_nosort = 100*(magdisptip_nosort(1) …
                magdisptip(1))/magdisptip(1)

        xlabel('Frequency, hz')
        ylabel('Magnitude, db mm')
        grid off

        pause

        hold on

        max_modes_plot = num_modes_used;

        for  pcnt = 1:max_modes_plot

                index = 2*pcnt;

                amode = a_nosort(index-1:index,index-1:index);

                bmode = b_nosort(index-1:index);

                cmode = cdisp_nosort(numdof,index-1:index);

                dmode = [0];

                sysdisptip_mode = ss(amode,bmode,cmode,dmode);

                [magdisptip_mode,phsdisptip_mode]=bode(sysdisptip_mode,frad)  ;
```

```
                magdisptip_modedb = 20*log10(magdisptip_mode);

                semilogx(f,magdisptip_modedb(1,:),'k-')

        end

        dc_gain_freq_nosort = freqlo*ones(size(freqnew_nosort));

        semilogx(dc_gain_freq_nosort(1:num_modes_used),20*log10 …
                (dc_gain_nosort(1:num_modes_used)),'ko:')

        pause

        hold off

%       plot sorted modal truncation

        semilogx(f,magdisptipdb(1,:),'k-',f,magdisptipdb_sort(1,:),'k.-')
        title(['sorted modal truncation:  cantilever tip displacement for mid-length force, …
                first ',num2str(num_modes_used),' modes included'])
        legend('all modes','sorted partial modes',3)

        dcgain_error_percent_sort = 100*(magdisptip_sort(1) - magdisptip(1))/magdisptip(1)

        xlabel('Frequency, hz')
        ylabel('Magnitude, db mm')
        grid off

        pause

        hold on

        max_modes_plot = num_modes_used;

        for  pcnt = 1:max_modes_plot

                index = 2*pcnt;

                amode = a_sort(index-1:index,index-1:index);

                bmode = b_sort(index-1:index);

                cmode = cdisp_sort(numdof,index-1:index);

                dmode = [0];

                sysdisptip_mode = ss(amode,bmode,cmode,dmode);

                [magdisptip_mode,phsdisptip_mode]=bode(sysdisptip_mode,frad)  ;

                magdisptip_modedb = 20*log10(magdisptip_mode);

                semilogx(f,magdisptip_modedb(1,:),'k-')
```

```
        end

        dc_gain_freq_sort = freqlo*ones(size(freqnew_nosort));

        semilogx(dc_gain_freq_sort(1:num_modes_used),20*log10 …
                        (dc_gain_sort(1:num_modes_used)),'ko:')

        pause

        hold off

%       now use lsim to calculate step response to a unit force

        ttotal = 0.1;

        t = linspace(0,ttotal,200);

        u = ones(size(t));

        [disptip,ts] = lsim(sysdisptip,u,t);

        [disptip_nosort,ts_nosort] = lsim(sysdisptip_nosort,u,t);

        [disptip_sort,ts_sort] = lsim(sysdisptip_sort,u,t);

        plot(ts,disptip,'k-',ts_nosort,disptip_nosort,'k+-',ts_sort,disptip_sort,'k.-')
        title(['tip disp for mid-length step force, first ',num2str(num_modes_used) …
                        ,' modes included'])
        legend('all modes','unsorted partial modes','sorted partial modes')
        xlabel('time, sec')
        ylabel('displacement, mm')
        grid off
        pause
```

15.6.11 Reduced Models – Plotted Results – Four Modes Used

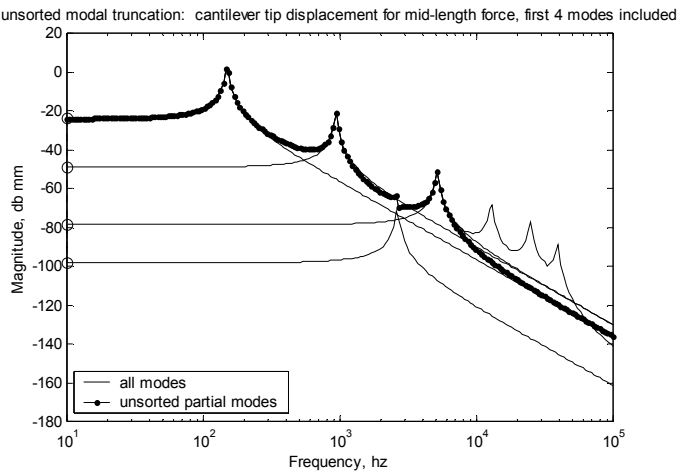

Figure 15.10: Cantilever tip displacement for mid-length force, first four modes included – unsorted modal truncation.

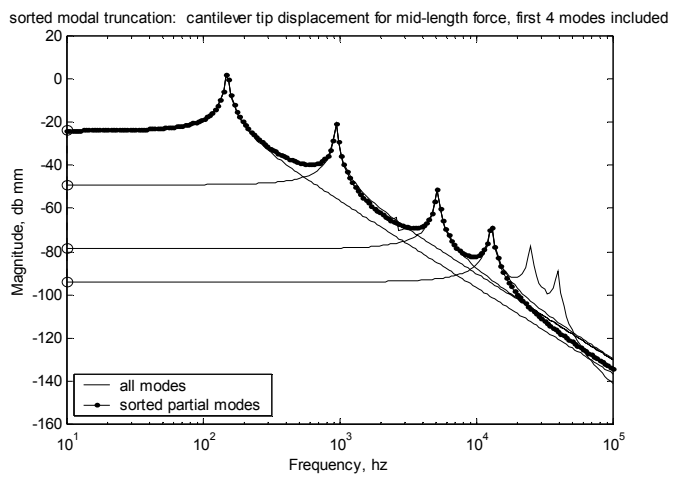

Figure 15.11: Cantilever tip displacement for mid-length force, first four modes included – sorted modal truncation.

Figure 15.10 depicts overall plus individual mode contributions for the four unsorted modes model. Note that the first four unsorted modes are used. The dc gain error relative to the full 10-mode model is +0.024% because the dc

gain terms for the eliminated modes are not included. Note that the last three peaks in the "all modes" response are missed because the modes are not included.

Figure 15.11 shows overall plus individual mode contributions for the four sorted modes model. Note that this time the third mode is skipped and the fifth mode is used instead because it has a higher dc gain. The dc gain error relative to the full 10-mode model is –0.027%.

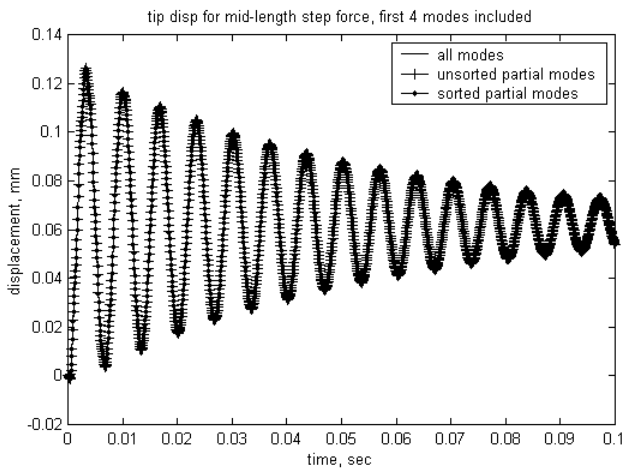

Figure 15.12: Comparison of step responses for all modes included and four modes included, unsorted and sorted.

Figure 15.12 shows step response for full, reduced unsorted and reduced sorted models. Because the dc gain for the two models is in error only by a fraction of a percent and because the eliminated modes are some 80db (four orders of magnitude) lower than the most significant first mode, there is no discernable difference in the responses of the full and two reduced models.

15.6.12 Modred Description

The MATLAB Control System Toolbox has a function, "modred" (MODel order REDuction), which can be used for reducing models while retaining the overall system dc gain. The "mdc" or "Matched DC" gain option for the function "modred" reduces defined states by setting the derivatives of the states to be eliminated to zero, then solving for the remaining states. The method essentially sets up the eliminated states to be "infinitely fast" and is analogous to Guyan reduction in that the low frequency effects of the eliminated states are included in the remaining states. The other option for

"modred" is the "del" option, which simply eliminates the defined states, typically associated with the higher frequency modes.

The derivation of the "mdc" option follows. We start with the state space description of the system:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned} \tag{15.6a,b}$$

Assume that we have a method of ordering the importance of the modes making up the **A**, **B** and **C** matrices, in our case using dc or peak gains. If we then rearrange and partition the matrices such that the states corresponding to the most important modes are separated from the less important modes, designating the important modes as $\mathbf{x}_r$ (reduced) and the unimportant modes to be eliminated as $\mathbf{x}_e$, we get

$$\begin{aligned} \begin{bmatrix} \dot{\mathbf{x}}_r \\ \dot{\mathbf{x}}_e \end{bmatrix} &= \begin{bmatrix} \mathbf{A}_{rr} & \mathbf{A}_{re} \\ \mathbf{A}_{er} & \mathbf{A}_{ee} \end{bmatrix} \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_e \end{bmatrix} + \begin{bmatrix} \mathbf{B}_r \\ \mathbf{B}_e \end{bmatrix} \mathbf{u} \\ \mathbf{y} &= \begin{bmatrix} \mathbf{C}_r & \mathbf{C}_e \end{bmatrix} \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_e \end{bmatrix} + \mathbf{D}\mathbf{u} \end{aligned} \tag{15.7a,b}$$

Expanding the matrices:

$$\begin{aligned} \dot{\mathbf{x}}_r &= \mathbf{A}_{rr}\mathbf{x}_r + \mathbf{A}_{re}\mathbf{x}_e + \mathbf{B}_r\mathbf{u} \\ \dot{\mathbf{x}}_e &= \mathbf{A}_{er}\mathbf{x}_r + \mathbf{A}_{ee}\mathbf{x}_e + \mathbf{B}_e\mathbf{u} \end{aligned} \tag{15.8a,b}$$

Setting the $\dot{\mathbf{x}}_e$ states equal to zero in (15.10) is analogous to setting (14.14) equal to zero in the Guyan reduction process. We are then, in effect, including the low frequency dc gain or static equilibrium characteristics of the eliminated modes in the reduced modes.

$$0 = \mathbf{A}_{er}\mathbf{x}_r + \mathbf{A}_{ee}\mathbf{x}_e + \mathbf{B}_e\mathbf{u} \tag{15.9}$$

Solving for $\mathbf{x}_e$:

$$\mathbf{x}_e = -\mathbf{A}_{ee}^{-1}\mathbf{A}_{er}\mathbf{x}_r - \mathbf{A}_{ee}^{-1}\mathbf{B}_e\mathbf{u} \tag{15.10}$$

Substituting back into the $\dot{\mathbf{x}}_r$ equation and grouping terms:

$$\begin{aligned}\dot{\mathbf{x}}_r &= \mathbf{A}_{rr}\mathbf{x}_r + \mathbf{A}_{re}\left(-\mathbf{A}_{ee}^{-1}\mathbf{A}_{er}\mathbf{x}_r - \mathbf{A}_{ee}^{-1}\mathbf{B}_e\mathbf{u}\right) + \mathbf{B}_r\mathbf{u} \\ &= \left(\mathbf{A}_{rr} - \mathbf{A}_{re}\mathbf{A}_{ee}^{-1}\mathbf{A}_{er}\right)\mathbf{x}_r + \left(\mathbf{B}_r - \mathbf{A}_{re}\mathbf{A}_{ee}^{-1}\mathbf{B}_e\right)\mathbf{u}\end{aligned} \tag{15.11}$$

Substituting back into the expanded output equations:

$$\begin{aligned}\mathbf{y} &= \mathbf{C}_r\mathbf{x}_r + \mathbf{C}_e\mathbf{x}_e + \mathbf{D}\mathbf{u} \\ &= \mathbf{C}_r\mathbf{x}_r + \mathbf{C}_e\left(-\mathbf{A}_{ee}^{-1}\mathbf{A}_{er}\mathbf{x}_r - \mathbf{A}_{ee}^{-1}\mathbf{B}_e\mathbf{u}\right) + \mathbf{D}\mathbf{u} \\ &= \left(\mathbf{C}_r - \mathbf{C}_e\mathbf{A}_{ee}^{-1}\mathbf{A}_{er}\right)\mathbf{x}_r + \left(\mathbf{D} - \mathbf{C}_e\mathbf{A}_{ee}^{-1}\mathbf{B}_e\right)\mathbf{u}\end{aligned} \tag{15.12}$$

The new matrices for the reduced model become:

$$\begin{aligned}\mathbf{A}_{red} &= \mathbf{A}_{rr} - \mathbf{A}_{re}\mathbf{A}_{ee}^{-1}\mathbf{A}_{er} \\ \mathbf{B}_{red} &= \mathbf{B}_r - \mathbf{A}_{re}\mathbf{A}_{ee}^{-1}\mathbf{B}_e \\ \mathbf{C}_{red} &= \mathbf{C}_r - \mathbf{C}_e\mathbf{A}_{ee}^{-1}\mathbf{A}_{er} \\ \mathbf{D}_{red} &= \mathbf{D} - \mathbf{C}_e\mathbf{A}_{ee}^{-1}\mathbf{B}_e\end{aligned} \tag{15.13a,b,c,d}$$

The new state equations are:

$$\begin{aligned}\dot{\mathbf{x}}_{red} &= \mathbf{A}_{red}\mathbf{x}_{red} + \mathbf{B}_{red}\mathbf{u} \\ \mathbf{y}_{red} &= \mathbf{C}_{red}\mathbf{x}_{red} + \mathbf{D}_{red}\mathbf{u}\end{aligned} \tag{15.14a,b}$$

We will see (Figure 15.14) that the high frequency portion of the response when reducing using "modred" does not roll off quickly with frequency as we are used to seeing. Rather, it will be "flat" with frequency. The reason for the shape of the "modred" high frequency asymptote is in the $\mathbf{D}_{red}$ term in (15.13d). In many cases, the direct transmission term $\mathbf{D}$ is zero. When using "modred," however, even if $\mathbf{D}$ is zero, there is still the $-\mathbf{C}_e\mathbf{A}_{ee}^{-1}\mathbf{B}_e$ portion of $\mathbf{D}_{red}$ to contend with. Repeating Figure 5.2 below, we can see the direct transmission term.

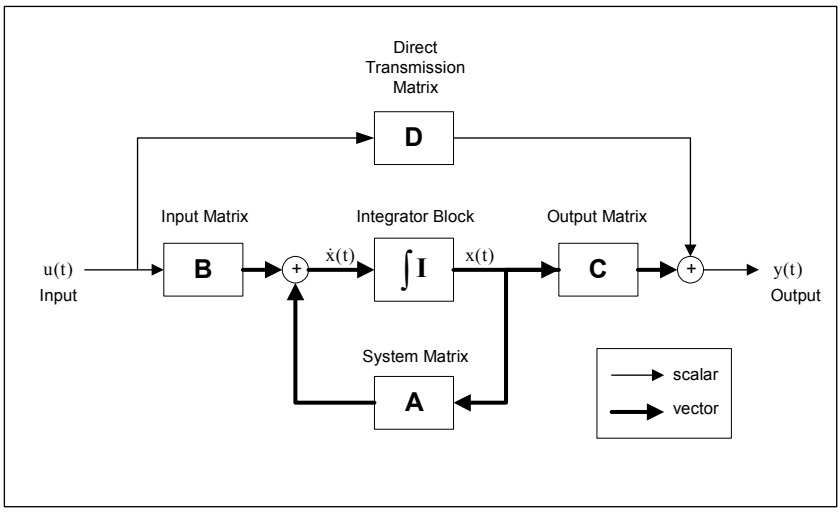

Figure 15.13: State space system block diagram.

At high frequencies, where the system matrix dynamics start to attenuate, the $-\mathbf{C}_e\mathbf{A}_{ee}^{-1}\mathbf{B}_e$ term of $\mathbf{D}_{red}$ starts to dominate the response – hence the "flat" high frequency response in Figure 15.15.

15.6.13 Defining Sorted or Unsorted Modes to be Used

The section of code below prompts the user to define whether the modes are to be sorted by dc gain or left in the original order for the "modred" operation. One argument of the "modred" command is to define the states to be eliminated. The states to be eliminated can be defined as a vector of arbitrary states or as a continuous partition of states. We will define them in the code below as a continuous block of states, from one index greater than the number of states to be kept to the total number of states. Therefore, if we sort by dc gain before using "modred," we would keep only the most important states. If we choose to use the unsorted states, we will be eliminating the higher frequency modes and keeping the lower frequency modes.

```
%       use modred to reduce, select whether to use sorted or unsorted modes for the reduction

        modred_sort = input('modred: enter "1" to use sorted modes for reduced runs, …
                    "enter" to use unsorted ... ');

        if  isempty(modred_sort)
                    modred_sort = 0
        end
```

```
if  modred_sort == 1                %  use sorted mode order
        xnnew = xn(:,index_sort(1:num_modes_total));
        freqnew = freqvec(index_sort(1:num_modes_total));
else                        %  use original mode order
        xnnew = xn(:,(1:num_modes_total));
        freqnew = freqvec((1:num_modes_total));
end
```

15.6.14 Defining System for Reduction

In this section we define a new set of "a," "b," "c" and "d" matrices which will be used with "modred."

```
%       define variables for all modes included system matrix, a
        w = freqnew*2*pi;           %       frequencies in rad/sec
        w2 = w.^2;
        zw = 2*zeta*w;
%       define size of system matrix
        asize = 2*num_modes_total;
%       setup all modes included "a" matrix, system matrix
        a = zeros(asize);
        for  col = 2:2:asize
        row = col-1;
        a(row,col) = 1;
        end
        for  col = 1:2:asize
        row = col+1;
        a(row,col) = -w2((col+1)/2);
        end
```

```
            for  col = 2:2:asize

            row = col;

            a(row,col) = -zw(col/2);

            end

%           setup input matrix b, state space forcing function in principal coordinates

%           f_physical is the vector of physical force
%           zeros at each output DOF and input force at the input DOF

            f_physical = zeros(numdof,1);                   %           start out with zeros

            f_physical(mid_node_row) = 1.0;                         %   input force at node
                        6,midpoint node

%           f_principal is the vector of forces in principal coordinates

            f_principal = xnnew'*f_physical;

%           b is the vector of forces in principal coordinates, state space form

            b = zeros(2*num_modes_total,1);

            for  cnt = 1:num_modes_total

                        b(2*cnt) = f_principal(cnt);

            end

%           setup cdisp and cvel, padded xn matrices to give the displacement and velocity
%           vectors in physical coordinates
%           cdisp and cvel each have numdof rows and alternating columns consisting of columns
%           of xnnew and zeros to give total columns equal to the number of states

%           all modes included cdisp and cvel

            for  col = 1:2:2*length(freqnew)

            for  row = 1:numdof

            cdisp(row,col) = xnnew(row,ceil(col/2));

            cvel(row,col) = 0;

            end

            end

            for  col = 2:2:2*length(freqnew)

            for  row = 1:numdof
```

```
            cdisp(row,col) = 0;
                  cvel(row,col) = xnnew(row,col/2);
            end
            end
%           define output
            d = [0];  %
```

15.6.15 Modred Calculations – "mdc" and "del"

This section defines a MATLAB state space, "ss," system using either the unsorted or sorted eigenvectors and eigenvalues from above, and then both the "mdc" and "del" options with "modred" to calculate two reduced systems. In order to be able to plot not only the overall frequency response from the reduced systems but also the individual mode contributions, we will use the "ssdata" function in MATLAB to define the reduced system matrices. In the next section we will use 2x2 submatrices of the reduced system matrix to define individual modal contributions. The "bode" command is then used to generate the magnitude/phase solution vectors, which are converted to db.

```
%           define state space system for reduction, ordered defined by modred_sort
            sysdisptip_red = ss(a,b,cdisp(tip_node_row,:),d);
%           define reduced matrices using matched dc gain method "mdc"
            states_elim = (2*num_modes_used+1):2*num_modes_total;
            sysdisptip_mdc = modred(sysdisptip_red,states_elim,'mdc');
            [adisptip_mdc,bdisptip_mdc,cdisptip_mdc,ddisptip_mdc] = ssdata(sysdisptip_mdc);
%           define reduced matrices by eliminating high frequency states, 'del
            sysdisptip_elim = modred(sysdisptip_red,states_elim,'del');
            [adisptip_elim,bdisptip_elim,cdisptip_elim,ddisptip_elim] = ssdata(sysdisptip_elim);
%           use "bode" command to generate magnitude/phase vectors for reduced systems
            [magdisptip_mdc,phsdisptip_mdc]=bode(sysdisptip_mdc,frad)  ;
            [magdisptip_elim,phsdisptip_elim]=bode(sysdisptip_elim,frad)  ;
%           convert magnitude to db
            magdisptip_mdcdb = 20*log10(magdisptip_mdc);
```

```
magdisptip_elimdb = 20*log10(magdisptip_elim);
```

15.6.16 Reduced Modred Models – Plotting Commands

This section plots the frequency responses with the individual mode contribution overlays for both the "mdc" and "del" options for "modred." The only difference between the code here and that of section 15.6.10 is that the cmode term goes from 1: instead of numdof: because we are using the results of the "modred" operation to define the reduced system matrix, which has only one row in cdisptip instead of numdof rows in cdisp. Once again, "lsim" is used to calculate the step response of the system.

```
%       plot modred using 'elim'

        semilogx(f,magdisptipdb(1,:),'k-',f,magdisptip_elimdb(1,:),'k.-')

        if  modred_sort == 1
                title(['reduced elimination:  tip disp for mid-length step force, …
                        first ',num2str(num_modes_used),' sorted modes included'])
        else
                title(['reduced elimination:  tip disp for mid-length step force, …
                        first ',num2str(num_modes_used),' unsorted modes included'])
        end

        legend('all modes','reduced elim',3)

        dcgain_error_percent_sort = 100*(magdisptip_elimdb(1) …
                        - magdisptip(1))/magdisptip(1)

        xlabel('Frequency, hz')
        ylabel('Magnitude, db mm')
        grid off

        pause

        hold on

%       now plot the overlay of the tip displacement magnitude with each mode contribution

        max_modes_plot = num_modes_used;

        for  pcnt = 1:max_modes_plot

                index = 2*pcnt;

                amode = adisptip_elim(index-1:index,index-1:index);

                bmode = bdisptip_elim(index-1:index);

                cmode = cdisptip_elim(1,index-1:index);
```

```
                    dmode = [0];

                    sysdisptip_mode = ss(amode,bmode,cmode,dmode);

                    [magdisptip_mode,phsdisptip_mode]=bode(sysdisptip_mode,frad)  ;

                    magdisptip_modedb = 20*log10(magdisptip_mode);

                    semilogx(f,magdisptip_modedb(1,:),'k-')

          end

          dc_gain_freq_sort = freqlo*ones(size(freqnew_nosort));

          pause

          hold off

%         modred using 'mdc'

          semilogx(f,magdisptipdb(1,:),'k-',f,magdisptip_mdcdb(1,:),'k.-')

          if  modred_sort == 1
                    title(['reduced matched dc gain:  tip disp for mid-length step force, …
                        first ',num2str(num_modes_used),' sorted modes included'])
          else
                    title(['reduced matched dc gain:  tip disp for mid-length step force, …
                        first ',num2str(num_modes_used),' unsorted modes included'])
          end

          legend('all modes','reduced mdc',3)

          dcgain_error_percent_nosort = 100*(magdisptip_mdcdb(1) …
                    - magdisptip(1))/magdisptip(1)

          xlabel('Frequency, hz')
          ylabel('Magnitude, db mm')
          grid off

          pause

          hold on

          max_modes_plot = num_modes_used;

          for  pcnt = 1:max_modes_plot

                    index = 2*pcnt;

                    amode = adisptip_mdc(index-1:index,index-1:index);

                    bmode = bdisptip_mdc(index-1:index);

                    cmode = cdisptip_mdc(1,index-1:index);
```

```
                    dmode = [0];

                    sysdisptip_mode = ss(amode,bmode,cmode,dmode);

                    [magdisptip_mode,phsdisptip_mode]=bode(sysdisptip_mode,frad)  ;

                    magdisptip_modedb = 20*log10(magdisptip_mode);

                    semilogx(f,magdisptip_modedb(1,:),'k-')

          end

          dc_gain_freq_nosort = freqlo*ones(size(freqnew_nosort));

          pause

          hold off

%         now use lsim to calculate step response to a unit force

          [disptip,ts] = lsim(sysdisptip,u,t);

          [disptip_elim,ts_elim] = lsim(sysdisptip_elim,u,t);

          [disptip_mdc,ts_mdc] = lsim(sysdisptip_mdc,u,t);

          plot(ts,disptip,'k-',ts_mdc,disptip_mdc,'k.-',ts_elim,disptip_elim,'k+-')

          if  modred_sort == 1
                    title(['modred cantilever tip disp for mid-length step force, …
                     first ',num2str(num_modes_used),' sorted modes included'])
          else
                    title(['modred cantilever tip disp for mid-length step force …
                        , first ',num2str(num_modes_used),' unsorted modes included'])
          end

          legend('all modes','reduced - mdc','reduced - elim')
          xlabel('time, sec')
          ylabel('displacement, mm')
          grid off
          pause

          end
```

15.6.17 Plotting Unsorted Modred Reduced Results – Eliminating High Frequency Modes

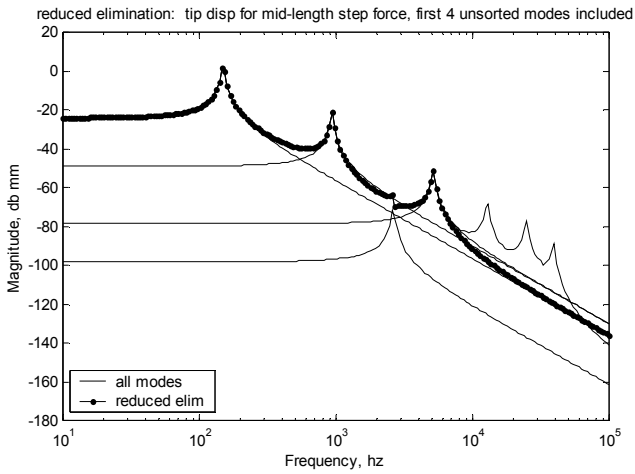

Figure 15.14: Cantilever tip displacement for mid-length force, first four modes included – unsorted modal truncation, modred "del" option.

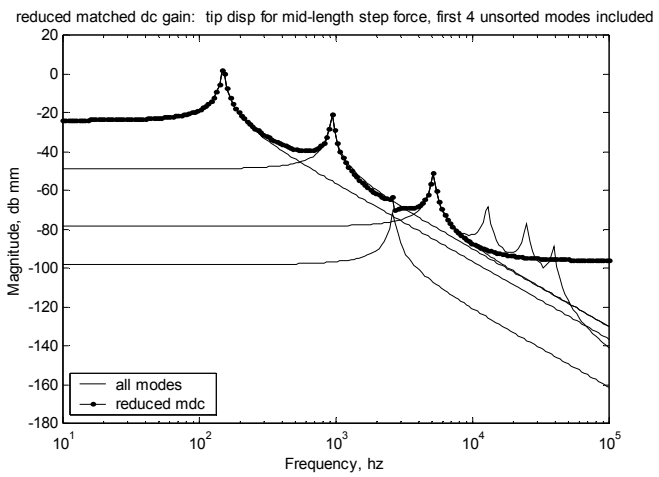

Figure 15.15: Cantilever tip displacement for mid-length force, first four modes included – unsorted modal truncation, modred "mdc" option.

Figure 15.14 shows overall frequency response with four overlaid individual mode contributions for the unsorted "del" "modred" option, with the six

highest frequency modes eliminated. Note that at high frequencies the reduced curve attenuates with frequency similar to the "all modes" curve.

Figure 15.15 shows overall frequency response with four overlaid individual mode contributions for the unsorted "mdc" "modred" option, with the six highest frequency modes reduced. Note the rise in the high frequency portion of the magnitude curve as a result of the matrix reduction operations discussed at the end of Section 15.6.12. Depending on the purpose of the model, the high frequency discrepancy may or may not be important.

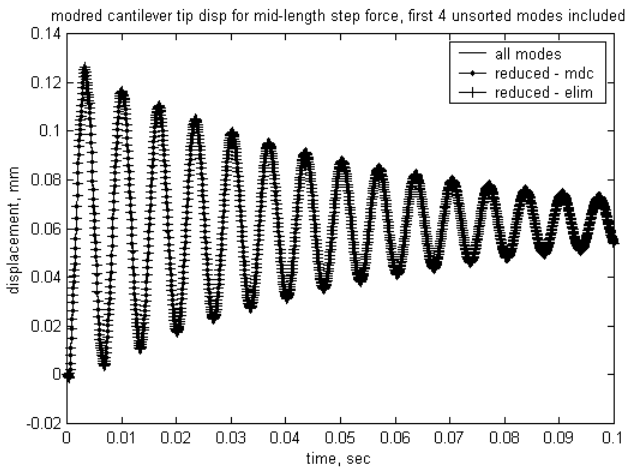

Figure 15.16: Comparison of step responses for all modes included and four modes included, "mdc" and "elim" "modred" options.

Figure 15.16 shows the overlay of step response for all mode model and "del" and "mdc" "modred" options. Note that there is no visible difference in the transient responses.

15.6.18 Plotting Sorted Modred Reduced Results – Eliminating Lower dc Gain Modes

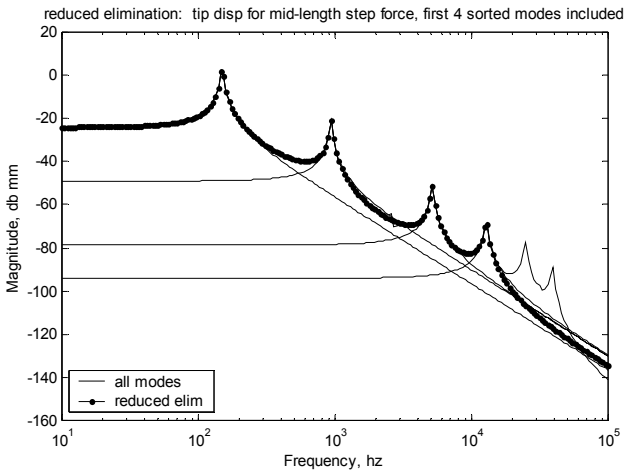

Figure 15.17: Cantilever tip displacement for mid-length force, first four sorted modes, modal truncation, "modred" "del" option.

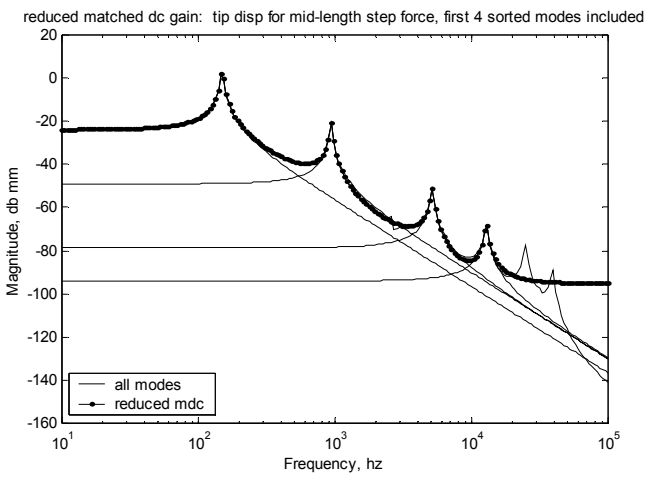

Figure 15.18: Cantilever tip displacement for mid-length force, first four sorted modes, "modred" "mdc" option.

Figure 15.17 shows overall frequency response with four overlaid individual mode contributions for the sorted "del" "modred" option, with the six lowest

dc gain modes eliminated. Figure 15.18 shows overall frequency response with four overlaid individual mode contributions for the unsorted "mdc" "modred" option, with the six lowest dc gain modes reduced. Again, note the lack of high frequency attenuation with frequency for the "modred" reduction.

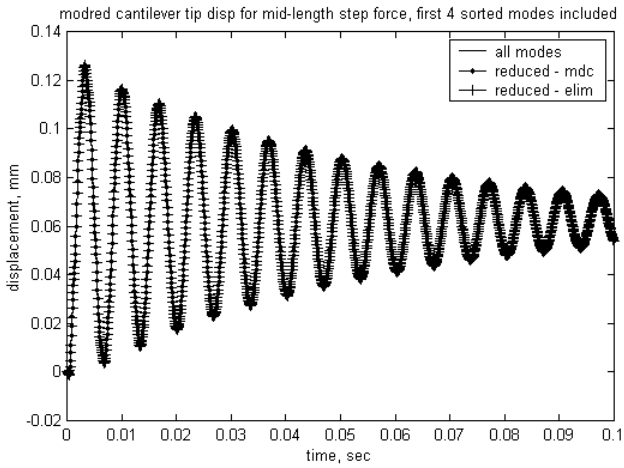

Figure 15.19: Comparison of step responses for all modes included and four sorted modes included, "mdc" and "elim" "modred" options.

Figure 15.19 depicts the overlay of step response for the all mode model and "del" and "mdc" "modred" options. Note that there is no visible difference in the transient responses.

15.6.19 Modred Summary

For this problem, where the dc gain of the response is dominated by the first several modes, there is not much difference between the sorted and unsorted responses. The "mdc" method minimizes low frequency errors by accounting for the dc gain of the unused modes but has high frequency behavior which deviates from the expected, and may not be desirable. The "del" method does not account for the dc gains of the unused modes, which can result in error in the low frequency portion of the frequency response. However, the "del" method has the advantage that it does not exhibit the unusual high frequency direct transmission matrix related behavior of the "mdc" method. If sorting of dc gain values is performed prior to the "del" operation, the system dc gain error may be acceptable while maintaining better high frequency performance.

15.7 ANSYS Code cantbeam_ss.inp Listing

The ANSYS code **cantbeam_ss.inp** solves for the eigenvalues and eigenvectors for a tip-loaded cantilever beam, with a sample output shown in Section 15.4. The user can define the number of elements to use for the cantilever and also choose whether to use the "Reduced" or "Block Lanczos" eigenvalue extraction method. The program then writes a frequency list out to a ".frq" file, outputs eigenvector listings to a ".eig" file and plots deformed/undeformed mode shapes to ".grp."

```
!  cantbeam_ss.inp, 0.075 thick x 2 wide x 20mm long steel cant
!  title automatically built based on number of elements and eigenvalue extraction method

/prep7

filename = 'cantbeam_ss'

! define number of elements to use

num_elem = 64

! define eigenvalue extraction method, 1 = reduced, 2 = block lanczos

eigext = 1

*if,eigext,eq,1, then
        nummodes = num_elem          !  only 1 displacement dof available for each element
*else
        nummodes = 2*num_elem        !  both disp and rotation dof's available for
                                     !      each  element
*endif

!       create the file name for storing data

!       first section of filename

aname = filename

!       second section of filename, number of elements

bname = num_elem

!       third section of filename, depends on eigenvalue extraction method

*if,eigext,ne,2, then
        cname = 'red'          !  reduced
*else
        cname = 'bl'           !  block Lanczos
*endif

! input the title, use %xxx% to substitute parameter name or parametric expression
```

```
aname_ti = 'cantbeam_ss - 0.075 thick x 2 wide x 20mm long steel cant'

/title,%aname_ti%, %bname%, %cname%

et,1,4                    ! element type for beam

! steel

ex,1,190e6                ! mN/mm^2
dens,1,7.83e-6            ! kg/mm^3
nuxy,1,.293

! real value to define beam characteristics

r,1,0.15,0.05,0.00007031,0.075,0.2          ! area, Izz, Iyy, TKz, TKy

! define plotting characteristics

/view,1,1,-1,1    ! iso view
/angle,1,-60      ! iso view
/pnum,mat,1       ! color by material
/num,1            ! numbers off
/type,1,0         ! hidden plot
/pbc,all,1        ! show all boundary conditions

csys,0                                  ! define global coordinate system

! nodes

n,1,0,0,0                                   ! left-hand node
n,num_elem+1,20,0,0                     ! right-hand node

fill,1,num_elem+1                       ! interior nodes

nall
nplo

! elements

type,1
mat,1
real,1
e,1,2
egen,num_elem,1,-1

! constrain left-hand end

nall
d,1,all,0                   ! constrain node 1, all dof's

! constrain all but uz and roty for all other nodes to allow only those dof's

nall
nsel,s,node,,2,num_elem+1
```

```
d,all,ux
d,all,uy
d,all,rotx
d,all,rotz

nall
eall
nplo
eplo

! ******************  eigenvalue run  ******************

fini                    ! fini just in case not in begin

/solu                   ! enters the solution processor, needs to be here to do editing below

allsel                  ! default selects all items of specified entity type, typically nodes, elements

nsel,s,node,,2,num_elem+1
m,all,uz

*if,eigext,eq,1,then                        ! use reduced method

        antype,modal,new
        modopt,reduc,nummodes       ! method - reduced Householdert
        expass,off                  ! key = off, no expansion pass, key = on,
                                    !   do expansion
        mxpand,nummodes,,,no        ! nummodes to expand,freq beginning,freq
                                    !  ending,elcalc = yes - calculate stresses
        total,num_elem,1            ! total masters, 1 is exclude rotations

*elseif,eigext,eq,2                         ! use block lanczos

        antype,modal,new
        modopt,lanb,nummodes        ! no total required for block lanczos
                                    !  because calculates all eigenvalues
        expass,off
        mxpand,nummodes,,,no

*endif

allsel

solve                   ! starts the solution of one load step of a solution sequence, modal here

fini

! plot first mode

/post1

/format,,,,,10000

set,1,1
```

```
pldi,1

! ********** output frequencies  **********

save,%aname%%bname%%cname%,sav

/output,%aname%%bname%%cname%,frq          ! write out frequency list to ascii file .frq

set,list

/output,term                                ! returns output to terminal

! **********  output eigenvectors  **********

! define nodes for output:  forces applied or output displacements

nall

/output,%aname%%bname%%cname%,eig        ! write out frequency list to ascii file .eig

*do,i,1,nummodes
        set,,i
        /page,,,1000
        prdisp
*enddo

/output,term

! **********  plot modes  **********

! pldi plots

/show,%aname%%bname%%cname%,grp,0          ! save mode shape plots to file .grp

allsel

/view,1,,-1,,                              ! side view for plotting
/angle,1,0
/auto

*do,i,1,nummodes
        set,1,i
   pldi,1
*enddo

/show,term
```