N93-26081

MODEL REDUCTION FOR SPACE STATION FREEDOM

Final Report

NASA/ASEE Summer Faculty Fellowship Program - 1992

Johnson Space Center

Prepared By:	Trevor Williams, Ph.D.
Academic Rank:	Associate Professor
University & Department:	University of Cincinnati Department of Aerospace Engineering Cincinnati, Ohio 45221

NASA/JSC	
Directorate:	Engineering
Division:	Guidance, Navigation & Aeronautics
Branch:	Control & Guidance Systems
JSC Colleague:	John Sunkel, Ph.D.
Date Submitted:	August 19, 1992
Contract Number:	NGT-44-005-803

ABSTRACT

Model reduction is an important practical problem in the control of flexible spacecraft, and a considerable amount of work has been carried out on this topic. Two of the best-known methods developed are *modal truncation* and *internal balancing*. Modal truncation is simple to implement but can give poor results when the structure possesses clustered natural frequencies, as often occurs in practice. Balancing avoids this problem but has the disadvantages of high computational cost, possible numerical sensitivity problems, and no physical interpretation for the resulting balanced "modes".

The purpose of this work is to examine the performance of the *subsystem balancing* technique developed by the investigator when tested on a realistic flexible space structure, in this case a model of the Permanently Manned Configuration (PMC) of Space Station Freedom. This method retains the desirable properties of standard balancing while overcoming the three difficulties listed above. It achieves this by first decomposing the structural model into subsystems of highly correlated modes. Each subsystem is approximately uncorrelated from all others, so balancing them separately and then combining yields comparable results to balancing the entire structure directly. The operation count reduction obtained by the new technique is considerable: a factor of roughly r^2 if the system decomposes into r equal subsystems. Numerical accuracy is also improved significantly, as the matrices being operated on are of reduced dimension, and the modes of the reduced-order model now have a clear physical interpretation; they are, to first order, linear combinations of repeated-frequency modes.

INTRODUCTION

Model reduction is a very important practical problem related to the control of flexible space structures (FSS), and a considerable amount of work has been carried out on this topic. Well-known methods include *modal truncation* [1], based either on the natural frequencies of the structure or its modal costs, and *balancing* [2] of the entire structure and then truncation to retain a dominant model for it. An advantage of the balancing approach is that it typically yields a more accurate reduced-order model than does simple modal truncation. This is particularly true when the structure possesses clustered natural frequencies, as is often the case for realistic flexible space structures. However, the disadvantages of balancing are its high computational cost, possible numerical sensitivity problems resulting from the large matrices being operated on, and the difficulty involved in providing a physical interpretation for the resulting balanced "modes".

The purpose of this paper is to investigate the practical performance of the alternative *subsystem balancing* technique when tested on a realistic flexible space structure. This method, introduced in [3][4] and further developed in [5], retains the desirable properties of standard balancing while overcoming the three difficulties listed above. This is achieved by first decomposing the structural model into subsystems of highly correlated modes, based on the *modal correlation coefficients* derived in [4] from the controllability and observability Grammian matrices [6] of the structure. Each subsystem is approximately uncorrelated from all others, so balancing each separately and concatenating the dominant reduced-order models obtained yields roughly the same result as balancing the entire structure directly. The computational cost reduction produced by this block-by-block technique is considerable: an operation count reduction by a factor of roughly $1/r^2$ if the system decomposes into r equal subsystems. The numerical accuracy of the resulting reduced-order model is also improved considerably, as the matrices being operated on are of reduced dimension; this avoids the numerical conditioning problems noted in [8][9] for standard balancing. Furthermore, the modes of the reduced model do now permit a clear physical interpretation. This is a consequence of the fact that each correlated subsystem must necessarily only include modes with close natural frequencies. The balanced modes of each subsystem are therefore, to first order, linear combinations of repeated-frequency modes, and so can themselves be taken as an equally valid set of physical modes. Balancing the entire structure, on the other hand, combines modes of widely differing frequencies, making interpretation difficult.

The results obtained using the software described in this report are for the Permanently Manned Configuration (PMC) of Space Station Freedom. Two different "stick models" [11] for this vehicle were studied, for two choices of solar array and radiator orientations. In both cases, the initial 202-mode flexible body models could be reduced to models with between 20 and 30 modes with very little loss of accuracy.

THEORETICAL BACKGROUND

Consider an n-mode model for the structural dynamics of a modally damped, non-gyroscopic, non-circulatory FSS with m actuators and p sensors, not necessarily collocated. This model can be written in modal form [1] as

$$\ddot{\boldsymbol{\eta}} + diag(2\zeta_i\omega_i)\dot{\boldsymbol{\eta}} + diag(\omega_i^2)\boldsymbol{\eta} = \hat{B}\mathbf{u},$$
$$\mathbf{y} = \hat{C}_r\dot{\boldsymbol{\eta}} + \hat{C}_d\boldsymbol{\eta}, \tag{1}$$

where $\boldsymbol{\eta}$ is the vector of modal coordinates, $\mathbf{u}$ that of applied actuator inputs and $\mathbf{y}$ that of sensor outputs, and ω_i and ζ_i are the natural frequency and damping ratio of the i^{th} mode, respectively. For the typical FSS [7], the $\{\zeta_i\}$ are quite low (e.g. 0.5 %), and the $\{\omega_i\}$ occur in clusters of repeated, or nearly repeated, frequencies as a result of structural symmetry.

Defining the state vector $\mathbf{x} = (\dot{\eta}_1, \omega_1\eta_1, \cdots, \dot{\eta}_n, \omega_n\eta_n)^T$ for this structure yields the state space representation $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$, $\mathbf{y} = C\mathbf{x}$, where $A = blkdiag(A)_i$, $B = (B_1^T, \cdots, B_n^T)^T$ and $C = (C_1, \cdots, C_n)$, with

$$A_i = \begin{pmatrix} -2\zeta_i\omega_i & -\omega_i \\ \omega_i & 0 \end{pmatrix}, B_i = \begin{pmatrix} \mathbf{b}_i \\ 0 \end{pmatrix} \text{and } C_i = \begin{pmatrix} \mathbf{c}_{ri} & \mathbf{c}_{di} / \omega_i \end{pmatrix}; \tag{2}$$

$\mathbf{b}_i$ is the i^{th} row of $\hat{B}$, and $\mathbf{c}_{ri}$ and $\mathbf{c}_{di}$ are the i^{th} columns of $\hat{C}_r$ and $\hat{C}_d$, respectively.

The problem studied here is that of obtaining a reduced-order model

$$\dot{\mathbf{x}}_r = A_r\mathbf{x}_r + B_r\mathbf{u},$$
$$\mathbf{y}_r = C_r\mathbf{x}_r \tag{3}$$

for this structure for which the normalized output error

$$\delta^2 = \int \|\mathbf{y}(t) - \mathbf{y}_r(t)\|_2^2 dt \Big/ \int \|\mathbf{y}(t)\|_2^2 dt \tag{4}$$

is acceptably small. Of course, the size of δ will depend on the order, n_r, chosen for the reduced model. A good model reduction procedure should ideally provide information allowing an intelligent choice for n_r to be made so as to achieve a specified δ value.

Two techniques for model reduction that have been extensively studied are those of modal truncation and internal balancing. The new method implemented in this report, subsystem balancing, can be regarded as an intermediate case between the two established techniques. Model reduction by subsystem balancing proceeds by first dividing the given structure into subsystems of highly correlated modes. Each subsystem is then balanced independently, and a reduced-order model for it generated by deleting all balanced states corresponding to Hankel singular values [2] below some specified threshold. Note that the singular value weighting described in [10] could be applied, if desired, without changing the argument in any way. Similarly, frequency weighting of the Hankel singular values can easily be incorporated to deal with input signals which have a known frequency spectrum. This is actually done in the present application, where the inputs are steps (representing thruster firings) rather than the impulses classically considered in

model reduction problems. The resulting reduced-order subsystem models so obtained are then combined to yield a dominant, approximately balanced, reduced-order model for the full system.

USER INTERFACE

This section describes the user interface to the model reduction package which was developed as part of this contract. This software consists of a library of Matlab m-functions, with mrmain calling all the other functions internally. The package is installed on the Sun SparcStation 2 *deimos* in the Integrated Analysis Laboratory in Building 16, and has also been produced in a Macintosh version. The documentation that follows details the user interface for mrmain; listings of this function, together with the second-level functions it calls, are given as an appendix. All functions have extensive in-line documentation, facilitating future use and/or modification.

Input arguments

om: The natural frequencies (rad/s) of the structure, input as either a row or column vector. Any rigid-body modes must precede the flexible modes and be represented by hard zero frequencies.

phia: The influence matrix, in mass-normalized coordinates, corresponding to the specified actuator locations. If the structure has n modes and m actuators, *phia* will be an (n x m) matrix.

phis: Similar to *phia*, but for sensor stations or positions of outputs of interest (e.g. solar array tips).

User responses

Output the time taken for each step?: The time required for each matrix decomposition, etc., is output to the screen if requested. This allows the progress of the model reduction procedure to be monitored, as well as giving an indication of which steps are the most computationally intensive.

Vectorize? (Faster, but requires more storage): In Matlab, for loops are typically an order of magnitude slower to excute than the equivalent "vectorized" operation. For instance, `s=0; for i=1:n, s=s+x(i); end;` runs considerably slower than does `s=x*ones(n,1)`. If vectorization is requested, computation of the system Grammian matrices and correlation coefficients is put into the form of vector-matrix operations rather than loops; this is indeed considerably faster, but requires some additional temporary storage arrays.

Structural damping ratio, % (default is 0.5%): The specified damping ratio is applied uniformly to all flexible modes of the full structural model.

Print frequencies in Hz?: The mean frequency of each subsystem can be output in either rad/s or Hz, as desired.

Desired controllability threshold?: This threshold value is used to determine which modes are correlated in a controllability sense. The system is then broken down into

disjoint sets of modes (subsystems), where modes with a controllability correlation coefficient greater than the specified threshold are deemed to be correlated. Taking a threshold value of 0 implies that all modes are considered correlated, i.e. the method reduces to standard balancing. Conversely, a threshold value of 1 implies that no modes are taken together: this is modal truncation. Intermediate values allow the dimensions of the resulting subsystems to be specified to a large extent; reducing the threshold reduces the number of subsystems, so increasing their dimension.

Desired overall threshold?: This threshold is used in a similar fashion to the controllability threshold, but both controllability and observability are now taken into account. This yields the final subsystem distribution output by the program (in *modemap*) and used to obtain the reduced-order model.

Compare step responses?: If requested, the step responses of the full and reduced-order models are computed, plotted, and the relative differences (i.e. reduced-order model error) output for each input-output channel.

Desired truncation measure?: Two types of measure can be used to define the number of modes retained in the reduced-order model. If a positive integer is input, this is taken to be precisely the desired reduced-order model. On the other hand, if a real number in the interval [0, 1) is input, this is taken to be the desired relative error in the reduced-order model step response, and the model order required to achieve this is computed. (Note that this later option is only an approximation, and should only be used as such.)

Output arguments

am, bm, cm: The reduced-order state-space model obtained.

modemap: This matrix specifies which physical modes are grouped into which subsystems in the decomposition based on overall correlation coefficients. The i^{th} column of *modemap* lists the modes making up the i^{th} of these subsystems.

SUMMARY OF RESULTS

Results will now be provided which illustrate the behavior of the subsystem balancing technique when applied to a structural model [11] of the Permanently Manned Configuration (PMC) of Space Station Freedom. This structure possesses light damping (estimated to be 0.5% of critical), and a large number of closely-spaced vibration modes (202 flexible modes below 10 Hz). Two configurations of the PMC were investigated: in the first, the solar arrays are in the station yz-plane ($\alpha = \beta = 0$) and the radiators in the xy-plane ($\gamma = 0$); in the second, the arrays are in the xy-plane and the radiators in the xz-plane ($\alpha = \gamma = 90^{\circ}, \beta = 0$). The inputs to these models are the 12 Reaction Control System (RCS) thrusters, i.e. the port/starboard and upper/lower x, y and z jets. The measured outputs are the 3 angular rates sensed by the rate gyros on the station avionics pallet. (The movements at other positions of interest, for instance the solar array tips, could also be considered if desired; the method remains exactly the same.)

A first point to examine is the efficiency of subsystem balancing as compared with that of standard balancing. Matlab function obalreal in the Robust Control Toolbox is a reliable

implementation of Moore's balancing algorithm; applying this to the PMC models considered requires approximately 3 hours on a SparcStation 2. By contrast, the subsystem balancing implementation provided by mrmain requires approximately 3 *minutes*. Furthermore, the bulk of the operations in subsystem balancing are order(n^2), due in part to the use of closed-form Grammians [6], whereas standard balancing is order(n^3). The efficiency advantages of the new approach will therefore become only more pronounced as larger systems are examined.

The role of the threshold coefficient in determining subsystem dimensions can be seen from the following table. The first column gives various choices for the controllability correlation threshold parameter, and columns 2 and 3 show the resulting maximum subsystem dimensions for the two PMC configurations studied (for input axis port upper x). It can be seen that these dimensions do indeed decrease as the threshold increases, as expected. Also, both systems exhibit broadly similar behavior. It can be noted that the evolution of subsystem dimensions is fairly discontinuous: for instance, large changes occur for thresholds between 0.10 and 0.15, whereas there are hardly any differences between 0.30 and 0.45. A consequence of this is that it is not always possible to find a threshold value which will yield a particular maximum subsystem order. However, it is possible to obtain a good working value which gives a totally acceptable subsystem partition. For the system studied here, a maximum subsystem dimension of about 30 leads to about 36-38 individual subsystems (some of which consist of single modes), a good balance; threshold values giving this distribution were chosen as nominal. Using these thresholds, the original 202-mode flexible models were found to be reducible to models with only 20 to 30 modes without introducing significant errors into the resulting step responses.

TABLE 1. - MAXIMUM SUBSYSTEM DIMENSIONS VERSUS THRESHOLD

Threshold	Max dim, $\alpha = 0°$	Max dim, $\alpha = 90°$	α=90°, ζ=1%
0.00	202	202	202
0.05	165	165	199
0.10	146	131	165
0.15	76	87	131
0.20	55	51	118
0.25	55	31	118
0.30	30	20	87
0.35	30	20	76
0.40	17	20	31
0.45	17	18	31

The fourth column of the table illustrates the effect of damping on model reduction. Damping of flexible structures is a very difficult quantity to model, so there is considerable uncertainty in the damping levels to be chosen for Space Station Freedom. If a value of 1% of critical is used instead of the previous "nominal" lavel of 0.5%, the consequent broadening of the peaks of each mode increases the coupling between modes, so increasing the subsystem dimensions. This does not pose any problem, however; increasing the threshold value to 0.4 will again allow the desired dimensions to be obtained.

REFERENCES

1. R.R. Craig, *Structural Dynamics*, New York: Wiley, 1981.

2. B.C. Moore, 'Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction', *IEEE Transactions on Automatic Control*, Vol. 26, Feb. 1981, pp. 17-32.

3. T.W.C. Williams and W.K Gawronski, 'Model Reduction for Flexible Spacecraft with Clustered Natural Frequencies', invited paper, 3rd NASA/NSF/DoD Workshop on Aerospace Computational Control, Oxnard, CA, Aug. 1989.

4. W.K Gawronski and T.W.C. Williams, 'Model Reduction for Flexible Space Structures', *Journal of Guidance, Control, and Dynamics*, Vol. 14, Jan.-Feb. 1991, pp. 68-76.

5. T.W.C. Williams and M. Mostarshedi, 'Model Reduction Results for Flexible Space Structures', presented at 5th NASA/DoD Control-Structures Interaction Technology Conference, Lake Tahoe, Nevada, Mar. 1992.

6. T.W.C. Williams, 'Closed-Form Grammians and Model Reduction for Flexible Space Structures', *IEEE Transactions on Automatic Control*, Vol. 35, Mar. 1990, pp. 379-382.

7. M.J. Balas, 'Trends in Large Space Structure Control Theory: Fondest Hopes, Wildest Dreams', *IEEE Transactions on Automatic Control*, Vol. 27, 1982, pp. 522-535.

8. M.S. Tombs and I. Postlethwaite, 'Truncated Balanced Realization of a Stable Non-Minimal State-Space System', *International Journal of Control*, Vol. 46, 1987, pp. 1319-1330.

9. M.G. Safanov and R.Y. Chiang, 'A Schur Method for Balanced Model Reduction', *Proc. American Control Conference*, 1988, pp. 1036-1040.

10. R.E. Skelton and P. Kabamba, 'Comments on "Balanced Gains and Their Significance for L^2 Model Reduction"', *IEEE Transactions on Automatic Control*, Vol. 31, Aug. 1986, pp. 796-797.

11. K. Schultz, 'PMC Flex Body Data Transfer', Memo SMD-92-2519, Lockheed Engineering & Sciences Company, Houston, Texas, Apr. 7, 1992.

MATLAB PROGRAM LISTINGS

```
function
[am,bm,cm,modemap]=mrmain(om,phia,phis);
%
% An M-function to perform model reduction based
on
% subsystem balancing for a uniformly-damped
% flexible structure with rate outputs
%
% All other functions in this package are called
% by this main routine.
%
% Arguments:
%
% In:  Frequency vector om (rad/s), with any
%      rigid-body modes (om(i)=0) leading;
%      influence matrices phia (actuators)
%      and phis (sensors), one row per mode
%
% Out: Reduced-order state-space model {am,bm,cm}
%      of flexible-body dynamics;
%      the i-th column of modemap lists those modes
%      making up the i-th unreduced subsystem
%

% Trevor Williams, NASA JSC, August 19, 1992
%
n=max(size(om)); % Number of modes considered
%
% Strip off any rigid-body modes
%
nflex=sum(sign(om)); nrigid=n-nflex;
om=om(nrigid+1:n);
b=phia(nrigid+1:n,:);
c=phis(nrigid+1:n,:)';
n=nflex;
%
timeout=0;
stime=input('Output the time taken for each step? ',
's');
if stime == 'y' timeout=1; end; % Time o/p wanted
%
vect=0;
svect=input('Vectorize? (Faster, but requires more
storage) ', 's');
if svect == 'y' vect=1; end; % Vectorization wanted
%
% Enter specified damping ratio
%
ze=input('Structural damping ratio, % (default is
0.5%) ');
if isempty(ze) == 1 ze=0.5; end; ze=ze/100;
%
radhz=1;
shz=input('Print frequencies in Hz? ', 's');
if shz == 'y' radhz=1/(2*pi); end; % Output format
%
% First compute controllability correlation coeffs
%
t0=clock;
ro=cccalc(om,ze,b,vect);
if timeout ~= 0 t1=etime(clock,t0);
  s=[' Ro-c completed after ', num2str(t1), ' s'];
  disp(' '); disp(s);
  end;
%
% Next, find observability Grammian for entire sys
%
t0=clock;
wo=cfgram(om,ze,c',-1,vect);
if timeout ~= 0 t1=etime(clock,t0);
  s=[' Wo completed after ', num2str(t1), ' s'];
  disp(' '); disp(s);
  end;
%
% Input desired controllability correl'n threshold
%
disp(' ');
disp(' Threshold values should lie between 0 and 1;')
disp(' lower values give fewer (but larger)
subsystems.')
disp(' Enter a negative value when finished.')
%
xdum=1; % Dummy: gives inelegant indefinite loop
while xdum > 0
   romin=input('Desired controllability threshold? ');
   if romin >= 1 romin=1-eps,end;
   if romin >= 0
%
%  Determine subsystems of correlated modes
%
     t0=clock;
     [isort nsub]=subsys(ro,romin);
     if timeout ~= 0 t1=etime(clock,t0);
       s=[' Internal decomposition took ', num2str(t1), '
s'];
       disp(' '); disp(s);
       end;
     kmax=max(size(nsub)); % Num of subsystems
     s=[' Yields ', int2str(kmax), ' subsystems;
maximum size '];
     s=[s, int2str(max(nsub)), ', minimum ',
int2str(min(nsub))];
     disp(' '); disp(s);
%
   else xdum = -1;
   end;
%
  end;
%
% Set up index to reorder Wo (agrees with isort)
%
iwsort(2:2:2*n)=2*isort;
iwsort(1:2:2*n-1)=2*isort-ones(1,n);
%
```

```
% Operate on each subsystem in turn
%
i1=1;
nmax=max(nsub); % Greatest subsystem order
wctot=[];
t0=clock;
for k=1:kmax
   nsubk=nsub(k);
   i2=i1+nsubk-1; ivect=isort(i1:i2);
   iwvect=iwsort(2*i1-1:2*i2);
%
%  First find its controllability Grammian
%
   wck=cfgram(om(ivect),ze,b(ivect,:),1,vect);
%
%  ... then find its singular value decomposition
%
   [uk,sk,vk]=svd(wck);
   wck=uk*sqrt(sk);
%
%  Finally, apply to correct row & col blocks of Wo
%
   wo(iwvect,:)=wck'*wo(iwvect,:);
   wo(:,iwvect)=wo(:,iwvect)*wck;
%
   wctot=[wctot [wck; zeros(2*(nmax-
nsubk),2*nsubk)]];
   i1=i2+1;
   end;
if timeout ~= 0 t1=etime(clock,t0);
  s=[' Wbar completed after ', num2str(t1), ' s'];
  disp(' '); disp(s);
  end;
%
%  Find matrix of overall correlation coefficients
%
t0=clock;
ro=occalc(wo,vect);
if timeout ~= 0 t1=etime(clock,t0);
  s=[' Ro-o completed after ', num2str(t1), ' s'];
  disp(' '); disp(s);
  end;
%
% Now try various overall threshold values
%
xdum = 1;  % Dummy variable again
while xdum > 0
   romin2=input('Desired overall threshold? ');
   if romin2 >= 1 romin2=1-eps,end;
   if romin2 >= 0
%
%    Determine new subsystems of correlated modes
%
    t0=clock;
    [isort2 nsub2]=subsys(ro,romin2);
    if timeout ~= 0 t1=etime(clock,t0);
      s=[' Final decomposition took ', num2str(t1), '
s'];
      disp(' '); disp(s);
      end;
    kmax2=max(size(nsub2));  % Num of subsystems
      s=[' Yields ', int2str(kmax2), ' subsystems;
 maximum size '];
      s=[s, int2str(max(nsub2)), ', minimum ',
 int2str(min(nsub2))];
      disp(' '); disp(s);
%
   else xdum = -1;
   end;
%
  end;
%
% Now that the final subsystems are defined,
% need to define corresponding Wo ordering
%
iwsort2(2:2:2*n)=2*isort2;
iwsort2(1:2:2*n-1)=2*isort2-ones(1,n);
%
% Find balancing transformation for each subsystem;
% store it and the weighted Hankel singular values
%
i1=1;
nmax=max(nsub2);  % Greatest subsystem order
ombar=zeros(1,kmax2);
modemap=zeros(nmax,kmax2);
hsv2t=[];
ttot=[];
t0=clock;
%
for k=1:kmax2
   nsubk=nsub2(k);
   i2=i1+nsubk-1; ivect=isort2(i1:i2);
   iwvect=iwsort2(2*i1-1:2*i2);
   modemap(1:nsubk,k)=ivect'+nrigid*ones(nsubk,1);
%
%  Solve this sym eigen-/singular value problem
%
   [tk,hsv2k,vk]=svd(wo(iwvect,iwvect));
   hsv2k=diag(hsv2k);
   [hsv2k ihsv]=sort(-hsv2k); hsv2k=-hsv2k;
   tk=tk(:,ihsv);  % Sort in descending order
   ttot=[ttot [tk; zeros(2*(nmax-nsubk),2*nsubk)]];
%
%  Perform ad hoc step response frequency weighting
%
   ombar(k)=sum(om(ivect))/nsubk;
   hsv2k=hsv2k/(ombar(k)*ombar(k));
   hsv2t=[hsv2t; hsv2k];
%
  i1=i2+1;
  end;
hsv2sum=sum(hsv2t);
[hsv2s ihsv]=sort(-hsv2t); hsv2s=-hsv2s;
if timeout ~= 0 t1=etime(clock,t0);
  s=[' Hankel singular values took ', num2str(t1), ' s'];
  disp(' '); disp(s);
  end;
%
% Compute the balanced full-order system
%
clear wo;  % Need the space for am (2nx2n here!)
t0=clock;
```

```
[am bm cm]=ssmodal(om,ze,b,c);
if timeout ~= 0 t1=etime(clock,t0);
  s=[' Full modal model took ', num2str(t1), ' s'];
  disp(' '); disp(s);
  end;
%
% Now apply Wc similarity transformation, by blocks
%
t0=clock;
[am bm cm]=blkmult(am,bm,cm,wctot,iwsort,nsub);
if timeout ~= 0 t1=etime(clock,t0);
  s=[' Wc similarity took ', num2str(t1), ' s'];
  disp(' '); disp(s);
  end;
%
% Then apply T similarity transformation, by blocks
%
t0=clock;
[am bm cm]=blkmult(am,bm,cm,ttot,iwsort2,nsub2);
if timeout ~= 0 t1=etime(clock,t0);
  s=[' T similarity took ', num2str(t1), ' s'];
  disp(' '); disp(s);
  end;
%
% Set up for step response calculations, if wanted
%
disp(' ');
sstep=input('Compare step responses? ', 's');
if sstep == 'y'
  t=4/(ze*min(om));   % Time for decay
  t=100*round(t/100); % Round to nearest hundred
  dt=t/100; t=[dt:dt:t]; % 100 points
%
  m=size(b); m=m(2);  % Number of inputs
  p=size(c); p=p(1);  % Number of outputs
%
% Compute and store step responses of full system
  yf=[];
  t0=clock;
  for iu=1:m
    yf=[yf modstep(om,ze,b,c,iu,t)];
    end;
%
  if timeout ~= 0 t1=etime(clock,t0);
    s=[' Full step response took ', num2str(t1), ' s'];
    disp(' '); disp(s);
    end;
  end;
%
% Try various different truncation measures
%
disp(' ');
disp(' Enter either the desired number of modes
(integer > 0)')
disp(' or the acceptable approx relative output error (<
1);')
disp(' enter a negative quantity when finished.')
%
xdum = 1; % Dummy variable again
while xdum > 0
  cutoff=input('Desired truncation measure? ');
  if isempty(cutoff) == 1 cutoff=1; end; % Safety
  t0=clock;
  if cutoff < 0 xdum = -1;
  else
    if cutoff >= 1 nrom=min(cutoff,n); % # modes
    else % Find num modes from desired rel err
      abserr2=cutoff*cutoff*hsv2sum;
      test=0; i=n;
      while test <= abserr2
        test=test+hsv2s(2*i)+hsv2s(2*i-1);
        i=i-1;
        end;
      nrom=i+1;
    end;
%
%   Find number of modes kept from each subsys
%
    hsv2min=hsv2s(2*nrom);
    nsubr=zeros(1,kmax2);
    ioff=0;
    for k=1:kmax2
      for i=1:nsub2(k)
        if abs(hsv2t(2*(ioff+i))) >= hsv2min
          nsubr(k)=nsubr(k)+1;
          end;
        end;
      ioff=ioff+nsub2(k);
      end;
%
%   Finally, find truncation index to give ROM
%
    i1=1;
    iavect=[];
%
    for k=1:kmax2
      i2=i1+2*nsubr(k)-1;
      iavect=[iavect iwsort2(i1:i2)];
      i1=i1+2*nsub2(k);
      end;
%
% Alter cm so as to give zero steady-state error
%
    delcm=0*cm;
    g=am(iavect,iavect)\bm(iavect,:);
    x=-cm(:,iavect)*g/(g'*g);
    delcm(:,iavect)=x*g';
%
    if timeout ~= 0 t1=etime(clock,t0);
      s=[' Model dimension found after ',
num2str(t1), ' s'];
      disp(' '); disp(s);
      end;
%
%   Output subsystem information
%
    disp(' ');
    disp(' Subsystem   Number   Mean freq');
    s=' dimension retained ';
    if shz == 'y' s=[s, ' (Hz)']; else s=[s, '(rad/s)']; end;
    disp(s);
%
```

```
      for k=1:kmax2
         spad=' ';
         if nsub2(k) >= 10 spad=' '; end;
         if nsub2(k) >= 100 spad=''; end;
         s=['    ', spad, int2str(nsub2(k))];
%
         spad=' ';
         if nsubr(k) >= 10 spad=' '; end;
         if nsubr(k) >= 100 spad=''; end;
         s=[s, '        ', spad, int2str(nsubr(k))];
%
         spad='  ';
         prombar=radhz*ombar(k);
         if prombar >= 10 spad=' '; end;
         if prombar >= 100 spad=' '; end;
         if prombar >= 1000 spad=''; end;
         s=[s, '    ', spad, num2str(prombar)];
         disp(s);
         end;
%
%    Compare step responses if requested
%
     if sstep == 'y'
       for iu=1:m
         t0=clock;
         amp=am(iavect,iavect);
         bmp=bm(iavect,:);
         cmp=cm(:,iavect)+delcm(:,iavect);
         yr=blkstep(amp,bmp,cmp,iu,t);
         if timeout ~= 0 t1=etime(clock,t0);
           s=[' ROM step response took ', num2str(t1),
' s'];
           disp(' '); disp(s);
           end;
%
         for io=1:p
           iyf=(iu-1)*p+io;  % Access correct
response
           plot(t,[yf(:,iyf) yr(:,io)]);
           s=['Step responses, nrom = ',
int2str(nrom)];
           s=[s, '; input ', int2str(iu)];
           s=[s,', output ', int2str(io)];
           title(s); grid;
           xlabel('Time (s)');
           ylabel('Outputs');
           pause;
%
           plot(t,(yf(:,iyf)-yr(:,io)));
           s=['Step error, nrom = ', int2str(nrom)];
           s=[s, '; input ', int2str(iu)];
           s=[s, ', output ', int2str(io)];
           title(s); grid;
           xlabel('Time (s)');
           ylabel('Error');
           pause;
           yerr=norm(yf(:,iyf)-
yr(:,io))/norm(yf(:,iyf));
           s=[' 2-norm relative output error of ',
num2str(yerr)];
           disp(' '); disp(s);
           end;
         end;
       end;
     end;
%
%  Finally, store the chosen reduced-order model
%
am=am(iavect,iavect);
bm=bm(iavect,:);
cm=cm(:,iavect)+delcm(:,iavect);
#
#########################################
#
function ro=cccalc(om,ze,b,vect);
%
% An M-function to construct the controllability
% correlation coefficients of a uniformly-
% damped flexible structure
%
% The flag vect ~= 0 for Matlab-style vectorized
% operations: faster, but requires extra temp store

% Trevor Williams, NASA JSC, August 19, 1992
%
n=max(size(om));  % Number of modes considered
%
if vect == 0
%
%  "Standard" loop over matrix locations
%
  ro=eye(n);        % Initialization (saves time)
  betad=zeros(n,1);  %    "     "
%
%  First compute contribution from B
%
  for i=1:n
    for j=i+1:n
      ro(i,j)=b(i,:)*b(j,:)';
      end;
    betad(i)=max(b(i,:)*b(i,:)', eps);
    end
%
  for i=1:n
    for j=i+1:n
      ro(i,j)=abs(ro(i,j))/sqrt(betad(i)*betad(j));
      end;
    end
%
%  Now add the frequency effects (Frob-norm version)
%
  for i=1:n
    for j=i+1:n
      g=om(j)/om(i);
      temp=8*ze*ze*g;
      num=(g-1)^2*(g^2+1);
      num=sqrt(temp*((g*temp)+num));
      den=(g+1)*((g-1)^2+(temp/2));


        ro(i,j)=(num/den)*ro(i,j);
```

```
      ro(j,i)=ro(i,j);
      end;
    end
else
%
% Calculations in Matlab-style vectorized form:
% first compute contribution from B
%
  ro=b*b';
  ro=abs(ro);
%
% Avoid singularities caused by zero entries in b,c
%
  betad=max(diag(ro), eps*ones(n,1));
  ro=ro-diag(diag(ro)); % Zero all diagonals
  ro=ro+diag(betad);    % Put back diag, or eps
%
  betads=sqrt(diag(betad));
  ro=betads\abs(ro)/betads;
%
% Now add the frequency effects (Frob-norm version)
%
  temp=8*ze*ze;
%
  som=size(om);
  if som(1) > 1   % om entered as a column vector
    g=(om.^(-1))*om';
  else            % om entered as a row vector
    g=(om.^(-1))'*om;
  end
%
  num=((g-ones(n)).^2).*((g.^2)+ones(n));
  num=sqrt((temp*g).*((temp*(g.^2))+num));
  den=(g+ones(n)).*(((g-ones(n)).^2)+(temp*g/2));
  ro=(num./den).*ro;
end
#
##########################################
#
function w=cfgram(om,ze,b,cobs,vect);
%
% An M-function to compute the closed-form
% Grammians of a uniformly-damped flexible
% structure with rate measurements
%
% The flag cobs == 1 for controllability,
%              -1 for observability.
%
% The flag vect ~= 0 for Matlab-style vectorized
% operations: faster, but requires extra temp store

% Trevor Williams, NASA JSC, August 19, 1992
%
n=max(size(om)); % Number of modes considered
w=zeros(2*n); % Initialization (saves time later)
%
if vect == 0
%
% "Standard" loop over matrix locations
%
% Compute each (2 x 2) Grammian block in turn,
% and store in correct upper & lower locations in W
%
for i=1:n
  omi=om(i);
  iw=2*i-1;
  for j=i:n
    omj=om(j);
    jw=2*j-1;
    t1=ze*(omi+omj);
    t2=(omj*omj)-(omi*omi);
    t3=2*omi*omj;
    t4=t3*t1;
    dij=2*t3*(t1*t1)+(t2*t2);
    wij=[t4 cobs*omj*t2; -cobs*omi*t2 t4];
    wij=((b(i,:)*b(j,:)')/dij)*wij;
    w(iw:iw+1,jw:jw+1)=wij;
    w(jw:jw+1,iw:iw+1)=wij';
    end;
  if b(i,:)*b(i,:)' < eps  % Avoid singularity
    w(iw:iw+1,iw:iw+1)=eps*eye(2);
    end;
  end;
else
%
% Calculations in Matlab-style vectorized form:
% first compute contribution from B
%
  i1=(1:2:2*n-1);
  i2=(2:2:2*n);
%
  som=size(om);
  if som(1) > 1   % om entered as a column vector
    t1=ze*(om*ones(1,n)+ones(n,1)*om');
    t2=(ones(n,1)*(om.^2)'-(om.^2)*ones(1,n));
    omj=ones(n,1)*om';
    t3=2*om*om';
  else            % om entered as a row vector
    t1=ze*(om'*ones(1,n)+ones(n,1)*om);
    t2=(ones(n,1)*(om.^2)-(om.^2)'*ones(1,n));
    omj=ones(n,1)*om;
    t3=2*om'*om;
  end
%
  d=2*t3.*(t1.^2)+(t2.^2);
  beta=b*b';
%
% Avoid singularities caused by zero entries in b,c
%
  betad=max(diag(beta), eps*ones(n,1));
  beta=beta-diag(diag(beta)); % Zero all diagonals
  beta=beta+diag(betad);    % Put back diag, or eps
%
  w(i1,i1)=beta.*t3.*t1./d;
  w(i1,i2)=cobs*beta.*omj.*t2./d;
  w(i2,i1)=-cobs*beta.*omj'.*t2./d;
  w(i2,i2)=beta.*t3.*t1./d;
end
#
##########################################
#
function [isort, nsub]=subsys(ro,romin);
```

```
%
% An M-function to determine those modes which are
% deemed to be correlated, based on a given matrix
% of modal correlation coefficients and a
% specified threshold value
%
% Outputs:
%
%  isort: sorting required to produce subsystems
%  nsub: dimensions of the subsystems

% Trevor Williams, NASA JSC, August 19, 1992
%
n=max(size(ro));  % Number of modes considered
%
% Perform various initializations
%
num=0;      % Total number of modes grouped
nsub=[];    % No subsystem dimensions yet
isort=[1:]; % Modal ordering: unchanged so far
kvect=(n+1)*ones(1,n); % Makes unsorted modes last
kflag=0; % 1 for subsystem 1, 2 for subsys 2, etc.
%
% Characterize each subsystem in turn
%
while num < n
%
% Initializations for this subsystem
%
  nsubk=0;            % No modes yet found
  kflag=kflag+1;      % Increment flag
  itest=isort(num+1); % Mode to be tested 1st
  jtest=isort(num+1:n); % Test for corr'n to i
%
  while isempty(itest) == 0
%
%    itest contains a set of modes to be tested
%
    inew=[]; % No modes for next pass yet
%
    for i=itest
      for j=jtest
        if ro(i,j) >= romin & kvect(j) > n
%
%           Mode j is a new mode, correlated
%           to i: store in this subsystem
%
          kvect(j)=kflag;
          nsubk=nsubk+1;
          if j ~= i inew=[inew j]; end;
          end;
        end;
      end;
%
%    Pick up new set of modes (if any) to test
%
    itest=inew;
    end
%
% This subsystem is finished: store its data
%
  num=num+nsubk;    % Total number of modes
  nsub=[nsub nsubk]; % Subsystem dimensions
  [kvsort isort]=sort(kvect); % Sorted modes
  end
#
#########################################
#
function ro=occalc(w,vect);
%
% An M-function to compute the overall
% correlation coefficients of a uniformly-
% damped flexible structure
%
% The flag vect ~= 0 for Matlab-style vectorized
% operations: faster, but requires extra temp store

% Trevor Williams, NASA JSC, August 19, 1992
%
n=max(size(w))/2; % Number of modes considered
%
if vect == 0
%
% "Standard" loop over matrix locations
%
 ro=eye(n); % Initialization (saves time later)
 for i=1:n
   iw=2*i-1;
   wiif=norm(w(iw:iw+1,iw:iw+1),'fro');
   for j=i+1:n
     jw=2*j-1;
     wjjf=norm(w(jw:jw+1,jw:jw+1),'fro');
     wijf=norm(w(iw:iw+1,jw:jw+1),'fro');
     ro(i,j)=wijf/sqrt(wiif*wjjf);
     ro(j,i)=ro(i,j);
     end;
   end
else
%
% Frobenius norm calculation in Matlab vector form
%
 n2=2*n;
 e=eye(n2)+diag(ones(n2-1,1),-1);
 e=e(:,1:2:n2-1);
%
% Calculate Frobenius norms of each (2x2) block
%
 ro=e'*(w.*w)*e;
 ro=sqrt(ro);
%
% Now normalize
%
 rodiags=diag(sqrt(diag(ro)));
%
 ro=rodiags\ro/rodiags;
end
#
#########################################
#
function
[amodal,bmodal,cmodal]=ssmodal(om,ze,b,c);
%
```

```
% An M-function to construct the modal state
% space model corresponding to a uniformly-
% damped flexible structure

% Trevor Williams, NASA JSC, August 19, 1992
%
n=max(size(om));      % Number of modes considered
%
amodal=zeros(2*n);
adiag=[-2*ze -1;1 0];  % "Template" for diags of a
%
m=size(b); m=m(2);
bmodal=zeros(2*n,m);
%
p=size(c); p=p(1);
cmodal=zeros(p,2*n);
%
for i=1:n
   i2=2*i-1;
   amodal(i2:i2+1,i2:i2+1)=om(i)*adiag;
   bmodal(i2,:)=b(i,:);
   cmodal(:,i2)=c(:,i);
   end
#
##########################################
#
function
[am,bm,cm]=blkmult(am,bm,cm,wctot,iwsort,nsub);
%
% An M-function to apply a similarity
% transformation stored as ordered blocks
% to a given state space model

% Trevor Williams, NASA JSC, August 19, 1992
%
kmax=max(size(nsub));  % Number of subsystems
%
iw1=1;
for k=1:kmax
   nsubk2=2*nsub(k);
   iw2=iw1+nsubk2-1;
   iwvect=iwsort(iw1:iw2);  % Required state order
%
%  Retrieve k-th block of similarity transformation
%
   wck=wctot(1:nsubk2,iw1:iw2);
%
%  Premult row blocks of am, bm by inverse of wck
%
   am(iwvect,:)=wck\am(iwvect,:);
   bm(iwvect,:)=wck\bm(iwvect,:);
%
%  Postmultiply column blocks of am, cm by wck
%
   am(:,iwvect)=am(:,iwvect)*wck;
   cm(:,iwvect)=cm(:,iwvect)*wck;
%
   iw1=iw2+1;
   end
#
##########################################
#
function y=modstep(om,ze,b,c,iu,t);
%
% An M-function to compute the step response
% of a state space model in "symmetric"
% modal form of a flexible structure

% Trevor Williams, NASA JSC, August 19, 1992
%
n=max(size(om));    % Number of modes
tmax=max(size(t));  % Length of time vector
p=size(c); p=p(1);  % Number of outputs
di=zeros(p,1);
%
y=zeros(tmax,p);
adiag=[-2*ze -1; 1 0];
%
for i=1:n
%
%  Set up required submatrices
%
   ai=om(i)*adiag;
   bi=[b(i,iu); 0];
   ci=[c(:,i), 0*c(:,i)];
%
%  Add step response of this mode to total
%
   y=y+step(ai,bi,ci,di,1,t);
   end
#
##########################################
#
function y=blkstep(a,b,c,iu,t);
%
% An M-function to compute the step response
% of a state space model of a flexible structure
% with A block diagonal (modulo ordering!)

% Trevor Williams, NASA JSC, August 19, 1992
%
tmax=max(size(t));    % Length of time vector
p=size(c); p=p(1);    % Number of outputs
di=zeros(p,1);
%
y=zeros(tmax,p);
%
%  Determine OVERALL block structure of a, directly
%
[iasort,nasub]=subsys(abs(a)+abs(a'),eps);
kmax=max(size(nasub));
%
i1=1;
for k=1:kmax
   i2=i1+nasub(k)-1;
   ia=iasort(i1:i2);
%
%  Add step response of this block to total
%
   y=y+step(a(ia,ia),b(ia,iu),c(:,ia),di,1,t);
   i1=i2+1;
   end
```