SAND2024-01791

SANDIA REPORT

Printed February 14, 2024

Sandia National Laboratories

Sierra/SD – Theory Manual – 5.18

Sierra Structural Dynamics Development Team

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

ABSTRACT

Sierra/SD provides a massively parallel implementation of structural dynamics finite element analysis, required for high fidelity, validated models used in modal, vibration, static and shock analysis of structural systems. This manual describes the theory behind many of the constructs in **Sierra/SD**. For a more detailed description of how to use **Sierra/SD**, we refer the reader to *User's Manual*.

Many of the constructs in **Sierra/SD** are pulled directly from published material. Where possible, these materials are referenced herein. However, certain functions in **Sierra/SD** are specific to our implementation. We try to be far more complete in those areas.

The theory manual was developed from several sources including general notes, a *programmer_notes* manual, the user's notes and of course the material in the open literature.

This page left blank

CONTENTS

LIST OF FIGURES

This page intentionally left blank.

LIST OF TABLES

This page intentionally left blank.

Acknowledgments

The **Sierra/SD** software package is the collective effort of many individuals and teams. A core Sandia National Laboratories based **Sierra/SD** development team is responsible for maintenance of documentation, testing, and support of code capabilities. This team includes Dagny Beale, Gregory Bunting, David Day, Clark Dohrmann, Payton Lindsay, Justin Pepe, and Julia Plews.

The **Sierra/SD** team also works closely with the Sierra Inverse and Plato teams to jointly enhance and maintain several capabilities. This includes contributions from Ryan Alberdi, Wilkins Aquino, Brett Clark, Sean Hardesty, Clay Sanders, Chandler Smith, Adam Sokolow, Benjamin Treweek, Timothy Walsh, and Ray Wildman.

The **Sierra/SD** team works closely with other Sierra teams on core libraries and shared tools. This includes the DevOps, Sierra Toolkit, Solid Mechanics, Fluid Thermal Teams. Additionally, analysts regularly provide code capabilities as well as help review and verify code capabilities, testing, and documentation. Other individuals not already mentioned directly contributing to the **Sierra/SD** documentation, testing, and code base during the last year include Simon Bignold, Leah Brinkman, Samuel Browne, Michael Buche, Mark Chen, Nathan Crane, Jared Crean, David Glaze, Mark Hamilton, Sidharth Joshi, Andrew Kimler, Dong Lee, Mario LoPrinzi, Kevin Manktelow, Matthew Mosby, Devin OConnor, Tolu Okusanya, Heather Pacella, Krishen Parmar, Kendall Pierson, Tim Shelton, Greg Sjaardema, Matt Staten, Johnathan Vo, Tyler Voskuilen, Alan Williams, and Riley Wilson.

Historically dozens of other Sandia staff, students, and external collaborators have also contributed to the **Sierra/SD** product and its documentation.

Many other individuals groups have contributed either directly or indirectly to the success of the **Sierra/SD** product. These include but are not limited to;

- Garth Reese implemented the original **Sierra/SD** code base. He served as principal investigator and product owner for **Sierra/SD** for over twenty years. His efforts and contributions led to much of the current success of **Sierra/SD**.
- The ASC program at the DOE which funded the initial **Sierra/SD** (Salinas) development as well as the ASC program which still provides the bulk of ongoing development support.
- Line managers at Sandia Labs who supported this effort. Special recognition is extended to David Martinez who helped establish the effort.
- Charbel Farhat and the University of Colorado at Boulder. They have provided incredible support in the area of finite elements, and especially in development of linear solvers.
- Carlos Felippa of U. Colorado at Boulder. His consultation has been invaluable, and includes the summer of 2001 where he visited at Sandia and developed the Hexshell element for us.
- Danny Sorensen, Rich Lehoucq and other developers of ARPACK, which is used for eigenvalue problems.
- Esmond Ng who wrote *sparspak* for us. This sparse solver package is responsible for much of the performance in **Sierra/SD** linear solvers.
- The *metis* team at the University of Minnesota. *Metis* is an important part of the graph partitioning schemes used by several of our linear solvers. These are copyright 1997 from the University of Minnesota.

- Padma Raghaven for development of a parallel direct solver that is a part of the linear solvers.
- The developers of the SuperLU Dist parallel sparse direct linear solver. It is used through GDSW for a variety of problems.
- Leszek Demkowicz at the University of Texas at Austin who provided the HP3D[67] library and has worked with the **Sierra/SD** team on several initiatives. The HP3D library is used to calculate shape functions for higher order elements.

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

1. INTRODUCTION

Solution Spaces. Sierra uses nodal discretizations exclusively. All the degrees of freedom, or **DOF**s, are defined at the nodes. The active DOFs depend on the physics and the boundary conditions. Certain tasks, such as transmitting data between **Sierra/SD** and MATLAB, depend on users converting data between different sets of active DOFs. The documentation of how to perform these tasks assume that the user understands the dimensions of different sets of DOFs.

NASTRAN developed terminology 1.1 for the different sets of dofs, and **Sierra/SD** uses simplified version. To give you an idea, consider a modal analysis of a structure run in serial. Shell elements are mixed with solid elements. No boundary conditions are applied. There are 9938 nodes and 9 MPCs.

To output the required maps and other m-files, in the input deck add to the `outputs` both `mfile` and `ASetMap`. To output the eigenvectors to the **Exodus** file, also add `disp` to `outputs`.

For this model, we have the following dimensions.

1. #nodes=9938
2. full set= #nodes * 9 dofs/node = 89442
3. structural set= #nodes * 6 dofs/node = 59628
4. `G-set` = # active dofs before boundary conditions = 42708
5. `A-set` = analysis set = # equations to be solved = 42699

There are 3 dofs/node for solid elements. Shells and beams have 6. Acoustic, thermal, and electrical DOFs are also included in the G-set. In aggregate, the total number of active dofs is 42708 before boundary conditions and MPCs are applied. There are no boundary conditions in the model, but there are 9 MPC equations, each of which eliminates 1 dof, so the Aset is reduced to 42699.

`*_Disp*.m` files are written in a reduced structural set which may or may not contain the full solution vector, depending on the specifics of the model. These `m-files` use a legacy format which is not well understood by our current development team. Our most robust and user-friendly output is available in `exodus` format.

The matrices `Mssr` and `Kssr` contain the mass and stiffness matrices in the `A-set`. They are symmetric matrices and only one half of the off diagonal is stored. To get the complete matrix within MATLAB,

```
K = Kssr + tril(Kssr,-1)';
```

The full eigenvectors (in the structural set) are available in the output exodus file. To get them use the `SEACAS` command `exo2mat`.

```
> exo2mat example-out.exo
```

Within MATLAB, the data can be converted to a properly shaped matrix.

```
>>> load example-out
>>> phi = zeros(nnodes*6,nsteps);
>>> temp = (0:nnodes-1)*6;
>>> phi(temp+1,:)=nvar01;
>>> phi(temp+2,:)=nvar02;
>>> phi(temp+3,:)=nvar03;
>>> phi(temp+4,:)=nvar04;
>>> phi(temp+5,:)=nvar05;
>>> phi(temp+6,:)=nvar06;
```

We now have phi as a matrix with each column corresponding to an eigenvector. However, phi is dimensioned at 59628 x 10 for this example. Note that 59628 is the number of nodes times 6. We can't multiply phi by K for example - the dimensions don't match. To do this we need a map.

We have one map in our directory. `ASetMap_a.m` is the map from the structural set to the A set. Thus, we can reduce `phi` to the `A-set` by combining it with `ASetMap_a`. Generally the `G-set` map is not output, but is used internally.

```
>>> p2=zeros(max(max(ASetMap_a)),nsteps);
>>> for j=1:nnodes*8
>>>  i=ASetMap_a(j);
>>>  if ( i > 0 )
>>>    p2(i,:)=phi(j,:);
>>>  end
>>> end
```

This is slow. A faster, but less straightforward method is shown here.

```
>>> mapp1=ASetMap_a+1;
>>> temp=zeros(max(max(mapp1)),nsteps);
>>> temp(mapp1,:)=phi;
>>> p2=temp(2:max(max(mapp1)),:);
```

We can do all the neat things like `p2'*K*p2`.

To get back to the structural set, we again use this map. For example, if we have a vector of dimension 42699,

```
>>> x=1:42699';
>>> XX = zeros(59628,1);
>>> for i=1:59628
>>>   if ( ASetMap_a(i)>0 )
>>>     XX(i)=x(ASetMap_a(i));
>>>   end
>>> end
```

An optimization is to do instead

```
>>> temp=[ 0 x'];
>>> X2=temp(mapp1);
```

1.1. Matrix Dimensions: Terminology

The previous section is complicated enough to stand out from other documentation. This section defines some terminology used in the previous section. The various *spaces* are listed in Table 1-1. A discussion of each follows.

Space	Description
Full-set	biggest possible set. 9 * number of nodes
Structural-set	6 * number of nodes This is the space that is typically written to exodus.
Assembly-set	This is the space to which we assemble matrices. It represents those dofs that have been "touched" by elements.
S-set	degrees of freedom eliminated by SPC
Common-set M-set	Assembly minus S-set degrees of freedom eliminated by MPC
Analysis-set	dimension of matrices sent to solvers.

Table 1-1. – Sierra/SD solution spaces.

Full-set This space is referenced by many of our solvers. We then provide a map from this space to the Analysis-set using ASetMap. Every node has 9 degrees of freedom (3 translations, 3 rotations, acoustic, voltage, and thermal). Virtual nodes may have been added to handle generalized dofs.

Structural-set This is identical to the *full-set* except that it contains only structural degrees of freedom (translations and rotations). It and contains all the structural dofs of the model including virtual nodes.

Assembly-set The assembly set is the space to which matrices are assembled. It includes dofs that may later be eliminated by SPCs or MPCs. It includes all dofs that are touched.

$$\text{Assembly-set} = \text{Analysis-set} \cup \text{S-set} \cup \text{M-set}$$

Currently, the only map to the assembly set is found in the node array. However, there is no user interface to the node array.

S-set This is the list of degrees of freedom that are eliminated by single point constraints (SPC).

Common-set The "Common" set includes the Assembly set, with the S-set removed. This set is common to all solvers, in contrast to the analysis set which may have different dimensions for serial and parallel solvers.

M-set This is the list of degrees of freedom that are eliminated using multipoint constraints (or MPCs). When using constraint elimination in serial, the dimension of the problem is reduced by the number of MPC constraints. In contrast, in solvers that use Lagrange multipliers, the stiffness matrix is unchanged by introduction of the constraints. Note however, that the solution vector will include extra Lagrange multipliers.

Analysis-set The analysis set is the matrix dimension that will be sent to the solver. Note that it may depend on the solver. With constraint elimination, the M-set may not be empty, while solvers that use Lagrange multipliers will always have an empty *M-set*.

Solution-set As noted above, in parallel solutions with Lagrange multipliers, we pass a left-hand side matrix of dimension equal to the Analysis set. However, the solution vector returned is of length Analysis-set plus the number of Lagrange multipliers. This is the solution-set length.

G-set Unfortunately, while the sets above are well-defined, the G-set is not. At various times it has been used to refer to the Full, Structural or assembly set. This confusion spreads throughout the documentation and the comments in the notes.

Revised Set definition Example. Consider the problem in Figure 1-1. The model consists of 4 real nodes, one MPC, one superelement (with one generalized dof), and single point constraints sufficient to clamp the left-hand side, and keep the rest of the model in one dimension.

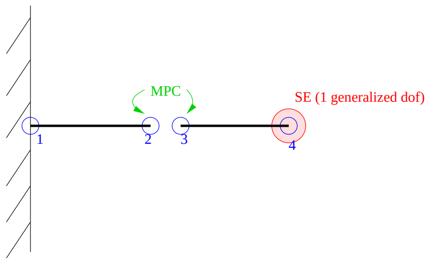


Figure 1-1. – Example for Set Definition.

Full-set There are 4 real nodes, plus 1 virtual node (generated for the generalized dof). Thus,

$$size(Full) = (4 + 1)9 = 45$$

Assembly-set The two elements are beams, with 6 dofs per node. The superelement touches the generalized dof on the virtual node.

$$size(Assembly) = (4)6 + 1 = 25$$

S-set Degrees of freedom are eliminated by clamping 6 dofs on node 1, and by eliminating 5 dofs each on the 3 remaining nodes.

$$size(S) = 6 + 15 = 21$$

Common-set After elimination of the S-set, the common set is,

$$size(Common) = 25 - 21 = 4$$

All solvers use this space initially. The following cases are different for each solver.

M-set The size of the M-set is one, but what that means to the analysis depends on the solver. For serial solvers with constraint elimination, the matrix size is reduced by one. For Lagrange multiplier solvers, we keep our matrices at the same size, but augment the solution space by one Lagrange multiplier.

Analysis-set For serial, constraint elimination solvers, the analysis set is 3. For Lagrange multiplier problems, the left-hand side matrix stays at the Common-set dimension, but constraint equations are passed in separately, and Lagrange multipliers are part of the solution vector.

Solution-set For serial solvers, the Solution-set is always equal to the analysis-set (which is 3 in this example). For Lagrange multiplier solvers, the solution-set in this example is 5.

1.2. Rotational Degrees of Freedom

Beams, shells and some other specialty elements use rotational degrees of freedom (DOF) in addition to the three translational DOF. Rotational DOF permit direct application of moments and allow efficient computations of structural element response such as bending. Rotational DOF are also important for management of rigid bodies. In our applications two methods are used to manage rotational DOF. Full rotation tensors are used for large deformation nonlinear response, while infinitesimal rotations angles are typically used for small strain, linear response such as eigen analysis.

Euler Angles. The rotation of a rigid body is often described using a rotation tensor with for example Euler angles. Note that there are several of definitions of these angles, and that the order of application does matter.

> Euler angles are a means of representing the spatial orientation of any frame of the space as a composition of rotations from a reference frame. In the following the fixed system is denoted in lowercase (x, y, z) and the rotated system is denoted in upper case letters (X, Y, Z).
>
> The definition is Static. The intersection of the xy and the XY coordinate planes is called the line of nodes (N).
>
> α is the angle between the x-axis and the line of nodes.
>
> β is the angle between the z-axis and the Z-axis.
>
> γ is the angle between the line of nodes and the X-axis.
>
> This previous definition is called $z\ x\ z$ convention and is one of several common conventions; others are $x\ y\ z$ and $z\ y\ x$. Unfortunately the order in which the angles are given and even the axes about which they are applied has never been "agreed" upon. When using Euler angles the order and the axes about which the rotations are applied should be supplied.
>
> Euler angles are one of several ways of specifying the relative orientation of two such coordinate systems. Moreover, different authors may use different sets of angles to describe these orientations, or different names for the same angles. Therefore, a discussion employing Euler angles should always be preceded by their definition. (Wikipedia)

In each definition Euler angles use a series of 3 rotations about 3 different axis to represent the orientation of a body in space. For example, in the case of the $z\ x\ z$ convention, these angle define the following rotation matrix.

$$\mathbf{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Because matrix multiplication is not commutative, the solution depends on the order of rotation. Rotation of a vector by this angle is a tensor product with this matrix. i.e. $v' = Rv$.

Infinitesimal Rotational Angles. Here the matrix representation of the cross product is denoted $\textsc{Spin}(u)$. For all $\vec{x}$, there holds $\vec{u} \times \vec{x} = \textsc{Spin}(\vec{u})\vec{x}$.

Most linear, small deformation FE applications apply the small angle approximation. We expand all trigonometric functions as polynomials of their arguments and retain only first order terms in the angles. Thus, $\sin(\theta) \sim \theta$, and cross terms are eliminated. With these approximations, the order of rotation becomes unimportant, and the component contributions to the rotation matrix are commutable. For a rotation about x,y, z of α, β, γ we have:

$$\mathbf{R} = I + \textsc{Spin}\left(\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix}\right).$$

The coordinates are independent of each other. There are obvious limitations, as the approach does not conserve length for larger rotations. This is often apparent in animation of mode shapes; the modes are computed under a small angle approximation, but are often displayed with a finite deformation.

Quaternions. Euler angles and full rotation tensors define the rotations of a body. Computational efficiency is optimized using mathematically equivalent quaternion algebra. **Sierra/SD** uses the full rotation tensor, and **Sierra/SM** uses quaternions.

Linear vs. Nonlinear Solutions. Linear solutions use the infinitesimal rotation angle formulations. All nonlinear solutions maintain a large rotation capability and use the full rotation tensor. Nonlinear solutions using linear elements (or linearized tangent stiffness matrix terms) require conversion between these forms.

Mixed Variable Solutions. Many linear element have been constructed which are for use in some parts of nonlinear applications. For example, a large ship may be include a linearized model of an engine as part of the model. As long as the engine is undergoing small deformations, it is reasonable to employ such a linearized model, even if another part of the ship is subject to large strain and large rotation. In general, **Sierra/SD** allows the user to specify that certain material blocks in a model are linear, even in a nonlinear analysis. This also necessitates translation between these alternate (and non-equivalent) forms.

Incremental Angular Update. Update of the rotation tensor following an incremental solution of a small deformation is accomplished as follows. Let us call the initial rotation tensor, R_{init}. We compute a small rotation increment expressed in terms of its small rotation angles, $(\alpha, \beta, \gamma)^T$. From the rotation increment, we compute a rotation increment quaternion as follows.

$$\begin{aligned} \theta &= \sqrt{\left(\alpha^2 + \beta^2 + \gamma^2\right)} & q_2 &= c\alpha \\ q_1 &= \cos(\theta/2) & q_3 &= c\beta \\ c &= \sin(\theta/2)/\theta & q_4 &= c\gamma \\ & & q &= q/|q| \end{aligned}$$

The quaternion is then converted to a rotation tensor,

$$R_\nabla = \begin{bmatrix} 2(q_1^2 + q_2^2) - 1 & 2(q_2q_3 - q_4q_1) & 2(q_2q_4 + q_3q_1) \\ 2(q_2q_3 + q_4q_1) & 2(q_1^2 + q_3^2) - 1 & 2(q_3q_4 - q_2q_1) \\ 2(q_2q_4 + -q_3q_1) & 2(q_3q_4 + q_2q_1) & 2(q_1^2 + q_4^2) - 1 \end{bmatrix}$$

The updated rotation tensor is,

$$R_{update} = R_{\nabla} R_{init}$$

Thus, the rotation increment is treated as a full angle update.

Consequence for Linear Elements in nonlinear solutions. The consequence of this update is that there may be significant differences between a nonlinear solution and a linear solution, even when both are applied to a linear element. The approximations applied for infinitesimal rotations are significant, and are not reciprocal, i.e. information is lost in that approximation. Nonlinear solutions should permit large rotations with most elements. Linear solutions are valid only in the range of small deformations.

1.3. Mass Properties

Mass properties are computed using the method of Baruch and Zemel.[17] The total mass, location of the center-of-gravity, and the moment of inertia tensor are all calculated for most element types using the mass matrix and a set of rigid-body vectors. However, acoustic elements and superelements use a different procedure. Both methods are discussed below.

Calculations for General Elements The mass properties are computed using rigid-body vectors. At a node, with coordinates (x, y, z), the translational and the rotational rigid-body vectors are,

$$R_x = \begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix}, R_y = \begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix}, R_z = \begin{bmatrix}0\\0\\1\\0\\0\\0\end{bmatrix}, \quad R_{rx} = \begin{bmatrix}0\\-z\\y\\1\\0\\0\end{bmatrix}, R_{ry} = \begin{bmatrix}z\\0\\-x\\0\\1\\0\end{bmatrix}, R_{rz} = \begin{bmatrix}-y\\x\\0\\0\\0\\1\end{bmatrix}.$$

These vectors are assembled on an element level. As an example, for a three-node triangle element,

$$R_{rx} = [0, -z_1, y_1, 1, 0, 0,\ 0, -z_2, y_2, 1, 0, 0,\ 0, -z_3, y_3, 1, 0, 0]^T.$$

The total mass for an element depends on the element mass matrix, $[M_e]$,

$$M = R_x^T M_e R_x = R_y^T M_e R_y = R_z^T M_e R_z.$$

The total mass for the model is computed by summing over all the elements

$$M_{total} = \sum_{i=1}^{Nel} R_x^T [M_e] R_x. \tag{1.3.1}$$

Note that the x, y, and z-direction equations produce the same result. **Sierra/SD** uses the x-direction equation.

In a similar manner, the location of the center-of-gravity can be found by

$$x_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nel} R_{rz}^T [M_e] R_y, \tag{1.3.2}$$

$$y_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nel} R_{rx}^T [M_e] R_z, \tag{1.3.3}$$

$$z_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nel} R_{ry}^T [M_e] R_x. \tag{1.3.4}$$

$$\begin{array}{lll}
I_{xx} = \sum_{i=1}^{Nel} R_{rx}^T [M_e] R_{rx} & I_{xy} = \sum_{i=1}^{Nel} R_{rx}^T [M_e] R_{ry} & I_{xz} = \sum_{i=1}^{Nel} R_{rx}^T [M_e] R_{rz} \\
 & I_{yy} = \sum_{i=1}^{Nel} R_{ry}^T [M_e] R_{ry} & I_{yz} = \sum_{i=1}^{Nel} R_{ry}^T [M_e] R_{rz} \\
 & & I_{zz} = \sum_{i=1}^{Nel} R_{rz}^T [M_e] R_{rz}
\end{array}$$

The mass properties procedure applies to Conmass, Beam2, Truss, TiBeam, Nbeam, Quad4, Quad8, QuadM, Tet4, Tet10, TriaShell, Tria3, Tria6, Hex8, Hex20, Wedge6, and Wedge15 elements.

Acoustic and superelements Although acoustic element blocks are made up of element types listed above, acoustic elements only have 1 degree-of-freedom per node. Thus, the rigid-body vectors presented above cannot be used without modification. Similarly, superelement can have any number of degrees-of-freedom depending on how the element was formed. Because of this, a different method is used to compute mass properties for superelements and acoustic elements.

The mass properties for these elements can be computed with somewhat less accuracy than the method presented above by lumping the mass matrix of each element, then summing the contribution from each node. This is the method implemented in **Sierra/SD**.

The total mass is

$$M_{total} = \sum_{i=1}^{Nnode} M_i$$

where M_i is the mass at node i. The center-of-gravity is

$$x_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nnode} M_i x_i, \tag{1.3.5}$$

$$y_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nnode} M_i y_i, \tag{1.3.6}$$

$$z_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nnode} M_i z_i \tag{1.3.7}$$

where x_i, y_i, and z_i, are the global coordinates of node i. The components of the inertia tensor are,

$$\begin{array}{lll}
I_{xx} = \sum_{i=1}^{Nnode} M_i (y_i^2 + z_i^2), & I_{xy} = - \sum_{i=1}^{Nnode} M_i x_i y_i, & I_{xz} = - \sum_{i=1}^{Nnode} M_i x_i z_i, \\
 & I_{yy} = \sum_{i=1}^{Nnode} M_i (x_i^2 + z_i^2) & I_{yz} = - \sum_{i=1}^{Nnode} M_i y_i z_i \\
 & & I_{zz} = \sum_{i=1}^{Nnode} M_i (x_i^2 + y_i^2),
\end{array}$$

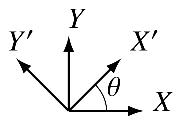


Figure 1-2. – Original and rotated coordinate frames.

1.4. Coordinate Systems

Coordinate systems are provided for some applications including:

1. specification of boundary constraints (SPCs)
2. specification of multi-point constraints (MPCs)
3. specification of material property rotations for anisotropic materials.
4. specification of spring directions (see subsection 5.18).
5. specification of output coordinate systems (in history files only).

Coordinate systems are not supported for other applications including

1. specification of nodal locations,
2. specification of new coordinate systems in any but the basic system.

Coordinate systems for cartesian, cylindrical and spherical coordinates may be defined. In the case of non-cartesian systems, the XZ plane is used for defining the origin of the θ direction only.

Each new coordinate system X' carries with it a rotation matrix, R, that rotates to the basic coordinate system X to the new coordinate system

$$X' = RX.$$

R is a function of the current spatial location except in the cartesian system, in which case R is constant, orthonormal, and

$$X = R^T X', \qquad R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.4.1}$$

For example consider the cartesian system as shown in Figure 1-2. The new system (marked by primes) is rotated by θ from the old system with the new X' axis in the first quadrant of the old system.

This page intentionally left blank.

2. STRUCTURAL SOLUTION PROCEDURES

Among the mechanics codes developed at *Sandia National Labs***Sierra/SD** has the unique ability to combine a variety of different solution procedures. These range from modal superposition based solutions to nonlinear transient. As described in the *User's Manual*, these solutions can be combined (or chained) in solution cases. This section describes the theory behind individual procedures. Details about particular finite elements are provides in Section 5.

2.1. Linear transient analysis

For linear and nonlinear transient dynamics, the time integrator in **Sierra/SD** is either the Newmark-Beta method or the generalized alpha method. The Generalized Alpha method supersedes the Hilbert-Hughes-Taylor method.

Linear structural analysis finite element discretization of the momentum equation, with external load F^{ext}, leads to the differential equation

$$Ma(t) + \hat{C}v(t) + Kd(t) = F^{ext}(t), \quad v = \dot{d}, \quad a = \ddot{d},$$

where damping matrix $\hat{C} = C + \alpha M + \beta K$ is the is the sum of the standard damping matrix C (say from a dashpot) and proportional damping terms. In the generalized alpha method the state at the $n+1$st time step is determined from

$$\begin{array}{rcl} M\left[(1-\alpha_m)a_{n+1} + \alpha_m a_n\right] & + & \hat{C}\left[(1-\alpha_f)v_{n+1} + \alpha_f v_n\right] + \\ K\left[(1-\alpha_f)d_{n+1} + \alpha_f d_n\right] & = & (1-\alpha_f)F^{ext}(t_{n+1}) + \alpha_f F^{ext}(t_n) \end{array} \tag{2.1.1}$$

The parameters α_f and α_m are constrained to achieve second-order accuracy and maintain unconditional stability,

$$\begin{array}{c} \alpha_m < \alpha_f \leq \frac{1}{2} \\ \gamma_n = \frac{1}{2} - \alpha_m + \alpha_f \\ \beta_n \geq \frac{1}{4} + \frac{1}{2}(\alpha_f - \alpha_m) \end{array}$$

By specifying the input parameter $0 \leq \rho \leq 1$, the user selects parameters satisfying these constraints

$$\begin{array}{rcl} \alpha_f & = & \rho/(1+\rho) \\ \alpha_m & = & (2\rho - 1)/(1+\rho) \\ \beta_n & = & (1-\alpha_m+\alpha_f)\cdot(1-\alpha_m+\alpha_f)/4 \\ \gamma_n & = & 1/2 - \alpha_m + \alpha_f \end{array}$$

In the *maximally* damped, $\rho = 0$, note that $\alpha_f = 0$ and $\alpha_m = -1$. The *undamped* case is $\rho = 1$, at which $\alpha_f = \alpha_m = \frac{1}{2}$, which yields $\beta_n = \frac{1}{4}$, and $\gamma_n = \frac{1}{2}$ as in the undamped Newmark-beta method. For later use, we also define

$$F^{ext}_{n+1+\alpha_f} = (1-\alpha_f)F^{ext}(t_{n+1}) + \alpha_f F^{ext}(t_n) \tag{2.1.2}$$

There are two options for evaluating $F^{ext}_{n+1+\alpha_f}$. More will be given on this in Section 2.1.2.

While the displacements and velocities resulting from the generalized alpha method are second-order accurate, accelerations are only first order accurate. [1] Fortunately, second-order accuracy can be obtained for accelerations through an observation that,

$$\alpha_f a_n^{post} + \left(1-\alpha_f\right) a_{n+1}^{post} = \alpha_m a_n + \left(1-\alpha_m\right) a_{n+1}, \tag{2.1.3}$$

where a^{post} is the second-order accurate post processed acceleration. The above equation is implemented by storing the additional vector a_n^{post} so that the updated a_{n+1}^{post} can be computed and output by the code.

Sierra/SD uses the undamped Newmark-beta method if no damping parameter is specified (in the input file),

$$\alpha_f = \alpha_m = 0,\;\; \beta = \frac{1}{4}, \gamma = \frac{1}{2}, \quad M a_{n+1} + \hat{C} v_{n+1} + K d_{n+1} = F^{ext}(t_{n+1}).$$

In terms of the Newmark parameters β_n and γ_n, the time integration scheme is

$$\begin{aligned} d_{n+1} = &\; d_n + \Delta t v_n + \frac{\Delta t^2}{2}\left[(1-2\beta_n)a_n + 2\beta_n a_{n+1}\right] \\ v_{n+1} = &\; v_n + \Delta t\left[(1-\gamma_n)a_n + \gamma_n a_{n+1}\right] \end{aligned} \tag{2.1.4}$$

To derive the displacement-based implementation, first solve these equations for the acceleration and velocity in terms of displacement,

$$\begin{aligned} a_{n+1} &= \tfrac{1}{\beta_n \Delta t^2}\left[d_{n+1} - d_n - v_n \Delta t\right] - \tfrac{1-2\beta_n}{2\beta_n} a_n \\ v_{n+1} &= v_n + \Delta t\left[(1-\gamma_n)a_n + \gamma_n a_{n+1}\right] \\ &= v_n + \Delta t\left[(1-\gamma_n)a_n + \tfrac{\gamma_n}{\beta_n \Delta t^2}\left[d_{n+1} - d_n - v_n\Delta t\right] - \gamma_n \tfrac{1-2\beta_n}{2\beta_n} a_n\right] \end{aligned} \tag{2.1.5}$$

Substitute equation (2.1.5) into equation (2.1.1) and collect terms to obtain for the undamped Newmark-beta method

$$\begin{aligned} \left[M\tfrac{1}{\beta_n\Delta t^2} + \hat{C}\tfrac{\gamma_n}{\beta_n \Delta t} + K\right] d_{n+1} = &\; F_{n+1}^{ext} + \\ & -\hat{C}\left[v_n + \Delta t(1-\gamma_n)a_n - \tfrac{\gamma_n}{\beta_n\Delta t}\left[d_n + \Delta t v_n\right] - \tfrac{\gamma_n \Delta t(1-2\beta_n)}{2\beta_n} a_n\right] + \\ & +M\left[\tfrac{1}{\beta_n \Delta t^2}\left[d_n + v_n\Delta t\right] + \tfrac{1-2\beta_n}{2\beta_n} a_n\right] \end{aligned}$$

or for the generalized alpha method,

$$\begin{gathered} \left[M\tfrac{(1-\alpha_m)}{\beta_n\Delta t^2} + \hat{C}(1-\alpha_f)\tfrac{\gamma_n}{\beta_n\Delta t} + K(1-\alpha_f)\right] d_{n+1} = \\ F_{n+1+\alpha_f}^{ext} - K\alpha_f d_n \\ -\hat{C}\left[\alpha_f v_n + (1-\alpha_f)\left[v_n + \Delta t(1-\gamma_n)a_n + \tfrac{\gamma_n}{\beta_n\Delta t}\left[-d_n - \Delta t v_n\right] - \tfrac{\gamma_n\Delta t(1-2\beta_n)}{2\beta_n} a_n\right]\right] \\ +M\left[-\alpha_m a_n + \tfrac{1-\alpha_m}{\beta_n\Delta t^2}\left[d_n + v_n\Delta t\right] + (1-\alpha_m)\tfrac{1-2\beta_n}{2\beta_n} a_n\right] \end{gathered} \tag{2.1.6}$$

There are three matrix-vector products on the right-hand side of this equation, one for each of the system matrices M, K, and C.

[1] see AlphaStudy.doc in **Sierra/SD** documentation, for details on convergence and post processing discussed here.

2.1.1. *Predictor Corrector Adjustment*

The linear system in 2.1.6 can be solved using high-performance linear iterative solvers such as GDSW. In this context, it would be beneficial to take the initial iterate closer to the expected solution to increase the efficiency of the solver. Thus, the system, which is of the form $\mathbf{A}\mathbf{d}_{n+1} = \mathbf{r}_{n+1}$, can be solved using the following steps:

$$\begin{aligned} \mathbf{d}_{ext} &= \mathbf{d}_n + \Delta t \mathbf{v}_n + \tfrac{\Delta t^2}{2}\mathbf{a}_n, \\ \bar{\mathbf{r}} &= \mathbf{r}_{n+1} - \mathbf{A}\mathbf{d}_{ext}, \\ \mathbf{A}\bar{\mathbf{d}} &= \bar{\mathbf{r}}, \\ \mathbf{d}_{n+1} &= \bar{\mathbf{d}} + \mathbf{d}_{ext}. \end{aligned} \tag{2.1.7}$$

In the above $\mathbf{d}_{ext}$ is the initial estimate of $\mathbf{d}_{n+1}$, obtained using Taylor series extrapolation (essentially assuming that the acceleration remains unchanged in the current time step). We noticed that *the above predictor-corrector implementation 2.1.7 is crucial to ensure that accurate results are obtained for realistic relative solver tolerances* (direct implementation of 2.1.6 could result in high-frequency oscillations that can pollute the solution even after applying filters). Naturally, the approach 2.1.7 also results in accelerated convergence of the GDSW solver resulting in computational savings.

Unfortunately, the predictor-corrector implementation in 2.1.7 resulted in an undesirable side effect, namely growth in error in the constraint equations. The relative error for displacement constraints appear to grow as $n^{1.5}$, where n is the number of time steps, but the reason is not clear at this time. However, a simple modification of the predictor expression by eliminating the velocity and acceleration terms appear to make the growth milder, proportional to $\sqrt{n}$, and is thus employed in the code:

$$\begin{aligned} \mathbf{d}_{ext} &= \mathbf{d}_n \\ \bar{\mathbf{r}} &= \mathbf{r}_{n+1} - \mathbf{A}\mathbf{d}_{ext}, \\ \mathbf{A}\bar{\mathbf{d}} &= \bar{\mathbf{r}}, \\ \mathbf{d}_{n+1} &= \bar{\mathbf{d}} + \mathbf{d}_{ext}. \end{aligned} \tag{2.1.8}$$

2.1.2. *Prescribed Accelerations*

Prescribed accelerations can be applied in **Sierra/SD** to nodesets or sidesets, as described in *User's Manual*. Here we give a brief description of the theory behind the implementation.

To simplify matters, we consider the case when the acceleration of a single DOF is prescribed as $a_o f(t)$, where a_o is the amplitude, and $f(t)$ is the function describing the time dependence. The extension to multiply prescribed DOFs is a matter of an external loop.

Given $f(t)$, we compute two numerical integrals as follows.

$$\begin{aligned} a(t) &= a_o f(t) \\ v(t) &= v_0 + \int_0^t a(t) = v_0 + \int_0^t a_o f(t)dt = v_0 + a_o(\smallint f(t)) \\ d(t) &= d_0 + \int_0^t v(t)dt = d_0 + v_0 t + \int_0^t \int_0^t a_o f(t)dt = d_0 + v_0 t + a_o(\smallint\smallint f(t)) \end{aligned} \tag{2.1.9}$$

where we have defined $\smallint f(t)$ and $\smallint\smallint f(t)$ to denote the first and second integrals of the function $f(t)$, and d_0 and v_0 denote the initial displacement and velocity. $\smallint f(t)$ and $\smallint\smallint f(t)$ are computed numerically in **Sierra/SD**.

Given these functions, we can statically condense the prescribed degrees of freedom, and bring the resulting terms to the right-hand side. First, we define m_i to be the column of the mass matrix associated with the prescribed dof, and c_i and k_i are similarly defined for the damping and stiffness matrices. We first write the Gset version of equation 2.1.1. We put subscripts of g on the system matrices and right-hand side to denote that the prescribed boundary conditions have not yet been eliminated (hence are Gset).

$$\begin{aligned} M_g\left[(1-\alpha_m)a_{n+1}+\alpha_m a_n\right] &+ \hat{C}_g\left[(1-\alpha_f)v_{n+1}+\alpha_f v_n\right]+ \\ K_g\left[(1-\alpha_f)d_{n+1}+\alpha_f d_n\right] &= (1-\alpha_f)F_g^{ext}(t_{n+1})+\alpha_f F_g^{ext}(t_n) \end{aligned} \tag{2.1.10}$$

Next, condense out the prescribed DOFs and move their contributions to the right-hand side, noting that fixed DOFs do not contribute. As this reduces the system matrices to Aset form, the subscripts are dropped. To reduce everything to the Aset, the right-hand side terms are also condensed.

$$M\left[(1-\alpha_m)a_{n+1}+\alpha_m a_n\right]+\hat{C}\left[(1-\alpha_f)v_{n+1}+\alpha_f v_n\right] \tag{2.1.11}$$

$$+K\left[(1-\alpha_f)d_{n+1}+\alpha_f d_n\right]= \tag{2.1.12}$$

$$(1-\alpha_f)F^{ext}(t_{n+1})+\alpha_f F^{ext}(t_n) \tag{2.1.13}$$

$$-(1-\alpha_f)a_o\left[f(t_{n+1})m_i+\textstyle\int f(t_{n+1})c_i+\iint f(t_{n+1})k_i\right] \tag{2.1.14}$$

$$-\alpha_f a_o\left[f(t_n)m_i+\textstyle\int f(t_n)c_i+\iint f(t_n)k_i\right]. \tag{2.1.15}$$

$$\tag{2.1.16}$$

I.e., prescribed accelerations add to the right-hand side both the column of M corresponding to the prescribed dof scaled by the time function $f(t)$ and also the corresponding columns from $\hat{C}$ and K scaled by $\int f(t)$ and $\int\int f$ respectively. For statics problems, this procedure reduces to only a contribution from K, and this is also included in **Sierra/SD**.

2.1.3. *Nonlinear transient analysis*

Nonlinear transient simulations use an algorithm that builds the standard nonlinear transient procedure[19] on the generalized alpha integration instead of the Newmark-beta integration. The equation of motion is

$$\begin{aligned} M\left[(1-\alpha_m)a_{n+1}+\alpha_m a_n\right]+\hat{C}\left[(1-\alpha_f)v_{n+1}+\alpha_f v_n\right]+ \\ (1-\alpha_f)F_{n+1}^{int}+\alpha_f F_n^{int}=(1-\alpha_f)F^{ext}(d_{n+1})+\alpha_f F^{ext}(d_n) \end{aligned} \tag{2.1.17}$$

where F_{n+1}^{int} and F_n^{int} are the internal forces at the current and previous time steps, respectively. Note that the external loads may depend on displacement, as in the case of follower loads.

Before proceeding, note that there are variants of a nonlinear generalized alpha method. due to the nonlinearity of F^{ext} and F^{int}. Equation 2.1.17 interpolates the current and previous displacements,

$$\begin{aligned} F_{n+1+\alpha_f}^{int} &= (1-\alpha_f)F^{int}(d_{n+1})+\alpha_f F^{int}(d_n) \\ F_{n+1+\alpha_f}^{ext} &= (1-\alpha_f)F^{ext}(d_{n+1})+\alpha_f F^{ext}(d_n). \end{aligned} \tag{2.1.18}$$

instead of the interpolated displacement,

$$\begin{aligned} F_{n+1+\alpha_f}^{int} &= F^{int}((1-\alpha_f)d_{n+1}+\alpha_f d_n) \\ F_{n+1+\alpha_f}^{ext} &= F^{ext}((1-\alpha_f)d_{n+1}+\alpha_f d_n), \end{aligned} \tag{2.1.19}$$

Comparisons have shown little difference in the results on simple test problems.

Typically, the external load F^{ext} is a piece-wise linear function of time, in which case the two variants are equivalent, with a couple notable exceptions. First, the two variants yield different loads if two consecutive time steps are in two different linear segments. Second, if polynomial or loglog interpolation functions used instead of linear interpolation, the two variants yield different loads. For problems with very large time steps or involving nonlinear interpolation, different results are to be expected.

Using the tangent stiffness method, we replace F^{int}_{n+1} as

$$F^{int}_{n+1} = F^{int}_{n} + K_t \Delta d \tag{2.1.20}$$

where K_t is the tangent stiffness matrix, defined as $K_t = \partial F^{int}/\partial u$, and $\Delta d = d_{n+1} - d_n$. Also, we use equations 2.1.5, which are the same as in the linear case.

First, we substitute equations 2.1.5 and 2.1.20 into equation 2.1.17. This results in the following equations, which are almost identical to the ones from the linear case

$$\left[M\frac{(1-\alpha_m)}{\beta_n \Delta t^2} + \hat{C}(1-\alpha_f)\frac{\gamma_n}{\beta_n \Delta t} + K_t(1-\alpha_f)\right] d_{n+1} =$$

$$F^{ext}_{n+1+\alpha_f} - \alpha_f F^{int}_n - (1-\alpha_f)\left[F^{int}_n - K_t d_n\right]$$

$$-\hat{C}\left[\alpha_f v_n + (1-\alpha_f)\left[v_n + \Delta t(1-\gamma_n)a_n + \frac{\gamma_n}{\beta_n \Delta t}\left[-d_n - \Delta t v_n\right] - \frac{\gamma_n \Delta t(1-2\beta_n)}{2\beta_n}a_n\right]\right]$$

$$+M\left[-\alpha_m a_n + \frac{1-\alpha_m}{\beta_n \Delta t^2}\left[d_n + v_n \Delta t\right] + (1-\alpha_m)\frac{1-2\beta_n}{2\beta_n}a_n\right]$$

Finally, we want the unknown to be $\Delta d = d_{n+1} - \hat{d}$, where $\hat{d}$ is the current iterate of displacement. To accomplish this, we subtract the appropriate terms from both sides, which yields, after collecting terms

$$\begin{aligned}\left[M\frac{(1-\alpha_m)}{\beta_n \Delta t^2} + \hat{C}(1-\alpha_f)\frac{\gamma_n}{\beta_n \Delta t} + K_t(1-\alpha_f)\right]\Delta d = \\ F^{ext}_{n+1+\alpha_f} - (1-\alpha_f)\hat{F}^{int} - \alpha_f F^{int}_n - C\left[(1-\alpha_f)\hat{v} + \alpha_f v_n\right] \\ -M\left[(1-\alpha_m)\hat{a} + \alpha_m a_n\right]\end{aligned} \tag{2.1.21}$$

where again hats denote current iterates of acceleration, velocity, etc. Note that we have re-defined $\Delta d = d_{n+1} - \hat{d}$, which is different than the previous definition that was given. Also, we note that $\hat{F}^{int} = F^{int}_n + K_t(\hat{d} - d_n)$.

With the Newmark-beta time integrator ($\gamma_n = \frac{1}{2}$, $\beta_n = \frac{1}{4}$, $\alpha_f = \alpha_m = 0$, equation 2.1.21 reduces to

$$\left[M\frac{4}{\Delta t^2} + \hat{C}\frac{2}{\Delta t} + K_t\right]\Delta d = F^{ext}_{n+1} - \hat{F}^{int} - C\hat{v} - M\hat{a} \tag{2.1.22}$$

which is the same equation given by Belytschko et al.[19]

We note that equation 2.1.21 can be written as

$$A\Delta d = res \tag{2.1.23}$$

where A is the dynamic matrix, Δd is the change in displacement from the previous Newton iteration to the current Newton iteration, and res is the residual, i.e. the amount by which the equations of motion (equation 2.1.17) are not satisfied by the current iterate. The residual can be written from the previous equations as

$$res = F^{ext}_{n+1} - \hat{F}^{int} - C\hat{v} - M\hat{a} \tag{2.1.24}$$

2.1.3.1. Nonlinear Transient Analysis with Constraints

In the previous section, the assumption was made that there were no multi-point constraint equations. These extra equations introduce Lagrange multipliers that need to be included in the nonlinear equations. In this section, we will describe how to include constraint equations into the nonlinear solution method based on Newton's method.

Equation 2.1.23 is correct if there are no constraint equations in the problem. When constraint equations are involved, we will show that this generalizes to the following

$$\begin{bmatrix} A & G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} \Delta d \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} res \\ 0 \end{bmatrix} \tag{2.1.25}$$

where the residual is defined with an additional term due to the constraints

$$res = F_{n+1}^{ext} - \hat{F}^{int} - C\hat{v} - M\hat{a} - G^T\hat{\lambda} \tag{2.1.26}$$

where G is the matrix representation of the constraint equations, $\hat{\lambda}$ is the current Newton iterate of the Lagrange multipliers, and $G^T\hat{\lambda}$ represents a force due to constraints. Note that when the problem has no constraint equations, equations 2.1.25 and 2.1.26 reduce to equations 2.1.23 and 2.1.24.

We can arrive at equations 2.1.25 through some simple arguments similar to the unconstrained case. The second equation

$$G\Delta d = Gd_{n+1} - G\hat{d} = 0 \tag{2.1.27}$$

is a simple argument that the linear solver always returns solutions that satisfy $Gd = 0$, and thus the difference $Gd_{n+1} - G\hat{d}$ must also be zero.

The first equation can be deduced by including an additional constraint force term into the residual equation. We will work with the Newmark method, i.e. $\gamma_n = \frac{1}{2}, \beta_n = \frac{1}{4}, \alpha_f = \alpha_m = 0$ to keep the discussion simple. The case with the generalized alpha method is a simple extension of what follows. We write the total internal force, including constraint force terms, as

$$F_{tot}(\hat{d},\hat{\lambda}) = F^{int}(\hat{d}) + M\hat{a} + C\hat{v} + G^T\hat{\lambda} \tag{2.1.28}$$

The incremented total force is given by

$$\begin{aligned} F_{tot}(d_{n+1},\lambda_{n+1}) &= F_{tot}(\hat{d},\hat{\lambda}) + \frac{\partial F_{tot}}{\partial \hat{d}}\Delta d + \frac{\partial F_{tot}}{\partial \hat{\lambda}}\Delta\lambda \\ &= F_{tot}(\hat{d},\hat{\lambda}) + A\Delta d + G^T\Delta\lambda \end{aligned}$$

The force balance says that

$$F_{n+1}^{ext} = F_{tot}(d_{n+1},\lambda_{n+1}) \tag{2.1.29}$$

Simplifying, we obtain

$$A\Delta d + G^T\Delta\lambda = F_{n+1}^{ext} - \hat{F}^{int} - C\hat{v} - M\hat{a} - G^T\hat{\lambda} \tag{2.1.30}$$

which corresponds to the first equation in the system of equations given by equation 2.1.25.

Damping Source	Discussion
linear dashpots	Contributes directly to the C matrix described in equation 2.1.1. The matrix is constant.
proportional damping	Also, known as Rayleigh damping, $$\alpha M_o + \beta K_o$$ The damping is proportional to velocity. Note that the effective damping matrix is constant. Damping is *not* proportional to the tangent matrix, K_t.
linear viscoelasticity	Determined by material parameters.
nonlinear energy loss	Many nonlinear elements contribute to this form of damping. It does not generate a damping matrix term, and often moves energy from lower frequencies to higher frequencies. An example is the Iwan element.
nonlinear material	Similar to nonlinear elements.
numerical damping	No damping matrix is generated. Most of the energy loss is at frequencies above the Nyquist frequency. Controlled by parameter `RHO`.

Table 2-1. – Sources of Damping in the Solution.

2.1.3.2. Damping in Nonlinear Solutions

Some sources of damping in the solution of linear and nonlinear solutions have been identified. It is useful to list them for comparison, as in Table 2-1. Note in particular, that proportional damping, common in linear systems, requires a different definition in nonlinear systems, and will also require explicit formation of a damping matrix.

2.2. Damping of Flexible Modes Only

Here we outline the method used in **Sierra/SD** to ensure that various damping models do not affect the rigid body response of a structure. [2]. A more detailed explanation of the theory which involves less restrictive assumptions and describes connections with the present approach can be found in the document *dampFlexMode.tex*, which appears in the **Sierra/SD** documents repository. The sensitivity of this approach to errors in the K is discussed in *filterrbm_error.tex*.

Consider the standard equilibrium equations given by

$$M\ddot{x} + C\dot{x} + Kx = f, \qquad (2.2.1)$$

where M is the mass matrix, C is the damping matrix, K is the stiffness matrix, x is the response vector, and f is the applied force vector. Let the columns of the matrix Φ_r span the rigid body modes of the structure.

[2]The technique is also known as filtering the rigid body modes, hence the name `filterRBM`

That is,

$$K\Phi_r = 0. \tag{2.2.2}$$

Typically, there are six rigid body modes (3 translational and 3 rotational), and it is assumed this is the case. Consider next a proportional damping model in which

$$C = \alpha K + \beta M, \tag{2.2.3}$$

where α and β are non-negative constants. Since the mass matrix M is nonsingular, we will have $C\Phi_r \neq 0$ for mass proportional damping when $\beta > 0$. Thus, the damping model will dissipate the energy of the rigid body modes. Some analysts would like to include mass proportional damping, but only have it damp the flexible modes.

We may express the response vector x as

$$x = \Phi_r q_r + \Phi_f q_f, \tag{2.2.4}$$

where q_r and q_f are vectors of generalized coordinates associated with the rigid body and flexible modes, respectively. Further,

$$\Phi_f^T M \Phi_r = 0. \tag{2.2.5}$$

Substituting (2.2.4) into (2.2.1), using (2.2.2), and setting

$$C\Phi_r = 0 \tag{2.2.6}$$

gives us

$$M(\Phi_r \ddot{q}_r + \Phi_f \ddot{q}_f) + C\Phi_f \dot{q}_f + K\Phi_f q_f = f. \tag{2.2.7}$$

First assume that C and K are symmetric. We then find from (2.2.2) and (2.2.6) that

$$\Phi_r^T C = 0, \quad \Phi_r^T K = 0, \tag{2.2.8}$$

Pre-multiplying (2.2.7) by Φ_r^T and substitution of (2.2.5) and (2.2.8) gives us

$$\Phi_r^T M \Phi_r \ddot{q}_r = \Phi_r^T f. \tag{2.2.9}$$

If the rigid body modes are M-orthonormal, i.e. $\Phi_r^T M \Phi_r = I$, we then obtain

$$\ddot{q}_r = \Phi_r^T f. \tag{2.2.10}$$

Substituting (2.2.10) into (2.2.7) and using the notation $x_f = \Phi_f q_f$ gives us

$$M\ddot{x}_f + C\dot{x}_f + Kx_f = (I - M\Phi_r\Phi_r^T)f. \tag{2.2.11}$$

From (2.2.4) we see that the total response is given by

$$x = \Phi_r q_r + x_f, \tag{2.2.12}$$

where the dynamics associated with q_r and x_f are governed by (2.2.10) and (2.2.11).

Notice that the dynamics for the flexible part of the response, i.e. (2.2.11), is the original equilibrium equations in (2.2.1) with a modified force vector. This modified for vector can be calculated efficiently as

$$(I - M\Phi_r\Phi_r^T)f = f - M(\Phi_r(\Phi_r^T f)). \tag{2.2.13}$$

The rigid body response governed by (2.2.10) can be numerically integrated using the same scheme as for the flexible response.

If f is a known force vector that does not depend on the response, then we do not need to concern ourselves with stability issues since all we've done is modified the force vector in a *stable* manner. If, however, the force vector depends on the response, then stability issues could arise. It should be mentioned though that these potential issues could arise even in our existing capabilities for coupling **Sierra/SD** to other simulation codes that do not use the present damping approach.

Usability Question Certain expedient spatial discretizations of floating structures lead to a stiffness matrix $\tilde{K}$ with the nonphysical property $\tilde{K}\Phi \neq 0$. Given M, C and $\tilde{K}$, f determines $\tilde{x}$. If, moreover, the rigid body modes Φ are undamped, we get a solution y. Is y "better" than $\tilde{x}$? A cumbersome discretization determines K such that

$$\Phi_r^T K = 0, \quad K\Phi_r = 0. \tag{2.2.14}$$

In practice $K = \tilde{K} - VV^T$ the matrices differ by a symmetric low rank perturbation, and VV^T is sparse.

Our fundamental tool is

$$P = I - \Phi_r \Phi_r^T M.$$

In general neither $P^T\tilde{K}$ nor $\tilde{K}P$ satisfies equation (2). If there exists H such that $\tilde{K}\Phi = M\Phi H$, then (not obvious) $P^T\tilde{K} = \tilde{K}P$. Using filterrbm is like transforming $\tilde{K}$ to $P^T\tilde{K}P = K + P^T VV^T P$. This has the advantage of projecting out the rigid body modes from V.

2.3. Random Vibration

Details of random vibration analysis are presented in several papers[3]. These few paragraphs document what was implemented.

2.3.1. Algorithm

Initially a model decomposition is determined, $K\Phi = M\Phi\Omega^2$ normalized so that $\Phi^T M \Phi = I$. For $j = \sqrt{-1}$, the modal frequency response is,

$$q_i(f) = \frac{1}{\omega_i^2 - \omega^2 + 2j\omega\omega_i\gamma_i}, \quad f = \frac{\omega}{2\pi}.$$

Note that if other damping (such as mass and stiffness proportional damping) is used, then the effective γ_i is used here. For the ath load and the ith mode shape, define

$$Z_a^i = \sum_k \phi_{ik} F_k^a = \langle \phi_i, F^a \rangle.$$

$Z = \Phi^T F$ contains the spatial contributions from the mode shapes and is also frequency independent. The number of rows in Z is the number of modes, and the number of columns in Z is the number of loads.

$S^{a,b}(f)$ is the (a, b) entry of the Hermitian cross-correlation matrix between loads. Letting Z_i denote row i of Z,

$$\Gamma_{ij} = q_i^*(Z_i S(f) Z_j^T) q_j \delta f,$$

[3]see for example, reference.[115]

or

$$\Gamma = \text{diag}(q^*) Z S(f) Z^T \text{diag}(q) \delta f$$

For each mode shape,ϕ, each element, there is a displacement with a corresponding element stress, ψ. The (i, j) pair of modes contributes $\psi_i^T A \psi_j \Gamma_{ij}$ to the von Mises stress. The velocity and acceleration contributes similar terms to the 2^{nd} and 4^{th} moments of von Mises stress, respectively.

2.3.2. Power Spectral Density

The displacement power spectral output may also be written as follows,

$$G_{mn}(f) \;=\; \sum_{i,j} \sum_{a,a'} q_i^*(f) q_j(f) \phi_{im} \phi_{jn} Z_a^i S^{a,a'}(f) Z_{a'}^j \tag{2.3.1}$$

Note that there is no δf coefficient here.

If the output displacement degrees of freedom are restricted to a single node, the subscripts m and n are applicable to the 3 degrees of freedom at a single location. Because the response directions may not be independent, the matrix may not be diagonal.

By summing over the loads we may reduce the power spectral expression to a sum on modal contributions.

$$G_{mn}(f) \;=\; \sum_{i,j} \phi_{im} \phi_{jn} \mathcal{G}_{ij}(f) \tag{2.3.2}$$

where

$$\mathcal{G}_{ij}(f) = q_i^*(f) q_j(f) \sum_{a,a'} Z_a^i Z_{a'}^j S^{a,a'}(f) \tag{2.3.3}$$

Note that, except for the Z_a^i (which only needs to be computed once), all the terms in equation 2.3.3 are known on each subdomain.

At each frequency, f, there is a 3 by 3 complex Hermitian output displacement spectral density matrix G and an output acceleration spectral density matrix, $G\omega^4$.

2.3.3. Tensor Transformations of PSD

The output PSD is a Hermitian tensor, $A^T = A^*$. The output PSD is defined as the correlation of the acceleration, i.e.

$$A_{PSD}(\omega) = a(\omega) a(\omega)^\dagger, \tag{2.3.4}$$

where $a(\omega)$ is the complex acceleration vector. On a single node, A is a 3 x 3 complex tensor. The tensor rotation can be derived from the rotation of the vectors. Let $\bar{a} = Ra$ be the acceleration expressed in a new coordinate frame and computed from the acceleration in the basic frame multiplied by an orthogonal transformation matrix R. Because $R^{-1} = R^T$, we have $a = R^T \bar{a}$. See section 1.4 for a discussion of coordinate systems and vector transformations.

$$\begin{aligned} A_{PSD} &= a a^\dagger && (2.3.5) \\ &= R^T \bar{a} (R^T \bar{a})^\dagger && (2.3.6) \\ &= R^T \bar{a} \bar{a}^\dagger R && (2.3.7) \\ &= R^T \bar{A}_{PSD} R && (2.3.8) \end{aligned}$$

Therefore, $\bar{A}_{PSD} = R\, A_{PSD} R^T$.

2.3.4. RMS Output

The RMS output for degree of freedom m is given by,

$$\begin{aligned} X_{rms} &= \sqrt{\int G_{mm}(f)df} \\ &= \sqrt{\int \sum_{i,j} \phi_{im}\phi_{jm}\mathcal{G}_{ij}(f)df} \\ &= \sqrt{\sum_{i,j} \phi_{im}\phi_{jm}\Gamma_{ij}} \end{aligned} \quad (2.3.9)$$

where $\Gamma_{ij} = \int \mathcal{G}_{ij}(f)df$.

2.3.4.1. Truncation.

Note that equation 2.3.9 involves a summation over modes weighted by Γ_{ij}. This summation is an order N^2 operation which can retard performance if there are many modes. Often many of the terms in Γ are small. Rows and columns of the sum may be eliminated with no impact on the overall solution of X_{rms}.[4]

2.3.4.2. Parallelization.

The parallel result can be arrived at by computing Z_a^i on each subdomain, and then summing the contributions of each subdomain. Note that Z_a^i contains the spatial contribution of the input force. At boundaries that interface force must be properly normalized like an applied force is normalized for statics or transient dynamics by dividing by the cardinality of the node. Once Z has been summed, Γ_{ij} may be computed redundantly on each subdomain. The only communication required is the sum on Z (a matrix dimensioned at the number of loads by the number of modes).

The acceleration power spectral density is $G_{mm}(\omega)\omega^4$. Subsection 7.2.5 provides details about transforming power spectra to an output coordinate system.

2.3.4.3. Displacement Interference (Relative_Disp)

A common requirement is understanding the probability of interference of two nodes. The *difference displacement spectrum* of a degree of freedom on two different points is a similar expression.

$$X_{KL}^2(f) = (X_K(f) - X_L(f))(X_K(f) - X_L(f))^* \quad (2.3.10)$$

$$= X_K(f)X_K^*(f) + X_L(f)X_L^*(f) - X_K(f)X_L^*(f) - X_L(f)X_K^*(f) \quad (2.3.11)$$

$$= G_{KK}(f) + G_{LL}(f) - G_{KL}(f) - G_{LK}(f) \quad (2.3.12)$$

[4] A similar truncation can be performed if the quantity of interest is acceleration rather than displacement. In that case, truncation may be performed on $\Gamma_{ij}\omega_i^2\omega_j^2$.

Likewise, the RMS value may be computed.

$$(X_{KL})^{rms} = \sqrt{\int X_{KL}^2 df} \tag{2.3.13}$$

$$= \sqrt{\sum_{i,j} \left(\phi_{iK}\phi_{jK} + \phi_{iL}\phi_{jL} - \phi_{iK}\phi_{jL} - \phi_{iL}\phi_{jK}\right)\Gamma_{ij}} \tag{2.3.14}$$

As with the displacement spectrum, when the different coordinate directions are not independent, off diagonal contributions can be important. This development must be extended to all the dependent degrees of freedom.

This information can be computed between two points using the output keyword Relative_Disp and a Joint2G element.

2.3.5. RMS Stress

A description of the algorithm for computation of the von Mises RMS stress is included in the reference at the beginning of this chapter. Two methods are available, but both use the integrated modal contribution Γ_{ij} as the basis for their computation. The more complete method relies on a singular value decomposition. Portions of that method are touched on below

2.3.6. Matrix properties for RMS stress

Since $S(f)$ is Hermitian, it follows that Γ_{qq} is also necessarily Hermitian. It will not in general be real. The complex valued singular value decomposition (SVD) is computed using the LAPACK `zgesvd` routine. The results from the SVD of an Hermitian matrix are real eigenvalues (stored in X), and complex vectors, stored in Q. The LAPACK routines for Hermitian eigenvalue problems (`zhetrd`,`zsteqr`) would be more efficient.

At the element level another SVD is computed. In this case we are computing the singular values of the matrix C.

$$C = XQ^{\dagger}BQX$$

where,

$$B = \Psi^T A \Psi$$

B is symmetric. It can be shown that $Q^{\dagger}BQ$ is Hermitian. If we examine a single element of C we can see that it contains the sum over all the terms in an Hermitian matrix. That sum is necessarily real, since it can be computed by adding the lower half with its transpose and then summing the diagonal. Let,

$$A_{ij} = \sum_{m,n} Q^*_{mi} B_{mn} Q_{nj} = \sum_{m,n} a_{ij}$$

But,

$$A^*_{ji} = \sum_{m,n} Qm, j * B_{mn} Q^*_{ni} = \sum_{m,n} Q_{nj} B_{mn} Q^*_{mi} = \sum_{m,n} a^*_{ij}$$

We therefore only need use the real `svd` routines to compute the results at each output location.

The `svd` calculations provide the information needed to truncate or reduce the model. As the size of the model grows, the number of modes required for an analysis tends also to grow. However, the computational time for computing the `svd` is proportional to matrix dimension cubed. On the other hand, the `svd(Γ)` is only computed once. However, the computation of each decomposition of C occurs at each output location and can significantly affect performance. In the model problem where the dimension of C was allowed to remain the same as the number of modes, increasing the number of modes from 20 to 100 changed the time for the analysis by factor of more than 100 (close to the predicted 5^3). Unfortunately the desired models may have many hundreds of modes.

The `svd(Γ)` provides important information about the number of independent processes. Note that C includes the `svd` values from this calculation. We truncate by computing all the `nmodes x nmodes` terms in B, but only retaining `Cdim` columns of Q, where `Cdim` is chosen so the values of X are not too small. Thus, $X[\texttt{Cdim}]/X[0] > 10^{-14}$. This restricts the dimension of C to a small number, while retaining all components that contribute significantly to its value. As a result, the entire calculation appears to scale approximately linearly with the number of modes.

2.4. Modal Frequency Response Methods

The **Sierra/SD** implementation of the modal acceleration method is described in this section. Separate cases are considered when the structure does and does not have rigid body modes.

2.4.1. No Rigid Body Modes

We first consider the frequency domain version of the equations of motion.

$$(-\omega^2 M + j\omega C + K)\hat{u} = \hat{f} \tag{2.4.1}$$

Consider the modal approximation

$$\hat{u} \approx \sum_{i=1}^{N} \phi_i q_i \tag{2.4.2}$$

where N is the number of retained modes, ϕ_i is the i^{th} mode shape, and q_i is the i^{th} modal dof. For modal damping, one obtains the uncoupled equations

$$(-\omega^2 m_i + j\omega c_i + k_i)q_i = \phi_i^T \hat{f} \tag{2.4.3}$$

for $i = 1, \ldots, N$ where

$$\begin{aligned} m_i &= \phi_i^T M \phi_i && (2.4.4)\\ c_i &= \phi_i^T C \phi_i && (2.4.5)\\ k_i &= \phi_i^T K \phi_i && (2.4.6)\\ & && (2.4.7) \end{aligned}$$

are the modal mass, modal damping, and modal stiffness of the i^{th} mode. Solving equation 2.4.3 for q_i leads to

$$q_i = (\phi_i^T \hat{f})/(-\omega^2 m_i + j\omega c_i + k_i) \tag{2.4.8}$$

Replacing $(-\omega^2 M + j\omega C)\hat{u}$ in equation 2.4.1 with the modal approximation

$$(-\omega^2 M + j\omega C) \sum_{i=1}^{N} \phi_i q_i \tag{2.4.9}$$

leads to

$$K\hat{u} = \hat{f} + (\omega^2 M - j\omega C) \sum_{i=1}^{N} \phi_i q_i \tag{2.4.10}$$

Recall that the mode shapes satisfy the eigenvalue problem

$$K\phi_i = \omega_i^2 M \phi_i \tag{2.4.11}$$

where ω_i is the circular frequency of the i^{th} mode. Provided $\omega_i \neq 0$, one obtains

$$K^{-1} M \phi_i = \phi_i / \omega_i^2 \tag{2.4.12}$$

In addition, see Eq. (18.14) of Craig, the damping matrix C can be expressed as

$$C = \sum_{i=1}^{N} \left(\frac{2\zeta_i \omega_i}{m_i} \right) (M\phi_i)(M\phi_i)^T \tag{2.4.13}$$

where ζ_i is the damping ratio of the i^{th} mode. Substituting equations 2.4.12 and 2.4.13 into equation 2.4.10 and solving for $\hat{u}$ leads to

$$\hat{u} = K^{-1}\hat{f} + \sum_{i=1}^{N} (\omega^2/\omega_i^2 - 2\zeta_i j\omega/\omega_i)\phi_i q_i \tag{2.4.14}$$

The acceleration frequency response, $\hat{a}$, can be obtained by multiplying equation 2.4.14 by $-\omega^2$.

2.4.2. *Rigid Body Modes*

The procedure outlined here describes how the modal acceleration method can be used in the case when the structure has rigid body modes. The main difference between the approach presented here and Craig's method[39] (pp. 368-371) is in the way that the flexible response is computed using the singular stiffness matrix. Craig removes the rigid body modes from the stiffness matrix using constraints. In our approach, we first orthogonalize the right-hand side with respect to the rigid body modes, and then use an iterative solver to solve the singular system directly. Although the two methods are equivalent the latter is much more convenient from the implementation point of view. Note, however, that the implementation is likely to fail on a single processor since the direct solvers in **Sierra/SD** are unable to manage a singular stiffness matrix.

The equations of interest are the frequency domain equations of motion

$$-\omega^2 M u + j\omega C u + K u = f \tag{2.4.15}$$

Since the stiffness matrix may be singular, we first split the solution into a rigid body part and a flexible part.

$$
\begin{aligned}
u(\omega) &= u_R(\omega) + u_E(\omega) && (2.4.16)\\
&= \Phi_R q_R(\omega) + \Phi_E q_E(\omega) && (2.4.17)
\end{aligned}
$$

where the subscript R refers to rigid body mode contributions, and E refers to contributions from flexible modes. We define N as the total number of degrees of freedom, N_R as the number of rigid body modes and N_E the number of flexible modes, where $N = N_R + N_E$. Then, Φ_R is an $N \times N_R$ matrix of rigid body eigenvectors, Φ_E is an $N \times N_E$ matrix of flexible eigenvectors, q_R is a vector of dimension N_R, and q_E is a vector of dimension N_E. We assume mass normalized eigenvectors.

We substitute equation 2.4.17 into equation 2.4.15, and pre-multiply both sides by Φ_R^T and Φ_E^T. This yields two sets of equations, after using orthogonality and the fact that $K\Phi_R = 0$.

$$-\omega^2 q_R + j\omega C_R q_R = \Phi_R^T f \qquad (2.4.18)$$

$$-\omega^2 q_E + j\omega C_E q_E + K_E q_E = \Phi_E^T f \qquad (2.4.19)$$

where C_R, C_E are diagonal matrices containing the modal damping contributions, and K_E is a diagonal matrix containing the eigenvalues. In particular, the i^{th} diagonal entry of C_E is $2\omega_i\zeta_{E_i}$, and the i^{th} diagonal entry of C_R is $2\omega_i\zeta_{R_i}$. For most applications, C_R is null. Solving these equations we obtain the component-wise values of the coefficients

$$q_{R_i} = \frac{\Phi_{R_i}^T f}{-\omega^2 + j\omega C_{R_i}} \qquad (2.4.20)$$

$$q_{E_i} = \frac{\Phi_{E_i}^T f}{-\omega^2 + j\omega C_{E_i} + \omega_{E_i}^2} \qquad (2.4.21)$$

Equation 2.4.19 can be solved for q_E, and substituting this into equation 2.4.17, we obtain

$$u = \Phi_R q_R + \Phi_E K_E^{-1} \Phi_E^T f + \omega^2 \Phi_E K_E^{-1} q_E - j\omega \Phi_E K_E^{-1} C_E q_E \qquad (2.4.22)$$

The first term in equation 2.4.22 is known. The third and fourth terms of equation 2.4.22 can be computed by modal truncation, and in fact these are the same as the second and third terms of equation 2.4.14. The second term in equation 2.4.22 is the static correction, and is not readily computable in the present form since all flexible modes would have to be known to compute it.

To compute the second term in equation 2.4.22, we note that the matrix $a_E = \Phi_E K_E^{-1} \Phi_E^T$ is the inverse of the elastic stiffness matrix, that is, the stiffness matrix without the rigid body components. Craig gives a procedure of constraining the rigid body modes in the stiffness matrix to compute the product $a_E f$. This procedure would require re-sizing the global stiffness matrix midway through the modalfrf solution procedure, and this is tedious from the code development standpoint.

A more convenient approach is to use GDSW to solve the system $Ku = f_E$, where f_E is obtained by orthogonalizing the right-hand side f with respect to the rigid body modes, via Gram Schmidt. If K is singular and f is orthogonal to the rigid body modes, then GDSW can be applied to $Ku = f$

Though in theory u is already orthogonal to the rigid body modes after the GDSW solve, numerical round-off may result in a small loss of orthogonality (especially if the solver tolerance is large). The resulting solution we denote by u_E. Then,

$$u_E = \Phi_E K_E^{-1} \Phi_E^T f \qquad (2.4.23)$$

and thus all terms in equation 2.4.22 are known. Thus the modal frequency response can be computed using equation 2.4.22.

We note that the orthogonalizations referred to above involve only the standard dot products. That is, to make f orthogonal to one rigid body mode ϕ_i, the Gram Schmidt factor is

$$\alpha = \frac{\phi_i^T f}{\phi_i^T \phi_i} \tag{2.4.24}$$

and then

$$f_E = f - \alpha\phi \tag{2.4.25}$$

These dot products do not involve the mass matrix. They are the standard dot products.

2.4.3. Example

Finally, we present an example of the performance of this method as compared to the standard modal displacement method. The example is a beam composed of 320 hex8 elements. The beam is free-free, so that all rigid body modes are present. The frequency response is computed up to 9000 Hz, and 15 modes are used in the modal expansions. The 15th mode had a frequency of 11362 Hz. In Figure 2-1, the two methods are compared with the direct frequency response approach. It is seen that the modal acceleration method gives a significantly improved performance over the modal displacement method.

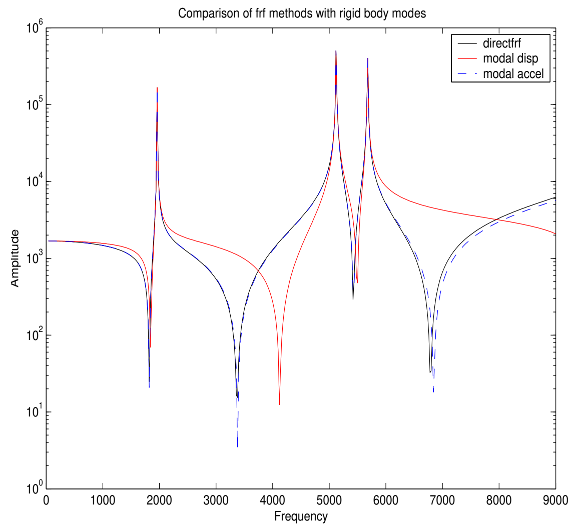


Figure 2-1. – A comparison of the modal displacement, modal acceleration, and direct frequency response approaches. The modal acceleration method gives a better approximation to the direct approach than the modal displacement method.

2.5. Fast Modal Solutions

Because modal based solutions such as modal transient do not require a linear solve, they can hasten the solution of linear problems. However, in the standard approach, these solutions may not show the performance that could be achieved. This is because the standard approach manipulates a lot of data when the model size is large, see Figure 2-3. We here address a method for much higher performance provided that output is required on a modest data set and that the force is simple.

1. Compute all eigenvectors, $(K - \lambda M)\Phi = 0$.
2. Compute the applied load (in modal coordinates) at each time,
$$f^i = \sum_k \Phi_{ki} F_k^{ext}.$$
3. Compute the modal system response from equation 2.5.4.
4. Expand from modal to *full* physical space.
$$X_{n+1}^k = \sum_{i<\text{Nmodes}} q_{n+1}^i \Phi_{ki}.$$
5. Collapse the physical space to the output degrees of freedom.
$$\tilde{x} = \text{subset}(X)$$

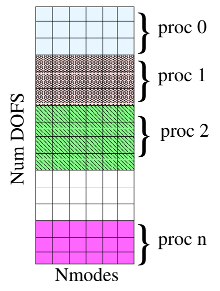


Figure 2-2. – The parallel data (matrices and vectors Φ and X) are partitioned by processor.

Figure 2-3. – Standard Modal Transient Algorithm. Note that while the output is required on only a small part of the model, a calculation of data on all degrees of freedom is performed first, and results are then collapsed to the reduced model.

2.5.1. *Modal Solution Summary*

Using the trapezoidal rule, Newmark-Beta integrator[5] equation 2.1.6 may be condensed to,

$$\left[\frac{4}{\Delta t^2}M + \frac{2}{\Delta t}\hat{C} + K\right] d_{n+1} = F_{n+1}^{ext} + \hat{C}\left[v_n + \frac{2}{\Delta t}d_n\right] + M\left[\frac{4}{\Delta t^2}d_n + \frac{4}{\Delta t}v_n + a_n\right] \tag{2.5.1}$$

Also,

$$v_{n+1} = -v_n + \frac{2}{\Delta t}(d_{n+1} - d_n) \tag{2.5.2}$$
$$a_{n+1} = -a_n + \frac{4}{\Delta t^2}(d_{n+1} - d_n) - \frac{4}{\Delta t}v_n \tag{2.5.3}$$

With the usual modal transformation, $d_k = \sum_i \Phi_{ki} q$, $\lambda_i = \Phi_i^T K \Phi_i$, and $\Phi^T M \Phi = I$, we may write the equivalent modal equations.

$$a_i q_{n+1}^i = q_n^i + f_{n+1}^i + \tilde{f}^i \tag{2.5.4}$$

[5]This implies that $\alpha_m = \alpha_f = 0$, $\beta_n = 1/4$, and $\gamma_n = 1/2$.

where

$$\begin{aligned} a_i &= \frac{4}{\Delta t^2} + \frac{2}{\Delta t}\gamma_i + \lambda_i \\ f_{n+1}^i &= \sum_k \Phi_{ki} F_k^{ext} \\ \tilde{f}^i &= \ddot{q}_n + \left(\frac{4}{\Delta t}\dot{q}_n + \frac{4}{\Delta t^2} q_n \right) + \gamma_i \left(\dot{q}_n + \frac{2}{\Delta t} q_n \right) \end{aligned}$$

and,

γ_i is the modal damping

These are uncoupled equations. The solution for each modal coordinate is independent of any other.

2.5.2. *Parallel Fast Modal*

Typically the objective is to measure the response in a small region, such as data output to a history file. Large amounts of data are processed, only to reduce the data at each time step to a reduced system. The parallel computer processing is being expended to process large vectors that are not needed, and for which no useful output is provided. If the reduced set may easily fit on a single processor, and if the modal force may be adequately determined, then a streamlined algorithm may be used.

The fast algorithm is illustrated in Figure 2-4 for transient dynamics, and in Figure 2-5 for modal frequency response. The same set of equations are now solved, but since the entire physical model exists on all processors, we can compute the sum of terms in parallel.

1. Begin with eigenvalues, λ, and *reduced* eigenvectors, ϕ. We also need the generalized components of modal force, $\zeta_i^s(\omega) = \sum_k \Phi_{ki} \hat{F}_k^s(\omega)$.
2. Compute the time response of the modal system response in parallel. Each processor gets only a subset of modes, and solves equation 2.5.4 independently.
3. Compute the response on the physical space using the sum of modes as a sum across processors. NOTE: this is restricted to the reduced physical space.

$$\tilde{x}_k = \sum_p^{N_{proc}} \sum_i^{Nmodes_{proc}} \phi_{ki} q_i$$

Figure 2-4. – Fast Modal Transient Algorithm.

1. Begin with eigenvalues, λ, and *reduced* eigenvectors, ϕ. We also need the generalized components of modal force, $\zeta_i^s(\omega) = \sum_k \Phi_{ki}\hat{F}_k^s(\omega)$.
2. Compute the frequency response of the modal system response in parallel. Each processor gets only a subset of modes, and solves the following equation independently.

   $$q_i(\omega) = \frac{f_i^q(\omega)}{\omega^2 - \omega_i^2 - 2j\gamma_i\omega\omega_i}$$

   where $\omega = \sqrt{\lambda_i}$ and $j = \sqrt{-1}$.
3. Compute the response on the physical space using the sum of modes as a sum across processors. NOTE: this is restricted to the reduced physical space.

   $$\tilde{x}_k = \sum_p^{N_{proc}} \sum_i^{Nmodes_{proc}} \phi_{ki} q_i$$

   Alternatively, each processor may be assigned the computation of a frequency range, and compute all the modal contributions to that range. A processor sum would gather all the results for output.

Figure 2-5. – Fast Modal Frequency Response Algorithm.

2.5.3. Determination of Modal Force

The fast algorithm outlined in the previous section depends on determination of the modal force vector, $f^i(t)$. But, the physical loads may be applied to degrees of freedom other than those in the limited output set, so that the eigenvector, Φ of the full system would be required.

However, in most cases,[6] the force in the physical coordinates is computed as a sum of spatial and temporal terms.[7]

$$F^{ext}(x,t) = \sum_s^{Nsets} \hat{F}^s(x)\delta^s(t)$$

Typically, each spatial function $\hat{F}^s$ is determined by a nodeset, sideset or body load input, while the temporal term, $\delta^s(t)$, is a multiplier defined in a `FUNCTION` section. We may thus write,

$$\begin{aligned} f^i(t) &= \sum_k \Phi_{ki} F^{ext}(x_k,t) && (2.5.5) \\ &= \sum_k \Phi_{ki} \sum_s^{Nsets} \hat{F}_s(x)\delta^s(t) \\ &= \sum_s^{Nsets} \zeta_s^i \delta^s(t) && (2.5.6) \end{aligned}$$

[6]If user defined functions of space are included, this situation is violated, and the fast algorithm cannot be used.

[7]What is described here for the time domain also applies in the frequency domain. They are products of spatial and frequency components.

where,

$$\zeta_s^i = \sum_k \Phi_{ki} \hat{F}_k^s \tag{2.5.7}$$

Thus, a necessary part of the preparation for a fast modal solution includes calculation of the generalized components of force, ζ_s^i.

2.6. Eigenvalue Problems

The eigen solution method computes a user-specified number of the lowest-frequency modes of

$$(K - \omega^2 M)\phi = 0. \tag{2.6.1}$$

The eigenvalue (or mode) ω^2 and eigenvector (or mode shape) ϕ correspond to the solution $u(t) = \phi e^{i\omega t}$ with frequency $\omega/(2\pi)$. The frequency and the mode shape are reported to the user. The mode shapes are mass orthogonal, i.e., $\phi_i^T M \phi_j = \delta_{ij}$. The default diagnostic output, including the residual norms $\|(K - \omega^2 M)\phi\|$, are labeled by eigenvalue ω^2.

Some approaches can be used to solve this system, and their relative merits are understood (see[8]). For large systems, direct (or dense) methods such as the QR algorithm or Jacobi transformations are tremendously more expensive than the methods used in **Sierra/SD**. In **Sierra/SD**, we rely on the shifted and inverted Lanczos algorithm as implemented in ARPACK[92] . A detailed scalability study is available in SAND 2019-1217.[27]

Different solution methods are available for many of the different eigenvalue problems. Note that Rayleigh damping, $C = \alpha M + \beta K$, does not change the mode shapes and changes the mode frequencies as in a single-degree-of-freedom problem.

The shift (σ) and invert transform leads to a problem whose largest modes are the modes of interest. The result of subtracting $\sigma M \phi$ from both sides of equation (2.6.1) is

$$(K - \sigma M)\phi = M\phi(\omega^2 - \sigma). \tag{2.6.2}$$

The eigenvalue problem exposed to ARPACK emerges by multiplying both sides of (2.6.2) by $(K - \sigma M)^{-1}(\omega^2 - \sigma)^{-1}$:

$$(K - \sigma M)^{-1} M \phi = (\omega^2 - \sigma)^{-1}\phi. \tag{2.6.3}$$

For example, users are expected to understand that the shift corresponding to the frequency f is $4\pi^2 f^2$.

The linear solvers available with the eigen solution case all require positive-definite systems. For this reason, the shift must be negative. Generally speaking, increasing the magnitude of the shift makes solving the linear systems easier and solving the eigenvalue problem harder. In theory, using the Helmholtz linear solver, the capability could be implemented to determine the modes nearest to an arbitrary positive user-specified shift. The demand for this capability has never justified the risk and expense of implementation.

Structural dynamics eigenvalue problems have some unique features all revolving around the challenging nature of the corresponding linear systems. Results are typically insensitive to the linear solver relative residual norm threshold (the default is 10^{-6}). One exception is the case of computing many (thousands) of modes, in which case it is necessary to start out with a smaller tolerance (say 10^{-12}) to avoid convergence problems at the higher frequencies. P

2.7. Modal Analysis of Linearly Damped Structures

Modal solvers are provided for all common types of linearly damped structures. The eigenvalue and eigenvectors are complex valued. The algorithms are designed for internally damped structures such as linear viscoelastic materials. In general users specify the solution method for the eigenvalue problem.

One of the packages is called `Ceigen`. The parameters of `Ceigen` to be aware of are `eig_tol`, `nmodes`, and `viscofreq`. The first two parameters, `eig_tol` and `nmodes` will be familiar to **Sierra/SD** users that solve eigenvalue problem for undamped structures. `eig_tol` is the convergence tolerance for the eigenvalues, and `nmodes` is the number of requested eigenvalues. `viscofreq` approximates the first flexible mode of the structure. The default value for `eig_tol` is $1.e-8$.

The eigenvalue problem for linearly damped structures is an instance of a quadratic eigenvalue problem.

$$\left[K + \lambda D + \lambda^2 M\right] \phi = 0 \tag{2.7.1}$$

where,

$$\begin{aligned} K &= \text{the stiffness matrix} \\ D &= \text{the damping matrix} \\ M &= \text{the mass matrix} \\ \lambda &= \text{the complex frequency.} \end{aligned}$$

The matrices are independent of frequency. The first adjustment to make is that the eigenvalues $\lambda = i\omega + \gamma$ correspond to the eigenvalue ω^2 of the undamped, $D = 0$, problem. There are other solvers. The Anasazi solver is similar to CEigen. From the point of view of a user, the two methods are very similar.

Solvers similar to the algorithms used in Abaqus are also supported. The Projection and Superposition solvers resemble the Abaqus solvers. Also the S A solver is available for structural acoustic problems.

CEigen Input File Specification. The **Sierra/SD** input file specification is similar to the specification for transient simulations. To change a working **Sierra/SD** input file for a transient problem into a **Sierra/SD** input file for `Ceigen`, change the Solution and Parameters blocks. The example below illustrates how the Solution and Parameter blocks are modified for modal analyses.

```
SOLUTION
case ceigen
ceigen nmodes 20
viscofreq=1.e+4
END
PARAMETERS
eig_tol 1.E-5
wtmass=0.00259
END
```

The parameter wtmass is an example of a parameter that was needed for the transient simulation, and is still needed for modal analyses.

2.7.1. Output File Format

The output is similar to the output for the undamped eigenvalue problem. The results file contains any requested data. Supplemental information is written to the screen that is useful for algorithm development.

The Results file `foo.rslt` tabulates the values $\lambda/(2\pi)$ for (λ_i) that solve equation (2.7.2). Pure real eigenvalues are not written to the Results file.[8] If λ_i has been found with i in the range, $1 \leq i \leq 24, 27 \leq i \leq 34$, then the missing eigenvalues $(\lambda_i)_{25 \leq i \leq 26}$ are real eigenvalues that are omitted. The number of eigenvalues written in the Results file is `nmodes` at most.

As is the case with the undamped eigenvalue problem, **Sierra/SD** will print a table to the screen. The table is titled "Ritz values (Real, Imag) and direct residuals", and has four columns of real numbers. The number of eigenvalues that are computed may be larger or smaller than the number requested. Some real eigenvalues may appear among the converged eigenvalues. The table will contain any converged real eigenvalues (zero in column two). Columns three and four are two different residual norms for each eigenvalue. Eigenvalues with large residual norms are not converged. The residual norm in the third column is less sensitive to the linear system relative residual norm bound than the residual norm in the fourth column is After each implicit restart, all the approximate eigenvalues are printed to the screen.

2.7.2. Some Back Ground

The eigenvalue problem for an undamped structure

$$\mathbf{K}\Phi = \mathbf{M}\Phi\Omega^2, \quad \Phi^T\mathbf{M}\Phi = I,$$

$\Omega = \oplus_i \omega_i$, has been discussed elsewhere in this document. **Sierra/SD** returns the frequencies $\omega/(2\pi)$. `Ceigen` solves a similar problem. `Ceigen` solves the quadratic eigenvalue problem

$$[\mathbf{M}\lambda^2 + \mathbf{D}\lambda + \mathbf{K}]u = 0, \quad u^T u = 1. \tag{2.7.2}$$

In the undamped case, $\mathbf{D} = \mathbf{0}$, $\lambda = i\omega$.

A second order linear differential equation is the same as a first order system. Similarly a quadratic eigenvalue problem is the same as a matrix eigenvalue problem of twice the size.

Linear problems such as matrix eigenvalue problems are solvable in that it is possible to find all solutions. For matrix eigenvalue problems the key idea is deflation. One big subspace is used to compute all the eigenvalues. Small eigenvalues tend to be computed early and are deflated from the problem. The reward for deflation is that the gravest remaining eigenvalues are much more likely to be computed next. For general nonlinear eigenvalue problems on the other hand, no robust algorithms are known to the author.

Viscoelasticity. The eigenvalue problem for viscoelastic problems[41] in the most simple case (one term Prony series) has the form

$$[\mathbf{M}s^2 + \mathbf{D}(s)s + \mathbf{K}]u = 0. \tag{2.7.3}$$

$$\mathbf{K} = \mathbf{B}E_\infty, \ \mathbf{D}(s)s = \mathbf{B}(E_g - E_\infty)f(s),$$

$$f(s) = s/(s+a) = 1 - (s/a+1)^{-1}.$$

[8] Real modes correspond to an over-damped mode with no oscillatory component. These are not physical as discussed below.

Prony series damping in the time domain creates a frequency domain problem with real eigenvalues that are not physical.[41] Some care is needed to avoid the real eigenvalues in computations.

Here is a sketch of justification that the Prony series problem has real eigenvalues. The eigenvalue problem has a closed form solution in terms of the eigenvalues of the undamped problem. The one term Prony series damping increases the degree of the characteristic equation from two to three, and the third root must be real.

Viscofreq. The eigenvalue problem in equation (2.7.3) is not a quadratic eigenvalue problem $(\mathbf{M}, \mathbf{D}, \mathbf{K})$. The obvious approximation is to evaluate $\mathbf{D}(s)$ at some fixed s_o near to the wanted eigenvalues. The user parameter `viscofreq`= ω is a real number such that $s_o = i\omega$. In a later release $s_o = r + i\omega$ for some internally computed value r.

Using a value of `viscofreq` that is much too small may degrade performance. As `viscofreq` increases, the eigenvalues do change, and **Sierra/SD** converges more quickly. The cluster of real eigenvalues moves left, away from zero, and it becomes possible to compute more of the complex eigenvalues. Over-estimates of `viscofreq` are safer than underestimates.

Suppose that $s_o = r + i\omega$. A different quadratic eigenvalue problem is used.[41] Both $\mathbf{D}$ and $\mathbf{K}$ are modified. The approximation is more accurate for problems in which r is much more accurate than ω. Also, $(\mathbf{M}, \mathbf{D}, \mathbf{K})$ are all real matrices. The eigenvalues and eigenvectors come in complex conjugate pairs.

Important to be aware that no constant damping matrix inherits the property of $\mathbf{D}(s)$ that

$$\lim_{s\to\infty} \mathbf{D}(s) = 0.$$

Physically, this means that the eigenvalues in equation (2.7.2) that are far from `viscofreq` are over-damped. If for a given mode shape, s_o is closer to the real eigenvalue of equation (2.7.3) than either complex conjugate pair, then `Ceigen` may return the real eigenvalue. For example equation (2.7.3) has many real eigenvalues clustered left of $-a$.

2.7.3. *Trust Regions and Real Modes*

The eigenvalue problem is solved using ARPACK. The convergence criteria in the ARPACK package use a trust region. CEigen will compute the right-most eigenvalues of the eigenvalue problem in equation (2.7.2). If the k-th mode does not satisfy the convergence tolerance, and $k \le$`nmodes`, then ARPACK is not converged, no matter how many other eigenvalues are converged.

The authors have gone to great lengths to filter out real eigenvalues. Nonetheless in problems with a cluster of real eigenvalues among the right-most eigenvalues, it is difficult to compute eigenvalues high into the frequency range. If such a problem arises, increase `eig_tol` (multiply by ten), increase `nmodes` (add ten), and most importantly increase `viscofreq` (double).

2.7.4. *ViscoFreq - Approximating the Response of Viscoelastics*

The viscoelastic mass matrix can be considered to be independent of frequency. However, the damping and stiffness matrices can be functions of frequency, depending on the formulation. There are two possible formulations. The first one results in a complex, frequency dependent damping matrix, and a real-valued, frequency independent stiffness matrix. The second results in a frequency- dependent, real-valued damping matrix and a frequency-dependent, real valued stiffness matrix. We chose the second formulation to avoid having to deal with a complex-valued damping matrix. The two formulations are the same up to the order of the linearization error.

Consider the simplest possible viscoelastic material, characterized by a single term of the Prony series. The equation of motion for a 1D system with this material is given below. The full 3D case is similar, except that it has separate terms for the bulk and shear components.

$$\left[K_\infty + sD(s) - s^2 M\right] u = f(s) \tag{2.7.4}$$

Here, s is the Laplace transform frequency, $f(s)$ is the frequency dependent force, and the damping matrix is now a function of frequency.

$$D(s) = (E_G - E_\infty)\frac{1}{s + 1/\tau}B \tag{2.7.5}$$

with E_∞, the Young's modulus for high frequencies, E_G the modulus for low (or glassy) frequencies, τ is the Prony series relaxation time, and $K_\infty = E_\infty B$ is the stiffness at high frequencies.

Equation 2.7.4 has two linearizations, since for the quadratic eigenvalue problem, we may only solve equations of the form in equation 2.7.1, i.e. quadratic in λ or s.

2.7.4.1. User Specified frequency of linearization

We define viscofreq, ω and $s_\omega = r + i\omega$, which is the complex number about which the linearization takes place. In the current methodology, r is zero.

First, we split $D(s_\omega)$ into its real and imaginary components by multiplying by $\frac{(r+1)-i\omega\tau}{(r+1)-i\omega\tau}$.

$$\begin{aligned} D(s) &= (E_G - E_\infty)\frac{1}{s + 1/\tau}B && (2.7.6)\\ &= (E_G - E_\infty)\frac{\tau}{i\omega\tau + (r\tau + 1)}B && (2.7.7)\\ &= \frac{\tau((r\tau+1) - i\omega\tau)}{(r\tau+1)^2 + \omega^2\tau^2}(E_G - E_\infty)B && (2.7.8)\end{aligned}$$

Then we also temporarily replace the s in front of $sD(s)$ with s_ω. This gives,

$$\begin{aligned} sD(s) &= (i\omega + r)D(i\omega + r) && (2.7.9)\\ &= \frac{\tau(i\omega + r) + \omega^2\tau^2 + r^2\tau^2}{(r+1)^2 + \omega^2\tau^2}(E_G - E_\infty)B && (2.7.10)\end{aligned}$$

Finally, we replace $i\omega + r$ with s to go to the quadratic eigenvalue problem. This results in a contribution to the stiffness matrix, and a real damping matrix.

$$\left[\left(E_\infty + (E_G - E_\infty)\frac{\omega^2\tau^2 + r^2\tau^2}{(r+1)^2 + \omega^2\tau^2}\right)B + s\left(\frac{\tau}{(r+1)^2 + \omega^2\tau^2}\right)(E_G - E_\infty)B + s^2 M\right]\phi = 0 \tag{2.7.11}$$

Thus, we see that the damping matrix is real, but the stiffness matrix gets an additional (positive) real contribution.

Practically of course, the systems are far more complex. Typically, there is more than one material, and that material has some Prony terms. Equation 2.7.11 is modified, but the overall effect is the same, i.e. the stiffness matrix is increased by a viscoelastic term, and the damping term is also modified. Effectively we have the following.

$$\tilde{K}(r+i\omega) = \sum_{elem} \tilde{K}_{elem}(r+i\omega) \tag{2.7.12}$$

where $\tilde{K}_{elem}$ is the modified stiffness matrix.

$$\tilde{K}_{elem}(r+i\omega) = K_{elem} + \text{imag}(D_{elem}(r+i\omega))$$

Likewise,

$$\tilde{D}_{elem}(r+i\omega) = \text{real}(D(r+i\omega)) \tag{2.7.13}$$

The *linearized* eigenvalue problem determines λ,

$$\left[\tilde{K}(r+i\omega) + i\lambda\tilde{D}(r+i\omega) - \lambda^2 M\right]\phi = 0. \tag{2.7.14}$$

2.7.4.2. A Simple Error Estimate

The accuracy of the eigenvalues of equation 2.7.11 as eigenvalues of equation 2.7.4 may be estimated.

First, we define the distance from a given computed eigenvalue, s_c, to the point of linearization, s_ω as δ.

$$\delta = s_c - s_\omega \tag{2.7.15}$$

Note that δ is a complex-valued quantity.

Next, we define the residual as the vector resulting from inserting s_c and the corresponding computed eigenvalue, ϕ_c, into equation 2.7.4.

$$\left(s_c^2 M + s_c D(s_c) + K\right)\phi_c = res \tag{2.7.16}$$

The residual, as defined in equation 2.7.16, is a computable quantity. If the residual is large, then the error in the computed eigenvalue and eigenvector is large. However, the more interesting question from the analyst's perspective is how large may δ be for one to expect accurate eigenvalues.

2.8. Linear Buckling

Buckling is the catastrophic failure of a structure under a specific load. Linear buckling is an approximation to that solution which is accurate in many load environments. Texts on the subject include Cook.[37]

In linear buckling analysis, a sample load is applied to the structure. The material and geometric stiffness matrices are computed, and an eigenvalue problem is used to determine under what load the total stiffness becomes singular. More specifically,

$$K_t = K_{\text{mat}} + K_{\text{geom}},$$

and

$$\left(K_{\text{mat}} - \lambda K_{\text{geom}}\right)\psi = 0 \tag{2.8.1}$$

Determination of the eigenvalue λ provides the scale factor that multiplies the sample load to determine the buckling load. The eigenvector ψ is an arbitrarily-normalized shape of the buckling deformation.

2.8.1. Eigen Problem Methods for Buckling

Note that (2.8.1) has the same form as equation (2.6.1) for the vibrational eigenvalue problem, with M being replaced by K_{geom}. For this reason, the numerical methods used to solve these problems are closely related, and it is recommended that the reader begin by reviewing Section 2.6.

The buckling problem is solved using a shift/invert strategy similar to that used in dynamics. The operator solved for buckling is,

$$\left(K_{\text{mat}} - \sigma K_{\text{geom}}\right)^{-1} K_{\text{mat}}; \tag{2.8.2}$$

c.f. (2.6.3). The main issue for the user is how to select an appropriate shift σ.

Some challenges arise in computing the solution because, unlike M, the matrix K_{geom} typically is not positive definite:

1. Because K_{geom} is not positive definite, we orthogonalize and normalize the vectors with respect to K_{mat}.
2. When K_{mat} is singular, the solution method can fail or give unexpected results. Most buckling problems clamp one end of the structure, so that is rarely a problem.
3. There are solutions possible when K_{mat} is singular, such as a piano wire that is singular until tensioned. We don't address these problems with our software, but encourage the analyst to explore that space.
4. Selection of an appropriate value for the shift becomes important. Some principles may be applied.
   a) The matrix $A = K_{\text{mat}} - \sigma K_{\text{geom}}$ is key.
   b) Formulation of (2.8.2) requires that $\sigma \neq 0$.
   c) σ should scale K_{geom} so it is large enough to modify K_{mat}.
   d) The eigenvalue solver will find solutions σ.
   e) Convergence is rapid if σ is chosen such that A is nearly singular. However, if A is singular, our linear solvers will fail.
   f) The sign of σ is important. Typically, loads that put the structure in compression should apply a positive value for σ.
5. For buckling, a negative or a positive shift σ may be appropriate depending upon the sign of the load. It is easy to get this wrong and converge to a mode other than the first buckling mode, or not to converge at all.

2.8.2. Buckling with Constraints

In this section, we derive the buckling equation (2.8.2) with constraints. Consider a structure with mass matrix M and stiffness matrix K. Our first problem of interest is to solve an eigenvalue problem in which the displacements u are subject to the constraints $Cu = 0$. Here, the rows of the constraint matrix C are assumed to be linearly independent.

As a starting point, let's first develop the unforced equations of motion using Lagrange's equations. The Lagrangian L can be defined as

$$L = T - U - \lambda^T Cu,$$

where the kinetic energy T and potential energy U are given by

$$T = \dot{u}^T M\dot{u}/2, \qquad U = u^T Ku/2,$$

and λ is a vector of Lagrange multipliers. Lagrange's equations of motion are

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{u}}\right) - \frac{\partial L}{\partial u} &= 0, \\ -\frac{\partial L}{\partial \lambda} &= 0, \end{aligned}$$

which can be expressed concisely as

$$\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \ddot{u} \\ \ddot{\lambda} \end{bmatrix} + \begin{bmatrix} K & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} u \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Assuming a solution of the form $u = \hat{u}e^{i\omega t}$ and $\lambda = \hat{\lambda}e^{i\omega t}$ leads to the eigenvalue problem

$$\underbrace{\begin{pmatrix} K & C^T \\ C & 0 \end{pmatrix}}_{\equiv \widetilde{K}}\begin{pmatrix} \hat{u} \\ \hat{\lambda} \end{pmatrix} = \omega^2 \underbrace{\begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}}_{\equiv \widetilde{M}}\begin{pmatrix} \hat{u} \\ \hat{\lambda} \end{pmatrix} \tag{2.8.3}$$

Thus, we can write the system as

$$\begin{aligned} x &= \begin{pmatrix} \hat{u} \\ \hat{\lambda} \end{pmatrix} \\ \widetilde{K}x &= \omega^2 \widetilde{M}x, \end{aligned}$$

Following the discussion in Section 2.6, this problem can be transformed as follows:

$$\widetilde{K}x - \sigma\widetilde{M}x = \omega^2\widetilde{M}x - \sigma\widetilde{M}x,$$

implying that

$$(\widetilde{K} - \sigma\widetilde{M})^{-1}\widetilde{M}x = (\omega^2 - \sigma)^{-1}x. \tag{2.8.4}$$

Solution of this transformed eigenvalue problem (2.8.4) can be done with the shift-invert mode in ARPACK. The linear system to be solved involves the matrix

$$\widetilde{K} - \sigma\widetilde{M} = \begin{pmatrix} K - \sigma M & C^T \\ C & 0 \end{pmatrix}, \tag{2.8.5}$$

which has the same constraint requirements as for a statics solve. The solver still needs to handle the constraints in the same manner despite the subtraction of σM. Note that the matrix $\widetilde{M}$ appearing after the matrix inverse in (2.8.4) does not include the constraint matrix C.

The buckling problem:

$$\min_{\substack{\hat{u} \text{ s.t.} \\ C\hat{u}=0}} \frac{1}{2}\hat{u}^T(K - \mu K_g)\hat{u}$$

has Lagrangian

$$L(\hat{u}, v) = \frac{1}{2}\hat{u}^T(K - \mu K_g)\hat{u} + v^T C\hat{u},$$

with partial derivatives

$$\begin{aligned} 0 = \frac{\partial L}{\partial \hat{u}} =& (K - \mu K_g)\hat{u} + C^T v \\ 0 = \frac{\partial L}{\partial v} =& C\hat{u}, \end{aligned}$$

implying the eigenvalue problem

$$\underbrace{\begin{pmatrix} K & C^T \\ C & 0 \end{pmatrix}}_{=\widetilde{K}} \begin{pmatrix} \hat{u} \\ v \end{pmatrix} = \mu \underbrace{\begin{pmatrix} K_g & 0 \\ 0 & 0 \end{pmatrix}}_{\equiv \widetilde{K_g}} \begin{pmatrix} \hat{u} \\ v \end{pmatrix}, \tag{2.8.6}$$

directly analogous to (2.8.3), with $x^T = \begin{pmatrix} \hat{u}^T & v^T \end{pmatrix}$.

The transformations used to solve the ARPACK buckling mode problem are somewhat different. Begin with multiplication of both sides by $\sigma \neq 0$:

$$\sigma \widetilde{K} x = \sigma \mu \widetilde{K_g} x,$$

and subtract $\mu \widetilde{K} x$ from both sides, leading to

$$\sigma \widetilde{K} x - \mu \widetilde{K} x = \sigma \mu \widetilde{K_g} x - \mu \widetilde{K} x,$$

implying that

$$(\mu - \sigma)\widetilde{K} x = \mu(\widetilde{K} - \sigma \widetilde{K_g})x$$

which can be rearranged to the form

$$(\widetilde{K} - \sigma \widetilde{K_g})^{-1} \widetilde{K} x = \frac{\mu}{\mu - \sigma} x. \tag{2.8.7}$$

The matrix required for the linear solves in this transformed problem has the same form as in (2.8.5), i.e.,

$$\widetilde{K} - \sigma \widetilde{K_g} = \begin{pmatrix} K - \sigma K_g & C^T \\ C & 0 \end{pmatrix}, \tag{2.8.8}$$

which implies that the constraint handling required by the linear solver itself is the same in both cases.

A critical difference between (2.8.4) and (2.8.7) is the form of the matrix that appears after the matrix inverse: $\widetilde{M}$ vs $\widetilde{K}$. Explicitly, these are:

$$\widetilde{K} = \begin{pmatrix} K & C^T \\ C & 0 \end{pmatrix}$$
$$\widetilde{M} = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}.$$

The matrix $\widetilde{K}$ is a semi-inner-product only for vectors $x^T = \begin{pmatrix} \hat{u}^T & v^T \end{pmatrix}$ such that $Cu = 0$. Thus, we must ensure that the vectors generated by the Arnoldi iteration always satisfy the constraint equations. In the code, it was necessary to implement an extra reorthogonalization step to accomplish this.

2.8.3. *Geometric Stiffness*

The geometric stiffness matrix, K_{geom}, is computed in two ways.

Stress: The Sierra transfer process uses stress as the variable to compute the tangent stiffness matrix. Stress is ideal in this case because the Sierra transfer also modifies the base coordinates of the nodes to match the deformed location. The stress is the only remaining variable in this formulation. It is important because we don't need the stress history (which could involve plasticity or other nonlinearities) to compute that tangent matrix.

Displacement: When **Sierra/SD** does its own nonlinear update, the tangent matrices are computed from the existing displacement variables. Element stress is not used at all.

These two methods of computation are equivalent in the small strain, small displacement world that is appropriate for a linear buckling calculation. The stress method is utilized for isoparametric solids. However, this method is not available for shells and beams. With these elements the geometric stiffness matrix uses a displacement based method.

2.8.3.1. Isosolid Elements.

The family of isogeometric continuum elements apply the following algorithms.

$$K_{\text{geom}} = \int_{\text{elem}} (\sigma : T) J dV \tag{2.8.9}$$

where,

$$T_{ij} = \frac{dN_i}{dx}' \frac{dN_j}{dx} - \text{sym}\left(\frac{dN_j}{dx}\right) \text{sym}\left(\frac{dN_i}{dx}\right)$$

Here sym(y) is the symmetric part of the matrix, the : represents a tensor product, dN/dx is the spatial derivative of the element shape function, and J is the Jacobian.

2.8.3.2. Corotational Shells.

The geometric stiffness contributions for corotational shells uses a formulation by Bjørn Haugen ([73]). Details are needed.

2.9. Component Mode Synthesis

Component mode synthesis (CMS) in **Sierra/SD** follows the Craig-Bampton method. In this method the model is reduced using fixed interface modes and constraint modes. The method is outlined in some detail in Craig (reference[39] Chapter 19). It is summarized below. Note that in **Sierra/SD** we do *not* permit any flexibility in the interface boundary options. Only fixed interface modes are supported.

CMS is typically applied to eigenvalue analysis, but it may be used in other analyses. Here we describe only the eigen analysis application. Within **Sierra/SD** only a subset of the standard CMS method is available. **Sierra/SD** may reduce *an entire model* to a set of interface degrees of freedom with the corresponding system matrices and transformed matrices. **Sierra/SD** may also read in a reduced system for solution within its framework.

CMS by these methods is always a linear model, with support for linear elasticity only. The reduction is based on an eigen reduction and linear superposition.

2.9.1. Reduction of superelement matrices

The entire model of a structure may be reduced to the interface degrees of freedom and generalized degrees of freedom associated with internal modes of vibration. Consider the general eigenvalue problem, with the system matrices partitioned into interface degrees of freedom, C, and the complement, the vibration modes, V.

$$\left(\begin{bmatrix} K_{vv} & K_{vc} \\ K_{cv} & K_{cc} \end{bmatrix} - \lambda \begin{bmatrix} M_{vv} & M_{vc} \\ M_{cv} & M_{cc} \end{bmatrix} \right) \begin{bmatrix} u_v \\ u_c \end{bmatrix} = 0 \tag{2.9.1}$$

Within **Sierra/SD** we consider only the cases where K_{vv} is nonsingular (i.e. positive definite). For the Craig-Bampton method clamping the interface degrees of freedom must remove all the zero energy modes of the structure.

The Craig-Bampton method reduces the physical degrees of freedom, u, to generalized coordinates, p, using a set of pre-selected component modes, Ψ.

$$u = \Psi p \tag{2.9.2}$$

The component modes, $\Psi = [\Phi, \psi]$, are the eigen-modes Φ, the fixed interface problem,

$$K_{vv}\Phi = M_{vv}\Phi\Lambda_{vv}$$

and the constraint modes ψ. In the fixed interface eigenvalue problem homogeneous Dirichlet boundary conditions are imposed on the interface, i.e. $\Phi_c = \mathbf{0}$. We retain only a (user specified) subset of the modes in the fixed interface problem. Additionally, the constraint modes, ψ, are the static condensation of the problem. Each column of ψ is the solution of the static problem where one interface degree of freedom has unit displacement, and all other interface degrees of freedom are fixed. As shown in the reference Craig ([39]),

$$\psi = -K_{vv}^{-1}K_{vc} \tag{2.9.3}$$

Note that our requirement that K_{vv} is positive definite implies that these solutions are well-defined.

Reduced System

In terms of the transformation matrix

$$T = \begin{bmatrix} \Phi & \psi \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{2.9.4}$$

the reduced system is $\mu = T^T M T$ and $\kappa = T^T K T$, which can be written,

$$\mu = \begin{bmatrix} \mu_{kk} & \mu_{kc} \\ \mu_{ck} & \mu_{cc} \end{bmatrix}, \qquad \kappa = \begin{bmatrix} \kappa_{kk} & \kappa_{kc} \\ \kappa_{ck} & \kappa_{cc} \end{bmatrix}, \tag{2.9.5}$$

in terms of

$$\begin{aligned} \mu_{kk} &= I_{kk} \\ \mu_{kc} &= \mu_{ck}^T = \phi^T (M_{vv}\psi + M_{vc}) \\ &= \phi^T M_{vv}\psi + (M_{cv}\phi)^T \\ \mu_{cc} &= \psi^T (M_{vv}\psi + M_{vc}) + M_{cv}\psi + M_{cc} \\ &= \psi^T M_{vv}\psi + (M_{cv}\psi)^T + M_{cv}\psi + M_{cc} \end{aligned} \tag{2.9.6}$$

and,

$$\begin{aligned} \kappa_{kk} &= \Lambda_{kk} \\ \kappa_{kc} &= \kappa_{ck} = 0 \\ \kappa_{cc} &= K_{cc} - K_{cv} K_{vv}^{-1} K_{vc} \\ &= K_{cc} + K_{cv}\psi \end{aligned} \tag{2.9.7}$$

Note that the coupling between the modal and interface portion of the system matrix occurs only in the mass matrix.

Parallelization Issues

The discussion above applies for direct solvers for which a system matrix is generated. Parallelization issues are straightforward, and cover 3 main areas 1) computation of fixed interface modes, 2) computation of constraint modes, and 3) matrix vector products.

1. **Fixed Interface Modes.** Since the process of computation of the eigenvalues is independent of the particular solver, there are no parallelization issues with respect to the eigenvalue problem. It is easily shown that parallel solvers result in the same eigen pairs as serial solvers. There is no reason to expect that any finite precision issues would be more important here than in other modal based solutions.
2. **Constraint Modes.** The constraint modes are different, in that we do not currently have a capability to compute enforced displacement in parallel. Recall that the constraint mode is the displacement on space "V" that is computed when a unit displacement is applied to a single degree of freedom on the interface. The serial equations are as follows.

$$\begin{bmatrix} K_{vv} & K_{vc} \\ K_{cv} & K_{cc} \end{bmatrix} \begin{bmatrix} u_v \\ u_c \end{bmatrix} = \begin{bmatrix} 0 \\ R \end{bmatrix} \tag{2.9.8}$$

Equation 2.9.3 uses the first of these only to solve for $u_v = \psi$. For a domain decomposition problem, the system matrices are written differently. We examine a two subdomain problem for clarity.

$$\begin{bmatrix} K_{1vv} & K_{1vc} & 0 & 0 & C_{1v}^T \\ K_{1cv} & K_{1cc} & 0 & 0 & C_{1c}^T \\ 0 & 0 & K_{2vv} & K_{2vc} & C_{2v}^T \\ 0 & 0 & K_{2cv} & K_{2cc} & C_{2c}^T \\ C_{1v} & C_{1c} & C_{2v} & C_{2c} & 0 \end{bmatrix} \begin{bmatrix} u_{1v} \\ u_{1c} \\ u_{2v} \\ u_{2c} \\ \mu \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ R \end{bmatrix} \tag{2.9.9}$$

We extract only the first and third rows to arrive at,

$$\begin{bmatrix} K_{1vv} & 0 & C_{1v}^T \\ 0 & K_{2vv} & C_{2v}^T \end{bmatrix} \begin{bmatrix} u_{1v} \\ u_{2v} \\ \mu \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \tag{2.9.10}$$

Here $f_i = K_{ivc} u_{ic}$. This system is the standard system of equations that is solved by the domain decomposition solver. The RHS is the sum of the individual subdomain terms.

3. **Matrix Vector Products.** There are two primary issues involved in the matrix vector products computed in parallel. First, there is the issue of duplication of some nodal quantities on the subdomain interfaces. Second, there is the issue of multipoint constraint handling.

   The products required in computing the reduced matrices of equations 2.9.5 through 2.9.7 are of the form, $a^T B c$, where a and c are vectors and B is a matrix. These are equivalent to element by element summations like those used in computing the total energy. Thus, the quantities must be summed on the interface. There is no need to divide by the number of shared interface degrees of freedom.

   Equation 2.9.1 partitioned according to Lagrange multipliers, χ, is

$$\left(\begin{bmatrix} K_{vv} & K_{vc} & C_v^T \\ K_{cv} & K_{cc} & C_c^T \\ C_v & C_c & 0 \end{bmatrix} - \lambda \begin{bmatrix} M_{vv} & M_{vc} & 0 \\ M_{cv} & M_{cc} & 0 \\ 0 & & 0 \end{bmatrix} \right) \begin{bmatrix} u_v \\ u_c \\ \chi \end{bmatrix} = 0 \tag{2.9.11}$$

   where χ are the Lagrange multipliers. But, we want these multipliers to be reduced out of the system (i.e. they should be in the “V” space), so it is useful to reorder the rows and columns of this equation.

$$\left(\begin{bmatrix} \tilde{K}_{vv} & \tilde{K}_{vc} \\ \tilde{K}_{cv} & K_{cc} \end{bmatrix} - \lambda \begin{bmatrix} \tilde{M}_{vv} & \tilde{M}_{vc} \\ \tilde{M}_{cv} & M_{cc} \end{bmatrix} \right) \begin{bmatrix} \tilde{u}_v \\ u_c \end{bmatrix} = 0 \tag{2.9.12}$$

   where,

$$\tilde{K}_{vv} = \begin{bmatrix} K_{vv} & C_v^T \\ C_v & 0 \end{bmatrix} \qquad \tilde{K}_{vc} = \begin{bmatrix} K_{vc} \\ C_c^T \end{bmatrix}$$

$$\tilde{M}_{vv} = \begin{bmatrix} M_{vv} & 0 \\ 0 & 0 \end{bmatrix} \qquad \tilde{u}_v = \begin{bmatrix} u_v \\ \chi \end{bmatrix}$$

   The matrix products are readily computed.

$$\begin{aligned} \tilde{M}_{vv}\tilde{u}_v &= M_{vv}u_v \\ \tilde{M}_{cv}\tilde{u}_v &= M_{cv}u_v \\ \tilde{K}_{cv}\tilde{u}_v &= K_{cv}u_v + C_c^T\chi \end{aligned}$$

   Thus, the mass products are simple – they do not require any special Lagrange multiplier treatment, but the stiffness product may require some such contribution. Note that if C_c is zero (as occurs if there is no constraint tied to the superelement interface) then the stiffness terms are likewise unchanged.

4. **Reduced transient problems and the inertia tensor**. CMS methods are often applied to the differential equation $Ku + M\ddot{u} = f$. Ideally the problem has a solution of the form $u(t) = Tq(t)$, using the transformation matrix defined in equation (2.9.4). These solutions are be computed from the reduced problem $\kappa q + \mu \ddot{q} = T^T f$. For a discretization of a floating structure, with rigid body modes R such that $KR = 0$, the solution satisfies the consistency condition $R^T M\ddot{u} = R^T f$.

   One way to impose the consistency condition uses the inertia matrix $I_{vv} = T^T R$. Suppose that there exists an S such that $R = TS + E$ has a solution, and the error E is negligible. Then the reduced consistency condition is $S^T \mu \ddot{q} = R^T f$. We use the solution S minimizing the norm of the error, E, and characterized by $T^T E = 0$. If T has full rank, then $S = (T^T T)^{-1} I_{vv}$. It is worthwhile to check that T is full rank and that κ and μ do not have common null spaces.

5. **Accuracy Issues.** The accuracy of the null space is determined by the sum of two large quantities (see equation 2.9.7). With iterative solvers, this may not be determined accurately enough to ensure stability of subsequent time history integration. Even unconditionally stable integration schemes like the trapezoidal Newmark-beta methods can become unstable if the stiffness matrix is indefinite.

   In our experience inaccurate solves decrease the accuracy of the rigid body energy modes with little impact on the remaining flexible modes. A post processing step corrects the rigid body modes. Two methods are used. The simpler method removes negative modes from the reduced matrix without affecting the eigenvector basis of the matrix. However, if the eigenvectors can be accurately determined using geometric means, then a better approach uses these known eigenvectors to correct both the eigenvalues and eigenvectors of the reduced matrix.

   To correct eigenvalues alone, we developed the following algorithm, based on the idea of matrix completion [40].

   a) We extract the interface portion of the reduced system matrix, κ_{cc}. Note that the portion of the matrix associated with generalized degrees of freedom (i.e. the fixed interface modes) should be positive definite.

   b) We perform an eigen analysis of this matrix.

   $$\kappa_{cc} = V\Delta V^T$$

   where $V_j i$ is the eigenvector, and Δ_i is the eigenvalue of mode i.

   c) We determine a corrected matrix,

   $$\tilde{\kappa}_{cc} = \kappa_{cc} - \sum_{j}^{negative\,modes} V_j \Delta_j V_j^T$$

   To correct both eigenvalues *and eigenvectors* of the corrupted null space, the algorithm is more involved. Details of the algorithm are presented in Figure 2-6. Most of the operations in the algorithm operate on matrices of order 12 or smaller, so the computational cost is minimal. The method does require practically exact zero energy modes.

1. Determine rigid body modes, R, of the interface. This is done geometrically. These are normalized so that $R^T R = I$. Typically there are 6 such vectors.
2. Let, $A = R^T \kappa_{cc} R$.
3. Compute a error vector, $U = \kappa_{cc} R - RA$. Note that $R^T U = 0$
4. Perform a QR factorization of the error vector. $U = SB$. Matrix S has orthonormal columns.
5. Define $Q = [R \ S]$
6. Compute the norm of the matrix composed of A and B.
$$\mu = \left\| \begin{bmatrix} A \\ B \end{bmatrix} \right\|$$
7. Compute the eigenvalues of A.
$$(A - \lambda I)\phi_a = 0$$
8. Compute $G = \mu^2 I - \lambda^2$.
9. $W = \phi_a \sqrt{G} \phi_a^T$
10. $D = -BW^{-1}AW^{-1}B^T$
11. define,
$$H = \begin{pmatrix} A & B^T \\ B & D \end{pmatrix}$$
note that $||H|| = \mu$.
12. Compute the correction,
$$\tilde{\kappa}_{cc} = \kappa_{cc} - QHQ^T$$

Figure 2-6. – Eigenvalue and Eigenvector corrections of Craig-Bampton reduced models

2.9.2. Craig-Bampton sensitivity analysis

Sierra/SD may compute the sensitivity of the reduced mass and stiffness matrices to design variables. In term of the transformation matrix (see equation (2.9.4))

$$\kappa = T^T K T \tag{2.9.13}$$

Sensitivity of the matrix to variations in a parameter may be obtained by differentiating this equation. There are several approaches to that operation.

Constant Vector The transformation matrix T, is treated as a constant. Thus, the original model and its derivative are transformed into the modal space of the original structure. If there are sufficient modes to span the space, this operation is exact. We designate T_o as the transformation matrix for that original modal space, and use forward differences to write the derivative.

$$\frac{d\kappa}{dp} \approx \frac{T_o^T \left(K(p+\Delta p) - K(p)\right) T_o}{\Delta p} \tag{2.9.14}$$

In the limit as Δp approaches zero, this should approach the exact solution provided that T_o spans the space.

However, practically we truncate the modal space spanned by T_o. In many real world cases, that truncation is unable to accurately represent the derivatives.

Finite Difference In this approach, we recompute the entire model, including the transformation matrix, at both the nominal and perturbed state. Thus, $K_1 = K(p + \Delta p)$ and $T_1 = T(p + \Delta p)$. Using forward differences,

$$\frac{d\kappa}{dp} \approx \frac{T_1^T K(p+\Delta p) T_1 - T_o^T K(p) T_o}{\Delta p} \tag{2.9.15}$$

The finite difference method accurately represents the state at both the nominal and perturbed states. In the limit as Δp approaches zero, the method converges to the true solution.

However, problems will be encountered if there are closely spaced (or repeated) modes.[53,85] Consider the reduced matrices, which have both physical and generalized degrees of freedom. If a closely spaced mode changes sort order in the matrix, the derivative is meaningless. With repeated modes, the issue is even more difficult as the eigenvectors of repeated modes may be linearly combined. Also, any eigenvector has an arbitrary sign. To help diagnose these problems, we output the mass cross orthogonality matrix.

$$A_{ij} = \phi_j^T M \phi_i \tag{2.9.16}$$

Product Rule The finite difference method is treated like an exact method. However, in the case of CB reduction, the changes in eigenvectors make the method complicated. Another approach would be to differentiate equation 2.9.13 using the product rule.

$$\frac{d\kappa}{dp} = \frac{dT^T}{dp} K T + T^T \frac{dK}{dp} T + T^T K \frac{dT}{dp} \tag{2.9.17}$$

Several means[66,106,135] are available to determine the derivatives of the fixed interface modes, ϕ, and constraint modes, ψ, which are the components of the transformation matrix. This approach blends the best features of both previous methods, but is more complex to develop.

This method is currently unimplemented.

2.10. Eigenvalue Sensitivity Analysis

Within **Sierra/SD** semi-analytic sensitivities may be computed for eigenvalues and eigenvectors. A rudimentary capability for sensitivity to linear transient response is also available, but has not found much practical value because the cost of the analysis is not significantly better than the cost of computing the response using finite differences. For details of the transient analysis formulation, see Alvin's paper,.[4]

For eigenvalue sensitivity, we begin with linear eigenvalue equation.

$$(K - \lambda M)\,\phi = 0 \tag{2.10.1}$$

The equation is differentiated with respect to a sensitivity parameter, p, and we consider the solution for a single eigen pair.

$$\begin{aligned} (dK - d\lambda_i M - \lambda_i dM)\,\phi_i + (K - \lambda_i M)\,d\phi_i &= 0 \qquad (2.10.2)\\ \phi_i^T\,(dK - d\lambda_i M - \lambda dM)\,\phi_i &= 0 \qquad (2.10.3)\end{aligned}$$

where we use the fact that $\phi_i^T(K - \lambda_i M)$ is zero. We note that $\phi^T M \phi$ is the identity to solve for the sensitivity.

$$d\lambda_i = \phi_i^T dK \phi_i - \lambda_i \phi_i^T dM \phi_i \tag{2.10.4}$$

The method is "semi-analytic" in that the matrices dK and dM are found by finite differences but then are applied to the analytic expression above. Because there are no linear solves required, the solution is straightforward and accurate.

The algorithm used for the solution of eigenvalue sensitivity is as follows.

1. Perform nominal eigenvalue solution.
2. Loop through parameters P, and modify as needed.
3. On an element by element basis compute,

$$\begin{aligned}\kappa &= (K + dK)\phi\\ \mu &= (M + dM)\phi\end{aligned}$$

4. compute the sensitivity, $d\lambda = \phi^T\kappa - \lambda\phi^T\mu$.

This element by element method conserves memory and is efficient. It has been implemented successfully for all parallel solvers. It has not been implemented for the *sparsepak* solver when MPCs are included in the model. The transformations required for multipoint constraints complicate the element by element calculation.

There are many algorithms[135] for computing eigenvector sensitivity. Nelson's method[106] expresses eigenvector sensitivity implicitly,

$$f_i = -\left(dK - \lambda_i dM - d\lambda_i M\right), \quad (K - \lambda_i M)d\phi_i = f_i,$$

requiring one linear solve per eigenvector sensitivity. It suffers from singularity issues with redundant modes and from accuracy limitations when only part of the modes are extracted. For computational efficiency, the linear solve uses a preconditioned conjugate gradient algorithm,

$$(K - \lambda_i M)w_i = f_i - (K - \lambda_i M)\Phi c_i \tag{2.10.5}$$

Because this operator is indefinite, we redefine the problem as,

$$w_i = \Psi x_i, \quad (\Psi^T(K - \lambda_i M)\Psi)x_i = \psi^T(f_i - (K - \lambda_i M)\Phi c_i). \tag{2.10.6}$$

The operator $(\Psi^T(K - \lambda_i M)\Psi)$ is positive definite as long as mode i and all modes below mode i are contained in Φ.

Forward sensitivity of linear transient dynamics solutions was not found to be useful. For details on sensitivity on the reduction of superelements see Section 2.9.2.

2.11. A posteriori error estimation for eigen analysis

The purpose of this section is to summarize two different approaches for a posteriori error estimation of eigen analysis. The first is an explicit error estimator,[89],[74] and the second is a quantity of interest approach.[108] The explicit approaches are described in chapter 2 of,[1] and the quantity of interest approaches are described in chapter 8 of the same book. However, since we are interested in the eigenvalue problem, the methodologies are somewhat different than the approaches described in,[1] though there are many similarities. Both the explicit and the quantity of interest approaches have the same goal - to use the computed solution to compute upper and lower bounds on the discretization error for the eigenvalues and eigenvectors. A drawback to the explicit approach is that unknown constants are present in the bounds, making final determination of the error more difficult. Because of this, an explicit estimator is more frequently used as an element indicators to drive adaptivity algorithms, rather than as an error estimator. The quantity of interest approach avoids the unknown constants, but is more work in terms of implementation.

2.11.1. Preliminaries

We seek a posteriori bounds on the error of the finite element solution of the eigenvalue problem for elasticity

$$-\rho\lambda u = (\Lambda + \mu)\nabla(\nabla \cdot u) + \mu\nabla^2 u = \nabla \cdot \sigma(u) \tag{2.11.1}$$

or

$$A_1(u) = -\lambda A_2(u) \tag{2.11.2}$$

where $A_1(u)$ and $A_2(u)$ are the partial differential operators implied by equation 2.11.1, λ and u are the unknown eigenvector and eigenvalue, and Λ and μ are the Lamé elasticity constants. We note that the right-hand side of equation 2.11.1 can be written either in terms of displacement, as in the first representation, or in terms of stress, as in the second representation of the right-hand side of the equation. The weak formulation of equation 2.11.1 is constructed by multiplying by a test function, and integrating by parts, with homogeneous boundary conditions. This leads to the weak formulation: Find $(\lambda, u) \in V \times R$ such that

$$B(u, v) = \lambda M(u, v) \quad \forall v \in V \tag{2.11.3}$$

where

$$B(u, v) = \int_\Omega \sigma(u)\epsilon(v)dx \tag{2.11.4}$$

and

$$M(u, v) = \int_\Omega \rho uv dx \tag{2.11.5}$$

After defining a finite element discretization, this reduces to: Find (u_h, λ_h) such that

$$Ku = \lambda Mu \tag{2.11.6}$$

where (u_h, λ_h) are the finite element approximations of the eigenvector and eigenvalue, and K, M, are the assembled stiffness and mass matrices.

2.11.2. An explicit error estimator

In *Larsen*[89] and *Rannacher*,[74] two independently derived error estimates are presented for the Laplace equation. While the two estimates differ, both incorporate an unknown constant, C, an element diameter term, h_e, and an element residual function, $\bar{\rho}$. In what follows we extend these estimates to the elasticity problem. The following two error estimates are given in[89] and[74] respectively. In what follows we use Larsen's results (equation 2.11.7) exclusively. [9]

$$|\lambda - \lambda_h| \leq c\lambda C_{e,0} \left(\sum_{e=1}^{N_e} h_e^4 \bar{\rho}(u_h, \lambda_h)^2 \right)^{\frac{1}{2}} \tag{2.11.7}$$

$$|\lambda - \lambda_h| \leq C_2 \sum_{e=1}^{N_e} h_e^2 \bar{\rho}(u_h, \lambda_h)^2 \tag{2.11.8}$$

where h_e is the element diameter, and

$$\bar{\rho}(u_h, \lambda_h)^2 = \int_{\Omega_e} \left(|A_1 u_h + \lambda_h A_2 u_h| + R_{flux} \right)^2 d\Omega_e \tag{2.11.9}$$

The first term on the right-hand side is the interior element residual, which is the differential stiffness operator A_1, defined in equation 2.11.2, applied to the computed element displacement combined with the computed eigenvalue times the differential mass operator A_2, also defined in equation 2.11.2, applied to the computed element displacement. This term is computed by representing the eigenvector as a summation

$$u_h(x) = \sum_{i=1}^{N} a_i N_i(x) \tag{2.11.10}$$

where a_i is the i^{th} entry in the eigenvector, and $N_i(x)$ is the i^{th} shape function, and then applying the gradient and divergence operators from equation 2.11.1 to the summation in equation 2.11.10.

We note that the quantity $A_1 u_h + \lambda_h A_2 u_h$ is expressed in the strong form, and thus is not the same as $Ku_h - \lambda_h M u_h$, though both expressions are on the element level. The difference can be seen by observing the first term $A_1 u_h$

$$A_1 u_h = \nabla \cdot \sigma(u_h) \tag{2.11.11}$$

That is, $A_1 u_h$ is the divergence of the stress (which is computed from the finite element displacement u_h). This is not the same as Ku_h, since Ku_h is in the weak form, and has been evaluated by integrating over the element against a test function. For example, if we consider linear elements, we have $A_1 u_h = \nabla \cdot \sigma(u_h) = 0$, since the stress is constant over the element. On the other hand, Ku_h is not zero.

[9]Equation 2.11.7 applies to elements with linear shape functions. The more general expression may be found in equation 2.11.57 or the reference.

The second term is the boundary or flux residual.

$$R_{flux} = (h_e vol(e))^{-1/2} \left[\int_{\Gamma_e} R^2 d\Gamma_e \right]^{1/2} \tag{2.11.12}$$

It has two different integrands depending on whether the face in question lies on a part of the boundary where traction or pressure boundary conditions are applied, or whether it is an interior face. When it lies on a boundary loaded face,

$$R = g - \sigma_{ij} n_j \tag{2.11.13}$$

where g is the applied traction or pressure load. Note that $g = 0$ for eigen problems. When the face is an interior face,

$$R = \left[\sigma_{ij} n_j \right] = \sigma^a_{ij} n_j - \sigma^b_{ij} n_j \tag{2.11.14}$$

where σ^a and σ^b are the stress tensors in the two adjacent elements, element 'a' and element 'b'. Note that because the integrand is squared, computing the flux residual in parallel requires parallel communication.

We note the intuitive nature of the upper bound in equation 2.11.7. As the element size h_e tends to zero, the right-hand sides of the estimate goes to zero, due to the multiplication by the element sizes h_e. Keep in mind also that the $\bar{\rho}$ term includes an integral over a volume and that $\sum_{e=1}^{N_e} ||\text{const}||$ is a constant.

There are two important issues in applying the results in Larsen's reference to general elasticity problems. The first of these is the extension to elasticity. The second is the extension to multiple materials. These are covered in the following sections.

2.11.3. Error estimates for elasticity

This section was provided by Ulrich Hetmaniuk to help us with problems in scaling the Laplace equation to the elasticity problem. It addresses issues of both mass and stiffness scaling. A similar development was provided by Clark Dohrmann. The development herein builds upon Larsen's development,[89] and uses quantities defined there.

We consider the eigenvalue problem

$$\begin{aligned} -\mu\Delta\mathbf{u} - (\Lambda + \mu)\nabla(\nabla\cdot\mathbf{u}) = -\nabla\cdot\sigma(\mathbf{u}) &= \theta\rho\mathbf{u} \quad \text{in } \Omega & (2.11.15) \\ \mathbf{u} &= \mathbf{0} \quad \text{on } \partial\Omega & (2.11.16) \end{aligned}$$

where the Lamé constants Λ and μ satisfy

$$\Lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)} \tag{2.11.17}$$

We define also a weak formulation: find $(\mathbf{u}, \theta) \in \mathbb{V} \times \mathbb{R}$

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) &= \theta b(\mathbf{u}, \mathbf{v}), \quad \forall\, \mathbf{v} \in \mathbb{V} & (2.11.18) \\ b(\mathbf{u}, \mathbf{u}) &= 1 & (2.11.19) \end{aligned}$$

where

$$a(\mathbf{u}, \mathbf{v}) = \int_\Omega \sigma(\mathbf{u}) \cdot \epsilon(\mathbf{v}) dx \tag{2.11.20}$$

and

$$b(\mathbf{u}, \mathbf{v}) = \int_\Omega \rho\mathbf{u} \cdot \mathbf{v} dx \tag{2.11.21}$$

We follow the approach in the paper by M. Larson to derive an *a posteriori* error estimator. We use most of his notation.

Residual

The definition (3.7) for the residual becomes, on a triangle τ,

$$R(\mathbf{u}_h, \theta_h)_{|\tau} = \frac{1}{\sqrt{\rho}}|\nabla \cdot \sigma(\mathbf{u}_h) + \theta_h \rho \mathbf{u}_h| + \sqrt{\frac{1}{h\, vol(\tau)} \int_{\partial\tau \setminus \partial\Omega} \left(\mathbf{n} \cdot \left[\frac{\sigma(\mathbf{u}_h)}{2\sqrt{\rho}}\right]\right)^2} \tag{2.11.22}$$

Note that we have

$$R(\mathbf{u}_h, \theta_h) \equiv R(\mathbf{u}_h, \theta_h, \rho, E, \nu) \tag{2.11.23}$$

and that R satisfies the following scaling properties

$$R(\frac{\mathbf{u}_h}{\sqrt{\alpha}}, \frac{\theta_h}{\alpha}, \alpha\rho, E, \nu) = \frac{1}{\alpha} R(\mathbf{u}_h, \theta_h, \rho, E, \nu) \tag{2.11.24}$$

$$R(\mathbf{u}_h, \alpha\theta_h, \rho, \alpha E, \nu) = \alpha R(\mathbf{u}_h, \theta_h, \rho, E, \nu) \tag{2.11.25}$$

Stability estimates

The equation (3.10) becomes

$$||D^{2+s}\mathbf{v}|| \leq C_{e,s} \sqrt{b\left(\left(\frac{-1}{\rho}\nabla \cdot \sigma\right)^{1+s/2}(\mathbf{v}), \left(\frac{-1}{\rho}\nabla \cdot \sigma\right)^{1+s/2}(\mathbf{v})\right)} \tag{2.11.26}$$

Note that

$$\Lambda + \mu = \frac{E}{2(1+\nu)(1-2\nu)} \quad , \quad \frac{\mu}{\Lambda + \mu} = 1 - 2\nu \tag{2.11.27}$$

Then, we get

$$C_{e,s} = c \frac{\rho^{(1+s)/2}}{(\Lambda + \mu)^{(2+s)/2}} \tag{2.11.28}$$

Note that we have

$$C_{e,s} \equiv C_{e,s}(\rho, E, \nu) \tag{2.11.29}$$

and that $C_{e,s}$ satisfies the following scaling properties

$$C_{e,s}(\alpha\rho, E, \nu) = \alpha^{(1+s)/2} C_{e,s}(\rho, E, \nu) \tag{2.11.30}$$

$$C_{e,s}(\rho, \alpha E, \nu) = \frac{1}{\alpha^{(2+s)/2}} C_{e,s}(\rho, E, \nu) \tag{2.11.31}$$

A posteriori estimates

We make also the assumption (2.6) : there are $0 \leq \delta < 1$ and $h_0 > 0$ such that

$$\max_{\theta_i \notin \Theta} \left|\frac{\theta_h - \theta}{\theta_i - \theta}\right| \leq \delta \quad , \quad ||Q_\Theta \mathbf{u}_h||^2 \leq \delta \tag{2.11.32}$$

for all meshes such that $\max h(x) \leq h_0$. Using $p = 1$, $k = 2$, $\beta_0 = 0$, and $\beta_1 = 1$, the final estimate on the eigenvalues becomes

$$\frac{\theta_h - \theta}{\theta} \leq \frac{c}{1-\delta} C_{e,0} \sqrt{\rho} ||h^2 R(\mathbf{u}_h, \theta_h)|| \tag{2.11.33}$$

The estimates on the error in the discrete eigenvector are now

$$\sqrt{b(\mathbf{e}_\Theta, \mathbf{e}_\Theta)} \leq \frac{c}{1-\delta} C_{e,0}(1 + \max_{\theta_i \notin \Theta} \frac{\theta}{|\theta_i - \theta|}) \sqrt{\rho} ||h^2 R(\mathbf{u}_h, \theta_h)|| \tag{2.11.34}$$

$$\sqrt{a(\mathbf{e}_\Theta, \mathbf{e}_\Theta)} \leq \frac{c\sqrt{\rho}}{1-\delta}(C_c + C_{e,0} \max_{\theta_i \notin \Theta} \frac{\theta\theta_i^{1/2}}{|\theta_i - \theta|} h_{max}) ||hR(\mathbf{u}_h, \theta_h)|| \tag{2.11.35}$$

where C_c is related to the coercivity constant

$$||D\mathbf{v}|| \leq C_c \sqrt{a(\mathbf{v}, \mathbf{v})} \tag{2.11.36}$$

In Ciarlet's book *("The finite element method for elliptic problems")*, the coercivity constant is given

$$a(\mathbf{v}, \mathbf{v}) \geq 2\mu ||D\mathbf{v}|| \quad \Rightarrow \quad C_c = \frac{c}{\sqrt{2\mu}} \tag{2.11.37}$$

2.11.4. Explicit Estimator - Multiple Materials

To date, we have not seen any publication which extends the explicit error estimator to multiple materials. We don't believe that there are significant issues, and present the approach used in **Sierra/SD** here. There are two main constraints from the explicit error estimator formulations that must be maintained.

1. The eigenvectors, u_h must be unit normalized, i.e.$\|u_h\| = 1$. This is important for mass scaling so that a change of units does not change the fractional error in the solution. It is an essential part of both Larsen's development and Ulrich's extension to elasticity. See equation 2.11.19.
2. The extensions must maintain finite element consistency so that as h goes to zero there is no inconsistency.

The second of these can be evaluated by examination of the residuals (as in equation 2.11.9). Both the internal and the flux terms of the residuals are unchanged by most scaling operations provided that materials remain constant within an element. Note that the evaluation of the flux jump (equation 2.11.12) is insensitive to multiple materials since the normal component of stress discontinuity should go to zero even for disparate materials.

Eigenvector normalization could be addressed in several ways. The eigenvectors computed in **Sierra/SD** are mass normalized, i.e. $u^T M u = I$. We renormalize for error estimation in the following manner.

1. A dimensionless mass matrix, $\bar{M}$ is computed using unit density material.
2. We compute a scale factor
$$m_\alpha = u^T \bar{M} u \tag{2.11.38}$$
3. The eigenvectors are renormalized as $u \leftarrow u/\sqrt{m_\alpha}$.

In addition to eigenvector renormalization, we move the evaluation of the scaling constant, C_e, s, from equation 2.11.28 inside the summation of equation 2.11.7. This maintains the proper scaling with respect the element stiffness terms.

A recent paper by Bernardi and Verfurth[23] has shown that an explicit estimator can be used in the presence of multiple materials. For static Laplace equation, he derived multiplicative constants for the interior and flux contributions that make the multiplicative constant in front of the estimator independent of jumps in material properties. In what follows we extend this approach to the eigenvalue problem, and to elasticity problems. We will follow the same approach as in that paper, i.e. first constructing the lower bound, and then the upper bound. The proper choices for the coefficients will result from the upper and lower bound estimates.

First, we note a commonly used form for an explicit estimator.

$$\|\mathbf{u_h} - \mathbf{u}\|_\alpha \leq c \sum_K \left(h \|R_i(\mathbf{u_h}, \theta_h)\|^2_{L^2(K)} + \sqrt{h} \| \frac{[\sigma_n(\mathbf{u_h})]}{2} \|^2_{L^2(\partial K)} \right)^{\frac{1}{2}} \tag{2.11.39}$$

where $R_i(\mathbf{u_h}, \theta_h) = |\nabla \cdot \sigma(\mathbf{u}_h) + \theta_h \rho \mathbf{u}_h|$, $[\sigma_n(\mathbf{u_h})]$ is the jump in stress across the element boundary ∂K, and $\| \cdot \|_\alpha$ is the energy norm. This estimator can be shown to give both an upper and a lower bound on the error. As written, this estimator does not account for discontinuities in material properties, since the constant c in front of the estimator would depend on the jumps in material properties.

We note that the estimator, written in this form, is essentially the same as the one proposed by Larson. For example, by writing the boundary term as an integral of a constant function, scaled by the volume of the element, then we can write equation 2.11.39 in the form

$$\|u_h - u\|_\alpha \leq c \sum_K \left(\|h R_i(u_h, \theta_h) + \frac{\sqrt{h}}{Vol(K)} \frac{[\sigma_n(\mathbf{u_h})]}{2} \|^2_{L^2(K)} \right)^{\frac{1}{2}} \tag{2.11.40}$$

which is the same expression given by Larson in the case of linear elements. We note that this estimator is in terms of the energy norm, whereas Larson gives his results in terms of the L^2 norm. This results in the difference of one power of h in equation 2.11.40.

The approach in Bernardi is to replace the estimator in equation 2.11.39 by

$$\|\mathbf{u_h} - \mathbf{u}\|_\alpha \leq c \sum_K \left({\mu_K}^2 \|R_i(\mathbf{u_h}, \theta_h)\|^2_{L^2(K)} + \mu_e \| \frac{[\sigma_n(\mathbf{u_h})]}{2} \|^2_{L^2(\partial K)} \right)^{\frac{1}{2}} \tag{2.11.41}$$

where μ_K and μ_e are chosen in such a way that the resulting estimator is both an upper and lower bound on the error, and the constant c is independent of the jumps in material properties.

Before beginning, we redefine the original PDE as follows

$$\frac{-\nabla \cdot \sigma}{\rho} = \theta \mathbf{u} \tag{2.11.42}$$

the corresponding bilinear forms as

$$a(\mathbf{u}, \mathbf{v}) = \int_\Omega \frac{1}{\rho} \sigma(\mathbf{u}) \cdot \epsilon(\mathbf{v}) d\mathbf{x}$$
$$b(\mathbf{u}, \mathbf{v}) = \int_\Omega \mathbf{u} \cdot \mathbf{v} d\mathbf{x}$$

and the corresponding interior residual as

$$R_i(\mathbf{u_h}, \theta_h) = | \frac{\nabla \cdot \sigma(\mathbf{u}_h)}{\rho} + \theta_h \mathbf{u}_h | \tag{2.11.43}$$

By dividing through by ρ, we include the density in the energy norm. This will be important later on when the coefficients in equation 2.11.41 are selected.

As in Bernardi, we need the following identities, which follow from equation 2.11.3

$$a(\mathbf{u}-\mathbf{u_h},\mathbf{v}) = \theta b(\mathbf{u},\mathbf{v}) - a(\mathbf{u_h},\mathbf{v}) \tag{2.11.44}$$

$$\begin{aligned}\theta b(\mathbf{u},\mathbf{v}) - a(\mathbf{u_h},\mathbf{v}) = \sum_K \int_K \left(\theta\mathbf{u} + \frac{1}{\rho}\nabla\cdot\sigma(\mathbf{u}_h)\right)\mathbf{v}d\mathbf{x} - \\ \sum_e \int_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right]\cdot\mathbf{v}d\tau\end{aligned} \tag{2.11.45}$$

where the summation $\sum_e$ is over all edges (in 2D) or over all faces (in 3D). We also use equations 2.11 in Bernardi's paper.

The lower bound will be considered first. We set $w_K = \Psi_K R_i(\mathbf{u_h},\theta_h)$, where Ψ_K comes from equation 2.11 in Bernardi's paper. We will also make use of the following inequality for the bilinear form

$$\begin{aligned} a(\mathbf{u},\mathbf{v})_K &\leq \|\mathbf{u}\|_\alpha\|\mathbf{v}\|_\alpha && (2.11.46)\\ &\leq \alpha_K\|\mathbf{u}\|_1\|\mathbf{v}\|_1 && (2.11.47)\end{aligned}$$

where $\alpha_K = \frac{C_K}{\rho_K}$, and C_K is the maximum eigenvalue of the material property matrix, and ρ_K is the density of the element.

For the interior part of the residual, we have

$$\begin{aligned}\|R_i(u_h,\theta_h)\|^2_{L^2(K)} &\leq \gamma_1^2\int_K\left[\frac{1}{\rho}\nabla\cdot\sigma(\mathbf{u_h})+\theta_h\mathbf{u_h}\right]\cdot\mathbf{w_K}d\mathbf{x}\\ &= -\gamma_1^2\int_K\frac{1}{\rho}\sigma(\mathbf{u_h})\cdot\epsilon(\mathbf{w_K})d\mathbf{x}+\gamma_1^2\int_K\theta_h\mathbf{u_h}\cdot\mathbf{w_K}\\ &= \gamma_1^2 a(\mathbf{u}-\mathbf{u_h},\mathbf{w_K})_K - \gamma_1^2\theta\int_K\mathbf{u}\cdot\mathbf{w_K}d\mathbf{x}+\gamma_1^2\theta_h\int_K\mathbf{u_h}\cdot\mathbf{w_K}d\mathbf{x}\\ &\leq \gamma_1^2\left[\|\mathbf{u}-\mathbf{u_h}\|_{\alpha(K)}\gamma_2 h_K^{-1}\alpha_K^{\frac{1}{2}}+\|\theta_h\mathbf{u_h}-\mathbf{u}\theta\|_{L^2(K)}\right]\\ &\times \|R_i(u_h,\theta_h)\|_{L^2(K)}\end{aligned} \tag{2.11.48}$$

where we note that, since Ψ_K is a bubble function, the boundary terms vanish in the integration by parts on the second line of the above equation.

This implies that

$$\|R_i(u_h,\theta_h)\|_{\alpha(K)} \leq \gamma_1^2\left[\|\mathbf{u}-\mathbf{u_h}\|_{\alpha(K)}\gamma_2 h_K^{-1}\alpha_K^{\frac{1}{2}}+\|\theta_h\mathbf{u_h}-\mathbf{u}\theta\|_{L^2(K)}\right]$$

or, multiplying through by μ_K,

$$\mu_K\|R_i(u_h,\theta_h)\|_{\alpha(K)} \leq \gamma_1^2\left[\|\mathbf{u}-\mathbf{u_h}\|_{\alpha(K)}\mu_K\gamma_2 h_K^{-1}\alpha_K^{\frac{1}{2}}+\mu_K\|\theta_h\mathbf{u_h}-\mathbf{u}\theta\|_{L^2(K)}\right]$$

We assume that the computed eigenpair θ_h and $\mathbf{u_h}$ are closer to the exact solution θ and $\mathbf{u}$ than any other exact eigenpair. This assumption is also made by Larson, in equation 2.6. With this assumption, the term

$\|\theta_h\mathbf{u_h} - \mathbf{u}\theta\|_{L^2(K)}$ is a higher order term compared with $\|\mathbf{u} - \mathbf{u_h}\|_{\alpha(K)}$, and thus will decay to zero at a faster rate. This was also shown in the paper by Duran.[52] Thus, we select μ_K based on the term $\|\mathbf{u} - \mathbf{u_h}\|_{L^2(K)}$ only. If we select $\mu_K = h_K\alpha_K^{-\frac{1}{2}}$ then the right-hand side is independent of the jumps in material properties.

For the boundary term, we first choose $w_e = \Psi_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right]$, where again Ψ_e comes from equation 2.11 in Bernardi. Then, using equation 2.11.48 we have

$$\begin{aligned}
\| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|^2_{L^2(e)} &\leq \gamma_3^2 \int_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot \mathbf{w_e} d\tau \\
&= \gamma_3^2 \sum_K \int_K \left(\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta_h \mathbf{u}_h\right) \cdot \mathbf{w_e} - \gamma_3^2 \sum_K a(\mathbf{u} - \mathbf{u_h}, \mathbf{w_e}) \\
&+ \gamma_3^2 \sum_K \int_K (\theta\mathbf{u} - \theta_h\mathbf{u_h}) \cdot \mathbf{w_e} \\
&\leq c\gamma_3^2 \left(\sum_K \gamma_5 h_e^{\frac{1}{2}} \|R_i(u_h, \theta_h)\|_{L^2(K)} + \sum_K \gamma_4 h_e^{-\frac{1}{2}} \alpha_K^{\frac{1}{2}} \|\mathbf{u} - \mathbf{u_h}\|_\alpha \right.\\
&+ \left. \gamma_5 h_e^{\frac{1}{2}} \sum_K \|\mathbf{u}\theta - \mathbf{u_h}\theta_h\|_{L^2(K)}\right) \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)} \\
&\leq c\gamma_3^2 \left[\sum_K h_e^{-\frac{1}{2}} \alpha_K^{\frac{1}{2}} \|\mathbf{u} - \mathbf{u_h}\|_\alpha + \sum_K h_e^{\frac{1}{2}} \|\theta_h\mathbf{u_h} - \theta\mathbf{u}\|_{L^2(K)}\right] \\
&\times \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)}
\end{aligned} \tag{2.11.49}$$

where in the above equation, $\sum_K$ denotes a summation over elements, but only those elements that border the edge e. Also, in the previous estimate we collected constants involving γ and combine with the constant c, where possible.

This implies that

$$\mu_e^{\frac{1}{2}} \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)} \leq c\gamma_3^2 \mu_e^{\frac{1}{2}} \left[\sum_K h_e^{-\frac{1}{2}} \alpha_K^{\frac{1}{2}} \|\mathbf{u} - \mathbf{u_h}\|_\alpha + \sum_K h_e^{\frac{1}{2}} \|\theta_h\mathbf{u_h} - \theta\mathbf{u}\|_{L^2(K)}\right]$$

We see that if we choose $\mu_e = h_e \max\left(\alpha_{K1}, \alpha_{K2}\right)^{-1}$, where subscripts 1 and 2 denotes the two neighboring elements that contain the edge or face e, then the right-hand side (neglecting the higher order term) is independent of the jumps in material properties.

Now we construct the upper bound. We start with a few identities that will be needed along the way.

$$\begin{aligned}
\int_\Omega \left(\frac{1}{\rho}\nabla \cdot \sigma(\mathbf{u}_h) + \theta\mathbf{u}\right) \cdot (\mathbf{w} - \mathbf{w_h}) = -a(\mathbf{u_h}, \mathbf{w} - \mathbf{w_h}) + \\
\sum_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot (\mathbf{w} - \mathbf{w_h}) + \int_\Omega \theta\mathbf{u}(\mathbf{w} - \mathbf{w_h})
\end{aligned} \tag{2.11.50}$$

This implies that

$$a(\mathbf{u_h}, \mathbf{w} - \mathbf{w_h}) = \sum_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot (\mathbf{w} - \mathbf{w_h})$$
$$+ \int_\Omega \theta \mathbf{u} \cdot (\mathbf{w} - \mathbf{w_h}) - \int_\Omega \left(\frac{1}{\rho}\nabla \cdot \sigma(\mathbf{u}_h) + \theta\rho\mathbf{u}\right) \cdot (\mathbf{w} - \mathbf{w_h}) \tag{2.11.51}$$

We will use the previous result in the upper bound on the energy norm of the error. Let $\mathbf{w} = \mathbf{u} - \mathbf{u_h}$. Then

$$\|\mathbf{u} - \mathbf{u_h}\|_\alpha^2 = a(\mathbf{u} - \mathbf{u_h}, \mathbf{w}) = a(\mathbf{u} - \mathbf{u_h}, \mathbf{w} - \mathbf{w_h}) \tag{2.11.52}$$

where the last equality follows from Galerkin orthogonality. Breaking the previous expression into element-wise quantities, and using equation 2.11.51, we obtain

$$\begin{aligned}
\|\mathbf{u} - \mathbf{u_h}\|_\alpha^2 &= \sum_K a(\mathbf{u} - \mathbf{u_h}, \mathbf{w} - \mathbf{w_h}) \qquad (2.11.53)\\
&= \sum_K a(\mathbf{u}, \mathbf{w} - \mathbf{w_h}) - \sum_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot (\mathbf{w} - \mathbf{w_h})\\
&- \sum_K \int_K \theta\mathbf{u} \cdot (\mathbf{w} - \mathbf{w_h}) + \sum_K \int_K \left(\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\mathbf{u}\right) \cdot (\mathbf{w} - \mathbf{w_h})\\
&= \sum_K \int_K \left(\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\mathbf{u}\right) \cdot \mathbf{w} - \mathbf{w_h} - \sum_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot (\mathbf{w} - \mathbf{w_h})\\
&\leq \sum_K \mu_K \|\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\mathbf{u}\|_{L^2(K)} \mu_K^{-1} \|\mathbf{w} - \mathbf{w_h}\|_{L^2(K)}\\
&+ \sum_e \mu_e^{\frac{1}{2}} \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)} \mu_e^{\frac{1}{2}} \|\mathbf{w} - \mathbf{w_h}\|_{L^2(e)}\\
&\leq \left[\sum_K \mu_K^2 \|\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\mathbf{u}\|_{L^2(K)}^2 + \sum_e \mu_e \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)}^2\right]^{\frac{1}{2}}\\
&\times \left[\sum_K \mu_K^{-2} \|\mathbf{w} - \mathbf{w_h}\|_{L^2(K)}^2 + \sum_e \mu_e^{-1} \|\mathbf{w} - \mathbf{w_h}\|_{L^2(e)}^2\right]^{\frac{1}{2}}
\end{aligned}$$

Equation 2.16 in Bernardi's paper shows that

$$\left[\sum_K \mu_K^{-2} \|\mathbf{w} - \mathbf{w_h}\|_{L^2(K)}^2 + \sum_e \mu_e^{-1} \|\mathbf{w} - \mathbf{w_h}\|_{L^2(e)}^2\right]^{\frac{1}{2}} \leq c\|\mathbf{w}\|_\alpha \tag{2.11.54}$$

With this result, we have

$$\|\mathbf{u} - \mathbf{u_h}\|_\alpha \leq c \left[\sum_K \mu_K^2 \|\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u_h}) + \theta\rho\mathbf{u}\|_{L^2(K)}^2 + \sum_e \mu_e \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)}^2\right]^{\frac{1}{2}} \tag{2.11.55}$$

which is the desired upper bound. We note that we would also obtain higher order terms in the above expression by adding and subtracting terms of the kind $\int_K \theta_h \mathbf{u_h} dx$, but the same argument could be made as before.

2.11.5. *Explicit Estimator Summary*

Summarizing, the implementation of the explicit error estimator involves the following steps. These steps have to be carried out for each eigenvalue separately.

1. Renormalize the eigenvectors as in section 2.11.4, equation 2.11.38.
2. Loop through all elements in the mesh. Compute the surface flux residuals for each face. Share that residual vector at each surface Gauss point with neighboring elements to determine the stress jump 2.11.14. Integrate over all faces (by summing at surface Gauss points) to determine R_{flux} (eq 2.11.12).
3. Loop through all elements in the mesh. At each interior Gauss point of each element,
   a) Compute the interior residual,
   $$a_1 = |A_1(u_h) + \lambda_h A_2(u_h)|$$
   b) Compute the integrand,
   $$(a_1 + R_{flux})^2$$
   Note that R_{flux} is a constant over the element.
   c) Sum at Gauss points to obtain the element contribution,
   $$\bar{\rho}^2 = \int_{\Omega_e} (a_1 + R_{flux})^2 d\Omega_e \sim \sum_{i}^{N_{Gauss}} w_i (a_1(x_i) + R_{flux})^2$$
4. Compute the global contribution to the error. For elements with linear shape functions, this may be written,
   $$\frac{|\lambda - \lambda_h|}{\lambda} \leq c \left(\sum_{e=1}^{N_e} (C_{e,0} h_e^2 \bar{\rho})^2 \right)^{\frac{1}{2}} . \qquad (2.11.56)$$
   Where (as shown in Section 2.11.3, equation 2.11.28),
   $$C_{e,0}^2 = \frac{\rho}{(\Lambda + \mu)^2}$$
   and ρ, Λ and μ are the material density and the Lamé constants respectively. The more general expression for elements of order p is,
   $$\frac{|\lambda - \lambda_h|}{\lambda^{(p+1)/2}} \leq c \left(\sum_{e=1}^{N_e} (C_{e,p-1} h_e^{(p+1)} \bar{\rho})^2 \right)^{\frac{1}{2}} . \qquad (2.11.57)$$
   We note that although the constant, c, in equation 2.11.56 is unknown, it is estimated to be of order 1. The constant depends on the details of the mesh, and in particular on the minimum angle in the elements.

2.11.6. Approach II - quantity of interest estimator

In,[108] an error estimator is derived for the elasticity equation, using the eigenvalues as the quantity of interest. The estimate is of the form

$$\eta^{\lambda}_{low} = -\eta^{2}_{upp} \tag{2.11.58}$$
$$\eta^{\lambda}_{upp} = -\eta^{2}_{low} \tag{2.11.59}$$

where η^{λ}_{low} is a lower bound on $\lambda - \lambda_h$, and η^{λ}_{upp} is an upper bound on $\lambda - \lambda_h$. Note that both quantities are necessarily negative,[10] since the computed eigenvalues are always larger than the exact ones.

The quantities η_{upp} and η_{low} are computed using the *element residual method*. This method involves solving a small linear system on each element to obtain an error representation for that element, and then the element contributions are accumulated to obtain the total errors. The element linear system is

$$-B(\Phi_K, v) = R(v,0) + \int_{\partial K} g_{\gamma,K} v ds \quad \forall v \in W_K \tag{2.11.60}$$

or

$$K_b a = f \tag{2.11.61}$$

where a is the vector of coefficients that represent the function Φ_K. In other words, $\Phi_K = \sum_{i=1}^{Nshape_{bubble}} a_i N_i$, where N_i is the i^{th} bubble shape function. The left-hand side K_b is the element stiffness matrix, but evaluated using bubble functions rather than the standard element shape functions. This is necessary since the standard element stiffness matrix is singular and thus equation 2.11.61 would otherwise not be solvable. The right-hand side consists of two terms, an interior residual term for the interior of the element, and a stress jump term on the element boundary. This is similar to the interior and boundary residual terms that were encountered in the explicit error estimator, though the exact formulas for these terms are somewhat different. The first term is

$$R(v,0) = B(u_h, v) - \lambda_h M(u_h, v) \tag{2.11.62}$$

Equation 2.11.62 can be most efficiently evaluated using the following method.[111] We evaluate the first term first.

$$B(u_h, v) = \int_K B^T_{bubble} \sigma(x) dx \tag{2.11.63}$$

where B^T_{bubble} is the standard 'B' matrix, or the matrix of derivatives of the element shape functions, except that it is using the bubble shape functions rather than the standard shape functions. Note that the result of equation 2.11.63 is a vector of length $3xNshape_{bubble}$, where $Nshape_{bubble}$ is the number of bubble shape functions. We note that the routine ForceFromStress in IsoSolid.C already performs the computation needed for equation 2.11.63, with the only change being the use of the matrix B^T_{bubble} rather than the standard B^T, and thus this code could be re-used.

The second term can be evaluated in a similar way.

$$M(u_h, v) = \int_K u_h(x) v(x) dx \tag{2.11.64}$$

[10]for consistent mass only.

Note that $u_h(x)$ is a known function. This term is also a vector of length $3xNshape_{bubble}$. The three entries corresponding to the i^{th} bubble shape function are as follows

$$\int_K u_{1h}(x)\phi_i(x)dx, \quad \int_K u_{2h}(x)\phi_i(x)dx, \quad \int_K u_{3h}(x)\phi_i(x)dx.$$

where u_{1h}, u_{2h}, and u_{3h} are the x, y, and z components of u_h, and ϕ_i is the i^{th} bubble shape function.

The boundary term consists of the following. $g_{\gamma,K}$ is the traction on the element boundary, or

$$\int_{\partial K} g_{\gamma,K} v ds \quad = \quad \int_{\partial K} \left[\sigma_{ij} n_j\right] v ds \tag{2.11.65}$$

where $\left[\sigma_{ij} n_j\right]$ denotes the *averaged* stress on the element faces. For two adjacent elements, element 'a' and element 'b', it is the average of their stress traction vectors.

$$\left[\sigma_{ij} n_j\right] = \frac{1}{2}\left(\sigma^a_{ij} n_j + \sigma^b_{ij} n_j\right) \tag{2.11.66}$$

Again, the test (shape) function in this case, 'v' is the bubble function rather than the standard element shape function. We note that the boundary integral term in equation 2.11.60 and equation 2.11.65 is over all faces of the element in question. Thus, if the implementation of this term proceeds one face at a time, then there will be a nodal summation step to get the complete right-hand side vector corresponding to the boundary integral term. We could also write this term as

$$\int_{\partial K} g_{\gamma,K} v ds = \sum_{i=1}^{N_{faces}} \int_{\partial K_i} g_{\gamma,K} v ds \tag{2.11.67}$$

where ∂K_i is the i^{th} face of element 'K'. Note that the test functions, v become the element shape functions when restricted to an element. Thus, for a given element bubble shape function ϕ_{bubble}, and a given face, we can write the previous equation as

$$\int_{\partial K_i} g_{\gamma,K} \phi_{bubble} ds \tag{2.11.68}$$

Note that $g_{\gamma,K}$ is a 3-vector, and so for a given bubble shape function, and a given face, $\int_{\partial K_i} g_{\gamma,K} \phi_{bubble} ds$ is also a 3-vector. We then take this 3-vector and project it into the element right-hand side. After looping through all faces and all bubble shape functions, we end up with a vector that is of length $3 * Nshape_{bubble}$.

Once the linear systems 2.11.61 are solved on each element, the upper bound, η_{up} from equation 2.11.59 can be computed as follows

$$\eta_{upp} = \sqrt{\sum_K B(\Phi_K, \Phi_K)} \tag{2.11.69}$$

This equation can also be written as follows. If we represent the function Φ_K as a summation of coefficients multiplied by the bubble shape functions,

$$\Phi_K = \sum_{i=1}^{Nshape_{bubble}} a_i N_i \tag{2.11.70}$$

then

$$\eta_{upp} = \sqrt{\sum_K B(\Phi_K, \Phi_K)} = \sqrt{\sum_K a^T K_b a} \tag{2.11.71}$$

Finally, using equation 2.11.59, we have an upper bound on the error in the eigenvalue.

A lower bound on the error in the eigenvalue can also be computed. This is described in detail in,[108] and we summarize here.

First, we define a function $\chi \in V$, which we will define shortly. Once the function χ is defined, the lower bound can be computed as follows

$$\eta_{low} = \frac{|R_p(\chi, 0)|}{\sqrt{B(\chi, \chi)}} \tag{2.11.72}$$

The quantities in both the numerator and denominator can be computed by looping through all elements and computing the corresponding element-wise quantities (using equation 2.11.62), and then summing globally.

Summarizing, to implement the quantity of interest approach for eigenvalue error estimation, we have the following steps. These must be carried out for each eigenvalue.

1. Loop over all elements. Construct the bubble stiffness matrix, K_b in equation 2.11.61, in the same way that standard element stiffness matrix is constructed, but using the bubble shape functions.
2. Loop over all elements. Construct the right-hand side of equation 2.11.61. This consists of the interior part, equation 2.11.62, and the boundary part, equation 2.11.65.
3. Loop over all elements and solve the linear systems 2.11.61, to obtain the error functions Φ_K.
4. Compute the upper bound on the error in the eigenvalue using equation 2.11.71.
5. Compute the lower bound on the error in the eigenvalue using equation 2.11.72.

2.12. Nonlinear Distributed Damping using Modal Masing Formulation

This provides a method for implementing nonlinear distributed damping into a subsystem with a nonlinear transient solution. This is a method developed to model the nonlinear damping response of a subsystem. It implements the damping in a nonlinear manner with the use of an internal force term. The damping is modeled by an Iwan model and distributed to the subsystem by a modal expansion. This method augments the internal force vector through a modal Masing formulation.

Subsystem Distributed Damping Formulation with Iwan Model. Given a system that contains a subsystem exhibiting nonlinear damping behavior, the equation of motion for the subsystem, denoted by B, can be written in typical finite element form as:

$$\mathbf{M_B u_B} + \mathbf{C_B u_B} + \mathbf{K_B u_B} = \mathbf{F_B} + \mathbf{F_B^J}, \tag{2.12.1}$$

where $\mathbf{M_B}$, $\mathbf{C_B}$, $\mathbf{K_B}$ are the mass, damping, and stiffness matrices of the subsystem B derived from a low-load response, $\mathbf{u_B}$ is the discretized nodal displacements, a superposed dot denotes time differentiation, $\mathbf{F_B}$ represents the external forces, and $\mathbf{F_B^J}$ is a distribution of internal nonlinear damping forces to be discussed later.

A modal expansion is used to distribute the damping to the subsystem; therefore, the problem is formulated in modal coordinates. Let $\mathbf{\Phi}_B$ be the matrix whose columns are the eigenvectors of the $(\mathbf{M_B}, \mathbf{K_B})$ system and define modal coordinates in subsystem body B

$$\mathbf{u_B} = \mathbf{\Phi_B q_B}, \tag{2.12.2}$$

where q_B is a vector of modal coordinates. It is assumed that the eigenvectors are mass normalized. Pre-multiplying Eq. (2.12.1), by $\mathbf{\Phi}_B^T$, yields

$$[\mathbf{\Phi}_B^T\mathbf{M_B\Phi_B}]\mathbf{q_B} + [\mathbf{\Phi_B^T C_B \Phi_B}]\mathbf{q_B} + [\mathbf{\Phi_B^T K_B \Phi_B}]\mathbf{q_B} = \mathbf{\Phi_B^T F_B} + \mathbf{\Phi_B^T F_B^J}, \tag{2.12.3}$$

To derive a nonlinear distributed damping system, the force term $\mathbf{\Phi}_B^T\mathbf{F_B^J}$ is modeled by a four parameter Iwan model:[118,119]

$$\mathbf{\Phi}_B^T\mathbf{F_B^J} = \mathbf{F_{\Phi B}^J} = -\int_{\mathbf{0}}^{\infty} \mathbf{diag}(\rho(\phi))[\mathbf{q(t)} - \boldsymbol{\beta}(\mathbf{t}, \phi)]\mathbf{d}\phi, \tag{2.12.4}$$

where ρ is the population density of Jenkins elements of strength ϕ (not to be confused with the eigenvectors), and $\beta(t, \phi)$ is the current *modal* displacements of the sliders in the Iwan model.[119] This force term is solved in a discretized form with the integration from zero to ϕ_{max}:[119]

$$\mathbf{F_{\Phi B}^J} = -\sum_{\mathbf{m=1}}^{\mathbf{N}} \mathbf{F_m(t)} - \mathbf{F}_\delta\mathbf{(t)} + \mathbf{K_0 q(t)}, \tag{2.12.5}$$

where the integral in Eq. (2.12.4) is numerically integrated with intervals, $\Delta\phi_m$, such that,

$$\sum_{m=1}^{N} \Delta\phi_m = \phi_{max}, \tag{2.12.6}$$

with ϕ_m being the midpoint of each interval $\Delta\phi_m$ in the numerical integration. The, term, $F_m(t)$ is derived as:[119]

$$F_m(t) = \begin{cases} R\frac{\phi_{r,m}^{2+\chi} - \phi_{l,m}^{2+\chi}}{2+\chi}\text{sgn}\,[q(t) - \boldsymbol{\beta}(t)] & \text{if } \parallel q(t) - \boldsymbol{\beta}(t) \parallel = \phi_m \\ R\frac{\phi_{r,m}^{1+\chi} - \phi_{l,m}^{1+\chi}}{1+\chi}\,[q(t) - \boldsymbol{\beta}(t)] & \text{if } \parallel q(t) - \boldsymbol{\beta}(t) \parallel < \phi_m \end{cases} \tag{2.12.7}$$

with $\phi_{r,m}$ and $\phi_{l,m}$ being the right and left side of each sub-interval, $\Delta\phi_m$, and R and χ are a parameters of the Iwan model. The term, $F_\delta(t)$, is found:[119]

$$F_\delta(t) = \begin{cases} S[q(t) - \boldsymbol{\beta}(t)] & \text{if } [q(t) - \boldsymbol{\beta}(t)] < \phi_m \\ S\phi_{max}\text{sgn}[q(t) - \boldsymbol{\beta}(t)] & \text{otherwise} \end{cases} \tag{2.12.8}$$

where S is an Iwan parameter. The final term, $K_0 q(t)$ in Eq. (2.12.5), is an elastic restoring force in the Iwan model that is included in the $F_m(t)$ term, but also in the overall subsystem stiffness matrix, $\mathbf{K_B}$. Therefore, it needs to be subtracted, so as not to include the elastic force twice. The term K_0 is the stiffness of the Iwan model under small applied loads (where slip is infinitesimal). This is calculated from the Iwan parameters as

$$K_0 = \frac{R\phi_{max}^{\chi+1}}{\chi+1} + S = \frac{R\phi_{max}^{\chi+1}}{\chi+1}(1+\beta) \tag{2.12.9}$$

Transferring to physical degrees of freedom provides the following for the equation of motion:

$$\mathbf{M_B u_B} + \mathbf{C_B u_B} + \mathbf{K_B u_B} = \mathbf{F_B} + \mathbf{\Phi_B^{-T} F_{\Phi B}^J} \tag{2.12.10}$$

To avoid the possibility of an ill-conditioned and difficult pseudo-inversions, recognize that $\mathbf{M_B \Phi_B} = \mathbf{\Phi_B^{-T}}$, yielding:

$$\mathbf{M_B u_B} + \mathbf{C_B u_B} + \mathbf{K_B u_B} = \mathbf{F_B} + \mathbf{M_B \Phi_B F^J_{\Phi B}} \tag{2.12.11}$$

Given the above EOM, a typical nonlinear analysis can be performed, recognizing that the force term $\mathbf{M_B \Phi_B F^J_{\Phi B}}$ is a function of the displacement. However, care must be exercised in the implementation, as the modal displacement will need to be passed to the Iwan function for updating internal forces.

2.12.1. *Subsystem Distributed Damping with a Linear Damper*

It is possible to derive the same basic formulation as above, but for a linear damping. This provides a check into the formulation as the results should be the same as a model with a modal damping parameter.

The only required change from the above derivation is in the nonlinear internal force term, $\mathbf{F^J_{\Phi B}}$. This term will need to be appropriate for a viscous damper; thus, a function of the modal velocity. A formulation can be found as the following:

$$\mathbf{F^J_{\Phi B}} = \mathbf{F^J_{\Phi B i}} = -\mathbf{2}\varsigma_\mathbf{i}\omega_\mathbf{i}\mathbf{q_i}, \tag{2.12.12}$$

where subscript i represents the mode, ς_i is the damping ratio for mode i, ω_i is the frequency for mode i, and $\dot{\alpha}$ is the modal velocity. Here I am trying to see how many subscripts I can possibly add.

Reduced Model. To reduce computational demand, a reduced set of eigenvectors ($\mathbf{\Phi}_B^R$) can be calculated for the subsystem and used in place of the total subsystem eigenvector, $\mathbf{\Phi}_B$.

Full System Model. Implementation of the full system with nodal degrees of freedom, u, is accomplished with a typical projection matrix, P, from the full system to the subsystem.

$$u_B = Pu \tag{2.12.13}$$

The EOM simplifies to

$$\mathbf{Mu} + \mathbf{Cu} + \mathbf{Ku} = \mathbf{F} + \mathbf{P^T M_B \Phi^R_B F^J_{\Phi B}} \tag{2.12.14}$$

2.13. Shock Response Spectra

Theory for computation of a shock response spectrum may be found in the papers by Smallwood.[123,124] The theory is not repeated here. Many analysts use the MATLAB scripts developed by Smallwood to perform this analysis. MATLAB provides a nice, interactive environment for this analysis once the time integration has been performed in **Sierra/SD**. **Sierra/SD** performs identical calculations.

2.14. Superposition for superelement recovery

A Craig-Bampton reduction generates a transformation matrix consisting of a combined set of fixed interface and constraint modes. These modes may be stored in an **Exodus** file. We call this "`se-base.exo`". A netcdf file containing the reduced order model, "`se.ncf`" is also created at this time. Subsequently, this reduced model is inserted into a residual model for superelement analysis, say a transient analysis. That analysis outputs the standard **Exodus** results, "`mesh-out.exo`" and results on the netcdf file, "`se-out.ncf`". The point is to recover the response on the original interior degrees of freedom of the superelement.

The transient response on the interior degrees of freedom is,

$$u_k(t_n) = \sum_i^{nmodes} q_i(t_n)\phi_{ik} + \sum_j^{nconstraint} w_j(t_n)\psi_{jk} \tag{2.14.1}$$

where,

$u_k(t_n)$	=	is the displacement at interior dof k
t_n	=	is the time step
q_i	=	is the amplitude of a generalized dof for mode i
ϕ_{ik}	=	is the fixed interface mode i at dof k
w_j	=	is the amplitude of interface dof j
ψ_{jk}	=	is the constraint mode j at dof k

The amplitudes q_i and w_j are found in "`se-out.ncf`", while the mode shapes, ϕ_{ik} and ψ_{jk} are found in "`se-base.exo`". The "superposition" solution combines these results and writes a new output file containing the results.

2.15. Coupled Electro-Mechanical Physics

The finite element method was used to derive the coupled equations of motion underlying the coupled electro-mechanical physics package. The theoretical details are documented in the referenced Sand report.[28]

2.16. High Cycle Fatigue and Damage

The theory for fatigue analysis is developed from "Random Vibrations, theory and practice".[133] From equation WPO:10.58, the wideband damage is a correction to the narrowband damage.

$$D = \lambda D_{NB}$$

For Narrow Band damage, λ is 1, but other damage models (such as that proposed by Wirsching and Light), use λ as a modifier to adapt Narrow Band damage to Wide Band processes. Narrow Band damage is defined as:

$$D_{NB} = \frac{\nu_o^+\tau}{A}(\sqrt{2}\sigma_s F_{SS})^m \Gamma\left(\frac{m}{2}+1\right) \tag{2.16.1}$$

Note that this equation assumes that the value of A used in the material's S-N curve is based on peak stress. If it is calculated based on stress range, narrowband damage is instead express as:

$$D_{NB} = \frac{\nu_o^+\tau}{A}(2\sqrt{2}\sigma_s F_{SS})^m \Gamma\left(\frac{m}{2}+1\right)$$

Both practices are common in material data. We use the definition in equation (2.16.1) in this work. The Fatigue Stress Scale (F_{SS}) is a parameter to convert stress units from the simulation's unit system to the unit system of the material. Here,

m	negative of slope of S-N curve, default=3.
ν_o^+	rate of crossings
τ	is the exposure time (or duration)
A	strength coefficient of material
σ_s	RMS stress
F_{SS}	Fatigue Stress Scale

The rate of zero crossings may be computed as, $\nu_o^+ = \sqrt{M_2/M_0}$ from equation WPO:6.24. Here M_j is a stress moment, which is readily computed in **Sierra/SD**. Within the modal random vibration module, RMS stress moments are computed. These are related to the stress moments.

$$M_0 = (V_{RMS}/(2\pi))^2, \quad M_2 = \left(V_{RMS2}/(2\pi)^2\right)^2, \quad M_4 = \left(V_{RMS4}/(2\pi)^3\right)^2.$$

Therefore,

$$\nu_o^+ = V_{RMS2}/(2\pi \cdot V_{RMS})$$

The RMS stress is the primary output of the modal random vibration analysis.

Material and random loads must be provided as user input, and the other quantities are readily determined from the analysis. D_{NB} is well-defined. There are various methods of computing the correction factor λ. A few are outlined below.

Sensitivity to Stress The narrow band damage parameter (eq. 2.16.1), is nonlinear in the stress. Effectively, $D_{nb} \propto \sigma^m$. Thus, doubling the stress when $m = 3$ results in an 8 fold increase in damage rate. However, m may be as high as 14 for many real materials. Doubling the stress increases the damage rate by $2^{14} = 16384$.

2.16.1. Competing Damage Models

Wirsching and Light: applies equation WPO:10.60. This is described in [132]. Compute:

$$\begin{aligned} a(m) &= 0.926 - 0.033\,m & \alpha &= \frac{\nu_o^+}{\nu_p} \\ b(m) &= 1.587\,m - 2.323 & \epsilon &= \sqrt{1-\alpha^2} \\ \nu_p &= \sqrt{M_4/M_2} & \lambda &= a(m) + [1-a(m)](1-\epsilon)^{b(m)}. \end{aligned}$$

Ortiz, Chen and Perng: applies equation WPO:10.62.

$$k = 2/m, \quad \beta = \sqrt{\frac{M_2 M_k}{M_0 M_{k+2}}}, \quad \lambda = \beta/\alpha.$$

Lutes and Larsen: applies equation WPO:10.68.

$$\lambda = \frac{(M_k)^{1/k}}{\nu_o^+} \tag{2.16.2}$$

Steinberg: The Steinberg approach for calculating fatigue can be useful as a simple check of fatigue failure. The Steinberg approach uses the assumption that the RMS of the stress is representative of a 1σ event, and that the peak stress of any given cycle is a random value. As such, it calculates a cumulative damage as the summation:

$$n_i = \nu_o^+ \,\tau \,\mathrm{erf}\left(\frac{i}{\sqrt{2}}\right), \quad N_i = \frac{A}{(i\,\sigma_s)^m}, \quad D = \sum_{i=1}^{\infty} \frac{n_i}{N_i}. \tag{2.16.3}$$

The Steinberg approach is ideally suited to loads that operate at one frequency, or a narrowband of frequencies. There is also the problem of choosing an acceptable number of terms to calculate. Eventually, the magnitude of the stress becomes great enough to cause low-cycle failure, and the equations for high-cycle fatigue breakdown. To avoid this, and to make the calculation inexpensive, it is common to limit ourselves to only the first 3 terms of the series.

Dirlik: This method is described in Mrsnik̆ ([103]). Define,

$$\begin{aligned}
x_m &= \frac{M_1}{M_0}\sqrt{\frac{M_2}{M_4}} & R_d &= \frac{\alpha_2 - x_m - G_1^2}{1 - \alpha_2 - G_1 + G_1^2} \\
Z &= \frac{s}{\sqrt{M_o}} & G_2 &= \frac{1 - \alpha_2^- G_1 + G_1^2}{1 - R_d} \\
\alpha_2 &= \frac{M_2}{\sqrt{M_0 M_4}} & G_3 &= 1 - G_1 - G2 \\
G_1 &= \frac{2(x_m - \alpha_2^2)}{1 + \alpha_2^2} & Q &= \frac{1.25(\alpha_2 - G_3 - G_2 R_d)}{G_1}
\end{aligned}$$

Then,

$$\bar{D} = C^{-1}\nu_p M_o^{\frac{k}{2}}\left[G_1 Q^k \Gamma(1+k) + (\sqrt{2})^k \Gamma\left(1 + \frac{k}{2}\right)(G_2|R_d|^k + G_3)\right]$$

Typically, these correction methods provide similar results. The Ortiz and Lutes methods require the moment M_k, which could vary by material block, and is expensive to compute. The Wirsching method is somewhat simpler, and will be followed as a first development.

3. ACOUSTICS SOLUTION METHODS

In this section, we discuss the partial differential equations behind the acoustic formulations used in Sierra Structural Dynamics. We also discuss discretization procedures, mesh matching conditions on the wet surface, exterior boundary conditions, and various loading scenarios including scattering. As the first step, we show how to derive the acoustic wave equation from the fluid dynamics equations. This will then lead into a discussion of the coupled equations of motion.

3.1. Derivation of Acoustic Wave Equation

Under certain assumptions, fluid motion can be approximated as small-amplitude linear wave propagation. We give a short background on the assumptions that go into the derivation of the acoustic wave equation. In the most general case the fluid motion is governed by the compressible Navier Stokes equations. In the case of small-amplitude wave propagation, viscosity is typically neglected, and a polytropic relationship is assumed between pressure and density in the fluid. In this case, the fluid motion is described by the nonlinear Euler equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) = q \tag{3.1.1}$$

$$\rho \frac{\partial v}{\partial t} + \rho v \cdot \nabla v + \nabla p = f \tag{3.1.2}$$

where equations (3.1.1) and (3.1.2) represent mass and momentum conservation, respectively, and p, ρ and v represent the fluid pressure, density, and velocity. The right-hand side terms consist of mass injection q (density per unit time) and body force f (force per unit volume). Note that these are both nonlinear equations, and thus allow for both fluid convection and wave propagation. In addition, we note that a nonlinear pressure-density relation exists for a given fluid

$$p = p(\rho). \tag{3.1.3}$$

Equations (3.1.1), (3.1.2), and (3.1.3) are nonlinear, but they can be linearized under the assumptions of small fluid motion. First, we decompose the field variables into ambient (background) values plus small perturbations:

$$\begin{aligned} p &= p_0 + \delta p \\ \rho &= \rho_0 + \delta \rho \\ v &= 0 + \delta v. \end{aligned} \tag{3.1.4}$$

We say that the perturbations δp, $\delta \rho$, and δv are all $O(\delta)$. Since the background velocity is equal to zero, v itself is also $O(\delta)$.

Next, we insert equations (3.1.4) into equations (3.1.1), (3.1.2), and (3.1.3), and in keeping with the linearization process we neglect terms that involve products of perturbations. This yields the following:

$$\begin{aligned} q &= \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) \\ &= \frac{\partial}{\partial t}(\rho_0 + \delta\rho) + \nabla \cdot ((\rho_0 + \delta\rho)\delta v) \\ &= \underbrace{\frac{\partial \rho_0}{\partial t}}_{=0} + \frac{\partial \delta\rho}{\partial t} + \rho_0 \nabla \cdot \delta v + \underbrace{\delta\rho \nabla \cdot \delta v + \delta v \nabla \cdot \delta\rho}_{=O(\delta^2)} \\ &\approx \frac{\partial \delta\rho}{\partial t} + \rho_0 \nabla \cdot \delta v \end{aligned} \tag{3.1.5}$$

$$\begin{aligned} f &= \rho \frac{\partial v}{\partial t} + \rho v \cdot \nabla v + \nabla p \\ &= (\rho_0 + \delta\rho)\frac{\partial \delta v}{\partial t} + (\rho_0 + \delta\rho)\delta v \cdot \nabla \delta v + \nabla(p_0 + \delta p) \\ &= \rho_0 \frac{\partial \delta v}{\partial t} + \underbrace{\delta\rho \frac{\partial \delta v}{\partial t}}_{=O(\delta^2)} + \underbrace{(\rho_0 + \delta\rho)\delta v \cdot \nabla \delta v}_{=O(\delta^2)} + \underbrace{\nabla p_0}_{=0} + \nabla \delta p \\ &\approx \rho_0 \frac{\partial \delta v}{\partial t} + \nabla \delta p \end{aligned} \tag{3.1.6}$$

$$p(\rho) = p_0 + \frac{\partial p}{\partial \rho}(\rho_0)\delta\rho + \dots, \tag{3.1.7}$$

where we have linearized the pressure-density relation (3.1.7) by taking only the first term in a Taylor series expansion. This implies that to first order,

$$\delta p = \frac{\partial p}{\partial \rho}(\rho_0)\delta\rho. \tag{3.1.8}$$

It is useful to make the definition

$$c^2 \equiv \frac{\partial p}{\partial \rho}(\rho_0). \tag{3.1.9}$$

That c is in fact the speed of acoustic wave propagation follows below.

Combining equations (3.1.5), (3.1.6), and (3.1.8), we arrive at the linear Euler equations

$$\begin{aligned} \frac{1}{c^2}\frac{\partial \delta p}{\partial t} + \rho_0 \nabla \cdot \delta v &= q \\ \rho_0 \frac{\partial \delta v}{\partial t} + \nabla \delta p &= f \end{aligned} \tag{3.1.10}$$

Taking the time derivative of the first of equations (3.1.10), and the divergence of the second of equations (3.1.10), we arrive at the linear wave equation

$$\begin{aligned} \frac{\partial q}{\partial t} - \nabla \cdot f &= \frac{\partial}{\partial t}\left(\frac{1}{c^2}\frac{\partial \delta p}{\partial t} + \rho_0 \nabla \cdot \delta v\right) - \nabla \cdot \left(\rho_0 \frac{\partial \delta v}{\partial t} + \nabla \delta p\right) \\ &= \frac{1}{c^2}\frac{\partial^2 \delta p}{\partial t^2} + \underbrace{\rho_0 \frac{\partial}{\partial t}\nabla \cdot \delta v - \rho_0 \nabla \cdot \frac{\partial \delta v}{\partial t}}_{=0} - \Delta \delta p \\ &= \frac{1}{c^2}\frac{\partial^2 \delta p}{\partial t^2} - \Delta \delta p. \end{aligned} \tag{3.1.11}$$

It is often useful to employ a formulation of the acoustic wave equation based on a velocity potential ψ rather than the acoustic pressure δp. This approach can simplify the formulation of problems in structural acoustics, and can also yield symmetric rather than unsymmetric linear systems. There are a variety of definitions that can be employed. As the name velocity potential implies, among the most well-known choices is:

$$\delta v = \nabla \psi. \tag{3.1.12}$$

Let us consider the implications of this choice vis-a-vis equation (3.1.10). Plugging equation (3.1.12) into equation (3.1.10) and reordering derivatives, we obtain

$$\begin{aligned} 0 &= \rho_0 \frac{\partial \nabla \psi}{\partial t} + \nabla \delta p \\ &= \nabla \left(\rho_0 \frac{\partial \psi}{\delta t} + \delta p \right) \end{aligned} \tag{3.1.13}$$

Therefore, we have

$$\delta p = -\rho_0 \frac{\partial \psi}{\partial t}. \tag{3.1.14}$$

With the definition in equation (3.1.14), time integration of the velocity potential ψ is necessary to recover the physical pressure. The fluid density ρ_0 must also be available to perform this conversion, which may create some bookkeeping headaches. An alternative choice for the velocity potential is to make the definition

$$\delta p = \frac{\partial \psi}{\partial t}. \tag{3.1.15}$$

In this case, it follows from equation (3.1.10) that

$$\nabla \psi = -\rho_0 \delta v, \tag{3.1.16}$$

i.e., we have removed ρ_0 from the relation between pressure and the velocity potential but made it appear in relating the velocity potential to $\nabla \psi$.

In either case, a derivation similar to that employed above for the pressure-based wave equation can be used to show that the velocity potential also satisfies a wave equation[110]

$$\frac{1}{c^2} \frac{\partial^2 \psi}{\partial t^2} - \Delta \psi = 0. \tag{3.1.17}$$

We use this fact later on for coupled system of equations.

In the following sections, we find it convenient to drop the δs and write v,p to indicate the perturbations δv,δp.

3.2. Coupled Structural Acoustics

In this subsection, we present the coupling of the acoustic wave equation derived in the previous section with the structural dynamic equations of an elastic structure. Excellent review articles[72,57] have been written on the subject. In this section we focus on the details relevant to the **Sierra/SD** implementation.

3.2.1. Discussion of Matching vs Non-Matching Meshes on Wet Surface

Having the same mesh density in the acoustic fluid and solid may be inefficient, since the two domains typically require significantly different mesh densities to achieve a given level of discretization accuracy. It is also impractical in many applications since the mesh generation process may be performed separately for the two domains. Generating conforming meshes on the wet interface may be difficult, if not impossible, even given the most sophisticated mesh generation software. Illustrative examples include the hull of a ship, or the skin of an aircraft. In these cases, the structural and fluid meshes are typically created independently, and have different mesh density requirements. Joining them into a single, monolithic mesh is often impractical.

Although methods for joining dissimilar meshes are well-known in structural mechanics,[6,50,91,113] few papers exist in the area of dissimilar structural acoustic meshes. Mandel[99] considered parallel domain decomposition techniques for structural acoustics in the frequency domain, on mismatched fluid/solid meshes. Nonconforming discretizations on the wet interface were handled by duplicating acoustic and structural degrees of freedom on either side of the wet interface, and imposing coupling equations that enforce continuity of pressure and displacement. The duplicated degrees of freedom were then included in a dual-primal, parallel domain decomposition strategy. Only two-dimensional, frequency-domain problems were considered. Flemisch et al.[65] studied both fluid-fluid and structure-fluid coupling on mismatched meshes. For fluid-fluid coupling, a mortar approach was taken, whereas for structural acoustic coupling, the coupling matrices were assembled in normal fashion and used across the wet interface to coupled the fluid-solid responses. Only time-domain, serial solutions were considered.

Several recent references considered a displacement-based acoustic formulation, which was then coupled to an elasticity formulation on mismatched fluid/solid meshes. Alonzo[3] used an adaptive method with error estimation to refine the fluid/solid meshes accordingly. The error estimator demanded different mesh densities on the fluid and solid interface, as expected. Bermudez[21] also considered a displacement-based acoustic formulation, but used an integral constraint on the wet interface, along with a static condensation procedure to eliminate the acoustic degrees of freedom. In both of the preceding references, Raviart-Thomas elements were needed to avoid spurious modes in the fluid. These modes would have been automatically eliminated with the use of a potential formulation in the fluid.

In the following sections, a new technique is presented for structural acoustic analysis in the case of nonconforming fluid/solid interface meshes. We first construct a simple method for coupling mismatched fluid/fluid meshes, based on a set of linear constraint equations. Using static condensation, we show how these constraint equations can be eliminated from the final system of equations. We then demonstrate that the same approach can be taken to couple mismatched fluid/solid meshes, provided that the coupling matrices that are typically used for conforming fluid/solid meshes are calculated at a set of nodes with both structural and acoustic degrees of freedom, and that extra ("ghost") degrees of freedom are introduced to couple the structural or acoustic terms to the other side of the interface. With this arrangement, the structural acoustic coupling resembles a conforming method with like degrees of freedom linked across the interface via MPC equations. Then the conforming structure to acoustic coupling operators ensure a weak continuity of particle velocity and stress between the structural degrees of freedom and collocated acoustic degrees of freedom on the shared side of the interface. Note either the structural degrees of freedom can be ghosted to the acoustic side of the interface or the acoustic degrees of freedom can be ghosted to the structural side of the interface. Either arrangement may be more appropriate depending on the mesh density of the two regions.

In the case that the fluid/solid meshes are conforming, our approach reduces to standard methods for conformal structural acoustic coupling.

3.2.2. *The Coupled Equations and Their Discretizations*

In this section, we review the governing equations of acoustics and structural acoustics, along with their corresponding weak formulations, and then we present our approach for the nonconforming discretization. We begin with the case when all meshes are conforming, and then we extend this to the nonconforming case.

3.2.2.1. The Sierra/SD Velocity Potential Formulation

There are several common formulations for acoustics and structural acoustics. Some details are outlined briefly here. Table 3-1 summarizes the formulations used in **Sierra/SD**.

Problem Space	**Formulation**
Acoustics Source Loading	Velocity Potential: (3.1.15)
Acoustics Enforced Acceleration	Pressure
Structural Acoustics. Loading must be through source loading only.	Negative Velocity Potential: (3.1.15) but multiplied by -1 to maintain symmetry.
Acoustics or structural acoustics with infinite elements	Velocity Potential: (3.1.15). The infinite elements are not symmetric.

Table 3-1. – Acoustic Formulations.

3.2.2.2. Conforming Structural Acoustics

We begin by constructing a weak formulation of the linear acoustic wave equation for conforming meshes. Subsequently, we consider conforming structural acoustics. In this section, we will use the relation (3.1.15) between pressure and the velocity potential ψ, but write ρ_f instead of ρ_0 as the density of the fluid to use ρ_s for the solid density. Surface normal vectors are denoted by $\hat{n}$.

Recall that the linear acoustic wave equation (3.1.17) is given by

$$\frac{1}{c^2}\frac{\partial^2\psi}{\partial t^2} - \Delta\psi = 0. \tag{3.2.1}$$

Note that this implies that we do not include volume (body) forces on the fluid. A weak formulation of equation (3.2.1) can be constructed by multiplying with a test function and integrating by parts. We denote the fluid domain by Ω_f and its boundary by $\partial\Omega = \partial\Omega_n \bigcup \partial\Omega_d$, where the subscripts n and d refer to the portions of the boundary where Neumann and Dirichlet boundary conditions are applied. We also assume that the fluid is initially at rest, i.e. $\psi(x,0) = \partial_t\psi(x,0) = 0$, which is sufficient for most applications.

Denoting by $V_f(\Omega_f)$ the function space for the fluid, the weak formulation can be written as follows. Find the velocity potential $\psi : [0,T] \longrightarrow V_f(\Omega_f)$ such that

$$\frac{1}{c^2}\int_\Omega \frac{\partial^2\psi}{\partial t^2}\phi dx + \int_\Omega \nabla\psi \cdot \nabla\phi dx = \int_{\partial\Omega} \phi\nabla\psi \cdot \hat{n} ds = -\int_{\partial\Omega_n} \rho_f \phi v \cdot \hat{n} ds \tag{3.2.2}$$

$\forall \phi \in V_f(\Omega_f)$, where the fluid velocity v is prescribed on the Neumann portion of the fluid boundary, Ω_n.

Inserting a finite element discretization $\phi(x) = \sum_{i=1}^{N} \phi_i N_i(x)$ into equation (3.2.2) results in the system of equations

$$M\ddot{\psi} + K\psi = f_a, \tag{3.2.3}$$

where N is the vector of shape functions, $M = \int_{\Omega_f} \frac{1}{c^2} N N^T dx$ is the mass matrix, $K = \int_{\Omega_f} \nabla N \cdot \nabla N^T dx$ is the stiffness matrix, and $f_a = -\int_{\partial\Omega_n} \rho_f v \cdot \hat{n} N^T dx$ is the external forcing vector from Neumann boundary conditions.

For structural acoustics, the second order equations of motion for the solid and the wave equation for the fluid are

$$\rho_s \frac{\partial^2 u}{\partial t^2} - \nabla \cdot \sigma = f, \qquad \frac{1}{c^2}\frac{\partial^2 \psi}{\partial t^2} - \Delta\psi = 0. \tag{3.2.4}$$

Here $u = (u_x, u_y, u_z)$ corresponds to the displacement of the structure, σ is the structural stress tensor, ρ_s is the density in the solid, and f denotes the body forces on the solid. Subsequently, the subscripts s and f will refer to solid and fluid, respectively.

The fluid/solid or wet interface is designated by $\partial\Omega_{wet}$. The normal to $\partial\Omega_{wet}$ points from solid into the fluid. In linear acoustics the boundary conditions on $\partial\Omega_{wet}$ are

$$\nabla\psi \cdot \hat{n} = -\rho_f \partial_t u \cdot \hat{n}, \qquad \sigma \cdot \hat{n} = -\frac{\partial\psi}{\partial t}\hat{n}. \tag{3.2.5}$$

These boundary conditions correspond to continuity of velocity and stress at the wet interface respectively.

The weak formulation of the coupled problem is constructed by multiplying the two partial differential equations in equation (3.2.4) by test functions and integrating by parts. Denoting by $V_s(\Omega_s)$ and $V_f(\Omega_f)$ the function spaces for the solid and fluid, respectively, we have the following weak formulation.

Find the mapping $(v, \psi) : [0, T] \to V_s(\Omega_s) \times V_f(\Omega_f)$ such that

$$\begin{aligned}
\int_{\Omega_s} \rho_s \frac{\partial^2 t}{\partial t^2} w dx + \int_{\Omega_s} \sigma : \nabla^s w dx - \int_{\partial\Omega_{wet}} (\sigma \cdot \hat{n}) w ds = \int_{\Omega_s} f w dx + \int_{\partial\Omega_n} (\sigma \cdot \hat{n}) w ds, \\
\frac{1}{c^2}\int_{\Omega_f} \frac{\partial^2 \psi}{\partial t^2} \phi dx + \int_{\Omega_f} \nabla\psi \cdot \nabla\phi dx + \int_{\partial\Omega_{wet}} (\nabla\psi \cdot \hat{n})\phi ds \\
= \int_{\partial\Omega_n} (\nabla\psi \cdot \hat{n})\phi ds
\end{aligned} \tag{3.2.6}$$

$\forall w \in V_s(\Omega_s)$ and $\forall \phi \in V_f(\Omega_f)$, where $\partial\Omega_n$ is the portion of the solid and fluid boundaries that has applied loads, and f is used to denote body forces on the solid. Also, $\nabla^s = \frac{1}{2}\left(\nabla + \nabla^T\right)$ is the symmetric part of the gradient operator. If Dirichlet boundary conditions were applied to part of the structure, or if the fluid had a portion of its boundary subjected to Dirichlet conditions, then the Sobolev spaces $V_s(\Omega_s)$ and $V_f(\Omega_f)$ would be modified accordingly to correspond to spaces that have those same boundary conditions. Recall that the normal is defined to be positive going from solid into the fluid.

Next, we insert the boundary conditions from equation (3.2.5), and we define $\sigma \cdot \hat{n} = g$ on the solid portion of $\partial\Omega_n$, and $\nabla\psi \cdot \hat{n} = -\rho_f \partial_t u \cdot \hat{n}$ on the fluid portion of $\partial\Omega_n$. This leads to the following weak formulation. Find the mapping $(v, \psi) \quad [0, T] \to V_s(\Omega_s) \times V_f(\Omega_f)$ such that

$$\begin{aligned} \int_{\Omega_s} \rho_s \frac{\partial^2 t}{\partial t^2} w dx + \int_{\Omega_s} \sigma : \nabla^s w dx + \int_{\partial\Omega_{wet}} \frac{\partial\psi}{\partial t} \hat{n} w ds = \int_{\Omega_s} f w dx + \int_{\partial\Omega_n} g w ds, \\ \frac{1}{c^2} \int_{\Omega_f} \frac{\partial^2\psi}{\partial t^2} \phi dx + \int_{\Omega_f} \nabla\psi \cdot \nabla\phi dx - \rho_f \int_{\partial\Omega_{wet}} (\partial_t u \cdot \hat{n}) \phi ds = \\ -\rho_f \int_{\partial\Omega_n} (\partial_t u \cdot \hat{n}) \phi ds \end{aligned} \tag{3.2.7}$$

$\forall w \in V_s(\Omega_s)$ and $\forall\psi \in V_f(\Omega_f)$.

Assuming a linear constitutive model for the solid, and inserting the spatial discretizations $u = (u_x, u_y, u_z) = (\sum u_{x_i} N_i, \sum u_{y_i} N_i, \sum u_{z_i} N_i)$ and $\phi = \sum \phi_i N_i$ into equation (3.2.7) yields the following semi-discrete system of linear ordinary differential equations in time

$$\begin{bmatrix} M_s & 0 \\ 0 & M_f \end{bmatrix} \begin{bmatrix} \ddot{u} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} C_s & L \\ -\rho_f L^T & C_f \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} K_s & 0 \\ 0 & K_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} = \begin{bmatrix} f_s \\ f_f \end{bmatrix}. \tag{3.2.8}$$

where M_s, C_s, and K_s denote the mass, damping, and stiffness matrices for the solid, and M_f, C_f, and K_f denote the same for the fluid. The coupling matrices are denoted by L and L^T. Coupling between fluid and structure, and any damping in the fluid or solid separately, is accounted for by the damping matrices. The quantities f_s and f_f denote the external forces on the solid and fluid, respectively.

3.2.2.3. Nonconforming Structural Acoustics

In the case of nonconforming fluid/solid discretizations, equations (3.2.6) and (3.2.7) contain some extra technicalities. In this section we first describe a simple procedure for coupling two acoustic domains which share a common boundary, but with nonconforming discretizations. This method serves as a stepping stone to the case of nonconforming structural acoustics.

To enforce continuity of appropriate field variables between the two different surfaces, the degrees of freedom and element surfaces involved in the coupling need to be known a priori. Given the surface meshes of the fluid and solid, this information is non-trivial to obtain, especially in parallel, since adjacent element surfaces may reside on different processors.

The ACME and Dash package[26] have been developed as tools to determine surface contact conditions between general surfaces in three dimensions. These surfaces can take the form of boundaries of finite element discretizations, as in our case, or they can be analytic surfaces. In either case, search algorithms are employed to determine node-face interactions between the opposing surfaces, based on search tolerances. A given node is determined to be in contact with a given face of the adjacent surface if the distance from the node to the adjacent element face is within the defined search tolerance. The contact package can compute contact conditions between most of the standard three-dimensional finite elements, including hexahedrons, tetrahedrons, and prisms. Once these interactions are defined, one can devise enforcement algorithms to enforce continuity of the appropriate field variables. Once surface constraints are known, we derive our own enforcement algorithms, as explained below.

We consider the situation shown in Figure (3-1). Here there are 2 interacting acoustic domains, and two contact surfaces. We adopt a node-face approach, where one of the two interacting surfaces contains tied faces and the other tied nodes. We denote surface 1 as the face-surface, and surface 2 as node-surface. For a transient acoustic simulation involving the two meshes shown in Figure 3-1, we would have to solve the system of equations given in equation (3.2.3), which would involve degrees of freedom from both acoustic domains, subject to the constraint that the velocity potential is continuous across the nonconforming interface. The extra equations corresponding to this constraint can be derived from a simple consideration of the contact geometry.

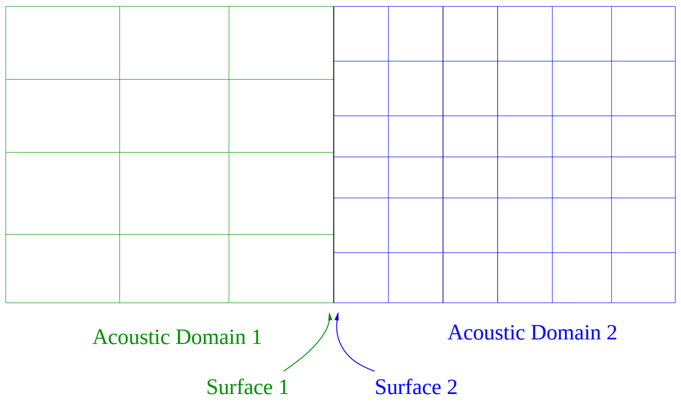


Figure 3-1. – Two interacting acoustic domains, with nonconforming meshes at the common interface. In this case surface 1 is defined to be the face-surface, and surface 2 is the node-surface.

In Figure 3-2, node x from surface 1 is impinging on element face y of surface 2.

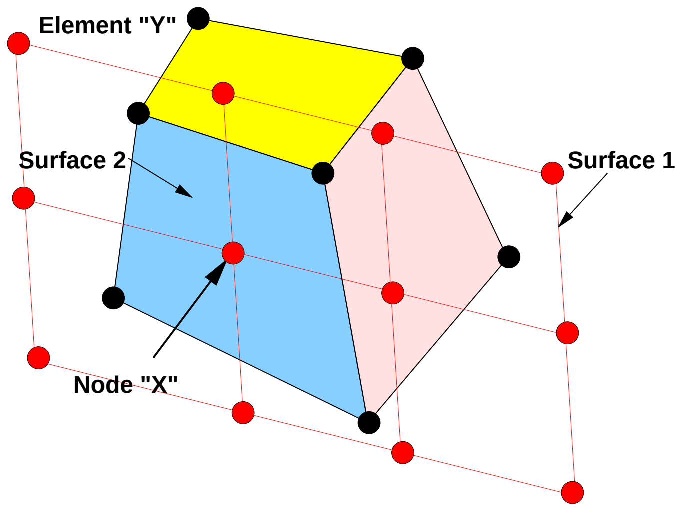


Figure 3-2. – A node-face interaction on the structural acoustic interface.

If contact determines that the distance from node x to element face y is within the user-defined search tolerance, a constraint relation will be needed to enforce continuity of velocity potential. The constraint relation for this interaction can be written in the form

$$\psi^a = \sum_{i=1}^{4} c_i \psi_i^b, \tag{3.2.9}$$

where ψ^a is the velocity potential at node x on surface 1, and ψ_i^b are the velocity potentials at the four nodes of element face y on surface 2. The coefficients c_i are determined from the position of node x relative to the positions of the nodes on element face y on surface 2. More precisely, $c_i = N_i(\xi, \eta)$ are the values of the surface shape functions corresponding to the nodes on the surface of element y in Figure 3-2, and ξ and η are the dimensionless surface coordinates of the location of node x on the surface of element y. Thus, the velocity potential at node x is constrained to be equal to the value that would be predicted by a finite element interpolation on the surface of element y.

For example, in the special case that face y is square and node x lies at the center of the face y, the coefficients c_i would all be equal to $\frac{1}{4}$, indicating that the constraint is an average. This can be seen by considering the surface shape functions corresponding to a plane bilinear element on a square $\xi = -1, 1$, $\eta = -1, 1$.

$$N = \frac{1}{4}[(1-\xi)(1-\eta), (1+\xi)(1-\eta), (1+\xi)(1+\eta), (1-\xi)(1+\eta)]^T. \tag{3.2.10}$$

If node x were at the center of element y, then $\xi = \eta = 0$, and all coefficients would be $\frac{1}{4}$. If x were off-center, these coefficients would change accordingly. If the surface of element y were a triangle instead of a square, (indicating a tetrahedron instead of a hexahedron), the procedure would be the same, except the shape functions in equation 3.2.10 would be different.

We use this approach, sometimes called standard node collocation or inconsistent tied contact,[50] for the nodes/elements on the interacting surfaces. This results in a set of linear constraints that enforces continuity of velocity potential at discrete points between the two acoustic meshes.

It is well known that inconsistent tied contact results in constraints which do not meet convergence criteria for finite elements. In particular, meshes which rely on these methods do not always pass the static patch test for structures.[49,91,113,134] Other methods such as mortar methods, provide more accurate, but more complex approaches. Fundamentally, these methods are similar to those presented here, as the concepts of tying the acoustic degrees of freedom through a system of constraint equations apply.

These constraint equations can be expressed as[37]

$$C\Phi = 0, \tag{3.2.11}$$

where C is a matrix that contains the constraint coefficients from the node-face interactions, and vector Φ contains all degrees of freedom for the problem. The vector Φ can be partitioned as

$$\Phi = \begin{bmatrix} \Phi_f \\ \Phi_n \end{bmatrix}, \tag{3.2.12}$$

where Φ_n contains all node-surface acoustic degrees of freedom and Φ_f the face-surface degrees of freedom. With this partition, equation (3.2.11) can be written as

$$C_m\Phi_f + C_s\Phi_n = 0. \tag{3.2.13}$$

We note that the matrix C_s is diagonal either for the constraint enforcement approach used here or for a dual mortar method.[134,113] If the constraint equations are linearly independent (assuming there are no redundant constraints), then the matrix C_s is also nonsingular. The node-surface degrees of freedom can be condensed from the stiffness matrix by using $\Phi_n = C_{ms}\Phi_f$, where we define $C_{ms} = -C_s^{-1}C_m$. Additional details are provided later.

Next, we examine the dimensions of the constraint matrices defined above, and their relation with the number of acoustic and structural nodes on the wet interface. We define n_s as the number of nodes on the structural side of the wet surface, and n the total number of degrees of freedom for the problem. The dimensions of C_s is then seen to be n_s by n_s, while the dimensions of C_m is n_s by $n - n_s$. For example, consider the mesh shown in Figure (3-1). If we assume that the domain on the right is a structural domain (instead of acoustic), we would have $n_s = 7$. In addition, only 5 columns of C_m would have nonzero entries.

The condensation expression[37] holds,

$$\tilde{K} = K_{mm} + K_{ms}C_{ms} + C_{ms}^T K_{sm} + C_{ms}^T K_{ss} C_{ms}, \tag{3.2.14}$$

as do the similar expressions for mass and damping. While static condensation does generate non-diagonal matrices, it does not significantly effect the sparsity of $\tilde{K}$ or $\tilde{M}$, since these are *local* constraint equations that involve only a few degrees of freedom. After condensing out the node-surface acoustic degrees of freedom in equation (3.2.3), we obtain a modified system of equations

$$\tilde{M}\ddot{\psi} + \tilde{K}\psi = \tilde{f}_a, \tag{3.2.15}$$

where the tilde superscripts indicate that the node-surface constraints have been condensed out. Note that the vector ψ only contains the interior degrees of freedom (corresponding to nodes that are not on the interacting surfaces), and the face-surface degrees of freedom on the contact surface, since the node-surface degrees of freedom have been eliminated. Equations (3.2.15) can also be solved in the frequency domain, as follows

$$\left[s^2\tilde{M} + \tilde{K}\right]\psi = \tilde{f}_a, \tag{3.2.16}$$

where s is the frequency parameter that comes from the Laplace transform.

In the case of structural acoustics, the algorithm for the nonconforming fluid/fluid meshes can be used as a stepping stone to the nonconforming solid/fluid meshes. In this approach ghost structural or acoustic degrees of freedom are added to one side of the wet interface. Due to the ghost degrees of freedom collocated structural and acoustic degrees of freedom are present one side of the wet interface (e.g. three displacement and one velocity potential degree of freedom). Two surface integrals in equation (3.2.7), i.e. $\int_{\partial\Omega_{wet}} \partial_t\psi\hat{n}w ds$ and $\rho_f \int_{\partial\Omega_{wet}} \partial_t u \cdot \hat{n}\phi ds$, are evaluated to couple the structural acoustic coupling terms at these collocated degrees of freedom. Across the interface the like degrees of freedom (the "true" degrees of freedom and their ghost counterparts) are tied together using the same set of linear constraint equations that were developed for the nonconforming structure/structure case.

In addition to equations (3.2.8), we have a set of linear constraint equations that couple shared degrees of freedom across the wet interface. As in the structure/structure case, these constraint equations represent the relations between the face-surface and node-surface degrees of freedom, and they take the same form given by equation (3.2.11). Upon condensing these constraints out of the system of equations, (3.2.8), we obtain a modified system of equations

$$\begin{bmatrix} \tilde{M}_s & 0 \\ 0 & \tilde{M}_f \end{bmatrix}\begin{bmatrix} \ddot{u} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \tilde{C}_s & \tilde{L} \\ -\rho_f\tilde{L}^T & \tilde{C}_f \end{bmatrix}\begin{bmatrix} \dot{u} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \tilde{K}_s & 0 \\ 0 & \tilde{K}_f \end{bmatrix}\begin{bmatrix} u \\ \psi \end{bmatrix} = \begin{bmatrix} \tilde{f}_s \\ \tilde{f}_f \end{bmatrix}, \tag{3.2.17}$$

where again the tilde superscripts represent the matrices with constraints condensed out. Note that, in this case, the structural matrices (and coupling matrices) must be modified during the constraint removal process. This is because of the coupling matrices L and L^T involve uncondensed degrees of freedom. To solve this system of equations, we use the generalized alpha time integration method,[34] which is a generalization of the Newmark-beta method.

In addition to the transient analysis formulation outlined above, an advantage of our coupling procedure is that it can be applied equally well to nonconforming structural acoustic problems for both eigenvalue analysis, and frequency domain analysis. The coupling terms lead to a quadratic eigenvalue problem.

$$\left(\begin{bmatrix} \tilde{K}_s & 0 \\ 0 & -\tilde{K}_f/\rho_f \end{bmatrix} + \lambda \begin{bmatrix} \tilde{C}_s & \tilde{L} \\ \tilde{L}^T & -\tilde{C}_f/\rho_f \end{bmatrix} + \lambda^2 \begin{bmatrix} \tilde{M}_s & 0 \\ 0 & -\tilde{M}_f/\rho_f \end{bmatrix}\right) \begin{bmatrix} u \\ \psi \end{bmatrix} = 0 \tag{3.2.18}$$

In the case of zero damping, this is a gyroscopic system with imaginary eigenvalues, and complex eigenvectors.

The frequency domain equation can be obtained by a Fourier transform of the time domain equation. This results in following complex-valued system of equations.

$$\left(\begin{bmatrix} \tilde{K}_s & 0 \\ 0 & -\tilde{K}_f/\rho_f \end{bmatrix} + i\omega \begin{bmatrix} \tilde{C}_s & \tilde{L} \\ \tilde{L}^T & -\tilde{C}_f/\rho_f \end{bmatrix} - \omega^2 \begin{bmatrix} \tilde{M}_s & 0 \\ 0 & -\tilde{M}_f/\rho_f \end{bmatrix}\right) \begin{bmatrix} u \\ \psi \end{bmatrix} = \begin{bmatrix} \tilde{f}_s \\ -\tilde{f}_f/\rho_f \end{bmatrix}. \tag{3.2.19}$$

In the next section on numerical results, we present results from all cases, including time domain, frequency domain, and eigenvalue analysis simulations.

Our method can be summarized by the diagram in Figure (3-3). In the shown example the structural nodes on the wet interface are augmented with the acoustic degree of freedom. Consequently, these nodes each have four degrees of freedom. In this example the acoustic degrees of freedom are constrained across the interface via an acoustic-to-acoustic MPC. The structure to acoustic coupling is enforced on the structure side of the interface which has conforming structural and acoustic degrees of freedom.

One case that requires special care for structural acoustic coupling is double wetted shells (a structural shell sandwiched between two acoustic domains.) For this case the structural velocities at the shell and the two acoustic domains should be identical. However, the acoustic pressure potentials at the two acoustic domains are not identical. To correctly run this case, the structural degrees of freedom should be tied with MPCs across the three domains and the structure-to-acoustic coupling terms be evaluated on the acoustic domains. This enables two separate and potentially disjoint acoustic degrees of freedom to be present at the interface. The proper setup for this case is shown in Figure (3-4).

The dual mortar method[134,113] generates a similar set of constraint equations.

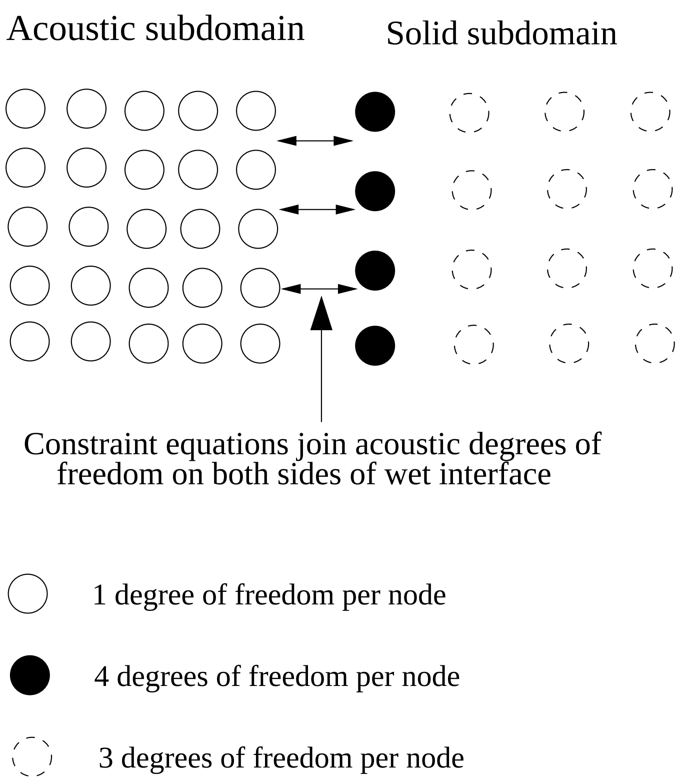

Acoustic subdomain
Solid subdomain
Constraint equations join acoustic degrees of freedom on both sides of wet interface
1 degree of freedom per node
4 degrees of freedom per node
3 degrees of freedom per node

Figure 3-3. – Illustration of our method for structural acoustic meshes with nonconforming interfaces. Ghost acoustic degrees of freedom are added to the structural side of the wet interface, and then connected to the adjacent acoustic surface with constraint equations. The resulting nodes in the mesh can then have either one acoustic degree of freedom (shown by a circle), three displacement degrees of freedom (shown by a dashed circle), or one acoustic degree of freedom and three displacement degrees of freedom (shown by a black-filled circle).

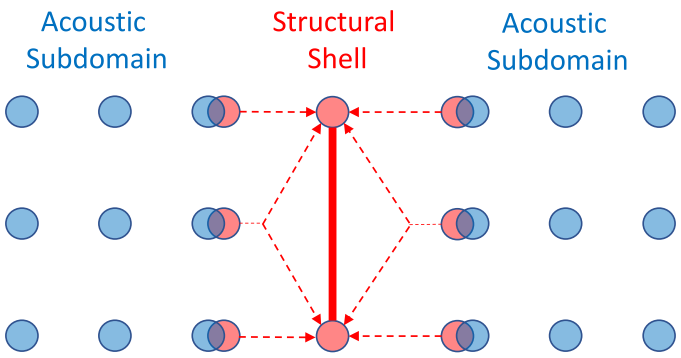


Figure 3-4. – Nonconformal Structural Acoustic Tying for Doubled Wetted Shell.

3.3. Acoustic Scattering

Acoustic scattering refers to the interaction of plane acoustic waves with solid bodies which are immersed in an infinite acoustic fluid. The plane waves are assumed to originate from infinity, and after impinging on the solid body, they continue to propagate to infinity. In scattering simulations, the velocity potential is decomposed into a sum of the incident potential, and scattered potential

$$\psi^{tot} = \psi^{in} + \psi^{sc} \tag{3.3.1}$$

where ψ^{tot} is the total potential, ψ^{in} is the incident potential, and ψ^{sc} is the scattered potential. The incident potential is a known quantity, and the scattered potential is unknown. Thus, in the final formulation, the incident potential becomes part of the right-hand side forcing function, and the scattered potential remains on the left-hand side as an unknown.

We recall that the linear wave equation in terms of the total velocity potential is given by

$$\frac{1}{c^2}\ddot{\psi}^{tot} - \Delta\psi^{tot} = 0 \tag{3.3.2}$$

Decomposing this into incident and scattered fields, we have

$$\left[\frac{1}{c^2}\ddot{\psi}^{in} - \Delta\psi^{in}\right] + \left[\frac{1}{c^2}\ddot{\psi}^{sc} - \Delta\psi^{sc}\right] = 0 \tag{3.3.3}$$

Since the incident wave is assumed to satisfy the wave equation, the first part of the expression can be dropped, and we are left with

$$\frac{1}{c^2}\ddot{\psi}^{sc} - \Delta\psi^{sc} = 0 \tag{3.3.4}$$

This implies that we can solve for the scattered potential directly. The effect of the incident field is then accounted for in the boundary conditions on the wet surface.

For scattering in the context of the coupled structural acoustic problem, it is most convenient to solve for the scattered acoustic potential in the fluid and the total displacement field in the structure. With that assumption, we have the following partial differential equations

$$\begin{aligned} \rho_s u_{tt}^{tot} - \nabla \cdot \sigma &= F, \\ \frac{1}{c^2}\ddot{\psi}^{sc} - \Delta\psi^{sc} &= 0 = 0. \end{aligned} \tag{3.3.5}$$

Here u^{tot} corresponds to the total displacement of the structure, σ is the structural stress tensor, ρ_s is the density in the solid, and F denotes body forces on the solid. Subsequently, subscripts s and f refer to solid and fluid, respectively.

In the case of linear acoustics, the boundary conditions on the fluid/solid interface (wet interface, which is designated by $\partial\Omega_{wet}$), are

$$\frac{\partial\psi^{tot}}{\partial n} = -\rho_f \dot{u}_n^{tot} \tag{3.3.6}$$

$$\sigma_n = -\dot{\psi}^{tot}\hat{n} = -\left[\dot{\psi}^{in} + \dot{\psi}^{sc}\right]\hat{n} \tag{3.3.7}$$

where ρ_f is the density of the fluid, and $\hat{n}$ is the surface normal vector. These boundary conditions correspond to continuity of velocity and stress at the wet interface. For equation (3.3.6), we note that we rearrange the terms for convenience

$$\begin{aligned} \frac{\partial \psi^{tot}}{\partial n} &= \frac{\partial \psi^{in}}{\partial n} + \frac{\partial \psi^{sc}}{\partial n} \\ &= -\rho_f \dot{u}_n^{tot} \end{aligned} \tag{3.3.8}$$

Rearranging, we have

$$\frac{\partial \psi^{sc}}{\partial n} = -\rho_f \dot{u}_n^{tot} - \frac{\partial \psi^{in}}{\partial n} \tag{3.3.9}$$

Equations (3.3.9) and (3.3.7) are in the form that we can insert them directly into the variational formulation (3.2.6), with the recognition that the unknowns are the total structural displacement and scattered velocity potential. Carrying this through, and assuming a linear constitutive model for both the solid and fluid, the time domain equations of motion can be represented by the following semi-discrete system of linear ordinary differential equations

$$\begin{bmatrix} M_s & 0 \\ 0 & \frac{-1}{\rho_a} M_a \end{bmatrix} \begin{bmatrix} \ddot{u}^{tot} \\ \ddot{\psi}^{sc} \end{bmatrix} + \begin{bmatrix} C_s & L \\ L^T & \frac{-1}{\rho_a} C_a \end{bmatrix} \begin{bmatrix} \dot{u}^{tot} \\ \dot{\psi}^{sc} \end{bmatrix} + \begin{bmatrix} K_s & 0 \\ 0 & \frac{-1}{\rho_a} K_a \end{bmatrix} \begin{bmatrix} u^{tot} \\ \psi^{sc} \end{bmatrix} = \begin{bmatrix} f_s \\ \frac{-1}{\rho_a} f_a \end{bmatrix}, \tag{3.3.10}$$

where M_s, C_s, and K_s denote the mass, damping, and stiffness matrices for the solid, M_a, C_a, K_a denote the same for the acoustic fluid, ρ_a is the density of the acoustic fluid, and u and ψ denote the structural displacement and fluid velocity potential. The coupling matrices are denoted by L and L^T. Coupling between fluid and structure, and any damping in the fluid or solid separately, is accounted for by the damping matrices. The quantities f_s and f_a denote the external forces on the solid and fluid, respectively.

The acoustic load f_a for the scattering problem can be written in the form

$$f_a = -\int_{\partial\Omega_n} \frac{\partial \psi^{in}}{\partial n} \phi ds \tag{3.3.11}$$

where again ϕ is a test function. Since $\frac{\partial \psi^{in}}{\partial n}$ is a known quantity, we can integrate equation (3.3.11) to obtain the loading on the fluid side of the wet interface.

The expression for loading on the structure due to scattering loads is given by

$$f_s = \int_{\partial\Omega_n} \dot{\psi}^{in} w ds \tag{3.3.12}$$

where w is a test function for the structural discretization. Since $\dot{\psi}^{in}$ is a known quantity, the force on the solid body can be computed from equation (3.3.12). Note that equations (3.3.11) and (3.3.12) require the spatial and temporal derivatives of the incident field, ψ^{inc}. Thus, even if ψ^{in} is known, methods for computing its spatial and temporal derivatives are also required.

Inserting the expressions for f_a and f_s from equations (3.3.11) and (3.3.12) into equations (3.3.10), we can solve for the responses of the acoustic fluid and solid body to incident acoustic waves. The only requirement on ψ^{in} is that it satisfies the acoustic wave equation. Note that the solution to equations (3.3.10) will give the scattered acoustic potential. To compute the total acoustic potential, we would need to add the incident and scattered potentials together, as in equation (3.3.1). Also, we note that the loads from equations (3.3.11) and (3.3.12) are generated by a single incident wave. For multiple incident waves (as in the case of a diffuse field), the right-hand side of equations (3.2.17) involve a simple superposition of the incident waves.

3.3.1. *Frequency Domain scattering*

The incident potential satisfies the wave equation, and for a plane wave takes the form

$$\psi^{in} = Ae^{i[k\cdot\mathbf{x}-\omega t]} \tag{3.3.13}$$

where $\omega = 2\pi f$ is the circular frequency of the wave, f is the frequency in Hz, k is the vector wave number, and $\mathbf{x}$ is the vector coordinates of a point in space. The vector wave number has three components, $k = (k_x, k_y, k_z)$, which define the direction of propagation of the wave. For example, for a wave propagating strictly in the x direction, we would have $k = (k_x, 0, 0)$, where $k_x = \frac{\omega}{c}$ would be the standard wave number from one-dimensional wave propagation. The parameter A is a scalar constant that defines the magnitude of the wave. Although A can be made to vary with frequency, we will only consider the case where A is a scalar constant. This implies that all incoming plane waves have the same amplitude (but different frequencies). In the frequency domain, the time portion of the expression in equation (3.3.13) drops out, and we are left with

$$\psi^{in} = Ae^{ik\cdot\mathbf{x}} \tag{3.3.14}$$

We consider a three-dimensional elastic body, which is immersed in an infinite acoustic fluid, and subjected to impinging plane waves from infinity in the frequency domain. The equations of motion of the coupled system are given by

$$-\omega^2\begin{bmatrix}\tilde{M}_s & 0\\ 0 & \tilde{M}_a\end{bmatrix}\begin{bmatrix}u^{tot}\\ \psi^{sc}\end{bmatrix}+i\omega\begin{bmatrix}\tilde{C}_s & \tilde{L}\\ -\rho_f\tilde{L}^T & \tilde{C}_f\end{bmatrix}\begin{bmatrix}u^{tot}\\ \psi^{sc}\end{bmatrix}+\begin{bmatrix}\tilde{K}_s & 0\\ 0 & \tilde{K}_a\end{bmatrix}\begin{bmatrix}u^{tot}\\ \psi^{sc}\end{bmatrix}=\begin{bmatrix}\tilde{f}_s\\ \frac{-1}{\rho_a}\tilde{f}_a\end{bmatrix}. \tag{3.3.15}$$

We recall that the portion of the acoustic load f_a that comes from Neumann boundary conditions can be computed from equation (3.3.11). Given equation (3.3.14), we define $n = (n_x, n_y, n_z)$ to be the surface normal of the solid body. We also let $k = \frac{\omega}{c}(\text{dir}_x, \text{dir}_y, \text{dir}_z)$, where $(\text{dir}_x, \text{dir}_y, \text{dir}_z)$ define the direction cosines of the direction of propagation of the incident plane wave. Then, we have

$$\frac{\partial\psi^{in}}{\partial n} = \nabla\psi^{in}\cdot n = i\frac{\omega}{c}\left[n_x\text{dir}_x + n_y\text{dir}_y + n_z\text{dir}_z\right]Ae^{ik\cdot x} \tag{3.3.16}$$

Inserting this expression into equation (3.3.11), and integrating, we obtain the loading on the acoustic fluid due to scattering.

For the loading on the structure, we recall the expression for loading on the structure due to Neumann boundary conditions in equation (3.3.12). In the frequency domain case, $\sigma_n = n\dot{\psi}^{in} = in\omega\psi^{in} = in\omega Ae^{i(k\cdot x)}$. Inserting this expression into equation (3.3.12), and integrating, we obtain the loading on the solid body due to scattering.

Finally, we examine the complex-valued loads presented in equations (3.3.11) and (3.3.12). We make two observations regarding these loads.

1. These loads have real and imaginary parts, and thus even for a single plane wave, they cannot be combined into a single vector, even though they have the same multiplication factor A. Currently, **Sierra/SD** combines load vectors that have the same time function into a single array. For the case of complex loads in the frequency domain, this translates into combining the real and imaginary parts into a single array if they have the same “time" function, which in this case corresponds to the multiplication factor A. A temporary work-around is to use distinct time functions for the real and imaginary parts in the input deck. (even if the time functions themselves are identical). Otherwise, if the same time function is used, the real and imaginary parts would be combined into a single vector in **Sierra/SD**.

2. We have considered the case where the coefficient A is a scalar constant, but we could also consider the case where $A = A(\omega)$ is a function of frequency. This would correspond to multiple plane waves of different amplitudes impinging on the structure. Since the spatial parts of these loads varies with frequency, they could not be computed by adding the spatial parts together before multiplying by the coefficient $A(\omega)$. Thus, we would have an inconsistency with the current approach in **Sierra/SD** of adding the spatial parts together before multiplying by the time function (which in this case would be $A(\omega)$).

3.4. Nonlinear Acoustics

Linear acoustic theory is based on the assumptions of small amplitude waves and a linear constitutive theory of the fluid medium. Although these assumptions hold for many vibro-acoustic interactions, they are invalid in sound fields with high sound pressure levels,[105] i.e. sound fields that have *finite amplitude waves*. Finite amplitude waves can be generated in interior fields when resonance occurs,[55] in the far-field of atmospheric and underwater explosions,[32] in tire noise generation,[68] and in many aeroacoustic sources (such as sonic booms).[71] Nonlinear effects increase with the frequency of the waves, and thus the study of nonlinear acoustics has also become important in high-frequency applications such as ultrasound.[77,31] Unlike the linear acoustic wave equation, the nonlinear counterparts can handle waves with finite amplitude, and allow more accurate modeling of nonlinear constitutive models in the fluid.

The classical Kuznetsov equation[87] treats three-dimensional nonlinear acoustic waves to second order in nonlinearity. Recently, Soderholm[125] generalized Kuznetsov's equation using the exact equation of state, rather than a series expansion. The nonlinear terms in these wave equations imply that the sound speed depends on the stress state in the fluid. This leads, eventually, to the formation of weak shocks (small discontinuities in acoustic pressure). For a mono-frequency source, energy will be gradually transferred from lower harmonics to higher harmonics, leading to a steepening of an initially smooth wave. Weak shocks radiated from a structure lead to unpleasant cracking noise, and when impinging on a structure they cause a different response than smooth acoustic waves. Thus, it is important to characterize their effects in both noise radiation and structural coupling problems.

The governing equations of acoustics can be formulated in terms of particle displacement, or scalar-based quantities such as acoustic pressure or velocity potential. In particle displacement approach, the mesh moves with the waves, whereas in the latter approaches the mesh is fixed. The primary advantage of the displacement approach is its easy coupling with a Lagrangian solid mechanics code, since the unknowns are the same as for the solids. The displacement approach has been studied in,[109,33,131] though these references dealt only with the linear case. Since ideal fluids have zero shear modulus, this approach suffers from an infinite dimensional null space consisting of rotational modes in the fluid. Numerically, this leads to *spurious modes* that pollute the computed solution. These modes can be eliminated through the use of penalty formulations, but this can result in poor conditioning. Displacement formulations for acoustics are also prone to mesh tangling in the case of large displacements in either the solid or the fluid, making them inappropriate for many applications.

In the Eulerian approach, the unknown is typically acoustic pressure or velocity potential. In problems without structural coupling, the mesh remains stationary. In addition, the null space consists only of the constant pressure mode, which makes these formulations more stable for numerical computations. On the other hand, for coupled solid/fluid problems, the Eulerian formulation requires a coupling mechanism between fluid and solid to handle the different degrees of freedom used to discretize the fluid/solid domains. In the case of small structural displacements, this coupling mechanism reduces to coupling operators that couple acoustic pressure and structural displacements between fluid and solid. In the case of large structural displacements or rotations, methods such as the Arbitrary Lagrangian-Eulerian (ALE) approach, which have been developed for aeroelastic coupling,[58,59] could also be applied to the structural acoustics problem. An alternative approach in the case of large structural motion is an Eulerian method for the fluid allowing the solid/fluid boundary to cuts through fluid elements. Regardless of the approach taken for the structural coupling, we have chosen the Eulerian approach for acoustic discretization, since it avoids the null space issues eluded to earlier.

Unlike the rich history of finite element formulations in nonlinear solid mechanics, the finite element formulation of nonlinear acoustic equations for fluids has received considerably less attention. Cai et al[31] recently used finite elements and parallel computations to solve Kuznetsov's equation to model ultrasonic waves. In a sequence of works, Hoffelner et al[77] also used a finite element method to solve Kuznetsov's equation. Later,[76] they used their method to simulate acoustic streaming and radiation force, two important acoustic phenomena that cannot be captured from linear theory. Kagawa[82] took a similar approach in solving Kuznetsov's equation, except that additional approximations were made to the equation before discretization. Vanhille et al[130] used finite differences and finite volume methods to solve a nonlinear acoustic wave equation in the Lagrangian framework.

In this section, we present a finite element implementation of the Kuznetsov wave equation. We derive the full tangent operator for the spatial discretization, and give an implementation of a time discretization scheme using the generalized alpha method. We then derive a formulation for coupling the Kuznetsov equation to the equations of motion of an elastic solid.

To illustrate ideas, we begin with the linear acoustic wave equation

$$\frac{1}{c^2}\frac{\partial^2 \phi}{\partial t^2} - \Delta\phi = 0 \tag{3.4.1}$$

where ϕ is the velocity potential ($\phi = \nabla u$, where u is the particle velocity), and c is the speed of sound. The derivation of this equation neglects both convective and constitutive nonlinearities.

The nonlinear isentropic equation of state for air can be written as follows

$$\frac{P}{P_0} = \left(\frac{\rho}{\rho_0}\right)^{\gamma} \tag{3.4.2}$$

where P and P_0 are the total and reference pressures, ρ and ρ_0 are the current and reference densities. γ is the ratio of specific heats, and is equal to 1.4 for air. Equation 3.4.2 can then be combined with the conservation of momentum and conservation of mass for the fluid to derive nonlinear wave equations. In Soderholm's approach, equation 3.4.2 is used directly. In Kuznetsov's approach, it is first expanded in a Taylor series about the isentrope $s = s_0$[71]

$$p = P - P_0 = \left(\frac{\partial P}{\partial \rho}\right)_{s_0,\rho_0}(\rho-\rho_0) + \frac{1}{2}\left(\frac{\partial^2 P}{\partial \rho^2}\right)_{s_0,\rho_0}(\rho-\rho_0)^2 + \ldots \tag{3.4.3}$$

which can be written compactly as

$$p = A\left(\frac{\rho-\rho_0}{\rho_0}\right) + \frac{B}{2}\left(\frac{\rho-\rho_0}{\rho_0}\right)^2 + \ldots \tag{3.4.4}$$

where $A = \rho_0 \left(\frac{\partial P}{\partial \rho}\right)_{s_0,\rho_0} \equiv \rho_0 c_0^2$, and $B = \rho_0^2 \left(\frac{\partial^2 P}{\partial \rho^2}\right)_{s_0,\rho_0}$. Since $\left(\frac{\partial P}{\partial \rho}\right)_{s_0,\rho_0} = c_0^2$ is the square of the linear speed of sound, we see from the expansion that the ratio of the first two terms is

$$\frac{B}{A} = \frac{\rho_0}{c_0^2} \left(\frac{\partial^2 P}{\partial \rho^2}\right)_{s_0,\rho_0} \tag{3.4.5}$$

The parameter B/A accounts for the nonlinear constitutive law of the fluid up to second order. A table of values of B/A for various fluids can be found in texts on nonlinear acoustics.[71]

For linear acoustics, only the first term in the expansion 3.4.4 is retained. In that case, we have

$$p = A\left(\frac{\rho - \rho_0}{\rho_0}\right) = c_0^2(\rho - \rho_0) \tag{3.4.6}$$

which implies that the stiffness of the fluid is the square of the linear speed of sound.

Kuznetsov's equation uses the above Taylor series expansion of the equation of state, but truncates all terms past the second. It also accounts for convective nonlinearities to second order. The equation is derived by combining the Taylor series expansion of the equation of state with the conservation of mass and momentum. The result is the following..[87,55,98,105]

$$\frac{1}{c^2}\frac{\partial^2 \phi}{\partial t^2} - \Delta\phi - \frac{1}{c^2}\frac{\partial}{\partial t}\left(b(\Delta\phi) + \frac{B/A}{2c^2}\left(\frac{\partial \phi}{\partial t}\right)^2 + (\nabla\phi)^2\right) = 0 \tag{3.4.7}$$

where ϕ is defined as $p = \rho_f \frac{\partial \phi}{\partial t}$, and p is the acoustic pressure. The first two terms in equation 3.4.7 are the same as in equation 3.4.1, but the fourth and fifth terms are nonlinear. The third term is a linear absorption term. It is grouped with the nonlinear terms to indicate deviation from the linear wave equation. The parameter b is for absorption in the fluid due to viscosity and thermal conductivity.

Equation 3.4.7 was originally developed in terms of the velocity potential. Here, instead of solving for the velocity potential, we prefer to solve for ψ such that $p = \dot{\psi}$. This implies that $\phi = \frac{1}{\rho}\psi$. Inserting this relation into equation 3.4.7 yields

$$\frac{1}{c^2}\frac{\partial^2 \psi}{\partial t^2} - \Delta\psi - \frac{1}{c^2}\frac{\partial}{\partial t}\left(b(\Delta\psi) + \frac{B/A}{2\rho c^2}\left(\frac{\partial \psi}{\partial t}\right)^2 + \frac{(\nabla\psi)^2}{\rho}\right) = 0 \tag{3.4.8}$$

This is done only for convenience, since the acoustic pressure can easily be computed during post processing as $p = \dot{\psi}$. For simplicity, we will still refer to ψ as the velocity potential in the remainder of this paper.

Soderholm[125] derived a higher order nonlinear acoustic equation that accounts for nonlinearities to higher order. In this approach, the exact equation of state, equation 3.4.2, is used directly, rather than the second order expansion of Kuznetsov's equation. This equation is only valid for air, whereas Kuznetsov's equation can be used for any fluid that has a tabulated value of $\frac{B}{A}$. After combining the equation of state with the conservation of mass and momentum, the following equation results

$$\frac{1}{c_0^2}\frac{\partial^2 \phi}{\partial t^2} - \Delta\phi - \frac{b}{c_0^2}\frac{\partial}{\partial t}(\Delta\phi) + \frac{1}{c_0^2}\frac{\partial}{\partial t}(\nabla\phi)^2$$
$$+\frac{1}{2c_0^2}\nabla\phi \cdot \nabla(\nabla\phi)^2 + \frac{\gamma - 1}{c_0^2}\left(\frac{\partial}{\partial t}\phi + \frac{1}{2}(\nabla\phi)^2\right)\Delta\phi = 0$$

We note that Soderholm's equation is a generalization of the exact relation given by equation 3.26 in,[71] which was derived for the case of a lossless fluid. The only difference is the term $\frac{b}{c_0^2}\frac{\partial}{\partial t}(\Delta\phi)$, which accounts for absorption.

The range of validity of nonlinear wave equations is typically given in terms of acoustic mach number.

$$M = \frac{u}{c_0} \tag{3.4.9}$$

where u is the particle velocity, and c_0 is the linear speed of sound. Rough guidelines are given in.[98] For the Kuznetsov equation, a limit of $M \leq 0.1$ is given. For a third order wave equation, a limit of $M \leq 0.7$ is given. These are useful guidelines for the acoustic analyst, who needs to decide which equation applies to their needs.

In summary, three-dimensional nonlinear acoustic waves in thermo-viscous fluids can be modeled using equations derived by Kuznetsov and, more recently, by Soderholm. These equations include the linear wave equation as a special case. Kuznetsov's equation generalizes the linear wave equation to include nonlinearities to second order and linear dissipation. Soderholm's equation is an additional generalization that allows for higher degrees of nonlinearity. The dissipative term in Soderholm's equation is the same as in Kuznetsov's equation.

3.4.1. Weak Formulations

In this paper we will only work with Kuznetsov's equation, since we are interested in a formulation that is valid for any fluid. A weak formulation of equation 3.4.8 can be constructed by multiplying with a test function and integrating by parts. We denote the fluid domain by Ω_f and its boundary by $\partial\Omega = \partial\Omega_n \bigcup \partial\Omega_d$, where the subscripts n and d refer to the portions of the boundary where Neumann and Dirichlet boundary conditions are applied. We also assume that the fluid is initially at rest, i.e. $\psi(x,0) = \dot{\psi}(x,0) = \ddot{\psi}(x,0) = 0$, which is sufficient for most applications.

Denoting by $V_f(\Omega_f)$ the function space for the fluid, and $\Gamma = \partial\Omega_f$, the weak formulation can be written as follows. Find the mapping $\psi : \quad [0,T] \longrightarrow V_f(\Omega_f)$ such that

$$\begin{aligned}\frac{1}{c^2}\int_\Omega \ddot{\psi}\phi dx + \int_\Omega \nabla\psi\cdot\nabla\phi dx \qquad &+\frac{1}{c^2}\int_\Omega b\nabla\dot{\psi}\cdot\nabla\phi dx \\ -\frac{B}{A\rho c^4}\int_\Omega \ddot{\psi}\dot{\psi}\phi dx \quad &-\frac{2}{\rho c^2}\int_\Omega \nabla\dot{\psi}\cdot\nabla\psi\phi dx = \\ \int_\Gamma \frac{\partial\psi}{\partial n}\phi ds = \qquad &-\int_\Gamma \rho_f(\dot{u}_n + \frac{b}{c^2}\ddot{u}_n)\phi ds\end{aligned} \tag{3.4.10}$$

$\forall\phi \in V_f(\Omega_f)$, where $\dot{u}_n$, and $\ddot{u}_n$ are the prescribed particle velocity and acceleration on the Neumann portion of the fluid boundary. Here we use ϕ to denote the test function, and not the velocity potential as denoted earlier. We note that for air, $\frac{b}{c^2}$ is of the order $1e^{-10}$ under normal conditions, and thus it is sufficient to drop the acceleration term and approximate the right-hand side as $-\int_{\partial\Omega_n}\rho_f\dot{u}_n\phi ds$. We will make this approximation in the remainder of this paper.

We note that an interesting feature of the weak formulation of equation 3.4.8 is that the integration by parts only occurs on the linear elliptic terms. The nonlinear terms are not integrated by parts.

3.4.2. *Spatial and Temporal Discretization*

A finite element formulation of equation 3.4.10 is constructed by representing the unknown by a finite summation $\psi(x) = \sum_{i=1}^{n} \psi_i N_i(x) = \psi^T N$, and substituting in equation 3.4.10. This leads to the following set of nonlinear ordinary differential equations in time

$$F_{int}(\ddot{\psi}(x,t), \dot{\psi}(x,t), \psi(x,t)) = F_{ext}(x,t) \tag{3.4.11}$$

where

$$F^{int} = \frac{1}{c^2}\int_\Omega \ddot{\psi}\phi dx + \int_\Omega \nabla\psi \cdot \nabla\phi dx \tag{3.4.12}$$

$$+\frac{1}{c^2}\int_\Omega b\nabla\dot{\psi} \cdot \nabla\phi dx - \frac{1}{\rho c^4}(B/A)\int_\Omega \ddot{\psi}\dot{\psi}\phi dx - \frac{2}{\rho c^2}\int_\Omega \nabla\dot{\psi} \cdot \nabla\psi\phi dx \tag{3.4.13}$$

and

$$F_{ext} = -\int_{\partial\Omega_n} \rho_f \ddot{u}_n \phi ds \tag{3.4.14}$$

F^{int} is the internal force, which depends on ψ and its first two time derivatives, and F^{ext} is the external force. We note that $\ddot{\psi}$ and $\dot{\psi}$ depend on ψ through the time discretization scheme, and thus we could write equation 3.4.11 as

$$F_{int}(\psi(x,t)) = F_{ext}(x,t) \tag{3.4.15}$$

To linearize equation 3.4.11, we could use a finite difference approach, in which the tangent matrix is derived by differencing the internal force function with respect to an incremental displacement. Alternatively, we could derive a full Newton tangent matrix by taking partial derivatives with respect to the independent variables. We have taken the latter approach, since it reveals explicitly the fact that the tangent matrix is nonsymmetric.

We define $\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}$ as the current iterates, and $\psi, \dot{\psi}, \ddot{\psi}$ as the unknowns. The tangent equations can be derived by expanding the left-hand side of equation 3.4.11 in a Taylor series. If we truncate all terms beyond the constant and linear contributions, we obtain

$$F_{int}(\psi, \dot{\psi}, \ddot{\psi}) \approx F_{int}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) + \left[\frac{\partial F_{int}}{\partial \psi}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) + \frac{\partial F_{int}}{\partial \dot{\psi}}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}})\frac{\partial \dot{\psi}}{\partial \psi} + \frac{\partial F_{int}}{\partial \ddot{\psi}}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}})\frac{\partial \ddot{\psi}}{\partial \psi}\right]\Delta\psi = F_{int}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) + A\Delta\psi \tag{3.4.16}$$

where $\Delta\psi = \psi - \tilde{\psi}$, and $\tilde{\psi}$ is the current iterate. The full tangent matrix A is defined as

$$A = \left[\frac{\partial F_{int}}{\partial \psi}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) + \frac{\partial F_{int}}{\partial \dot{\psi}}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}})\frac{\partial \dot{\psi}}{\partial \psi} + \frac{\partial F_{int}}{\partial \ddot{\psi}}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}})\frac{\partial \ddot{\psi}}{\partial \psi}\right] \tag{3.4.17}$$

Since $\Delta\psi$ is unknown, we approximate it as $\Delta\tilde{\psi} = \tilde{\psi} - \tilde{\tilde{\psi}}$, where $\tilde{\tilde{\psi}}$ is the previous iterate. Thus, as convergence occurs, the current and previous iterates become identical.

We have chosen the generalized alpha time integration scheme[34] to discretize equation 3.4.11 in time. The generalized alpha method is based on the generalized Newmark method. The flexibility of this method is useful in this case, since it can be made to be either implicit or explicit (e.g. central difference), depending on the problem at hand. In displacement form, the generalized Newmark method first needs an update equation. Given $\Delta\tilde{\psi}$, and a previous iterate $\tilde{\tilde{\psi}}$, we compute an updated current iterate as

$$\tilde{\psi} = \tilde{\tilde{\psi}} + \Delta\tilde{\psi} \tag{3.4.18}$$

Then, we use $\tilde{\psi}$ to compute updated first and second time derivatives as follows

$$\begin{aligned}
\tilde{\ddot{\psi}} &= \frac{1}{\beta\Delta t^2}\left[\tilde{\psi} - \psi_n - \dot{\psi}_n\Delta t\right] - \frac{1-2\beta}{2\beta}\ddot{\psi}_n \\
\tilde{\dot{\psi}} &= \dot{\psi}_n + \Delta t\left[(1-\gamma)\ddot{\psi}_n + \gamma\tilde{\ddot{\psi}}\right] \\
&= \dot{\psi}_n + \Delta t\left[(1-\gamma)\ddot{\psi}_n + \frac{\gamma}{\beta\Delta t^2}\left[\tilde{\psi} - \psi_n - \dot{\psi}_n\Delta t\right] - \gamma\frac{1-2\beta}{2\beta}\ddot{\psi}_n\right]
\end{aligned} \tag{3.4.19}$$

where γ, β are the integration parameters for the Newmark method, and $\dot{\psi}_n$, $\ddot{\psi}_n$ are the first and second time derivatives from the previous time step. Note that, as $\Delta\tilde{\psi} \longrightarrow 0$, $\tilde{\psi} \longrightarrow \psi_{n+1}$, indicating that the current iterate has converged to the value at the next time step, step $n+1$.

We can simplify by noting that, from equation 3.4.19,

$$\begin{aligned}
\frac{\partial\dot{\psi}}{\partial\psi} &= \frac{\gamma}{\beta\Delta t} \\
\frac{\partial\ddot{\psi}}{\partial\psi} &= \frac{1}{\beta\Delta t^2}
\end{aligned} \tag{3.4.20}$$

We also make the following definitions, which define the tangent stiffness, damping, and mass matrices

$$\begin{aligned}
\frac{\partial F_{int}}{\partial\psi}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) &= K_t \\
\frac{\partial F_{int}}{\partial\dot{\psi}}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) &= C_t \\
\frac{\partial F_{int}}{\partial\ddot{\psi}}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) &= M_t
\end{aligned} \tag{3.4.21}$$

where K_t, C_t, and M_t denote the tangent stiffness, damping, and mass matrices. The tangent matrices are the derivatives of the internal force, but evaluated at the current Newton iteration. Substituting equations 3.4.20 and 3.4.21 into equation 3.4.16 yields

$$F_{int}(\psi, \dot{\psi}, \ddot{\psi}) = F_{int}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) + \left[K_t + \frac{\gamma}{\beta\Delta t}C_t + \frac{1}{\beta\Delta t^2}M_t\right]\Delta\psi \tag{3.4.22}$$

Finally, substituting equation 3.4.22 into equation 3.4.11 yields

$$\left[K_t + \frac{\gamma}{\beta\Delta t}C_t + \frac{1}{\beta\Delta t^2}M_t\right]\Delta\psi = F_{ext} - F_{int}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) = Res \tag{3.4.23}$$

Note that the right-hand side of equation 3.4.23 is the residual, or the difference between the external force and the internal force at the current Newton iteration. As convergence occurs, the residual goes to zero.

We derive explicit expressions for K_t, C_t, and M_t. We have

$$\begin{aligned} K_t &= \frac{\partial F_{int}}{\partial \psi}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) \\ &= \int_\Omega \nabla N^T \cdot \nabla N dx - \frac{2}{\rho c^2} \int_\Omega (\nabla \tilde{\dot{\psi}} \cdot \nabla N^T) N dx \end{aligned} \tag{3.4.24}$$

$$\begin{aligned} C_t &= \frac{\partial F_{int}}{\partial \dot{\psi}}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) \\ &= \frac{1}{c^2} \int_\Omega b \nabla N^T \cdot \nabla N dx - \frac{2}{\rho c^2} \int_\Omega (\nabla \tilde{\psi} \cdot \nabla N^T) N dx && (3.4.25) \\ &- \frac{1}{\rho c^4} B/A \int_\Omega \tilde{\ddot{\psi}} N^T N dx && (3.4.26) \\ & && (3.4.27) \end{aligned}$$

$$\begin{aligned} M_t &= \frac{\partial F_{int}}{\partial \ddot{\psi}}(\tilde{\psi}, \tilde{\dot{\psi}}, \tilde{\ddot{\psi}}) \\ &= \frac{1}{c^2} \int_\Omega N^T N dx - \frac{1}{\rho c^2} B/A \int_\Omega \tilde{\psi} N^T N dx \end{aligned} \tag{3.4.28}$$

where N is the vector of element shape functions.

For the full Newton method, these tangent matrices need to be reformed at each iteration of the Newton loop. The tangent damping and tangent stiffness matrices are *nonsymmetric*, since some terms involve products of shape functions with gradients of shape functions. However, we note that the *initial* tangent matrices are all symmetric, since at time $t = 0$, we have $\psi = 0$, $\dot{\psi} = 0$ and $\ddot{\psi} = 0$ by assumption. In that case, we have

$$K_{t_0} = \int_\Omega \nabla N^T \cdot \nabla N dx \tag{3.4.29}$$

$$C_{t_0} = \frac{1}{c^2} \int_\Omega b \nabla N^T \cdot \nabla N dx \tag{3.4.30}$$

$$M_{t_0} = \frac{1}{c^2} \int_\Omega N^T N dx \tag{3.4.31}$$

In this work we chose the Newton method for the nonlinear solution, and thus we could use any of the variants of this method, some requiring more and less frequent updating of the tangent matrices. In the case of the full Newton method, the nonsymmetric tangent matrices would need to be reformed at each iteration. In the initial Newton method, only the initial symmetric tangent needs to be formed. The numerical experiments conducted thus far indicate that excellent convergence behavior is observed even with the initial Newton method.

3.4.3. *Structural Coupling*

The second order equations of motion for the solid and the Kuznetsov equation for the fluid are

$$\begin{aligned}\rho_s u_{tt} - \nabla \cdot \sigma = f \\ \frac{1}{c^2}\frac{\partial^2 \psi}{\partial t^2} - \Delta\psi - \frac{1}{c^2}\frac{\partial}{\partial t}\left(b(\Delta\psi) + \frac{B/A}{2\rho c^2}\left(\frac{\partial\psi}{\partial t}\right)^2 + \frac{(\nabla\psi)^2}{\rho}\right) = 0\end{aligned} \tag{3.4.32}$$

Here u corresponds to the displacement of the structure, σ is the structural stress tensor, and subscripts s and f refer to solid and fluid, respectively. The equations of motion for the solid in equation 3.4.32 are written in the most general form, which could include both material and geometric nonlinearities. However, since we are only considering small structural displacements, these will be specialized to the linear elasticity equations.

In the case of linear acoustics, the boundary conditions on the fluid/solid interface (wet interface, which is designated by $\partial\Omega_{wet}$), are

$$\begin{aligned}\frac{\partial\psi}{\partial n} &= -\rho_f \dot{u}_n \\ \sigma_n &= -\dot{\psi}\hat{n}\end{aligned} \tag{3.4.33}$$

where $\hat{n}$ is the surface normal vector. These correspond to continuity of velocity and stress on the wet interface. In the case of nonlinear acoustics, the second condition is replaced by[98]

$$\sigma_n = -\hat{n}\left(\dot{\psi} + \frac{1}{c^2}\dot{\psi}^2 - \frac{1}{2}(\nabla\psi)^2 + b\Delta\psi\right) \tag{3.4.34}$$

The linear approximation of condition 3.4.34 is

$$\sigma_n = -\dot{\psi}\hat{n} \tag{3.4.35}$$

In,[77,31] numerical results were presented on the solution of Kuznetsov's equation, and the approximation 3.4.35 was used to convert from velocity potential to pressure as a post-processing step. In our case we also use this approximation as a post-processing step, and additionally, we use equation 3.4.35, rather than equation 3.4.34 to approximate the structural acoustic coupling. This is an additional approximation, but it is consistent with the previous studies.[77,31] Using relation 3.4.34 would lead to nonlinear boundary integral terms, and result in a nonsymmetric formulation.

The weak formulation of the coupled problem is constructed by multiplying the two partial differential equations in equation 3.4.32 by test functions and integrating by parts. Denoting by $V_s(\Omega_s)$ and $V_f(\Omega_f)$ the function spaces for the solid and fluid, respectively, we have the following weak formulation.

Find the mapping $(u, \psi): \quad [0, T] \to V_s(\Omega_s) \times V_f(\Omega_f)$ such that

$$\begin{aligned}
\int_{\Omega_s} \rho_s \ddot{u} w dx + \int_{\Omega_s} \sigma : \nabla^s w dx - \int_{\partial\Omega_{wet}} \sigma_n w ds = \int_{\Omega_s} f w dx + \int_{\partial\Omega_n} \sigma_n w ds \\
\frac{1}{c^2} \int_{\Omega_f} \ddot{\psi} \phi dx + \int_{\Omega_f} \nabla\psi \cdot \nabla\phi dx + \int_{\partial\Omega_{wet}} \frac{\partial\psi}{\partial n} \phi ds \\
+ \frac{b}{c^2} \int_{\Omega_f} \nabla\dot{\psi} \cdot \nabla\phi dx - \frac{B/A}{\rho c^4} \int_{\Omega_f} \ddot{\psi}\dot{\psi}\phi dx - \\
\frac{2}{\rho c^2} \int_{\Omega_f} \nabla\dot{\psi} \cdot \nabla\psi\phi dx = \int_{\partial\Omega_n} \frac{\partial\psi}{\partial n} \phi ds
\end{aligned} \tag{3.4.36}$$

$\forall w \in V_s(\Omega_s)$ and $\forall \phi \in V_f(\Omega_f)$, where $\partial\Omega_n$ is the portion of the solid and fluid boundaries that has applied loads, and f is used to denote body forces on the solid. Also, $\nabla^s = \frac{1}{2}\left(\nabla + \nabla^T\right)$ is the symmetric part of the gradient operator. If Dirichlet boundary conditions were applied to part of the structure, or if the fluid had a portion of its boundary subjected to Dirichlet conditions, then the Sobolev spaces $V_s(\Omega_s)$ and $V_f(\Omega_f)$ would be modified accordingly to correspond to spaces that have those same boundary conditions. We also note that in the integration on the wet interface, the normal is defined to be positive going from solid into the fluid.

Next, we insert the boundary conditions from equation 3.4.33, and we define $\sigma_n = g$ on the solid portion of $\partial\Omega_n$, and $\frac{\partial\psi}{\partial n} = -\rho_f u_n$ on the fluid portion of $\partial\Omega_n$. This leads to the following weak formulation. Find the mapping $(u, \psi): \quad [0, T] \to V_s(\Omega_s) \times V_f(\Omega_f)$ such that

$$\begin{aligned}
\int_{\Omega_s} \rho_s \ddot{u} w dx + \int_{\Omega_s} \sigma : \nabla^s w dx + \int_{\partial\Omega_{wet}} \dot{\psi} \hat{n} w ds = \int_{\Omega_s} f w dx + \int_{\partial\Omega_n} g w ds \\
\frac{1}{c^2} \int_{\Omega_f} \ddot{\psi} \phi dx + \int_{\Omega_f} \nabla\psi \cdot \nabla\phi dx - \rho_f \int_{\partial\Omega_{wet}} \dot{u}_n \phi ds \\
+ \frac{b}{c^2} \int_{\Omega_f} \nabla\dot{\psi} \cdot \nabla\phi dx - \frac{B/A}{\rho c^4} \int_{\Omega_f} \ddot{\psi}\dot{\psi}\phi dx - \\
\frac{2}{\rho c^2} \int_{\Omega_f} \nabla\dot{\psi} \cdot \nabla\psi\phi dx = -\rho_f \int_{\partial\Omega_n} \dot{u}_n \phi ds
\end{aligned} \tag{3.4.37}$$

$\forall w \in V_s(\Omega_s)$ and $\forall \psi \in V_f(\Omega_f)$. Equations 3.4.37 are a nonlinear system of equations, since the fluid wave equation is nonlinear.

Inserting the spatial discretizations $u = \sum u_i N_i$ and $\phi = \sum \phi_i N_i$ into equation 3.4.37 yields the following semi-discrete system of nonlinear ordinary differential equations in time

$$\begin{bmatrix} M_s & 0 \\ 0 & M_f \end{bmatrix} \begin{bmatrix} \Delta\ddot{u} \\ \Delta\ddot{\psi} \end{bmatrix} + \begin{bmatrix} C_s & L \\ -\rho_f L^T & C_f \end{bmatrix} \begin{bmatrix} \Delta\dot{u} \\ \Delta\dot{\psi} \end{bmatrix} + \begin{bmatrix} K_s & 0 \\ 0 & K_f \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} Res_s \\ Res_f \end{bmatrix} \tag{3.4.38}$$

where M_s, C_s, and K_s denote the mass, damping, and stiffness matrices for the solid, and M_f, C_f, and K_f denote the same for the fluid. The coupling matrices are denoted by L and L^T. Coupling between fluid and structure, and any damping in the fluid or solid separately, is accounted for by the damping matrices. The quantities Res_s and Res_f denote the residuals in the solid and fluid, respectively (recall equation 3.4.23).

$$\begin{aligned}
Res_s = F_s^{ext} - M_s\tilde{\ddot{u}} - C_s\tilde{\dot{u}} - L\tilde{\dot{\psi}} - K_s u \\
Res_f = F_f^{ext} - F_f^{int}(\tilde{\ddot{\psi}}, \tilde{\dot{\psi}}, \tilde{\psi})
\end{aligned} \tag{3.4.39}$$

Equation 3.2.8 is solved using Newton's method, in conjunction with the time discretization scheme that was introduced earlier. The nonlinear terms in the fluid wave equation are accounted for in the right-hand side in the initial Newton method, but for a full Newton update, the matrices M_f, C_f, and K_f would all need to be updated using equations 3.4.24, 3.4.27, and 3.4.28.

For the initial Newton method, equation 3.4.38 can be symmetrized in a number of ways. For example, the second equation can be multiplied by $\frac{-1}{\rho_f}$. This makes the system symmetric, but the matrices are indefinite.

To solve the coupled system of equations (3.2.8), we could either treat the 2×2 block system as a monolithic system of equations and integrate it directly, or we could use a staggered, loose coupling scheme. For the numerical examples presented next, we integrate the system directly.

Finally, we note that most numerical methods for absorbing boundary conditions in acoustics have been developed for the linear case. The development of absorbing boundary conditions for nonlinear acoustics is an important area of research, but we do not pursue that subject here. In this paper we use first-order absorbing boundary conditions of the form

$$\frac{\partial \psi}{\partial n} = -\frac{1}{c}\frac{\partial \psi}{\partial t} \tag{3.4.40}$$

This condition leads to an additional contribution to the matrix C_f from equation 3.4.38. Equation 3.4.40 is, or course, an additional approximation that neglects nonlinear terms. We mention that Cai[31] made a similar approximation when simulating nonlinear acoustic fields.

3.5. SA_eigen

The quadratic eigenvalue problem which we address in this solution method is given by the equation below.

$$\left(K + \lambda C + \lambda^2 M\right) \phi = 0 \tag{3.5.1}$$

where, K is the stiffness matrix,
C is a damping and coupling matrix, and
M is a mass matrix.

More specifically, for a structural acoustic system.

$$\left(\begin{bmatrix} K_s & 0 \\ 0 & K_a \end{bmatrix} + \lambda \begin{bmatrix} C_s & L \\ -\rho_a L^T & C_a \end{bmatrix} + \lambda^2 \begin{bmatrix} M_s & 0 \\ 0 & M_a \end{bmatrix} \right) \begin{bmatrix} \phi_s \\ \phi_a \end{bmatrix} = 0 \tag{3.5.2}$$

Here the subscripts refer to structural or acoustic domains, ρ_a is the density of the fluid and L is a coupling matrix. Note that for this formulation, ϕ_a represents the acoustic velocity potential, which relates to the time derivative of the acoustic pressure, $\phi_a = \nabla \dot{u}_a$.

If C contains only coupling terms, then it is skew. Readers will recognize this as the eigenvalue problem for a spinning structure with real eigenvalues [54]. However, if there is additional damping in the system, as from ρC damping on the acoustic domain, then C is of mixed symmetry, and the eigenvalues and eigenvectors are complex. The stiffness matrix is symmetric positive semi-definite, while the mass matrix is symmetric positive definite.

Table 3-2. – Potential Basis Functions for Subdomain Reduction.

Name	Basis Function
Free-Free modes	The unconstrained eigenvectors of each subdomain are computed and used as the columns of T. When the number of columns in T equals the number of rows, this basis is complete.
Craig-Bampton	The eigenvectors of each subdomain are computed with the interface fixed. These eigenvectors are supplemented with constraint modes computed by fixing all the interface degrees of freedom except one. That dof receives a unit static deformation. This method has been shown to converge near optimally for structure/structure interactions.

While various methods are available for solving the generalized, linear eigenvalue problem,[1] solution of the quadratic eigenvalue problem is more challenging. The approach followed here is to transform the problem into a reduced space, solve the corresponding dense matrix system completely, and prolongate to the original space. The challenge, of course, is to properly choose that space.

In general, if the eigenvector, ϕ, can be written in terms of generalized coordinates, q, then this approach may be taken. For a given transformation matrix, T, which determines ϕ given q, we have the following.

$$\phi = Tq \tag{3.5.3}$$

$$T^\dagger \left(K + \lambda C + \lambda^2 M\right) Tq = 0 \tag{3.5.4}$$

$$\left(\tilde{k} + \lambda \tilde{c} + \lambda^2 \tilde{m}\right) q = 0 \tag{3.5.5}$$

Note that the only restriction on T is that we may adequately write $\phi = Tq$. In other words, T must span the space of the eigenvectors. In particular, T need not be unitary or even orthogonal. However, for the transformation to be useful for a model reduction, there must be many fewer columns than rows in T. Note that $T^\dagger$ is the transpose, complex conjugate of T, and that the left and right eigenvectors of equation 3.5.2 are complex conjugates of each other.

The structural/acoustics problem may be viewed as a two subdomain problem.[2] There are a variety of basis functions that have been examined for connecting such subdomains. Two common sets are listed in Table 3-2.

We here investigate only the free-free method. Though this method has proved to converge slowly for structure/structure problems, the coupling between the structural and acoustic domains is often weak. This may be adequate. For the problems of interest, a full Craig-Bampton type solution is almost certainly overkill, and will result in a dense matrix too large for standard solution methods. We may find it advantageous to augment the free-free modes by adding basis functions near the surface. Some thoughts that have been considered include the following.

- A uniform pressure mode could be added to both the acoustic and structural responses.

[1]The generalized linear eigenvalue problem is $(K - \lambda M)\phi = 0$.
[2]There is no requirement that each subdomain be topologically connected in any special way.

- We could consider the static acoustic modes that are generated by the deformations of the structural eigen analysis. We anticipate that the structural deformations will have a larger control over acoustic modes, so we may not need to be as concerned about the impact of the acoustic pressures on the structure, but we may want to include these too. Could a subset of modes be identified that would aid in model completeness and convergence?
- Spline or boundary expansions are possible.

3.5.1. *Quadratic Modal Superposition*

Consider the system

$$M\ddot{u} + C\dot{u} + Ku = f(t) \tag{3.5.6}$$

where M, C, and K are the mass, damping, and stiffness matrices. Standard methods may be used to solve the eigenvalue equation derived from 3.5.6 only in the case where the eigenvectors of K and M also diagonalize C (as in proportional damping for example). In practice this never happens. For a general damping matrix, no procedures are available to directly solve the eigenvalue problem. For an excellent survey article on quadratic eigenvalue systems, see the article by Tisseur.[129]

However, the second order system may be transformed to a larger, first order system which does have a known solution. We *linearize* the system as follows. Define,

$$w = \begin{bmatrix} \dot{u} \\ u \end{bmatrix} \tag{3.5.7}$$

If we consider the eigenvalue problem corresponding to equation 3.5.6, we would set the right-hand side $f(t)$ to zero. Then, there are many options for the linearization, but the one chosen for QEVP is

$$\begin{bmatrix} M & \mathbf{0} \\ \mathbf{0} & K \end{bmatrix} w = \begin{bmatrix} \mathbf{0} & M \\ -M & -C \end{bmatrix} \dot{w} \tag{3.5.8}$$

We assume a solution of the form $w = \phi e^{\lambda t}$, and arrive at the eigenvalue problem,

$$A\phi = \lambda B\phi \tag{3.5.9}$$

where

$$A = \begin{bmatrix} M & \mathbf{0} \\ \mathbf{0} & K \end{bmatrix}, \tag{3.5.10}$$

and

$$B = \begin{bmatrix} \mathbf{0} & M \\ -M & -C \end{bmatrix} \tag{3.5.11}$$

Equation 3.5.9 yields the "right" eigenvectors. As is seen later, we also need the "left" eigenvectors, which correspond to the eigenvalue problem,

$$\psi^\dagger A = \lambda \psi^\dagger B \tag{3.5.12}$$

We denote the left eigenvectors as ψ_i to distinguish them from the right eigenvectors ϕ_i.

3.5.2. *Diagonalization and Modal Superposition*

Symmetric system matrices are always diagonalizable, using the matrix formed by their eigenvectors. However, when nonsymmetric matrices, such as those of equation 3.5.8, may be *impossible* to diagonalize. This has significant implications for modal superposition techniques, since if A and B cannot be diagonalized by pre and post multiplying by matrices of eigenvectors, then the reduced (modal) equations of motion will be coupled. The primary advantages of modal superposition would be lost.

As discussed in the literature,[129,117,88] one case where the matrices A and B are diagonalizable is if the eigenvalues are distinct. If there are repeated eigenvalues, then the matrix is still diagonalizable, as long as the eigenvectors corresponding to repeated eigenvalues are linearly independent. This can be summarized by the theory of geometric and algebraic multiplicities of eigenvalues,

- The *algebraic multiplicity* of an eigenvalue is defined as the number of times that this eigenvalue is repeated in the list of eigenvalues of the matrix.
- The *geometric multiplicity* of an eigenvalue is the dimension of the space spanned by its eigenvectors. Thus, for an eigenvalue with an algebraic multiplicity of 2, the geometric multiplicity would be 2 if the corresponding eigenvectors are linearly independent, and 1 if they are linearly dependent.
- An $n \times n$ matrix is diagonalizable if and only if the geometric multiplicity is equal to the algebraic multiplicity for every eigenvalue λ.

In short, for the matrix to be diagonalizable, the eigenvectors corresponding to repeated eigenvalues must be linearly independent. If the eigenvalues are all distinct, then the matrix is always diagonalizable.

It is also interesting to discuss the circumstances under which the eigenvalues and eigenvectors of A and B come in complex conjugate pairs. When this is the case, significant savings in storage and computational time can be achieved. The general rule is simple to prove.[97] If the entries in a matrix are all real-valued, then any complex eigenvalues or eigenvectors that arise must come in complex conjugate pairs. To prove this, we note that for a matrix with all real- valued entries, the determinant must be a real number. On the other hand, the determinant is also equal to the product of the eigenvalues. Thus, if some eigenvalues are complex, the only way that the product

$$\det(A) = \lambda_1 \lambda_2 ... \lambda_n \tag{3.5.13}$$

can be a real number is if all complex eigenvalues have a conjugate pair. For example, if λ_n and λ_{n+1} are complex conjugates, then we have

$$\lambda_n \lambda_{n+1} = (\lambda_n^r + j\lambda_n^i) * (\lambda_n^r - j\lambda_n^i) = \left[\lambda_n^r\right]^2 + \left[\lambda_n^i\right]^2 \tag{3.5.14}$$

The last expression after the equal sign is a real number. We can also conclude that if a matrix has any complex entries, then the eigenvalues and eigenvectors are not necessarily complex conjugates.

To diagonalize A and B, we define a matrix corresponding to the right-eigenvectors that are computed from equation 3.5.9.

$$W = [\phi_1 \phi_2 ... \phi_{2n}] \tag{3.5.15}$$

We can also define a matrix corresponding to the left-eigenvectors from equation 3.5.12.

$$U = [\psi_1 \psi_2 ... \psi_{2n}] \tag{3.5.16}$$

Representing the solution as $w = \sum_{i=1}^{2n} z_i \phi_i$, and the loading as,

$$g(t) = \begin{bmatrix} f(t) \\ 0 \end{bmatrix} \tag{3.5.17}$$

we have[129]

$$-\alpha_i z_i(t) + \beta_i \dot{z}_i(t) = \psi_i^\dagger g(t) \tag{3.5.18}$$

where $\alpha_i = \psi_i^\dagger A \phi_i$ and $\beta_i = \psi_i^\dagger B \phi_i$. When modes are mass normalized, $\beta_i = 1$ and $\alpha_i = \lambda_i$. We note that the † symbol represents a conjugate transpose, not a transpose. This is a complex-valued uncoupled scalar equation for each degree of freedom in the system, which can be integrated in time. We have no general solution of the original second order system. Superposition must be performed on the linearized system. This is a first order system of differential equations. Different time integration methods are needed.

Time Domain Superposition

Equation 3.5.18 can be integrated numerically, using first-order time integrators. However, another approach is to use the analytical solution.

$$z_i(t) = \int_0^t \psi_i^* g(\tau) e^{-\lambda_i(t-\tau)} d\tau \tag{3.5.19}$$

Finally, given the solution for each $z_i(t)$, we compute $w = \sum_{i=1}^{2n} z_i \phi_i$, and extract the solution $u(t)$ from the upper half of $w(t)$. We note that in the time domain, the final solution $w(t)$ must be real-valued, even though both ϕ_i and z_i are, in general complex. It is easy to show that this is the case. First, as noted earlier, we recall that the eigenvectors ϕ_i come in complex conjugate pairs. Equation 3.5.18 implies that z_i also comes in conjugate pairs. We note that

$$w = \sum_{i=1}^{2n} z_i \phi_i = \sum_{i=1}^{n} \left[z_i \phi_i + \bar{z}_i \bar{\phi}_i \right] \tag{3.5.20}$$

Noting that $z_i \phi_i + \bar{z}_i \bar{\phi}_i$ is a real number, we see that the total summation is also a real number.

Frequency Domain Superposition

For the frequency domain solution, we assume a time-harmonic loading and response.

$$g(t) = g_0 e^{i\omega_{ex} t} \tag{3.5.21}$$
$$z_i(t) = z_i e^{i\omega_{ex} t} \tag{3.5.22}$$
$$\tag{3.5.23}$$

where ω_{ex} is the frequency of the external excitation, and g_0 is a spatial vector of loadings at that frequency. Substituting these relations into equation 3.5.18, we obtain the equations for complex modal frequency response

$$\left[-\alpha_i + i\omega\beta_i \right] z_i = \psi_i^\dagger g_0 \tag{3.5.24}$$

This can also be written as,

$$z_i = \frac{\psi_i^\dagger g_0}{-\alpha_i + i\omega\beta_i} \tag{3.5.25}$$

We note that the denominator will go to zero if $\alpha_i = i\omega\beta_i$, as is expected, in the case of resonance. A standard approach[25] of stabilizing the solution near resonances is to add a small amount of modal damping. In state space, this corresponds to a adding a real-valued term in the denominator of equation 3.5.25. Thus, when $\alpha_i = i\omega\beta_i$ this additional term would prevent a singular response. This additional real term takes the form

$$z_i = \frac{\psi_i^\dagger g_0}{\gamma_i - \alpha_i + i\omega\beta_i} \tag{3.5.26}$$

where γ_i is the modal damping, and is a real number.

As before, the solution of the displacement degrees of freedom is a superposition of modal solutions.

$$\begin{aligned} w(\omega) &= \sum_{i=1}^{2n} z_i(\omega)\phi_i && (3.5.27) \\ &= \sum_{i=1}^{2n} \frac{\phi_i \psi_i^\dagger g_0}{\gamma_i - \alpha_i + i\omega\beta_i} && (3.5.28) \end{aligned}$$

3.5.3. Theory for modal superposition with sa_eigen

In the case of the **sa_eigen** solution case, the eigenvalue problem is solved in a reduced space. Recalling equation 3.5.6, and the transformation $u = T\hat{u}$, we can transform equation 3.5.6 into a reduced space as

$$\hat{m}\ddot{\hat{u}} + \hat{c}\dot{\hat{u}} + \hat{k}\hat{u} = \hat{f} \tag{3.5.29}$$

where $\hat{m} = T^T M T$, $\hat{c} = T^T C T$, $\hat{k} = T^T K T$, and $\hat{f} = T^T f$. We note that the superscript ˆ is used from here on to denote the reduced space. If we then define

$$\hat{q} = \begin{bmatrix} \hat{u} \\ \dot{\hat{u}} \end{bmatrix} \tag{3.5.30}$$

As was done for the full system for the QEVP method, we project this into the first order system[3].

$$\hat{A}\hat{q} - \hat{B}\hat{q} = g(\hat{t}), \tag{3.5.31}$$

$$\hat{A} = \begin{bmatrix} 0 & I \\ -\hat{k} & -\hat{c} \end{bmatrix}, \hat{B} = \begin{bmatrix} I & 0 \\ 0 & \hat{m} \end{bmatrix}, \hat{g} = \begin{bmatrix} 0 \\ -\hat{f} \end{bmatrix}.$$

Assuming a solution of the form $\hat{q} = \hat{\phi} e^{\lambda t}$, we arrive at the eigenvalue problem

$$\hat{A}\hat{\phi} = \lambda \hat{B}\hat{\phi} \tag{3.5.32}$$

where we emphasize that $\hat{\phi}$ is in the state-space form of the reduced problem. This eigenvalue problem is solved with the DGGEV algorithm from LAPACK.

Once the eigenvalue problem 3.5.32 is solved, methods of the previous section can be applied for solution of the scalar modal equations of the linearized system and projection back to the reduced space and finally to physical space.

[3] also known as a state space solution

We transform equation 3.5.31 into the frequency domain.

$$\hat{A}\hat{q} - i\omega_{ex}\hat{B}\hat{q} = \hat{g}(\omega) \tag{3.5.33}$$

where ω_{ex} is the frequency of the external excitation. We assume that the solution can be represented as $\hat{q} = \sum_{i=1}^{2n} \hat{z}_i\hat{\phi}_i$. Substituting this into equation 3.5.33, and pre-multiplying by the left eigenvectors $\hat{\psi}_i$, we obtain

$$\hat{\alpha}_i\hat{z}_i - i\hat{\beta}_i\omega_{ex}z_i = \hat{\psi}_i{}^{\dagger}\hat{g} \tag{3.5.34}$$

where $\hat{\alpha}_i = \hat{\psi}_i{}^{\dagger}\hat{A}\hat{\phi}_i$ and $\hat{\beta}_i = \hat{\psi}_i{}^{\dagger}\hat{B}\hat{\phi}_i$. This scalar equation, 3.5.34 can be solved for $\hat{z}_i$. The solution in reduced space, $\hat{q}$ can be obtained from $\hat{q} = \sum_{i=1}^{2n} \hat{z}_i\hat{\phi}_i$. Given $\hat{q}$, $\hat{u}$ can be extracted from the upper half of $\hat{q}$, as per equation 3.5.30. Finally, once $\hat{u}$ is known, the original solution u can be computed from the relation $u = T\hat{u}$.

3.5.4. *Discussion of Eigenvectors and Superposition*

There are several important points to consider for the eigenvectors of this problem.

- The left and the right eigenvectors of the linearized system diagonalize the characteristic matrices A and B. However, the eigenvectors do *not* diagonalize the matrices of the original second order equation, 3.5.6. This means that the modal equations are coupled in the second order system, and most simplifications for superposition are available only on the linearized, first order system.
- The left eigenvectors can be computed from the solution of the transposed equation. Thus, for symmetric systems, left and right eigenvectors are identical.
- Eigenvectors of the linearized, nonsymmetric systems are often not normalized as expected. In many cases the eigenvectors are not even completely orthogonal, even when they may be linearly independent.

3.5.5. *Notes on Implementation*

Some questions are answered next on the implementation of the superposition algorithm with regard to the specific linearizations used in the Anasazi and sa_eigen solvers.

1. Can the state-space left and/or right eigenvectors be decomposed into a vector in one half and then that same vector multiplied by the eigenvalue in the other half?
2. Does the nonzero part of the state-space force vector occupy the top or bottom half of the vector, and does it have a minus sign in front of it?
3. Under what circumstances are there relations between the left and right eigenvectors, such as $\phi_{left} = \phi_{right}$ or $\phi_{left} = \left(\phi_{right}\right)^{\dagger}$?

The answers to any of these questions depends on the specific linearization of interest. Here we examine only 2 linearizations, which have been considered earlier, and which will be repeated here for convenience.

$$\begin{bmatrix} M & \mathbf{0} \\ \mathbf{0} & K \end{bmatrix} w = \lambda \begin{bmatrix} 0 & M \\ -M & -C \end{bmatrix} w \tag{3.5.35}$$

$$\begin{bmatrix} 0 & I \\ -K & -C \end{bmatrix} w = \lambda \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} w \tag{3.5.36}$$

For the first question, we consider the right and left eigenvectors separately. For the right eigenvectors, a simple substitution reveals that the right eigenvector for equation 3.5.35 can be decomposed as

$$w = \begin{bmatrix} \lambda u \\ u \end{bmatrix} \tag{3.5.37}$$

whereas the second linearization (equation 3.5.36) has right eigenvectors that decompose in the opposite way.

$$w = \begin{bmatrix} u \\ \lambda u \end{bmatrix} \tag{3.5.38}$$

For the left eigenvectors, we write the equations corresponding to the left eigenvectors as

$$\left[w_t^T w_b^T\right] \begin{bmatrix} M & \mathbf{0} \\ \mathbf{0} & K \end{bmatrix} = \lambda \left[w_t^T w_b^T\right] \begin{bmatrix} \mathbf{0} & M \\ -M & -C \end{bmatrix} \tag{3.5.39}$$

$$\left[w_t^T w_b^T\right] \begin{bmatrix} 0 & I \\ -K & -C \end{bmatrix} = \lambda \left[w_t^T w_b^T\right] \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} w \tag{3.5.40}$$

Multiplying out the terms in equation 3.5.39, we find that

$$w_t^T M = \lambda w_b^T M \tag{3.5.41}$$

which, for nonsingular M, yields

$$w_t = \lambda w_b \tag{3.5.42}$$

Thus, for the linearization in equation 3.5.35, the left eigenvectors can be decomposed in a similar manner as the right eigenvectors when the mass matrix is nonsingular.

Multiplying out the terms in equation 3.5.40, we find that

$$w_b^T K = \lambda w_t^T \tag{3.5.43}$$

Or, for symmetric K,

$$K w_b = \lambda w_t \tag{3.5.44}$$

Thus, for the linearization described by equation 3.5.36, the left eigenvectors cannot be decomposed as the right eigenvectors were.

When forces are present in the system, we can rewrite equations 3.5.35 and 3.5.36 as

$$\begin{bmatrix} M & \mathbf{0} \\ \mathbf{0} & K \end{bmatrix} w - \begin{bmatrix} 0 & M \\ -M & -C \end{bmatrix} \dot{w} = \begin{bmatrix} 0 \\ f \end{bmatrix} \tag{3.5.45}$$

$$\begin{bmatrix} 0 & I \\ -K & -C \end{bmatrix} w - \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} \dot{w} = \begin{bmatrix} 0 \\ -f \end{bmatrix} \tag{3.5.46}$$

Thus, for both linearizations 3.5.35 and 3.5.36 the state-space force vector has a zero top half, and for linearization 3.5.35 the non-zero bottom half is multiplied by a negative sign. This answers the second question above.

To answer the third question, we first consider the results given in Table 1.1 of.[129] In this table, relationships between the left and right eigenvectors are given for various symmetry relations of M, C, and K. In particular, property $P7$ from this table states that if M, K are Hermitian, $C = -C^\dagger$ is skew-Hermitian, and M is positive definite, then if x is a right eigenvector of λ, then x is also a left eigenvector of $-\lambda^\dagger$. Since we only consider real-valued matrices, we expect the eigenvalues of the systems of interest to be imaginary, and thus $-\lambda^\dagger = \lambda$. Thus, property $P7$ states that the left and right eigenvectors of λ are the same. The results in this table define the left and right eigenvectors as follows

$$\lambda^2 Mu + \lambda Cu + Ku = 0 \tag{3.5.47}$$

$$w^\dagger \lambda^2 M + w^\dagger \lambda C + w^\dagger K = 0 \tag{3.5.48}$$

for right and left eigenvectors u and w, respectively. By taking the conjugate transpose of equation 3.5.47, and noting that $C = -C^\dagger$ and $-\lambda^\dagger$, we obtain

$$u^\dagger \lambda^2 M + u^\dagger \lambda C + u^\dagger K = 0 \tag{3.5.49}$$

from which the result $P7$ from Table 1.1 in[129] is obtained.

We note that the results from Table 1.1[129] are with respect to the quadratic eigenvalue problem, not the linearized versions. Since equations 3.5.47 and 3.5.48 could be linearized in some ways, we would expect the conclusions to change when we go to the linearized problem. For example, we again consider the case when M, K are Hermitian, $C = -C^\dagger$ is skew-Hermitian, and M is positive definite. With these conditions on M, K, and C, we consider the linearizations given by equations 3.5.35 and 3.5.36, which can be written concisely as

$$Au = \lambda Bu \tag{3.5.50}$$

In the case of equation 3.5.35, we have that A is symmetric, whereas B is skew-symmetric. In the case of equation 3.5.36, we have that A is nonsymmetric, and B is symmetric. If we take the conjugate transpose of equation 3.5.50, we have the corresponding equation for the left eigenvectors

$$u^\dagger A^\dagger = u^\dagger \lambda^\dagger B^\dagger \tag{3.5.51}$$

For linearization 3.5.35, we have $A^\dagger = A$, $B^\dagger = -B$, and $\lambda^\dagger = -\lambda$. This gives

$$u^\dagger A = u^\dagger \lambda B \tag{3.5.52}$$

which implies that the left and right eigenvectors of linearization 3.5.35 coincide.

In the case of equation 3.5.36, we have that A is nonsymmetric and B is symmetric. Thus, when we take the conjugate of equation 3.5.50, we have

$$u^\dagger A^\dagger = u^\dagger \lambda^\dagger B^\dagger \tag{3.5.53}$$

which, from symmetry conditions, reduces to

$$u^\dagger A^\dagger = -\lambda u^\dagger B \tag{3.5.54}$$

Thus, since A is nonsymmetric, no relation can be deduced between the left and right eigenvectors.

Similar conclusions can be drawn about a different version of equation 3.5.35. If we multiply the lower equation by -1, we obtain

$$\begin{bmatrix} M & \mathbf{0} \\ \mathbf{0} & -K \end{bmatrix} w = \lambda \begin{bmatrix} 0 & M \\ M & C \end{bmatrix} w \tag{3.5.55}$$

or $Aw = \lambda Bw$. Since $C = -C^\dagger$, the matrix B is nonsymmetric. Then, taking conjugate transposes of both sides of equation 3.5.55, we see that we cannot draw conclusions about relations between the left and right eigenvectors. This is the same problem seen in equation 3.5.54.

3.5.6. Complex Eigenvector Orthogonalization

Let's assume that there is a complete set of eigenvectors (no Jordan blocks). An eigenvalue of multiplicity m has an m dimensional eigenspace. Some solvers, such as DGGEV do not generate an orthonormal bases for these subspaces. If such orthogonalization is required, the procedure in Figure 3-5 may be followed to orthogonalize two eigenvectors with a common eigenvalue.

Given two modes with a common eigenvalue, λ, and with left and right eigenvectors, ψ_i and ϕ_j, we orthogonalize with respect to a matrix B.

$$\psi_1^\dagger B\phi_1 = \beta_{11} \tag{3.5.56}$$
$$\psi_1^\dagger B\phi_2 = \beta_{12} \tag{3.5.57}$$
$$\psi_2^\dagger B\phi_1 = \beta_{21} \tag{3.5.58}$$

We modify ψ_2 and ϕ_2 to ensure that $\beta_{12} = \beta_{21} = 0$. Let $\hat{\psi}$ be the corrected eigenvector.

$$\hat{\psi}_2 = \psi_2 - \epsilon\psi_1$$

We require that $\hat{\psi}_2^\dagger B\phi_1 = 0$. Then,

$$0 = \hat{\psi}_2^\dagger B\phi_1 \tag{3.5.59}$$
$$= (\psi_2 - \epsilon\psi_1)^\dagger B\phi_1 \tag{3.5.60}$$
$$= \beta_{21} - \epsilon\beta_{11} \tag{3.5.61}$$

Thus,

$$\hat{\psi}_2 = \psi_2 - \frac{\beta_{21}}{\beta_{11}}\psi_1 \tag{3.5.62}$$

For the right eigenvector,

$$\hat{\phi}_2 = \phi_2 - \frac{\beta_{12}}{\beta_{11}}\phi_1 \tag{3.5.63}$$

Figure 3-5. – Complex EigenVector orthogonalization.

3.6. Modal Augmentation with Residual Vectors

The **residual_vectors** solution method Modal truncation augmentation (MTA)[46] provides a method to represent the modes not retained in the eigendecomposition. It is particularly useful in component mode synthesis approaches where multiple models are joined together. In NASTRAN, MTA vectors are referred to as ‘residual vectors’. The theory of MTA[46] is established. We use the following terminology:

N	Number of degrees of freedom
nev	Number of retained eigenvalues/eigenvectors from the `eigen` solution
nf	Number of applied forces and/or moments
$\boldsymbol{M}$	Mass matrix of size N×N
$\boldsymbol{K}$	Stiffness matrix of size N×N
Φ	Matrix with eigenvectors as columns, size N×(nev)
Ω^2	Diagonal matrix of eigenvalues, size (nev)×(nev)
$\boldsymbol{R}_0$	Applied spatial load vector, size N×(nf)
$\boldsymbol{R}_s$	Modally represented spatial load vector, size N×(nf)
$\boldsymbol{R}_t$	Force truncation vector, size N×(nf)
$\boldsymbol{X}$	Static displacements due to applied loads, size N×(nf)
$\bar{\omega}^2$	Diagonal matrix of reduced eigenvalues, size (nf)×(nf)
$\bar{\boldsymbol{Q}}$	Matrix of reduced eigenvectors, size (nf)×(nf)
$\boldsymbol{P}$	Matrix of modal truncation (residual) vectors, size N×(nf)

The algorithm for computing MTA vectors is:

1. Solve the generalized eigenvalue problem

$$\boldsymbol{K}\Phi = \boldsymbol{M}\Phi\Omega^2$$

   for `nev` eigenvalues and eigenvectors. This is done by first specifying `eigen` in a multicase solution procedure.

2. Compute the force truncation vector

$$\boldsymbol{R}_t = \boldsymbol{R}_0 - \boldsymbol{R}_s = \boldsymbol{R}_0 - \boldsymbol{M}\Phi\Phi^T\boldsymbol{R}_0.$$

3. Compute the static displacements $\boldsymbol{X}$ due to the force truncation vector $\boldsymbol{R}_t$ by solving $\boldsymbol{K}\boldsymbol{X} = \boldsymbol{R}_t$.

4. If rigid body modes are present, orthogonalize $\boldsymbol{X}$ to them. The optional input `nrbms` allows the user to specify the number of rigid body modes present.

5. Form the reduced matrices $nf \times nf$,

$$\bar{\boldsymbol{K}} = \boldsymbol{X}^T\boldsymbol{K}\boldsymbol{X}, \quad \bar{\boldsymbol{M}} = \boldsymbol{X}^T\boldsymbol{M}\boldsymbol{X}.$$

6. Solve the reduced generalized eigenvalue problem $\bar{\boldsymbol{K}}\bar{\boldsymbol{Q}} = \bar{\boldsymbol{M}}\bar{\boldsymbol{Q}}\bar{\omega}^2$

7. Form the modal truncation (residual) vectors: $\boldsymbol{P} = \boldsymbol{X}\bar{\boldsymbol{Q}}$

8. Construct the pseudo modal set: $\tilde{\Phi} = [\Phi|\boldsymbol{P}]$.

In Sierra-SD, the multi-case solution strategy is:

1. Solve the eigenvalue problem
2. For each column of $\boldsymbol{R}_0$, solve a statics problem
3. Solve a `residual_vectors` problem to form the pseudo modal set

3.7. Wet Modes or Added Mass

Analysts want to compute the structural normal modes for a structure partially submerged in a fluid. In appropriate approximations, this may be analyzed as a real eigen problem of the structure with added mass on the wetted surface.

Fluid loading of the real eigenvalue problem is performed by separating the solution domain into structural and acoustic regions. A real eigen analysis is performed on the acoustic domain which generates a mass loading correction for a subsequent real eigen analysis of the structure.

3.7.1. Case I - matching meshes at wet interface

After finite element discretization, a submerged coupled structural acoustic system obeys the following discrete formulation.

$$-\omega^2 \begin{bmatrix} M_s & 0 \\ 0 & \frac{-1}{\rho_f} M_f \end{bmatrix} \begin{bmatrix} u \\ \phi \end{bmatrix} + i\omega \begin{bmatrix} C_s & L \\ L^T & \frac{-1}{\rho_f} C_f \end{bmatrix} \begin{bmatrix} u \\ \phi \end{bmatrix} +$$

$$\begin{bmatrix} K_s & 0 \\ 0 & \frac{-1}{\rho_f} K_f \end{bmatrix} \begin{bmatrix} u \\ \phi \end{bmatrix} = \begin{bmatrix} f_s \\ f_a/\rho_f \end{bmatrix} \tag{3.7.1}$$

where M_s, C_s, and K_s denote the mass, damping, and stiffness matrices for the solid,[4] M_f, C_f, and K_f denote the same for the fluid, f_s and f_a denote loadings on the structure and fluid, and u and ϕ are the structural displacement and acoustic velocity potential, respectively. The coupling matrices are denoted by L and L^T. C_f may represent a non-reflecting boundary condition on the exterior of the fluid. Coupling between fluid and structure is accounted for by the matrices L and L^T. Due to the presence of the damping terms, this eigenvalue problem is *quadratic*. In the special case $C_s = C_f = 0$, the system is called *gyroscopic* since the eigenvalues are real valued, even though a damping matrix is present.

The goal of the added mass approach is to simplify equation (3.7.1) by considering only the incompressible limit. This can be achieved by taking the limit $c_f \to \infty$, where c_f is the speed of sound in the fluid. The latter condition implies an incompressible fluid, which has infinite sound speed. It is important to note that these limits are only applied to the acoustic equation in the system (3.7.1), and not the structural equation. Since we are only interested in eigen analysis, we set $f_s = f_a = 0$ for the remainder of this note.

If we consider the limiting condition $c_f \to \infty$ applied to the second equation in the system (3.7.1), we see that the term $\frac{\omega^2}{\rho_f} M_f \phi$ will vanish, since the acoustic mass matrix M_f has a factor of $\left(\frac{1}{c_f}\right)^2$ built into it.

[4]In a ship floating in water, the structural stiffness matrix, K_s will typically contain 6 zero energy modes. Addition of buoyancy terms converts three of these to bounce, roll and pitch modes, but three singularities typically remain.

Similarly, as $c_f \to \infty$ the fluid damping, due to either an exterior boundary condition or infinite elements, vanishes. For absorbing boundaries, this can be seen by considering the corresponding damping matrix

$$C_{f_{ij}} = \frac{1}{c_f} \int_{\partial\Omega_e} N_i N_j d\Omega_e \tag{3.7.2}$$

where the integral is evaluated over the exterior boundary $\partial\Omega_e$, and N_i, N_j are the standard finite element shape functions evaluated over Ω_e. Thus, the term C_f has a factor of $\frac{1}{c_f}$ built in, which implies that it can also be neglected. Physically, this implies that an incompressible fluid provides no radiation damping. For infinite elements, the damping matrix is different than absorbing boundaries, but it is still pre-multiplied by $\frac{1}{c_f}$.

$$C_{f_{ij}} = \frac{1}{c_f} \int_{\Omega_e} D\mathbf{N}_i \nabla\mu \cdot \nabla\mathbf{N}_j - \mathbf{N}_i\mathbf{N}_j \nabla D \cdot \nabla\mu - D\mathbf{N}_j \nabla\mathbf{N}_i \cdot \nabla\mu dV \tag{3.7.3}$$

where $\mathbf{N}_i$, μ, and D are components of infinite element shape functions, and here the integral extends over the entire exterior domain Ω_e instead of being on the boundary. Again, due to the pre-multiplication of $\frac{1}{c_f}$, we can neglect the infinite element damping matrix for incompressible fluids.

Additionally, we neglect structural damping and set $C_s = 0$. Simplifying the second equation in the system (3.7.1) in these ways yields,

$$\phi = i\omega\rho_f K_f^{-1} L^T u \tag{3.7.4}$$

This also implies that

$$i\omega\phi = -\omega^2 \rho_f K_f^{-1} L^T u \tag{3.7.5}$$

If we define $\lambda = \omega^2$, and substitute the previous results into the first equation in the system (3.7.1), we obtain

$$-\lambda \left[M_s + \rho_f L K_f^{-1} L^T \right] u + K_s u = 0 \tag{3.7.6}$$

The added mass matrix is

$$M_a = \rho_f L K_f^{-1} L^T \tag{3.7.7}$$

To make the acoustic stiffness matrix K_f invertible, most practitioners assign Dirichlet boundary conditions $p = 0$ on the exterior surface.[5] Also, standard practice is to mesh the fluid to the extent of one or two structural diameters away from the structure. As one takes more and more fluid, the eigenvalues should converge to fixed values (although not precisely the same values as would be obtained from a full complex eigen solution).

As an alternative to the Dirichlet boundary condition, one can use the spherical absorbing condition, not the plane wave condition from equation 3.7.2. The spherical condition is more accurate, and since it contributes an extra term to the stiffness matrix, it eliminates the need for the Dirichlet boundary condition. This term takes the form

$$K_{spherical_{ij}} = \frac{1}{R} \int_{\partial\Omega_e} N_i N_j d\Omega_e \tag{3.7.8}$$

where R is the radius of curvature of the absorbing domain, and N_j is a shape function on the exterior (absorbing) boundary of the surface. This term would then get appended to the acoustic stiffness matrix K_f, rendering it nonsingular, without the need for the Dirichlet boundary condition.

Equation (3.7.6) is an eigenvalue problem in terms of structural unknowns only. For both absorbing boundaries and infinite elements, the matrix M_a is real-valued, and independent of frequency. In the case

[5]Throughout further discussions, we assume that K_f is symmetric, positive definite.

of either absorbing boundaries or simple Dirichlet boundary conditions, it is also symmetric, and thus is in the form of a standard eigenvalue problem that will yield real-valued modes. The eigen solver typically requires an symmetric positive definite capacitance matrix, M. The linear solver must still address issues with singular K_s.

For infinite elements, however, K_f is nonsymmetric, and thus the matrix M_a is also nonsymmetric. In general, this will lead to complex modes, which are undesirable for added mass calculations. Thus, a symmetrization of K_f may be needed if infinite elements are to be used with added mass. This may be important, as the Dirichlet boundary condition approach may require a large acoustic mesh to obtain converged wet modes, whereas infinite elements typically allow for a much smaller (ellipsoidal) mesh.

Modal Solution of Acoustic Domain. The above procedure requires a solution of the acoustic domain at each step of the system eigen problem. This may be simplified by use of a modal expansion of the acoustic domain. We begin with the coupled system of equations, simplified by the limits of infinite acoustic velocity. The eigen equation may be summarized.

$$\left(-\omega^2 \begin{bmatrix} M_s & 0 \\ 0 & 0 \end{bmatrix} + i\omega \begin{bmatrix} 0 & L \\ L^T & 0 \end{bmatrix} + \begin{bmatrix} K_s & 0 \\ 0 & \frac{-1}{\rho_f} K_f \end{bmatrix}\right) \begin{bmatrix} u \\ \phi \end{bmatrix} = 0 \tag{3.7.9}$$

We consider a modal solution of the acoustic domain which diagonalizes the acoustic stiffness matrix. Specifically, we define $\phi = \psi q$ such that $\psi^T K_f \psi = \Lambda_f$, a diagonal matrix. Substituting into the lower equation of (3.7.9), we have,

$$i\omega L^T u = \frac{K_f}{\rho_f} \psi q \tag{3.7.10}$$

We pre-multiply by ψ^T, and solve for q.

$$q = i\omega \rho_f \Lambda_f^{-1} \psi^T L^T u \tag{3.7.11}$$

Substitution of q in the top equation of (3.7.9) results in a simplified expression for the mass loaded structural eigen problem.

$$\left(-\omega^2 [M_s + \tilde{M}_a] + K_s\right) u = 0 \tag{3.7.12}$$

where,

$$\tilde{M}_a = \rho_f L \psi \Lambda_f^{-1} \psi^T L^T \tag{3.7.13}$$

The eigenvalue problem above is real. The mass matrix contribution is real and symmetric. However, as in the physical solution above, the mass matrix is full on the wet surface boundary, and is not typically assembled. The modal solution does not require a linear solve at each iteration of the eigen solver, but by not assembling the mass matrix we cannot utilize the shift-invert strategies available in ARPACK.

Decomposition Issues

The linear solver depends on effective decompositions for accurate, robust, high performance solutions. In these methods, care must be taken for effective load balance. Rebalancing may be useful. It may be possible to require the linear solver to rebalance. Alternatively, we may want a decomposition that is independent in the fluid and structural domains.

Modal Truncation

The methods in this section are useful only if a reasonable modal truncation can be developed for the acoustic domain. The only requirement on the basis is that the eigenvectors diagonalize K_f. Thus, we could solve the standard eigenvalue problem, $(K_f - \lambda I)\psi = 0$, the generalized eigen problem with the fluid mass matrix, $(K_f - \lambda M_f)\psi = 0$, or use any other capacitance matrix. It is not clear which of these solutions would provide the best model for modal truncation. We also do not have any experience on the number of modes needed for effective truncation.

3.7.2. Case II - mismatched meshes at wet interface

When the meshes are mismatched at the wet interface, extra acoustic degrees of freedom are created on the structural side of the wet interface, and these degrees of freedom have zero stiffness. Also, the coupling matrix L is only active on the virtual acoustic degrees of freedom on the structural side of the wet interface. However, because of the manner in which linear constraint equations are handled in GDSW, the issue of virtual vs physical acoustic dofs does not impact the necessary algorithm development for the added mass mat-vec product.

Element Matrix Approximations. In the limits of infinite acoustic velocity, the contributions to the mass and damping matrices for the fluid go to zero. We consider here the stiffness matrix for an element in volumetric domain and for an infinite element. The infinite element formulation is described in equation (6.1.17) of the infinite element section (6.1.2). As shown in this section, the infinite element is not a function of either ω or c_o, and thus is unchanged in the infinite velocity approximation. Likewise, the volumetric stiffness is defined in equation (3.2.3) of Section 3. It is also independent of frequency or acoustic velocity. Standard element formulations apply for both stiffness matrix contributions in the limits of infinite acoustic velocity.

This page intentionally left blank.

4. MATERIAL

4.1. Anisotropic Materials

A theoretical development for anisotropic elasticity is presented emphasizing the numbering convention.

Linear Anisotropic Elasticity. Linear elasticity asserts that the stress is a linear function of the strain:

$$\sigma_{ij} = C^4_{ijkl}\epsilon_{kl}$$

Where C^4_{ijkl} are the Cartesian components of the fourth order constitutive tensor and the Einstein convention of summation on repeated indices is used.

4.1.1. Stress Vectors

By definition, the strain is symmetric. Further, we make the usual constitutive assumption that the stress is symmetric. This permits the representation of the 3x3 stress matrix and the 3x3 strain matrix each by a column vector having six rows.

$$s = \left\{ \begin{matrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{matrix} \right\}$$

and,

$$e = \left\{ \begin{matrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{23} \\ 2\epsilon_{13} \\ 2\epsilon_{12} \end{matrix} \right\}.$$

This is the Voigt notation. Note that this mapping from σ to s and from ϵ to e is not universal. This is the numbering used in Malvern and is popular in the materials science world, but it differs from the numbering used in NASTRAN and from the numbering in Abaqus. Although s and e are called the "stress vector" and the "strain vector", they do not map from one coordinate system to another as true vectors do. How that mapping is done is discussed in a later section.

We use the above to map the fourth-order tensor C^4_{ijkl} into a 6x6 matrix of material parameters. This is done with the aid of the matrices that formally map σ to s and from ϵ to e.

$$e_n = E_{nij}\epsilon_{ij} \tag{4.1.1}$$

and

$$\epsilon_{ij} = e_n F_{nij} \tag{4.1.2}$$

where

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad E_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad E_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$E_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad E_5 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad E_6 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.1.3}$$

and

$$F_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad F_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad F_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$F_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1/2 \\ 0 & 1/2 & 0 \end{bmatrix} \quad F_5 = \begin{bmatrix} 0 & 0 & 1/2 \\ 0 & 0 & 0 \\ 1/2 & 0 & 0 \end{bmatrix} \quad F_6 = \begin{bmatrix} 0 & 1/2 & 0 \\ 0 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} \tag{4.1.4}$$

We note that the stress mappings are also achieved with the above third order quantities:

$$s_n = F_{nij}\sigma_{ij} \tag{4.1.5}$$

and

$$\sigma_{ij} = s_n E_{nij} \tag{4.1.6}$$

From Equations 4.1.1 and 4.1.2 or Equations 4.1.5 and 4.1.6 we see that,

$$E_{mij}F_{nij} = \delta_{mn} \tag{4.1.7}$$

Substituting Equations 4.1.2 and 4.1.6 into Equation 4.1 and simplifying with Equation 4.1.7, we find

$$s_m = C_{mn}e_n \tag{4.1.8}$$

where

$$C_{mn} = F_{mij}C^4_{ijkl}F_{nkl} \tag{4.1.9}$$

This shows how to find the 6x6 matrix C_{ij} in terms of the fourth order tensor components C^4_{ijkl}. The material description may also be provided in terms of the components of C_{ij}.

4.1.2. Strain Energy and Orientation

Consider the situation where the matrix of material parameters is provided in a Cartesian coordinate system different from the global coordinate system in which strains are calculated. Because stress and strain are tensors, they transfer from one coordinate system to another by:

$$\sigma_{ij} = R_{ai}\hat{\sigma}_{ab}R_{bj} \tag{4.1.10}$$

and

$$\epsilon_{ij} = R_{ai}\hat{\epsilon}_{ab}R_{bj} \tag{4.1.11}$$

where σ_{ij} and ϵ_{ij} are the stress and strain components calculated in some other (global) Cartesian system and R_{ai} are the components of the rotation matrix that rotates the basis vectors in that global system to that with respect to which the material properties are defined. A basis vector $\hat{b}_a$ in the local, material frame is expressed in terms of the basis vectors of the global system by:

$$\hat{b}_a = R_{ai}b_i \tag{4.1.12}$$

where b_1, b_2, and b_3 are the basis vectors of the global frame.

From Equations 4.1.5, 4.1.6, and 4.1.9, we find following

$$s_m = (F_{mij}E_{nab}R_{ai}R_{bj})\hat{s}_n. \tag{4.1.13}$$

From Equations 4.1.1, 4.1.2, and 4.1.11, we find the more useful relationship

$$e_m = (E_{mij}F_{nab}R_{ai}R_{bj})\hat{e}_n. \tag{4.1.14}$$

The above two transformations are simplified:

$$s = T^T\hat{s} \tag{4.1.15}$$

and

$$e = T\hat{e} \tag{4.1.16}$$

where the 6x6 transformation matrix, T, is defined

$$T_{nk} = E_{nij}F_{kab}R_{ai}R_{bj} = tr\left(E_n^T R F_k R^T\right) \tag{4.1.17}$$

Noting that

$$s = \hat{C}\hat{e}, \tag{4.1.18}$$

and substituting Equations 4.1.15 and 4.1.16 into Equation 4.1.18, we further find

$$s = T^T\hat{C}Te. \tag{4.1.19}$$

Comparing the above with Equation 4.1.8, we finally find that

$$C = T^T\hat{C}T \tag{4.1.20}$$

which was the main point of this exercise.

Note also that the components of arrays E_n and F_n are mostly zero, with the rest either 1 or 1/2. As in [16] Equation 3.34, the simplified (with Maple) product matrix is

$$T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T22, \end{bmatrix} \tag{4.1.21}$$

$$T_{11} = \begin{bmatrix} R_{11}^2 & R_{12}^2 & R_{13}^2 \\ R_{21}^2 & R_{22}^2 & R_{23}^2 \\ R_{31}^2 & R_{32}^2 & R_{33}^2 \end{bmatrix}, \quad T_{12} = \begin{bmatrix} R_{13}R_{12} & R_{13}R_{11} & R_{13}R_{11} \\ R_{23}R_{22} & R_{23}R_{21} & R_{23}R_{21} \\ R_{33}R_{32} & R_{33}R_{31} & R_{33}R_{31} \end{bmatrix},$$

$$\begin{bmatrix} T_{21} = 2R_{21}R_{31} & R_{22}R_{32} & R_{23}R_{33} \\ 2R_{11}R_{31} & R_{12}R_{32} & R_{13}R_{33} \\ 2R_{11}R_{21} & R_{12}R_{22} & R_{13}R_{23} \end{bmatrix}, \begin{bmatrix} T_{22} = R_{23}R_{32} + R_{22}R_{33} & R_{23}R_{31} + R_{21}R_{33} & R_{22}R_{31} + R_{21}R32 \\ R_{13}R_{32} + R_{12}R_{33} & R_{13}R_{31} + R_{11}R_{33} & R_{12}R_{31} + R_{11}R32 \\ R_{13}R_{22} + R_{12}R_{23} & R_{13}R_{21} + R_{11}R_{23} & R_{12}R_{21} + R_{11}R22 \end{bmatrix}. \tag{4.1.22}$$

The Maple code to perform the above calculations follows.

```
with(linalg);
E[1] := matrix(3,3,[ [1,0,0],[0,0,0],[0,0,0]]);
E[2] := matrix(3,3,[ [0,0,0],[0,1,0],[0,0,0]]);
E[3] := matrix(3,3,[ [0,0,0],[0,0,0],[0,0,1]]);
E[4] := matrix(3,3,[ [0,0,0],[0,0,1],[0,1,0]]);
E[5] := matrix(3,3,[ [0,0,1],[0,0,0],[1,0,0]]);
E[6] := matrix(3,3,[ [0,1,0],[1,0,0],[0,0,0]]);
F[1] := E[1];
F[2] := E[2];
F[3] := E[3];
F[4] := (1/2)*E[4];
F[5] := (1/2)*E[5];
F[6] := (1/2)*E[6];
R := matrix(3,3);

for k from 1 to 6 do
FRR[k] := matrix(3,3);
FRR[k] := evalm ( R &* F[k] &*transpose(R));
od;

T := matrix(6,6);
for k from 1 to 6 do
for n from 1 to 6 do
T[n,k] := 0;
for i from 1 to 3 do
for j from 1 to 3 do
T[n,k] := T[n,k] +evalm(FRR[k][i,j])*E[n][i,j];
od; od;
od; od;

readlib(C);
C(T);

read("/home/djsegal/Maple/tools/maple2mif.mpl");
M := maple2mif();
fprintf("/home/djsegal/MPP/notes/temp.mif",'%s',M(eval(T))) ;
```

4.2. Viscoelastic Materials

Here we describe the integration of viscoelastic structures using the generalized alpha method. For the proper choice of the parameters of the generalized alpha method, the results below reduce to those corresponding to the Newmark-beta method.

4.2.1. Equations of motion

The equations of motion of elastodynamics in three dimensions are given by

$$\begin{aligned} u_{tt} - \nabla \cdot \sigma &= f(x,t) \quad \Omega && (4.2.1) \\ u(x,t) &= 0 \quad x \in \Gamma_D && (4.2.2) \\ \sigma(x,t) &= g(x,t) \quad x \in \Gamma_N && (4.2.3) \\ & && (4.2.4) \end{aligned}$$

where $u = (u_x, u_y, u_z)$ is the vector of displacements, σ is the stress tensor, and $f(x,t)$ is the body force. The boundary of Ω is divided into Dirichlet Γ_D and Neumann Γ_N subregions.

The Dirichlet conditions lead to the space of admissible functions

$$V = \left[v \in H^1(\Omega), v(x) = 0, x \in \Gamma_D \right] \tag{4.2.5}$$

The equation of motion, along with boundary conditions, is cast into the weak form in the standard way

$$\int_\Omega u_{tt} \cdot v + \int_\Omega \sigma \cdot \nabla_s v dx = \int_\Omega f(x,t) \cdot v dx + \int_{\Gamma_N} g(x,t) \cdot v ds \quad \forall v \in V \tag{4.2.6}$$

where an integration by parts has been carried out on the middle term, and $\nabla_s = \frac{1}{2}(\nabla + \nabla^T)$ denotes the symmetric part of the gradient operator.

4.2.2. Constitutive equations

The representation of the time-dependent moduli for a viscoelastic material is commonly written in the form of a Prony series

$$\begin{aligned} G(t) &= G_{\text{inf}} + (G_0 - G_{\text{inf}})\zeta_G(t) && (4.2.7) \\ \zeta_G(t) &= \sum_i c_i e^{-\frac{t}{s_i}} && (4.2.8) \end{aligned}$$

where G_0 is the glassy modulus, G_{inf} is the rubbery modulus, and c_i, s_i are coefficients used to fit the Prony series representation to the experimentally measured relaxation curve. A similar expression holds for $K(t)$, with different values for the constants, and possibly a different number of terms in the series. Assuming an isotropic viscoelastic constitutive law, we only need to consider two rate-dependent material properties. In this presentation, we will work in terms of the bulk K and shear G moduli, since experimental data is typically given in terms of these two parameters.

The constitutive model for an elastic material can be written in terms of the shear and bulk moduli

$$\sigma = D\epsilon = (KD_K + GD_G)\epsilon \tag{4.2.9}$$

where K, G are the scalar bulk and shear moduli, and as is shown in equation 9.4.7 in,[37]

$$D_K = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D_G = \begin{bmatrix} 4/3 & -2/3 & -2/3 & 0 & 0 & 0 \\ -2/3 & 4/3 & -2/3 & 0 & 0 & 0 \\ -2/3 & -2/3 & 4/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This constitutive law can be generalized to a linear viscoelastic material as follows

$$\sigma(x,t) = (G_0 - G_{\inf})D_G \int_0^t \zeta_G(x, t-\tau)\frac{\partial \epsilon(x,\tau)}{\partial \tau} d\tau + G_{\inf} D_G \epsilon(x,t) + \\ (K_0 - K_{\inf})D_K \int_0^t \zeta_K(x, t-\tau)\frac{\partial \epsilon(x,\tau)}{\partial \tau} d\tau + K_{\inf} D_K \epsilon(x,t) \tag{4.2.10}$$

The above expression is then used to represent the stress in the weak form of the equations of motion, 4.2.6.

Given a finite dimensional subspace $V_h \subset V$, we represent the approximate solution in the standard way

$$u_h(x,t) = \sum_{i=1}^{n} \phi_i(x)\eta_i(t) \tag{4.2.11}$$

where $V_h = span(\phi_i)$, and $\eta(t)$ represents the unknown time dependence. We also denote $\Phi(x) = [\phi_i(x)]$ as the matrix having ϕ_i as the i^{th} column. Inserting this into the equations of motion, and rearranging, we obtain

$$M\ddot{\eta}(t) + (G_0 - G_{\inf})K_1 \int_0^t \zeta_G(t-\tau)\dot{\eta}(\tau)d\tau + \\ (K_0 - K_{\inf})K_1 \int_0^t \zeta_K(t-\tau)\dot{\eta}(\tau)d\tau + K_2\eta(t) = f(t) \tag{4.2.12}$$

where

$$M = \int_\Omega \rho(x)\Phi^T(x)\Phi(x)dx \tag{4.2.13}$$

is the mass matrix,

$$K_1 = (G_0 - G_{\inf})\int_\Omega B^T D_G B dx + (K_0 - K_{\inf})\int_\Omega B^T D_K B dx \tag{4.2.14}$$

$$K_2 = G_{\inf}\int_\Omega B^T D_G B dx + K_{\inf}\int_\Omega B^T D_K B dx \tag{4.2.15}$$

are the stiffness matrices, and

$$f(t) = \int_\Omega f(x,t)\cdot v(x)dx + \int_{\Gamma_N} g(x,t)\cdot v(x)ds \tag{4.2.16}$$

is the right-hand side. The corresponding element matrices are defined by breaking the integrals into element wise contributions.

Equation 4.2.12 represents a system of Volterra integro-differential equations. Without the inertial term, 4.2.12 represents a system of Volterra integral equations of the first kind. The standard form for implicit time integration schemes is

$$M\ddot{\eta}(t) + C\dot{\eta}(t) + K\eta(t) = \hat{f}(t). \tag{4.2.17}$$

Here C is a *constant* damping matrix. Is the system of equations 4.2.12 reducible to standard form? $f\hat{(t)}$ is a modified right-hand side that will include a portion of the viscoelastic convolution term. We demand that C be independent of time, since this will eliminate the need for refactoring the left-hand side at each time step. The damping (integral) term in equation 4.2.12 is time-dependent. However, we will show that it is possible to split this integral term into a time-dependent and a time-independent part. The time-independent parts remain on the left-hand side and become the damping matrix, whereas the time-dependent parts can be carried to the right-hand side, since they are known quantities. Once the equations 4.2.12 are reduced to the system 4.2.17, the standard time integrators for structural dynamics can be employed.

For simplicity, we consider the case of only a single Prony series term. The results for more terms can be obtained by adding together the results for a single term. The integral in equation 4.2.12 can be split into two parts (considering only a single Prony series term)

$$\int_0^t e^{\frac{t-\tau}{s}}\dot{\eta}(\tau)d\tau = \int_0^{t_i} e^{\frac{t-\tau}{s}}\dot{\eta}(\tau)d\tau + \int_{t_i}^t e^{\frac{t-\tau}{a}}\dot{\eta}(\tau)d\tau \tag{4.2.18}$$

$$= e^{\frac{\Delta t}{s}}\int_0^{t_i} e^{\frac{t_i-\tau}{s}}\dot{\eta}(\tau)d\tau + \int_{t_i}^t e^{\frac{t-\tau}{s}}\dot{\eta}(\tau)d\tau \tag{4.2.19}$$

where the first term is a loading history term that is *known* at time t_i. Consequently, it can be treated as an additional load and brought to the right-hand side. The remaining term can be split into two terms, one containing coefficients of $\dot{\eta}$, and the other containing coefficients of $\dot{\eta}_i$. The former is unknown and thus becomes $C\dot{\eta}$, whereas the latter is known and thus also contributes to the right-hand side.

To evaluate the term

$$\int_{t_i}^t e^{\frac{t-\tau}{s}}\dot{\eta}(\tau)d\tau \tag{4.2.20}$$

we first need a representation for the velocity $\eta\dot{(\tau)}$ in the interval $\tau \in [t_i, t]$. We present two choices, both of which are second order accurate.

4.2.3. *Linear Representation of Velocity*

The first is consistent with the Newmark-beta method, which presumes a constant acceleration within the time step. With this assumption, the velocity must vary linearly within the time step. Thus,

$$\eta\dot{(t)} = \eta\dot{(t_i)} + \frac{\ddot{\eta} + \eta\ddot{(t_i)}}{2}(t - t_i) \tag{4.2.21}$$

where $\ddot{\eta}$ is the (unknown) acceleration at current time t, and $\eta\ddot{(t_i)}$ is the previous acceleration. Although equation 4.2.21 is the correct representation for velocity, it is inconvenient in that it would lead to (after inserting into equation 4.2.20) a contribution to the mass matrix. This is undesirable, since it would interfere with the use of a lumped mass matrix. Thus, we re-write the velocity distribution in an equivalent form

$$\eta(t) = \eta(\dot{t_i}) + \frac{\dot{\eta} - \eta(\dot{t_i})}{\Delta t}(t - t_i) \tag{4.2.22}$$

We note that equations 4.2.21 and 4.2.22 are equivalent representations of the velocity. By inserting equation 4.2.22 into equation 4.2.20 we obtain

$$\int_{t_i}^{t} e^{\frac{t-\tau}{s}} \dot{\eta}(\tau) d\tau = \left[s + \frac{s^2}{\Delta t} \left(e^{\frac{\Delta t}{s}} - 1 \right) \right] \dot{\eta} + \left[-s e^{\frac{-\Delta t}{s}} + \frac{s^2}{\Delta t} \left(1 - e^{\frac{-\Delta t}{s}} \right) \right] \dot{\eta}_i \tag{4.2.23}$$

The first term involves a coefficient times the unknown $\dot{\eta}$, which is the unknown velocity at the current time, and thus it must remain on the left-hand side as a damping term contribution. The damping matrix implied by this term is

$$C = c_K (s_K + \frac{s_K^2}{\Delta t} (e^{\frac{-\Delta t}{s_K}} - 1)) \mathbf{B}^{\mathbf{T}} \mathbf{D_K} \mathbf{B} + c_G (s_G + \frac{s_G^2}{\Delta t} (e^{\frac{-\Delta t}{s_G}} - 1)) \mathbf{B}^{\mathbf{T}} \mathbf{D_G} \mathbf{B} \tag{4.2.24}$$

The second term is known, and thus it can be added to the load vector.

4.2.4. Midpoint Representation of Velocity

A second implicit scheme can be derived by using the midpoint rule on the velocity in the viscoelastic term. The only difference from the linear approach described above is in equation 4.2.23.

$$\dot{\eta}(t) = \frac{\dot{\eta} + \eta(\dot{t_i})}{2} \tag{4.2.25}$$

This leads to

$$\int_{t_i}^{t} e^{\frac{t-\tau}{s}} \dot{\eta}(\tau) d\tau = \frac{s}{2} \left(1 - e^{\frac{\Delta t}{s}} \right) \dot{\eta} + \frac{s}{2} \left(1 - e^{\frac{\Delta t}{s}} \right) \dot{\eta}_i \tag{4.2.26}$$

In the same way as for the linear velocity approach, we use the term involving $\dot{\eta}$ to construct a damping matrix, and the remaining known terms are carried to the right-hand side.

The midpoint scheme is inconsistent in that a different discretization scheme is used for the viscoelastic term than was used for the overall time integration. The linear representation of velocity is a consistent scheme. However, both approaches are second order accurate.

5. ELEMENTS

Structural dynamics is a rich and extensive field. Finite element tools such as **Sierra/SD** have been used for decades to describe and analyze a variety of structures. The same tools are applied to large civil structures (such as bridges and towers), to machines, and to micron sized structures. This has necessarily led to a wealth of different element libraries. Details of these element libraries are presented in this section. For information on the solution procedures that tie these elements together, please refer to Section 2.

5.1. Corrections to Element Matrices

Several elements generate element matrices that may need corrections. For example, the stiffness matrix generated from Craig-Bampton reductions may not be positive definite, and may not have the proper null space. Infinite acoustic elements have a similar problem with the mass matrix. These errors are typically small, but may lead to unstable systems. Correcting the errors is an important step.

The errors are removed using an eigen decomposition. We compute the eigenvalues and eigenvectors of the element matrix of concern.

$$(A - \lambda I)\phi = 0$$

where A is the matrix of concern, λ are the eigenvalues and ϕ are the eigenvectors. Computation of the eigen problem on a small element matrix is not expensive. We normalize the eigenvectors such that $\phi^T\phi = I$. It follows that $\phi^T = \phi^{-1}$. We correct the element matrix by computing,

$$\tilde{A}_{jk} = A_{jk} - \sum_{i}^{\lambda_i<0} \phi_{ij}\lambda_i\phi_{ik} \tag{5.1.1}$$

The element matrix $\tilde{A}$ replaces matrix A in subsequent calculations. The correction of the null space vectors (and the element matrix) is optionally performed for Craig-Bampton models. See Figure 2-6.

5.2. Mass lumping

A consistent mass matrix is used by default. A lumped mass matrix is used to apply gravity loads, and is available for most solution cases. Several mass lumping techniques are outlined in the literature.[75] Summing mass across rows is an established method. It works for most volumetric elements. It is used in SD.

For elements both with translational and rotational DOFs, the row sums are segregated. With a 2 node beam with 6 dofs per node, the sum for rows $\{1,2,3\}$ includes columns $\{1,2,3\}$ and $\{7,8,9\}$. Rotational lumping uses the same row sum method for rotational inertias. The sum for rows $\{4,5,6\}$ includes columns $\{4,5,6\}$ and $\{10,11,12\}$. Rotational lumping uses the

5.3. Selective integration

In theory, selective integration applies to any 3D isoparametric elements. The implementation applies selective integration to elements with linear shape functions (such as hex8 or wedge6). The first step is to explain how to evaluate certain operators on the shape functions. Later the operators will be integrated into K.

5.3.1. Derivation

The strategy for avoiding over stiffness with respect to bending begins with splitting the strain into deviatoric and dilatational parts. An isotropic, linearly elastic material has strain energy density

$$p = \frac{1}{2}(2G\epsilon + \lambda tr(\epsilon)I) \bullet \epsilon \tag{5.3.1}$$

with some re-arrangement, this can be shown to be:

$$p = G\hat{\epsilon} \bullet \hat{\epsilon} + \frac{1}{2}\beta(tr(\epsilon))^2 \tag{5.3.2}$$

where $\hat{\epsilon} = \epsilon - \frac{1}{3}tr(\epsilon)I$.

The contribution to strain energy density from the deviatoric strain is separated from the contribution from the dilatational strain. The contributions are integrated separately. First, the strains are expressed in terms of nodal degrees of freedom.

The deformation field depends linearly on the nodal DOFs. The displacement gradient does too. It should be possible to expand each quantity as follows.

Let P_j be the node associated with the $j$the degree of freedom and let s_j be the direction associated with that degree of freedom. The displacement field is:

$$\vec{u}(x) = \tilde{N}^{P_j}(x)u_{s_j}^{P_j}\vec{e}_{s_j} \tag{5.3.3}$$

where summation takes place over the degree of freedom j.

Similarly, the displacement gradient is:

$$\vec{\nabla}\vec{u}(x) = (\frac{\partial}{\partial x_k})\tilde{N}^{P_j}(x)u_{s_j}^{P_j}\vec{e}_{s_j}\vec{e}_k \tag{5.3.4}$$

We define the shape deformation tensor W^j corresponding to the j the nodal degree of freedom:

$$W^j(x) = (\frac{\partial}{\partial u_{s_j}^{P_j}})\vec{\nabla}\vec{u}(x) \tag{5.3.5}$$

which, with Equation 5.3.4 yields:

$$W^j(x) = (\frac{\partial}{\partial x_k})\tilde{N}^{P_j}(x)\vec{e}_{s_j}\vec{e}_k \tag{5.3.6}$$

The symmetric part of this tensor and the strain tensor are,

$$S^j(x) = \frac{1}{2}(W^j(x) + W^j(x)^T), \quad \epsilon(x) = S^j(x) u_{s_j}^{P_j}.$$

From the above, we construct the dilatational and deviatoric portions of the strain in terms of the nodal displacement components:

$$tr(\epsilon(x)) = b^j(x) u_{s_j}^{P_j} \tag{5.3.7}$$

where

$$b^j(x) = tr(S^j(x)) \tag{5.3.8}$$

Similarly,

$$\hat{\epsilon}(x) = \hat{B}^j(x) u_{s_j}^{P_j} \tag{5.3.9}$$

where

$$\hat{B}^j(x) = S^j(x) - \frac{1}{3} b^j(x) I \tag{5.3.10}$$

To evaluate K use the constitutive equation 5.3.2 and

$$K_{m,n} = \frac{\partial^2}{\partial u_{s_m}^{P_m} \partial u_{s_n}^{P_n}} \int_{volume} p(x) dV(x) \tag{5.3.11}$$

Combine this with the expressions for strain in terms of the nodal DOFs,

$$\begin{aligned} K_{m,n} = G &\int_{volume} (\hat{B}^m(x))^T \bullet \hat{B}^n(x) dV(x) \\ &+\beta \int_{volume} b^m(x) b^n(x) dV(x) \end{aligned} \tag{5.3.12}$$

5.3.1.1. Implementation

From the above it is seen that once the shape deformation tensor W^j is found, the rest of the calculation follows naturally. Next the tensor components are derived. The components of W^j are

$$W_{mn}^j = \vec{e}_m \cdot W^j \cdot \vec{e}_n \tag{5.3.13}$$

$$= \delta_{m,s_j} (\frac{\partial}{\partial x_n}) \tilde{N}^{P_j}(x) \tag{5.3.14}$$

The partial derivative $(\frac{\partial}{\partial x_n})\tilde{N}^{P_j}(x)$ is calculated from

$$(\frac{\partial}{\partial x_n}) \tilde{N}^{P_j}(x(\xi)) = (\frac{\partial}{\partial \xi_\alpha}) N^{P_j}(\xi) J_{\alpha,n}^{-1} \tag{5.3.15}$$

where

$$J_{m,\gamma} = \frac{\partial}{\partial \xi_\gamma} x_m(\xi) \tag{5.3.16}$$

and

$$N(\xi) = \tilde{N}(x(\xi)) \tag{5.3.17}$$

Selective element integration, discussed in Section 5.4, is applied to all isoparametric solid elements.

5.4. Integration of Isoparametric Solids

A selective integration method for isoparametric solids is described that satisfies the standard conditions, including the patch test, and at the same time accommodates anisotropic materials.

The matrix of elastic constants connects the stress, s, and strain, v, vectors,

$$s = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} = Cv, \quad v = \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{23} \\ 2\epsilon_{13} \\ 2\epsilon_{12} \end{bmatrix}.$$

Virtual work will be used to derive the stiffness matrix.

$$\delta W = \int_V s^T \delta v dV = \int_V v^T C \delta v dV \tag{5.4.1}$$

If we select the above volume to be that of an element and use the strain-displacement matrices associated with each nodal degree of freedom,

$$v(x) = \sum_j B_j(x) u_j \tag{5.4.2}$$

where u_j is the j^{th} nodal degree of freedom, the virtual work becomes

$$\delta W = u_j \delta u_k \int_V B_j(x)^T C B_k(x) dV \tag{5.4.3}$$

Since the element stiffness matrix is defined by

$$\delta W = u_j \delta K_{ij} \tag{5.4.4}$$

we conclude that

$$K_{ij} = \int_V B_j(x)^T C B_k(x) dV \tag{5.4.5}$$

Next the strain-displacement vectors are decomposed into deviatoric and dilatational components.

$$B_j(x) = B_j^D(x) + B_j^V(x) \tag{5.4.6}$$

where,

$$B_j^V(x) = d_j(x) \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.4.7}$$

and $3d_j(x)$ is the sum of the first three rows of $B_j(x)$. $B_j^D(x)$ is defined by equation 5.4.6. Substitution of equation 5.4.6 into equation 5.4.5 yields:

$$\begin{aligned} K_{ij} \quad = \quad & \int_V B_j^D(x)^T C B_k^D(x) dV + \int_V B_j^V(x)^T C B_k^V(x) dV + \cdots \\ & + \int_V B_j^V(x)^T C B_k^D(x) dV + \int_V B_j^D(x)^T C B_k^V(x) dV \end{aligned} \tag{5.4.8}$$

In the case of isotropic materials, the deviatoric and dilatational portions of the strain are orthogonal with respect to the matrix of material constants. The last two integrals in equation (5.4.8) vanish. Finally parasitic shear is mitigated by using special cubature rules for each contribution to the stiffness matrix in equation (5.4.8).

Uniform Strain-Displacement Matrices. The purpose of this section is to explain the treatment for anisotropic materials. The first new tool is the element averaged strain displacement matrices.

$$\bar{B}_k = \frac{1}{V} \int_V B_k(x) dV \tag{5.4.9}$$

For hexahedrons, these are the strain-displacement matrices,[63,64] and lead to "uniform strain" elements. Elements formed by the above strain/displacement matrices are "soft", having properties similar to elements formed by single point integration. Hex elements of this sort display spurious zero-energy modes. In what follows, we consider linear combinations of this strain-displacement matrix formulation with the consistent formulation of equation (5.4.2).

The uniform strain matrices are also separable into dilatational and deviatoric parts.

$$\bar{B}_k = \bar{B}_k^V + \bar{B}_k^D \tag{5.4.10}$$

Mixed Integration. This selective integration method builds on one presented by Hughes.[78] We can achieve the effect of softening elements by forming the strain displacement matrices from combinations of the consistent strain-displacement and the uniform strain displacement matrices.

$$\hat{B}_k(x) = \alpha \bar{B}_k^V + (1-\alpha) B_k^V(x) + \beta \bar{B}_k^D + (1-\beta) B_k^D(x) \tag{5.4.11}$$

(14) Note that for all values of α and β, the above correctly captures uniform strains. It is in how the non-uniform strains contribute to the stiffness matrix that the particular values of α and β make a difference. By setting values of α and β according to the following table, we recover the standard integration forms:

α	β	**Integration**
1	1	Flanagan and Belytschko
0	0	Full Integration
1	0	Selective Integration

We note that setting $\alpha = 1$ and using an intermediate value of β, we can achieve performance comparable to that of the Flanagan and Belytschko element but without admitting hour-glass modes.

5.4.1. *Mean Quadrature with Selective Deviatoric Control*

In this section we discuss the implementation of the mean quadrature element in **Sierra/SD**. This work is a result of a collaboration with Sam Key.[84]

We first examine the element stiffness matrix resulting from a fully integrated element

$$K = \int_V B^T C B dV \tag{5.4.12}$$

where K is the stiffness matrix, V is the volume of the element, B is the standard strain-displacement matrix, and C is the matrix of material constants. When implemented in the standard way, this element behaves poorly for nearly-incompressible materials, and is too stiff even on materials with moderate Poisson ratios.

A standard approach for softening the element formulation in the presence of nearly incompressible materials is to replace the matrix B with its mean quadrature counterpart, $\tilde{B}$,

$$\tilde{B} = \int_V B dV \tag{5.4.13}$$

This alleviates problems associated with nearly incompressible materials, but the resulting stiffness matrix exhibits hourglass modes. These modes can be removed either through hourglass control methods, or by adding in some of the missing deviatoric components. We use the latter method. B and $\tilde{B}$ split into volumetric and deviatoric components, i.e.

$$\begin{aligned} \tilde{B} &= \tilde{B}_V + \tilde{B}_D \\ B &= B_V + B_D \end{aligned} \tag{5.4.14}$$

With these decompositions, we define

$$\hat{B} = \tilde{B}_V + \tilde{B}_D + sd(B_D - \tilde{B}_D) \tag{5.4.15}$$

where sd is a parameter between 0 and 1. When $sd = 0$, the element corresponds to a mean quadrature element. When $sd = 1$, the element corresponds to mean quadrature on the volumetric part, but with full integration on the deviatoric component.

With this new definition of $\hat{B}$, we can define the stiffness matrix for this element as

$$K = \int_V \hat{B}^T C \hat{B} dV \tag{5.4.16}$$

5.4.2. *Bubble Functions*

Low order finite elements tend to behave poorly when subjected to bending loads. The bubble hex elements have been shown to give much better bending performance, without increasing the number of degrees of freedom in the element.[126,79,95] In this section we give a brief review of the theory behind this element.

The representation of displacement at the element level in the standard hex8 element is

$$\mathbf{u} = \sum_{\mathbf{i=1}}^{\mathbf{8}} \mathbf{u_i N_i}(\xi) = \mathbf{u^T N} \tag{5.4.17}$$

where u is the element displacement, N_i is the i^{th} shape function, $\mathbf{N}$ is the vector of shape functions, and ξ is the vector of reference element coordinates. The bubble element augments the standard finite element basis functions with additional bubble functions. The representation of displacement at the element level for the bubble element takes the form

$$\mathbf{u} = \sum_{\mathbf{i=1}}^{\mathbf{8}} \mathbf{u_i N_i}(\xi) + \sum_{\mathbf{i=1}}^{\mathbf{3}} \mathbf{a_i P_i}(\xi) = \mathbf{u^T N} + \mathbf{a^T P} \tag{5.4.18}$$

where $P_i(\xi)$ are the bubble functions, P is the vector of bubble functions, a_i are the unknown coefficients for the bubble functions, and a is the vector of unknown coefficients for the bubble functions. The corresponding expression for element strain is given as

$$\epsilon = \mathbf{Bu} + \mathbf{Ga} \tag{5.4.19}$$

where B and G are the appropriate[126,79] derivatives of the shape functions. Note that B is a $6x24$ matrix, whereas G is a $6x9$ matrix.

The corresponding element stiffness and load terms can be assembled into a block $2times2$ system

$$\begin{bmatrix} K & E^T \\ E & H \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{5.4.20}$$

where $K = \int_e B^T C B dV$ is the $24x24$ element stiffness matrix corresponding to standard element shape functions, $H = \int_e G^T C G dV$ is the $9x9$ stiffness matrix corresponding to bubble shape functions, $E = \int_e G^T C B dV$ is the $9x24$ matrix corresponding to products of bubble and standard shape functions, and f is the element load vector. The bubble unknowns a are local to each element, and may be eliminated, yielding the modified element stiffness matrix

$$\hat{K} = K - E^T H^{-1} E \tag{5.4.21}$$

The order of K remains 24×24.

With one of two supported corrections, the bubble hex element passes the patch test, assuring convergence. First, G is evaluated at the element centroid[126] instead of the Gauss points. Second the average value of G is determined[79] and subtracted from G.

In **Sierra/SD**, we have taken the second approach. A new G matrix is defined, $\hat{G}$, that is constructed by subtracting the average value of G from G.

$$\hat{G} = G - \frac{1}{V_e} \int_e G dV \tag{5.4.22}$$

We replace G with $\hat{G}$ in the above equations. We note that, in the implementation of this element in **Sierra/SD**, it was found that after implementing the correction described above, the element passed the patch test. Without the correction, the element failed all the patch tests.

With the bubble element, stress is a function of the thickness. Stress is determined from the strain. The solution procedure determines is element displacement vector $\mathbf{u}$. Equation 5.4.19 for the strain depends on the bubble DOFs $\mathbf{a}$. Due to equation 5.4.20,

$$\mathbf{a} = \mathbf{H^{-1} E u} \tag{5.4.23}$$

5.4.2.1. Nonlinear analysis of Bubble functions

The bubble element can be used in nonlinear analysis. The procedure[79] is reviewed next. Although the assumed strain approach was used instead of the assumed displacement method, both lead to the same procedure.

We will give the necessary modifications for a nonlinear static analysis. The governing equation is

$$F^{int}(\mathbf{u}, \alpha) = \mathbf{F^{ext}} \tag{5.4.24}$$

It separates into two equations

$$F_1^{int} = \int_\Omega B^T \sigma d\Omega = F^{ext} \tag{5.4.25}$$

$$F_2^{int} = \int_\Omega G^T \sigma d\Omega = 0 \tag{5.4.26}$$

The stress is given by $\sigma = C\epsilon$, where ϵ is given by equation 5.4.19.

The quantities $\mathbf{u}$ and α denote the unknowns, and $\hat{\mathbf{u}}$ and $\hat{\alpha}$ represent the current iterates of displacement and bubble unknowns. The two term Taylor's series for internal force is

$$F_1^{int}(\mathbf{u}, \alpha) \approx \mathbf{F_1^{int}}(\hat{\mathbf{u}}, \hat{\alpha}) + \frac{\partial \mathbf{F_1^{int}}}{\partial \mathbf{u}} \mathbf{\Delta u} + \frac{\partial \mathbf{F_1^{int}}}{\partial \alpha} \mathbf{\Delta} \alpha \tag{5.4.27}$$

$$F_2^{int}(\mathbf{u}, \alpha) \approx \mathbf{F_2^{int}}(\hat{\mathbf{u}}, \hat{\alpha}) + \frac{\partial \mathbf{F_2^{int}}}{\partial \mathbf{u}} \mathbf{\Delta u} + \frac{\partial \mathbf{F_2^{int}}}{\partial \alpha} \mathbf{\Delta} \alpha \tag{5.4.28}$$

We define

$$K_T = \frac{\partial F_1^{int}}{\partial \mathbf{u}} \tag{5.4.29}$$

$$E_T = \frac{\partial F_1^{int}}{\partial \alpha} \tag{5.4.30}$$

$$H_T = \frac{\partial F_2^{int}}{\partial \alpha} \tag{5.4.31}$$

where the subscript T denotes tangent matrices that are computed at the current configuration. Using these definitions and substituting equations 5.4.28 into equations 5.4.26, we obtain

$$\begin{bmatrix} K_T & (E^T)_T \\ E_T & H_T \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \mathbf{a} \end{bmatrix} = \begin{bmatrix} Res_\mathbf{u} \\ Res_\alpha \end{bmatrix} \tag{5.4.32}$$

where

$$Res_\mathbf{u} = F^{ext} - F_1^{int}(\hat{\mathbf{u}}, \hat{\alpha}) \tag{5.4.33}$$

$$Res_\alpha = -F_2^{int}(\hat{\mathbf{u}}, \hat{\alpha}) \tag{5.4.34}$$

In equation 5.4.26 and others, σ and B depend on displacement u and bubble unknowns α. Using the chain rule, the tangent matrices are

$$K_T = \frac{\partial \int_\Omega B^T \sigma d\Omega}{\partial \mathbf{u}} = \int_\Omega \frac{\partial B^T}{\partial \mathbf{u}} \sigma d\Omega + \int_\Omega B^T \frac{\partial \sigma}{\partial \mathbf{u}} d\Omega \tag{5.4.35}$$

$$E_T = \frac{\partial \int_\Omega B^T \sigma d\Omega}{\partial \alpha} = \int_\Omega \frac{\partial B^T}{\partial \alpha} \sigma d\Omega + \int_\Omega B^T \frac{\partial \sigma}{\partial \alpha} d\Omega \tag{5.4.36}$$

$$H_T = \frac{\partial \int_\Omega G^T \sigma d\Omega}{\partial \alpha} = \int_\Omega \frac{\partial G^T}{\partial \alpha} \sigma d\Omega + \int_\Omega G^T \frac{\partial \sigma}{\partial \alpha} d\Omega \tag{5.4.37}$$

In each expression, on the right-hand side the first and second terms are geometric and material stiffnesses respectively.

The deformation gradient is used to evaluate $\frac{\partial B^T}{\partial \mathbf{u}}$ and $\frac{\partial B^T}{\partial \alpha}$. New notation is needed. $\mathbf{X}$ is the initial configuration, $\mathbf{x}$ is the current configuration, and $\mathbf{u} = \mathbf{x} - \mathbf{X}$ is the displacement. Note that

$$F = \frac{\partial x}{\partial X} = I + \frac{\partial u}{\partial X} = I + u^T \frac{DN}{DX} + \alpha^T \frac{DP}{DX} \tag{5.4.38}$$

$$\frac{\partial F}{\partial u} = \frac{DN}{DX} \tag{5.4.39}$$

$$\frac{\partial^2 F}{\partial u^2} = 0 \tag{5.4.40}$$

This implies that $\frac{\partial^2 F}{\partial u^2} = 0$. Therefore,

$$e = \frac{1}{2}(F^T F - I) \tag{5.4.41}$$

$$B = \frac{\partial \epsilon}{\partial u} = F\frac{\partial F}{\partial u} \tag{5.4.42}$$

$$\frac{\partial B}{\partial u} = F\frac{\partial^2 F}{\partial u^2} + \frac{\partial F}{\partial u}\frac{\partial F}{\partial u} = \frac{\partial F}{\partial u}\frac{\partial F}{\partial u} \tag{5.4.43}$$

Similarly, we can construct these equations for the bubble functions

$$e = \frac{1}{2}(F^T F - I) \tag{5.4.44}$$

$$G = \frac{\partial \epsilon}{\partial \alpha} = F\frac{\partial F}{\partial \alpha} \tag{5.4.45}$$

$$\frac{\partial G}{\partial \alpha} = F\frac{\partial^2 F}{\partial \alpha^2} + \frac{\partial F}{\partial \alpha}\frac{\partial F}{\partial \alpha} = \frac{\partial F}{\partial \alpha}\frac{\partial F}{\partial \alpha} \tag{5.4.46}$$

where similar identities have been used

$$F = \frac{\partial x}{\partial X} = I + \frac{\partial u}{\partial X} = I + u^T \frac{DN}{DX} + \alpha^T \frac{DP}{DX} \tag{5.4.47}$$

$$\frac{\partial F}{\partial \alpha} = \frac{DP}{DX} \tag{5.4.48}$$

$$\frac{\partial^2 F}{\partial \alpha^2} = 0 \tag{5.4.49}$$

For the cross terms, we have

$$e = \frac{1}{2}(F^T F - I) \tag{5.4.50}$$

$$B = \frac{\partial \epsilon}{\partial u} = F\frac{\partial F}{\partial u} \tag{5.4.51}$$

$$\frac{\partial B}{\partial \alpha} = F\frac{\partial^2 F}{\partial u \partial \alpha} + \frac{\partial F}{\partial u}\frac{\partial F}{\partial \alpha} = \frac{\partial F}{\partial u}\frac{\partial F}{\partial \alpha} \tag{5.4.52}$$

where, again we justify that the second term vanishes as follows

$$F = \frac{\partial x}{\partial X} = I + \frac{\partial u}{\partial X} = I + u^T \frac{DN}{DX} + \alpha^T \frac{DP}{DX} \tag{5.4.53}$$

$$\frac{\partial F}{\partial u} = \frac{DN}{DX} \tag{5.4.54}$$

$$\frac{\partial^2 F}{\partial u \partial \alpha} = 0 \tag{5.4.55}$$

In a similar manner as was done for the linear element, the bubble degrees of freedom can be condensed from equations 5.4.34. This results in the equation

$$(K_T - E_T^T H_T^{-1} E_T)\Delta\mathbf{u} = \mathbf{Res_u} - \mathbf{E_T^T H_T^{-1} Res_\alpha} \tag{5.4.56}$$

Thus, the full tangent operator for the bubble element is given by

$$K_T - E_T^T H_T^{-1} E_T \tag{5.4.57}$$

the internal force is given by

$$F_1^{int}(\hat{\mathbf{u}}, \hat{\alpha}) - E_T^T H_T^{-1} F_2^{int}(\hat{\mathbf{u}}, \hat{\alpha}) \tag{5.4.58}$$

and the residual is given by two terms

$$Res_{\mathbf{u}} - E_T^T H_T^{-1} Res_\alpha \tag{5.4.59}$$

These equations describe the nonlinear analysis of the bubble element.

5.5. Quadratic isoparametric solids

Quadratic elements (elements with bilinear or higher order shape functions) such as the hex20 and tet10 are naturally soft and do not need to be softened by positive values of G and β (see sections 5.3 and 5.4 for definitions of G and β). Therefore, the values $G = 0$ and $\beta = 0$ are recommended.

5.5.1. *Shape functions and integration points*

The shape functions and Gauss points for hex20 elements use a standard ordering. The nodal ordering (and shape functions) follows the ordering in the **Exodus** manual. Gauss points are input and output using the ordering developed by Thompson.[128] Internally, the Gauss points are located at element coordinates (and order) shown in Table 5-2.

Shape Function	A_0	A_1	A_2	A_3	A_4	A_5
$N_1 = (1-\xi)t/2$	1/2	-1/2	-1/2	-1/2	1/2	1/2
$N_2 = (1-\xi)r/2$		1/2			-1/2	
$N_3 = (1-\xi)s/2$			1/2			$-1/2$
$N_4 = (1+\xi)t/2$	1/2	-1/2	-1/2	1/2	-1/2	-1/2
$N_5 = (1+\xi)r/2$		1/2			1/2	
$N_6 = (1+\xi)s/2$			1/2			1/2

Table 5-1. – Shape functions and coefficients.

number	label suffix	X	Y	Z
1	111	0	0	0
2	112	0	0	A
3	110	0	0	-A
4	121	0	A	0
5	122	0	A	A
6	120	0	A	-A
7	101	0	-A	0
8	102	0	-A	A
9	100	0	-A	-A
10	211	A	0	0
11	212	A	0	A
12	210	A	0	-A
13	221	A	A	0
14	222	A	A	A
15	220	A	A	-A
16	201	A	-A	0
17	202	A	-A	A
18	200	A	-A	-A
19	011	-A	0	0
20	012	-A	0	A
21	010	-A	0	-A
22	021	-A	A	0
23	022	-A	A	A
24	020	-A	A	-A
25	001	-A	-A	0
26	002	-A	-A	A
27	000	-A	-A	-A

Table 5-2. – Hex20 Gauss Point Locations. The constant A=0.77459666924148. The unit element is 2x2x2, with a volume of 8 cubic units.

5.6. Wedge Shape Functions

The shape functions are given explicitly as in.[78] These are provided as bi-linear polynomials in r, s, t, and ξ, where r and s are independent coordinates of the triangular cross-subsections, $t = 1 - r - s$, and ξ is the coordinate in the third direction. For our purposes, it is necessary to expand the shape functions as polynomials in r, s, and ξ:

$$N_k = A_0^k + A_1^k r + A_2^k s + A_3^k \xi + A_4^k r\xi + A_5^k s\xi \tag{5.6.1}$$

No. Points	r	s	ξ
1	1/3	1/3	0
2	1/3	1/3	-1/$\sqrt{3}$
	1/3	1/3	1/$\sqrt{3}$
6	1/6	1/6	-1/$\sqrt{3}$
	2/3	1/6	-1/$\sqrt{3}$
	1/6	2/3	-1/$\sqrt{3}$
	1/6	1/6	1/$\sqrt{3}$
	2/3	1/6	1/$\sqrt{3}$
	1/6	2/3	1/$\sqrt{3}$

Table 5-3. – Wedge element integration rules.

5.7. Tet10

The degree 2 integration rule (see for example Appendix 3.1 of[78]) based on values at the four vertices is used for the stiffness matrix. The mass matrix depends on integrals of polynomials two degrees higher than the stiffness matrix. Higher order integration is required to determine a consistent (exact) mass matrix than is required for the stiffness matrix. The 16-point integration comes from.[80] (Using 4-point integration to try to estimate the mass matrix of a natural element resulted in a 30 by 30 mass matrix with several zero eigenvalues.) A 16-point integration with degree of exactness 6 from[80] is used for the mass matrices. Lower order cubature rules are often sufficient, and in these cases they are used for efficiency.

5.8. Hex20 shape functions and gradients

The shape functions a determined from the monomials

$$p_i(\varepsilon) = \varepsilon_1^{r_i} \varepsilon_2^{s_i} \varepsilon_3^{t_i}.$$

for the non-negative integers $\{r_i, s_i, t_i\}_{1 \le i \le 20}$ such that

$$r_i^2 + s_i^2 + t_i^2 \le 7.$$

The derivation of a cardinal basis starts with the `rst` matrix.

$$S_{20} = \{(I, J, K) : I^2 + J^2 + K^2 < 8\}.$$

The shape functions $\{N_i(r,s,t)\}_{1\le i\le 20}$ are linear combinations of the p_i satisfying $N_i(r_j,s_j,t_j) = \delta_{i,j}$,

$$\vec{N} = A\vec{p}. \tag{5.8.1}$$

The element has 20 nodes. A is a 20×20 matrix. Wouldn't A be 60×60 ?

We find the 400 term $A-$matrix values. Let $\vec{\varepsilon_i}$ denote the natural coordinate value at the ith node. We have $A\vec{p}(\vec{\varepsilon}_1) = \vec{e}_1 \equiv (1,0,0,\ldots,0)^T$, and, in general, $A\vec{p}(\vec{\varepsilon}_i) = \vec{e}_i$.

$$[\vec{\varepsilon}_1, \vec{\varepsilon}_2, \ldots, \vec{\varepsilon}_{20}] = [A][\vec{p}(\vec{\varepsilon}_1), \vec{p}(\vec{\varepsilon}_2), \ldots, \vec{p}(\vec{\varepsilon}_{20})]$$

or,

$$I = AP$$

or,

$$A = P^{-1}$$

The SD source code labels A as hc20.

The gradients are also linear combination of the p_i, $\frac{\partial \vec{N}}{\partial \varepsilon_j}$, $(j = 1,2,3)$, determined by differentiating equation 5.8.1,

$$\frac{\partial \vec{N}}{\partial \varepsilon_j} = A\frac{\partial \vec{p}}{\partial \varepsilon_j}$$

The $\partial\vec{p}/\partial\varepsilon_j$ may be written as a linear combination of the p_k via the following three equations.

$$\frac{\partial p_i}{\partial \varepsilon_1} = r_i \varepsilon_1^{r_i-1}\varepsilon_2^{s_i}\varepsilon_3^{t_i} \tag{5.8.2}$$

$$\frac{\partial p_i}{\partial \varepsilon_2} = s_i \varepsilon_1^{r_i}\varepsilon_2^{s_i-1}\varepsilon_3^{t_i} \tag{5.8.3}$$

$$\frac{\partial p_i}{\partial \varepsilon_3} = t_i \varepsilon_1^{r_i}\varepsilon_2^{s_i}\varepsilon_3^{t_i-1} \tag{5.8.4}$$

while noting that equations 5.8.2, 5.8.3 and 5.8.4 are zero if r_i, s_i, or t_i is zero, respectively. The matrices B_j with $j = 1,2,3$ are sought such that,

$$\frac{\partial \vec{N}}{\partial \varepsilon_j} = B_j\vec{p}.$$

Evaluating $\partial\vec{N}/\partial\varepsilon_j$ and $\vec{p}$ at all 20 nodes, we have,

$$\left[\frac{\partial \vec{N}}{\partial \varepsilon_j}(\vec{\varepsilon}_1), \frac{\partial \vec{N}}{\partial \varepsilon_j}(\vec{\varepsilon}_2), \ldots, \frac{\partial \vec{N}}{\partial \varepsilon_j}(\vec{\varepsilon}_{20})\right] = B_j\,[\vec{p}(\vec{\varepsilon}_1), \vec{p}(\vec{\varepsilon}_2), \ldots, \vec{p}(\vec{\varepsilon}_{20})] \tag{5.8.5}$$

Matrix equation 5.8.5 can be inverted to solve for B_j with $j = 1,2,3$. In `Hex20.C`, `AB1` is B_1 , `AB2` is B_2, and `AB3` is B_3.

5.8.1. Shape Function Ordering

The above method results in elements which satisfy the requirements that the evaluation of shape function i on node i is one. However, the implementation does not ensure compatibility with standard node ordering from **Exodus**. We've provided a re-ordering function to ensure this.

5.8.2. *Anisotropy*

Anisotropic materials require special care in the rotation of the matrix of material parameters when those parameters are given in some coordinate system other that in which the element matrices are calculated. The formulae for rotating those matrices are derived in 4.1.

5.9. Hexshell usage and limitations

A Hexshell[60] element has the behavior of a standard shell element and the mesh topology of a brick. Thin regions meshed with the solid brick topology may be modelled with Hexshells without concern for the large element aspect ratios.

Hexshells require an thickness direction. It is important to be able to identify that direction. SD implements four such methods

natural The *natural* ordering of the nodes in the element can determine the thickness direction. This is the method used by Carlos to develop the element. I believe that the connectivity for the element will indeed have to be modified to properly interface to his software.

sideset The placement of a sideset on one (or both) thickness faces of the elements uniquely identifies the thickness direction.

topology The topology may be used to identify the thickness direction if the Hexshell is in a sheet. Another hypothesis is that the sheet does not intersect itself. The thickness direction connects the sheet's free surfaces. Further, once the thickness direction is established for one element, the thickness direction propagates to the adjacent elements.

projection The thickness direction could be determined by the closest projection to a coordinate direction.

We will try to support all the above methods. The *topology* method puts the least burden on the analyst. It is the least explicit however, and the most work to implement (especially in parallel). The next simplest (for the analyst) is the *projection* method. Sideset methods are burdensome for both the analyst and the developers. The *natural* method is the easiest to implement, but can be next to impossible for the analyst to use.

Input will be structured as follows. Keywords are associated with each method. At most one of the four keywords above may be entered. The default is *topology*. The mass properties of a layered Hexshell are

```
Block 9
Hexshell
orientation sideset='1,2' material=9
end
```

```
Block 9
Hexshell
orientation topology material=9
end
```

computed approximately as follows.

1. The volume fraction, f_i, and density, ρ_i, of each layer is determined.
2. The contribution of the mass of the element is added to the nodes as if an element of density $\bar{\rho} = \sum_i \rho_i f_i$ filled the entire element.

The net effect of this is that the mass is computed as if an average density were applied. This could introduce minor errors if the element is thick and is much denser on one side than another.

Materials for all Hexshell specifications can be defined as a function of temperature, with the temperatures defined through the **Exodus** file as element variables.

5.10. Membrane

In this section we provide the theory behind the tangent stiffness matrix for the quad membrane element in **Sierra/SD**. This element has stiffness in the in-plane directions, but has no stiffness out-of-plane. Also, it has no rotational degrees of freedom. The following formulation coincides with the Abaqus[7] membrane.

To begin, we define two orthogonal surface directions in the plane of the membrane l and m, and a normal vector n. Given these unit vectors, a local coordinate system (l, m, n) is implied. We consider the weak formulation of the internal force term for the membrane in the deformed configuration[19]

$$\delta W_{int} = \int_{\Omega} \delta \boldsymbol{D} : \boldsymbol{\sigma} d\Omega \tag{5.10.1}$$

where W_{int} is the virtual work, Ω is the domain of the membrane, $\boldsymbol{\sigma}$ is the stress tensor, and $\boldsymbol{L} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} = \boldsymbol{D} + \boldsymbol{W}$ is the deformation gradient. The rate-of-deformation $\boldsymbol{D}$ and spin tensors $\boldsymbol{W}$ are defined as

$$\boldsymbol{D} = \frac{1}{2}\left[\left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\right) + \left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\right)^T\right] \tag{5.10.2}$$

$$\boldsymbol{W} = \frac{1}{2}\left[\left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\right) - \left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\right)^T\right] \tag{5.10.3}$$

The updated Lagrangian formulation is used. Thus, the integral in equation 5.10.1 is over the current (deformed) configuration of the membrane.

W is a skew-symmetric tensor, and the tensor product of a skew-symmetric tensor with a symmetric tensor vanishes. Equation 5.10.1 reduces to

$$\delta W_{int} = \int_{\Omega} \delta \boldsymbol{L} : \boldsymbol{\sigma} d\Omega \tag{5.10.4}$$

Equation 5.10.4 is written in terms of the global coordinate system. In the formation of the tangent stiffness matrix, we wish to use the fact that all stress components normal to the plane of the membrane are zero. Hence, when considering equation 5.10.1 in terms of the (l, m, n) coordinate system of the membrane, we can eliminate the out-of-plane terms and write as

$$\delta W_{int} = \int_{\Omega} \delta L_{lm} : \sigma_{lm} d\Omega \tag{5.10.5}$$

where $l, m = 1, 2$ are the indices for the in-plane coordinate system of the membrane, $L_{lm} = \frac{\partial u_l}{\partial x_m}$, and σ_{lm} is the 2×2, in-plane stress tensor.

Next, we need to relate the derivatives in the plane of the element to those in the global coordinate system. This is because the numerical integration of the tangent stiffness matrix takes place in the plane of the element (and hence involves derivatives with respect to in-plane coordinates), whereas the derivatives in equation 5.10.5 are in terms of global coordinates. We can express the in-plane displacement in terms of the out-of-plane displacement as

$$u_l = \boldsymbol{u} \cdot \boldsymbol{l} \tag{5.10.6}$$

$$u_m = \boldsymbol{u} \cdot \boldsymbol{m} \tag{5.10.7}$$

$$u_n = \boldsymbol{u} \cdot \boldsymbol{n} \tag{5.10.8}$$

The relationship between the derivatives can be computed

$$\frac{\partial \boldsymbol{u}}{\partial x_l} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial x_l} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} \boldsymbol{e}_l \tag{5.10.9}$$

where $\boldsymbol{e}_l$ is the unit vector in the l direction. Similar expressions hold for the other components. Taking the dot product of both sides of the previous equation with the unit vector in the m direction, $\boldsymbol{e}_m$, we arrive at

$$\frac{\partial u_m}{\partial x_l} = \boldsymbol{e}_m \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} \boldsymbol{e}_l \tag{5.10.10}$$

Next, we consider the expression given for the tangent operator in[7]

$$\int_\Omega \delta \boldsymbol{D} : \boldsymbol{C} : d\boldsymbol{D} + \boldsymbol{\sigma} : \left(\delta \boldsymbol{L}^T \cdot d\boldsymbol{L} - 2\delta \boldsymbol{D} \cdot d\boldsymbol{D} \right) d\Omega \tag{5.10.11}$$

Due to the vanishing out-of-plane stress, and invariance through the thickness, the thickness factors out, and this can be written as an area integral

$$t \int_A \delta \boldsymbol{D} : \boldsymbol{C} : d\boldsymbol{D} + \boldsymbol{\sigma} : \left(\delta \boldsymbol{L}^T \cdot d\boldsymbol{L} - 2\delta \boldsymbol{D} \cdot d\boldsymbol{D} \right) dA \tag{5.10.12}$$

The first term is recognized as the material stiffness, and the second is the geometric stiffness term. In particular, the material stiffness term is precisely the same as the standard form of the material stiffness in three dimensions, expect that it is restricted to two dimensions. The geometric stiffness term is more involved, and we elaborate some more on that.

First, we consider the deformation gradient in the plane of the element

$$L_{lm} = \boldsymbol{e}_l \frac{\partial \boldsymbol{u}}{\partial x_m} \tag{5.10.13}$$

We have

$$\delta L_{lm} = \boldsymbol{e}_l \frac{\partial \delta \boldsymbol{u}}{\partial x_m} \tag{5.10.14}$$

$$\delta L_{lm}^T = \left(\frac{\partial \delta \boldsymbol{u}}{\partial x_m} \right)^T \boldsymbol{e}_l^T \tag{5.10.15}$$

$\boldsymbol{e}_l^T \boldsymbol{e}_m = \delta_{lm}$ implies that

$$\boldsymbol{L}^T \boldsymbol{L} = \left(\frac{\partial \boldsymbol{u}}{\partial x_m} \right)^T \boldsymbol{e}_l^T \boldsymbol{e}_m \frac{\partial \boldsymbol{u}}{\partial x_l} = \left(\frac{\partial \boldsymbol{u}}{\partial x_m} \right)^T \frac{\partial \boldsymbol{u}}{\partial x_l} \tag{5.10.16}$$

since $\boldsymbol{e}_l^T \boldsymbol{e}_m = \delta_{lm}$.

The rate of deformation $\boldsymbol{D}$ is the symmetric part of $\boldsymbol{L}$. Thus, we can write

$$D_{lm} = \frac{1}{2}\left(\boldsymbol{e}_l \frac{\partial \boldsymbol{u}}{\partial x_m} + \boldsymbol{e}_m \frac{\partial \boldsymbol{u}}{\partial x_l}\right) \tag{5.10.17}$$

With these relations, we can expand the expression for the geometric stiffness, as

$$t\int_A \sigma_{lm}\left[\left(\frac{\partial \delta\boldsymbol{u}}{\partial x_m}\right)^T \frac{\partial \boldsymbol{u}}{\partial x_l} - \frac{1}{2}\sum_{\gamma=1}^{2}\left(\boldsymbol{e}_\gamma \frac{\partial \delta \boldsymbol{u}}{\partial x_l} + \boldsymbol{e}_l \frac{\partial \boldsymbol{u}}{\partial x_\gamma}\right)\left(\boldsymbol{e}_\gamma \frac{\partial \delta \boldsymbol{u}}{\partial x_m} + \boldsymbol{e}_m \frac{\partial \boldsymbol{u}}{\partial x_\gamma}\right)\right] dA \tag{5.10.18}$$

The material stiffness term can be integrated with a selective deviatoric approach, in much the same was as for a volumetric element. First, we note that after finite element discretization, the material stiffness term in equation 5.10.12 can be written as

$$K_{mat} = \int_V B^T C B dV \tag{5.10.19}$$

where K is the stiffness matrix, V is the volume of the element, B is the two-dimensional strain-displacement matrix

We define the mean quadrature counterpart to B,

$$\tilde{B} = \int_V B dV \tag{5.10.20}$$

B and $\tilde{B}$ split into volumetric and deviatoric components, i.e.

$$\begin{aligned} \tilde{B} &= \tilde{B}_V + \tilde{B}_D \\ B &= B_V + B_D \end{aligned} \tag{5.10.21}$$

With these decompositions, we define

$$\hat{B} = \tilde{B}_V + \tilde{B}_D + sd(B_D - \tilde{B}_D) \tag{5.10.22}$$

where sd is a parameter between 0 and 1. When $sd = 0$, the element corresponds to a mean quadrature element. When $sd = 1$, the element corresponds to mean quadrature on the volumetric part, but with full integration on the deviatoric component.

With this new definition of $\hat{B}$, we can define the stiffness matrix for this element as

$$K = \int_V \hat{B}^T C \hat{B} dV \tag{5.10.23}$$

This is the approach taken for integrating the material stiffness term in equation 5.10.12

5.11. 6 noded Triangle

This section reviews the derivation of the triangular shell element (TriaShell) element. The membrane DOFs (u, v, θ_z) are decoupled from the bending DOFs (w, θ_x, θ_y). Allman's triangle[2] models the membrane response. The discrete Kirchhoff triangle[18] (DKT) models the bending response.

Allman's Triangular Element Allman's formulation after the substitutions $\cos(\gamma_{ij}) = \frac{y_{ji}}{l_{ij}}$ and $\sin(\gamma_{ij}) = \frac{-x_{ji}}{l_{ij}}$, is

$$\begin{aligned} u = u_1\psi_1 + u_2\psi_2 + u_3\psi_3 + \tfrac{1}{2}y_{21}(\omega_2 - \omega_1)\psi_1\psi_2 + \\ \tfrac{1}{2}y_{32}(\omega_3 - \omega_2)\psi_2\psi_3 + \tfrac{1}{2}y_{13}(\omega_1 - \omega_3)\psi_3\psi_1 \end{aligned} \tag{5.11.1}$$

$$\begin{aligned} v = v_1\psi_1 + v_2\psi_2 + v_3\psi_3 + \tfrac{1}{2}x_{21}(\omega_2 - \omega_1)\psi_1\psi_2 \\ -\tfrac{1}{2}x_{32}(\omega_3 - \omega_2)\psi_2\psi_3 - \tfrac{1}{2}x_{13}(\omega_1 - \omega_3)\psi_3\psi_1 \end{aligned} \tag{5.11.2}$$

The stiffness and mass matrices ($[K]_{AT}$, $[M]_{AT}$) are found using general finite element procedures. The element has a mechanism that introduces spurious low energy modes. The mechanism arises if the deformations are all zero and the rotations are all the same. A "fix"[37] has been implemented.

Discrete Kirchhoff Element The DKT[18] element has 9 DOFs. It is obtained by transforming a 12 DOF element with mid-side nodes to a triangle with the nodes at the vertices only. This is obtained as follows. Using Kirchhoff theory, the transverse shear is set to zero at the nodes. And the rotation about the normal to the edge is imposed to be linear. Using these constraints, a nine DOF bending element is derived (DKT) using the shape functions for the six-node triangle. Unfortunately, the variation of w over the element cannot be explicitly written. Therefore, the w variation over the element needs to be calculated before the mass matrix can be obtained.

As stated, the equation for w is not explicitly stated over the element in the derivation by Batoz *et al.*. Using a nine DOF element, a complete cubic cannot be written, since 10 quantities would be needed to get a unique polynomial. The strategy taken here is that the stiffness matrix produced using for the DKT element provides reasonable results, and the derivation of the mass matrix is not as critical. So, the equation for w[136] as

$$\begin{aligned} w = \;& \alpha_1\psi_1 + \alpha_2\psi_2 + \alpha_3\psi_3 + \\ & +\alpha_4\psi_1\psi_2 + \alpha_5\psi_2\psi_3 + \alpha_6\psi_3\psi_1 + \\ & +\alpha_7{\psi_1}^2\psi_2 + \alpha_8{\psi_2}^2\psi_3 + \alpha_9{\psi_3}^2\psi_1 \end{aligned} \tag{5.11.3}$$

Our AT and DKT element stiffness and mass matrix derivations used Maple. The consistent mass matrix derivation follows the standard finite element procedure. And mass lumping of translational DOFs are found as usual. Mass lumping for the rotational DOFs, however, are set to $\frac{1}{125}$ of the translation terms.

The complication in the derivation of the combined AT and DKT shell element is the derivation of DKT element mass matrix. We used an incomplete family of polynomials. We think that this did not affect the result.

Verification and Validation. Results for our AT element agree with the published results.[2] The square plate in pure bending and a cantilevered beam with a parabolic tip load are used as verification examples. The mass matrix verification is limited to noting that mass is conserved in the u, v directions.

The DKT element is validated against experimental data for a triangular fin.[18] The first 10 eigenvalues for the triangular fin (cantilever) match very well. In addition, the DKT element is verified by using a cantilevered beam and matching deflection results at the tip. If $\nu = 0$, then results should match very closely with Euler-Beam theory results, and they did.

Finally, the AT/DKT element is verified by comparing with published results from Ref..[56] Tables 5-4 and 5-5 show that our elements match exactly with ABAQUS to the number of digits shown. The first column is the result produced by Ertas *et al.*, the second column is the result produced by ABAQUS, and the third column is the result produced by **Sierra/SD** using this DKT/AT element.

DOF	AT/DKT	ABAQUS	AT/DKT!
x	0.000	0.000	0.000
y	0.000	0.000	0.000
z	-1.405×10^{-2}	-1.398×10^{-2}	-1.398×10^{-2}
θ_x	3.337×10^{-2}	3.337×10^{-2}	3.337×10^{-2}
θ_y	3.106×10^{-2}	3.089×10^{-2}	3.089×10^{-2}
θ_z	0.000	0.000	0.000

Table 5-4. – Comparison of deflections at Node 2.

DOF	AT/DKT	ABAQUS	AT/DKT!
x	0.000	0.000	0.000
y	0.000	0.000	0.000
z	1.949×10^{-2}	1.955×10^{-2}	1.955×10^{-2}
θ_x	3.363×10^{-2}	3.363×10^{-2}	3.363×10^{-2}
θ_y	-2.686×10^{-2}	-2.702×10^{-2}	-2.702×10^{-2}
θ_z	0.000	0.000	0.000

Table 5-5. – Comparison of deflections at Node 3.

5.12. 3 noded Triangle

The triangular shell used most in **Sierra/SD** is the **Tria3** element developed by Carlos Felippa of the University of Colorado in Boulder. This element is similar to the **TriaShell** element presented in Section 5.11. Full details of the theory behind the element is out of the scope of this document, but details may be found in references[5,62] and.[61] Unfortunately, these references omit any mention of how this element handles the bending part.

5.13. Shell Offset

Consider a shell offset, with an offset vector, $\vec{v}$. Notice that $\vec{v}$ could be defined at each nodal location in what follows, but for this development, we assume a single offset $\vec{v}$ which applies to all nodes. That is, consider the offset of a single node. Define a coordinate system at the node, with variables u. On the offset beam the coordinate system is $\tilde{u}$.

u is related to $\tilde{u}$. The constraint of a constant offset may be stated that the displacement difference of the two systems must be orthogonal to $\vec{v}$, i.e. $(u - \tilde{u}) = \vec{v} \times \vec{\kappa}$, where $\vec{\kappa}$ is the rotation at the nodes.

Thus, we can write,

$$\begin{pmatrix} \tilde{u} \\ \kappa \end{pmatrix} = [L] \begin{pmatrix} u \\ \kappa \end{pmatrix} \tag{5.13.1}$$

For multiple nodes each diagonal block of L depends on the offset of the corresponding node. We can use this transformation matrix to eliminate the degrees of freedom associated with $\tilde{u}$. The energy of the shell can be written,

$$E_{strain} = 0.5 \begin{Bmatrix} \tilde{u} \\ \kappa \end{Bmatrix}^T [\tilde{K}] \begin{Bmatrix} \tilde{u} \\ \kappa \end{Bmatrix} \tag{5.13.2}$$

But with this substitution,

$$E_{strain} = 0.5 \left\{ \begin{array}{c} u \\ \kappa \end{array} \right\}^T \left[L^T \tilde{K} L \right] \left\{ \begin{array}{c} u \\ \kappa \end{array} \right\} \tag{5.13.3}$$

If we let $K = L^T \tilde{K} L$, then

$$E_{strain} = 0.5 \left\{ \begin{array}{c} u \\ \kappa \end{array} \right\}^T [K] \left\{ \begin{array}{c} u \\ \kappa \end{array} \right\} \tag{5.13.4}$$

Thus, $\tilde{u}$ has been eliminated, and the equations may be put in terms of the output variables.

5.14. Beam2

The 2-noded beam[37] element uses under-integrated cubic shape functions. Isotropic material models are supported. Torsional effects are accounted for in the axis of the beam. The area and bending moments are constants independent of position in the beam.

Attributes are read from the **Exodus** file for each element.

1. The cross sub-sectional area of the beam (Attribute 1)
2. The first bending moment, I_1. (Attribute 2).
3. The second bending moment, I_2. (Attribute 3).
4. The torsional moment, J_k. (Attribute 4).
5. The orientation of the beam (Attributes 5, 6 and 7)

   The orientation should not be aligned with the beam axis. In the event of an improperly specified orientation, a warning will be written, and a new orientation selected. The orientation is an x,y,z triplet specifying a direction. It does not need to be perpendicular to the beam axis, nor is it required to be normalized. The orientation vector, and the beam axis define the plane for the first bending direction.

Torsion

As outlined in Blevins,[24] the stiffness properties of beam torsion are governed by J_k (Attribute 4), while the mass properties are derived from the polar moment of inertia, $J_{polar} = I_1 + I_2$. This representation is accurate for beams with closed cross sections, but will have significant error for more open sections. Warping in open sections is not accounted for in this standard beam formulation.

$$
\begin{array}{cccccccccccc}
AE/L & 0 & 0 & 0 & 0 & 0 & -AE/L & 0 & 0 & 0 & 0 & 0 \\
 & R_1 & \beta & 0 & -L\beta/2 & LR_1/2 & 0 & -R_1 & -\beta & 0 & -L\beta/2 & LR_1/2 \\
 & & R_2 & 0 & -LR_2/2 & L\beta/2 & 0 & -\beta & -R_2 & 0 & -LR_2/2 & L\beta/2 \\
 & & & GJ/L & 0 & 0 & 0 & 0 & 0 & -GJ/L & 0 & 0 \\
 & & & & k_2 & -\beta L^2/3 & 0 & L\beta/2 & -LR_2/2 & 0 & k_4 & -\beta L^2/6 \\
 & & & & & k_1 & 0 & LR_1/2 & -L\beta/2 & 0 & -\beta L^2/6 & k_3 \\
 & & & & & & AE/L & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & R_1 & \beta & 0 & L\beta/2 & -LR_1/2 \\
 & & & & & & & & Ri_2 & 0 & LR_2/2 & -L\beta/2 \\
 & & & & & & & & & GJ/L & 0 & 0 \\
 & & & & & & & & & & k_2 & -\beta L^2/3 \\
 & & & & & & & & & & & k_1
\end{array}
$$

Figure 5-1. – Nbeam Element Stiffness Matrix.

5.15. Nbeam

Beam/bar elements are a major component in many structural Finite Element Models (FEM). It is important to employ a beam/bar element which includes transverse shear and torsion in addition to axial and bending stiffness. Additionally, the mass formulation needs to include rotary inertia. The Nbeam element is an implementation of the NASTRAN CBAR element. The stiffness matrix is identical to the CBAR. The mass matrix is a new formulation to this implementation providing a diagonal mass matrix w/ rotary inertia included.

The Nbeam element stiffness matrix is based on Timoshenko beam theory.[112] The formulation differs in the inertia coupling formulation. The derivation of this specific form is provided in [[96]]. The exact form of the stiffness matrix implemented in **Sierra/SD** is shown in Figure 5-1.

The following derived[94] quantities are used depending on the value of I_{12}.

If $I_{12} = 0$	If $I_{12} \neq 0$
$\beta = 0$	$\beta = \frac{12EI_{12}}{L^3}$
$R_1 = \frac{12EI_1}{L^3}\left[1 + \frac{12EI_1}{s_1 AGL^2}\right]^{-1}$	$R_1 = \frac{12EI_1}{L^3}$
$R_2 = \frac{12EI_2}{L^3}\left[1 + \frac{12EI_2}{s_2 AGL^2}\right]^{-1}$	$R_2 = \frac{12EI_2}{L^3}$

The rest of the quantities are valid for any value of I_{12}.

$$
\begin{array}{lcll}
k_1 & = & \dfrac{L^2R_1}{4} + \dfrac{EI_1}{L} & \\
k_2 & = & \dfrac{L^2R_2}{4} + \dfrac{EI_2}{L} & \\
k_3 & = & \dfrac{L^2R_1}{4} - \dfrac{EI_1}{L} & \\
k_4 & = & \dfrac{L^2R_2}{4} - \dfrac{EI_2}{L} & \\
s_1 & = & A_y/A & \text{shear factor} \\
s_2 & = & A_z/A & \text{shear factor}
\end{array}
$$

$$\begin{array}{cccccccccccc}
m' & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & m' & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & m' & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & m'J/A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & m'I_2/A_z & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & m'I_1/A_y & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & m' & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & m' & 0 & 0 & 0 & 0 \\
 & & & & & & & & m' & 0 & 0 & 0 \\
 & & & & & & & & & m'J/A & 0 & 0 \\
 & & & & & & & & & & m'I_2/A_z & 0 \\
 & & & & & & & & & & & m'I_1/A_y
\end{array}$$

Figure 5-2. – Nbeam mass matrix.

Table 5-6. – Nbeam Parameters.

Description	Keyword	Exodus Attributes
Cross-Sectional Area	Area	1
First Bending Moment	I1	2
Second Bending Moment	I2	3
Cross Inertia	I12	N/A
Torsional Moment	J	4
Beam Orientation	orientation	5-7
Y-axis Shear Area Factor	Shear_factor_1	N/A
Z-axis Shear Area Factor	Shear_factor_2	N/A
Offset Vector At 1st Node	offset	8-10
Offset Vector At 2nd Node	-	11-13

The Nbeam mass matrix is given in Figure 5-2. The mass quantity m' is defined as $m' = \rho AL/2$.

If the local coordinate system is not the global coordinate system, then the transformation to global coordinates introduces off diagonal terms to the mass matrix in the rows corresponding to rotary inertia. In **Sierra/SD** the mass matrix is lumped by setting off diagonals to zero and not adding them to a diagonal. Total rotary mass contributions are reduced. An alternative is to set off diagonals to zero and add them to a diagonal; this increases total rotary mass contributions.

Element properties are specified in the text input file. The required parameters are listed in Table 5-6.

The parallel axis theorem is used to account for offsets. The offset vector is defined as a vector from the bending neutral axis of the beam to the nodal location. All other quantities are derived from the material data and the element length.

Torsion

As outlined in Blevins,[24] the stiffness properties of beam torsion are governed by J_k, while the mass properties are derived from the polar moment of inertia, $J_{polar} = I_1 + I_2$. This representation is accurate for beams with closed cross sections, but will have significant error for more open sections. Warping in open sections is not accounted for in this standard beam formulation.

5.16. Navy quadrilateral

Many structural components on naval vessels, including the hull, bulkheads and decks are made from plate, be it steel, aluminum or a composite material. As such, plate and shell elements are essential to any finite element analysis of ships or submarines. It is important to employ an element that is shear deformable and can also accommodate orthotropic layers. The nquad is a four-noded isoparametric element that is designed to be similar to the NASTRAN CQUAD4 element.

This section is based on material in chapter 4 of [114] Note that this material does not appear in later editions.

The development of the stiffness matrix draws from the plane elasticity and bending formulations found in [114]. The membrane and bending components are decoupled. The membrane stiffness terms are derived from the integrals in equation 4.156 in [114]:

$$K_{ij}^{11} = \int_{\Omega^e} \left(C_{11} \frac{\partial \psi_i}{\partial x} \frac{\partial \psi_j}{\partial x} + C_{33} \frac{\partial \psi_i}{\partial y} \frac{\partial \psi_j}{\partial y} \right) dxdy \tag{5.16.1}$$

$$K_{ij}^{12} = K_{ij}^{21} = \int_{\Omega^e} \left(C_{12} \frac{\partial \psi_i}{\partial x} \frac{\partial \psi_j}{\partial y} + C_{33} \frac{\partial \psi_i}{\partial y} \frac{\partial \psi_j}{\partial x} \right) dxdy \tag{5.16.2}$$

$$K_{ij}^{22} = \int_{\Omega^e} \left(C_{33} \frac{\partial \psi_i}{\partial x} \frac{\partial \psi_j}{\partial x} + C_{22} \frac{\partial \psi_i}{\partial y} \frac{\partial \psi_j}{\partial y} \right) dxdy \tag{5.16.3}$$

where the C_{ij} are the elastic material constants for plane stress

$$C_{11} = C_{22} = \frac{E}{1-\nu^2} \quad C_{12} = \frac{\nu E}{1-\nu^2} \quad C_{33} = \frac{E}{2(1+\nu}$$

and the ψ_i are the bilinear element shape functions (see equation 4.31 in [114]) over the element Ω^e. For a rectangle of width a and height b,

$$\begin{aligned} \psi_1 &= (1-\xi/a)(1-\eta/b) \\ \psi_2 &= \frac{\xi}{a}(1-\eta/b) \\ \psi_3 &= (1-\xi/a)\frac{\eta}{b} \\ \psi_4 &= \frac{\xi}{a}\frac{\eta}{b}. \end{aligned}$$

The membrane stiffness matrix is of the form:

$$\begin{bmatrix} K^{11} & K^{12} \\ K^{21} & K^{22} \end{bmatrix}$$

assuming the displacement vector is of the form $\{u_1, v_1, u_2, v_2, ...\}$.

The bending terms are organized here into a block 3 by 3 matrix,

$$\begin{bmatrix} K^{11} & K^{12} & K^{13} & \\ & K^{22} & K^{23} sym & K^{33} \end{bmatrix} \begin{bmatrix} w \\ S_x \\ S_y \end{bmatrix} = \begin{bmatrix} f^1 \\ f^2 \\ f^3 \end{bmatrix}.$$

The bending stiffness terms, based on the shear deformation theory of plates, are based on the integrals in equation 4.226 in [114]:

$$\begin{aligned}
K_{ij}^{11} &= \int_{\Omega^e} \left(D_{44} \frac{\partial\psi_i}{\partial x}\frac{\partial\psi_j}{\partial x} + D_{55}\frac{\partial\psi_i}{\partial y}\frac{\partial\psi_j}{\partial y} \right) dx\, dy \\
K_{ij}^{12} &= \int_{\Omega^e} \left(D_{44} \frac{\partial\psi_i}{\partial x}\psi_j \right) dx\, dy \\
K_{ij}^{13} &= \int_{\Omega^e} \left(D_{55} \frac{\partial\psi_i}{\partial y}\psi_j \right) dx\, dy \\
K_{ij}^{22} &= \int_{\Omega^e} \left(D_{11} \frac{\partial\psi_i}{\partial x}\frac{\partial\psi_j}{\partial x} + D_{33}\frac{\partial\psi_i}{\partial y}\frac{\partial\psi_j}{\partial y} + D_{44}\psi_i\psi_j \right) dx\, dy \\
K_{ij}^{23} &= \int_{\Omega^e} \left(D_{12} \frac{\partial\psi_i}{\partial x}\frac{\partial\psi_j}{\partial y} + D_{33}\frac{\partial\psi_i}{\partial y}\frac{\partial\psi_j}{\partial x} \right) dx\, dy \\
K_{ij}^{33} &= \int_{\Omega^e} \left(D_{33} \frac{\partial\psi_i}{\partial x}\frac{\partial\psi_j}{\partial x} + D_{22}\frac{\partial\psi_i}{\partial y}\frac{\partial\psi_j}{\partial y} + D_{55}\psi_i\psi_j \right) dx\, dy
\end{aligned}$$

where the D_{ij} are the isotropic elastic material constants (defined for example in equation 4.221 of [114]:

$$\begin{aligned}
D_{11} &= D_{22} = \frac{Eh^3}{12(1-\nu^2)} \\
D_{12} &= \nu D_{11} \\
D_{33} &= \frac{Gh^3}{12} \\
D_{44} &= D_{55} = Ghk
\end{aligned}$$

where h is the thickness of the plate and k is the shear correction factor. The bending stiffness matrix is of the form:

$$\begin{bmatrix} [K^{11}] & [K^{12}] & [K^{13}] \\ & [K^{22}] & [K^{23}] \\ sym & & [K^{33}] \end{bmatrix}$$

assuming the displacement matrix is of the form $\{w_1,\ \theta_{x1},\ \theta_{y1}, w_2,\ \theta_{x2},\ \theta_{y2},\ \ldots\}$ To minimize the effect of locking, reduced integration on the shear terms (i.e., those involving D_{44} and D_{55}) is used.

The stabilization method from Belytschko[20] is used for the Nquad element. Using single point integration $K_s^{[1x1]}$ for the shear stiffness matrix leads to hourglass modes for some problems. Using full integration $K_s^{[2x2]}$ can cause shear locking in some problems. Belytschko recommends a shear stiffness matrix given as $K_s = (1-\varepsilon)K_s^{[1x1]} + \varepsilon K_s^{[2x2]}$, a linear combination of the reduced integration and full integration shear stiffness matrices. The fraction, $\varepsilon = rt^2/A$ is a function of thickness and area. Here $r = 0.03$, t is the element thickness and A the area of the shell. This automatic selection of ε is more successful for thinner plates; ε should never exceed 1.

The layered shell formulation, also based on first-order shear deformation theory, draws from [107], particularly equations 3.4-5 and 3.4-6 found therein.

The stiffness matrices developed for the isotropic and laminate cases do not account for in-plane rotational stiffness. A fictitious stiffness for the θ_z d.o.f. is provided by equation 12.3-4 in [37]. The resulting element stiffness matrix is 24 x 24, accounting for 6 d.o.f at each of the four nodes.

A consistent mass matrix is formed based on equation 4.235 in:[114]

$$M_{ij} = \int_{\Omega^e} \rho h \psi_i \psi_j \, dx \, dy$$

where ρ is the material density. The diagonal mass matrix is derived by row summation.

Element level strains are expressed by equation 4.147 in:[114]

$$\{\varepsilon\}_e = [B]_e \{\Delta\}_e$$

where the five terms in $\{\varepsilon\}_e$ are ε_x, ε_y, τ_{xy} and the two transverse shear strains γ_{yz} and γ_{zx}. The 5 x 24 matrix $[B]_e$ is formed by the element shape functions and their derivatives and the 24 x 1 vector $\{\Delta\}_e$ are the nodal displacements. The membrane and bending strain-displacement relationships are found, respectively, in equations 11.1-3 and 11.1-4 in [37]:

Membrane:

$$\varepsilon_x = u,_x \qquad \varepsilon_y = v,_y \qquad \gamma_{xy} = (u,_y + v,_x)$$

Bending:

$$\begin{aligned} \varepsilon_x &= -z\theta_{y,x} & \gamma_{xy} &= -z(\theta_{y,y} + \theta_{x,x}) \\ \varepsilon_y &= -z\theta_{x,y} & \gamma_{yz} &= w,_y - \theta_x \\ & & \gamma_{zx} &= w,_x - \theta_y \end{aligned}$$

Note that the bending equations are altered from 11.1-4 in [37]. In that reference, a rotation about the x-axis is expressed as θ_y and a rotation about the y-axis is θ_x x. These definitions have been reversed in the above equations.

The user provides element properties in the **Sierra/SD** input deck. The required parameters are:

1. Element thickness.
2. Material ID, which contains the required material properties (E, ν, ρ).
3. For the layered shell case, each layer must have specified its own material ID (such as an orthotropic_layer), thickness and fiber orientation.

5.17. Truss

The truss element implementation[37] pages 214-216 uses linear shape functions. Torsional stiffness vanishes, unlike the NASTRAN truss element. Area is independent of position in the truss. The following parameter is read from the **Exodus** file.

1. The cross sub-sectional area of the truss (Attribute 1)

5.18. Spring

Spring elements have mass 0. Stiffnesses K_x, K_y, and K_z are set in the input deck.

- The force generated in a *Spring* element should be collinear with the nodes. Typically, a spring element connection between coincident nodes generates 0 torque.

- *Springs* attach 3 DOFs. If some spring constants vanish, then the associated DOF has 0 stiffness. However, the degree of freedom will remain in the A-set 1 matrices. Adjacent elements provide stiffness entries connecting the spring to the model. If the other DOFs are not attached to adjacent elements, then the stiffness is singular.

The element stiffness matrix $\tilde{K}$ =

$$\tilde{K}_{11} = \text{diag}(K_x, K_y, K_z), \left[\begin{array}{cc} \tilde{K}_{11} & -\tilde{K}_{11} \\ -\tilde{K}_{11} & \tilde{K}_{11} \end{array} \right]. \tag{5.18.1}$$

For R_i in $SO(3)$ as described in section 1.4, the frame $\tilde{u}_i$ is transformed from the unrotated frame u_i by $T = \text{diag}(R_1, R_2)$,

$$\left[\begin{array}{c} u_1 \\ u_2 \end{array} \right] = [T] \left[\begin{array}{c} \tilde{u}_1 \\ \tilde{u}_2 \end{array} \right].$$

The spring nodes rotate together, $R_1 = R_2$. For $K_{ij} = R^T \tilde{K}_{ij} R$,

$$K = \begin{pmatrix} K_{11} & K_{12} \\ K_{12} & K_{22} \end{pmatrix}$$

5.19. Superelements

A superelement has reduced mass and stiffness matrices generated by a model reduction process such as component mode synthesis 2.9. Superelement generation typically saves the element in a file. Subsequent analysis a system (or residual structure) typically read the element from its file.

Superelements may contain sensitivity matrices 2.9.2. A point estimate of the superelement mass or stiffness matrix may be computed as a Taylor series expansion and used as part of a standard analysis. The approximate reduced matrix is given by the expansion.

$$K_r(p) \approx K_r(p_o) + \frac{dK_r}{dp}(p - p_o) \tag{5.19.1}$$

where p is the sensitivity variable, p_o is the nominal value of that variable and $K_r(p)$ represents the reduced order matrix evaluated at an arbitrary point in parameter space.

5.20. Gap

The gap element is a nonlinear spring which has a stiffness matrix that is dependent on displacement. In the element coordinate frame, the stiffness matrix has the same form as the matrix in equation 5.18.1 with the replacements:

Spring	**Gap**	
	Open	**Closed**
K_x	KU	KL
K_y	$KT \times KU/KL$	KT
K_z	$KT \times KU/KL$	KT

Note that typically $KL \gg KU$.

The two nodes of the gap element must rotate together. Spring elements are the same. The matrix transforms exactly as the matrix for a spring element.

5.21. Rigid Elements

Sierra/SD supports standard *pseudo*elements for rigid bodies. These include,

- Rrod - a rigid truss element, infinitely stiff in extension, but with no coupling to bending degrees of freedom. An element creates one constraint equation.
- RBar - a rigid beam, with up to 6 constraint equations per element.
- RBE2 - a rigid solid. With up to $6(n-1)$ degrees of freedom deleted, where n is the number of nodes. An RBE2 can stiffen a structure.
- RBE3 - an averaging type solid. This connects to many nodes, but removes up to 6 dofs on the reference node.

A rigid element has infinite stiffness and zero mass. In the input **Exodus** mesh beam elements represent rigid elements. In the input text file the corresponding block selects the type of rigid element.

Internally rigid elements are all stored and applied as special multi-point constraints. The RBE2 is a type of RBar (multiple instances). Elements all activate DOFs, but not ordinary MPCs. A rigid element is an MPC that activates DOFs.

Considerations for NASTRAN users

Rigid elements are intended to provide a capability similar to NASTRAN rigid elements. However, the differences can be significant. One difference is due to the solvers. **Sierra/SD** solvers manage the separation of dependent and independent DOFs, freeing the analyst from having to manage this complexity. Specification of rigid elements in NASTRAN implies this relation. If applied in the most common ways (such as an RBar constraining 6 dofs), the elements are the same. If some but not all DOFs are constrained, and if the NASTRAN autospc capability is invoked, significant differences are possible.

5.21.1. Rrod

An Rrod is a *pseudo*element which is infinitely stiff in the extension direction. The constraints for an Rrod may be conveniently stated as ensuring that the dot product of the translation and the beam axial direction for a Rrod vanishes. Each Rrod adds one constraint equation.

Consider the geometry of Figure 5-3. The equation of constraint for the Rrod is

$$L_x du_x + L_y du_y + L_z du_z = 0 \tag{5.21.1}$$

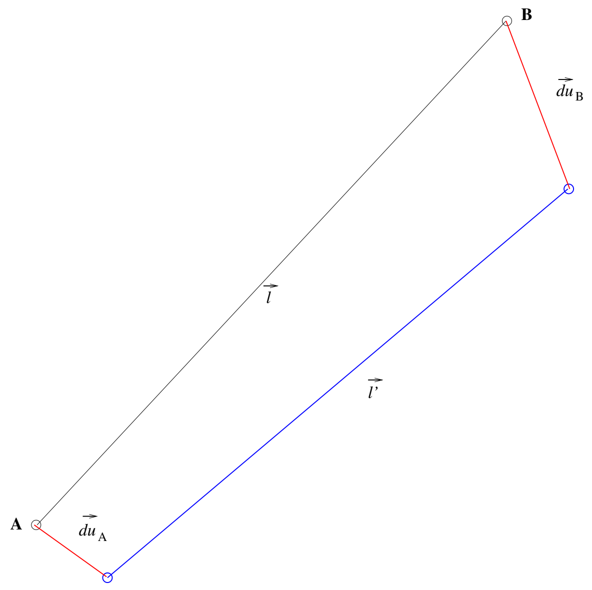


Figure 5-3. – Rigid Element Geometry.

The undeformed and deformed extents of the bar may be expressed as $\vec{L}$ and $\vec{l}$. After deformation, $\vec{du} = \vec{du}_B - \vec{du}_A$. The undeformed and deformed bars have components

$$\begin{aligned} L_x &= x_B - x_A & l_x &= L_x + du_x \\ L_y &= y_B - y_A & l_y &= L_y + du_y \\ L_z &= z_B - z_A & l_z &= L_z + du_z. \end{aligned}$$

5.21.2. RBar

An RBar is a *pseudo*element which is infinitely stiff in all the directions. An RBar can stiffen a structure. The constraints for an RBar may be summarized as follows.

1. the rotations at either end of the RBar coincide,
2. the extension of the bar is zero,
3. translations at one end of the bar are consistent with rotations.

Apparently the last two of these constraints may be specified mathematically by requiring that the translation be the cross product of the rotation vector and the bar direction.

$$\vec{T} = \vec{R} \times \vec{L}$$

where $\vec{T}$ is the translation difference of the bar (defined as $\vec{U}_2 - \vec{U}_1$),

$\vec{R}$ is the rotation vector, and

$\vec{L}$ is the vector from the first grid to the second.

The three constraints in the cross product, together with the three constraints requiring identical rotations at both ends of the bar form the six required constraint equations. Referring to Figure 5-3, the six constraint equations are [1]

$$\begin{aligned} du_x + l_y R_z - l_z R_y &= 0 \quad (5.21.2) \\ du_y + l_z R_x - l_x R_z &= 0 \quad (5.21.3) \\ du_z + l_x R_y - l_y R_x &= 0 \quad (5.21.4) \\ R_{x_a} &= R_{x_b} \quad (5.21.5) \\ R_{y_a} &= R_{y_b} \quad (5.21.6) \\ R_{z_a} &= R_{z_b} \quad (5.21.7) \end{aligned}$$

Partial Constraints on an RBar

NASTRAN permits application of some constraints on an RBar. For example, one can apply the first 3 constraints, and ignore the constraints on rotation alone. In addition, NASTRAN permits control of which end of the bars is constrained, and can split dependent and independent degrees of freedom between the nodes. Although NASTRAN permits fewer than 6 dependent dofs, SD requires 6 independent dofs.

Sierra/SD uses two attributes in the **Exodus** file to partially constrain an RBar. An attribute labeled "CID_FLAG_INDEP"is the constraint flag associated with the independent dofs. It should always be "123456", and it is always associated with the first node of the bar. The second attribute, "CID_FLAG_DEPEND", establishes the dependent degrees of freedom on the second node of the bar. This attribute determines which of the equations above are applied. For example, if `CID_FLAG_DEPEND = 123000` then the first three constraint equations are applied.

With partial application of the constraint equations, the results can be confusing. If equations 5.21.5-5.21.7 are not applied, then the rotation terms in 5.21.2 are appropriate only to the independent node. This is not always what is anticipated by the analyst. It is not possible to allocate DOFs to arbitrary ends of the bar. For this reason, the rotation may differ from what is produced by NASTRAN. Recall that applying CID_FLAG_INDEP = CID_FLAG_DEPEND = 1 results in an Rrod type constraint.

5.21.3. RBE3

The RBE3 applies distributed forces to many nodes. The structure is not stiffened.

The RBE3 uses the concept of a reference node. The theory follows the MSC documentation included in section 5.22. RBE3 element is a simplification of the NASTRAN RBE3 element. One simplification is that the RBE3 supports one weight that is applied to all the nodes. The NASTRAN RBE3 element supports different weights for each of its nodes.

Earlier implementations of the RBE3 differed significantly from the MSC NASTRAN implementations 5.22.

[1] For a zero length bar, the first three constraints are modified to become $du_x = du_y = du_z = 0$.

5.21.3.1. Characteristic Length.

An element characteristic length is computed to allow scaling the equations. The distance between the reference point (subscript q) and a connected point (subscript i) is expressed by the components

$$\begin{aligned} L_{i,x} &= x_i - x_q &\quad (5.21.8)\\ L_{i,y} &= y_i - y_q &\quad (5.21.9)\\ L_{i,z} &= z_i - z_q &\quad (5.21.10)\\ L_i &= \sqrt{L_{i,x}^2 + L_{i,y}^2 + L_{i,z}^2} &\quad (5.21.11)\end{aligned}$$

The characteristic length of the element is the average of these lengths,

$$L_c = \sum_{i=1}^{N_c} |L_i|/N_c, \qquad (5.21.12)$$

where N_c is the number of connected points. If L_c is computed as a binary zero it is changed to a value of unity.

To ensure that the element is invariant to a change of scale, the weighting functions `w1` through `w6` provided by the user are modified to produce a connected grid point's weighting matrix.

$$W = \text{diag}(w_1, w_2, w_3, w_4 L_c^2, w_5 L_c^2, w_6 L_c^2).$$

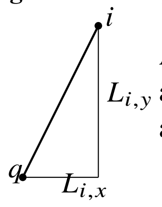
That is, the rotational DOF coefficients are scaled by the square of the characteristic length.

5.21.3.2. Equilibration.

Conventional equilibration equations are applied. These equations relate a force applied at the reference point to an equivalent force and moment applied at the reference node as illustrated in Figure 5-4. The loads at the connection point, i, relate to the loads at the reference point.

$$P_q = S_{iq}^T P_i, \quad S_{iq} = \begin{bmatrix} 1 & 0 & 0 & 0 & L_{i,z} & -L_{i,y} \\ & 1 & 0 & -L_{i,z} & 0 & L_x \\ & & 1 & L_{i,y} & -L_{i,x} & 0 \\ & & & 1 & 0 & 0 \\ & 0 & & & 1 & 0 \\ & & & & & 1 \end{bmatrix} \qquad (5.21.13)$$

Figure 5-4. – Equilibration of loads.



A force of $-\hat{e}_1$ at point i is equivalent to a force of $-\hat{e}_1$ and a moment of $\tau_z = L_{i,y}$ at point q.

5.21.3.3. Assembled Constraint.

As shown in Section 5.22 (equation 5.22.1), the loads on the set of all connection nodes may be computed from the load on the reference node. **S** is a concatenation of the individual S_{iq},

$$\mathbf{S} = \begin{bmatrix} S_{1,q} \\ S_{2,q} \\ \ldots \\ S_{N_c,q} \end{bmatrix}. \tag{5.21.14}$$

$$G_{qi} = A^{-1}\mathbf{S}'\mathbf{W}, \tag{5.21.15}$$

and

$$P_i = G'_{qi} P_q. \tag{5.21.16}$$

Similarly,

$$W = \text{diag}(W_1, W_2, \ldots, W_c),$$

and A is an order 6 weighting matrix.

$$A = \mathbf{S}^T W \mathbf{S} \tag{5.21.17}$$

We require that A be non-singular, which corresponds to a requirement that the RBE3 be non-mechanistic. The constraint relation follows directly from G_{qi}, i.e. define the 6 by $(6 + 6N_c)$ matrix,

$$\mathbf{C} = [\ -I_{qq} \quad G_{qi}\ ] \tag{5.21.18}$$

and apply the constraint,

$$\mathbf{C}\begin{bmatrix} u_q \\ u_i \end{bmatrix} = 0. \tag{5.21.19}$$

Each row of **C** contains the constraint coefficients for one of the six possible constraints in the RBE3.

5.22. MSC documentation of the NASTRAN RBE3 element

The documentation of the modern RBE3 element is provided by MSC Software from their web page.[104]

Solution#:	4494	Last Modified Date:	06/01/00 09:06:19 AM
Product Line:	MSC NASTRAN	Product Name:	MSC NASTRAN (1002 or 1004)
Product Version:		Product Feature:	
Article Type:	FAQ	Publish:	Y

The RBE3 element is a volume or surface spline element similar to the RSPLINE line spline element. The purpose of this memorandum is to develop a method for computing the terms in the equations of constraint generated by the element.

A sample Bulk Data Entry for the element is :

```
$       EID     [blank] REFGRID REFC    WT1     C1      G1,1    G1,2
RBE3    15              5       123456  1.0     123     10      20

$       G1,3    G1,4    WT2     C2 . .
,       30      40

$       UM      G1      C1      G2      C2      . . .
,       UM      10      123     20      23      30      3
```

The grid points 10 through 40, entered in the Gi,j fields on the entry, are connected to a reference grid point (number 5). The number of connected points, N_c, is unlimited. The physical principle used to generate the constraint equation coefficients is that the motion of a body connected to the reference grid point produces a weighted least-squares best fit to the actual motions at the other connected grid points. The reference point is connected by 1 through 6 DOFs (REFC specification). The connected points are also connected by 1 through 6 DOFs (Ci specification) with a weighting factor Wti. The UM data is optional, and is explained below.

The reference is the original design document for this element. Over the years some changes have been made in the interests of better theory and increased numerical robustness. Those changes are incorporated in this document as though this were the original design document, to avoid the awkwardness of first explaining older behaviors and then the present behavior. The original equations of the reference are derived with conventional variational principles applied to displacement variables. The derivation used here is based on force variable principles. This has proven to be more intuitive and better understood by some engineers. The results derived by the displacement method theory and force method theory are identical.

5.22.1. Generation of unit weighting functions

The element is designed to allow use of any coordinate system at any connected grid point, the global coordinate system in NASTRAN parlance. In the interests of clarity the equations are first developed for a system where all variables are defined in one common coordinate system (the basic coordinate system), then modified to allow global coordinates. An element characteristic length is computed to allow scaling the equations. The distance between the reference point (subscript q) and a connected point (subscript i) is expressed by the components

$$
\begin{aligned}
L_{i,x} &= x_i - x_q \\
L_{i,y} &= y_i - y_q \\
L_{i,z} &= z_i - z_q \\
L_i &= \sqrt{L_{i,x}^2 + L_{i,y}^2 + L_{i,z}^2}
\end{aligned}
$$

The characteristic length of the element is the average of these lengths, $L_c = \sum_{i=1}^{c} |L_i|/c$, where c is the number of connected points. If L_c is computed as a binary zero it is changed to a value of unity. User weighting functions w_i produce a dimensionless nodal weighting matrix.

$$\tilde{w}_i = w_i L_c^2, \quad W = \mathrm{diag}(w_1, w_2, w_3, \tilde{w}_4, \tilde{w}_5, \tilde{w}_6).$$

Conventional equilibrium equations are developed,

$$S_{iq} = \begin{bmatrix} 1 & 0 & 0 & 0 & z & -y \\ & 1 & 0 & -z & 0 & x \\ & & 1 & y & -x & 0 \\ & & & 1 & 0 & 0 \\ & 0 & & & 1 & 0 \\ & & & & & 1 \end{bmatrix}$$

This matrix expresses the loads that must be applied to the reference point to react loads applied at a connected point,

$$P_q = S_{iq}^T P_i$$

The equilibrium matrix can also be used to generate a loading pattern on the connected points due to a load on the reference point. Let Pq_{in} be a set of arbitrary loads on the reference point. When this load is applied, it is "beamed out" as loads on the connected points,

$$Pi = \begin{bmatrix} P_1 \\ P_2 \\ \dots \\ P_c \end{bmatrix} = \begin{bmatrix} W_1 & & & \\ & W_2 & & \\ & & \dots & \\ & & & W_c \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_c \end{bmatrix} XPq_{in} = WS_{iq}$$

X is a 6 by 6 matrix to be determined. The criterion used in its determination is that the load distribution mechanism should be in equilibrium. The equilibrium condition is that

$$Pq_{out} = \begin{bmatrix} S_1' & S_2' & \dots & S_c' \end{bmatrix} P_i = S_{iq}^T P_i$$

Then

$$Pq_{out} = S_{iq}^T W S_{iq} X Pq_{in}$$

$$G_{qi}^T = WSX \tag{5.22.1}$$

If $Pq_{out} = Pq_{in}$, then

$$X = (S_{iq}^T W S_{iq})^{-1}, \quad P_i = WSXPq = G_{qi}^T Pq$$

5.22.1.1. Transformation.

The direction cosine matrix T_i expresses the transformation between u_i, the values in basic coordinates, and $\tilde{u}_i$, the values in global coordinates:

$$u_i = T_i \tilde{u}_i$$

The transformed equilibrium equations and weighting matrices are

$$S_{iq} = \begin{bmatrix} T_1 S_1 \\ T_2 S_2 \\ \dots \\ T_c S_c \end{bmatrix}$$

The transformed weighting matrix in global coordinates is

$$W_i = T_i' W_i T_i$$

The transformed A matrix is

$$A_i = S_{iq}' W_i S_{iq}$$

$$A = \sum_i A_i$$

It is shown in the reference that the introduction of global coordinates modifies G_{qi} as shown:

$$G_{qi} = T_i A^{-1} [S_{iq}] W_i$$

This implies the dual relationship between displacements

$$u_q = G_{qi} u_i$$

Cast in the NASTRAN convention of constraint equations,

$$R_{qi} = [\ -I_{qq} \quad G_{qi}\ ]$$

and,

$$R_{qi} \begin{bmatrix} u_q \\ u_i \end{bmatrix} = 0.$$

R_{qi} is the rows of the matrix of MPC coefficients for one RBE3 element.

5.22.2. *Selection of dependent dofs (Optional)*

The default selection for dependent DOFs (m-set) are the REFC DOFs listed for the REFGRID. There are modeling applications where it is convenient to use these DOFs in a set exclusive from the dependent set, such as the analysis set (a-set). The dependent DOFs may be moved to the connected DOFs with the optional UM data. The number of DOFs must match the number of REFC DOFs, and the selected DOFs in the UM data must have non-zero weighting functions. If the subset of Rgi associated with these DOFs is named Rmm, the Rqi matrix is pre-multiplied by the inverse of this quantity,

$$R_{qi} = R_{mm}^{-1} R_{qi} = [-I_{mm} | R_{mm}^{-1} R_{mn}]$$

The user is required to select a UM set that produces an R_{mm} matrix that is stable for inversion. There are TANs that describe techniques for selection of a good set of UM variables. The uncoupling of the dependent equations allows some to be discarded, as described in the next section.

Equation selection. The total R_{qi} is generated above. It has 6 rows. Six or fewer rows are transmitted to the system constraint matrix R_{mg}, depending on the `REFC` data. This data consists of a packed integer with up to 6 numbers in the range of 1 to 6, and describes which rows are to be passed to R_{mg}. The remaining rows are discarded.

5.22.3. Features for dimension independence

A good finite element should produce the same results regardless of the units of measure used in the model. That is, the same structure modeled in millimeters, centimeters, or inches should provide identical results. The RBE3 gains this valuable characteristic by scaling the rotation weights with an element characteristic length,L_c, as described above. The effect of this scaling is demonstrated here by an example. In the interests of simplicity all geometry is in the basic coordinate system and the only non-zero offsets are in the z direction. The T matrix is then an identity matrix, and need not be listed in these equations. Consider the problem, defined by the S_{iq} matrix above and W_i matrices below, where

$$\begin{aligned} x &= x_i - x_q &&= 0, \\ y &= y_i - y_q &&= 0, \\ z &= z_i - z_q &&>< 0 \end{aligned}$$

The user inputs up to six weighting factors w1 through w6. The weighting factors for rotation are multiplied by $Lcsq = Lc^2$, the square of the characteristic lengths of the element. These modified terms are underlined in the matrix below, for example, $\tilde{w_4} = L_c^2 w_4$. The modified weighting factor matrix is then

$$W = \begin{bmatrix} w_1 & & & & & \\ & w_2 & & & & \\ & & w_3 & & & \\ & & & w_4 L_c^2 & & \\ & & & & w_5 L_c^2 & \\ & & & & & w_6 L_c^2 \end{bmatrix}$$

The contribution for grid point i to the equilibrium matrix A is

$$A = S'WS = \begin{bmatrix} w_1 & 0 & 0 & 0 & w_1 z & 0 \\ & w_2 & 0 & -w_2 z & 0 & 0 \\ & & w_3 & 0 & 0 & 0 \\ & & & L_c^2 w_4 + z^2 w_2 & 0 & 0 \\ Sym & & & & L_c^2 w_5 + z^2 w_1 & 0 \\ & & & & & L_c^2 w_6 \end{bmatrix}$$

The diagonal terms for rotation (for example A_{55}) have the form $L_c^2 w_i + z^2 w_j$, where w_i is the rotational weighting term, and w_j the translation term active in rotation weighting because of offsets. The motivation for modifying the rotation term can be seen in this addition of effects. Both L_c^2 and z^2 are in the same units of measure. When a model is changed from centimeters to millimeters, for example, the ratio of rotation effects to offset effects is unchanged. This modification of the rotation term allows the solution in the area of the RBE3 element to be the same for all units of measure. As z and L_c are related by a common factor the ratio of moment terms coming in directly from applied moments ($L_c^2 w_5$) stays in constant ratio to the moment terms from offsets ($z^2 w_1$) regardless of whether lengths are measured in centimeters, millimeters, or inches. This modification of the moment weight term provides dimension independence.

This example also provides an opportunity to discuss another counter-intuitive behavior of the RBE3 element, the difference between the user-supplied weighting functions and the actual values used in the corresponding coefficients of the constraint matrix. Let us simplify the expression of A above by setting $z_i = 0.0$. A becomes a diagonal matrix, which when inverted and multiplied by W to form G, becomes an identity matrix. The weighting factors are scaled to provide equilibrium. There may be little correlation between the values in the weighting matrix and the values in the coefficients of the constraint matrix. The requirements for equilibrium may change these values radically. Similarly, it shows that the significance of the weighting factors is in their ratio to one another. If all are multiplied by 10, for example, the inversion of the A matrix, used to impose equilibrium, removes this factor of 10 so that the coefficients of the constraint matrix are unchanged.

Stability issues. The solution requires the inverse of A. It may be ill-conditioned for linear equation solution. It is first equilibrated to make the inversion more stable. Let A_d be the diagonal terms of A. It is pre- and post-multiplied by the inverse of A_d,

$$A = A_d^{-1} A A_d^{-1}$$

This makes the diagonal terms of A unity. Any term multiplied by A is first multiplied by A_d. A matrix decomposition subroutine is used that provides an inverse conditioning number. As this number approaches zero the solution becomes more ill-conditioned. A belt-and-suspenders check that is less mathematical and more engineering-oriented is made by also computing the largest term in $[A^{-1}A - I]$, which should be a computational zero, and outputting this value when it passed a certain threshold. If the element is determined to be pathologically ill-conditioned it causes a user fatal error exit.

The RBE3 element is independent of the units of measure. For example, a structure modeled in centimeters will provide the same results when modeled in millimeters. An old formulation of the RBE3 element that did not provide dimension independence can be reproduced by setting the characteristic length L_c to one.

5.22.3.1. Theory

The modeler inputs a reference grid point, its connectivity, a weighting factor for other connected grid points, their connectivity, and the connected grid point ids. An RBE3 element used for testing this new capability of the form

```
$       EID     [blank] REFGRID REFC    WT      C       G1      G2
RBE3,   123,     ,      4       123456  1.0     123456  1       2
$       G3
,       3
```

The modeler's intent here is to connect grid point 4, for all 6 of its DOFs to the 1, 2, and 3 grid points, for their DOFs, with a uniform weighting factor for all. The element divides forces applied to point 4 to the other grid points in a manner that is influenced by their geometry and weighting factors, in a manner that maintains equilibrium. Define a line from the reference point to a connected point as an arm of the element. In the revised theory, a characteristic length, L_c of the element is calculated from the average length of its arms. The square of this length is used to modify the weighting of the connected rotation DOFs. The element is described and derived in TAN 4494. Some results of that derivation are used here. The constraint equation terms applied to a connected point u_i and the reference point u_q are

$$u_q = G_{qi} u_i$$

The constraint matrix itself has the following components:

$$G_{qi} = T_i A^{-1} S_{iq} W_i$$

T_i is a rotation matrix that is an identity matrix when GID$_i$ and GID$_q$ are in parallel coordinate systems. It will be dropped from this discussion. S_{iq} is the traditional matrix for transmitting rigid body motion between point “i” and point “q”. It has unit terms on the diagonal, and offset lengths on coupling terms between translation and rotation in the upper triangle. W_i is the user-supplied weighting functions, and A a matrix used to force the element to meet equilibrium requirements. All MSC NASTRAN constraint-type (R-) elements must meet an equilibrium condition, to avoid any possibility of internal constraints in the element. It is tedious and instructive to work out a simple example by hand, for a simple geometry. We will instead look at typical terms.

The A matrix is generated by finding the resultants of loads applied at the connected points, measured at the reference point. The 5,5 term for a single connected point is shown in the referenced TAN to be

$$A_{55} = w_5 + z_i^2 w_2.$$

When A is inverted, this term operates on the corresponding $S_{iq} w_i$ term

$$Giq_{55} = w_5 / (w_5 + z_i^2 w_1)$$

If z_i is zero, the effects of this normalization is to "wash out" the w_5 weighting term, so that the coefficient is 1.0. If z_i is not zero, the ratio of translation load effects $z_i^2 w_1$ to rotation loads effects w_5 is

$$Ratio = w_5 / (z_i^2 w_1)$$

This leads to a dimensional dependence, in that the ratio changes when the model is converted from millimeters to centimeters, for example. This undesirable behavior is eliminated by multiplying the rotation weighting factors by the square of the characteristic length, L_c,

$$Ratio = L_c^2 * w_5 / (z_i^2 w_1)$$

If z_i (and L_c) have their units of measure changed, the ratio stays constant. If this modified weighting constant is used on the 5,5 term

$$Giq_{55} = L_c^2 w_5 / (L_c^2 w_5 + z_i^2 w_1)$$

If $z_i = 0.0$ the weighting terms wash out. If it is non-zero the denominator of this quantity is constant with changes in units of measure.

Note that answers will change only when rotations are given connectivity for the connected DOFs, and then only when the rotations at the connected DOFs are part of a redundant load path. This is because the element is required to meet equilibrium conditions to avoid internal constraints, that is, single point constraints that do not appear in the SPCFORCE output. If the load path is statically determinate the equations used to impose equilibrium will adjust the values of internal loads in the element as needed to meet equilibrium, regardless of the value of the weighting functions. Always meeting equilibrium requirements ensures that there will be no internal SPC forces in the element.

5.23. Interpolation within an Element

It can be useful to sample a field within an element. This is necessary for verification of the input for temperature fields applied at integration points, as in a X-ray deposition. If the fields are known at a variety of points inside an element, we can use that information to determine the fields at an arbitrary location. In the case of infinite elements, the fields "interior" to the element project to the entire space beyond the element surface. Several means may be used to perform this interpolation. In **Sierra/SD** we use a least squares projection onto a Pascal space, and then apply the Pascal shape functions to generate the interpolated function. The least squares solution requires that there be more sample points than there are shape functions.

As an example, consider temperatures applied at the Gauss integration points of a Hex20. The coordinates of the 27 integration points are defined in Table 5-2. For a quadratic fit of the data, we can complete the Pascal triangle to obtain the shape functions listed in Table 5-7. We generate a shape matrix, A, for which each entry in the matrix is given as follows.

$$A_{ij} = P_j(\xi_i)$$

Here, ξ_i is the element coordinate of the i^{th} integration point.

index	**Function, P_i**
1	1
2	η_1
3	η_2
4	η_3
5	η_1^2
6	$\eta_1\eta_2$
7	$\eta_1\eta_3$
8	η_2^2
9	$\eta_2\eta_3$
10	η_3^2

Table 5-7. – Pascal Shape functions for 3D elements of order 2.

The coefficients of the Pascal shape functions, b, are given by the solution to the least squares minimization problem.

$$minimize||x - Ab||$$

where x is the vector of known temperature values at the 27 integration points in the element, A is the shape matrix defined above and b the vector of coefficients to determine. This problem is solved using the `LAPACK` function `dgels` in **Sierra/SD**.

Once the coefficient vector is known, the solution at any location within the element may be determined by expansion of the shape functions at the location of interest.

$$T(\eta_1, \eta_2, \eta_3) = \sum_i b_i P_i(\eta_1, \eta_2, \eta_3)$$

where P_i are the shape functions of Table 5-7.

6. BOUNDARY CONDITIONS AND INITIAL CONDITIONS

6.1. Acoustic and Structural Acoustic

In this section, we describe the various boundary conditions available in **Sierra/SD** for acoustics and structural-acoustics. In each case we discuss the governing equations and discretization approaches.

6.1.1. Absorbing Boundaries

The need to truncate acoustic domains arises in exterior problems, where the fluid or solid domain is infinite or semi-infinite. In these cases, the domain could be truncated either with infinite elements, or absorbing boundary conditions. We describe below the simple absorbing boundary conditions that have been implemented in **Sierra/SD**. Infinite elements (see section (6.1.2)) are also implemented in **Sierra/SD**. We describe the cases of an acoustic space and an elastic space separately.

6.1.1.1. Acoustic Space

The implementation of absorbing boundary conditions begins by considering the weak formulation of the equations of motion, in equations (3.2.6). On an absorbing boundary, one needs to consider the term

$$\int_{\partial\Omega_n} \frac{\partial\psi}{\partial n}\phi ds, \tag{6.1.1}$$

which arises from the integration by parts on the acoustic space. Absorbing boundary conditions are typically derived by applying impedance matching conditions to equation (6.1.1), in such a way that the boundary completely absorbs waves of a given form. For example, the simplest absorbing boundary conditions consist of plane wave and spherical wave conditions,[57] which are either the zero-th order accurate Sommerfeld condition

$$\frac{\partial\psi}{\partial n} = \frac{-1}{c_f}\frac{\partial\psi}{\partial t} \tag{6.1.2}$$

or the first order accurate Bayliss-Turkel condition

$$\frac{\partial\psi}{\partial n} = \frac{-1}{c_f}\frac{\partial\psi}{\partial t} - \frac{1}{R}\psi \tag{6.1.3}$$

where R is the radius of the absorbing spherical boundary.

Inserting equation (6.1.2) into equation (6.1.1), we obtain a term proportional to $\dot{\psi}$, which becomes a damping matrix. Inserting equation (6.1.3) into equation (6.1.1), we obtain two matrix terms, one that contributes to the damping matrix, and another that contributes to the stiffness matrix. Note that in the limit of large R, the spherical wave condition reduces to the plane wave condition, since for large enough radius, the spherical wave begins to resemble a plane wave.

Both conditions (6.1.2) and (6.1.3) are implemented in **Sierra/SD**.

6.1.1.2. Elastic Space

In the case of an elastic space, similar absorbing boundary conditions can be applied as were in the acoustic space, except the boundary has to absorb both pressure and shear waves. In the case of an acoustic medium, only pressure waves are of interest. Thus, the elastic space is more complicated.

The equation of motion for an elastic space can be written as

$$\rho u_{tt} - \nabla \cdot \sigma = f \tag{6.1.4}$$

where ρ is the material density, u_{tt} is the second time derivative of displacement, σ is the stress, and f is the forcing. A weak formulation of this equation can be constructed by multiplying with a test function and integrating by parts.

$$\int_V \rho u_{tt} w dV + \int_V \sigma : \nabla w dV - \int_{\partial V} \sigma_s w dS = \int_V f \cdot w dV \tag{6.1.5}$$

where w is the test function, and σ_s is the traction vector on ∂V, the boundary of volume V. The absorbing boundary condition is imposed on the portions of ∂V that point into the infinite space. In this derivation, we assume that this includes the entire boundary ∂V. If only part of the boundary pointed into the infinite space, the derivation would be the same.

Considering the term

$$\int_{\partial V} \sigma_s w dS \tag{6.1.6}$$

we note that the traction vector σ_s can be decomposed into its normal and tangential components, i.e. $\sigma_s = \sigma_n + \sigma_t$. Then, we apply the conditions

$$\begin{aligned} \sigma_n &= -\rho c_L v_n \\ \sigma_t &= -\rho c_T v_t \end{aligned} \tag{6.1.7}$$

where c_L and c_T are the longitudinal and shear wave speeds in the medium, and v_n, v_t are the normal and tangential components of velocity vectors on the surface. Inserting these relations into equation (6.1.6) yields two absorbing boundary matrices. Since these matrices involve the velocities, they become part of the overall damping matrix of the structure.

6.1.2. *Infinite Elements for Acoustics*

Infinite elements have been around since the mid 1970's. Excellent review articles can be found in,[9].[69]

In the early formulations, only frequency-domain formulations were considered, and system matrices were developed that depended on frequency in a nonlinear manner. Though these formulations worked in the frequency domain, there was no clear approach for transforming them to the time domain. As a result, time domain formulations for infinite elements were delayed for some time. The formulations[9,35] in the time domain formulation would involve convolution integrals that could be used with the frequency-dependent system matrices. However, storing the time histories for the convolution integrals would be a significant burden for a time-domain code.

In the early 1990's, Astley[10,15,12] derived a conjugated formulation that resulted in system matrices that were independent of frequency. This allowed the frequency domain formulation to be readily transformed to the time domain, in the same way that is typically done in linear structural dynamics. He also derived a scheme for post-processing the infinite element degrees of freedom to compute the far-field response at points outside of the acoustic mesh. This approach followed from a time-shift applied to the infinite element degrees of freedom.

The exterior acoustic problem consists of finding a solution p, outside of some bounded region Ω_i. We refer to Figure (6-1) for a description of the geometry. We have an interior domain Ω_i, and an exterior domain Ω_e, and a boundary Γ that separates the inner and outer domains. We wish to find the acoustic pressure p in Ω_e. In the exterior domain Ω_e, the acoustic pressure must satisfy the acoustic wave equation

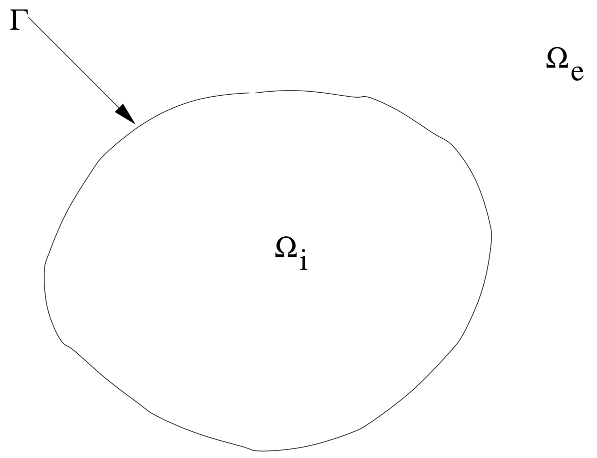


Figure 6-1. – Domains Ω_i and Ω_e and interface Γ for the exterior acoustic problem.

$$\frac{1}{c^2}\ddot{p} - \Delta p = 0 \tag{6.1.8}$$

a Neumann boundary condition on Γ

$$\frac{\partial p}{\partial n} = g(x,t) \tag{6.1.9}$$

and the Sommerfeld radiation condition at infinity

$$\frac{\partial p}{\partial r} + \frac{1}{c}\frac{\partial p}{\partial t} \longrightarrow \frac{1}{r} \tag{6.1.10}$$

as $r \longrightarrow \infty$.

We note that the weight and test functions are chosen such that the Sommerfeld condition is satisfied identically. Then, the weak formulation reads as follows

$$\int_{\Omega_e} \frac{1}{c^2}\ddot{p}q + \nabla p \cdot \nabla q dV = \int_\Gamma gqdS \tag{6.1.11}$$

In the frequency domain, the counterpart to equation (6.1.11) is as follows

$$-k^2 \int_{\Omega_e} pqdV + \int_{\Omega_e} \nabla p \cdot \nabla q dV = \int_\Gamma gqdS \tag{6.1.12}$$

where $k = \frac{\omega}{c}$.

We will focus on conjugated infinite element formulations, which implies specific choices for the trial and weight functions for the infinite elements. For the trial functions, we have

$$\phi_j(x,\omega) = P_j(x)e^{-ik\mu(x)} \tag{6.1.13}$$

and for the weight functions, we have

$$\psi_j(x,\omega) = D(x)P(x)e^{ik\mu(x)} \tag{6.1.14}$$

where $P(x)$, $D(x)$, and $\mu(x)$ are as yet undefined functions of x, and $k = \frac{\omega}{c}$ is the wavenumber. The choice of these functions will determine the particular infinite element approach. In our case, the exponential in the weight functions involves a conjugate of the exponential in the trial functions. This results in the exponential canceling out in the system matrices, thus rendering the matrices independent of frequency.

Given these trial functions, the solution $p(x,\omega)$ can be written in an expansion

$$p(x,\omega) = \sum_{i=1}^{N} q_j(x,\omega)\phi_j(x,\omega) \tag{6.1.15}$$

Substituting these expressions for trial and weight functions into equation (6.1.12), we obtain for following expression

$$\int_{\Omega_e} (P_i\nabla D + D\nabla P_i + ikDP_i\nabla\mu)\cdot\left(\nabla P_j - ikP_j\nabla\mu\right)q_i - k^2DP_iP_jq_i dV \tag{6.1.16}$$

Separating out terms of ω, we obtain the following expressions for the stiffness, mass and damping matrices

$$K_{ij} = \int_{\Omega_e} (P_i\nabla D + D\nabla P_i)\cdot\nabla P_j dV \tag{6.1.17}$$

$$C_{ij} = \frac{1}{c}\int_{\Omega_e} DP_i\nabla\mu\cdot\nabla P_j - P_iP_j\nabla D\cdot\nabla\mu - DP_j\nabla P_i\cdot\nabla\mu dV \tag{6.1.18}$$

$$M_{ij} = \frac{1}{c^2}\int_{\Omega_e} DP_iP_j(1-\nabla\mu\cdot\nabla\mu)dV \tag{6.1.19}$$

Consider the phase function $\mu(x)$. First, we note that the series expansions for the trial functions (the i^{th} term is given by equation (6.1.13)), assume an outwardly propagating wave. The exact solution from which these trial functions are derived involves a source point for the wave. We denote the distance from that source point to a point on the base surface by a. The phase function is then defined by

$$\mu(x) = r - a \tag{6.1.20}$$

In spherical coordinates, the gradient of a function is equal to

$$\nabla f(r,\theta,\phi) = \hat{r}\frac{\partial f}{\partial r} + \frac{1}{r}\frac{\partial f}{\partial \phi}\hat{\phi} + \frac{1}{rsin(\phi)}\frac{\partial f}{\partial \theta}\hat{\theta} \tag{6.1.21}$$

Since the expression for $\mu(x)$ depends only on r, we have

$$\nabla\mu(x) = \hat{r} \tag{6.1.22}$$

Thus, $\nabla\mu(x)\cdot\nabla\mu(x) = 1$. This implies that when the boundary defining the infinite elements is a spherical surface, the mass matrix from equation (6.1.19) is identically zero. This makes sense, since it ensures that the modes are outgoing, and that there are no standing waves. Since a numerical integration of equation (6.1.19) will never come out identically zero, the question then becomes whether to include this numerical mass in the time integration, or whether to neglect it from the outset. This has important implications in the stability of the time integration, as outlined in.[13]

In terms of discretizing the infinite domain, infinite elements can be classified into 2 main approaches: the separable approach, and the mapped approach. In the separable approach, the exterior domain is assumed to be in a separable coordinate system, such as spherical or spheroidal. In the mapped approach, the nodes on the exterior boundary are mapped into parent elements using a special mapping functions that map the infinite domain into a finite reference element domain. The mapped approach is advantageous because it allows a more arbitrary placement of nodes on the exterior surface. The separable approach requires the exterior nodes to conform to a specific boundary, and thus this approach places more restrictions on the mesh generation process.

6.1.2.1. Infinite Element Shape Functions

In our work, we have chosen the mapped approach due to its flexibility in mesh generation. The integrals in equations (6.1.17), (6.1.19), and (6.1.18) are over an infinite domain, Ω_e. To perform numerical integration of these integrals, we first must map onto a unit reference element, as in standard finite elements. The mapping is as follows

$$x = \sum_{j=1}^{N} M_j(s,t,v)x_j \tag{6.1.23}$$

where x is a point in the infinite domain, x_j are the coordinates of the mapping points, s, t define the base coordinates of the *base* plane of the infinite element (which lies on the exterior surface of the acoustic mesh), and v is the base coordinate in the infinite direction. If we consider a point on the exterior surface, and its radial point a_i, then the base coordinate along the radial edge emanating from this point is given by,

$$v_i = 1 - 2a_i/r_i \tag{6.1.24}$$

Equivalently,

$$r_i - a_i = a_i \frac{1+v_i}{1-v_i} \tag{6.1.25}$$

Where r_i is a radial distance from a virtual source point (or virtual origin). Each node on the infinite element boundary may have a source point, as illustrated in Figure (6-2). Generally, the source point is positioned to ensure that rays are normal to the surface.[10,14] The mapping ensures that as the element coordinate v approaches 1, the physical radial coordinate, r approaches infinity; thus mapping an infinite space onto a unit element.

The virtual source point can provide an orthogonal basis in the radial direction. For non-spherical meshes, one virtual source point is needed for each point on the infinite element boundary to ensure that the radial expansions are normal to the surface and orthogonal to the surface shape functions, $S_i(s,t)$. This permit writing the mapping function as a product of spatially separated terms, $M_i(s,t,v) = S_i(s,t)R_i(v)$. This orthogonality is also necessary to ensure that the mass matrix remains positive semi-definite. The mass matrix (from equation (6.1.19)) includes the term $1 - \nabla\mu \cdot \nabla\mu$. The magnitude of the gradient term, $\nabla\mu$, is 1.0 when the source is normal to the surface. It is greater than one otherwise, which leads to an indefinite matrix, and can produce instability in dynamic integration.

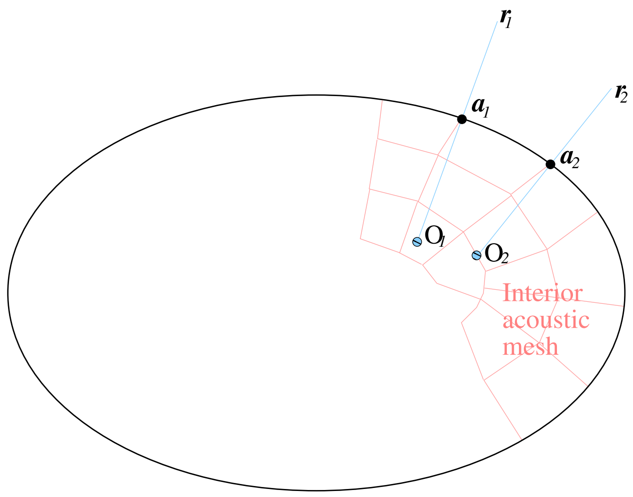


Figure 6-2. – Infinite Element Radial Mapping. Each node on the infinite element boundary may have an origin, O_i, (called a virtual source point) and an effective nominal radius, a_i. The source point is chosen to ensure that rays are normal to the surface. For a spherical boundary, all virtual source points are at the center of the sphere.

In **Sierra/SD** two methods are used to generate the source point location. The first travels the normal vector a fixed distance b, where b is the dimension of the minor axis. The second method provides an offset that intersect a plane normal to the vector and passing through the origin of the ellipsoid. These two methods are illustrated in Figure (6-3).

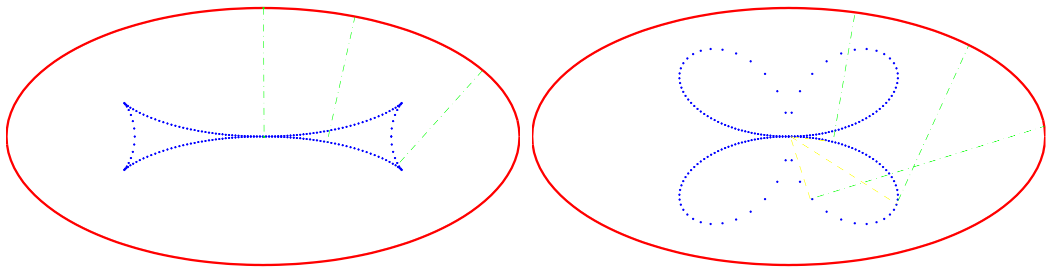


Figure 6-3. – Methods of Locating Source Point. On the left, the source point is located on the surface normal, a distance b into the structure, where b is the minor axis dimension. On the right, the source point is located along the surface normal such that it intersects a plane normal to the vector, and containing the ellipsoid centroid.

The radial point a is interpolated over the infinite element base, to give

$$a(s,t) = \sum_{i=1}^{N} a_i S_i(s,t) \tag{6.1.26}$$

where $S_i(s,t)$ is the implied surface shape function of the base element on the exterior surface. In this way, tetrahedrons or hexahedrons may be used in the acoustic mesh. For the infinite elements, the only difference is the surface shape functions $S_i(s,t)$. The radial interpolation is independent of the underlying finite element. The mapping functions $M_j(s,t,v)$ given in equation (6.1.23) are constructed as tensor products of the surface shape functions $S_i(s,t)$ and radial basis mapping functions. The radial basis mapping functions are typically defined to be linear functions that map the finite domain into the infinite domain. These functions are given as,

$$\begin{aligned} m_1(v) &= \frac{2v}{v-1} \\ m_2(v) &= \frac{1+v}{1-v} \end{aligned} \tag{6.1.27}$$

Thus, when $v = -1$, we have that $m_1(v) = 1$ and $m_2(v) = 0$. When $v = 1$, we have $m_1(v) = -\infty$ and $m_2(v) = \infty$. In this way, the infinite domain is mapped to a finite domain.

The mapping functions $M_j(s,t,v)$ are defined as tensor products of the surface shape functions $S_i(s,t)$ with the radial mapping functions from equation (6.1.27). For example, for an 8-node hex, the surface shape functions are defined as,

$$\begin{aligned} S_1(s,t) &= \frac{(1+s)(1+t)}{4} \\ S_2(s,t) &= \frac{(1+s)(1-t)}{4} \\ S_3(s,t) &= \frac{(1-s)(1+t)}{4} \\ S_4(s,t) &= \frac{(1-s)(1-t)}{4} \end{aligned} \tag{6.1.28}$$

Then, the 8 functions $M_i(s,t,v)$ can be constructed by crossing each $S_i(s,t)$ from equation (6.1.28) with an $m_j(v)$ from equation (6.1.27).

Equation (6.1.25) can then be used to compute the phase function $\mu(x)$ at an arbitrary point

$$\mu(x) = r - a = \sum_{i=1}^{N}(r - a_i)S_i(s,t) = \sum_{i=1}^{N} a_i S_i(s,t)\frac{1+v}{1-v} = a(s,t)\frac{1+v}{1-v} \tag{6.1.29}$$

With $\mu(x)$ defined, we consider $P(x)$. The l^{th} shape function $P(x)$ is defined as

$$P_l(x) = \frac{1}{2}S_i(s,t)(1-v)Q_j(v) \tag{6.1.30}$$

where $Q_j(v)$ is a polynomial in a single variable. Various choices of $Q_j(x)$ have been investigated, including Lagrangian,[10,15] Legendre,[11] Jacobi,[51] and rational (integrated Jacobi).[44] Lagrangian shape functions result in ill conditioned infinite element matrices. The other three choices all appear to give acceptable levels of conditioning. Dreyer[51] showed that the Jacobi polynomials in general give a better condition than the Legendre polynomials. Regardless of the choice for $Q(x)$, equations (6.1.23) and (6.1.30) imply that $P(x)$ will be a function of the reference element coordinates r, s, t, and thus can be integrated over the reference element.

The function $D(x)$ is defined as

$$D(x) = \left(\frac{1-v}{2}\right)^2 \tag{6.1.31}$$

We have defined $P(x)$, $\mu(x)$, and $D(x)$, in terms of the reference element coordinates r, s, t. The integrals in equations (6.1.17), (6.1.18), and (6.1.19) can all be evaluated by standard Gaussian quadrature over the reference unit element (either hex or tet).

6.1.3. Computation of solution at far-field points

After the solution to the acoustic problem is complete, the values of the coefficients in the expansion of equation (6.1.15) are known. The next step is then to compute the solution at far-field points outside of the acoustic mesh. We consider two cases below, one where the polynomial functions $P(x)$ in equation (6.1.13) is a Lagrangian shape function, and the other where $P(x)$ is a more general polynomial (like a Legendre or Jacobi polynomial). In the former case, the functions $P(x)$ are associated with particular nodes having values of 1 at the node and 0 at the other nodes. In the latter case, this property does not hold.

We assume that we wish to compute the solution at a node d that is at a location x_d, and a radial distance $r = ||x_d||$ from the origin. This point is located on a radial line with a corresponding radial point a. Thus, for this point we have $\mu_d = r - a$., We have

$$p(x_d, \omega) = \sum_{i=1}^{N} q_j(\omega) P_j(x_d) e^{-ik\mu_d} \tag{6.1.32}$$

Note that 'N' in this case is the number of infinite element basis functions within the infinite element that includes the point d. In the case of Lagrangian polynomials, we have the property that the function is equal to 1 at the node of interest and is equal to 0 at the other nodes. Thus, in the case that the point x_d coincides with a node in the infinite element, we have the expression

$$p(x_d, \omega) = q_d(\omega) e^{-ik\mu_d} \tag{6.1.33}$$

where $q_d(\omega)$ is the infinite element shape function corresponding to node d. Equivalently, we have

$$q_d(\omega) = p(x_d, \omega) e^{ik\mu_d} \tag{6.1.34}$$

Thus, the pressure at the node d is equal to the corresponding value of the coefficient of the infinite element expansion corresponding to that node, multiplied by the factor $e^{-ik\mu_d}$, where μ_d is equal to the distance (along the radial line) from the boundary of the acoustic domain to the node d.

If we take the inverse Fourier transform of equation (6.1.34), we get

$$q_d(t) = p(x_d, t + \frac{d}{c}) \tag{6.1.35}$$

Thus, the pressure time history at node d is equal to a time-shifted value of the infinite element degree of freedom $q_d(t)$ corresponding to node d. This makes physical sense in that it would take the wave additional time equal to $\frac{d}{c}$ to reach the point d.

Next we consider the case when $P(x)$ is not a Lagrangian polynomial. In this case, the point d could not be associated with any particular node. In this case, we still have the relation

$$p(x_d, \omega) = \sum_{i=1}^{N} q_j(\omega) P_j(x_d) e^{-ik\mu_d} \tag{6.1.36}$$

except in this case, the polynomials $P(x)$ do not necessarily vanish at d. Thus, again bringing the exponential to the other side of the equation, we have

$$p(x_d, \omega) e^{ik\mu_d} = \sum_{i=1}^{N} q_j(\omega) P_j(x_d) \tag{6.1.37}$$

Taking inverse Fourier transforms, we arrive at the result

$$p(x_d, t + \frac{d}{c}) = \sum_{i=1}^{N} q_j(t) P_j(x_d) \tag{6.1.38}$$

Since all quantities on the right-hand side of equation (6.1.38) are known after the finite/infinite element solution is complete, we can post-process to compute the pressure at the field point x_d.

6.1.4. Point sources

Point acoustic sources are common in acoustic modeling, and we provide some capability for doing this in **Sierra/SD**. Here we describe the theory behind this implementation. The theory of point sources[86,110] in acoustics is typically formulated by considering a pulsating sphere of radius R, centered at the point x_s. Upon taking the limit as the radius of the sphere goes to zero, one obtains the equation for an acoustic point source.

We consider a point source that is injecting mass into the acoustic domain at a rate

$$\dot{m}_s(t) = \rho Q_s(t) \tag{6.1.39}$$

where $\dot{m}_s$ is the mass per unit time of fluid that is being injected into the domain, ρ is the density of the fluid, and $Q_s(t)$ is the volume velocity (volume per unit time) of the fluid that is entering the acoustic domain. More on this will be given later in Section 6.3 on Lighthill's approach, and its connection with the point source. We can construct a point source consistent with the mass injection rate q defined in equation (3.1.1) via multiplication of $\dot{m}_s$ by a Dirac delta function (which itself has units of one over volume). Because $\partial q/\partial t$ appears in the wave equation (3.1.11), one more time derivative of $\dot{m}_s$ is required:[110]

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} - \nabla^2 p = -\ddot{m}_s(t)\delta(x - x_s), \tag{6.1.40}$$

where p is the acoustic pressure at a point in the domain, c is the speed of sound, and ρ is the fluid density. We note that the volume velocity can also be written as the time derivative of the volume in the source

$$Q_s(t) = \frac{dV}{dt} \tag{6.1.41}$$

where V is the volume enclosed by the source. Equation (6.1.41) is valid for a spherical source enclosing a volume V, but in the case of a point source we shrink the radius to zero. The volume velocity, Q_s, is also sometimes called the *source strength*. It is the integral of the normal component of surface velocity over the spherical surface of the source. Since the surface velocity is the same everywhere on the surface of the sphere, the source strength is

$$Q_s = \int_S v_n dS = v_n \int_s dS = 4\pi a^2 v_n \tag{6.1.42}$$

where a is the radius of the sphere, and v_n is the normal component of velocity on the surface. By considering the volume increase for a pulsating sphere, it is easy to see that equations (6.1.41) and (6.1.42) are the same.

We note that in the **Sierra/SD** implementation of acoustics, we use the time derivative of pressure rather than the pressure directly. We also scale the equation by density, since this is needed when the fluid properties are not constant. Thus, we would modify equation (6.1.40) as follows

$$\frac{1}{\rho c^2}\frac{\partial^2 \psi}{\partial t^2} - \frac{\nabla^2 \psi}{\rho} = -\frac{\dot{m}_s(t)}{\rho}\delta(x - x_s), \tag{6.1.43}$$

where $p = \partial\psi/\partial t$. Equivalently, this gives

$$\frac{1}{\rho c^2}\frac{\partial^2\psi}{\partial t^2} - \frac{\nabla^2\psi}{\rho} = -Q_s(t)\delta(x - x_s) \tag{6.1.44}$$

In the frequency domain, equation (6.1.40) is typically written as

$$\left(\nabla^2 + k^2\right)\phi = -4\pi A\delta(x - x_s) \tag{6.1.45}$$

where A is called the *amplitude* of the source. The solution to equation (6.1.45) in an unbounded domain can be shown to be

$$\phi = \frac{A}{r}e^{jkr} \tag{6.1.46}$$

where $r = |x - x_s|$ is the distance from the source to the point x in the domain, and $k = \frac{\omega}{c}$ is the wavenumber. Assuming a time-harmonic expression for $Q_s(t) = Qe^{j\omega t}$, it follows from equation (6.1.44) that Q and A are related by

$$Q = \frac{4\pi A}{\rho}. \tag{6.1.47}$$

The solution ϕ can therefore be expressed as

$$\phi = \rho Q\frac{e^{jkr}}{4\pi r} \tag{6.1.48}$$

or due to $\psi = \partial p/\partial t$, as

$$p = j\omega\rho Q\frac{e^{jkr}}{4\pi r}. \tag{6.1.49}$$

Specification of dV/dt in equation (6.1.44) and d^2V/dt^2 in equation (6.1.40) is covered in *User's Manual*.

A finite element formulation of the previous equation can be constructed as usual, by multiplying the previous equation by a test function, and integrating by parts. We note that the domain of integration must include the point x_s, the location of the point source. Also, we note that the integration against the delta function $\delta(x - x_s)$ is a duality pairing, rather than an integral, since the integral of a delta function is not defined. In what follows, we assume that the point x_s lies on a node in the finite element mesh. This will facilitate the modeling, since we will typically define the point source on a nodeset or nodelist consisting of a single node.

Denoting by $V_f(\Omega_f)$ the function space for the fluid, the weak formulation can be written as follows. Find the mapping $\psi : [0, T] \to V_f(\Omega_f)$ such that

$$\int_\Omega \frac{\ddot{\psi}}{\rho c^2}\phi dx + \int_\Omega \frac{\nabla\psi \cdot \nabla\phi}{\rho}dx = -\int_{\partial\Omega_n} \dot{u}_n\phi ds + Q_s(t)$$

$\forall\phi \in V_f(\Omega_f)$, where $\dot{u}_n$ is the prescribed velocity on the Neumann portion of the fluid boundary. We note that the first term on the right-hand side is a surface excitation force, and thus only contributes nonzero terms on nodes that lie on the surface $\int_{\partial\Omega_n}$. The second term comes from the point source, and only contributes a nonzero term on the node where the point source is located.

Inserting a finite element discretization $\phi(x) = \sum_{i=1}^N \phi_i N_i(x)$ into equation (6.1.50) results in the system of equations

$$M\ddot{\psi} + K\psi = f_a, \tag{6.1.50}$$

where N is the vector of shape functions, $M = \int_{\Omega_f} \frac{1}{\rho c^2} N N^T dx$ is the mass matrix, $K = \int_{\Omega_f} \frac{\nabla N \cdot \nabla N^T}{\rho} dx$ is the stiffness matrix, and $f_a = \int_{\partial\Omega_n} \dot{u}_n N^T dx + Q_s(t)$ is the external forcing vector from Neumann boundary conditions.

If $Q = \frac{dV}{dt}$ is computed with a void element in Presto, equation (6.1.50) can be used to compute the right-hand side term and the corresponding acoustic response.

6.1.5. *Perfectly Matched Layers*

The perfectly matched layers are described in detail in Bunting et al.[29] Given a structure S surrounded by bounded interior domain Ω_i, and an exterior domain Ω_e, the exterior acoustics problem consists of determining the acoustic pressure, p, in domain $\Omega_e \cup \Omega_i$. We refer to Figure 6-4 for a schematic of the geometry. In a domain truncation strategy, boundary conditions are applied to the outermost boundary Γ_e of Ω_i.

To illustrate the ideas, we assume an acoustic pressure wave propagating in the x-direction, with wavenumber $k = \frac{\omega}{c}$, where ω is the circular frequency, and c is the speed of sound. The wave takes the form

$$p(x) = p_0 e^{ikx} \tag{6.1.51}$$

As written, this wave is undamped, and will propagate indefinitely with no change of shape. However, if we allow the wave to propagate on a coordinate system that has *complex* coordinates $\tilde{x} = a(x) + ib(x)$, where $a(x)$ and $b(x)$ are functions of x, then the equation of the wave becomes[81]

$$p(\tilde{x}) = p_0 e^{ik\tilde{x}} = p_0 e^{i(ka(x)+ikb(x))} = p_0 e^{-kb(x)} e^{ika(x)} \tag{6.1.52}$$

We observe that this wave corresponds to *damped* wave propagation, with decay coefficient equal to $kb(x)$. For a coordinate stretching of $b(x) > 0$, this wave will decay exponentially fast, which is the case considered in this paper. If $b(x) < 0$, then the wave will grow exponentially fast.

In order for equation (6.1.52) to be a solution to a wave equation, that wave equation must itself be written in a coordinate system that is complex, rather than real-valued. On the other hand, the corresponding finite element implementation is most easily derived on a real-valued coordinate system. Thus, though the governing partial differential equations of the PML are written in a complex coordinate field, the corresponding weak formulation is mapped to a real coordinate system, to facilitate the finite element implementation.

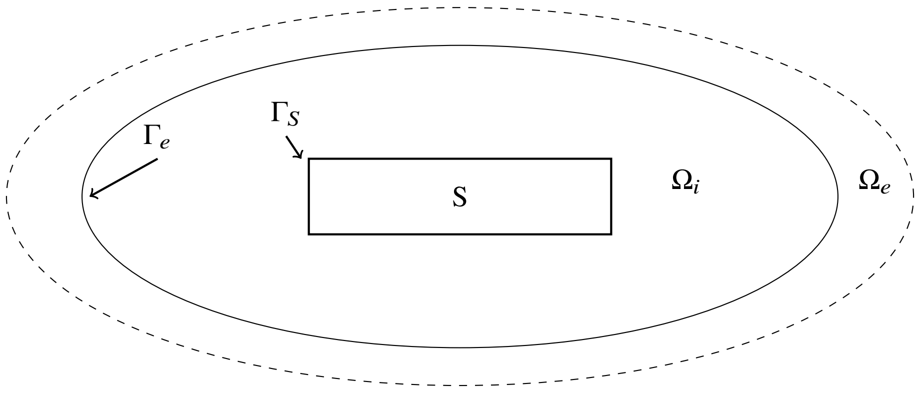


Figure 6-4. – Domains Ω_i and Ω_e and interface Γ for the exterior acoustic problem.

To build up to the ellipsoidal PML formulation, the following sections provide derivations of rectangular, rotated rectangular, and spherical PML. These provide the building blocks for the ellipsoidal case. We will subsequently show that the ellipsoidal formulation reduces to the spherical and rectangular cases by choosing equal and large radii of curvature, respectively.

6.1.5.1. Cartesian PML

We define the PML domain as being a parallelepiped of dimension $(2\bar{a}, 2\bar{b}, 2\bar{c})$, centered at the origin, with an interior parallelepiped hole of dimension $(2a, 2b, 2c)$. Practically, this would correspond to the case where the structure of interest, as complex shape it may have, was surrounded by an acoustic mesh that terminated at the boundary of the inner parallelepiped. The PML would then occupy the region between the inner and outer parallelepiped boundaries. A simple shift can be applied if the domain is not origin-centered.

The PML formulation can be broken down into three steps. First the analytic continuation is used to map the Helmholtz equation into the complex plane. Then the weak form is formulated on the complex plane, and the chain rule is applied to map between the complex and real plane. Finally, the results from the chain rule give a weak formulation over the real-valued domain, but with the dissipative properties stemming from the transformation to complex coordinates.

6.1.5.1.1. Step 1. Analytic continuation The PML equations can be written in either first or second order form. Here we consider the implementation of second order form. In the interior $\Omega = \Omega_I$, the acoustic pressure must satisfy the acoustic Helmholtz equation

$$-\Delta p - k^2 p = 0 \tag{6.1.53}$$

where $k = \frac{\omega}{c}$, and p is the acoustic pressure, a prescribed Neumann boundary condition on Γ_S

$$\frac{\partial p}{\partial n} = g(x, \omega) \tag{6.1.54}$$

and the Sommerfeld radiation condition for outgoing waves at infinity[101]

$$\left| \frac{\partial p}{\partial r} - ikp \right| = O\left(\frac{1}{r^2}\right), \qquad r \to \infty \tag{6.1.55}$$

where $k = \frac{\omega}{c}$. We note that equation (6.1.53) involves constant coefficients, meaning that the speed of sound and density in the fluid are assumed to be constant. More specifically, equation (6.1.53) is undamped, meaning that the waves will not attenuate as they propagate through the medium.

Equation (6.1.53) is written in terms of real coordinates. As illustrated earlier, the waves will decay in the PML if the coordinates are considered as complex-valued rather than real-valued. Thus, we use *analytic continuation* to map the Helmholtz equation into the complex plane

$$\tilde{\Delta} p - k^2 p = 0 \tag{6.1.56}$$

where the change of coordinates for the x-direction is defined as:

$$\tilde{x} = x - \frac{i}{\omega} \int_x^a \sigma(\xi) d\xi \quad a < x < \bar{a} \tag{6.1.57}$$

$$\tilde{x} = x + \frac{i}{\omega} \int_a^x \sigma(\xi) d\xi \quad - \bar{a} < x < -a \tag{6.1.58}$$

Similar expressions describe the coordinate transformations for the other two coordinate axes.

6.1.5.1.2. Step 2. Weak formulation over complex-valued domain We note that the weak formulation of equation (6.1.56) can be constructed using either a *bilinear* or *sesquilinear* formulation.[43,45] The difference is only whether complex conjugation is applied to the test functions. In standard finite element methods for acoustics, these formulations lead to the same discrete system of equations. However, with PML the formulations yield different numerical methods. In this paper we take the bilinear approach, since it yields a complex-symmetric system of linear equations that can be exploited in the linear solver. The bilinear weak form of equation (6.1.56) seeks $p \in V_f(\tilde{\Omega}_I)$ such that

$$\int_{\tilde{\Omega}_I} [\langle \tilde{\nabla} p, \tilde{\nabla} q \rangle - k^2 pq\,] d\tilde{\Omega}_I = \int_{\tilde{\Gamma}_S} gq d\tilde{\Gamma}_S \tag{6.1.59}$$

where the tildes indicate quantities defined over the complex extension of the domain Ω_I, and q represents the test function.

6.1.5.1.3. Step 3: Apply the chain rule From equation (6.1.58) and the Fundamental Theorem of Calculus, we see that

$$\frac{\partial \tilde{x}}{\partial x} = \gamma_x(x) = 1 \pm \frac{i}{\omega}\sigma(x) \tag{6.1.60}$$

Similar expressions hold for the y and z coordinates. This implies that the gradients of acoustic pressure can be transformed between the real and complex domains using a Jacobian

$$\boldsymbol{\nabla} p = \boldsymbol{J}_{cart} \tilde{\boldsymbol{\nabla}} p \tag{6.1.61}$$

where the Jacobian matrix for the Cartesian coordinate system $\boldsymbol{J}_{cart}$ is defined as

$$\boldsymbol{J}_{cart} = \begin{bmatrix} \gamma_x & 0 & 0 \\ 0 & \gamma_y & 0 \\ 0 & 0 & \gamma_z \end{bmatrix} \tag{6.1.62}$$

Conversely, we can map from the complex to the real derivatives using the inverse of the Jacobian.

$$\tilde{\boldsymbol{\nabla}} p = \boldsymbol{J}_{cart}^{-1} \boldsymbol{\nabla} p \tag{6.1.63}$$

where

$$\boldsymbol{J}_{cart}^{-1} = \begin{bmatrix} \frac{1}{\gamma_x} & 0 & 0 \\ 0 & \frac{1}{\gamma_y} & 0 \\ 0 & 0 & \frac{1}{\gamma_z} \end{bmatrix} \tag{6.1.64}$$

The scale factor that maps $\tilde{\Omega}_I$ into Ω_I is the determinant of the Jacobian,

$$W_{cart} = \gamma_x \gamma_y \gamma_z \tag{6.1.65}$$

6.1.5.1.4. Step 4: Revert to real-valued weak formulation

Using the previous results and the determinant relation from equation (6.1.65), the corresponding weak version of the Helmholtz equation is given as follows. Find $p \in V_f(\Omega_I)$ such that

$$\int_{\Omega_I} \left[(\boldsymbol{J}_{cart}^{-1} \nabla p) \cdot (\boldsymbol{J}_{cart}^{-1} \nabla q) - k^2 pq \right] W_{cart} d\Omega_I = \int_{\Gamma_S} gq dS. \tag{6.1.66}$$

We note that we can turn this into a Helmholtz equation with variable coefficients as follows

$$\int_{\Omega_I} [\boldsymbol{A}\langle \nabla p, \nabla q\rangle - k^2 pq] W_{cart}\ d\Omega_I = \int_{\Gamma_S} gq d\Gamma_S \tag{6.1.67}$$

where $\boldsymbol{A} = W_{cart} \boldsymbol{J}_{cart}^{-1} \boldsymbol{J}_{cart}^{-T}$. We note that $\boldsymbol{A}$ is a symmetric matrix, which follows from our choice to use a bilinear formulation rather than sesquilinear. Matrix $\boldsymbol{A}$ can be interpreted in a general way, without being tied to the cartesian coordinate system. The Jacobian matrices account for the different scaling factors for the various coordinate systems. Note that equation (6.1.67) achieves all the goals that were set from the beginning - a symmetric weak formulation over the real-valued domain, but with built-in dissipative properties stemming from the transformation to complex coordinates.

In the following sections, we will derive PML equations for rotated Cartesian, spherical, and ellipsoidal coordinates. In all cases, the weak formulation will be precisely the same as in equation (6.1.67), but with a different Jacobian matrix $\boldsymbol{J}$ and corresponding determinant W. Thus, we will only derive expressions for $\boldsymbol{J}$ in each of the coordinate systems.

6.1.5.2. Rotated Cartesian Coordinates

In this section we consider the case where the PML surface is extruded from a flat plane that is oriented at an arbitrary angle in three-dimensional space. If we define $\boldsymbol{x} = x_i, i = 1, 2, 3$ as the unrotated coordinates and $\boldsymbol{x}' = x_i', i = 1, 2, 3$ as the coordinates in the rotated coordinate system, we have

$$R = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{6.1.68}$$

where a_{ij} is the direction cosine between the x_i and x_i' axis. This defines the transformation as follows

$$\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x} \tag{6.1.69}$$

The Jacobian matrix for this case can be computed from the chain rule[100]

$$\boldsymbol{J}_{rotcart} = \frac{\partial(\tilde{x}, \tilde{y}, \tilde{z})}{\partial(x, y, z)} = \frac{\partial(\tilde{x}, \tilde{y}, \tilde{z})}{\partial(x'y', z')} \frac{\partial(x', y', z')}{\partial(x, y, z)} = \begin{bmatrix} \gamma_x & 0 & 0 \\ 0 & \gamma_y & 0 \\ 0 & 0 & \gamma_z \end{bmatrix} \boldsymbol{R} = \boldsymbol{J}_{cart}\boldsymbol{R} \tag{6.1.70}$$

The inverse of this matrix is given as

$$\boldsymbol{J}_{rotcart}^{-1} = \boldsymbol{R}^T \boldsymbol{J}_{cart}^{-1} \tag{6.1.71}$$

Thus, the coefficient matrix for this case is given by

$$\begin{aligned} \boldsymbol{A} &= W_{rotcart} \boldsymbol{J}_{rotcart}^{-1} \boldsymbol{J}_{rotcart}^{-T} \\ &= W_{rotcart} \boldsymbol{R}^T \boldsymbol{J}_{cart}^{-1} (\boldsymbol{J}_{cart}\boldsymbol{R})^{-T} \\ &= W_{cart} \boldsymbol{R}^T \boldsymbol{J}_{cart}^{-1} \boldsymbol{J}_{cart}^{-T} \boldsymbol{R} \end{aligned} \tag{6.1.72}$$

where we have used the fact that $W_{rotcart} = W_{cart}$. We see that this involves a simple rotation tensor transformation applied to the diagonal Jacobian matrix given in the unrotated case, equation (6.1.64). Thus, equation (6.1.67) applies, and can be used to construct the weak formulation in the rotated Cartesian case, but with a modified coefficient matrix $\boldsymbol{A}$ given in equation (6.1.72).

6.1.5.3. Spherical Coordinates

In a similar manner, we can derive the Jacobian matrix for a spherical PML. Though other researchers[122,36] have chosen to solve the spherical PML equations directly in spherical coordinates, we prefer to map the equations back to the Cartesian system to facilitate the finite element implementation. Thus, in this case our Jacobian needs to account for this additional transformation. The formulation for this case is given in.[100] The mapping from spherical to Cartesian coordinates is given as

$$\begin{aligned} x &= r\sin(\phi)\cos(\theta) \\ y &= r\sin(\phi)\sin(\theta) \\ z &= r\cos(\phi) \end{aligned} \tag{6.1.73}$$

The corresponding analytically continued coordinates are given as

$$\begin{aligned} \tilde{x} &= \tilde{r}\sin(\phi)\cos(\theta) \\ \tilde{y} &= \tilde{r}\sin(\phi)\sin(\theta) \\ \tilde{z} &= \tilde{r}\cos(\phi) \end{aligned} \tag{6.1.74}$$

Note that the complex coordinate stretching occurs only in the radial direction, as dissipative effect is not desired in the transverse directions. With these definitions the Jacobian matrix is given by the chain rule

$$\begin{aligned} \boldsymbol{J}_{spherical} = \frac{\partial(\tilde{x},\tilde{y},\tilde{z})}{\partial(x,y,z)} &= \frac{\partial(\tilde{x},\tilde{y},\tilde{z})}{\partial(r,\phi,\theta)}\frac{\partial(x,y,z)}{\partial(r,\phi,\theta)}^{-1} \\ = &\begin{bmatrix} \tilde{r}'\sin(\phi)\cos(\theta) & \tilde{r}\cos(\phi)\cos(\theta) & -\tilde{r}\sin(\phi)\sin(\theta) \\ \tilde{r}'\sin(\phi)\sin(\theta) & \tilde{r}\cos(\phi)\sin(\theta) & \tilde{r}\sin(\phi)\cos(\theta) \\ \tilde{r}'\cos(\phi) & -\tilde{r}\sin(\phi) & 0 \end{bmatrix} \\ &\begin{bmatrix} \sin(\phi)\cos(\theta) & r\cos(\phi)\cos(\theta) & -r\sin(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) & r\cos(\phi)\sin(\theta) & r\sin(\phi)\cos(\theta) \\ \cos(\phi) & -r\sin(\phi) & 0 \end{bmatrix}^{-1} \end{aligned} \tag{6.1.75}$$

Once again, equation (6.1.67) applies, and can be used to construct the weak formulation in the case of spherical coordinates, but with a modified coefficient matrix $\boldsymbol{A}$ given in equation (6.1.75).

We note that an advantage of the curvilinear PML formulation is that it is one-dimensional in the sense that the stretching only happens in one of the coordinate directions, in this case the radial direction. Thus, we can define the stretching as being in the radial direction only. This takes the form

$$\tilde{r} = r + \frac{i}{\omega}\int_R^r \sigma(\epsilon)d\epsilon \tag{6.1.76}$$

which implies that

$$\tilde{r}' = \frac{\partial\tilde{r}}{\partial r} = \gamma(r) = 1 + \frac{i}{\omega}\sigma(r) \tag{6.1.77}$$

6.1.5.4. Ellipsoidal Coordinates

In the case of ellipsoidal coordinates, we first must choose an appropriate coordinate system for the complex stretching of the PML. Ellipsoidal coordinates can be expressed in various ways, but we have found use of the coordinates developed by Burnett[30] to be the most convenient for defining the PML. We select the case of the prolate ellipsoid, with $a \geq b = c$. As in the spherical case, we prefer to solve the final equations in Cartesian coordinates rather than ellipsoidal. Thus, we will apply complex stretching to the ellipsoidal coordinate system, but will map the resulting equations back to Cartesian coordinates for the finite element solution. Once again, these transformations can be applied with the Jacobian.

We define an ellipsoidal radius[30] as

$$r = \frac{c_1 + c_2}{2} \tag{6.1.78}$$

where c_1 and c_2 are the distances of a given point on the ellipse to the two foci. We note that on the ellipsoidal surface, r is a constant, and is essentially a generalization of the notion of radial distance in the case of a sphere. Given the major and minor radii a and b of the ellipse, the distance to the focus along the major axis is given by $f = \sqrt{a^2 - b^2}$.

In terms of PML, we choose the direction of complex stretching to be along the direction defined in equation (6.1.78). We note that unlike the radial direction for a sphere, equation (6.1.78) defines curvilinear lines, and thus the PML layer will produce damping along those directions. This is necessary since if we were to define damping along straight-line paths (say in the direction normal to the ellipsoid surface), then the complex stretching would occur in all three directions r, ϕ, θ.

Given these parameters, the ellipsoidal coordinate system is defined as

$$\begin{aligned} x &= \sqrt{r^2 - f^2}\sin(\phi)\cos(\theta) \\ y &= \sqrt{r^2 - f^2}\sin(\phi)\sin(\theta) \\ z &= r\cos(\phi) \end{aligned} \tag{6.1.79}$$

Note that in the case of a sphere, $a = b = c$, which implies that $f = 0$, and these coordinates reduce to the spherical case. The stretched coordinates in the ellipsoidal case are given by

$$\begin{aligned} \tilde{x} &= \sqrt{\tilde{r}^2 - f^2}\sin(\phi)\cos(\theta) \\ \tilde{y} &= \sqrt{\tilde{r}^2 - f^2}\sin(\phi)\sin(\theta) \\ \tilde{z} &= \tilde{r}\cos(\phi) \end{aligned} \tag{6.1.80}$$

This implies that the transformation matrix is given as

$$\boldsymbol{J}_{ellipsoidal} = \frac{\partial(\tilde{x},\tilde{y},\tilde{z})}{\partial(x,y,z)} = \frac{\partial(\tilde{x},\tilde{y},\tilde{z})}{\partial(r,\phi,\theta)} \frac{\partial(x,y,z)}{\partial(r,\phi,\theta)}^{-1}$$

$$= \begin{bmatrix} \frac{\tilde{r}\tilde{r}'}{\sqrt{\tilde{r}^2 - f^2}}\sin(\phi)\cos(\theta) & \sqrt{\tilde{r}^2 - f^2}\cos(\phi)\cos(\theta) & -\sqrt{\tilde{r}^2 - f^2}\sin(\phi)\sin(\theta) \\ \frac{\tilde{r}\tilde{r}'}{\sqrt{\tilde{r}^2 - f^2}}\sin(\phi)\sin(\theta) & \sqrt{\tilde{r}^2 - f^2}\cos(\phi)\sin(\theta) & \sqrt{\tilde{r}^2 - f^2}\sin(\phi)\cos(\theta) \\ \tilde{r}'\cos(\phi) & -\tilde{r}\sin(\phi) & 0 \end{bmatrix} \tag{6.1.81}$$

$$\begin{bmatrix} \frac{r}{\sqrt{r^2 - f^2}}\sin(\phi)\cos(\theta) & \sqrt{r^2 - f^2}\cos(\phi)\cos(\theta) & -\sqrt{r^2 - f^2}\sin(\phi)\sin(\theta) \\ \frac{r}{\sqrt{r^2 - f^2}}\sin(\phi)\sin(\theta) & \sqrt{r^2 - f^2}\cos(\phi)\sin(\theta) & \sqrt{r^2 - f^2}\sin(\phi)\cos(\theta) \\ \cos(\phi) & -r\sin(\phi) & 0 \end{bmatrix}^{-1}$$

6.1.5.5. Ellipsoidal Coordinates with X axis as Major axis

The previous section assumed that the major axis of the ellipse was oriented along the z direction. For completeness, we show here how to adjust the formulation in the case when the major axis is along the x direction. In this case the ellipsoidal coordinate system is defined as

$$\begin{aligned} x &= rcos(\phi) \\ y &= \sqrt{r^2 - f^2}\sin(\phi)\sin(\theta) \\ z &= \sqrt{r^2 - f^2}\sin(\phi)\cos(\theta) \end{aligned} \tag{6.1.82}$$

Note that in the case of a sphere, $a = b = c$, which implies that $f = 0$, and these coordinates reduce to the spherical case. The stretched coordinates in the ellipsoidal case are given by

$$\begin{aligned} \tilde{x} &= \tilde{r}\cos(\phi) \\ \tilde{y} &= \sqrt{\tilde{r}^2 - f^2}\sin(\phi)\sin(\theta) \\ \tilde{z} &= \sqrt{\tilde{r}^2 - f^2}\sin(\phi)\cos(\theta) \end{aligned} \tag{6.1.83}$$

This implies that the Jacobian matrix is given as

$$\begin{aligned} \boldsymbol{J} = \frac{\partial(\tilde{x},\tilde{y},\tilde{z})}{\partial(x,y,z)} &= \frac{\partial(\tilde{x},\tilde{y},\tilde{z})}{\partial(r,\phi,\theta)}\frac{\partial(x,y,z)}{\partial(r,\phi,\theta)}^{-1} \\ = \begin{bmatrix} \tilde{r}'\cos(\phi) & -\tilde{r}\sin(\phi) & 0 \\ \frac{\tilde{r}\tilde{r}'}{\sqrt{\tilde{r}^2-f^2}}\sin(\phi)\sin(\theta) & \sqrt{\tilde{r}^2-f^2}\cos(\phi)\sin(\theta) & \sqrt{\tilde{r}^2-f^2}\sin(\phi)\cos(\theta) \\ \frac{\tilde{r}\tilde{r}'}{\sqrt{\tilde{r}^2-f^2}}\sin(\phi)\cos(\theta) & \sqrt{\tilde{r}^2-f^2}\cos(\phi)\cos(\theta) & -\sqrt{\tilde{r}^2-f^2}\sin(\phi)\sin(\theta) \end{bmatrix} \\ \begin{bmatrix} \cos(\phi) & -r\sin(\phi) & 0 \\ \frac{r}{\sqrt{r^2-f^2}}\sin(\phi)\sin(\theta) & \sqrt{r^2-f^2}\cos(\phi)\sin(\theta) & \sqrt{r^2-f^2}\sin(\phi)\cos(\theta) \\ \frac{r}{\sqrt{r^2-f^2}}\sin(\phi)\cos(\theta) & \sqrt{r^2-f^2}\cos(\phi)\cos(\theta) & -\sqrt{r^2-f^2}\sin(\phi)\sin(\theta) \end{bmatrix}^{-1} \end{aligned} \tag{6.1.84}$$

6.1.5.6. Relations Between the PML Formulations

It is clear that as the minor and major axis become equal, $a = b = c$, and hence $f = 0$. This implies that the Jacobian for ellipsoidal coordinates in equation (6.1.81) reduces to the spherical Jacobian given in equation (6.1.75).

As an additional step, we consider that the spherical Jacobian reduces to that of the Cartesian in the limiting case of a large radius of the inner sphere defining the PML boundary. This can be seen by considering equations (6.1.76) and (6.1.77), which we repeat here for convenience

$$\tilde{r} = r + \frac{i}{\omega}\int_R^r \sigma(\epsilon)d\epsilon \tag{6.1.85}$$

which implies that

$$\tilde{r}' = \frac{\partial\tilde{r}}{\partial r} = \gamma(r) = 1 + \frac{i}{\omega}\sigma(r) \tag{6.1.86}$$

As r and hence R become very large, we see from equation (6.1.76) that then $\tilde{r} \longrightarrow r$, since the imaginary term will become vanishingly small compared to r. However, from equation (6.1.77) we see no limiting change in $\tilde{r}'$ as r becomes large, since $\sigma(R) = 0$ and $\sigma(r)$ will be bounded by the thickness of the PML layer. Thus, going back to equation (6.1.75), we have:

$$\begin{aligned}
\boldsymbol{J}_{spherical} = \frac{\partial(\tilde{x},\tilde{y},\tilde{z})}{\partial(x,y,z)} &= \frac{\partial(\tilde{x},\tilde{y},\tilde{z})}{\partial(r,\phi,\theta)} \frac{\partial(x,y,z)}{\partial(r,\phi,\theta)}^{-1} \\
&= \begin{bmatrix} \tilde{r}' \sin(\phi)\cos(\theta) & \tilde{r}\cos(\phi)\cos(\theta) & -\tilde{r}\sin(\phi)\sin(\theta) \\ \tilde{r}' \sin(\phi)\sin(\theta) & \tilde{r}\cos(\phi)\sin(\theta) & \tilde{r}\sin(\phi)\cos(\theta) \\ \tilde{r}' \cos(\phi) & -\tilde{r}\sin(\phi) & 0 \end{bmatrix} \\
&\quad \begin{bmatrix} \sin(\phi)\cos(\theta) & r\cos(\phi)\cos(\theta) & -r\sin(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) & r\cos(\phi)\sin(\theta) & r\sin(\phi)\cos(\theta) \\ \cos(\phi) & -r\sin(\phi) & 0 \end{bmatrix}^{-1} \\
&\longrightarrow \begin{bmatrix} \tilde{r}' \sin(\phi)\cos(\theta) & r\cos(\phi)\cos(\theta) & -r\sin(\phi)\sin(\theta) \\ \tilde{r}' \sin(\phi)\sin(\theta) & r\cos(\phi)\sin(\theta) & r\sin(\phi)\cos(\theta) \\ \tilde{r}' \cos(\phi) & -r\sin(\phi) & 0 \end{bmatrix} \\
&\quad \begin{bmatrix} \sin(\phi)\cos(\theta) & r\cos(\phi)\cos(\theta) & -r\sin(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) & r\cos(\phi)\sin(\theta) & r\sin(\phi)\cos(\theta) \\ \cos(\phi) & -r\sin(\phi) & 0 \end{bmatrix}^{-1} \\
&= \begin{bmatrix} \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) & -\sin(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) & \cos(\phi)\sin(\theta) & \sin(\phi)\cos(\theta) \\ \cos(\phi) & -\sin(\phi) & 0 \end{bmatrix} \\
\begin{bmatrix} \gamma(r) & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{r} & 0 \\ 0 & 0 & \frac{1}{r} \end{bmatrix} &\begin{bmatrix} \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) & -\sin(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) & \cos(\phi)\sin(\theta) & \sin(\phi)\cos(\theta) \\ \cos(\phi) & -\sin(\phi) & 0 \end{bmatrix}^{-1}
\end{aligned} \tag{6.1.87}$$

For the cartesian case in the pure x direction, $\phi = \frac{\pi}{2}$ and $\theta = 0$.

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \tag{6.1.88}$$

and

$$J = \begin{bmatrix} \gamma(r) & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.1.89}$$

Similar substitutions can be applied for other values of ϕ and θ that show the Jacobian reduce to a rotation between spherical and cartesian coordinates. For off axes cases, the Jacobian will be a full matrix. Thus, the limiting case of a large radius for the PML surface reduces to a one-dimensional PML layer. Constructing a tensor product with PML layers in the other two directions produces a diagonal Jacobian matrix as given for the Cartesian case in equation (6.1.62).

6.2. Waterline Determination

We develop the approach for solution of a rigid body floating in a fluid. When the ship is treated as a rigid body, its equilibrium equations simplify to six equations in six unknowns that involve force and moment balances in three coordinate directions. However, from symmetry considerations we may assume that the displacements of the ship are zero in the plane of the waterline. Further, we assume that the angular rotation of the ship about an axis normal to the waterline is also zero. Thus, the six equilibrium equations can be reduced to three. For convenience, we take the ship to be fixed in space while the orientation of the waterline plane is described by in-plane rotations θ_1 and θ_2. The position of the ship mass center above and perpendicular to the waterline is denoted by the coordinate z. Additional details on the coordinate z and the angles θ_1 and θ_2 are provided in Section 6.2.1.

Since the three equilibrium equations are nonlinear in the angles θ_1 and θ_2, we employ Newton's method for their solution. The Newton step that is associated with the three equilibrium equations is obtained from the solution of the linear system

$$\boldsymbol{K}_T \begin{bmatrix} \Delta z \\ \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix} = - \begin{bmatrix} F_3 \\ M_1 \\ M_2 \end{bmatrix}, \tag{6.2.1}$$

where $\boldsymbol{K}_T$ is the tangent stiffness matrix. The terms Δz, $\Delta\theta_1$, and $\Delta\theta_2$ are incremental updates to the coordinate z and the two angles θ_1 and θ_2. The terms on the right-hand side of (6.2.1) involve the net force and moments acting about the ship center of mass due to buoyancy forces (pressure loads from water) and gravity. Again, more details are provided later on the precise form of these terms. Additional details on the implementation of Newton's method are provided in § 6.2.4

6.2.1. *Reference Frames*

The position vector of a node n in a fixed reference frame A can be expressed as

$$\boldsymbol{p}_n = x_{n,1}\boldsymbol{a}_1 + x_{n,2}\boldsymbol{a}_2 + x_{n,3}\boldsymbol{a}_3, \tag{6.2.2}$$

where $(x_{n,1}, x_{n,2}, x_{n,3})$ are the coordinates of the node and $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$ are unit vectors aligned with coordinate directions X_1, X_2, X_3. We note in the present context that $(x_{n,1}, x_{n,2}, x_{n,3})$ are the coordinates of the node in the **Exodus** finite element model used by **Sierra/SD**. Further, we take $\boldsymbol{a}_3$ to be directed vertically upward.

Consider a rigid body B with attached unit vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$ that are initially aligned with $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$. A rotation of B by θ_1 about the $\boldsymbol{a}_1$ direction results in

$$\boldsymbol{b}_1 = \boldsymbol{a}_1, \quad \boldsymbol{b}_2 = \cos\theta_1\boldsymbol{a}_2 + \sin\theta_1\boldsymbol{a}_3, \quad \boldsymbol{b}_3 = \cos\theta_1\boldsymbol{a}_3 - \sin\theta_1\boldsymbol{a}_2. \tag{6.2.3}$$

Next, consider a rigid body C with attached unit vectors $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ that are initially aligned with $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$. A rotation of C by θ_2 about the $\boldsymbol{b}_2$ direction gives us

$$\boldsymbol{c}_1 = \cos\theta_2\boldsymbol{b}_1 - \sin\theta_2\boldsymbol{b}_3, \quad \boldsymbol{c}_2 = \boldsymbol{b}_2, \quad \boldsymbol{c}_3 = \cos\theta_2\boldsymbol{b}_3 + \sin\theta_2\boldsymbol{b}_1. \tag{6.2.4}$$

Combining (6.2.3) and (6.2.4), we find

$$\boldsymbol{c}_1 = \cos\theta_2\boldsymbol{a}_1 + \sin\theta_2\sin\theta_1\boldsymbol{a}_2 - \sin\theta_2\cos\theta_1\boldsymbol{a}_3, \tag{6.2.5}$$

$$\boldsymbol{c}_2 = \cos\theta_1\boldsymbol{a}_2 + \sin\theta_1\boldsymbol{a}_3, \tag{6.2.6}$$

$$\boldsymbol{c}_3 = \sin\theta_2\boldsymbol{a}_1 - \cos\theta_2\sin\theta_1\boldsymbol{a}_2 + \cos\theta_2\cos\theta_1\boldsymbol{a}_3. \tag{6.2.7}$$

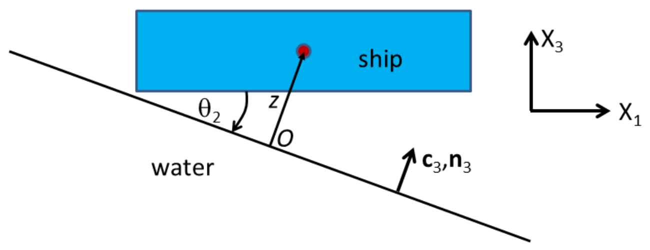


Figure 6-5. – Sketch showing ship, origin O of waterline frame, coordinate z, and angle θ_2.

For purposes of convenience, we choose unit vector $\boldsymbol{c}_3$ to be in the direction normal to the waterline and directed away from the water. Similarly, unit vectors $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are also attached to the waterline *frame*. Using summation notation, (6.2.5-6.2.7) can be expressed concisely as

$$\boldsymbol{c}_i = c_{ij}\boldsymbol{a}_j, \tag{6.2.8}$$

where the scalar coefficient $c_{ij} = \boldsymbol{c}_i \cdot \boldsymbol{a}_j$ and appears as the entry in row i and column j of the direction cosine matrix

$$D = \begin{bmatrix} \cos\theta_2 & \sin\theta_1 \sin\theta_2 & -\cos\theta_1 \sin\theta_2 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ \sin\theta_2 & -\sin\theta_1 \cos\theta_2 & \cos\theta_1 \cos\theta_2 \end{bmatrix}.$$

We note that the columns of D are orthonormal, i.e., $D^{-1} = D^T$.

The origin O of the waterline frame is chosen as the point of intersection of the line in direction $\boldsymbol{c}_3$ passing through the ship mass center with the plane of the water (see Figure 6-5). Thus, the position vector of the center of mass of the ship relative to O can be expressed as

$$\boldsymbol{p}_{cm/O} = z\boldsymbol{c}_3. \tag{6.2.9}$$

6.2.2. *Pressure at a Node*

We would like to express the position vector of a node as in (6.2.2) relative to O rather than the origin of reference frame A. To this end, let the position vector of the center of mass of the ship relative to the origin of A be expressed as

$$\boldsymbol{p}_{cm} = x_{cm,1}\boldsymbol{a}_1 + x_{cm,2}\boldsymbol{a}_2 + x_{cm,3}\boldsymbol{a}_3. \tag{6.2.10}$$

We note the coordinates $(x_{cm,1}, x_{cm,2}, x_{cm,3})$ are readily available from **Sierra/SD**. Next, let the position vector of O relative to the origin of A be expressed as

$$\boldsymbol{p}_O = x_{O,1}\boldsymbol{a}_1 + x_{O,2}\boldsymbol{a}_2 + x_{O,3}\boldsymbol{a}_3. \tag{6.2.11}$$

Since $\boldsymbol{p}_{cm} = \boldsymbol{p}_O + \boldsymbol{p}_{cm/O}$, it follows from the previous three equations and (6.2.8) that

$$x_{O,j} = x_{cm,j} - zc_{3j} \quad j = 1, 2, 3. \tag{6.2.12}$$

The pressure at node n depends on its depth below the waterline. Specifically,

$$\begin{aligned} p(n) &= -\rho g(\boldsymbol{p}_n - \boldsymbol{p}_O) \cdot \boldsymbol{c}_3 \\ &= -\rho g((x_{n,1} - x_{O,1})c_{13} + (x_{n,2} - x_{O,2})c_{23} + (x_{n,3} - x_{O,3})c_{33}), \end{aligned} \tag{6.2.13}$$

where ρ is the density of water and g is the acceleration of gravity. If the pressure calculated from (6.2.13) is negative, this indicates the node is above the waterline and we set $p(n) = 0$.

6.2.3. Waterline Plane Specification

The initial guess in the Solution section is defined by $\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3$ not on a line. Plowing on,

$$\boldsymbol{v}_1 := \boldsymbol{t}_2 - \boldsymbol{t}_1, \quad \boldsymbol{v}_2 := \boldsymbol{t}_3 - \boldsymbol{t}_1,$$

the unit normal to this plane is given by

$$\boldsymbol{n} = \frac{\boldsymbol{v}_1 \times \boldsymbol{v}_2}{\|\boldsymbol{v}_1 \times \boldsymbol{v}_2\|} = n_1\boldsymbol{a}_1 + n_2\boldsymbol{a}_2 + n_3\boldsymbol{a}_3. \tag{6.2.14}$$

If $\boldsymbol{n} \cdot \boldsymbol{a}_3 = n_3 < 0$, then we multiply $\boldsymbol{n}$ by -1 so that $\boldsymbol{n}$ points out of the water rather than into it.

We next show how to relate the waterline plane to the variables θ_1, θ_2 and z. Since $\boldsymbol{n} = \boldsymbol{c}_3$, we find from (6.2.7) and (6.2.14) that

$$\sin\theta_2 = n_1, \quad -\sin\theta_1\cos\theta_2 = n_2, \quad \cos\theta_1\cos\theta_2 = n_3, \tag{6.2.15}$$

from which follows

$$\theta_2 = \arcsin(n_1), \quad \theta_1 = \arctan(-n_2/n_3). \tag{6.2.16}$$

We will print a warning message if either $|\theta_1|$ or $|\theta_2|$ is greater than $\pi/4$ (45 degrees). Since the origin O is in the plane of the waterline, $\boldsymbol{n} = \boldsymbol{c}_3$, and $\boldsymbol{p}_O = \boldsymbol{p}_{cm} - \boldsymbol{p}_{cm/O}$, we find from (6.2.9) and (6.2.10) that

$$\begin{aligned} z &= (\boldsymbol{p}_{cm} - \boldsymbol{p}_O) \cdot \boldsymbol{n} \\ &= (x_{cm,1} - x_{O,1})n_1 + (x_{cm,2} - x_{O,2})n_2 + (x_{cm,3} - x_{O,3})n_3. \end{aligned} \tag{6.2.17}$$

We note in the previous expression that $\boldsymbol{p}_O$ may be replaced by either $\boldsymbol{t}_1$, $\boldsymbol{t}_2$ or $\boldsymbol{t}_3$ since these three points are also in the waterline plane.

As described later, Newton's method is used to solve one force and two equilibrium equations in terms of the coordinate z and the angles θ_1 and θ_2. After a converged solution is obtained, it is important for the analyst to confirm that the sideset used for the problem specification includes all element faces of the outer ship surface which contain one or more nodes below the waterline.

6.2.4. Net Force and Moment Calculation

With equation (6.2.13) in hand, **Sierra/SD** can be used to calculate and assemble the water pressure loads into equivalent nodal loads. This process involves the interpolation of nodal pressures to Gauss points and numerical integration. The equivalent nodal loads can then be used to determine the net force and moment acting on the ship. We outline a procedure for doing this calculation in the following paragraphs.

Let f_i denote the load vector for subdomain (processor) i resulting from water pressure loads. We note each row of f_i corresponds to a load for a particular degree of freedom. For example, row 7 of f_i may correspond to a force at a specific node in coordinate direction 3. The vector f_i is associated with a set $\mathbf{N}_i$ of nodes in subdomain i. Further, we note that the force vector $\boldsymbol{f}_n$ and the moment vector $\boldsymbol{m}_n$ at node $n \in \mathbf{N}_i$ can be extracted directly from f_i.

Let $\boldsymbol{r}_n := \boldsymbol{p}_n - \boldsymbol{p}_{cm}$ denote the position vector from the ship center of mass to node n. Summing contributions from all the nodes in $\mathbf{N}_i$, we find that the net force and moment contribution from subdomain i is given by

$$\boldsymbol{F}_i = \sum_{n \in \mathbf{N}_i} \boldsymbol{f}_n, \tag{6.2.18}$$

$$\boldsymbol{M}_i = \sum_{n \in \mathbf{N}_i} \boldsymbol{r}_n \times \boldsymbol{f}_n. \tag{6.2.19}$$

Summing contributions from all N subdomains, the net force and moment about the mass center of the ship is given by

$$\boldsymbol{F}_s = \sum_{i=1}^{N} \boldsymbol{F}_i = F_{s,1}\boldsymbol{a}_1 + F_{s,2}\boldsymbol{a}_2 + F_{s,3}\boldsymbol{a}_3 \tag{6.2.20}$$

$$\boldsymbol{M}_s = \sum_{i=1}^{N} \boldsymbol{M}_i = M_{s,1}\boldsymbol{a}_1 + M_{s,2}\boldsymbol{a}_2 + M_{s,3}\boldsymbol{a}_3. \tag{6.2.21}$$

Returning to (6.2.1), we have

$$F_3 = \boldsymbol{F}_s \cdot \boldsymbol{c}_3 - m_s g = c_{3,1}F_{s,1} + c_{3,2}F_{s,2} + c_{3,3}F_{s,3} - m_s g, \tag{6.2.22}$$

$$M_1 = \boldsymbol{M}_s \cdot \boldsymbol{c}_1 = c_{1,1}M_{s,1} + c_{1,2}M_{s,2} + c_{1,3}M_{s,3}, \tag{6.2.23}$$

$$M_2 = \boldsymbol{M}_s \cdot \boldsymbol{c}_2 = c_{2,1}M_{s,1} + c_{2,2}M_{s,2} + c_{2,3}M_{s,3}, \tag{6.2.24}$$

where m_s is the mass of the ship.

Newton's Method. The initial solution of the nonlinear equations applies Newton's method directly on the non-symmetric $\boldsymbol{K}_T$. The matrix $\boldsymbol{K}_T$ will in general be non-symmetric due to follower contributions. If convergence issues arise, we may be regularized using a variety of approaches.

The method can be summarized as follows.

1. Let $f(p)$ represent the force balance, with p, the parameters equal to z, θ_1, and θ_2.
2. Let $\boldsymbol{K}_T(p) = df(p)/dp$ represent the tangent stiffness matrix obtained by differentiating the force balance with respect to the input parameters.
3. For each iteration, Newton's method estimates a new parameter set,

$$p_{n+1} = p_n - \boldsymbol{K}_T^{-1} f(p_n)$$

4. Iteration continues until the force balance approaches zero.

Tangent Matrix. We apply finite differences together with (6.2.22-6.2.24) to calculate the tangent matrix, $\boldsymbol{K}_T$. We use a finite difference step size of 0.001 for the dimensionless variables θ_1 and θ_2, while the step size for z is 0.001 times a characteristic length of the ship.

6.3. Fluid Coupling through Lighthill's Tensor

Convective, turbulent flow may be effectively coupled to acoustic formulations for sound propagation using the Lighthill analogy. For convenience, we use a pressure formulation of the acoustic medium.

The inviscid Euler equations given in equation (3.1.10) including a source term are given by

$$\frac{\partial \rho}{\partial t} + \rho_0 \nabla \cdot \mathbf{u} = 0, \tag{6.3.1}$$

$$\rho_0 \frac{\partial \mathbf{u}}{\partial t} + \nabla p = \mathbf{S}, \tag{6.3.2}$$

where ρ_0 is a reference density, ρ is density, p is pressure, $\mathbf{u}$ is particle velocity, and $\mathbf{S}$ is a source term. We note that in equation (6.3.2) the Pressure and density are related as

$$c_0^2 \rho = p. \tag{6.3.3}$$

Pressure formulation.

The acoustic pressure formulation is obtained by combining the mass and momentum balance equations. The time derivative of (6.3.1) is

$$\ddot{\rho} + \rho_0 \nabla \cdot \dot{\mathbf{u}} = 0, \tag{6.3.4}$$

where a superposed dot represented partial differentiation with respect to time. The divergence of (6.3.2) is

$$\rho_0 \nabla \cdot \dot{\mathbf{u}} + \nabla^2 p = \nabla \cdot \mathbf{S}. \tag{6.3.5}$$

Substituting (6.3.3) and subsequently eliminating $\nabla \cdot \dot{\mathbf{u}}$, the acoustic pressure equation is

$$\frac{1}{c_0^2} \ddot{p} - \nabla^2 p = -\nabla \cdot \mathbf{S}. \tag{6.3.6}$$

Lighthill's analogy[93] is an approach to the problem of sound generation and propagation in turbulent flow. The equations of motion are rearranged into a scalar, inhomogeneous wave equation where the source terms are the noise generation due to turbulence in the fluid:

$$\ddot{\rho} - c_0^2 \nabla^2 \rho = \nabla \cdot (\nabla \cdot \mathbf{T}), \tag{6.3.7}$$

where $\mathbf{T}$ is known as the *Lighthill tensor*. It is expressed in Cartesian component form as

$$T_{ij} = \rho u_i u_j + (p - c_0^2 \rho)\delta_{ij} - \tau_{ij}, \tag{6.3.8}$$

where the tensor τ is the viscous stress tensor for the fluid.

The pressure form of (6.3.7) is

$$\ddot{p} - c_0^2 \nabla^2 p = c_0^2 \nabla \cdot (\nabla \cdot \mathbf{T}) \tag{6.3.9}$$

In **Sierra/SD**, only the pressure formulation of Lighthill's method as given by the above equations is implemented. This is in contrast to most acoustic solutions which employ a velocity potential formulation.

Hand-off of Lighthill Tensor from Fuego.

The incompressible form of the Lighthill tensor is given by,

$$T_{ij} = \rho u_i u_j - \tau_{ij}. \tag{6.3.10}$$

Fuego provides $\nabla \cdot \mathbf{T}$ of equation (6.3.10) as a *nodal variable* with an arbitrary name on an SD acoustic mesh. We implement the weak form of (6.3.9) in **Sierra/SD**.

6.4. Analysis of Rotating Structures

In addition to the standard mass and stiffness matrices that arise in linear structural dynamics, force-based matrices are also common. The most common include follower stiffness matrices from applied pressures, and Coriolis/centrifugal matrices in rotating structures. These notes describe the design of the interface for these additional matrices. We will focus on the following three terms

1. Follower stiffness matrix from applied pressure. This is a nonsymmetric term, but is symmetrized, and becomes part of the stiffness matrix.
2. Centrifugal stiffness in rotating structures. This is a symmetric term, and becomes part of the stiffness matrix.
3. Angular velocity adds a Coriolis damping matrix and a Coriolis virtual load. Angular acceleration adds both an acceleration matrix to the stiffness and and acceleration virtual load. Coriolis damping does not dissipate energy.

For rotating structures, the formulation is called the *Lagrangian* formulation to distinguish it from an *Eulerian* formulation developed for a customer who maintains all documentation and testing of the *Eulerian* formulation. Each models a structure in a coordinate system as in Figure 6-6, with fixed origin, rotating with user specified angular velocity ω, and angular acceleration $\dot{\omega}$.

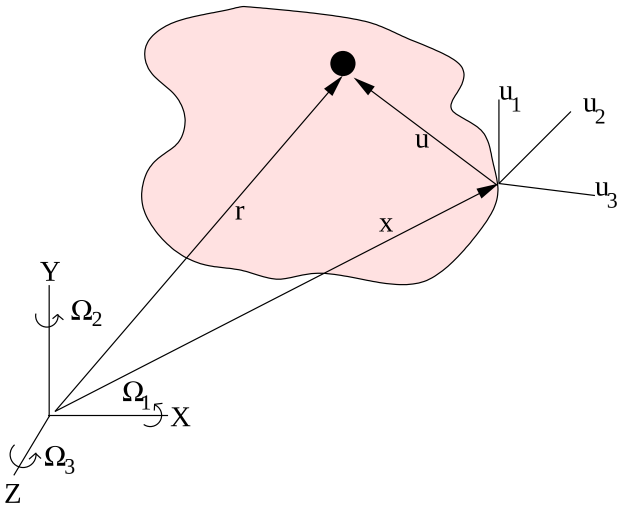


Figure 6-6. – A schematic of a structure that is rotating about fixed coordinate axes.

In the remainder of this document, the formulation is assumed to be the Lagrangian formulation.

The design of the implementation of this capability is documented.[127]

Symbol	Description
K_a	angular acceleration
K_g	geometric stiffness matrix
G	Coriolis matrix
K_c	centrifugal softening matrix

,

Symbol	Description
f_a	angular acceleration
f_c	centrifugal force

Table 6-1. – Notation for stiffness and damping matrices (left) and forces (right).

For readers puzzled by the absence of Euler's force in Table 6-1, the angular acceleration force f_a is also called Euler's force, and the angular acceleration matrix, K_a, is also called the Euler force matrix.

The formulation and discretization is defined completely in the remaining subsections 6.4.1. The *Lagrangian* approach[83] seeks deformations about the rotating coordinate system. Of the many papers on this topic, we found a few to be especially helpful.[90,54,121,120] In the notation of Table 6-1, the governing equation is,

$$M\ddot{u} + G\dot{u} + \left(K + K_g + K_a + K_c\right)u = f_a + f_c. \tag{6.4.1}$$

Equation (6.4.1) depends on the unknown K_g which is determined from a preliminary statics simulation,

$$\left(K + K_a + K_c\right)u = f_c + f_a. \tag{6.4.2}$$

We assume that essential boundary conditions have been applied. Furthermore we assume that these essential boundary conditions ensure that no rigid body modes are present. Equivalently, $K + K_a + K_c$ is nonsingular. The geometric stiffness matrix K_g is determined from the associated stress field.

In the current implementation, ω does not vary in time, even if $\dot{\omega}$ is nonzero. We assume that the user has chosen a sufficiently brief simulation that the change in ω is negligible.

A direct time stepping algorithm based on equation (6.4.1) will approximate the the time history of the structural displacement u about the rotating frame. Linear and nonlinear static and transient simulations are tested. No one on the current SD Team knows what the nonlinear methods do.

Once K_g has been determined, the associated eigenvalue problem is quadratic, with gyroscopic damping. The eigenvalue problem corresponding to the *ansatz* $\hat{u}(x,t) = u(x)e^{\lambda t}$. Setting the force terms to zero,

$$\left(\lambda^2 M + \lambda G + (K + K_g + K_a + K_c)\right)u = 0. \tag{6.4.3}$$

There are two motivations for using the symbol G for Coriolis' matrix. The first is that it is also called the gyroscopic damping. The element level Coriolis matrix of equation (6.4.11), is skew symmetric, implying that the eigenvalue $\lambda = \sqrt{-1}\omega$ for a real circular frequency ω. The second, and actually much more important, reason to use a G is to avoid confusion with the damping matrix and the constraint matrix, which are both denoted by C.

6.4.1. Formulation and Discretization

The weak form is derived from the kinematics. Structural element assembly is discussed first, before solid element assembly.

Symbol	Description
B	rotating frame
N	inertial frame
$\{\boldsymbol{b}_i\}_{i=1}^3$	basis aligned with B
$\boldsymbol{\omega}$	angular velocity
$\boldsymbol{u}$	displacement about B
$\boldsymbol{r}$	position vector
$\boldsymbol{x}$	position in B

Table 6-2. – Notation for Kinematics.

As summarized in Table 6-2, $\boldsymbol{\omega}$ is the angular velocity of B in N. The origin in B is fixed, and B has zero translation with respect to the inertial frame N. N and B share the same origin. Notice that the time derivative of $\boldsymbol{x}_i$ in the rotating frame is zero.

The displacement, $\boldsymbol{u}$, of a point in frame N rotating with respect to the frame B has position vector $\boldsymbol{r}$,

$$\boldsymbol{r} = \boldsymbol{x} + \boldsymbol{u}. \tag{6.4.4}$$

Here $\boldsymbol{x}$ is the undeformed position of the point in B. In terms of $\omega = \Omega_i \boldsymbol{b}_i$, the velocity of the point in an inertial frame N is then[83]

$$\boldsymbol{v} = \frac{{}^N d}{dt}\boldsymbol{r} = \frac{{}^B d}{dt}\boldsymbol{r} + \boldsymbol{\omega} \times \boldsymbol{r} = \dot{\boldsymbol{u}} + \boldsymbol{\omega} \times (\boldsymbol{x} + \boldsymbol{u}). \tag{6.4.5}$$

Taking another derivative, we find the acceleration of the point in an inertial frame to be

$$\boldsymbol{a} = \frac{{}^N d}{dt}\boldsymbol{v} = \ddot{\boldsymbol{u}} + 2\boldsymbol{\omega} \times \dot{\boldsymbol{u}} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times (\boldsymbol{x} + \boldsymbol{u})) + \dot{\boldsymbol{\omega}} \times \boldsymbol{x} + \dot{\boldsymbol{\omega}} \times \boldsymbol{u}. \tag{6.4.6}$$

This completes the description of the kinematics.

In an inertial reference frame, the homogeneous (zero force) equation of motion of a solid three dimensional body is

$$\rho\ddot{r} - \nabla \cdot \sigma = 0.$$

The only non-standard step in the derivation of the weak formulation, by multiplying equation (6.4.1) by a test function v, and ultimately integrating by parts, is the substitution equations (6.4.4) and (6.4.6). In order to emphasize the names of the different terms, Coriolis (g), acceleration (a), and centripetal (c) bilinear forms are introduced, along with the corresponding acceleration and centripetal force functions b_a and b_c,

$$\begin{aligned} g(\dot{\boldsymbol{u}}, v) &= 2\langle \omega \times \dot{\boldsymbol{u}}, v\rangle & & \\ a(\boldsymbol{u}, v) &= \langle \dot{\boldsymbol{\omega}} \times \boldsymbol{u}, v\rangle & b_a(v) &= -\langle \dot{\boldsymbol{\omega}} \times \boldsymbol{x}, v\rangle \\ c(\boldsymbol{u}, v) &= \langle \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{u}), v\rangle & b_c(v) &= -\langle \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{x}), v\rangle. \end{aligned}$$

To match the order of the terms in the governing matrix equation (6.4.1), arranging the resulting terms as,

$$\begin{aligned} \langle \ddot{\boldsymbol{u}}, v\rangle + g(\dot{\boldsymbol{u}}, v) + \int_V \sigma : \nabla v dV + a(\boldsymbol{u}, v) + c(\boldsymbol{u}, v) \\ = \int_S \sigma_n v dS + b_a(v) + b_c(v) \end{aligned} \tag{6.4.7}$$

Note that the stiffness includes both the stiffness K associated with the material properties, as well as the geometric stiffness K_g associated with the stress state due to the steady-state spinning problem.

The centrifugal matrix K_c, which corresponds to $c(\boldsymbol{u}, v)$ is symmetric. This can be demonstrated using the properties of the cross product. As expected,

$$g(\boldsymbol{u}, v) = -g(v, \boldsymbol{u}), \quad a(\boldsymbol{u}, v) = -a(v, \boldsymbol{u}).$$

A standard nodal finite element discretization is used. The vector shape function, $\vec{N}_i$, for node i and in coordinate direction k depends on the scalar i^{th} shape function $\phi_i(x)$ for node i and a column, $\vec{e}_k$, of the appropriate identity matrix,

$$N_i(x) = e_k\phi_i(x), \quad \hat{u}(x,t) = N_i(x)u_i(t).$$

The element matrices and forces are defined as usual by evaluating the integrals in equation (6.4.7) over the element, with entries,

$$\begin{aligned} G_{ij} &= g(N_i, N_j) \\ K_{a,ij} &= a(N_i, N_j) \qquad f_a^j = b_a(N_j) \\ K_{c,ij} &= c(N_i, N_j) \qquad f_c^j = b_c(N_j) \end{aligned} \tag{6.4.8}$$

Note that our isoparametric formulation evaluates forces using

$$\boldsymbol{x} = N_i \mathbf{x}_i.$$

Also as usual, the 3 by 3 spin matrix $\bar{\Omega}$ is defined so that $\omega \times r = \bar{\Omega}\, r$.

6.4.1.1. A Two Node Beam Element

Although the weak form defines assembly in terms of integrals over elements, the goal of this section is to explain how an implementation can be simplified by using equivalent algebraic expressions for element matrices and loads in terms of the element mass matrix for a generic element κ. For simplicity, the case studies here is a beam element with both translational and rotational DOFs The assembly of other structural elements is very similar to beam element assembly, as will be explained. The leading (or North West) 6 by 6 submatrix of the element mass matrix M^κ correspond to the DOFs of the first node of the element κ. Similarly, the trailing (or South East) 6 by 6 submatrix corresponds to the second node.

Equation (6.4.5) applies to the velocity $\boldsymbol{v}_i$ of node i of e in the inertial frame N. In general the subscript i refers to node i. In terms of the 6-vectors of DOFs v_i, $\dot{u}_i$, and u_i associated with $\boldsymbol{v}_i$, $\dot{\boldsymbol{u}}_i$, and $\boldsymbol{u}_i$,

$$v_i = \dot{u}_i + A u_i + b_i, \quad A = \begin{bmatrix} \bar{\Omega} & 0 \\ 0 & 0 \end{bmatrix}, \quad b_i = \begin{bmatrix} \omega \times x \\ \omega \end{bmatrix}.$$

There is an inconsistency here with the prior use of bold face type. In this section, only the vectors actually stored in a computer are not in bold face. One way to remember that the symbol A denotes the matrix representation of the cross product is that the Arabic word for cross product begins with an a. To express the right-hand sides in terms of ω, introduce

$$x_i = \begin{bmatrix} \boldsymbol{x}_i \\ 0 \end{bmatrix}, \quad w_i = \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \quad b_i = A x_i + w_i, \quad \dot{b}_i = \dot{A} x_i + \dot{w}_i. \tag{6.4.9}$$

A beam has $n_\kappa = 2$ nodes,

$$v_\kappa = \begin{bmatrix} v_1 \\ v_{n_\kappa} \end{bmatrix}, \quad u_\kappa = \begin{bmatrix} u_1 \\ u_{n_\kappa} \end{bmatrix}, \quad b_\kappa = \begin{bmatrix} b_1 \\ b_{n_\kappa} \end{bmatrix}.$$

The vectors w_κ and x_κ are defined analogously. For an element with an arbitrary number of nodes, v_κ has length $6n_\kappa$. In this section the symbol I_n denotes an n by n identity matrix,

$$A_\kappa = A \otimes I_{n_\kappa}, \quad v_\kappa = \dot{u}_\kappa + A_\kappa u_\kappa + b_\kappa.$$

With Lagrange's equation characterizing the state of physical systems including κ, in term of the kinetic energy T_κ,

$$\begin{aligned} 2T_\kappa &= \|v_\kappa\|^2_{M^\kappa} = \|\dot{u}_\kappa + A_\kappa u_\kappa + b_\kappa\|^2_{M^\kappa}, \\ \frac{d}{dt}\left(\frac{\partial T_\kappa}{\partial \dot{u}_\kappa}\right) &= M^\kappa(\ddot{u}_\kappa + \dot{A}_\kappa u_\kappa + A_\kappa \dot{u}_\kappa + \dot{b}_\kappa), \\ \frac{\partial T_\kappa}{\partial u_\kappa} &= A_\kappa^T M^\kappa(\dot{u}_\kappa + A_\kappa u_\kappa + b_\kappa). \end{aligned}$$

Subtracting,

$$\frac{d}{dt}\left(\frac{\partial T_\kappa}{\partial \dot{u}_\kappa}\right) - \frac{\partial T_\kappa}{\partial u_\kappa} = M^\kappa \ddot{u}_\kappa + (M^\kappa A_\kappa - A_\kappa^T M^\kappa)\dot{u}_\kappa + (M^\kappa \dot{A}_\kappa - A_\kappa^T M^\kappa A_\kappa)u_\kappa \\ + M^\kappa \dot{b}_\kappa - A_\kappa^T M^\kappa b_\kappa. \tag{6.4.10}$$

All that is left is to figure out how to interpret this equation. Due to equation (6.4.10),

$$G = M^\kappa A_\kappa - A_\kappa^T M^\kappa, \tag{6.4.11}$$

the centrifugal softening matrix $K_c = -A_\kappa^T M^\kappa A_\kappa$, and the acceleration matrix $K_a = M^\kappa \dot{A}_\kappa$. Substituting equation (6.4.9),

$$f_c = A_\kappa^T M^\kappa b_\kappa \qquad\qquad f_a = -M^\kappa \dot{b}_\kappa \tag{6.4.12}$$

$$= A_\kappa^T M^\kappa (A x_\kappa + w_\kappa) \qquad\qquad = -M^\kappa(\dot{A} x_\kappa + \dot{w}_\kappa) \tag{6.4.13}$$

The internal (strain) energy can be expressed as

$$U_\kappa = \frac{1}{2} u_\kappa^T (K^\kappa + K_g^\kappa) u_\kappa.$$

Ignoring external and damping forces for clarity, the equation of motion,

$$\frac{d}{dt}\left(\frac{\partial T_\kappa}{\partial \dot{u}_\kappa}\right) - \frac{\partial T_\kappa}{\partial u_\kappa} + \frac{\partial U_\kappa}{\partial u_\kappa} = 0,$$

reduces to equation (6.4.1) for a rotating structure.

6.4.1.2. Solid Element Assembly

In summary, contributions from angular acceleration to element stiffness matrices and load vectors are given by (6.4.14) and (6.4.15), respectively. Denote by I_k the k by k identity matrix. A solid element κ has 3 by 3 block diagonal

$$M = \int_V \phi_i \phi_j \rho dV \otimes I_3, \quad A = \bar{\Omega} \otimes I_{n_\kappa}.$$

The skew stiffness matrix contributions are due to bilinear forms resembling $\langle \boldsymbol{\omega} \times \boldsymbol{u}, v\rangle$. At a node, with vector of shape functions N and spin tensor $S(\omega)$, the corresponding matrix entries are $\langle N, SN\rangle$.

In terms of

$$\omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix},$$

you can formally write,

$$G = 2\begin{bmatrix} 0 & -\Omega_3 M & \Omega_2 M \\ \Omega_3 M & 0 & -\Omega_1 M \\ -\Omega_2 M & \Omega_1 M & 0 \end{bmatrix}.$$

We are assuming that M is block diagonal, with one block per DOF, which is true due to the connection between M and kinetic energy. However, using this expression to assemble G leaves many details to the reader.

Returning to the expression for acceleration in (6.4.6), notice that the term $2\boldsymbol{\Omega} \times \dot{\boldsymbol{u}}$ gave rise to the Coriolis matrix K_g. Notice also that the term $\dot{\boldsymbol{\Omega}} \times \boldsymbol{u}$ is obtained from $2\boldsymbol{\Omega} \times \dot{\boldsymbol{u}}$ simply by replacing 2 with 1, $\boldsymbol{\Omega}$ by $\dot{\boldsymbol{\Omega}}$ and $\dot{\boldsymbol{u}}$ by $\boldsymbol{u}$. Accordingly, the contribution from angular acceleration to the stiffness matrix is skew symmetric and given by

$$K_a = \begin{bmatrix} 0 & -\dot{\Omega}_3 M & \dot{\Omega}_2 M \\ \dot{\Omega}_3 M & 0 & -\dot{\Omega}_1 M \\ -\dot{\Omega}_2 M & \dot{\Omega}_1 M & 0 \end{bmatrix}. \tag{6.4.14}$$

The term $\dot{\boldsymbol{\Omega}} \times \boldsymbol{x}$ in (6.4.6) is associated in the weak formulation with

$$-\int_V \rho \boldsymbol{v} \cdot (\dot{\boldsymbol{\Omega}} \times \boldsymbol{x})\, dV,$$

where the minus sign originates from moving this term to the right hand side of the equilibrium equations. Notice the similarity of this term with the one in With the obvious modifications, the same module can assemble both G and K_a.

The implementation assembles matrices using the force, one column at a time First $\boldsymbol{u}$ and $\boldsymbol{x}$ are approximated using the same shape functions (isoparametric formulation). Second c_1, c_2 and c_3 are vectors of nodal coordinates in the corresponding directions. Third,

$$\omega = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix}$$

Angular acceleration contributes to the element load vector,

$$f_a = -K_a \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} \dot{\Omega}_3 M c_2 - \dot{\Omega}_2 M c_3 \\ \dot{\Omega}_1 M c_3 - \dot{\Omega}_3 M c_1 \\ \dot{\Omega}_2 M c_1 - \dot{\Omega}_1 M c_2 \end{bmatrix}, \tag{6.4.15}$$

A stiffness matrix can be expressed in terms of the mass matrix and the rotational velocity (or acceleration) by spelling out two details. Define $N_{m,i}$ to be the shape function at node m for the ith DOF. Recalling that Levi-Civita's symbol, $\epsilon_{i,j,k}$, is the sign of the permutation (i, j, k) if the indices are unique, vanishing otherwise,

$$K(n,i,m,k) = \sum_{j,k} \langle N_{n,i}, \epsilon_{i,j,k} \omega_j N_{m,k} \rangle =$$

$$\sum_{j,k} \epsilon_{i,j,k} \omega_j \langle N_{n,i}, N_{m,k} \rangle.$$

This is not the product of the mass matrix with any another matrix.

6.5. Random Pressure Loading

Input for random loads can be complicated. The most general type of input is the correlation matrix, which is the inverse Fourier transform of the spectral density matrix,[1] $S_{ij}(\omega)$.

$$c(\vec{x}_1, \vec{x}_2, t_1 - t_2) = E[P(\vec{x}_1, t_1) P(\vec{x}_2, t_2)] \tag{6.5.1}$$

[1] In the frequency domain we have the autospectral density matrix, and cross spectral density matrices which together form the spectral density matrix. It typically has units of $(PSI)^2/Hz$.

where $E[]$ is the expected value of the pressure at two locations on the surface at respective times.

This could be defined as a user defined function. In the most general case, that is the best means of a definition. However, defining that function is a real chore, and in many cases, the function can be more easily defined.

6.5.1. Specialization for Hypersonic Vehicles

Some simplifications can reduce the complexity of the correlation matrix. In the following paragraphs, we examine each of these, and arrive at a simplified parametric input for the correlation matrix.

Ergodic or Stationary Systems

Many variables change significantly during hypersonic flight. For example, the velocity of the body and the density of the air may depend on the portion of the trajectory. However, within limited time bounds of the trajectory, the system may be considered stationary. We represent this by writing the pressure as a product of a deterministic function and a stationary function of time and space.

$$P(\vec{x}, t) = \sigma(\vec{x}, t) Q(\vec{x}, t) \tag{6.5.2}$$

where, σ is a slowly varying, deterministic function, and Q contains all the random processes.

The pressure field applied to the hypersonic body is not stationary. One reason is the deceleration of the vehicle and the increase in dynamic pressure with time. However, we assume here that this non-stationary behavior can be modeled by $P = \sigma Q$, where Q is stationary and ergodic, and σ is a scaling or modulation function of time and space. This class of non-stationary model is called a modulated stationary process. Because Q is stationary, $E[Q(x_1, t_1)Q(x_2, t_2)]$ can be written as a function of $t_2 - t_1$, call it $\tau(t_2 - t_1)$. However, P is not stationary because $E[P(x_1, t_1)P(x_2, t_2)] = \sigma(x_1, t_1)\sigma(x_2, t_2)\tau(t_2 - t_1)$ cannot be written as a function only of $(t_2 - t_1)$; t_1 and t_2 appear in the σ terms.

This can simplify computation of the correlations of the pressure.

$$\begin{aligned} c(\vec{x}_1, \vec{x}_2, t_1, t_2) &= E[P(\vec{x}_1, t_1)P(\vec{x}_2, t_2)] && (6.5.3) \\ &= \sigma(\vec{x}_1, t_1)\sigma(\vec{x}_2, t_2)E[Q(\vec{x}_1, t_1)Q(\vec{x}_2, t_2)] && (6.5.4) \end{aligned}$$

Separation of spatial and temporal components

We may often separate the temporal and spatial components of the correlation function.

$$E[Q(\vec{x}_1, t_1)Q(\vec{x}_2, t_2)] = \pi(\vec{x}_1, \vec{x}_2)\, \tau(t_1, t_2) \tag{6.5.5}$$

Where $\pi(\vec{x}_1, \vec{x}_2)$ contains the spatial component of correlation, and $\tau(t_1, t_2)$ contains the temporal correlation.

Simplified Spatial Correlation

There is little data and few mathematical models of the spatial correlation of pressure on a body during hypersonic flight. A report by Corcos[38] is most commonly used. It describes the correlation variation as products of decaying exponentials. There is some evidence that the variables may be "self similar", at least in the flow direction, so the decay constants are scalable with the frequency and velocity. The self-similar properties are less well-established in the transverse directions.[42] The spatial component of correlation may be written as,

$$\pi(\vec{x}_1, \vec{x}_2) = \exp(-\alpha_z \Delta z) \exp(-\beta_t \Delta y) \tag{6.5.6}$$

In this expression, the spatial correlation terms depend on the separation in the stream (or flow) direction, Δz, and on the transverse separation, Δy.

Simplified Temporal Correlations

Aerodynamic models that predict the pressure power spectral density (PSD) on the surface of a hypersonic body are still under development. Many of these models predict a PSD that is only a weak function of the axial location. Thus, the PSD at the back of the body is a scaled version of those at the front. Further, with high velocities, the PSD is flat within the band of interest. Thus, the PSD may be represented as a product of a deterministic function of z and a single PSD. The correlations reflect this same product, and the deterministic function $\sigma()$ can be employed to carry this scaling. If the PSD is flat over the bandwidth, the temporal correlation may be further simplified. We may then write,

$$\tau(t_1, t_2) = \frac{\sin(\omega_c(t_1 - t_2))}{\omega_c(t_1 - t_2)} \tag{6.5.7}$$

where we use the fact that the Fourier transform of a constant frequency response with cutoff frequency ω_c is a $\sin(x)/x$.[2]

Temporal Interpolation and Filtering

As noted above, we have an assumption that there is a cutoff frequency. Anything above that frequency is out of band of the analysis, and can (should) be filtered. Equivalently, time steps less than $T = \pi/\omega_c$ should also be filtered. One way to approach this is to sample at an interval T, and interpolate using a $\sin(x)/x$ type filter as described below. Note that in addition to the benefit of filtering, sampling at an interval, T, can reduce the amount of memory used to store the temporal correlation.

Let $[-\nu^*, \nu^*],\ 0 < \nu^* < \omega_c$, be the frequency band of a deterministic function, $x(t)$, $-\infty < t < \infty$. Then,

$$x(t) = \lim_{n\to\infty} \sum_{k=-n}^{n} x(kT)\alpha_k(t, T) \tag{6.5.8}$$

where

$$\begin{aligned} \alpha_k(t, T) &= \frac{\sin[\pi(t/T - k)]}{\pi(t/T - k)} && (6.5.9) \\ &= \frac{\sin[\frac{\pi}{T}(t - kT)]}{\frac{\pi}{T}(t - kT)} && (6.5.10) \end{aligned}$$

[2]While a flat response results in a $\sin(x)/x$, which is the default, many PSD responses are *not* flat, so a user defined temporal function may be required.

"It is sufficient to know the values $x(kT)$, with k = ...,-2, -1, 0, 1, 2, ... to reconstruct the entire signal $x(t)$, $-\infty < t < \infty$."

Note:

$$\alpha_k = 1 \qquad \text{if } \frac{t}{T} = k \tag{6.5.11}$$

$$\alpha_k = 0 \qquad \text{if } \frac{t}{T} \text{ any other integer} \tag{6.5.12}$$

$$|\alpha_k| \qquad \text{decreases to zero as } \left|\frac{t}{T} - k\right| \text{ increases.} \tag{6.5.13}$$

Advancing the Coarse Temporal Solution

The strategy described involves computation of the solution on a coarse temporal grid, with interpolation to a fine time step as described above. The process for advancing the coarse time solution is described here.

The initial coarse solution, $Y(x,T)$, is given by the solution to the Cholesky factor of the correlation matrix.

$$Y = chol(\tilde{c})W \tag{6.5.14}$$

where

$\tilde{c}$	is the $d(2n+1)$ x $d(2n+1)$ correlation matrix
W	is a vector of zero mean, unit variance random variables, and
Y	is the properly correlated solution vector at the $2n+1$ coarse time values, 0, T, $2T$, ..., $(2n+1)T$ and the d sample locations.

6.5.1.1. Temporal Advancement

As described in texts on stochastic calculus (see[70] for example), we can compute the response of a Gaussian random vector when a portion of the vector is known. Consider a random vector Y, which is partitioned into a known part, $Y^{(1)}$, and a portion to be determined, $Y^{(2)}$. We may write, (see equation 2.109 of [[70]]),

$$\xi = (Y^{(2)}|Y^{(1)} = z) \tag{6.5.15}$$

$$\sim N(\hat{\mu}, \hat{c}) \tag{6.5.16}$$

where,

$$\hat{\mu} = \mu^{(2)} + c^{(2,1)}[c^{(1,1)}]^{-1}(z - \mu^{(1)}) \tag{6.5.17}$$

$$\hat{c} = c^{(2,2)} - c^{(2,1)}[c^{(1,1)}]^{-1}c^{(1,2)} \tag{6.5.18}$$

and $\mu^{(i)}$ is the mean on each portion of the solution.

In words, we can express the normal distribution of the unknown vector as a random distribution with mean $\hat{\mu}$ and variance given by the covariance matrix $\hat{c}$. The covariance does not depend on the previous samples but only on the partition of the original covariance matrix. The mean depends weakly on the previous sample, z.

The matrix c is partitioned as follows.

$c^{(1,1)}$ is $\tilde{c}$, the original correlation matrix. It is a square matrix of dimension $d(2n+1)$.

$c^{(2,2)}$ is the d x d correlation matrix associated with zero time lag.

$c^{(2,1)}$ is an additional set of d rows of the correlation matrix associated with the time lag $(2n+2)T$.

$$c = \left[\begin{array}{cccc|c} C(0) & C(T) & C(2T) & \ldots & C((2n+2)T) \\ C(T) & C(0) & C(T) & \ldots & C((2n+1)T) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \hline \\ C((2n+2)T) & C((2n+1)T) & C(2nT) & \ldots & C(0) \end{array}\right]$$

and $C(T)$ is the d x d correlation matrix evaluated on the d spatial points at time lag T.

6.5.1.2. Procedure

The solution is advanced as follows.

1. We augment the system to have $d(2n+2)$ equations. Thus, $c^{(1,1)}$ is the $d(2n+1)$ covariance previously calculated.
2. We use $b = chol(c^{(1,1)})$ to compute the desired mean of the new distribution. Specifically,

$$\begin{aligned} \hat{\mu} &= \mu^{(2)} + c^{(2,1)}(b^t b)^{-1}(z - \mu^{(1)}) && (6.5.19) \\ &= c^{(2,1)}(b^t b)^{-1} z && (6.5.20) \\ &= gz && (6.5.21) \end{aligned}$$

   where we have used the fact that both $\mu(1)$ and $\mu(2)$ are zero. We store the rectangular matrix $g = c^{(2,1)}(b^t b)^{-1}$. We no longer need the original covariance matrix $\tilde{c}$, nor its factor, b.
3. We reuse g to compute the revised correlation matrix.

$$\begin{aligned} \hat{c} &= c^{(2,2)} - c^{(2,1)}[c^{(1,1)}]^{-1} c^{(1,2)} && (6.5.22) \\ &= C(0) - gc^{(1,2)} && (6.5.23) \end{aligned}$$

   where $C(0)$ is the d x d correlation matrix for a time lag of zero. The matrix $\hat{c}$ is also d x d.
4. We perform a Cholesky factor on $\hat{c}$. This is the second such factor, and it is performed on a smaller space. It need be performed only on the first advancement as $\hat{c}$ is a constant.

$$\hat{b} = chol(\hat{c}) \qquad (6.5.24)$$

5. Compute the new distribution.

$$\begin{aligned} \xi &= \mathcal{N}(\hat{\mu}, \hat{c}) && (6.5.25) \\ &= \hat{\mu} + chol(\hat{c}) w && (6.5.26) \\ &= \hat{\mu} + \hat{b} w && (6.5.27) \end{aligned}$$

   where w is a zero mean, unit normal Gaussian basis.
6. Move solution vector solution, Y, up by one, and insert ξ in the new locations.

6.6. Removing Net Torques from Applied Loads

For structures without any connections to ground, there are six rigid body modes. Three modes correspond to rigid body translations, while the remaining three are for rigid body rotation about the center of mass of the structure. If the applied loads have a net torque about the center of mass, then we should expect the structure to eventually begin tumbling as time progresses. If the net torque vanishes, then the small strain approximation used in **Sierra/SD** is accurate since rotational deformations should remain small. This expectation holds even in the presence of large displacements caused by loads with significant translational rigid body components.

The purpose of these notes is to describe options for removing net torques from applied loads to avoid tumbling in **Sierra/SD** during transient analyses. One option assumes that the center of mass is known, while the second makes use of the mass matrix for the system finite element model. We note that net translational loads are not removed using either of these options. Only the mass matrix option is used in **Sierra/SD**.

Use of Mass Matrix. Let M and K denote the mass and stiffness matrices for the structure. Further, let Φ_{tran} and Φ_{rot} contain the translational and rotational rigid body modes. Both Φ_{tran} and Φ_{rot} have 3 columns, and for floating structures $K\Phi_{tran} = K\Phi_{rot} = 0$. We will assume the mass matrix M is symmetric and positive definite, while the stiffness matrix is assumed to be symmetric and have 6 rigid body modes as stated. Further, we assume for the damping matrix C that $C\Phi_{rbm} = 0$ and $\Phi_{rbm}^T C = 0$, where $\Phi_{rbm} = \begin{bmatrix} \Phi_{tran} & \Phi_{rot} \end{bmatrix}$. If rigid body motion of the structure does not cause any damping forces, then this assumption holds. One instance where this assumption on C does not hold is for models with mass proportional damping.

Consider a node i of the model that has both translational and rotational degrees of freedom. The rows of Φ_{rbm} associated with this node are given by

$$\Phi_{rbm}^i = \begin{bmatrix} 1 & 0 & 0 & 0 & r_{i3} & -r_{i2} \\ 0 & 1 & 0 & -r_{i3} & 0 & r_{i1} \\ 0 & 0 & 1 & r_{i2} & -r_{i1} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.6.1}$$

where $\boldsymbol{r}_i = r_{i1}\boldsymbol{e}_1 + r_{i2}\boldsymbol{e}_2 + r_{i3}\boldsymbol{e}_3$ is the position vector of node i in the global coordinate system. Note here that the origin for $\boldsymbol{r}_i$ is the origin of the global coordinate system and does not necessarily coincide with the center of mass of the system.

Sierra/SD mass orthonormalizes the rigid body modes. Namely,

$$\Phi_{rbm}^T M \Phi_{rbm} = I, \tag{6.6.2}$$

where I is the identity matrix (notice this equation also implies $\Phi_{rot}^T M \Phi_{rot} = I$). Moreover, the columns of Φ_{rbm} are orthonormalized from the leftmost column to the right so that the rigid body translational modes remain in the first three columns of Φ_{rbm}. Φ_{rot} is the mass-orthonormalized rigid body mode matrix for rotations.

The standard equations of motion can be expressed as

$$M\ddot{u} + C\dot{u} + Ku = f, \tag{6.6.3}$$

where u and f are the displacement and applied force vectors. Next, consider the approximation $u = \Phi_{rbm} q$, where q is a 6x1 vector. Substituting $u = \Phi_{rbm} q$ into (6.6.3) and pre-multiplying by Φ_{rbm}^T, it follows from (6.6.2) and the assumptions $K\Phi_{rbm} = 0$ and $C\Phi_{rbm} = 0$ that

$$\ddot{q} = \Phi_{rbm}^T f, \tag{6.6.4}$$

or, equivalently,

$$\ddot{q}_{tran} = \Phi_{tran}^T f, \tag{6.6.5}$$
$$\ddot{q}_{rot} = \Phi_{rot}^T f. \tag{6.6.6}$$

Notice from (6.6.6) that there will be rigid body rotational accelerations if $\Phi_{rot}^T f \neq 0$. We will consider a modified force vector of the form

$$\tilde{f} = f - M\Phi_{rot} s, \tag{6.6.7}$$

where s is a 3x1 vector to be determined from the condition

$$\Phi_{rot}^T \tilde{f} = 0. \tag{6.6.8}$$

Substitution of (6.6.7) into (6.6.8) and use of $\Phi_{rot}^T M \Phi_{rot} = I$ then gives us

$$s = \Phi_{rot}^T f, \tag{6.6.9}$$

and (6.6.7) then reads

$$\tilde{f} = f - M\Phi_{rot}(\Phi_{rot}^T f). \tag{6.6.10}$$

Examination of Flexible Modes

By pre-multiplying (6.6.10) by Φ_{rot}^T and using $\Phi_{rot}^T M \Phi_{rot} = I$ once again, one can confirm that $\Phi_{rot}^T \tilde{f} = 0$ as required to avoid rigid body rotational accelerations.

Let Φ_{flex} denote the mode shape matrix for the undamped flexible modes. The mode shape matrix for all the modes can be written as $\Phi = \begin{bmatrix} \Phi_{tran} & \Phi_{rot} & \Phi_{flex} \end{bmatrix}$. Notice since both $\Phi^T M \Phi$ and $\Phi^T K \Phi$ are diagonal, it follows that $\Phi_{flex}^T M \Phi_{rot} = 0$.

The generalized force associated with the flexible modes is given by

$$f_{flex} = \Phi_{flex}^T f. \tag{6.6.11}$$

Since $\Phi_{flex}^T M \Phi_{rot} = 0$, we then find

$$\begin{aligned} \tilde{f}_{flex} &= \Phi_{flex}^T f - \Phi_{flex}^T M \Phi_{rot}(\Phi_{rot}^T f) \\ &= f_{flex}. \end{aligned} \tag{6.6.12}$$

Thus, the generalized force vector $\tilde{f}_{flex}$ for the modified force vector is identical to the original one f_{flex}. This implies that the adjustments made to the original force vector do not modify the flexible response. This is a nice feature.

Parallelization Issues

When the model is decomposed by element[3] the mass matrix provides requisite information about duplication of nodal quantities on the boundaries. Thus, nodal quantities (which are replicated on subdomains which share a boundary) are only counted once in a dot product. However, for statics, there is no mass matrix, and the identity is substituted for the mass matrix. While the system matrix is the identity, the appropriate submatrix of the identity on each subdomain is *not* a subdomain identity matrix. It is a diagonal matrix with entries,

$$\tilde{I}^{sub}_{jj} = 1/\text{cardinality}_{node_j}$$

This definition of the subdomain identity submatrix, I^{sub} permits multiplication without duplication of values on the subdomain boundary. This submatrix must be used for orthogonalization and for the force correction (equation 6.6.10).

Filter of Output Displacements

The mass matrix also provides stabilization of the solution matrix. For statics solutions on floating structures, the solution matrix is the stiffness matrix, which is singular. Additional tools are in place to help the linear solver with this challenge. In particular, GDSW (see e.g.[47]) may solve such systems provided that the dimension of the null space is provided. However, small non-equilibrated forces or round off in the solver can still result in solution vectors in the range of the null space. For statics, these displacement vectors are also filtered to eliminate the rigid body component. The filtering uses equation 6.6.10, with the identity matrix replacing the mass matrix.

6.7. Traction Loads

In the traction loading of a side set, if the user specified coordinate frame C_u with basis

$$(\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{e}}_3)$$

is specified with the traction vector, it is used to determine the directions of application of the loads so that the third component remains the element normal vector, $\hat{\boldsymbol{n}}$.

Loads are applied in the projected coordinate frame C_p with basis

$$(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2, \hat{\boldsymbol{n}})$$

determined using the normal,

$$\hat{\boldsymbol{p}}_1 = \hat{\boldsymbol{e}}_2 \times \hat{\boldsymbol{n}}\ \rho_1, \qquad \hat{\boldsymbol{p}}_2 = \hat{\boldsymbol{n}} \times \hat{\boldsymbol{p}}_1\ \rho_2.$$

Here ρ_i are positive scalar normalization terms. The event $\hat{\boldsymbol{e}}_2 \times \hat{\boldsymbol{n}} = 0$ is handled by substituting $\hat{\boldsymbol{p}}_1 = \hat{\boldsymbol{e}}_1 \times \hat{\boldsymbol{n}}\rho_1$ and $\hat{\boldsymbol{p}}_2 = \hat{\boldsymbol{n}} \times \hat{\boldsymbol{p}}_1\ \rho_2$.

The direction in which forces will be applied depends on the coordinate systems. In particular side sets will need to be chosen (or subdivided) to ensure that $\hat{\boldsymbol{e}}_2 \times \hat{\boldsymbol{n}} \neq 0$.

In a cartesian coordinate frame, element normal vectors for tractions should not be aligned with the y direction of the applicable coordinate frame. In the cylindrical frame $(\boldsymbol{r}, \boldsymbol{\theta}, \boldsymbol{z})$ or a spherical coordinate frame $(\boldsymbol{r}, \boldsymbol{\theta}, \boldsymbol{\phi})$, element normal vectors aligned with the azimuthal direction are problematic.

[3]each element is one one subdomain.

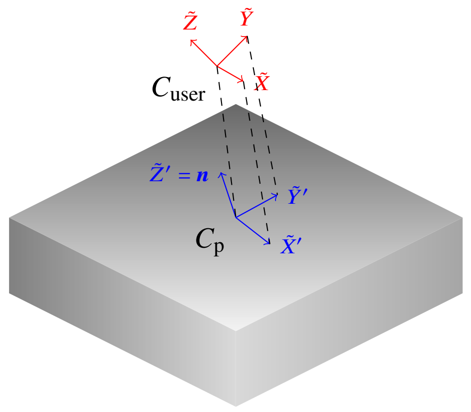


Figure 6-7. – Coordinate Frame Projection for Tractions

6.8. Consistent Loads Calculations

Starting with equation 4.1-6 from *Concepts and Applications of Finite Element Analysis* by Cook *et al.*[37],

$$\{r_e\} = \int_{V_e} [B]^T [E]\{\epsilon_0\} dV - \int_{V_e} [B]^T \{\sigma_0\} dV + \int_{V_e} [N]^T \{F\} dV + \int_{S_e} [N]^T \{\Phi\} dS \tag{6.8.1}$$

where each term is defined in Subsection 4.1 of the above mentioned reference. The load vector, $\{r_e\}$, is composed of four parts in equation 6.8.1. In this document, only the last part, which is the contribution of the surface tractions to the load vector, will be considered. Rewriting,

$$\{r_e\} = \int_{S_e} [N]^T \{\Phi\} dS \tag{6.8.2}$$

Here, the integral is calculated over the surface of the element on which the surface traction, $\{\Phi\}$, is applied. Therefore,

$$\{\Phi\} = [\Phi_x \; \Phi_y \; \Phi_z]^T \tag{6.8.3}$$

and $[N]$ is the shape function matrix of the element on which the surface tractions, $\{\Phi\}$, are applied. To generate a model for application inn **Sierra/SD**, $\{\Phi\}$ can be generated within PATRAN or other preprocessors by applying a spatial field to a specified side set. In **Sierra/SD** however, these spatial field values are available only on the surface nodes of the element. Using the nodal values of this surface traction, the value at any surface location must be determined using an interpolation function over the surface or side of the element. Since only one value per node may be specified on the side set in **Sierra/SD**, a surface traction may be applied only in one direction at a time. Therefore, $\{\Phi\}$ will be defined as,

$$\{\Phi\} = \begin{Bmatrix} n_x \\ n_y \\ n_z \end{Bmatrix} \Phi(x, y, z) \tag{6.8.4}$$

6.8.1. Elements with consistent loads

Consistent loads are implemented for the following 3-D and 2-D elements:

- Hex8, Hex20, Tet4, Tet10, Wedge6
- Tria3, TriaShell ,Tria6 (four Tria3s)
- QuadT (two Tria3s), Quad8T (1 QuadT and 4 Tria3s)

6.8.2. Pressure Loading

Here, we will consider only pressure loads on 3-D elements, such that

$$\{\Phi\} = \begin{Bmatrix} n_x \\ n_y \\ n_z \end{Bmatrix} \Phi(x, y, z) \tag{6.8.5}$$

where $[n_x, n_y, n_z]^T$ is the normal to the element face. Hence, the consistent loads can be calculated as,

$$\{r_e\} = \int_{S_e} [N]^T \{\Phi\} dS = \int_{S_e} [N]^T \Phi(x, y, z)(\vec{a} \times \vec{b}) dS_e \tag{6.8.6}$$

Here,

$$\vec{a} = [\frac{\partial x}{\partial r}, \frac{\partial y}{\partial r}, \frac{\partial z}{\partial r}]^T \tag{6.8.7}$$

$$\vec{b} = [\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s}]^T \tag{6.8.8}$$

where Φ is the pressure load, and (x, y, z) are the physical coordinate directions, and (r, s) are the local element directions for the face of the element. The normal may be obtained by taking the cross-product of $\vec{a}$ and $\vec{b}$.

6.8.3. Shape Functions for Calculating Consistent Loads

For 3-D elements, all the faces are either quadrilateral or triangular shaped. Hence, shape functions for quads and triangles could be used to evaluate the consistent loads. However, application of the shape functions for the 3-D elements, reduces code and “fits” better into the current finite element class structure. This is what is currently implemented. This requires a “mapping” of the 3-D elements’ faces to a 2-D plane. The additional overhead for using the 3-D elements is that each face of the element must have this “mapping” which states how the elements’ 3-D shape functions map to a 2-D element. For example, for a Hex20, the element coordinates (η_1, η_2, η_3) are defined in a particular way. For each face of the Hex20, defined by a side id, the face has a local coordinate system (r, s). The “mapping” defines how (r, s) are related to (η_1, η_2, η_3). This also helps define how 2-D Gauss points are mapped to the 3-D face. These mappings are available for all the linear and quadratic 3-D elements.

6.8.4. Shell Elements - consistent loads

All the 2-D elements (shell elements) compute loads based on the Tria3 shape functions. The consistent loads calculations for the Tria3 can be “copied” to the TriaShell. This way all the shell elements use the same consistent loads implementation. Since Carlos Felippa designed the Tria3, his consistent loads implementation is used. The portion for linearly varying pressure loads is shown here. If the loads are aligned along an edge, $\{q\}$, they need to be decomposed into (qs, qn, qt). Where (s, n, t) are coordinate directions along the element edge. Coordinate s varies along the element edge tangentially, n is normal to the element edge, and t is tangent to the element edge in the transverse direction, i.e., in the direction of the thickness. Once, the edge load is decomposed, the equations for consistent loads are,

$$f^1{}_s = \frac{1}{20}(7q_{s1} + 3q_{s2})L_{21} \qquad f^2{}_s = \frac{1}{20}(3q_{s1} + 7q_{s2})L_{21} \tag{6.8.9}$$

$$f^1{}_n = \frac{1}{20}(7q_{n1} + 3q_{n2})L_{21} \qquad f^2{}_n = \frac{1}{20}(3q_{n1} + 7q_{n2})L_{21} \tag{6.8.10}$$

$$f^1{}_t = \frac{1}{20}(7q_{t1} + 3q_{t2})L_{21} \qquad f^2{}_t = \frac{1}{20}(3q_{t1} + 7q_{t2})L_{21} \tag{6.8.11}$$

$$m^1{}_s = m^2{}_s = 0 \tag{6.8.12}$$

$$m^1{}_n = -\frac{1}{60}(3q_{t1} + 2q_{t2})L^2{}_{21} \qquad m^2{}_n = \frac{1}{60}(2q_{t1} + 3q_{t2})L^2{}_{21} \tag{6.8.13}$$

$$m^1{}_t = -\frac{1}{40}(3q_{n1} + 2q_{n2})L^2{}_{21} \qquad m^2{}_t = \frac{1}{40}(2q_{n1} + 3q_{n2})L^2{}_{21} \tag{6.8.14}$$

where q_{s1} is the value of q in the s direction at node 1 of the edge, L_{12} is the length of the edge. The superscripts 1,2 are the node numbers of the edge. Note, it is assumed here that the load q is per unit length, but this is not assumed when creating the sideset in PATRAN for example. Therefore, this distributed load is multiplied, in **Sierra/SD**, by the thickness of the triangle.

For a pressure load on the face of the Tria3, the equations become,

$$f^1{}_x = f^1{}_y = m^1{}_z = f^2{}_x = f^2{}_y = m^2{}_z = f^3{}_x = f^3{}_y = m^3{}_z = 0 \quad (6.8.15)$$

$$f^1{}_z = \left(\frac{8}{45}p_1 + \frac{7}{90}p_2 + \frac{7}{90}p_3\right)A \quad (6.8.16)$$

$$f^2{}_z = \left(\frac{7}{90}p_1 + \frac{8}{45}p_2 + \frac{7}{90}p_3\right)A \quad (6.8.17)$$

$$f^3{}_z = \left(\frac{7}{90}p_1 + \frac{7}{90}p_2 + \frac{8}{45}p_3\right)A \quad (6.8.18)$$

$$m^1{}_x = \frac{A}{360}[7(y_{31} + y_{21})p_1 + (3y_{31} + 5y_{21})p_2 + (5y_{31} + 3y_{21})p_3] \quad (6.8.19)$$

$$m^1{}_y = \frac{A}{360}[7(x_{13} + x_{12})p_1 + (3x_{13} + 5x_{12})p_2 + (5x_{13} + 3x_{12})p_3] \quad (6.8.20)$$

$$m^2{}_x = \frac{A}{360}[(5y_{12} + 3y_{32})p_1 + 7(y_{12} + y_{32})p_2 + (3y_{12} + 5y_{32})p_3] \quad (6.8.21)$$

$$m^2{}_y = \frac{A}{360}[(5x_{21} + 3x_{23})p_1 + 7(x_{21} + x_{23})p_2 + (3x_{21} + 5x_{23})p_3] \quad (6.8.22)$$

$$m^3{}_x = \frac{A}{360}[(3y_{23} + 5y_{13})p_1 + (5y_{23} + 3y_{13})p_2 + 7(y_{23} + y_{13})p_3] \quad (6.8.23)$$

$$m^3{}_x = \frac{A}{360}[(3x_{32} + 5x_{31})p_1 + (5x_{32} + 3x_{31})p_2 + 7(x_{32} + x_{31})p_3] \quad (6.8.24)$$

where $y_{ij} = y_i - y_j$ and $x_{ij} = x_i - x_j$, A is the area of the triangle, p_i is the value of the pressure load at node i, and (x_i, y_i) are coordinates of the triangle in 2-D space.

Finally, the “pseudo” elements (QuadT, Quad8T, Tria6) created by using triangles require overhead. For example, the Quad8T is composed of 1 QuadT and 4 Tria3s. However, since it is defined as a Quad8T, it has distribution factors at its 8 nodes, and these distribution factors have to be mapped to the 1 QuadT and the 4 Tria3s. The number of distribution factors is 3 however, if the load is applied to its edge. Therefore, this extra coding can be seen in the ElemLoad method of the shells’ classes.

6.9. Solution of Singular Linear Systems

It may be required on occasion to solve problems with singular coefficient matrices. For example, the static analysis of a structure that has no essential boundary conditions (free-free) will typically have six rigid body modes and the stiffness matrix is singular. In this subsection, we describe how singular linear systems are handled by the GDSW solver and also provide supporting theory. The development below is for serial runs, but the same approach is applied to the singular linear system associated with the coarse problem for multi-processor runs.

Consider a structure with a symmetric and positive semi-definite stiffness matrix K. The columns of the matrix Q span the null space of K. That is, $KQ = 0$ and $Q^T Q = I$, where I is an identity matrix. For example, Q can be obtained from Gram-Schmidt orthogonalization of the geometric rigid body modes.

We are interested in solving linear systems of the form

$$Ku = f. \quad (6.9.1)$$

Since K is singular, we must have $Q^T f = 0$ for a solution of (6.9.1) to exist. In other words, the force vector must be orthogonal to the rigid body modes. We may perform a simple Gaussian elimination process with row pivoting on the matrix Q to identify a set of linearly independent set of rows of Q. Without loss of generality, let Q_2 denote these rows of Q and let us express Q as

$$Q = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix},$$

where Q_2 is square and nonsingular by construction. Similarly, we express the stiffness matrix as

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}.$$

Our first step is to show that K_{11} is positive definite. To this end, consider a vector v of the form

$$v = \begin{bmatrix} v_1 \\ 0 \end{bmatrix},$$

where $v_1 \neq 0$. We may express v as

$$v = Qq + Q_\perp q_\perp,$$

where q and $q_\perp$ are vectors, $Q_\perp^T Q_\perp = I$ and $Q_\perp^T Q = 0$. Notice if $q = 0$, then $q_\perp \neq 0$ since $v_1 \neq 0$. Likewise, if $q \neq 0$, then we have from the lower block of the expression for v that

$$0 = Q_2 q + Q_{\perp 2} q_\perp.$$

Since Q_2 is nonsingular and $q \neq 0$, it follows that $q_\perp \neq 0$. Thus, in both cases we have $q_\perp \neq 0$ which implies $v_\perp = Q_\perp q_\perp \neq 0$. Consequently, since $v_\perp^T K v_\perp > 0$ for all $v_\perp = Q_\perp q_\perp \neq 0$, we have

$$v_1^T K_{11} v_1 = v^T K v = v_\perp^T K v_\perp > 0.$$

In other words, K_{11} is positive definite and thus nonsingular.

The following procedure is used in GDSW for solving (6.9.1) for serial runs. The same approach for multi-processor runs applies to the singular linear system for the coarse problem.

1. Make sure f is orthogonal to Q by calculating $f = f - Q(Q^T f)$.
2. Solve the linear system
$$\begin{bmatrix} K_{11} & 0 \\ 0 & I \end{bmatrix} \underbrace{\begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix}}_{\tilde{u}} = \begin{bmatrix} f_1 \\ 0 \end{bmatrix}.$$
3. Remove any null space component by calculating $u = \tilde{u} - Q(Q^T \tilde{u})$.

We next verify that the solution from this procedure satisfies (6.9.1). Notice from Step 2 that $\tilde{u}_2 = 0$ and

$$K_{11}\tilde{u}_1 + K_{12}\tilde{u}_2 = f_1. \tag{6.9.2}$$

The first block of equations in $KQ = 0$ reads as $K_{11}Q_1 + K_{12}Q_2 = 0$, which gives

$$K_{11}^{-1} K_{12} = -Q_1 Q_2^{-1}.$$

Since $Q^T f = 0$, we also have

$$Q_1^T f_1 + Q_2^T f_2 = 0.$$

From the previous two expressions it follows that

$$\begin{aligned} K_{21}K_{11}^{-1} f_1 &= -Q_2^{-T} Q_1^T f_1 \\ &= -Q_2^{-T}(-Q_2^T f_2) = f_2 \end{aligned}$$

It then follows from the previous equation and Step 2 that

$$K_{21}\tilde{u}_1 + K_{22}\tilde{u}_2 = K_{21}K_{11}^{-1} f_1 = f_2 \tag{6.9.3}$$

In summary, (6.9.2) and (6.9.3) verify that the $\tilde{u}$ calculated from the procedure satisfies $K\tilde{u} = f$. The final step of the procedure removes any null space component from $\tilde{u}$, and we can verify

$$Ku = K(\tilde{u} - Q(Q^T\tilde{u})) = K\tilde{u} = f$$

and

$$Q^T u = Q^T(\tilde{u} - Q(Q^T\tilde{u})) = Q^T\tilde{u} - Q^T\tilde{u} = 0.$$

7. CONTACT

7.1. Multipoint Constraints

User's Manual describes *MPC*s. Here coordinate system dependencies are discussed.

*MPC*s may be defined in any coordinate system. However, all nodes in the *MPC*s are defined in the same system. This is done for convenience in parsing, and not for any fundamental reason. Consider a constraint equation where each entry in the equation could be specified in a different coordinate system.

$$\sum_i C_i u_i^{(k_i)} = 0$$

where C_i is a real coefficient, and $u_i^{(k_i)}$ represents the displacement of degree of freedom i in degree of coordinate system k_i. We can transform to the basic coordinate system using $u_i^{(k_i)} = \sum_j R_{ji}^{(k_i)} u_j^{(0)}$, where $R^{(k_i)}$ is the rotation matrix for coordinate system k_i. Then we may write,

$$\sum_{i,j} C_i R_{ji}^{(k_i)} u_j^{(0)} = 0$$

or,

$$\sum_i C_i^{(k_i)} u_i^{(0)} = 0$$

where $C_i^{(k_i)} = \sum_j R_{ij}^{(k_i)} C_j$. Note however, that in this analysis, we have assumed that the dimension of C is 3. Thus, rotation into the basic frame will likely increase the number of coefficients.

Sierra/SD is designed to support constraints through at least two methods. These include a constraint transform method and Lagrange multipliers. Lagrange multiplier methods are used for all the parallel solvers. The serial solver uses constraint transform methods.

7.2. Constraint Transformations in General Coordinate Systems

In general, constraint equations can be applied in any coordinate system. We here describe the transformation equations and implications for general constraints in any coordinate system. The implications of this use in **Sierra/SD** are also outlined.

Consider a constraint equation,

$$C'u' = Q \tag{7.2.1}$$

where the primes indicate a generalized coordinate frame. The frame may be transformed to the basic coordinate system using equation 1.4.1, and

$$u' = Ru \tag{7.2.2}$$

Rewrite equation 7.2.1 as

$$\begin{aligned} C'Ru &= Q \\ Cu &= Q \end{aligned} \tag{7.2.3}$$

where $C = C'R$.

Thus, a general system of constraint equations may be easily transformed to the basic system. Further, the rotational matrix is a 3x3 matrix which may be applied to each node's degrees of freedom separately.

7.2.1. Decoupling Constraint Equations

We still have a coupled system of equations. We partition the space into constrained and retained degrees of freedom, and describe the constrained dofs in terms of its Schur complement.

$$u = \begin{bmatrix} u_r \\ u_c \end{bmatrix} \tag{7.2.4}$$

The whole constraint equation may be similarly partitioned.

$$\begin{bmatrix} C_r & C_c \end{bmatrix} \begin{bmatrix} u_r \\ u_c \end{bmatrix} = [Q] \tag{7.2.5}$$

Note that C_r is an cxr matrix, C_c is cxc, and Q is a vector of length c. Under most conditions Q is null.

This may be solved for u_c,

$$u_c = C_c^{-1}Q - C_c^{-1}C_r u_r \tag{7.2.6}$$

We must be concerned with cases where C_c may be either singular or over constrained. The former case occurs if we try to eliminate c equations, but the rank of C is less than c. This could occur if the equations are redundant. We can over constrain the system only if Q is nonzero. Both these situations need attention, but both can be dealt with.

We may also write the solution using a transformation matrix, T.

$$\begin{bmatrix} u_r \\ u_c \end{bmatrix} = [T]\,[u_r] + \tilde{Q} \tag{7.2.7}$$

where

$$T = \begin{bmatrix} 1 \\ C_{rc} \end{bmatrix} \tag{7.2.8}$$

$$C_{rc} = -C_c^{-1}C_r \tag{7.2.9}$$

and

$$\tilde{Q} = \begin{bmatrix} 0 \\ C_c^{-1}Q \end{bmatrix} = \begin{bmatrix} 0 \\ \breve{Q} \end{bmatrix} \tag{7.2.10}$$

7.2.2. Transformation of Stiffness Matrix

We assume a similar partition of the stiffness matrix. The equations for statics are then,

$$\begin{bmatrix} K_{rr} & K_{rc} \\ K_{cr} & K_{cc} \end{bmatrix} \begin{bmatrix} u_r \\ u_c \end{bmatrix} = \begin{bmatrix} R_r \\ R_c \end{bmatrix} \tag{7.2.11}$$

or,

$$[K]\,[T]\,u_r + [K]\,\left[\tilde{Q}\right] = R \tag{7.2.12}$$

and

$$T^T K T u_r = T^T \left\{R - K\tilde{Q}\right\} = \tilde{R} \tag{7.2.13}$$

We can define the reduced equations,

$$\tilde{K} = T^T K T = K_{rr} + K_{rc}C_{rc} + C_{rc}^T K_{cr} + C_{rc}^T K_{cc} C_{rc} \tag{7.2.14}$$

and,

$$\begin{aligned} \tilde{R} &= T^T R - T^T \begin{bmatrix} K_{rc}\breve{Q} \\ K_{cc}\breve{Q} \end{bmatrix} \\ &= R_r + C_{rc}^T R_c - K_{rc}\breve{Q} - C_{rc}^T K_{cc}\breve{Q} \end{aligned} \tag{7.2.15}$$

The solution in the retained system is

$$\tilde{K} u_r = \tilde{R} \tag{7.2.16}$$

The system may be solved using the reduced equations, and the constrained degrees of freedom may be solved using equation 7.2.6. Much of this is detailed in Cook, but without the constrained right-hand side.

For eigendecomposition of the mass matrix may be transformed like the stiffness matrix in equation 7.2.14. There is no force vector.

For transient dynamics the mass and stiffness matrix transform the same. The force vector and force vector corrections may be time dependent. There is currently no structure to store these time dependent terms in **Sierra/SD**.

7.2.3. Application to single point constraints

Our initial efforts at applying single point constraints (SPC) has been limited to the basic coordinate system. In that system the equations decouple, C_c is unity and C_{rc} is zero. Then equations 7.2.14 and 7.2.15 reduce to elimination of rows and columns.

To properly account for the coupling that occurs when the constraints are not applied in the basic coordinate system, we must generate all the constraint equation on the node. This may be up to a 6x6 system. I believe that there is no real conflict in first applying constraints in the basic system, then adding additional constraints in other systems.

The process for applying constraints can be summarized as follows.

1. Generate the constraint equation in the generalized coordinate system (equation 7.2.1).
2. Transform the constraint equation to the basic coordinate system (equation 7.2.2).

3. Determine the constraint degrees of freedom. It may need to be done in concert with the next step to keep from degrading the matrix condition.
4. Compute the two transformation matrices C_c^{-1} and C_{rc} from equations 7.2.5 and 7.2.9.
5. Compute the corrections to the force vector from equation 7.2.15. We currently do not have a structure to store these corrections, except for the case of statics.
6. Compute the reduced mass and stiffness matrices from equation 7.2.14.
7. Eliminate the constraint degrees of freedom from the mass and stiffness matrix.

   In addition, for post processing,
8. store the terms of the equations necessary to recover the constraint degrees of freedom (equation 7.2.6).

A few words about post processing could also prove useful. In the first implementation of **Sierra/SD**, constraints were applied only in the basic coordinate system. The degree of freedom to eliminate was obvious from the **Exodus** file, and its value was a constant (usually zero). In this later version, a more general approach must be used. We use the following strategy.

1. degrees of freedom directly constrained to zero are handled implicitly. This is done by setting the G-set vector to zero before merging in the A-set results. There is no storage cost for this.
2. Other degrees of freedom are managed using an spc_info object. An array of these objects will be stored globally. Each object contains the degree of freedom to fill, an integer indicating the number of other terms, a list of dofs/coefficients, and a constant. This facilitates solutions of the form,

$$u_{\text{spc}} = \text{constant} + \sum_{i}^{\text{retained dofs}} u_i C_i \tag{7.2.17}$$

7.2.4. Multi Point Constraints

The application to multi-point constraints is straight forward. The only difference is that the whole system of equations must be considered together. This changes the linear algebra significantly because the matrices must be stored in sparse format. However, the steps that are applicable for single point constraints also apply here. Subsection 7.1 deals more explicitly with MPCs.

7.2.5. Transformation of Power Spectral Densities

Note: The following is taken almost verbatim from Paez's book [133]. We identify how to transform output PSD.

Let $\mathbf{H}(f)$ denote a frequency response function vector for a given input (in the global system) expressed as,

$$\mathbf{H}(f) = H_1(f)\mathbf{e}_1 + H_2(f)\mathbf{e}_2 + H_3(f)\mathbf{e}_3$$

where $\mathbf{e}_i$ represents the unit vectors of this space. Note that $\mathbf{H}(f)$ is an output vector at a single location in the model. $\mathbf{H}(f)$ can also be expressed using an alternate set of unit vectors, $\tilde{\mathbf{e}}_i$.

$$\mathbf{H}(f) = \tilde{H}_1(f)\tilde{\mathbf{e}}_1 + \tilde{H}_2(f)\tilde{\mathbf{e}}_2 + \tilde{H}_3(f)\tilde{\mathbf{e}}_3$$

Taking the dot product of these two equations and equating the results, we have,

$$\tilde{H}_1(f) = \sum_{k=1}^{3} c_{ki} H_k(f) \tag{7.2.18}$$

where

$$c_{ki} = \mathbf{e}_k \cdot \tilde{\mathbf{e}}_i$$

The spectral density function $G_{ij}(f)$ (for a given input and at a single output location) can be expressed as,

$$G_{ij}(f) = \alpha H_i^*(f) H_j(f) \tag{7.2.19}$$

where α is a constant and superscript * denotes complex conjugate. Similarly for the alternative coordinate frame,

$$\tilde{G}_{ij}(f) = \alpha \tilde{H}_i^*(f) \tilde{H}_j(f)$$

We may use equation 7.2.18 to express $\tilde{G}$ in terms of the H_i. We may then use the spectral definition in equation 7.2.19 to provide the transformation of spectral densities.

$$\begin{aligned} \tilde{G}_{ij}(f) &= \alpha \left(\sum_{k=1}^{3} c_{ki} H_k^*(f) \right) \left(\sum_{m=1}^{3} c_{mj} H_m(f) \right) \\ &= \sum_{k=1}^{3} \sum_{m=1}^{3} c_{ki} c_{mj} G_{km} \end{aligned} \tag{7.2.20}$$

This can be expressed in matrix notation as $\tilde{G} = C^T G C$.

7.3. Orthogonality of MPC to Rigid Body Vectors

There are many requirements on multipoint constraints (MPCs). One that is essential is that the constraints must be orthogonal to rigid body rotations. By this we mean that the multipoint constraints must not constrain the system in a way that eliminates rigid body motion. This can be easily seen in modal analysis. An ungrounded system with MPCs must retain 6 rigid body modes. Transient and static analysis has the same issues, but here the problem may not be as obvious. Note that there are a variety of means of arriving at the weights for a set of constraints, such as tied data. A mortar method preserves rigid body motion with a different set of constraints. The weights for these systems may differ, but all must allow the body to freely rotate. Clearly each constraint equation must satisfy this orthogonality independently.

For tied data a nodal dof on the node-surface $\vec{x}_s$ is constrained to the nearest face by a row of C. R is a function of the coordinates. Effectively R is a function of the lofting. Particular solutions of the family of equations

$$C(\lambda) R(\lambda) = 0 \tag{7.3.1}$$

are determined, ensuring that C is a continuous function of the lofting parameter. In other words, enforcing orthogonality changes the constraints as little as possible.

7.3.1. *Beam Example*

Figure 7-1 illustrates a node $\vec{x}_3$ constrained to a beam with nodes $\vec{x}_1$ and $\vec{x}_2$. This beam is represented using a 2 dimensional coordinate frame, and has no rotational degrees of freedom. The X axis is aligned with the beam. There are two dof per node. The node $\vec{x}_3$ is located a distance d from the node $\vec{x}_1$.

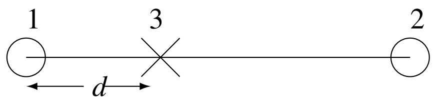


Figure 7-1. – Node Constrained Directly to Beam.

The displacement vector is defined as,

$$U = [u_{1x}\, u_{1y}\, u_{2x}\, u_{2y}\, u_{3x}\, u_{3y}] \tag{7.3.2}$$

The high level approach of sections 7.3.1 and 7.3.2 is to address certain deficiencies by activating different dof of nodes. Some Sierra codes do not allow for constraints that couple different dof of the same nodes.

The constraints keeping node $\vec{x}_3$ on the beam ($x_3 = x_1 + d$) are

$$C(0) = \begin{bmatrix} (1-d) & 0 & d & 0 & -1 & 0 \\ 0 & (1-d) & 0 & d & 0 & -1 \end{bmatrix} \tag{7.3.3}$$

and the corresponding three orthogonal rigid body vectors are,[1] The node
$\vec{x}_s = \vec{x}_3 = [x_3, y_3]^T$,$x_3 = [x_1, x_2][1-d, d]^T$, $y_3 = 0$. The origin o is chosen to make the rigid modes orthogonal, $o = x_1 + h$, $h = (x_2 - x_1)/2$. Finally, $x_3 = o + (2d-1)h$.

$$R(0)^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & -\theta & 0 & \theta & 0 & (2d-1)\theta \end{bmatrix}, \quad \theta = 1 \tag{7.3.4}$$

The constraints C are orthogonal ($C \cdot R = 0$) to the rigid body vectors, R.

7.3.2. *Offset Example*

A small offset of a tied node above the tied face is common for a variety of reasons. For example, tying together nodes on curved surfaces often introduces an offset from the plane of constraints, as is illustrated in Figure 7-2. Figure 7-3 shows the general case in which the third node is offset, L, along the positive Y axis.

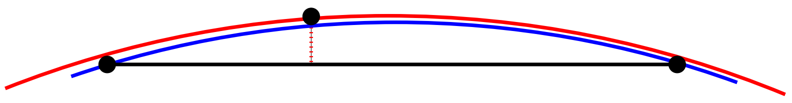


Figure 7-2. – Example Node on Face Constraint on Cylinder. The faceted faces produce a small offset from the nodal location of a point on the matching cylinder.

[1]We are using infinitesimal rotations where $\sin(\theta) = \theta$.

The point on the node-surface, $\vec{x}_s = \vec{x}_3 = [x_3, y_3]^T$, is lofted $y_3 = L$. The corresponding rigid body modes are

$$R(\lambda)^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & 1 & \lambda & (2d-1) \end{bmatrix}, \quad \lambda = L\text{sign}(1/2 - d)/h \tag{7.3.5}$$

What is important here is that the rotation rigid body mode gains an extra term. Rotation of this beam about the Z axis now has a term in X. These rotational rigid body modes are no longer orthogonal to the original constraints, 7.3.3.

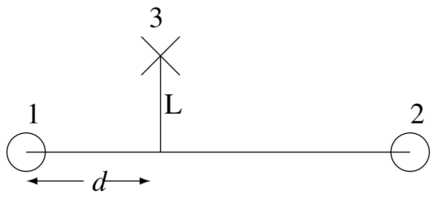


Figure 7-3. – Node Constrained Offset to Beam.

Row one of $C(0)$ is the problem; row two of $C(0)$ equals row two of $C(\lambda)$. In this paragraph, $c(\lambda)$ is row one of $C(\lambda)$. As a sparse vector, the graph of $c(\lambda)$ is the set of nonzeros. The only vector orthogonal to the RBM, with the same graph as $C(0)$, namely $[1, 0, -1, 0, 0, 0]$, does not constrain the node. The graph of $c(\lambda)$ will have to expand. Adding the y dof of active nodes to graph of C, the solution of equation 7.3.1 is

$$c(\lambda) = [1 - d, \lambda/2, d, -\lambda/2, -1, 0]$$

7.3.3. Correct MPC Equations

A solution to the problem can be obtained by using a projection onto the plane, as illustrated in Figure 7-4. The constraints for the projected node are determined from the standard shape functions of the element face, as in equation 7.3.3. However, we also maintain a perpendicular offset from that projection point on the face to the constrained node.

$$\vec{u}_s = \vec{u}_p + \vec{u}_r$$

and,

$$\vec{u}_r = \vec{\theta} \times \vec{\epsilon}$$

where $\vec{\theta}$ represents the rotation vector, and $\vec{\epsilon}$ represents the offset. When using shells and beams, we have $\vec{\theta}$ as a natural part of the rotational coordinates. For solids elements, we must compute $\vec{\theta}$.

Initially, one may conclude that higher order elements would alleviate the issues somewhat. Quadratic shape functions for these elements can properly represent second order geometry and displacements. However, multipoint constraints are inherently linear. We have not yet evaluated the effects of MPCs on curved, higher-order element faces.

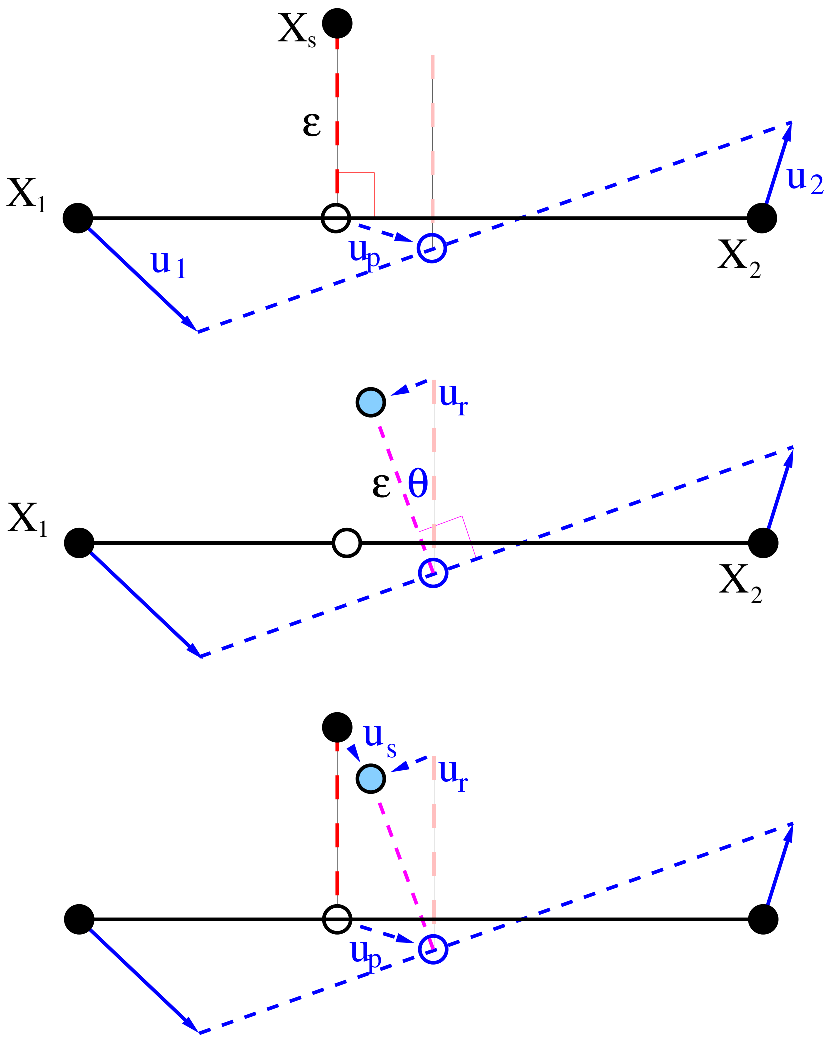


Figure 7-4. – Constraint Projection. Standard shape functions provide the constraint relations for the projected point, U_p. A rigid perpendicular offset maintains the proper geometry to retain rigid body invariance, and is used to compute $\vec{u}_r$. The total, $\vec{u}_s$ is the sum of these components.

7.3.4. Orthogonalization of Incorrect MPCs

A simple orthogonalization step can make the constraint weights once again orthogonal.[2] We compute,

$$\alpha = \vec{C} \cdot \vec{R}_i / ||\vec{R}_i||^2 \tag{7.3.6}$$

$$\tilde{C} \leftarrow \vec{C} - \alpha \vec{R}_i \tag{7.3.7}$$

where $\vec{C}$ represents the constraint equation, and $\vec{R}_i$ represents one of the *orthogonalized* rigid body modes. As long as they span a full space, we can restrict $\vec{R}$ to the nodes in the constraint interaction. This allows us to modify a constraint without generating terms that extend across the entire body. Typically, this operation will add terms to C that were previously zero. In general, this operation must be performed for all rigid body modes on each constraint.

The orthogonalization process of equation 7.3.7 works for shell and beam models that include rotational degrees of freedom on the nodes of the constraint. If rotational dofs are added to constraints applied only to solid elements, those constraints are ineffective because solid elements have no active rotational degrees of freedom. However, if the degrees of freedom in the constraint spans the space properly, these rotational degrees of freedom may be removed and only translational degrees of freedom retained. Equation 7.3.7 still applies, but now is restricted to the translational degrees of freedom on nodes in the constraint.

7.3.4.1. Orthogonalization on incomplete space.

In some cases, there are insufficient degrees of freedom in the constraint equation to adequately span the space of the rigid body vectors. With shells and beams this is not an issue because the six dofs on a single node can represent 6 orthogonal rigid body rotations. When only solid elements are active, a minimum of three nodes are required to represent the same six rigid body modes. When insufficient degrees of freedom are available in the constraint, a few possibilities are presented for ensuring rigid body invariance.

1. In some cases the constraint may be orthogonal to all rigid body modes. No modification is necessary.

   This is the case for two co-located nodes that are constrained by a rigid translation. It can be shown in this case, that the rotation vector (expressed only as translational terms) is a null vector. The orthogonality with that vector is trivially zero.

2. The constraint could be eliminated. This may be the correct solution for two nodes tied only by rotation. In some cases, this may change the response of the solution.

3. Additional degrees of freedom from neighboring nodes could be introduced into the constraint. See the discussion in Figure 7-5.

Detection:

A critical issue is the identification of conditions that result in bad solutions. This occurs when the orthogonalization of the vector results in a null vector. To avoid numerical round-off issues we define this such that,

$$\frac{\tilde{C}}{C} < \delta$$

Where $\tilde{C}$ is the updated constraint equation determined from equation 7.3.7 and δ is a small quantity.[3]

[2] Orthogonalization can be achieved in a variety of means. This is one simple approach.

[3] chosen as 1/1000.

No constraints are added to the system. That would change the solution. The number of nodes (dofs) that are involved in the orthogonalization of the RBM increases. This is much like adding an extra independent term to a RBE3 averaging element. Recall that we restricted the RBM to the nodes involved in the constraint. This was an arbitrary choice, determined to avoid creating constraint equations that span the space of the solution. In this effort we broaden the space to ensure that the reduced rigid body vectors are long enough to permit orthogonalization of each vector with respect to the constraints.
Generally, we want to add degrees of freedom that are physically near the nodes in the constraint, however addition of nodes that are collocated or co-linear with existing constraint nodes is not beneficial. We use the following strategy.

1. Determine the centroid of the MPC, $\vec{x}_o$, and a characteristic length, L.
2. Select the N nearest nodes from each processor, that are *not* part of the MPC. This requires a sort by location.
3. Communicate, and contract this list to the N nearest nodes in space.
4. Apply these additional degrees of freedom, and recompute the $\tilde{C}$ vector and norms.
5. If the norm is still zero, issue a message and abort.

Figure 7-5. – Additional Nodes in the MPC. Unimplemented.

7.3.5. *Adding the same dof of new nodes*

This section revisits the offset beam problem, discussed in section 7.3.2. Here the same dof of certain other nodes are added to the graph. The constrained node is $\vec{x}_5 = [x_5, y_5]^T$,$x_5 = [x_1, x_2][1-d, d]^T$, and $y_5 = L$. In node-face contact, the other vertices of the face that have been filtered out are the natural choice: $(\vec{x}_i)_{i=1}^4$. Typically

$$\vec{x}_3 \sim \vec{x}_2 + [0, \tilde{y}]^T, \quad \vec{x}_4 \sim \vec{x}_1 + [0, \tilde{y}]^T \tag{7.3.8}$$

The dimensionless parameters of interest are $\eta = \tilde{y}/h$, $\tilde{y} < 0$, and $\lambda = L\text{sign}(1/2 - d)/h$.

Hypothesis for x dof solution: $\eta + \lambda \neq 0$ or equivalently $\tilde{y} + L\text{sign}(1/2 - d) \neq 0$.

Differentiating equation (7.3.1), and once again letting c denote row one of C, $\dot{c}R + c\dot{R} = 0$,$c\dot{R} = [0, 0, 1]^T$. Nodes $\vec{x}_1$ and $\vec{x}_2$ handled the $c(0)$ term. Nodes $\vec{x}_3$ and $\vec{x}_4$ handle the $\dot{c}(0)$ term.

Define B as the result of removing the following rows and columns from R: remove the rows corresponding to the first 2 nodes, remove even rows corresponding to the y dof in c, and remove the middle column.

It helps to consider the case in which the approximation (7.3.8) is exact,

$$B = \begin{bmatrix} 1 & \eta \\ 1 & \eta \\ 1 & -\lambda \end{bmatrix}$$

The constraint is determined by $B^T\dot{c}(3:5) = [0, 1]^T$. The hypothesis is that B has full rank. If the approximation (7.3.8) is exact, $\eta + \lambda$ must be nonzero. More generally, the cross product of the columns is nonzero if and only if B has full rank, a condition that can be read off from the coordinates.

Solving $B^Tc(3:5) = [0, 1]^T$ is not trivial. Unfortunately this type of equation is typically solved via normal equations, whose inaccuracy increases with the need for accuracy. In terms of the economy size qr factorization of $B = QU$, (Q has the same size as B and U in $M(2, 2)$ is upper triangular),
$c(3:5) = QR^{-T}[0, 1]^T$. That means, for f such that $R^Tf = [0, 1]'$, the constraint is $c = [0; Qf]$.

7.3.6. *Lofted node-face constraints*

An element may or may not be tied to a node, $\vec{x}_s$, in a way that preserves rotations. This section is about detecting constraints that do not preserve rotations, and then modifying the constraints so that rotations are preserved. Lofting is a geometric characterization of the extent to which a node-face constraint preserves rotations.

To understand all of this, let's start with some simple cases: a node-face constraint tying a node to a planar triangular face, a planar quadrilateral face, a discussion of lofting, and then remarks the extent to a planar face accurately describes the general non-planar case.

A planar triangle is defined by three non-coincident nodes. A node-face constraint is not lofted if the constrained node is in the plane of the triangular face. The vertex coordinates determine the matrix

$$\tilde{R} = \begin{bmatrix} 1 & x_0 & y_0 & z_0 \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \end{bmatrix}$$

Recall the concept of barycentric coordinates. The vertices are coplanar if and only if $\tilde{R}$ has rank 3, in which case the plane is the 2d set of points of the form

$$\begin{bmatrix} 1 \\ \vec{x} \end{bmatrix}$$

in the range of $\tilde{R}^T$. Node triangular face contact involves the matrix

$$R = \begin{bmatrix} 1 & x_0 & y_0 & z_0 \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{bmatrix}, \quad (x_3, y_3, z_3) = \vec{x}_s \tag{7.3.9}$$

A node-face constraint, c, preserves rotations if and only if $c^T R = 0$. Or geometrically, node on planar triangular face constraints preserves rotations if and only if the constrained node is in the plane determined by the triangular face. A constraint that does not preserve constraints is *lofted* some nonzero distance λ above the plane,

$$\vec{x}_s = \vec{x}_p + \vec{n}\lambda$$

Here $\vec{x}_p$ is the orthogonal projection along the unit normal $\vec{n}$ of the lofted node onto the face.

The same argument applies to a planar quadrilateral. Although $\tilde{R}$ is 4 by 4 in this case, still has rank of only 3. Barycentric coordinates define a plane, as in the case of a triangle. Finally, R is 5 by 4 in this case.

In node-face constraints, if the nodes are not planar, then barycentric coordinates define a surface, instead of a plane. In the case of a quadrilateral, $\tilde{R}$ may have rank 4, but it is nearly singular.

A lofted constraint is fixed by adding nodes so that $\tilde{R}$ has a small condition number. This is done by adding the nodes of the element that contains the face. There are pathological cases in the SD test suite in which the "face" is a collection of nodes, and in these cases, nodes are added from one of the elements attached to one of the nodes.

There's a nifty construction of the new weights as a perturbation of the old weights, c, which not being documented anywhere else, will be documented here. The construction is reviewed in the case of a node tied to the quadrilateral face of a hexahedron. For the problem to be well posed, the new weights must be a perturbation that is proportional to λ. In light of this, it is helpful express the equations in terms of λ:

$$R = R(\lambda) = R(0) + e_5\lambda\vec{n}^T, \quad c = c(\lambda), \quad c(0)^T R(0) = 0$$

Our goal is to determine $c(\lambda)$ so that $c(\lambda)^T R(\lambda) = 0$. Substituting

$$c(\lambda) = c(0) + \lambda\dot{c}(0)$$

$$c(\lambda)^T R(\lambda) = \lambda(c(0)^T \dot{R}(0) + \dot{c}(0)R(\lambda))$$

Recalling that the last coordinate of $c(0)$ is -1, $c(0)^T \dot{R}(0) = -e_4\lambda(0, \vec{n}^T)$. After adding (in this case the other 4) nodes, there is a "reasonable" vector of weights s such that

$$R(0)^T s = \begin{bmatrix} 0 \\ \vec{n} \end{bmatrix}$$

Note that $c(0)$ had to be re-indexed after adding nodes. The nifty trick is the identity $R^T(\lambda)(I + c(0)e_9^T) = R^T(0)$. In particular

$$R^T(\lambda)(I + c(0)e_5^T)s = \begin{bmatrix} 0 \\ \vec{n} \end{bmatrix}, \quad \dot{c}(0) = (I + c(0)e_9^T)s \tag{7.3.10}$$

7.3.7. *Rotationally Invariant Spot Weld Constraints*

To support Spot Welds with finite gaps, a similar equation to (7.3.10) was applied to node-face constraints using a least-squares fit to rigid rotation.

Here, our goal is to create the 15-by-3 constraint matrix C_{eqn} referenced by (7.10.1). Where C_{eqn} is expressed using the derivative of the constrained node's displacement $\vec{u}_d$ with regard to displacements on the face $\vec{u}_f$.

$$C_{eqn} = \begin{bmatrix} \frac{\partial \vec{u}_d}{\partial \vec{u}_f} \\ -I \end{bmatrix} \tag{7.3.11}$$

That derivative is given as:

$$\frac{\partial \vec{u}_d}{\partial \vec{u}_f} = N_p^T + \left[(A_p^T A_p)^{-1} A_p^T (I - A_1 N_p)\right] \times \hat{g} \tag{7.3.12}$$

Where:

- $\hat{g}$: Gap vector from projected point to constrained node
- A_p : n-by-3 Rigid Rotation vectors of the face nodes
- A_1 : n-by-3 Rigid Translation vectors of the face nodes
- N_p : 3-by-n Shape function matrix(7.3.13)

$$N_p = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 \end{bmatrix} \tag{7.3.13}$$

To understand equation (7.3.12), we can rewrite it in terms of a rotation vector about the projected point $\vec{\theta}$:

$$\frac{\partial \vec{u}_d}{\partial \vec{u}_f} = N_p^T + \left[\frac{\partial \vec{\theta}}{\partial \vec{u}_f}\right] \times \hat{g} \tag{7.3.14}$$

Note that there are multiple valid ways to estimate $\frac{\partial \vec{\theta}}{\partial \vec{u}_f}$, but we found equation (7.3.15) to be the most robust when applied to poor quality elements.

$$\frac{\partial \vec{\theta}}{\partial \vec{u}_f} = (A_p^T A_p)^{-1} A_p^T (I - A_1 N_p) \tag{7.3.15}$$

7.4. Constraints and infinite eigenvalues

Constraints (in §7.1) modify equation (2.6.1) to an eigenvalue problem

$$A \begin{bmatrix} \phi \\ \lambda \end{bmatrix} = B \begin{bmatrix} \phi \\ \lambda \end{bmatrix} \omega^2 \tag{7.4.1}$$

$$A = \begin{bmatrix} K & C^T \\ C & 0 \end{bmatrix}, \quad B = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}.$$

The modes and mode shapes and modes satisfy the equation

$$K\phi + C^T \lambda = M\phi\omega^2, \tag{7.4.2}$$

Like superelements, Lagrange multipliers λ are not part of the finite element mesh interface. Lagrange multipliers are not exposed to users. When an eigenvalue problem is restarted, the Lagrange multipliers for the modes in the restart file are all set to zero.

The remainder of this section discusses a subtle issue that developers need to understand "once in a blue moon." If constraints are present then there are *infinite* modes

$$\begin{bmatrix} 0 \\ \lambda \end{bmatrix}, \quad B \begin{bmatrix} 0 \\ \lambda \end{bmatrix} = 0.$$

Approximate solutions of the constrained eigenvalue problem can be misleading if the infinite modes are not deflated. The deflation technique is due to Hans Weinberger. Fortunately in **Sierra/SD**, the deflation matches the Lagrange multiplier methods used to solve the linear systems,[47,48] and is handled, for the most part, behind the scenes. **Sometimes however, such as during debugging, it is necessary to understand this, and this section is included to address that case.**

But before diving in, let's go over what the constrained eigenvalue problem, equation (7.4.1), has in common with equation (2.6.1). Multiplying ϕ^T and row one of equation (7.4.1),

$$K\phi + C^T \lambda = M\phi\omega^2,$$

brings us to the unconstrained equation

$$\phi^T K\phi = \phi^T M\phi\omega^2.$$

The standard normalization

$$\phi^T(K, M)\phi = (\Lambda, I)$$

is used here too. Although

$$C\phi = 0,$$

note that

$$[0,\ \lambda]^T \begin{bmatrix} K & C^T \\ C & 0 \end{bmatrix} = [\lambda^T C, 0] \neq 0$$

is the force maintaining the constraints.

The elimination of the redundant constraints uses the partition (or more precisely reordering) $C = [C_r, C_c]$ so that C_c square and non-singular. This is done by the linear solver. The corresponding partition of ϕ into retained (independent) and constrained (dependent) vectors is

$$\phi = \begin{bmatrix} \phi_r \\ \phi_c \end{bmatrix}.$$

The constraint equation is $C_r\phi_r + C_c\phi_c = 0$, or $C_c^{-1}C_r\ \phi_r + \phi_c = 0$ or

$$C_{rc} = -C_c^{-1}C_r, \quad \phi_c = C_{rc}\phi_r. \tag{7.4.3}$$

The dimension of ϕ_c equals the dimension of λ. The partition also induces a change in the eigenvalue problem.

$$\begin{bmatrix} K_{dd} & K_{di} & C_r^T \\ K_{id} & K_{ii} & C_c^T \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_c \\ \lambda \end{bmatrix} = \begin{bmatrix} M_{dd} & M_{di} \\ M_{id} & M_{ii} \end{bmatrix} \begin{bmatrix} \phi_r \\ \phi_c \end{bmatrix} \lambda$$

To eliminate ϕ_c,

$$\begin{bmatrix} K_{dd} + K_{di}C_{rc} & C_r^T \\ K_{id} + K_{ii}C_{rc} & C_c^T \end{bmatrix} \begin{bmatrix} \phi_r \\ \lambda \end{bmatrix} = \begin{bmatrix} M_{dd} + M_{di}C_{rc} \\ M_{id} + M_{ii}C_{rc} \end{bmatrix} \phi_r\lambda \tag{7.4.4}$$

And finally to eliminate λ, in equation (7.4.4) subtract from row one $-C_{rc}^T$ times row two. For S defined by

$$S(K) = K_{dd} + K_{di}C_{rc} + C_{rc}^T K_{id} + C_{rc}^T K_{ii}C_{rc},$$

the reduced eigenvalue problem is

$$S(K)\phi_r = S(M)\phi_r\lambda$$

Given ϕ_r and λ, equation (7.4.3) determines ϕ_c. And λ is determined by

$$\lambda = C_c^{-T}\left(M_{id} + M_{ii}C_{rc} - K_{id} - K_{ii}C_{rc}\right)\phi_r$$

7.5. Sparsepak Contact Enforcement

Constraints may be eliminated using the constraint transform method. This is described in detail in Cook, chapter 9 (ref[37]). In this method, the analysis set is partitioned into constrained degrees of freedom and retained degrees of freedom. The constrained dofs are eliminated.

Unlike many Finite Element programs, **Sierra/SD** does not support user specification of constraint and residual degrees of freedom. The partition of constrained and retained degrees of freedom is performed simultaneously in the "Gauss()" routine. This routine performs full pivoting so the constrained degrees of freedom are guaranteed to be independent. Redundant specification of constraint equations is handled by elimination of the redundant equations and issue of a warning. User selection of constrained dofs in NASTRAN has led to serious difficulty to ensure that the constrained dofs are independent and never specified more than once.

For constraint elimination we have a constraint matrix $C = [C_r,\ C_c]$ where C_c is a square, non-singular matrix and C_r is the solution. We wish to solve for,

$$C_{rc} = -[C_c]^{-1}C_r$$

This is equivalent to the Gauss-Jordan elimination problem for $Kx = b$ if we let $C_r = b$, $C_c = K$ and $x = -C_{rc}$. There is one additional wrinkle: we have mixed the rows of C so C_c is intermingled with C_r. However, we only require that C_C be non-singular. Therefore, if we do a Gaussian elimination with full pivoting we should simultaneously obtain an acceptable reordering of C, and obtain C_{rc}.

In practice, it is not even necessary that C_c be non-singular. It is not uncommon for two identical constraints to be specified. The program issues a warning and continues.

Constraint transform methods do not currently support recovery of MPC forces.

The Gaussian elimination is presently being performed with a sparse package called "SuperLU," instead of a dense Gaussian elimination, to speed up the time to create C_{rc}. On some platforms, e.g., sgi and DEC, the BLAS routine `dmyblas2.c` in the CBLAS directory of of the SuperLU directory (need superlu-underscore-salinas.tar to create this) should be the one and only routine whose object file is placed into the SuperLU-BLAS library (presently called libblas-underscore-super.a) to be linked in to create the **Sierra/SD** executable. Failure to include this routine will cause failures of the type "Illegal value in call to DSTRV" on the above platforms, and including more than dmyblas2.c can cause slow performance on many platforms as the SuperLU-CBLAS could override the built-in BLAS routines. (The built-in routines are almost always faster.)

7.6. GDSW Contact Enforcement

A GDSW contact enforcement method is summarized. Maintaining constraints, i.e. given any $\tilde{u}$, finding "near by" $u = T\tilde{u}$ satisfying the constraints, is discussed at the end. Contact introduces a residual force to the momentum equation,

$$Ku + C^T\lambda = f \tag{7.6.1}$$

and the constraint

$$Cu = 0, \quad C \text{ is } r \times n, \quad r \ll n \tag{7.6.2}$$

A null space basis Z of rank $\leq n - r$ satisfies $CZ = 0$. The *full rank case*, $\mathrm{rank}(Z) = n - r$, is addressed here (with the complicated software handling the general case, and including many important optimizations). Displacements are of the form $u = Zv$, and the momentum equation, (7.6.1), reduces to $(Z^T KZ)v = Z^T f$.

Direct elimination is a null space basis method in which permutation matrices Q and P are found such that

$$0 = QCPu_P = C_S u_P = [C_{SI}, C_{SD}] \begin{bmatrix} u_{IP} \\ u_{DP} \end{bmatrix}, \quad u = Pu_P$$

Here D and I denote the dependent and independent sets. The full rank case has C_{SD} nonsingular for $|S| = |D| = r$. A clever notation is $C_{DS}C_{SD} = I$ and $C_{DS}C_{SI} = C_{DI}$. Independent displacements u_{IP} are independent of the constraints. Meanwhile u_{DP} depends on u_{IP} through the constraints,

$$u_{DP} + C_{DI}\, u_{IP} = 0, \quad Z = \begin{bmatrix} I \\ -C_{DI} \end{bmatrix}.$$

In practice an LU decomposition

$$C^T = P \begin{bmatrix} L_D \\ L_I \end{bmatrix} UQ$$

leads to

$$L_D^T\, u_{DP} + L_I^T\, u_{IP} = 0, \quad C_{DI} = L_D^{-T} L_I^T.$$

The transformation $T = PZP_I^T$ resets the dependent constraints, leaving the independent constraints invariant. Here $P = [P_D, P_I]$ so that in particular $\tilde{u}_{IP} = P_I^T \tilde{u}$.

7.7. Tied Friction

The work on tied surfaces with friction is under development. Details are maintained in our design documentation.

7.8. Mortar Methods

For simplicity, we only consider one of the three components of displacement in the following development; the same approach holds for the other two components of displacement. Let u_b and u_a denote displacements on the b and a sides of a mesh interface. Ideally, we would like to satisfy

$$u_a = u_b$$

at all locations on the interface. This restriction, however, is only practical for meshes which are conforming at the interface. Otherwise, displacements would be restricted to a low-order polynomial of degree equal to that of the lowest-order finite element on either side of the interface. As a result, the interface would be too stiff.

For mortar methods, the constraint $u_a = u_b$ is only satisfied in a weak sense. Specifically, the mortar constraints are of the form

$$\int_\Gamma \lambda(u_a - u_b)\, dx = 0, \tag{7.8.1}$$

where Γ denotes the interface and λ is a Lagrange multiplier. Notice the familiar inconsistent tied contact (node on face) constraints for node can be expressed in this form by choosing λ as a Dirac delta function for the subject node. For mortar methods it is important that constant functions are in the space of Lagrange multipliers. Dirac delta functions cannot be combined to obtain a constant. Thus, we should not expect the convergence rates of mortar and tied contact methods to be identical. Indeed, the convergence rates for tied contact are in general suboptimal.[22]

Let q_b and q_a denote vectors of nodal values of displacement on the b and a sides of the interface. Similarly, let q_λ denote a vector of discrete values of the Lagrange multiplier. The displacements and Lagrange multiplier are approximated (discretized) as follows:

$$u_b = \phi_b^T q_b, \tag{7.8.2}$$

$$u_a = \phi_a^T q_a, \tag{7.8.3}$$

$$\lambda = \phi_\lambda^T q_\lambda, \tag{7.8.4}$$

where ϕ_b and ϕ_a are vectors of shape functions for the b and a sides of the interface, and ϕ_λ is a vector of shape functions for the Lagrange multiplier. A discrete form of the mortar constraints are obtained from substitution of (7.8.2-7.8.4) into (7.8.1).

$$M_{ss} q_a + M_{sm} q_b = 0, \tag{7.8.5}$$

where

$$M_{ss} = \int_\Gamma \lambda_a \phi_a^T \, dx, \quad M_{sm} = \int_\Gamma \lambda_a \phi_b^T \, dx. \tag{7.8.6}$$

The *standard* mortar method implemented in ACME uses

$$\phi_\lambda = \phi_a. \tag{7.8.7}$$

In other words, the Lagrange multiplier shape functions are the same as the shape functions for the a side of the interface. We note in the mortar methods literature that Lagrange multiplier shape functions are often modified for a nodes on the boundary of the interface. The purpose for this modification is to avoid redundant constraints at the intersection of two or more interfaces. At present, we make no such modifications, but we will revisit this topic in a later section. Substitution of (7.8.7) into (7.8.6) gives

$$M_{ss}^{standard} = \int_\Gamma \phi_a \phi_a^T \, dx, \quad M_{sm}^{standard} = \int_\Gamma \phi_a \phi_b^T \, dx. \tag{7.8.8}$$

Although the matrix $M_{ss}^{standard}$ is sparse and positive definite, its inverse is dense. Thus, if one were to solve (7.8.5) for q_a in terms of q_b, each a node displacement would depend on all the b side nodal displacements in the general case. As a result, solvers which make use of this form of constraint elimination would suffer from significant memory and computational demands for interfaces with large numbers of nodes.

Dual mortar methods find and use a Lagrange multiplier basis which leads to a diagonal M_{ss} matrix. Each a node displacement depends on the b node displacements in a neighborhood of the a node. Eliminating the a node displacements is efficient. Elimination is also efficient with tied contact.

Let σ denote an element face on the a side of the interface. Further, let $\sigma(\Gamma)$ denote the set of all such faces on Γ. From (7.8.6) we then have

$$M_{ss} = \sum_{\sigma \in \sigma(\Gamma)} M_{ss\sigma}, \quad M_{sm} = \sum_{\sigma \in \sigma(\Gamma)} M_{sm\sigma}, \tag{7.8.9}$$

where

$$M_{ss\sigma} = \int_\sigma \phi_\lambda \phi_a^T \, dx, \quad M_{sm\sigma} = \int_\sigma \phi_\lambda \phi_b^T \, dx. \tag{7.8.10}$$

For the dual mortar method, we choose the vector ϕ_λ to be a linear combination of rows of ϕ_a. Specifically, for each a face σ we set

$$\phi_\lambda = A_\sigma \phi_a, \tag{7.8.11}$$

where A_σ is a transformation matrix. To have a method which passes constant stress patch tests (linear consistency), it must be possible to obtain a constant function from a linear combination of the rows of ϕ_λ. We see that A_σ equal to the identity matrix satisfies this condition since the sum of all a shape functions over σ is unity. In this case, however, we recover the standard mortar method. The present goal is to choose A_σ to satisfy the constant approximation property while also leading to a diagonal matrix M_{ss}. To this end, we follow the construction in[134] and:[113]

$$A_\sigma = D_\sigma (M_{ss\sigma}^{standard})^{-1}, \tag{7.8.12}$$

where

$$D_\sigma = \operatorname{diag}\left(\int_\sigma \phi_a \, dx \right). \tag{7.8.13}$$

Replacing ϕ_a in (7.8.8) by $A_\sigma \phi_a$, we obtain

$$M_{ss}^{dual} = \sum_{\sigma \in \sigma(\Gamma)} \int_\sigma A_\sigma \phi_a \phi_a^T \, dx = \sum_{\sigma \in \sigma(\Gamma)} A_\sigma M_{ss\sigma}^{standard} = \sum_{\sigma \in \sigma(\Gamma)} D_\sigma, \tag{7.8.14}$$

$$M_{sm}^{dual} = \sum_{\sigma \in \sigma(\Gamma)} \int_\sigma A_\sigma \phi_a \phi_b^T \, dx = \sum_{\sigma \in \sigma(\Gamma)} A_\sigma M_{sm\sigma}^{standard}. \tag{7.8.15}$$

Since each D_σ is diagonal, it follows that M_{ss}^{dual} is also diagonal.

Numerical integration over each a face σ is done in ACME by first decomposing σ into a set of triangular facets $t(\sigma)$ and then summing the contributions from each facet. Specifically, from ACME we have access to the integrals

$$M_{sst}^{standard} = \int_t \phi_a \phi_a^T \, dx, \quad M_{smt}^{standard} = \int_t \phi_a \phi_b^T \, dx, \tag{7.8.16}$$

where $t \in t(\sigma)$. By assembling contributions to σ, we then calculate

$$M_{ss\sigma}^{standard} = \int_\sigma \phi_a \phi_a^T \, dx = \sum_{t \in t(\sigma)} M_{sst}^{standard}. \tag{7.8.17}$$

With $M_{ss\sigma}^{standard}$ in hand, we then calculate

$$M_{sst}^{dual} = A_\sigma M_{sst}^{standard} = D_\sigma (M_{ss\sigma}^{standard})^{-1} M_{sst}^{standard}, \tag{7.8.18}$$

$$M_{smt}^{dual} = A_\sigma M_{sst}^{standard} = D_\sigma (M_{ss\sigma}^{standard})^{-1} M_{smt}^{standard}. \tag{7.8.19}$$

Since $M_{ss\sigma}^{standard}$ is symmetric and positive definite, it can be factored using the Cholesky decomposition. Accordingly, products with the inverse of $M_{ss\sigma}^{standard}$ in (7.8.18) and (7.8.19) can be obtained with calls to `LAPACK` routines `DPOTRF` and `DPOTRS`. It then only remains to calculate the entries of the diagonal matrix D_σ.

Let e denote a vector of the same length as ϕ_a and with all its entries equal to 1. Since the sum of shape functions in ϕ_a equals 1 in σ, we have

$$\phi_a^T e = 1. \tag{7.8.20}$$

From (7.8.17) we then obtain

$$M_{ss\sigma}^{standard} e = \int_\sigma \phi_a (\phi_a^T e)\, dx = \int_\sigma \phi_a\, dx. \tag{7.8.21}$$

It follows from (7.8.13) that

$$D_\sigma = \text{diag}\left(M_{ss\sigma}^{standard} e \right). \tag{7.8.22}$$

The procedure used to calculate the transformed mortar matrices M_{sst}^{dual} and M_{smt}^{dual} for the dual Lagrange multiplier basis is summarized as follows.

1. Calculate $M_{ss\sigma}^{standard}$ by assembling contributions from triangular facets as in (7.8.17).
2. Calculate the diagonal matrix D_σ according to (7.8.22).
3. Calculate the mortar matrices M_{sst}^{dual} and M_{smt}^{dual} for the dual Lagrange multiplier basis according to (7.8.18) and (7.8.19).

In summary, all that is needed is to replace the mortar matrices $M_{sst}^{standard}$ and $M_{smt}^{standard}$ for each triangular facet t by their dual basis counterparts M_{sst}^{dual} and M_{smt}^{dual}. The remainder of the coding in ACME remains the same. The only code changes on the **Sierra/SD** side is to pass a flag to ACME indicating whether to use the dual mortar method.

A subsection titled *Treatment of Interface Boundary* explaining the special treatment of constrained nodes on the interface boundary to avoid potential redundant constraint equations would be a welcome addition. There is also room here for a subsection titled *Nodal Coordinate Adjustments* dealing with how to initially move the constrained nodes to retain all six rigid body modes for curved interfaces or flat interfaces with initial gaps.

7.9. Correction For Dynamic Constraint Equilibrium

Multipoint constraints defined in an initial condition that is in equilibrium are homogeneous. The constraint equation applied to the displacement, velocity, or acceleration vanishes. A constraint generated at an equilibrium maintains equilibrium for all time.

Under some circumstances in a transient analysis, constraints can be generated in a non-equilibrium state. This occurs, for example, if two domains are initialized to different pressures and then connected via an MPC. Additionally, MPCs created in the middle of a run, such as on a moving mesh, are often created in a state that is at least subtly out of equilibrium. In this circumstance, it is required to bring the constraint into an equilibrium state as quickly as possible to enforce the intended continuity. Generally, immediate enforcement of a constraint on the primary variable will not regain equilibrium. For example, if enforcement of the constraint immediately eliminates a displacement jump, this will cause a large discontinuity of velocity at the constraint.

To remedy this situation, a special sequence of non-homogeneous constraints is generated that brings the constraint back to equilibrium as quickly as possible: specifically, in three transient time steps.

Section 2.1 gives a detailed description of the Newmark beta time integration method. Let d^+ and d^- indicate the displacement variable on either side of an interface at which a constraint is to be applied. The constraint violation across the interface is $u = d^+ - d^-$. At the current step, we know the values

$$\begin{aligned} u_n &= d_n^+ - d_n^- \\ \dot{u}_n &= v_n^+ - v_n^- \\ \ddot{u}_n &= a_n^+ - a_n^-, \end{aligned}$$

but time-stepping must be done in a special way to bring $u, \dot{u}, \ddot{u}$ back to zero. Although not required for the method to work, we simplify the following discussion by assuming the standard values of $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$. Rewriting in u equation 2.1.4 for the Newmark beta step, we obtain equations 7.9.1 and 7.9.2.

$$\dot{u}_{n+1} = \dot{u}_n + \frac{\Delta t}{2}(\ddot{u}_n + \ddot{u}_{n+1}) \tag{7.9.1}$$

$$u_{n+1} = u_n + \Delta t \dot{u}_n + \frac{\Delta t^2}{4}\ddot{u}_n + \frac{\Delta t^2}{4}\ddot{u}_{n+1} \tag{7.9.2}$$

The target value for the constraint violation, u_{n+1}, will be specified later. Equation 7.9.2 can thus be rearranged to provide the unknown acceleration $\ddot{u}_{n+1}$ as a function of the known initial conditions and u_{n+1}, shown in equation 7.9.3.

$$\ddot{u}_{n+1} = \frac{-\ddot{u}_n \Delta t^2 - 4u_n + 4u_{n+1} - 4\Delta t \dot{u}_n}{\Delta t^2} \tag{7.9.3}$$

Recursively applying equations 7.9.1 and 7.9.3 yields the acceleration and velocity at the end of three steps as a function of the assumed target values $u_{n+1}, u_{n+2}, u_{n+3}$ for the constraint violation:

$$\dot{u}_{n+1} = \frac{-2u_n + 2u_{n+1} - \Delta t \dot{u}_n}{\Delta t} \tag{7.9.4}$$

$$\ddot{u}_{n+2} = \frac{-\ddot{u}_{n+1}\Delta t^2 - 4u_{n+1} + 4u_{n+2} - 4\Delta t \dot{u}_{n+1}}{\Delta t^2} \tag{7.9.5}$$

$$\dot{u}_{n+2} = \frac{-2u_{n+1} + 2u_{n+2} - \Delta t \dot{u}_{n+1}}{\Delta t} \tag{7.9.6}$$

$$\ddot{u}_{n+3} = \frac{-\ddot{u}_{n+2}\Delta t^2 - 4u_{n+2} + 4u_{n+3} - 4\Delta t \dot{u}_{n+2}}{\Delta t^2} \tag{7.9.7}$$

$$\dot{u}_{n+3} = \frac{-2u_{n+2} + 2u_{n+3} - \Delta t \dot{u}_{n+2}}{\Delta t} \tag{7.9.8}$$

Next assume a formula that will set the target constraint violation for the next step in terms of the current displacement, velocity, and acceleration constraint violation. Assume there exist some unknown coefficients weighting the mismatch in current displacement, velocity, and acceleration as given in Equations 7.9.9, 7.9.10, 7.9.11.

$$u_{n+1} = C_d u_n + C_v \Delta t \dot{u}_n + C_a \Delta t^2 \ddot{u}_n \tag{7.9.9}$$

$$u_{n+2} = C_d u_{n+1} + C_v \Delta t \dot{u}_{n+1} + C_a \Delta t^2 \ddot{u}_{n+1} \tag{7.9.10}$$

$$u_{n+3} = C_d u_{n+2} + C_v \Delta t \dot{u}_{n+2} + C_a \Delta t^2 \ddot{u}_{n+2} \tag{7.9.11}$$

Equations 7.9.7, 7.9.8, 7.9.11 can be simultaneously solved to find the update coefficients that yield zero displacement, velocity, and acceleration at the end of the third step:

$$u_{n+3} = 0, \quad \dot{u}_{n+3} = 0, \quad \ddot{u}_{n+3} = 0. \tag{7.9.12}$$

Note that by plugging 7.9.9 into 7.9.10 to express u_{n+1} in terms of C_d, C_v, C_a, and 7.9.10 into 7.9.11 to express u_{n+2} in terms of C_d, C_v, C_a, the equations become non-linear in the unknown coefficients C_d, C_v, C_a. This solution yields the coefficients in equation 7.9.13:

$$C_d = \frac{3}{4}, \quad C_v = \frac{1}{2}, \quad C_a = \frac{1}{16}. \tag{7.9.13}$$

When the update coefficients are used to set a target constraint violation at the next step, then for any initial conditions the constraint will reach total equilibrium after three Newmark beta time steps. Once this equilibrium is reached, the target displacement for the constraint becomes zero and for all future steps the constraint is a standard homogeneous constraint. Two examples of the equations of motion utilizing the constraint update coefficients are given in figures 7-6 and 7-7.

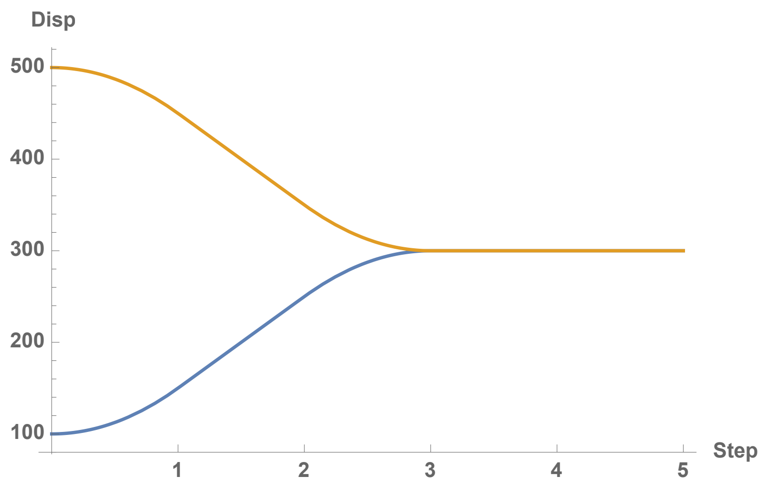


Figure 7-6. – Equilibration from $u_A = 100$ $u_B = 500$.

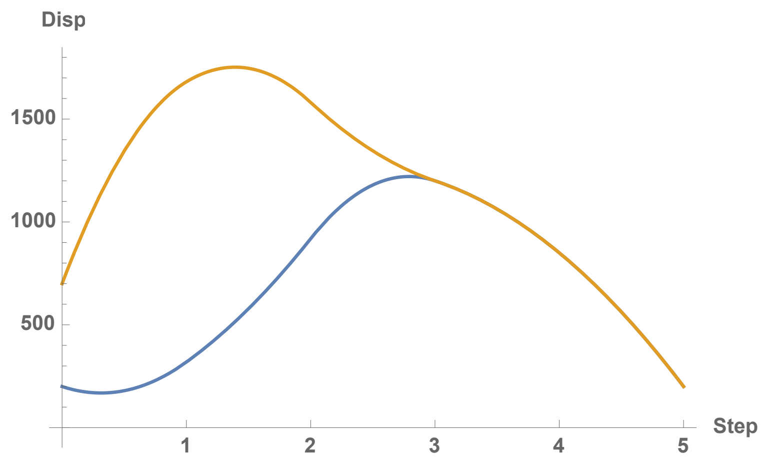


Figure 7-7. – Equilibration from $u_A = 200$ $u_B = 700$ $\dot{u}_A = -200$ $\dot{u}_B = 1600$ $\ddot{u}_A = -1000$ $\ddot{u}_B = 400$.

Implementation. The user interface is described in [116] in the General Commands chapter, GDSW section, especially in the Troubleshooting subsection. A few more detailed comments on contact enforcement are provided here.

In Sierra SD node-face tied contact, rigid elements, RBE3s, MPC equations introduce linear constraint equations to the linear system. In the GDSW linear solver linear constraint equations are segregated into Type 1 constraints and Type 2 constraints. Type 1 constraints are constraints where the number of terms in the constraint equation is less than a threshold (the default is 250). For example, a tied contact node constrained to the interior of a QUAD4 face would have 5 terms (1 for the node and 4 for the nodes of the face). Rigid elements like RBE2s also have a small number of terms and are thus Type 1 constraints.

Type 1 constraints are directly eliminated in GDSW prior to solving the linear system. This reduces the total number of equations to solve by the number of Type 1 constraints.

Type 2 constraints typically occur when RBE3 elements are used. RBE3 elements are "averaging" constraint equations used as a modeling convenience to distribute a load from a node to a larger surface. For example, one may like to connect a concentrated mass "uniformly" to the top of a cylindrical surface. An RBE3 element will introduce 6 or fewer constraint equations, but each constraint equation can have a whole bunch of terms proportional to the number of nodes on the "surface". Applying the constraint elimination algorithm used for Type 1 constraints to Type 2 constraints may introduce large dense blocks in the coefficient matrix. This increases the memory required to store the matrix and the factorizations of subdomain matrices. The current algorithm in GDSW for handling Type 2 constraints avoids potentially large increases in memory by requiring all constraints to be of Type 1.

Users may change it by setting max_numterm_C1 in the GDSW solver block. The big benefit of ensuring that all constraints are Type 1 is that the rate of convergence for the iterative solves is much higher.

7.10. Spot Welds

Spot Welds in **Sierra/SD** are defined as node-face connections between dissimilar meshes with user defined stiffnesses in the normal and tangential directions of the face.

Conceptually, spot welds can be represented by a 3-DOF linear spring attaching the constrained node at one end to a node-face contact MPC at the other. In practice however, we represent spot welds using 9-node quad elements which are exactly equivalent.

Element Matrices. We currently only define stiffness matrices for spot welds, but damping may be possible in the future.

Each 27-by-27 spot weld stiffness matrix is defined as:

$$[K_{elem}] = \left[C_{eqn}\right]\left[K_{spring}\right]\left[C_{eqn}\right]^T \tag{7.10.1}$$

Where C_{eqn} is a 27-by-3 set of node-face contact constraints, and K_{spring} is the 3-by-3 spring stiffness matrix in the global coordinate system.

Given the user defined normal(K_n) and tangential(K_t) stiffnesses, K_{spring} is defined as:

$$\left[K_{spring}\right] = [R]\begin{bmatrix} K_t & & \\ & K_t & \\ & & K_n \end{bmatrix}[R]^T \tag{7.10.2}$$

Where the rotation matrix R is chosen such that the local z axis($\hat{z}$) is parallel to the normal vector of the constrained face($\hat{n}$).

For non-planar faces, the normal vector $\hat{n}$ is evaluated at a point defined by projecting node 9 onto the contact face. The projection process is done by DASH during setup.

7.10.1. *An element block of possibly degenerate quad9 elements*

The ordering of quad 9 vertices shown in Figure 7-8 means that a spot weld assigns a quadrilateral to the first 8 quad nodes, with the dependent node last. A contact search may find triangles with 3 or 6 nodes, and quadrilaterals with 4 and 8 nodes. The element block consists only of quad9s, with varying repeated nodes depending on the number of nodes in the face. The repeated nodes always come from the face.

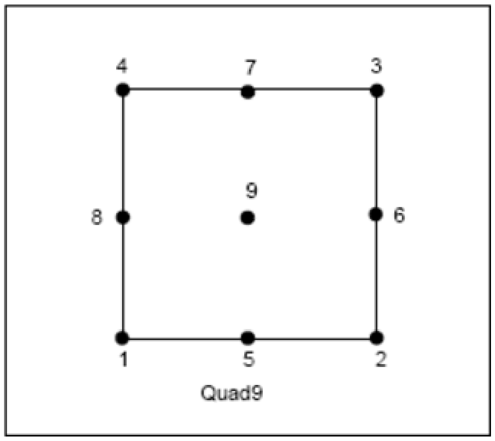


Figure 7-8. – Quad9 Element Topology.

7.10.2. *Stiffness Per Unit Area*

The user can build spot welds with either a constant stiffness at every node, or stiffness per unit area. When specifying stiffness per unit area, the area is evaluated on the surface owning the constrained nodes, not the constrained faces. This preserves solution convergence with mesh refinement.

This page intentionally left blank.

8. OUTPUT

8.1. L_2 Projection of Gauss Point Stresses

The purpose of this chapter is to provide some background material on how nodal stress projection calculations are performed in Sierra/SD. The first part provides a concise description of the L_2 projection, which involves solving a least squares problem, while the second part deals more with implementation details.

Given a square integrable stress function u and a finite element function $u_h \in U_h$, the stress projection problem is to find the u_h which minimizes

$$G(u_h) = \int_\Omega (u_h - u)^2 \, dx = (u_h - u, u_h - u),$$

where Ω is the domain for the problem and the inner product of two square integrable functions f and g is defined as

$$(f, g) = \int_\Omega fg \, dx.$$

For our purposes, U_h is associated with low-order elements such as the HEX8, TET4, and WEDGE6. Minimization of $G(u_h)$ gives the optimality conditions

$$(u_h - u, v_h) = 0 \quad \text{for all} \quad v_h \in U_h,$$

which is equivalent to

$$(u_h, v_h) = (u, v_h) \quad \text{for all} \quad v_h \in U_h.$$

An interesting point to mention is that the stress u is known only at Gauss points or element centroids. Thus, the inner product should be viewed as an approximation that is obtained in practice using numerical integration. As we show in the following development, finding u_h is equivalent to solving the linear system

$$Ma = b,$$

where M is the assembled finite element mass matrix for unit density, a is a vector of nodal stresses, and b is the assembled finite element *load* vector.

The domain for the problem is given by $\Omega = \bigcup_{i=1}^{K} \Omega_i$, where Ω_i is the domain of finite element i. A finite element function defined over Ω can be expressed as

$$u_h(x) = \sum_{j=1}^{N} a_j \phi_j(x) = a^T \phi(x),$$

where x denotes spatial position, N is the number of finite element nodes, a_j is the value at node j, $\phi_j(x)$ is the shape function for node j, and

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}, \quad \phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_N(x) \end{bmatrix}.$$

Our goal is to determine the nodal values in a which minimize the functional

$$G(a) = \int_{\Omega} (u_h(x) - u(x))^2 \, dx = \sum_{i=1}^{K} \int_{\Omega_i} (u_h(x) - u(x))^2 \, dx. \tag{8.1.1}$$

Within each Ω_i,

$$u_h(x) = \sum_{k=1}^{N_i} a_{ik} \phi_{ik}(x) = (R_i a)^T (R_i \phi(x)), \tag{8.1.2}$$

where N_i is the number of nodes for element i, a_{ik} are nodal values for element i, $\phi_{ik}(x)$ are shape functions for element i, and the superscript T denotes transpose. Further, $R_i a$ and $R_i \phi(x)$ select nodal values and shape functions from a and $\phi(x)$, respectively, for element i. Notice the number of rows in R_i is N_i and the number of columns is the total number of nodes N. In addition, each row of R_i has a single nonzero entry of 1.

Substituting (8.1.2) into (8.1.1), we find

$$G(a) = a^T \left(\sum_{i=1}^{K} R_i^T M_i R_i \right) a - 2a^T \sum_{i=1}^{K} R_i^T b_i + c, \tag{8.1.3}$$

where

$$M_i = \int_{\Omega_i} R_i \phi(x) \phi(x)^T R_i^T \, dx, \quad b_i = \int_{\Omega_i} R_i \phi(x) u(x) \, dx, \quad c = \int_{\Omega} u(x) u(x) \, dx.$$

Notice that M_i is the unit density mass matrix for element i and b_i is the *load* vector for element i, which depends on $u(x)$. The expression for G in (8.1.3) can be written succinctly as

$$G(a) = a^T M a - 2a^T b + c,$$

where

$$M = \sum_{i=1}^{K} R_i^T M_i R_i, \quad b = \sum_{i=1}^{K} R_i^T b_i.$$

Notice that M and b are the assembled finite element mass matrix and load vector, respectively. Minimization of G with respect to a then gives us

$$Ma = b,$$

which can be solved for the vector of nodal values a.

For our stress projection calculations, we currently obtain the element mass matrix M_i using the same numerical integration rule as for the element stiffness matrix of the higher-order element. Projection of HEX8 centroid stresses is an exception where the integration rule for the mass matrix is the same as the integration rule for the stiffness matrix of the HEX8 element. In contrast, the element load vector b_i is obtained using an integration rule consistent with the number of locations in Ω_i where $u(x)$ is available. These two integration rules may be different, as is the case for HEX8 elements where only a single point integration rule is used for b_i.

We refer the interested reader to a more comprehensive discussion of this topic in.[102]

INDEX

BIBLIOGRAPHY

[1] M. Ainsworth and J. T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. 1st ed. John Wiley & Sons, Inc., 2000 (cit. on p. 61).

[2] D. J. Allman. "A Compatible Triangular Element Including Vertex Rotations for Plane Elasticity Problems". In: *Comput. and Struct.* 19.1-2 (1996), pp. 1–8 (cit. on pp. 143, 144).

[3] A. Alonzo et al. "An Adaptive Finite Element Scheme to Solve Fluid-Structure Vibration Problems on Non-Matching Grids". In: *Computing and Visualization in Science* 4 (2001), pp. 67–78 (cit. on p. 82).

[4] Kenneth F. Alvin et al. "Incorporation of Sensitivity Analysis into a Scalable Massively Parallel Structural Dynamics FEM code". In: *Presented at the 5th U.S. Congress on Computational Mechanics*. Boulder, CO, Aug. 1999 (cit. on p. 60).

[5] Kenneth F. Alvin et al. "Membrane triangles with corner drilling freedoms – I. The EFF element". In: *Finite Elements in Analysis and Design* 12 (1992), pp. 163–187 (cit. on p. 145).

[6] M. Aminpour, J. Ransom, and S. McCleary. "A coupled analysis method for structures with independently modelled finite element subdomains". In: *Int. J. Numer. Meth. Engng.* 38 (1995), pp. 3695–3718 (cit. on p. 82).

[7] Anonymous. *Abaqus Theory Manual*. Dassault Systémes, 2011 (cit. on pp. 141, 142).

[8] Peter Arbenz et al. "A Comparison of Eigensolvers for Large-scale 3D Modal Analysis using AMG-Preconditioned Iterative Methods". In: *Int. J. Numer. Meth. Engng.* 1 (2003), pp. 1–21 (cit. on p. 44).

[9] R. J. Astley. "Infinite Elements Wave Problems: A Review of Current Formulations and an Assessment of Accuracy". In: *Int. J. Numer. Meth. Engng.* 49 (2000), pp. 951–976 (cit. on p. 166).

[10] R. J. Astley. "Transient Wave Envelope Elements for Wave Problems". In: *Journal of Sound and Vibration* 192.1 (1996), pp. 245–261 (cit. on pp. 166, 169, 171).

[11] R. J. Astley and J. P. Coyette. "Conditioning of Infinite Element Schemes for Wave Problems". In: *Communications in Numerical Methods in Engineering* 17 (2001), pp. 31–41 (cit. on p. 171).

[12] R. J. Astley, J. P. Coyette, and L. Cremers. "Three dimensional Wave Envelope Elements of Variable Order for Acoustic Radiation and Scattering Part II Formulation in the Time Domain". In: *Journal of the Acoustical Society of America* 103.1 (1998), pp. 64–72 (cit. on p. 166).

[13] R. J. Astley and J. A. Hamilton. "The Stability of Infinite Element Schemes for Transient Wave Problems". In: *Computer Meth. in Appl. Mech. Eng.* 195 (2006), pp. 3553–3571 (cit. on p. 169).

[14] R. J. Astley, G. J. Macaulay, and J. P. Coyette. "Mapped Wave Envelope Elements for Acoustical Radiation and Scattering". In: *Journal of Sound and Vibration* 170.1 (1994), pp. 97–118 (cit. on p. 169).

[15] R. J. Astley et al. "Three dimensional Wave Envelope Elements of Variable Order for Acoustic Radiation and Scattering Part I Formulation in the Frequency Domain". In: *Journal of the Acoustical Society of America* 103.1 (1998), pp. 49–63 (cit. on pp. 166, 171).

[16] B. A. Auld. *Acoustic Fields and Waves in Solids, Second Edition*. Vol. I. Robert E. Krieger Publishing Company, 1990 (cit. on p. 121).

[17] M. Baruch and Y. Zemel. “Mass Conservation in the Identification of Space Structures”. In: *AIAA Journal* 1239 (1989). ASME, ASCE, AHS, and ASC, Structures, Structural Dynamics and Materials Conference, pp. 710–712 (cit. on p. 21).

[18] Jean-Louis Batoz, Klaus-Jurgen Bathe, and Lee-Wing Ho. “A Study of Three-Node Triangular Plate Bending Elements”. In: *Int. J. Numer. Meth. Engng.* 15 (1980), pp. 1771–1812 (cit. on pp. 143, 144).

[19] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. 1st ed. John Wiley & Sons, 2000 (cit. on pp. 28, 29, 141).

[20] T. Belytschko, CS Tsay, and WK Liu. “A stabilization matrix for the bilinear Mindlin plate element”. In: *Computer Meth. in Appl. Mech. Eng.* 29.3 (1981), pp. 313–327 (cit. on p. 150).

[21] A. Bermudez, P. Gamallo, and R. Rodriguez. “A Hexahedral Face Element for Elastoacoustic Vibration Problems”. In: *JASA* 109.1 (2001), pp. 422–425 (cit. on p. 82).

[22] C. Bernardi, Y. Maday, and A. T. Patera. “A New Nonconforming Approach to Domain Decomposition: the Mortar Element Method”. In: *Nonlinear Partial Differential Equations and Their Applications. Collége de France Seminar, Vol XI (Paris, 1989-1991)*. vol 299 of Pitman Res. Math. Ser., Longman Sci. Tech., Harlow, 1994, pp. 13–51 (cit. on p. 223).

[23] C. Bernardi and R. Verfurth. “Adaptive finite element methods for elliptic equations with non-smoooth coefficients”. In: *Numerische Mathematik* 85 (2000), pp. 579–608 (cit. on p. 65).

[24] Robert D. Blevins. *Formulas for Natural Frequency and Mode Shape*. Malabar, FL, USA: Krieger, 1984 (cit. on pp. 146, 148).

[25] M. Brinkmeier et al. “A Finite Element Approach for the Simulation of Tire Rolling Noise”. In: *Journal of Sound and Vibration* 309.1-2 (2008), pp. 20–39 (cit. on p. 108).

[26] K. H. Brown et al. *ACME: Algorithms for Contact in a Multiphysics Environment*. Tech. rep. SAND2004-5486. Sandia National Laboratories, Oct. 2004 (cit. on p. 85).

[27] Gregory Bunting. *Strong and Weak Scaling of the Sierra/SD Eigenvector Problem to a Billion Degrees of Freedom*. Tech. rep. SAND 2019-1217. Sandia National Laboratories, 2019 (cit. on p. 44).

[28] Gregory Bunting, C.B. Smith, and T. Walsh. *Massively Parallel Capability in Sierra/SD for Vibration with Piezoelectrics*. Tech. rep. SAND2021-2373. PO Box 5800, Albuquerque, NM 87185-5800: Sandia National Laboratories, 2021 (cit. on p. 76).

[29] Gregory Bunting et al. “Parallel Ellipsoidal Perfectly Matched Layers for Acoustic Helmholtz Problems on Exterior Domains”. In: *Journal of Computational Acoustics* (2018) (cit. on p. 175).

[30] D. S. Burnett and R. L. Holford. “An Ellipsoidal Acoustic Infinite Element”. In: *Computer Meth. in Appl. Mech. Eng.* 164.1-2 (1998), pp. 49–76 (cit. on p. 180).

[31] X. Cai and A. Odegard. “Parallel Simulation of 3D Nonlinear Acoustic Fields on a Linux Cluster”. In: *IEEE International Conference on Cluster Computing*. 2000 (cit. on pp. 94, 95, 101, 103).

[32] K. Castor et al. “Long-Range Propagation of Finite-Amplitude Acoustic Waves in an Ocean Waveguide”. In: *JASA* 116.4 (2004), pp. 2004–2010 (cit. on p. 94).

[33] H. C. Chen and R. L. Taylor. “Vibration Analysis of Fluid-Solid Systems Using a Finite Element Displacement Formulation”. In: *Int. J. Numer. Meth. Engng.* 29 (1990), pp. 683–698 (cit. on p. 94).

[34] J. Chung and G. M. Hulbert. "A Time Integration Algorithm for Structural Dynamics with Improved Numerical Dissipation - The Generalized Alpha Method". In: *JAM* 60.2 (1993), pp. 371–375 (cit. on pp. 88, 98).

[35] J. L. Cippola and M. J. Butler. "Infinite Elements in the Time Domain using a Prolate Spheroidal Multipole Expansion". In: *Int. J. Numer. Meth. Engng.* 43 (1998), pp. 889–908 (cit. on p. 166).

[36] F. Collino and P. Monk. "The Perfectly Matched Layer in Curvilinear Coordinates". In: *SIAM J. Sci. Comp.* 19.6 (1998), pp. 2061–2090 (cit. on p. 179).

[37] R. D. Cook and M. E. Plesha D. S. Malkus. *Concepts and Applications of Finite Element Analysis*. 3rd. John Wiley & Sons, 1989 (cit. on pp. 49, 87, 88, 124, 144, 146, 150, 151, 201, 221).

[38] G. M. Corcos. "Resolution of Pressure in Turbulence". In: *J. Acoustical Society of America* 35.2 (1963), pp. 192–199 (cit. on p. 195).

[39] R. R. Craig. *Structural Dynamics: An Introduction to Computer Methods*. John Wiley & Sons, 1981 (cit. on pp. 38, 54).

[40] Chandler Davis and W. M. Kahan. "The Rotation of Eigenvectors by a Perturbation. III". In: *SIAM J. Numer. Anal.* 7.1 (1970), pp. 1–46 (cit. on p. 57).

[41] David M. Day and Tim Walsh. *Damped Structural Dynamics*. Tech. rep. SAND2007-2072. Sandia National Laboratories, 2007 (cit. on pp. 46, 47).

[42] Lawrence J. DeChant and Justin A. Smith. *Band Limited Correlation Estimates for A($\xi\omega$/U) and B($\eta\omega$/U) Using Beresh et. al. 2013 Data Sets*. Tech. rep. SAND2014-1123. Sandia National Laboratories, 2014 (cit. on p. 195).

[43] L. Demkowicz. *Computing with hp-Adaptive Finite Elements, Volume 1: One and Two Dimensional Elliptic and Maxwell Problems*. Chapman and Hall, CRC, 2007 (cit. on p. 177).

[44] L. Demkowicz and J. Shen. "A Few New (?) Facts about Infinite Elements". In: *Computer Meth. in Appl. Mech. Eng.* 195 (2006), pp. 3572–3590 (cit. on p. 171).

[45] L. Demkowicz et al. *Computing with hp-Adaptive Finite Elements, Volume 2: Frontiers, Three Dimensional Elliptic and Maxwell Problems with Applications*. Chapman and Hall, CRC, 2008 (cit. on p. 177).

[46] J. M. Dickens, J. M. Nagawa, and M. J. Wittbrodt. "A critique of mode acceleration and modal truncation augmentation methods for modal response analysis". In: *Comput. and Struct.* 62.6 (1997), pp. 985–998 (cit. on p. 113).

[47] Clark R. Dohrmann. *GDSW 101*. May 2008 (cit. on pp. 200, 219).

[48] Clark R. Dohrmann. *Some Notes on the 3-Level GDSW Solver*. Aug. 2005 (cit. on p. 219).

[49] Clark R. Dohrmann, S. Key, and M. Heinstein. "A Method for Connecting Dissimilar Finite Element Meshes in Two Dimensions". In: *Int. J. Numer. Meth. Engng.* 48 (2000), pp. 655–678 (cit. on p. 87).

[50] Clark R. Dohrmann, S. Key, and M. Heinstein. "Methods for Connecting Dissimilar Three-Dimensional Finite Element Meshes". In: *Int. J. Numer. Meth. Engng.* 47 (2000), pp. 1057–1080 (cit. on pp. 82, 87).

[51] D. Dreyer and O. von Estorff. "Improved Conditioning of Infinite Elements for Exterior Acoustics". In: *Int. J. Numer. Meth. Engng.* 58 (2003), pp. 933–953 (cit. on p. 171).

[52] R. Duran, C. Padra, and R. Rodriguez. "A Posteriori Error Estimates for the Finite Element Approximation of Eigenvalue Problems". In: *Mathematical Models and Methods in Applied Sciences* 13.8 (2003), pp. 1219–1229 (cit. on p. 68).

[53] M. S. Eldred, V. B. Venkayya, and W. J. Anderson. “Mode tracking issues in structural optimization”. In: *AIAA Journal* 33.10 (1995), pp. 1926–1933 (cit. on p. 59).

[54] M. Endo et al. “Flexible Vibration of a Thin Rotating Ring”. In: *Journal of Sound and Vibration* 92.2 (1984), pp. 261–272 (cit. on pp. 103, 189).

[55] B. O. Enflo and C. M. Hedberg. *Theory of Nonlinear Acoustics in Fluids*. Kluwer Academic Publishers, 2002 (cit. on pp. 94, 96).

[56] A. Ertas, J. T. Krafcik, and S. Ekwaro-Osire. “Explicit Formulation of an Anisotropic Allman/DKT 3-Node Thin Triangular Flat Shell Elements”. In: *Composite Material Technology* 37 (1991), pp. 249–255 (cit. on p. 144).

[57] G. C. Everstine. “Finite Element Formulations of Structural Acoustics Problems”. In: *Comput. and Struct.* 65.3 (1997), pp. 307–321 (cit. on pp. 81, 165).

[58] C. Farhat and P. Geuzaine. “Design and Analysis of Robust ALE Time-Integrators for the Solution of Unsteady Flow Problems on Moving Grids”. In: *Computer Meth. in Appl. Mech. Eng.* 193 (2004), pp. 4073–4095 (cit. on p. 95).

[59] C. Farhat, P. Geuzaine, and C. Grandmont. “The Discrete Geometric Conservation Law and the Nonlinear Stability of ALE Schemes for the Solution of Flow Problems on Moving Grids”. In: *J. Comp. Phys.* 174 (2001), pp. 669–694 (cit. on p. 95).

[60] C. A. Felippa. *The SS8 Solid-Shell Element: Formulation and a Mathematica Implementation*. Tech. rep. CU-CAS-02-03. Univ. Colo. at Boulder, 2002 (cit. on p. 140).

[61] Carlos A. Felippa and Scott Alexander. “Membrane triangles with corner drilling freedoms – III. Implementation and performance evaluation”. In: *Finite Elements in Analysis and Design* 12 (1992), pp. 203–239 (cit. on p. 145).

[62] Carlos A. Felippa and Carmelo Militello. “Membrane triangles with corner drilling freedoms – II. The ANDES element”. In: *Finite Elements in Analysis and Design* 12 (1992), pp. 189–201 (cit. on p. 145).

[63] D.P. Flanagan and T. Belytschko. “A Uniform Strain Hexahedron and Quadrilateral with Orthogonal Hourglass Control”. In: *Int. J. Numer. Meth. Engng.* 17 (1981). doi, pp. 679–706 (cit. on p. 131).

[64] D.P. Flanagan and T. Belytschko. “Simultaneous relaxation in structural dynamics”. In: *Journal of the Engineering Mechanics Division, ASCE* 107 (1981), pp. 1039–1055 (cit. on p. 131).

[65] B. Flemisch, M. Kaltenbacher, and B. Wohlmuth. “Elasto-acoustic and acoustic-acoustic coupling on non-matching grids”. In: *Int. J. Numer. Meth. Engng.* 67.13 (2006), pp. 1791–1810 (cit. on p. 82).

[66] R. L Fox and M. P. Kapoor. “Rate of Change of Eigenvalues and Eigenvectors”. In: *AIAA Journal* 6 (1968), pp. 2426–2429 (cit. on p. 59).

[67] F. Fuentes et al. “Orientation embedded high order shape functions for the exact sequence elements of all shapes”. In: *Computers and Mathematics with Applications* 70.1 (2015), pp. 353–458 (cit. on p. 14).

[68] M. J. Gagen. “Novel Acoustic Sources from Squeezed Cavities in Car Tires”. In: *JASA* 106.2 (1999), pp. 794–801 (cit. on p. 94).

[69] K. Gerdes. “A Review of Infinite Element Methods for Exterior Helmholtz Problems”. In: *Journal of Computational Acoustics* 8 (1 2000), pp. 43–62 (cit. on p. 166).

[70] Mircea Grigoriu. *Stochastic Calculus, Applications in Science and Engineering*. Birkhäuser, 2002 (cit. on p. 196).

[71] M. F. Hamilton and D. T. Blackstock. *Nonlinear Acoustics*. Academic Press, 1998 (cit. on pp. 94–97).

[72] I. Harari et al. “Recent Developments in Finite Element Methods for Structural Acoustics”. In: *Archives of Computational Methods in Engineering* 3 (1996), pp. 132–311 (cit. on p. 81).

[73] Bjørn Haugen. “Buckling and Stability Problems for Thin Shell Structures Using High Performance Finite Elements”. PhD thesis. Boulder: University of Colorado at Boulder, 1988 (cit. on p. 53).

[74] V. Heuveline and R. Rannacher. “A Posteriori Error Control for Finite Element Approximations of Elliptic Eigenvalue Problems”. In: *Advances in Computational Mathematics* 15 (2001), pp. 107–138 (cit. on pp. 61, 62).

[75] E. Hinton, T. Rock, and 0. C. Zienkiewicz. “A note on mass lumping and related processes in the finite element method”. In: *Earthquake Engineering & Structural Dynamics* 4 (1976), pp. 245–249 (cit. on p. 127).

[76] J. Hoffelner, H. Landes, and R. Lerch. “Calculation of Acoustic Streaming Velocity and Radiation Force Based on Finite Element Simulations of Nonlinear Wave Propagation”. In: *Proceedings of IEEE Ultrasonics Symposium* 1 (2000), pp. 585–588 (cit. on p. 95).

[77] J. Hoffelner et al. “Finite Element Simulation of Nonlinear Wave Propagation in Thermoviscous Fluids Including Dissipation”. In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 48.3 (2001), pp. 779–786 (cit. on pp. 94, 95, 101).

[78] Thomas J. R. Hughes. *The Finite Element Method–Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall, Inc, 1987 (cit. on pp. 131, 138).

[79] A. Ibrahimbegovic and E. L. Wilson. “A Modified Method of Incompatible Modes”. In: *Communications in Applied Numerical Methods* 7 (1991), pp. 187–194 (cit. on pp. 132–134).

[80] Y. Jinyun. “Symmetric Gaussian quadrature formulae for tetrahedral regions”. In: *Computer Meth. in Appl. Mech. Eng.* 43 (1984) (cit. on p. 138).

[81] Steven G Johnson. “Notes on Perfectly Matched Layers”. In: *Lecture notes, Massachusetts Institute of Technology* (2008) (cit. on p. 175).

[82] Y. Kagawa et al. “Finite Element Simulation of Nonlinear Sound Wave Propagation”. In: *Journal of Sound and Vibration* 154 (1992), pp. 125–145 (cit. on p. 95).

[83] T. Kane and D. Levinson. *Dynamics: Theory and Applications*. The Internet-First University Press, 2005. URL: `http://dspace.library.cornell.edu/handle/1813/62` (cit. on pp. 189, 190).

[84] Samuel W. Key. *personal communication*. Dec. 2003 (cit. on p. 132).

[85] Tae Soo Kim and Yoo Young Kim. “Mac-based mode-tracking in structural topology optimization”. In: *Comput. and Struct.* 74 (2000), pp. 375–383 (cit. on p. 59).

[86] Kinsler et al. *Fundamentals of Acoustics*. John Wiley & Sons, 1982 (cit. on p. 173).

[87] V. P. Kuznetsov. “Equations of Nonlinear Acoustics”. In: *Sov. Phys. Acoust.* 16 (1971), pp. 467–470 (cit. on pp. 94, 96).

[88] G. F. Lang. “Demystifying Complex Modes”. In: *Sound and Vibration Magazine* 28.8 (1989), pp. 36–40 (cit. on p. 106).

[89] Matts Larsen. “A Posteriori and a Priori Error Analysis for Finite Element Approximations of Self-Adjoint Elliptic Eigenvalue Problems”. In: *SIAM J. Numer. Anal.* 38.2 (2000), pp. 608–625 (cit. on pp. 61–63).

[90] R. Laurenson. “Modal Analysis of Rotating Flexible Structures”. In: *AIAA Journal* 14.10 (1976), pp. 1444–1450 (cit. on p. 189).

[91] T. Laursen and M. Heinstein. “Consistent mesh tying methods for topologically distinct discretized surfaces in nonlinear solid mechanics”. In: *Int. J. Numer. Meth. Engng.* 57 (2003), pp. 1197–1242 (cit. on pp. 82, 87).

[92] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide*. Philadelphia, PA, USA: SIAM, 1998 (cit. on p. 44).

[93] Michael J Lighthill. “On sound generated aerodynamically. I. General theory”. In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 211.1107 (1952), pp. 564–587 (cit. on p. 187).

[94] Abimael F.D. Loula, Thomas J.R. Hughes, and Leopoldo P. Franca. “Petrov-Galerkin formulations of the Timoshenko beam problem”. In: *Computer Meth. in Appl. Mech. Eng.* 63.2 (1987), pp. 115–132. ISSN: 0045-7825 (cit. on p. 147).

[95] R. H. MacNeal. *Finite Elements: Their Design and Performance*. Marcel Dekker, 1994 (cit. on p. 132).

[96] R. H. MacNeal. *The NASTRAN Theoretical Manual*. NASTRAN Theoretical Manual was first published by NASA through COSMIC. NASA no longer maintains NASTRAN and COSMIC no longer exists. Various vendors reproduce this manual with permission from NASA. None, 1972 (cit. on p. 147).

[97] G. Mahan. *Applied Mathematics*. Kluwer Academic Publishers, 2002 (cit. on p. 106).

[98] S. Makarov and M. Ochmann. “Nonlinear and Thermoviscous Phenomena in Acoustics, Part II”. In: *Acustica* 83.2 (1997), pp. 197–222 (cit. on pp. 96, 97, 101).

[99] J. Mandel. “An Iterative Substructuring Method for Coupled Fluid-Solid Acoustic Problems,” in: *J. Comp. Phys.* 177 (2002), pp. 95–116 (cit. on p. 82).

[100] P. J. Matuszyk and L. Demkowicz. “Parametric Finite Elements, Exact Sequences, and Perfectly Matched Layers”. In: *Computational Mechanics* 51.1 (2013), pp. 35–45 (cit. on pp. 178, 179).

[101] Ch Michler et al. “Improving the performance of Perfectly Matched Layers by means of hp-adaptivity”. In: *Numerical Methods for Partial Differential Equations* 23.4 (2007), pp. 832–858 (cit. on p. 176).

[102] Alejandro Mota et al. “Lie-group interpolation and variational recovery for internal variables”. In: *Computational Mechanics* 52 (2013), pp. 1281–1299 (cit. on p. 232).

[103] Matjaž Mršnik, Janko Slavič, and Miha Boltežar. “Frequency-domain methods for a vibration-fatigue-life estimation - Application to real data”. In: *International Journal of Fatigue* 47 (2013), pp. 8–17 (cit. on p. 78).

[104] *MSC support*. URL: `http://support.mscsoftware.com/` (cit. on p. 157).

[105] K. Naugolnykh and L. Ostrovsky. *Nonlinear Wave Processes in Acoustics*. Cambridge University Press, 1998 (cit. on pp. 94, 96).

[106] R. B. Nelson. “Simplified Calculation of Eigenvector Derivatives”. In: *AIAA Journal* 14.9 (1976), pp. 1201–1205 (cit. on pp. 59, 60).

[107] O. O. Ochoa and J. N. Reddy. *Finite Element Analysis of Composite Laminates*. Kluwer Academic Publishers, 1992 (cit. on p. 150).

[108] J. T. Oden and S. Prudhomme. “Error Estimation of Eigenfrequencies for Elasticity and Shell Problems”. In: *Mathematical Models and Methods in Applied Sciences* 13.3 (2003), pp. 323–344 (cit. on pp. 61, 71, 73).

[109] M. A. Hamdi Y. Ousset and G. Verchery. “A Displacement Method for the Analysis of Coupled Fluid-Structure Systems”. In: *Int. J. Numer. Meth. Engng.* 13 (1978), pp. 139–150 (cit. on p. 94).

[110] A. D. Pierce. *Acoustics: An Introduction to Its Physical Principles and Applications*. ASA, 1989 (cit. on pp. 81, 173).

[111] Serge Prudhomme. *personal communication*. Mar. 2004 (cit. on p. 71).

[112] J. S. Przemieniecki. *Theory Of Matrix Structural Analysis*. Dover Publications, 1968 (cit. on p. 147).

[113] Michael A. Puso. “A 3D mortar method for solid mechanics”. In: *Int. J. Numer. Meth. Engng.* 59 (2004), pp. 315–336 (cit. on pp. 82, 87, 89, 224).

[114] J. N. Reddy. *An Introduction to the Finite Element Method*. 1st ed. McGraw Hill, 1984 (cit. on pp. 149–151).

[115] Garth Reese, Rich Field, and Daniel J. Segalman. “A Tutorial on Design Analysis Using von Mises Stress in Random Vibration Environments”. In: *Shock and Vibration. Digest* 32.6 (2000) (cit. on p. 33).

[116] S D Team. *SD – User’s Manual*. Tech. rep. SAND2021-12518. living document with more recent versions. PO Box 5800, Albuquerque, NM 87185-5800: Sandia National Laboratories, 2022 (cit. on p. 227).

[117] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press UK, 1992 (cit. on p. 106).

[118] Daniel J. Segalman. *A Four-Parameter Iwan Model for Lap-Type Joints*. Tech. rep. SAND 2002-3828. Sandia National Laboratories, Nov. 2002 (cit. on p. 74).

[119] Daniel J. Segalman. “A Four-Parameter Iwan Model for Lap-Type Joints”. In: *Journal of Applied Mechanics* 72 (Sept. 2005), pp. 752–760 (cit. on p. 74).

[120] Daniel J. Segalman and Clark R. Dohrmann. *A Method for Calculating the Dynamics of Rotating Flexible Structures- Part II: Example Calculations*. Tech. rep. SAND93-1768J. PO Box 5800, Albuquerque, NM 87185-5800: Sandia National Laboratories, 1993 (cit. on p. 189).

[121] Daniel J. Segalman and Clark R. Dohrmann. *Dynamics of Rotating Flexible Structures by a Method of Quadratic Modes*. Tech. rep. SAND90-2737. PO Box 5800, Albuquerque, NM 87185-5800: Sandia National Laboratories, 1990 (cit. on p. 189).

[122] J. L. Shirron and T. E. Giddings. “A Finite Element Model for Acoustic Scattering from Objects Near a Fluid-Fluid Interface”. In: *Computer Meth. in Appl. Mech. Eng.* 196 (2006), pp. 279–288 (cit. on p. 179).

[123] David O. Smallwood. “An Improved Recursive Formula for Calculating Shock Response Spectra”. In: *Shock and Vibration Bulletin* 51.2 (1981), pp. 211–217 (cit. on p. 75).

[124] David O. Smallwood. “An Improved Recursive Formula for Calculating Shock Response Spectra”. In: *Shock and Vibration Bulletin* 56.1 (1986), pp. 285–287 (cit. on p. 75).

[125] L. H. Soderholm. “On the Kuznetsov Equation and Higher Order Nonlinear Acoustics Equations”. In: *Proc. 15th International Symposium on Nonlinear Acoustics, Göngen* 524.1 (2000), pp. 133–136 (cit. on pp. 94, 96).

[126] R. L. Taylor, P. J. Beresford, and E. L. Wilson. "A Non-conforming Element for Stress Analysis". In: *Int. J. Numer. Meth. Engng.* 10 (1976), pp. 1211–1219 (cit. on pp. 132, 133).

[127] S D Team. *SD Design Manual*. Tech. rep. SAND2021-4312. Sandia National Laboratories, 2021 (cit. on p. 188).

[128] D. Thompson, P. P. Pébay, and J. N. Jortner. *An Exodus II Specification for Handling Gauss Points*. Tech. rep. SAND2007-7169. Sandia National Laboratories, 2007 (cit. on p. 136).

[129] F. Tisseur and Karl Meerbergen. "The Quadratic Eigenvalue Problem". In: *SIAM Rev.* 43.2 (2001), pp. 235–286 (cit. on pp. 105–107, 111).

[130] C. Vanhille, C. Conde, and C. Campos-Pozuelo. "Finite Difference and Finite Volume Methods for Nonlinear Standing Ultrasonic Waves in Fluid Media". In: *Ultrasonics* 42 (2004), pp. 315–318 (cit. on p. 95).

[131] E. L. Wilson and M. Khalvati. "Finite Elements for the Dynamic Analysis of Fluid-Solid System". In: *Int. J. Numer. Meth. Engng.* 19 (1983), pp. 1657–1668 (cit. on p. 94).

[132] Paul H. Wirsching and Mark C. Light. "Fatigue under wide band random stresses". In: *Journal of the Structural Division, ASCE* 106.7 (1980), pp. 1593–1607 (cit. on p. 77).

[133] Paul H. Wirsching, Thomas L. Paez, and Keith Ortiz. *Random Vibrations: Theory and Practice*. Courier Corporation, 2006 (cit. on pp. 76, 210).

[134] Barbara I. Wohlmuth. "A Mortar Finite Element Method Using Dual Spaces for the Lagrange Multiplier". In: *SIAM J. Numer. Anal.* 38.3 (2000), pp. 989–1012 (cit. on pp. 87, 89, 224).

[135] Min Yu, Zhong-Sheng Liu, and Da-Jun Wang. "COMPARISON OF SEVERAL APPROXIMATE MODAL METHODS FOR COMPUTING MODE SHAPE DERIVATIVES". In: *Comput. and Struct.* 62.2 (1996), pp. 301–393 (cit. on pp. 59, 60).

[136] O. C. Zienkiewicz and R. L. Taylor. "The Finite Element Method". In: 4th ed. Vol. 2. McGraw-Hill Book Company Limited, 1991. Chap. 1, pp. 23–26 (cit. on p. 144).

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Technical Library	1911	sanddocs@sandia.gov

Hardcopy—Internal

Number of Copies	Name	Org.	Mailstop
1	Technical Library	1911	0845

This page left blank

Sandia National Laboratories