SANDIA REPORT
SAND2019-10728
Printed September 11, 2019

Sandia National Laboratories

Exactly and Easily Applying Experimental Boundary Conditions in Computational Structural Dynamics

Bunting, Gregory, Crane, Nathan K., Day, David M., Dohrmann, Clark R., Ferri, Brian A., Flicek, Robert C., Hardesty, Sean S., Lindsay, Payton, Miller, Scott T., Munday, Lynn B., Stevens, Brian L., and Walsh, Timothy F.

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

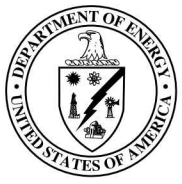
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods



ABSTRACT

Most experimental setups and environment specifications define acceleration loads on the component. However, Sierra Structural Dynamics cannot apply acceleration boundary conditions in modal transient analysis. Modal analysis of these systems and environments must be done through the application of a huge artificial force to a large fictitious point mass. Introducing a large mass into the analysis is a common source of numerical error. In this report we detail a mathematical procedure to directly apply acceleration boundary conditions in modal analyses without the requirement of adding a non-physical mass to the system. We prototype and demonstrate this procedure in Matlab and scope the work required to integrate this procedure into Sierra Structural Dynamics.

ACKNOWLEDGMENT

This work was supported by the Laboratory Directed Research and Developement (LDRD) express program.

This page intentionally left blank.

CONTENTS

LIST OF FIGURES

1. MOTIVATION

Most experimental setups and environment specifications define acceleration loads on components. However, for modal transient analysis, Sierra Structural Dynamics can apply forces, but not accelerations. Current practice is to add a huge artificial mass to the structure, and then apply a huge artificial force that causes that mass to move at approximately the desired acceleration. This is problematic for several reasons:

- If the mass is too small, the wrong problem is solved and an invalid result is obtained.
- If the mass is too large, linear systems become poorly conditioned, leading to a highly inaccurate solution and an invalid result.
- An appropriate size of the mass isn't known a priori, and must be determined via trial and error. The acceptable band is often narrow.

The results of a computational setup using such an artificial mass can be assessed based on the knowledge and experience of the analyst, and by repeating the run with a different mass or different linear solver parameters. This process both substantially increases the amount of time and effort required to produce reliable results, and is a significant source of potential error and uncertainty.

The purpose of this report is to detail a mathematical procedure to directly apply acceleration boundary conditions to modal analyses without adding non-physical mass or requiring analyst intervention. This procedure has been prototyped and demonstrated in Matlab, and scoping work has been done to determine how to integrate the procedure into the Sierra Structural Dynamics production analysis application.

2. INTRODUCTION

We begin our overview of the problem by partitioning the output displacements into the two pieces shown in Figure 1 : u_r and u_p, where the r subscript is used for the residual degrees of freedom and the p subscript indicates the degrees of freedom determined by the prescribed accelerations.

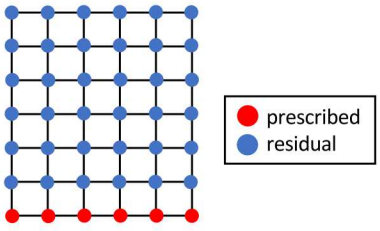


Figure 1. Partition of a discritezed structure's degrees of freedom into prescribed (red) and residual (blue).

The matrices are partitioned analogously, so that the system appears as follows:

$$K_{rr}u_r + M_{rr}\ddot{u}_r + K_{rp}u_p + M_{rp}\ddot{u}_p = f_r.$$

The force vector, f_r, may be restricted to the residual degrees of freedom. Since u_p is known, we move it to the right-hand side:

$$K_{rr}u_r + M_{rr}\ddot{u}_r = \underbrace{f_r - K_{rp}u_p - M_{rp}\ddot{u}_p}_{\equiv F} \tag{2.1}$$

For notational simplicity, we will drop the subscripts and write

$$Ku + M\ddot{u} = F, \tag{2.2}$$

keeping in mind that u includes the residual degrees of freedom, while the forces created by the prescribed motion of the boundary are incorporated into F. Until section 3.1, we omit the damping term $D\dot{u}$ from the discussion.

Direct solution of (2.2) via time stepping methods is straight-forward, but costly: due to the coupling between the different degrees of freedom, parallel communication is required at each time step. The approach used in Sierra Structural Dynamics, for both the modal transient analysis and modal FRF (frequency response function) solution cases, begins with a modal decomposition.

We solve the generalized eigenvalue problem

$$K\Phi = M\Phi\Lambda,\ \Phi^T M\Phi = I, \tag{2.3}$$

and write $u = \Phi x$. Note that because M is symmetric positive definite (SPD), Φ can always be chosen such that $\Phi^T M\Phi = I$. Equation (2.2) can then be rewritten

$$\underbrace{K\Phi}_{=M\Phi\Lambda} x + M\Phi\ddot{x} = F,$$

whence multiplication by Φ^T yields

$$\Lambda x + \ddot{x} = \Phi^T F. \tag{2.4}$$

Equation (2.4) is uncoupled, and can be solved easily. Perhaps surprisingly, even though the modes in Φ are computed on the residual degrees of freedom only, i.e., as if the prescribed degrees of freedom were fixed, the solution is exact if Φ includes a complete set of modes. But if it is necessary to compute all of the modes, then we have merely traded one difficult problem for another: the benefits of this approach are realized if (2.4) yields an accurate solution with only a small number of modes. However, with prescribed accelerations, this is not the case: numerical experiments in section 4.1 will show that nearly the full set of modes is required to obtain a reasonable solution, even when F is band-limited. The experimental setting to consider is a shaker table, where the prescribed degrees of freedom lie in a plane, and move as a rigid body. The difficulty stems from the incompatibility of the modes of Φ with rigid-body motions. Our approach to solving this problem is discussed in detail in section 3.

3. AUGMENTED MODAL BASIS

Building on the eigenvalue problem described in section 2, we augment the modes in Φ with a matrix of rigid-body modes Ψ. With three spatial dimensions, Ψ would have six columns, corresponding to three translational and three rotational rigid-body modes. The augmented basis is formed via the matrix

$$P = [\Phi, \Psi].$$

Expressing the displacement u via $u = Px$, equation (2.2) becomes

$$KPx + MP\ddot{x} = F.$$

Multiplying by P^T, we obtain

$$\underbrace{P^T K P}_{\equiv \hat{K}} x + \underbrace{P^T M P}_{\equiv \hat{M}} \ddot{x} = P^T F. \tag{3.1}$$

The matrices $\hat{K}$ and $\hat{M}$ take the form

$$\hat{K} = \begin{pmatrix} \Lambda & \Phi^T K \Psi \\ \Psi^T K \Phi & \Psi^T K \Psi \end{pmatrix}, \; \hat{M} = \begin{pmatrix} I & \Phi^T M \Psi \\ \Psi^T M \Phi & \Psi^T M \Psi \end{pmatrix}. \tag{3.2}$$

Thus, the system (3.1) is small, but still coupled: the matrices have an arrowhead structure with large diagonal blocks, but coupling between the rigid-body coordinates and eigen coordinates.

It is fairly straight-forward to transform the system (3.1) to diagonal form, as the system is small enough that we can afford to work with dense linear algebra methods. By solving the eigenvalue problem (note that $\hat{M}$ is SPD because M is SPD and P is full-rank)

$$\hat{K}V = \hat{M}V\Theta, \; V^T \hat{M} V = I, \tag{3.3}$$

and writing $x = Vy$ (so that $u = Px = PVy$), we obtain the system

$$\ddot{y} + \Theta y = (PV)^T F. \tag{3.4}$$

3.1. Augmented Modal Basis with Damping

Thus far, we have omitted discussion of the damping term $D\dot{u}$ from the exposition, beginning in (2.2). The standard damping formulation expresses D as a linear combination of K and M, say

$$D = \alpha K + \beta M,$$

so that

$$\Phi^T D \Phi = \alpha \Lambda + \beta I$$

is a diagonal matrix. However, construction of the augmented damping matrix $\hat{D}$ then proceeds analogously to (3.2):

$$\hat{D} = \alpha\hat{K} + \beta\hat{M} = \begin{pmatrix} \alpha\Lambda + \beta I & \Phi^T D\Psi \\ \Psi^T D\Phi & \Psi^T D\Psi \end{pmatrix}. \tag{3.5}$$

Although not diagonal, the matrix $\hat{D}$ is also transformed to diagonal form by V:

$$V^T\hat{D}V = \alpha V^T\hat{K}V + \beta V^T\hat{M}V = \alpha\Theta + \beta I.$$

Following this through the derivation, equation (3.4) would become

$$\ddot{y} + (\alpha\Theta + \beta I)\dot{y} + \Theta y = (PV)^T F.$$

The only unseemly aspect of the choice (3.5) is that damping is applied to the rigid-body modes, which may not be ideal from a physical point of view. However, if we choose $\hat{D}$ to be diagonal, whether by using a modal damping model such as

$$\hat{D} = \begin{pmatrix} 2\Gamma\Omega & 0 \\ 0 & 0 \end{pmatrix},$$

or by explicitly removing the arrowhead parts in order to eliminate the damping of rigid-body modes a la

$$\hat{D} = \begin{pmatrix} \alpha\Lambda + \beta I & 0 \\ 0 & 0 \end{pmatrix},$$

we encounter a different problem: simultaneous diagonalization of three real symmetric matrices is possible only under very special circumstances [1].

Therefore, we must choose between undamped rigid-body modes and a decoupled system. There is a third option, which is simply to solve (3.1): if we prefer undamped rigid-body modes, then in practice, there is little point in solving the eigenvalue problem (3.3), since it fails to decouple the equations.

4. 1D ANALYTIC SOLUTION

In this section, we will derive an analytic solution to a model problem to be used for testing purposes. Consider a string with transverse displacement $u(x,t)$, initially at rest ($u(x,0) = u_t(x,0) = 0$). The solution $u(x,t)$ obeys the wave equation:

$$u_{tt} = c^2 u_{xx}.$$

Suppose that the end points ($x = 0, x = L$) follow the prescribed motion $u(0,t) = u(L,t) = f(t)$. This mimics the situation of a rigid shaker table: the mode shapes of the string with fixed ends do a poor job of capturing a rigid motion of the entire string suggested by $f(t)$. We will make the ansatz that the displacement is equal to $f(t)$ plus some unknown function $p(x,t)$:

$$u(x,t) = f(t) + p(x,t),$$

with $p(0,t) = p(L,t) = 0$. Thus, we have

$$u_{tt}(x,t) = f''(t) + p_{tt}(x,t) = c^2 u_{xx}(x,t) = c^2 p_{xx}(x,t).$$

Taking the Laplace transform, we obtain

$$s^2 \hat{p}(x,s) - c^2 \hat{p}_{xx}(x,s) = -s^2 \hat{f}(s)$$

We can then write the standard series solution in space

$$\hat{p}(x,s) = \sum_{n=1}^{\infty} \hat{a}_n(s) \sin(k_n x)$$

for $\hat{p}$, defining $k_n = \frac{n\pi}{L}$ and $\omega_n = ck_n$. Using the orthogonality relations,

$$\int_0^L \sin(k_m x)\sin(k_n x)\,dx = \begin{cases} \frac{L}{2} & m = n \\ 0 & m \neq n \end{cases}$$

$$\int_0^L \sin(k_n x)\,dx = \begin{cases} \frac{2L}{n\pi} & n \text{ odd} \\ 0 & n \text{ even} \end{cases}$$

we obtain (for n odd)

$$\hat{a}_n(s) = \frac{-4s^2 \hat{f}(s)}{n\pi(s^2 + \omega_n^2)}.$$

For a given constant $\omega > 0$, we choose a sinusoidal end motion

$$f(t) = \sin(\omega t),$$

with Laplace transform

$$\hat{f}(s) = \frac{\omega}{\omega^2 + s^2}.$$

After an inverse Laplace transform, we obtain the solution

$$u(x,t) = \sin(\omega t) + \sum_{n=1,3,\ldots}^{\infty} a_n(t)\sin(k_n x), \tag{4.1}$$

$$a_n(t) = \frac{-4\omega}{n\pi(\omega^2 - \omega_n^2)}\left[\omega \sin(\omega t) - \omega_n \sin(\omega_n t)\right]. \tag{4.2}$$

4.1. Numerical Results

In this section, we compare the analytic solution (4.1) derived in section 4 with three different discrete approaches:

- The direct transient solution (2.2).
- The modal transient solution (2.4), with six modes.
- The augmented modal transient solution (3.4), with six modes plus one rigid-body mode.

This model problem is an excellent stand-in for our primary intended use case, which is a three-dimensional structure rigidly attached to a single-axis shaker table: although one-dimensional, the string also has internal motion, and boundaries that imply a simple rigid-body motion, i.e., a vector of all ones.

The code, implemented in Matlab, uses Backward Euler time stepping, and a Finite Element Method spatial discretization. It is included in appendix A. The results shown below were generated using a finer discretization (`np=1001`, `dt=1e-6`, `nt=10000`) than is written in the code; the values in appendix A allow the code to be run in real-time, in which case the results are qualitatively similar, but the agreement is not as good. The physical parameters are chosen to mimic a Cello A-string, with a fundamental frequency of 220 Hz. The frequency for motion of the ends is chosen to be $\omega = 2\pi \cdot 330$ Hz, i.e., halfway in between the first and second mode frequencies of the system.

Figures 2 and 3 demonstrate excellent agreement between the analytic solution and the direct transient method. The direct transient curve is difficult to see in the plots because it is covered by the analytic curve, with some small discrepancies in the vicinity of the kink produced by propagation and reflection of the wavefront produced by the end motions. With just six modes, all having zero displacement at the end points, the modal transient solution does a poor job of capturing the correct displacement - except at times when the end displacement is actually zero. As we had hoped, the augmented basis solution does quite well even with just six modes.

We also did some experiments comparing this with the con mass approach described in section 1. However, because this is a Matlab code that uses sparse-direct linear solves with double-precision arithmetic, the known issues with the con mass approach could be observed only in really extreme cases. Further work would be required to replicate and investigate these issues in the context of this example.

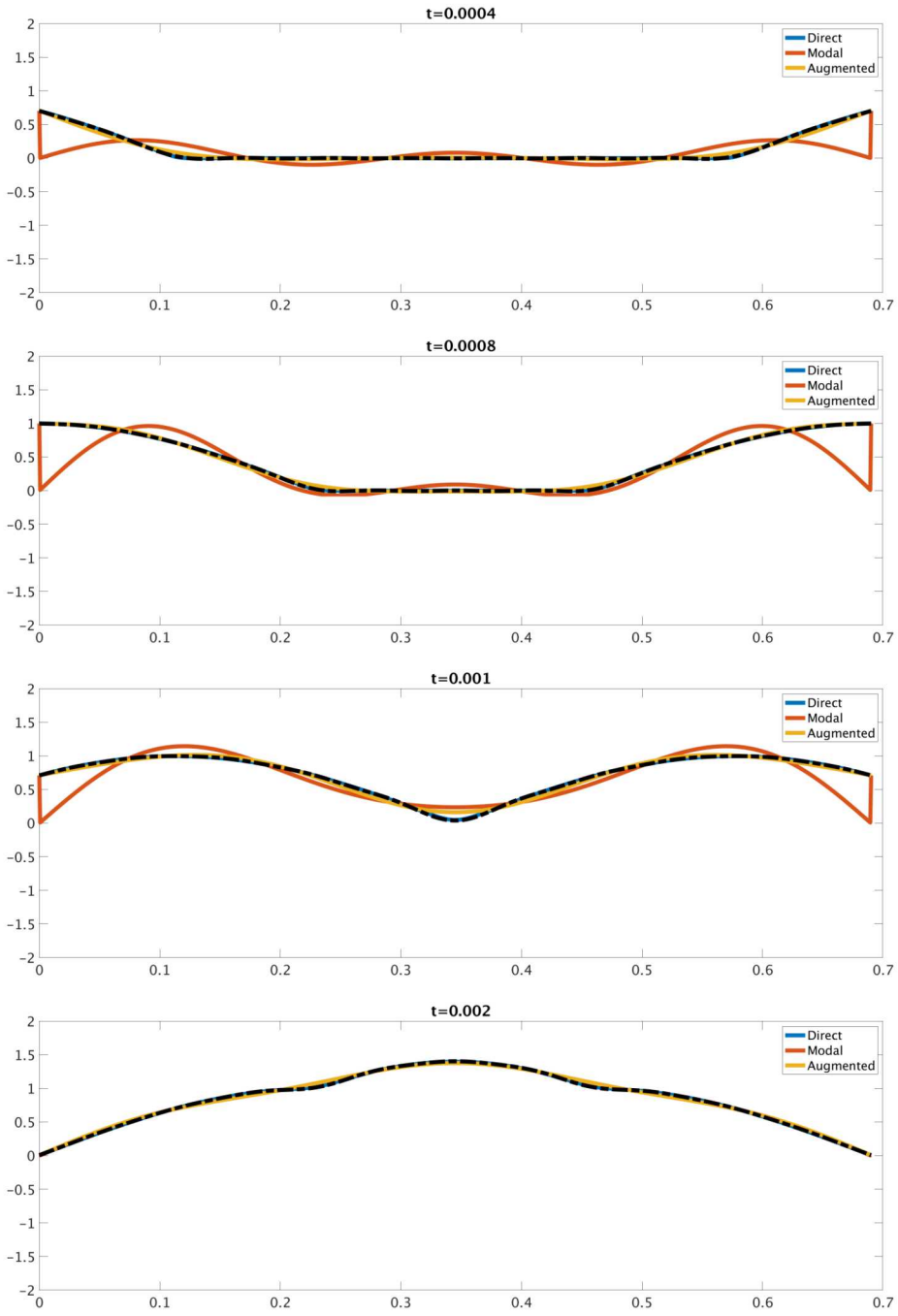


Figure 2. The string at $t = \frac{2\pi}{\omega}\{1, 2, 3, 4\}/8$, i.e. four snapshots over the first half of a cycle of the moving end points. Figure 3 shows the second half of a cycle. The code that produced these plots is given in appendix A. Black: the analytic solution (4.1). Blue: the direct transient solution (2.2). Red: the modal transient solution (2.4), with six modes. Yellow: the augmented modal transient solution (3.4), with six modes plus one rigid-body mode.

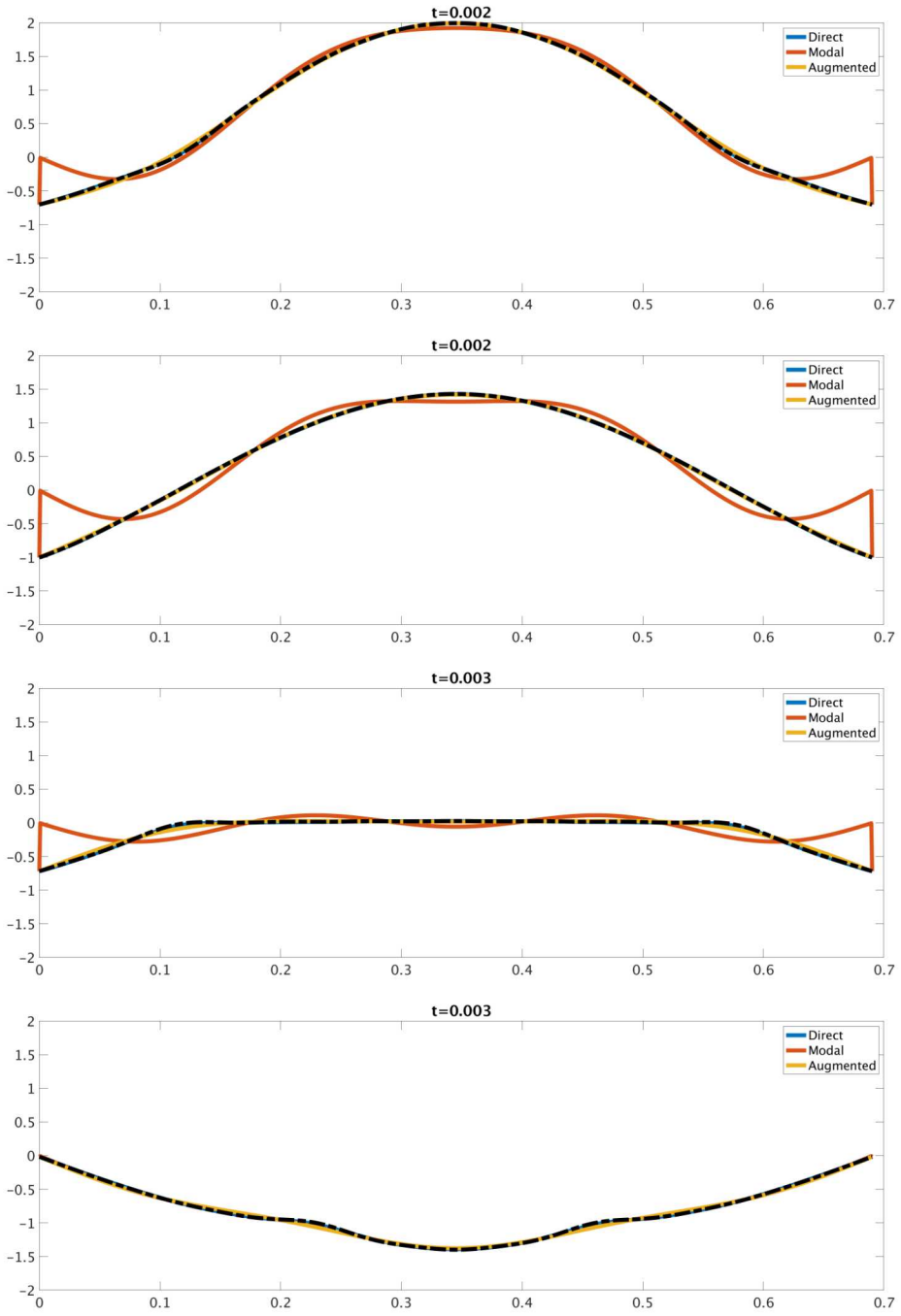


Figure 3. The string at $t = \frac{2\pi}{\omega}\{5, 6, 7, 8\}/8$, i.e. four snapshots over the second half of a cycle of the moving end points. Figure 2 shows the first half of a cycle. The code that produced these plots is given in appendix A. Black: the analytic solution (4.1). Blue: the direct transient solution (2.2). Red: the modal transient solution (2.4), with six modes. Yellow: the augmented modal transient solution (3.4), with six modes plus one rigid-body mode.

5. PATH FORWARD TO PRODUCTION ANALYSIS

The presented modal applied acceleration approach shows excellent results for the 1d problem. In order to gain mission impact from the research, the methodology must be integrated into an analyst-accessible work flow. The best delivery mechanism for this capability is the ASC-funded Sierra Structural Dynamics analysis application. The Sierra code suite is used daily for ND qualification analyses.

Consider equation (2.1) for the partitioned system, repeated here:

$$K_{rr}u_r + M_{rr}\ddot{u}_r = \underbrace{f_r - K_{rp}u_p - M_{rp}\ddot{u}_p}_{\equiv F}.$$

Within Sierra/SD it was found to be fairly straightforward to construct the matrices K_{rr}, M_{rr} and even to set up the eigenvalue problem (2.3). However, the maps required to form the matrices K_{rp}, M_{rp} were found not to be accessible in the part of the code where they were needed. In order to rectify this, a substantial refactor will be required. These same maps for K_{rp}, M_{rp} are also needed for a major FY20 development deliverable in Sierra/SD to report reaction forces from fixed constraints for use in fixture design.

The proposed path forward for this research is to revisit prototyping the modal acceleration application method in Sierra/SD once the reaction force deliverable is completed, then evaluate that augmented basis prototype for viability in more complex 3d problems. If the approach sees the same type of success as seen in 1d productionization of this research path, it should be pursued to create a functional and fully supported analysis capability.

6. CONCLUSIONS

An augmented basis numerical procedure to directly and exactly apply acceleration boundary conditions to modal transient analysis has been documented and demonstrated on a 1d problem. This method shows strong promise for exactly applying specified boundary conditions without analyst intervention. The work needed to integrate the modal basis method to the production Sierra Structural Dynamics analysis module has been scoped. Although significant development work will be required, this effort appears tractable. It is strongly recommended that this research line be continued in order to ultimately provide a more robust, less error-prone, and simpler modal transient analysis methodology to qualification analysis.

REFERENCES

[1] Mikhail Alekseevich Novikov. Simultaneous diagonalization of three real symmetric matrices. *Russian Mathematics*, 58(12):59–69, 2014.

A. MATLAB CODE

This appendix contains the Matlab code that was used to apply the augmented basis approach described in section 3 to the analytic solution developed in section 4. Note that none of these examples incorporate damping terms.

A.1. Physical Parameters

This script was used to generate the physical constants needed to assemble the matrices. The values were chosen to approximate the behavior of a Cello A-string.

```
%% physical parameters (Cello A-string)
tension = 133.447; % N (30 lbf)
len = 0.69; % m
freq = 220; % Hz
density = tension/(2*len*freq)^2; % kg / m
c = sqrt(tension/density); % transverse wave speed
```

A.2. Mesh

These scripts create a mesh that is a uniform subdivision of the string length. In that case, piecewise-linear finite elements are equivalent to finite differences, but it leaves the flexibility to create a non-uniform mesh if desired.

```
function mesh = gen_mesh(a, b, np)

mesh.p = linspace(a, b, np)';
mesh.e = [1 : (np-1); 2 : np]';
```

The following verifies some very simple assumptions about the mesh.

```
function okay = verify_mesh(mesh)

okay = 1;

okay = okay && all(sort(mesh.p) == mesh.p);

ab = mesh.p(mesh.e);

okay = okay && all(ab(:,2) > ab(:,1));
```

This function does the finite element matrix assembly. It uses standard piecewise-linear elements, with Matlab's quadrature applied to a function handle used to assemble the right-hand side vector.

```
function [K,M,F] = assemble(mesh, forcing)

ne = size(mesh.e,1);
np = size(mesh.p,1);
n_entries = 4*ne;

K_entries = zeros(n_entries, 1);
M_entries = zeros(n_entries, 1);
rows = zeros(n_entries, 1);
cols = zeros(n_entries, 1);

F = zeros(np, 1);

for ie = 1:ne
    global_ind = 4*(ie-1) + (1:4);

    points = mesh.e(ie, :);
    loc_rows = points'*ones(1,2);
    loc_cols = ones(2,1)*points;

    a = mesh.p(points(1));
    b = mesh.p(points(2));

    locK = 1/(b-a)*[ 1 -1;
                    -1  1];

    locM = (b-a)/6*[2 1;
                    1 2];

    K_entries(global_ind) = locK(:);
    M_entries(global_ind) = locM(:);
    rows(global_ind) = loc_rows(:);
    cols(global_ind) = loc_cols(:);

    f1 = integral(@(x)(forcing(x).*(1-(x-a)./(b-a))), a, b);
    f2 = integral(@(x)(forcing(x).*((x-a)./(b-a))), a, b);
    F(points) = F(points) + [f1; f2];
end

K = sparse(rows, cols, K_entries, np, np);
M = sparse(rows, cols, M_entries, np, np);
```

A.3. Solution

This code solves the second-order system

$$Ku + M\ddot{u} = F.$$

By adding velocity variables, the system is reduced to first-order form, and then time-stepping is performed with backward Euler.

The matrices are incorporated into a struct so that different types of systems, such as those in equations (2.2), (2.4), and (3.1), can easily be solved and compared simultaneously.

Time steps are plotted as they are computed. Optionally, a function handle can be passed to allow plotting of an analytic solution on top.

```
function solve_second_order_system(p, dt, nt, plot_scale, cases, f_xt,
   do_movie)

plot_frequency = 1;

nc = length(cases);
for ic = 1:nc
    n(ic) = size(cases{ic}.K,1);
    xm{ic} = zeros(2*n(ic),1);
    xi{ic} = zeros(2*n(ic),1);

    if isfield(cases{ic},'v0_ind') && isfield(cases{ic},'v0_val')
        xm{ic}(n(ic) + cases{ic}.v0_ind) = cases{ic}.v0_val;
    end

    cases{ic}.Kx = [cases{ic}.K, sparse(n(ic),n(ic));
                    sparse(n(ic),n(ic)), -speye(n(ic))];
    cases{ic}.Mx = [sparse(n(ic),n(ic)), cases{ic}.M;
                    speye(n(ic)), sparse(n(ic),n(ic))];
    cases{ic}.Fx = [cases{ic}.V'*cases{ic}.F; zeros(n(ic), 1)];
    cases{ic}.Ax = eye(2*n(ic)) + dt*(cases{ic}.Mx\cases{ic}.Kx);
end

err = zeros(nc,1);

set(0, 'DefaultLineLineWidth', 6);
set(0,'defaultAxesFontSize',20)

if do_movie
    vidObj = VideoWriter('movie.avi');
    open(vidObj);
end
```

```
32
33  for ic=1:nc
34      labels{ic} = cases{ic}.label;
35  end
36
37  figure(1), clf
38  for it = 1:nt
39      ti = (it-1)*dt;
40
41      fi = f_xt(p,ti);
42
43      for ic=1:nc
44          xi{ic} = cases{ic}.Ax\(xm{ic} + dt*(cases{ic}.Mx\cases{ic}.Fx)*...
45              cases{ic}.force_time(ti));
46          xm{ic} = xi{ic};
47      end
48
49      if mod(it,plot_frequency) == 0
50          figure(1), clf
51          for ic=1:nc
52              u_plot = cases{ic}.disp_vec*cases{ic}.force_time(ti);
53              u_plot(cases{ic}.free) = cases{ic}.V*xi{ic}(1:n(ic));
54
55              err(ic) = max(err(ic), norm(u_plot - fi));
56
57              plot(p, u_plot), hold on
58          end
59
60          plot(p, fi, 'k-.')
61
62          ylim(plot_scale)
63          legend(labels)
64          drawnow
65          pause(.05)
66          if do_movie
67              currFrame = getframe(gcf);
68              writeVideo(vidObj,currFrame);
69          end
70      end
71  end
72
73  if do_movie
74      close(vidObj);
75  end
76
```

```
for ic=1:nc
    fprintf('Case %d (%s):\n',ic,labels{ic});
    fprintf('cond(M) = %g\n',cond(full(cases{ic}.M)));
    fprintf('max error: %g\n',err(ic))
end
```

The following script implements the analytic solution (4.1).

```
function f = analytic_soln(x, t)

string_parameters

n_terms = 20;
disp_omega = 2*pi*330;

a = zeros(n_terms,1);
b = zeros(n_terms,1);
for n = 1:2:n_terms
    kn = n*pi/len;
    a(n) = -4*disp_omega^2/(n*pi*(disp_omega^2 - (kn*c)^2));
    b(n) =  4*disp_omega*kn*c/(n*pi*(disp_omega^2 - (kn*c)^2));
end

f = ones(size(x)) * sin(disp_omega*t);

for n=1:length(a)
    kn = n*pi/len;
    f = f + a(n)*sin(kn*x)*sin(disp_omega*t);
    f = f + b(n)*sin(kn*x)*sin(kn*c*t);
end
```

This script sets up and compares four different solution methods: direct transient (2.2), modal transient (2.4), the augmented system (3.1), and the analytic solution (4.1).

```
string_parameters

mesh = gen_mesh(0, len, 101);
assert(verify_mesh(mesh));

forcing = @(x)0;
[K,M,~] = assemble(mesh, forcing);

np = size(mesh.p,1);

free = 2:(np-1);
n_free = np-2;
```

```
13
14  K_free = tension*K(free,free);
15  M_free = density*M(free,free);
16
17  n_ew = 6;
18  [Phi,D] = eigs(K_free, M_free, n_ew, 'smallestreal');
19
20  %% prescribed acceleration (at ends)
21
22  disp_omega = 2*pi*330;
23  disp_time = @(t)sin(disp_omega*t);
24  disp_vec = zeros(np,1);
25  disp_vec(1) = 1;
26  disp_vec(end) = 1;
27
28  F_accel = disp_omega^2*density*M(free,:)*disp_vec - ...
29          tension*K(free,:)*disp_vec;
30
31  %% small eigenvalue problem for augmented matrix
32
33  Psi = ones(n_free, 1);
34
35  P = [Phi, Psi];
36  M_hat = P'*M_free*P;
37  K_hat = P'*K_free*P;
38
39  % symmetrize so that eig will return mass-normalized V
40  M_hat = 0.5*(M_hat + M_hat');
41  K_hat = 0.5*(K_hat + K_hat');
42
43  [V,Theta] = eig(K_hat, M_hat);
44
45  n_augment = size(P,2);
46
47  %% transient
48  dt = .00001;
49  nt = 1000;
50
51  base_case.disp_vec = disp_vec;
52  base_case.force_time = disp_time;
53  base_case.free = free;
54
55  direct = base_case;
56  direct.K = K_free;
57  direct.M = M_free;
```

```
direct.F = F_accel;
direct.V = speye(n_free);
direct.label = 'Direct';

modal = base_case;
modal.K = D;
modal.M = speye(n_ew);
modal.F = F_accel;
modal.V = Phi;
modal.label = 'Modal';

augment = base_case;
augment.K = Theta;
augment.M = speye(n_augment);
augment.F = F_accel;
augment.V = P*V;
augment.label = 'Augmented';

cases{1} = direct;
cases{2} = modal;
cases{3} = augment;

plot_scale = [-2 2];
solve_second_order_system(mesh.p, dt, nt, plot_scale, cases, @analytic_soln,
    0);
```

This page intentionally left blank.

DISTRIBUTION

Hardcopy—Internal

Number of Copies	Name	Org.	Mailstop
1	Nathan K. Crane	1542	0845
1	D. Chavez, LDRD Office	1911	0359

Email—Internal

Name	Org.	Sandia Email Address
Technical Library	01177	libref@sandia.gov

This page intentionally left blank.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.