SANDIA REPORT

SAND2009-0748
Unlimited Release
Printed February 2009

Salinas - Theory Manual
Version 2.8

Garth M. Reese, Timothy F. Walsh, Manoj K. Bhardwaj

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

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

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

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

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

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2009-0748
Unlimited Release
Printed February 2009

Salinas - Theory Manual
Version 2.8

Garth M. Reese
Timothy F. Walsh
Manoj K. Bhardwaj
Computational Solid Mechanics and Structural Dynamics Dept.

Sandia National Laboratories
P. O. Box 5800
Albuquerque, New Mexico 87185-0380

Abstract

Salinas 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 Salinas. For a more detailed description of how to use **Salinas** , we refer the reader to *Salinas, User's Notes*.

Many of the constructs in **Salinas** are pulled directly from published material. Where possible, these materials are referenced herein. However, certain functions in **Salinas** 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.

Contents

1 Solutions . . . 9
1.1 Time integration . . . 9
1.2 Linear transient analysis . . . 9
1.3 Prescribed Accelerations . . . 11
1.4 Nonlinear transient analysis . . . 12
1.5 Explicit Transient Dynamics . . . 14
1.6 Time integration with viscoelastic materials . . . 23
1.7 Linear Eigen Analysis . . . 27
1.8 Random Vibration . . . 27
1.9 Modal Frequency Response Methods . . . 30
1.10 Fast Modal Solutions . . . 33
1.11 Complex Eigen Analysis - Modal Analysis of Damped Structures . . . 38
1.12 SA_eigen . . . 43
1.13 Quadratic Modal Superposition . . . 45
1.14 Component Mode Synthesis . . . 50
1.15 A posteriori error estimation for eigen analysis . . . 55
2 Elements . . . 71
2.1 Isoparametric Solid Elements. Selective Integration . . . 71
2.2 Implementation . . . 73
2.3 Mean Quadrature Element with Selective Deviatoric Control . . . 73
2.4 Bubble Element . . . 74
2.5 Quadratic Isoparametric Solid Elements . . . 79
2.6 Wedge elements . . . 79
2.7 Tet10 elements . . . 80
2.8 Calculating shape functions and gradients of the Hex20 element . . . 80
2.9 Anisotropic Elasticity . . . 81
2.10 Triangular Shell Element . . . 81
2.11 Triangular Shell - Tria3 . . . 84
2.12 Two Node Beam . . . 84
2.13 Truss . . . 84
2.14 Springs . . . 85
2.15 Gap Elements . . . 86
2.16 Multi-Point Constraints, MPCs . . . 86
2.17 Rigid Elements . . . 87
2.18 Shell Offset . . . 92
2.19 Notes on Consistent Loads Calculations . . . 93
2.20 Coordinate Systems . . . 97
2.21 Constraint Transformations in General Coordinate Systems . . . 98
2.22 HexShells . . . 102
2.23 Membrane . . . 103
3 Loadings . . . 107
3.1 Matrices from Applied Forces . . . 107
3.2 Theory of Modal Analysis of Rotating Structures . . . 107
4 Linear Algebra Issues . . . 111
4.1 Solution Spaces . . . 111
4.2 Matrix Dimensions: Revision . . . 113

4.3 Rotational Degrees of Freedom 114
4.4 Orthogonality of MPC to Rigid Body Vectors 117
4.5 Mass Properties 119
5 Constraints and Contact 122
5.1 Tied Friction 122
5.2 Mortar Methods 122
References 126

Appendix

A Anisotropic Materials 127
A.1 Linear Anisotropic Elasticity 127
A.2 Stress Vectors 127
A.3 Strain Energy and Orientation 129
B Integration of Isoparametric Solids 132
B.1 Uniform Strain-Displacement Matrices 133
B.2 Mixed Integration 134
C MSC documentation of Nastran's RBE3 element 135
C.1 Abstract: Mathematical Specification of the Modern RBE3 Element 135
C.2 Abstract: RBE3 ELEMENT CHANGES IN VERSION 70.7 141
D Theory Notes for Acoustics and Structural Acoustics 144
D.1 Conforming and Nonconforming Linear Structural Acoustics 144
D.2 The Governing Equations and Their Discretizations 145
D.3 Conforming Structural Acoustics 145
D.4 Nonconforming Structural Acoustics 147
D.5 Acoustic Scattering 151
D.6 Absorbing Boundaries 154
Index 157

Figures

1 Simple constraint problem. 20
2 Comparison of Modal Displacement, Acceleration and DFRF 34
3 Standard Modal Transient Algorithm 35
4 Fast Modal Transient Algorithm 37
5 Fast Modal Frequency Response Algorithm 37
6 Complex EigenVector orthogonalization 51
7 Eigenvalue and Eigenvector corrections of Craig-Bampton reduced models 56
8 Equilibration of loads 90
9 Original, and rotated coordinate frames 98
10 Node Constrained Directly to Beam. 118
11 Node Constrained Offset to Beam. 118
D.1 Nonconforming Mesh of 2 Acoustic Domains 149
D.2 A node-face interaction on the structural acoustic interface. 149
D.3 Ghosting of Acoustic DOFS on the Structural Interface 152

Tables

1 Sources of Damping in the Solution 15
2 Potential Basis Functions for Subdomain Reduction 44
3 Comparison of deflections at Node 2 83
4 Comparison of deflections at Node 3 83
5 Salinas solution spaces 113

1 Solutions

One thing which makes **Salinas** somewhat unique among the many mechanics codes developed at *Sandia National Labs* is that **Salinas** combines a variety of different solution procedures. These range from modal superposition based solutions to nonlinear transient. As described in the *User's Notes* , these solutions can be combined (or chained) in solution cases. This section of the manual describes the theory behind these individual solutions. For details about particular finite elements, see section 2.

1.1 Time integration

For linear and nonlinear transient dynamics, we use a variant of the Newmark-Beta time integrator called the generalized alpha method.

1.2 Linear transient analysis

The equations of motion of the structure are

$$\begin{aligned} M\left[(1-\alpha_m)a_{n+1}+\alpha_m a_n\right] &+ \hat{C}\left[(1-\alpha_f)v_{n+1}+\alpha_f v_n\right]+ \\ K\left[(1-\alpha_f)d_{n+1}+\alpha_f d_n\right] &= (1-\alpha_f)F^{ext}(t_{n+1})+\alpha_f F^{ext}(t_n) \end{aligned} \tag{1.1}$$

where F^{ext} is the external load, α_f, α_m are the integration parameters for the generalized α method, and $\hat{C} = C + \alpha M + \beta K$. That is, the damping matrix is the sum of the standard damping matrix C plus the proportional damping terms.

In order to achieve second order accuracy and unconditional stability, we must satisfy the following conditions.

$$\begin{aligned} \alpha_m < \alpha_f <= \frac{1}{2} \\ \gamma_n = \frac{1}{2} - \alpha_m + \alpha_f \\ \beta_n \geq \frac{1}{4} + \frac{1}{2}(\alpha_f - \alpha_m) \end{aligned} \tag{1.2}$$

The code automatically computes these parameters such that they meet these criteria. Specifically,

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

We note some special cases of interest. If $\rho = 0$, we have that $\alpha_f = 0$ and $\alpha_m = -1$. This is the maximum damping case. If $\rho = 1$, we have that $\alpha_f = \alpha_m = \frac{1}{2}$, which yields $\beta_n = \frac{1}{4}$, and $\gamma_n = \frac{1}{2}$. This is similar to the classical undamped Newmark-beta method, although we note that it is a different algorithm since $\alpha_f = \alpha_m = \frac{1}{2}$ implies some lagging in the time-stepping procedure. The classical undamped Newmark-beta method has $\alpha_f = \alpha_m = 0$.

For later use, we also define

$$F^{ext}_{n+1+\alpha_f} = (1-\alpha_f)F^{ext}(t_{n+1}) + \alpha_f F^{ext}(t_n) \tag{1.3}$$

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

The time integration scheme is defined as follows

$$\begin{aligned} d_{n+1} &= d_n + \Delta t v_n + \frac{\Delta t^2}{2}\left[(1-2\beta_n)a_n + 2\beta_n a_{n+1}\right] \\ v_{n+1} &= v_n + \Delta t\left[(1-\gamma_n)a_n + \gamma_n a_{n+1}\right] \end{aligned} \tag{1.4}$$

where γ_n, β_n are the integration parameters for the Newmark method. In order to have a displacement-based method, we solve these equations for the acceleration and velocity in terms of displacement, which yields

$$\begin{aligned} a_{n+1} &= \frac{1}{\beta_n \Delta t^2}\left[d_{n+1} - d_n - v_n \Delta t\right] - \frac{1-2\beta_n}{2\beta_n} a_n \\ v_{n+1} &= v_n + \Delta t\left[(1-\gamma_n)a_n + \gamma_n a_{n+1}\right] \\ &= v_n + \Delta t\left[(1-\gamma_n)a_n + \frac{\gamma_n}{\beta_n \Delta t^2}\left[d_{n+1} - d_n - v_n \Delta t\right] - \gamma_n \frac{1-2\beta_n}{2\beta_n} a_n\right] \end{aligned} \tag{1.5}$$

Substituting these equations into the equation of motion, and collecting terms, we obtain

$$\begin{aligned} &\left[M\frac{(1-\alpha_m)}{\beta_n \Delta t^2} + \hat{C}(1-\alpha_f)\frac{\gamma_n}{\beta_n \Delta t} + K(1-\alpha_f)\right] d_{n+1} = \\ &F^{ext}_{n+1+\alpha_f} - K\alpha_f d_n \\ &-\hat{C}\left[\alpha_f v_n + (1-\alpha_f)\left[v_n + \Delta t(1-\gamma_n)a_n + \frac{\gamma_n}{\beta_n \Delta t}\left[-d_n - \Delta t v_n\right] - \frac{\gamma_n \Delta t(1-2\beta_n)}{2\beta_n} a_n\right]\right] \\ &+M\left[-\alpha_m a_n + \frac{1-\alpha_m}{\beta_n \Delta t^2}\left[d_n + v_n \Delta t\right] + (1-\alpha_m)\frac{1-2\beta_n}{2\beta_n} a_n\right] \end{aligned} \tag{1.6}$$

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.

1.3 Prescribed Accelerations

Prescribed accelerations can be applied in Salinas to nodesets or sidesets, as described in the users 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 degree of freedom 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 degrees of freedom is simply a matter of an external loop.

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

$$\begin{aligned} a(t) &= a_o f(t) \\ v(t) &= v_0 + \int_0^t a(t) = v_0 + \int_0^t a_o f(t)dt = v_0 + a_o(if(t)) \\ d(t) &= d_0 + \int_0^t v(t)dt = d_0 + v_0 t + \int_0^t \int_0^t a_o f(t)dt = d_0 + v_0 t + a_o(iif(t)) \end{aligned} \tag{1.7}$$

where we have defined $if(t)$ and $iif(t)$ to denote the first and second integrals of the function $f(t)$, and d_0 and v_0 denote the initial displacement and velocity. $if(t)$ and $iif(t)$ are computed numerically in Salinas.

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 1.1. We put subscripts of g on the system matrices and right hand side to denote that they do not yet have prescribed BCs condensed out (hence are Gset).

$$\begin{aligned} M_g[(1-\alpha_m)a_{n+1} + \alpha_m a_n] &+ \hat{C}_g[(1-\alpha_f)v_{n+1} + \alpha_f v_n] + \\ K_g[(1-\alpha_f)d_{n+1} + \alpha_f d_n] &= (1-\alpha_f)F_g^{ext}(t_{n+1}) + \alpha_f F_g^{ext}(t_n) \end{aligned} \tag{1.8}$$

Next, we condense out the prescribed degrees of freedom and move the contributions to the right hand side. We note that degrees of freedom that are fixed do not contribute to the right hand side. After this process, we remove the subscripts from the system matrices, since they are now in Aset

form. We also condense the right hand side terms, so that everything is Aset.

$$
\begin{aligned}
M\,[(1-\alpha_m)a_{n+1}+\alpha_m a_n] \quad &+ \quad \hat{C}\,[(1-\alpha_f)v_{n+1}+\alpha_f v_n]+ \\
K\,[(1-\alpha_f)d_{n+1}+\alpha_f d_n] \quad & \\
&= \quad (1-\alpha_f)F^{ext}(t_{n+1})+\alpha_f F^{ext}(t_n) \\
&- \quad (1-\alpha_f)a_o\,[f(t_{n+1})m_i+if(t_{n+1})c_i+iif(t_{n+1})k_i] \\
&- \quad \alpha_f a_o\,[f(t_n)m_i+if(t_n)c_i+iif(t_n)k_i]
\end{aligned}
\tag{1.9}
$$

This shows that prescribed accelerations result in a contribution to the right hand side that consists of products of the time function $f(t)$ with the column from the mass matrix corresponding to the prescribed dof, and products of the first and second integrals of $f(t)$ with the corresponding columns from the damping and stiffness matrices. For statics problems, this procedure reduces to only a contribution from the stiffness matrix, and this is also included in Salinas.

1.4 Nonlinear transient analysis

This section follows closely the nonlinear transient procedure given by Belytschko et al,[1] with the modification of using the generalized alpha integrator rather than the Newmark beta approach. In the case of a nonlinear transient analysis, the equation of motion is

$$
\begin{aligned}
M\,[(1-\alpha_m)a_{n+1}+\alpha_m a_n] \quad &+ \quad \hat{C}\,[(1-\alpha_f)v_{n+1}+\alpha_f v_n]+ \\
(1-\alpha_f)F^{int}_{n+1}+\alpha_f F^{int}_n \quad &= \quad (1-\alpha_f)F^{ext}(d_{n+1})+\alpha_f F^{ext}(d_n)
\end{aligned}
\tag{1.10}
$$

where F^{int}_{n+1} and F^{int}_n are the internal forces at the current and previous time steps, respectively. Note that we have written the external loads as functions of displacement, since in the most general case they could be follower loads.

Before proceeding, we note that there are two possible approaches for implementing the generalized alpha method, and in equation 1.10 we have taken one of these approaches. The difference lies in the treatment of the internal and external forces. The first approach is to evaluate them as follows

$$
\begin{aligned}
F^{int}_{n+1+\alpha_f} &= F^{int}((1-\alpha_f)d_{n+1}+\alpha_f d_n) \\
F^{ext}_{n+1+\alpha_f} &= F^{ext}((1-\alpha_f)d_{n+1}+\alpha_f d_n)
\end{aligned}
\tag{1.11}
$$

and the second is to evaluate two separate terms

$$
\begin{aligned}
F^{int}_{n+1+\alpha_f} &= (1-\alpha_f)F^{int}(d_{n+1})+\alpha_f F^{int}(d_n) \\
F^{ext}_{n+1+\alpha_f} &= (1-\alpha_f)F^{ext}(d_{n+1})+\alpha_f F^{ext}(d_n)
\end{aligned}
\tag{1.12}
$$

When both F^{ext} and F^{int} are linear functions, the two approaches are identical. For nonlinear problems, both F^{ext} and F^{int} could be nonlinear functions, and thus the two procedures are different. In the limit of very small time steps, these nonlinear functions effectively linearize and the two approaches again become the same. Thus the limiting behavior of the two approaches is the same.

We note that in most cases, the external load F^{ext} is treated as a piecewise linear function of time, and in those cases the two approaches yield the same result for the external load, though a couple of exceptions are worth mentioning. First, if two consecutive time steps lie within two different linear segments, then the two approaches above yield different loads. Second, although they are seldom used, polynomial and loglog interpolation functions are available in Salinas in addition to the commonly used linear interpolation, and in those cases different load vectors result from the above procedures. For problems with very large time steps and involving polynomial interpolation, different results are to be expected.

In Salinas we have chosen the second option, which evaluates both the internal force and external force at both times of interest, and forms a linear combination of the two. Comparisons have shown little difference in the results on simple test problems.

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

$$F^{int}_{n+1} = F^{int}_{n} + K_t \Delta d \tag{1.13}$$

where K_t is the tangent stiffness matrix, defined as $K_t = \partial F_{internal}/\partial u$, and $\Delta d = d_{n+1} - d_n$. Also, we use equations 1.5, which are the same as in the linear case.

First, we substitute equations 1.5 and 1.13 into equation 1.10. 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

$$\left[M\frac{(1-\alpha_m)}{\beta_n\Delta t^2}+\hat{C}(1-\alpha_f)\frac{\gamma_n}{\beta_n\Delta t}+K_t(1-\alpha_f)\right]\Delta d = \\ F^{ext}_{n+1+\alpha_f}-(1-\alpha_f)\hat{F}^{int}-\alpha_f F^{int}_n - C\left[(1-\alpha_f)\hat{v}+\alpha_f v_n\right] \\ -M\left[(1-\alpha_m)\hat{a}+\alpha_m a_n\right] \tag{1.14}$$

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 $\hat{F}^{int} = F^{int}_n + K_t(\hat{d}-d_n)$.

Upon using the Newmark beta time integrator ($\gamma_n = \frac{1}{2}$, $\beta_n = \frac{1}{4}$, $\alpha_f = \alpha_m = 0$, equation 1.14 reduces to

$$\left[M\frac{4}{\Delta t^2}+\hat{C}\frac{2}{\Delta t}+K_t\right]\Delta d = F^{ext}_{n+1}-\hat{F}^{int}-C\hat{v}-M\hat{a} \tag{1.15}$$

which is the same equation given by Belytschko et al.[1]

We note that equation 1.14 can be written as

$$A\Delta d = res \tag{1.16}$$

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 1.10) are not satisfied by the current iterate.

1.4.1 Damping in Nonlinear Solutions

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

1.5 Explicit Transient Dynamics

An transient dynamics capability using an explicit integrator has been developed for specialized applications. Note that Salinas remains a small strain application, even when using the explicit integrator. This integrator is used because it may be advantageous when interfacing with other applications which control the time step. The implicit integrator requires no linear solve of the stiffness matrix, and does not require a new factorization when the time step changes. It can be used with both linear and nonlinear elements.

Damping Source	**Discussion**
linear dashpots	Contributes directly to the C matrix described in equation 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 1. Sources of Damping in the Solution

1.5.1 Central Difference Operator

Consider the following equation for a spatially discretized finite element system in motion:

$$\mathbf{Ku} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{f}_{ext} \tag{1.17}$$

In the above equation, $\mathbf{u}$ represents the displacement vector, $\mathbf{K}$ represents the matrix of stiffness terms, $\mathbf{C}$ represents the matrix of damping terms, and $\mathbf{M}$ represents a matrix of mass terms. The vector $\mathbf{f}_{ext}$ is calculated from a system of applied loads.

The above equation of motion, Equation 1.17, is a system of ordinary differential equations with constant coefficients. Difference expressions can be used to approximate the velocities and accelerations in terms of the displacements appearing in the equation. A commonly used difference expression is the central difference operator. The central difference operator is as follows:

$$a_n = \frac{\frac{u_{n+1}-u_n}{\Delta t^{n+1/2}} - \frac{u_n - u_{n-1}}{\Delta t^{n-1/2}}}{(\Delta t^{n+1/2} + \Delta t^{n-1/2})/2} \tag{1.18}$$

In the above equation, $n+1$ denotes information at time t^{n+1}, n denotes information at time t^n, and $n-1$ denotes information at time t^{n-1}. The increment in time from t^n to t^{n+1} is $\Delta t^{n+1/2}$, and the increment in time from t^{n-1} to t^n is $\Delta t^{n-1/2}$. The term a_n is an acceleration value in the vector $\ddot{\mathbf{u}}$. The expression

$$\frac{u_{n+1} - u_n}{\Delta t^{n+1/2}} \tag{1.19}$$

is the velocity, $v_{n+1/2}$, at the half time step $\Delta t^{n+1/2}$. The term $v_{n+1/2}$ is a velocity value in the vector $\dot{\mathbf{u}}$. The expression

$$\frac{u_n - u_{n-1}}{\Delta t^{n-1/2}} \tag{1.20}$$

is the velocity, $v_{n-1/2}$, at the half time step $\Delta t^{n-1/2}$. The velocity is constant over a time step.

When a solution is known at time t^n and time t^{n-1}, the solution can be determined at time t^{n+1} from Equation 1.18, the central difference operator. We use the previous information to project the solution to time t^{n+1}. To understand how we project the solution ahead to time t^{n+1}, we return to the equation of motion. We use the equation of motion without the damping matrix to simplify our discussion. The equation of motion at time t^n is

$$\mathbf{Ku}_n + \mathbf{M}\ddot{\mathbf{u}}_n = \mathbf{f}_n^{ext}\,. \tag{1.21}$$

In the above equation, the product $\mathbf{Ku}_n$ is simply the internal force vector at time t^n. The above

equation of motion reduces to

$$\mathbf{M}\ddot{\mathbf{u}}_n = \mathbf{f}_n^{ext} - \mathbf{f}_n^{int} \,. \tag{1.22}$$

The acceleration vector at time t^n is calculated from

$$\ddot{\mathbf{u}}_n = \mathbf{M}^{-1}(\mathbf{f}_n^{ext} - \mathbf{f}_n^{int}) \,. \tag{1.23}$$

Now that we have the acceleration at time t^n, we can compute the velocity at the half time step $t^{n+1/2}$ and the displacement at the time step t^{n+1} with the following equations:

$$(v_{n+1/2})_i = (v_{n-1/2})_i + (a_n)_i(\Delta t^{n+1/2} + \Delta t^{n-1/2})/2 \tag{1.24}$$

$$(u_{n+1})_i = (u_n)_i + (v_{n+1/2})_i \Delta t^{n+1/2} \tag{1.25}$$

In the above equations, the subscript i denotes quantities associated with the ith degree of freedom. Once the vector $\mathbf{u}_{n+1}$ has been calculated, we can again advance the time step.

It is important to note that the central difference operator is conditionally stable. If the time step Δt exceeds the value $2/\sqrt{\lambda^2}$, where λ^2 is the maximum eigenvalue determined by the eigenvalue problem

$$\mathbf{K}\phi - \lambda^2 \mathbf{M}\phi = 0 \,, \tag{1.26}$$

the problem becomes unstable.

Typically, the mass matrix for an explicit, transient dynamics code is diagonalized (See Reference[2]). When the mass matrix is diagonalized, the acceleration for each degree of freedom can be written simply as

$$(a_n)_i = (f_n^{ext} - f_n^{int})_i/(m)_i \,. \tag{1.27}$$

The diagonalization is done for purposes of performance. When the mass matrix is diagonalized, the application of kinematic boundary conditions and certain constraints becomes extremely simple, and no linear solves are required.

Note that, in our above description of the implementation of the explicit scheme, if we include damping, the damping matrix $\mathbf{C}$ times the velocity vector produces a damping force vector that is added to the right hand side of Equation 1.22.

Now that we have outlined the basics of an explicit solution technique, we will consider how some of the basic functionality – kinematic boundary conditions, constraints, tied surfaces, and superelements – are implemented for an explicit solver.

1.5.2 Mass Matrix Solutions

A diagonal mass matrix simplifies the explicit integration in several ways. Most important of these is that there is no need for a linear solve as each degree of freedom is uncoupled from the rest. Speed of the solution is critical as the conditionally stable time step can require very small iterations. Other factors, such as implementation of constraints and boundary conditions, may also be affected by the form of the mass matrix. In addition, more accurate results for explicit integrators are obtained by using a lumped mass matrix. (For an implicit scheme, the more accurate results are obtained by using a consistent mass. See Reference.[3])

Salinas is designed around an implicit iteration scheme and powerful linear solvers are available in the package. There are several reasons to consider an approach where we do not require that the mass matrix be fully diagonal.

1. Superelements generated by Craig-Bampton type reductions contain full mass matrices. Since the mass matrix provides all the coupling to the generalized degrees of freedom, standard lumping approaches cannot be used. Several other approaches are available including reformulating the superelement (as is done in Abaqus), or other coordinate transformations that simplify the solution. The most straightforward approach is to solve the linear system for those coupled degrees of freedom.

2. Like super elements, inertias associated with rotating masses may not be easily lumped. These are typically 6x6 matrices, so existing codes typically handle these as a special case.

3. Elements such as beams may have mass terms that can be easily lumped in the element coordinate frame. Lumping in an arbitrary rotated frame may cause a dependence of the solution on rotation. This comes about because the rotational inertia for a drilling degree of freedom differs from that in bending. This is addressed in a variety of ways in different codes. For example, Nastran usually eliminates the mass of rotational dofs in beams. Presto insures that all rotational inertias are identical. In the limit as the element size goes to zero, these produce the same solution. However, maintaining a tridiagonal inertia could greatly reduce changes to existing code base and permit ready comparison with implicit solutions.

Recognizing the need for a rapid solution at each time step we propose lumping the mass matrix where feasible, but solving equation 1.23 for the remaining mass terms. Solid elements will have diagonal mass terms, shells and beams will be tridiagonal, mass elements will be 6x6 and superelement mass matrices will depend on the element.

Discussions with our linear solver folks indicate that these solves should be extremely fast. In most cases there will be little or no coupling outside the subdomain, so a sparse direct backsolve is all that is required at each time step. The solver preconditioner will be tuned for these special characteristics. We expect the linear solve to be much less expensive than the computation of internal forces.

It is important that this solution strategy be compatible with follow on approaches that may not use a full linear solve. We see no incompatibilities with the exception of the element formulations for diagonal versus partially lumped mass matrices. For UC-2, details of applying superelements without a system solve are to be addressed later.

1.5.3 Kinematic Boundary Conditions

A wide variety of kinematic boundary conditions can be implemented for an explicit solution technique. These boundary conditions are similar to those that can be found in an implicit code – fixed displacements, fixed rotations, prescribed displacement, etc. For the problem formulation in Equation 1.23, kinematic boundary conditions are enforced by adding reaction forces to the right-hand side. The reaction forces are such that the acceleration at time t^n results in the desired kinematic behavior at time t^{n+1}.

Suppose, for example, we want to fix the displacement component $(u)_i$ for all time. If degree of freedom i is associated with a diagonalized mass, we can enforce the boundary condition by adding a reaction force, $(f_n^{react})_i$ at each step n that is equal and opposite to the residual term $(f^{resid})_i$, where $(f^{resid})_i = (f_n^{ext} - f_n^{int})_i$. The right hand side term becomes

$$(f_n^{react})_i + (f^{resid})_i = 0\,, \tag{1.28}$$

and the acceleration term at time n also becomes zero. For this diagonalized mass case, the acceleration is simply $0/(m)_i$.

As a second example, suppose we want the velocity at the half time step $\Delta t^{n+1/2}$ for component i to have a value of v_b. The velocity at the half time step $t^{n-1/2}$ for component i has a value of v_a. Again, assume degree of freedom i is associated with a diagonal mass term $(m)_i$. Consider Equation 1.23. The acceleration, $(a_p)_i$, required to produce the prescribed velocity at the half time step $\Delta t^{n+1/2}$ is

$$(a_p)_i = \frac{v_b - va}{(\Delta t^{n+1/2} + \Delta t^{n-1/2})/2}\,. \tag{1.29}$$

If we add $(-f_n^{resid})_i + (m)_i(a_p)_i$ to the residual term $(f^{resid})_i$, then the acceleration component i at time t^n becomes

$$\frac{(f_n^{resid})_i - (f_n^{resid})_i + (m)_i(a_p)_i}{(m)_i}\,, \tag{1.30}$$

which is simply the value $(a_p)_i$ that produces the prescribed velocity v_b at the half time step.

As can be seen from the above examples, each kinematic boundary condition would require its own unique set of reaction forces to enforce the correct kinematic behavior.

1.5.4 Constraints

Constraints can be enforced by applying forces that enforce the constraint conditions. We demonstrate this by use of a specific example. Consider the simple linkage shown in Figure 1, which is used to enforce the constraint

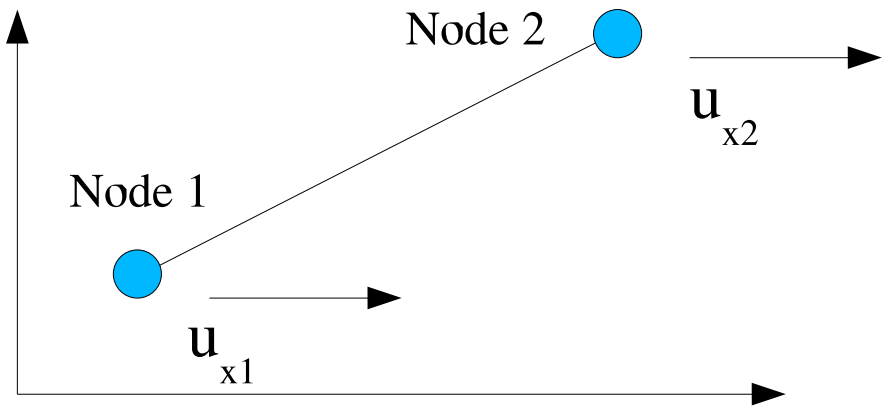


Figure 1. Simple constraint problem.

$$u_{x2} - u_{x1} = 0\,, \tag{1.31}$$

where u_{x2} is the displacement component in the x-direction at node 2 and u_{x1} is the displacement component in the x-direction at node 1. Suppose we use the equation for acceleration, Equation 1.27, and Equations 1.24 and 1.25 to obtain displacements for nodes 1 and 2 at time t^{n+1}; the displacements at nodes 1 and 2 are $(u_{n+1})_{x1}$ and $(u_{n+1})_{x2}$, respectively. The two displacements, $(u_{n+1})_{x1}$ and $(u_{n+1})_{x2}$, do not necessarily satisfy the constraint condition. The two displacements may give

$$(u_{n+1})_{x2} - (u_{n+1})_{x1} = g\,, \tag{1.32}$$

where $g \neq 0$. If we increment the displacement $(u_{n+1})_{x1}$ by βg and decrement the displacement $(u_{n+1})_{x2}$ by $(1-\beta)g$, then the constraint condition, Equation 1.31 is satisfied. We can adjust the displacements at nodes 1 and 2 by applying constraint forces. If we apply a force of

$$f_{x1} = \beta g m_1 / \mathbf{D}t \tag{1.33}$$

to node 1, this will result in a displacement increment of βg. In the above equation, m_1 is the mass at node 1 and

$$\mathbf{D}t = \frac{\Delta t^{n+1/2}}{(\Delta t^{n+1/2} + \Delta t^{n-1/2})/2}\,. \tag{1.34}$$

A force of

$$f_{x2} = -(1-\beta) g m_2 / \mathbf{D}t \tag{1.35}$$

at node 2 results in a displacement decrement of $-(1-\beta)g$. In the above equation, m_2 is the mass at node 2. By adding these constraint forces to the external force vector in Equation 1.23, we add forces to help satisfy the constraint condition as we step forward with the solution.

At this point, we have only to determine the value of β. The value of β can be determined by the use of conservation of momentum. The equation for conservation of momentum for this system is

$$m_1 v_{1i} + m_2 v_{2i} = m_1 v_{1f} + m_2 v_{2f}\,, \tag{1.36}$$

where v_{1i} is the initial velocity at node 1, v_{2i} is the initial velocity at node 2, v_{1f} is the final velocity at node 1, and v_{2f} is the final velocity at node 2. By initial, we mean before the application of the constraint force to enforce the constraint conditions. Final means after the application of the constraint force. The velocity terms are all in the x-direction. If we account for the constraint forces, the change in the velocity for node 1 becomes

$$m_1 \beta g/\Delta t^{n+1/2} \tag{1.37}$$

and the change in velocity for node 2 becomes

$$-m_2(1-\beta)g/\Delta t^{n+1/2}\,. \tag{1.38}$$

The conservation of momentum equation becomes

$$m_1 v_{1i} + m_2 v_{2i} = m_1(v_{1i} + \beta g/\Delta t^{n+1/2}) + m_2(v_{2i} - (1-\beta)g/\Delta t^{n+1/2})\,. \tag{1.39}$$

From the preceding equation we can solve for β and obtain

$$\beta = m_2/(m_1 + m_2)\,. \tag{1.40}$$

for the case of $m_1 = m_2$, the value for β is $1/2$, which is the expected result. Each node would move by $g/2$ to meet the constraint condition.

We can generalize the above process. The above approach can handle general constraint conditions such as

$$C_0 u_0 - \sum_{i=1,n}^{n} C_i u_i\,, \tag{1.41}$$

where $u_i, i = 0,1,2,...,n$ are displacement degrees of freedom and $C_i, i = 0,1,2,...,n$ are constraint coefficients. The constraint coefficients depend on the nature of the kinematic constrains and geometry. The above approach can also handle constraint conditions involving nodes with off-diagonal mass terms.

For conditions where constraints interact, an iterative process is required to calculated the set of forces required to enforce the constraints.

1.5.5 Contact with Tied Surfaces

Contact refers to the interaction of one or more bodies when they physically touch. This can include the interaction of one part of a surface against another part of the same surface, the surface of one body against the surface of another body, and so forth. Contact capabilities are provided in Sierra applications by the ACME module (Reference[4]). The contact algorithms in ACME are designed to ensure that surfaces do not inter-penetrate in a non-physical way, and that the surface behavior is computed correctly according to any user-specified surface model. For an explicit solver, ACME uses a two-step process. The first step is the detection of the overlap of surfaces. The second step is an enforcement phase to remove the overlap. Enforcement is accomplished with a kinematic approach rather than a penalty approach. In the kinematic approach, a set of constraint equations is calculated based on the initial penetration of one surface by another. The constraint equations are used to calculate contact forces to remove the inter-penetration of the surfaces. (A penalty approach can be thought of as introducing "stiff" springs between contact surfaces as a means of preventing inter-penetration. The spring forces reduce the overlap to some small tolerance.)

One of the options in ACME is tied surfaces. For the tied surface option, a node on a surface maintains its relative position on an opposing surface as the two surfaces deform. For tied surfaces, the detection phase is used initially to determine a set of initial constraint conditions. The enforcement phase uses these initial constraint conditions throughout the time history for the problem.

1.5.6 Superelements

Superelements consist of a reduced stiffness matrix, $\mathbf{K}_R$, and associated reduced mass matrix, $\mathbf{M}_R$. The superelement can include both interface (physical) degrees of freedom and generalized degrees of freedom. (The generalized degrees of freedom can be used to carry "extra" information about the superelement, such as information about behavior at high frequencies.) As an approach to using a super element with an explicit solver, the reduced mass matrix can be assembled into the mass matrix appearing on the left-hand side of Equation 1.22. At each time step n, we can compute the internal forces, $(\mathbf{f}_n^{int})_R$, for the superelement. The internal forces for the superelement are defined by

$$(\mathbf{f}_n^{f\,int})_R = \mathbf{K}_R(\mathbf{u}_n)_R\,, \tag{1.42}$$

where $(\mathbf{u}_n)_R$ is the displacement vector associated with the superelement degrees of freedom at time n. The internal forces associated with the superelement must be assembled into the f_n^{int} vector on the right-hand side of Equation 1.22.

1.5.7 Stable Time Step

There are two means to arrive at a stable time step.

1. The time step relates to the maximum eigenvalue of the system, $\tau = 2/\omega_{max}$, where ω_{max}^2 is the largest eigenvalue of the system.

$$(\mathbf{K} - \omega_{max}^2 \mathbf{M})\phi = 0 \tag{1.43}$$

2. an element by element method. The stable step relates to the shortest time for the signal to pass through the model.

The system level calculation is more expensive, but is the more accurate. Eigenvalues may be computed using the Lanczos method which is included in the ARPACK package. This is already being used within Salinas, but the time step calculation computes only the highest eigenvalue.

From the ARPACK documentation for DSAUPD, we are looking for a solution with `WHICH='LA'` to compute the largest eigenvalues, and with `method=2`. The "B" matrix is 'G', for a generalized eigen problem. The operator required is `inv(M)*K`.

1.6 Time integration with 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.

1.6.1 Equations of motion

The equations of motion of elastodynamics in three dimensions are given by

$$u_{tt} - \nabla \cdot \sigma = f(x,t) \quad \Omega \tag{1.44}$$
$$u(x,t) = 0 \quad x \in \Gamma_D \tag{1.45}$$
$$\sigma(x,t) = g(x,t) \quad x \in \Gamma_N \tag{1.46}$$
$$\tag{1.47}$$

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 Dirchlet Γ_D and Neumann Γ_N subregions.

The Dirichlet conditions lead to the space of admissible functions

$$V = \left[v \in H^1(\Omega), v(x) = 0, x \in \Gamma_D\right] \tag{1.48}$$

The equation of motion, along with boundary conditions, is cast into the weak form in the standard way

$$\int_\Omega u_{tt} \cdot v + \int_\Omega \sigma \cdot \nabla_s v dx = \int_\Omega f(x,t) \cdot v dx + \int_{\Gamma_N} g(x,t) \cdot v ds \quad \forall v \in V \tag{1.49}$$

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.

1.6.2 Constitutive equations

The representation of the time-dependent moduli for a viscoelastic material is commonly written in the form of a Prony series

$$G(t) = G_{\text{inf}} + (G_0 - G_{\text{inf}})\zeta_G(t) \tag{1.50}$$

$$\zeta_G(t) = \sum_i c_i e^{-\frac{t}{s_i}} \tag{1.51}$$

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

$$\sigma = D\varepsilon = (KD_K + GD_G)\varepsilon \tag{1.52}$$

where D_K, D_G are given in equation 9.4.7 in,[5] and K, G are the bulk and shear moduli. This constitutive law can be generalized to a linear viscoelastic material as follows

$$\begin{aligned}\sigma(x,t) = (G_0 - G_{\text{inf}})D_G \int_0^t \zeta_G(x,t-\tau)\frac{\partial\varepsilon(x,\tau)}{\partial\tau}d\tau + G_{\text{inf}}D_G\varepsilon(x,t) + \\ (K_0 - K_{\text{inf}})D_K \int_0^t \zeta_K(x,t-\tau)\frac{\partial\varepsilon(x,\tau)}{\partial\tau}d\tau + K_{\text{inf}}D_K\varepsilon(x,t)\end{aligned} \tag{1.53}$$

The above expression is then used to represent the stress in the weak form of the equations of motion, 1.49.

Given a finite dimensional subspace $V_h \subset V$, we represent the approximate solution in the standard way

$$u_h(x,t) = \sum_{i=1}^{n} \phi_i(x)\eta_i(t) \tag{1.54}$$

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 ith column. Inserting this into the equations of motion, and rearranging, we obtain

$$\begin{aligned}M\ddot{\eta}(t) + (G_0 - G_{\text{inf}})K_1 \int_0^t \zeta_G(t-\tau)\dot{\eta}(t)d\tau + \\ (K_0 - K_{\text{inf}})K_1 \int_0^t \zeta_K(t-\tau)\dot{\eta}(t)d\tau + K_2\eta(t) = f(t)\end{aligned} \tag{1.55}$$

where

$$M = \int_\Omega \rho(x)\Phi^T(x)\Phi(x)dx \tag{1.56}$$

is the mass matrix,

$$K_1 = (G_0 - G_{\text{inf}}) \int_\Omega B^T D_G B dx + (K_0 - K_{\text{inf}}) \int_\Omega B^T D_K B dx \tag{1.57}$$

$$K_2 = G_{\text{inf}} \int_\Omega B^T D_G B dx + K_{\text{inf}} \int_\Omega B^T D_K B dx \tag{1.58}$$

are the stiffness matrices, and

$$f(t) = \int_\Omega f(x,t) \cdot v(x) dx + \int_{\Gamma_N} g(x,t) \cdot v(x) ds \tag{1.59}$$

is the right hand side. The corresponding element matrices are defined simply by breaking the integrals into elementwise contributions.

Equation 1.55 represents a system of Volterra integro-differential equations. Without the inertial term, 1.55 represents a system of Volterra integral equations of the first kind. We now consider implicit schemes for integrating these equations in time. The goal is to reduce the system of equations 1.55 to a system in standard form

$$M\ddot{\eta}(t) + C\dot{\eta}(t) + K\eta(t) = \hat{f}(t) \tag{1.60}$$

where C is a *constant* damping matrix, and $\hat{f(t)}$ 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 1.55 is certainly 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 1.55 are reduced to the system 1.60, 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 1.55 can be split into two parts (considering only a single Prony series term)

$$\int_0^t e^{\frac{t-\tau}{s}} \dot{\eta}(t) d\tau = \int_0^{t_i} e^{\frac{t-\tau}{s}} \dot{\eta}(t) d\tau + \int_{t_i}^t e^{\frac{t-\tau}{a}} \dot{\eta}(t) d\tau \tag{1.61}$$

$$= e^{\frac{\Delta t}{s}} \int_0^{t_i} e^{\frac{t_i-\tau}{s}} \dot{\eta}(t) d\tau + \int_{t_i}^t e^{\frac{t-\tau}{s}} \dot{\eta}(t) d\tau \tag{1.62}$$

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.

In order to evaluate the term

$$\int_{t_i}^t e^{\frac{t-\tau}{s}} \dot{\eta}(t) d\tau \tag{1.63}$$

we first need a representation for the velocity $\dot{\eta(t)}$ in the interval $t \in [t_i, t]$. We present two choices, both of which are second order accurate.

1.6.3 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,

$$\dot{\eta}(t) = \dot{\eta}(t_i) + \frac{\ddot{\eta} + \ddot{\eta}(t_i)}{2}(t - t_i) \tag{1.64}$$

where $\ddot{\eta}$ is the (unknown) acceleration at current time t, and $\ddot{\eta}(t_i)$ is the previous acceleration. Although equation 1.64 is the correct representation for velocity, it is inconvenient in that it would lead to (after inserting into equation 1.63) 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

$$\eta(t) = \eta(t_i) + \frac{\dot{\eta} - \dot{\eta}(t_i)}{\Delta t}(t - t_i) \tag{1.65}$$

We note that equations 1.64 and 1.65 are equivalent representations of the velocity. By inserting equation 1.65 into equation 1.63 we obtain

$$\int_{t_i}^{t} e^{\frac{t-\tau}{s}} \dot{\eta}(t) d\tau = \left[s + \frac{s^2}{\Delta t}\left(e^{\frac{\Delta t}{s}} - 1\right)\right] \dot{\eta} + \left[-s e^{\frac{-\Delta t}{s}} + \frac{s^2}{\Delta t}\left(1 - e^{\frac{-\Delta t}{s}}\right)\right] \dot{\eta}_i \tag{1.66}$$

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

$$C = c_K(s_K + \frac{s_K^2}{\Delta t}(e^{\frac{-\Delta t}{s_K}} - 1))\mathbf{B^T D_K B} + c_G(s_G + \frac{s_G^2}{\Delta t}(e^{\frac{-\Delta t}{s_G}} - 1))\mathbf{B^T D_G B} \tag{1.67}$$

The second term is known, and thus it can be added to the load vector.

1.6.4 Midpoint Representation of Velocity

A second implicit scheme can be derived simply by using the midpoint rule on the velocity in the viscoelastic term. The only difference from the linear approach described above is in equation 1.66.

$$\dot{\eta}(t) = \frac{\dot{\eta} + \dot{\eta}(t_i)}{2} \tag{1.68}$$

This leads to

$$\int_{t_i}^{t} e^{\frac{t-\tau}{s}} \dot{\eta}(t) d\tau = \frac{s}{2}\left(1 - e^{\frac{\Delta t}{s}}\right) \dot{\eta} + \frac{s}{2}\left(1 - e^{\frac{\Delta t}{s}}\right) \dot{\eta}_i \tag{1.69}$$

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.

It should be noted that 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.

1.7 Linear Eigen Analysis

Linear Eigen analysis is a solution of the equation,

$$(K - \lambda M)\phi = 0 \tag{1.70}$$

The equation is considered *linear* in the sense that λ appears only to the first power. Solution of the equations involved is definitely not linear. Practically, there are many linear solves typically associated with a given eigen pair.

A number of approaches can be used to solve this system. We refer you to an excellent comparison report for a few of the iterative methods available (see 6). Direct methods such as the *QR* algorithm or Jacobi transformations are not scalable to very large systems. In any event, they do not parallelize well. In Salinas, we rely on the shifted and inverted Lanczos algorithm as implemented in `ARPACK`. Further, since the linear solvers that we have at our disposal are ensured convergent only for positive definite systems, we require a negative shift. Documentation on this method is available in the ARPACK package (see 7).

1.8 Random Vibration

Details of random vibration analysis are included in a number of papers[1]. These few paragraphs document what was implemented.

1.8.1 algorithm

The first step in the calculation is computation of a modal spatial contribution, Γ_{qq}, which is performed in `ComputeGammaQQ`. This is accomplished as follows.

Let the modal frequency response be defined as,

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

The modal force contribution from load a is,

$$\begin{aligned} F_{ia}(f) &= \sum_k \phi_{ik} f_k^a s_a(f) \\ &= Z_a^i s_a(f) \end{aligned}$$

where f_k^a is the k component of the force vector associated with load a, and $s_a(f)$ contains all of the frequency content of the force, but none of the spatial dependence. We have defined Z_a^i for each load that represents the sum of all the spatial contributions for mode i. It represents the frequency independent component of the force for load a.

$$Z_a^i = \sum_k f_k^a \phi_{ik}$$

[1] see for example, reference 8.

A transfer function to an output degree of freedom, k, from the input load a, may be written as a modal sum.

$$H_{ka}(f) = \sum_i F_{ia}(f) q_i(f) \phi_{ik}$$

where ϕ_{ik} is the eigenvector of mode i.

1.8.2 Power Spectral Density

The displacement power spectral output (at a single location) is a 3×3 matrix.

$$\begin{aligned} G_{mn}(f) &= \sum_{a,a'} H^*_{ma}(f) H_{na'}(f) \\ &= \sum_{i,j} \sum_{a,a'} F^*_{ia}(f) q^*_i(f) \phi_{im} F^*_{ja'}(f) q_j(f) \phi_{jn} \\ &= \sum_{i,j} \sum_{a,a'} q^*_i(f) q_j(f) \phi_{im} \phi_{jn} Z^i_a S^{a,a'}(f) Z^j_{a'} \end{aligned}$$

Here $S^{a,a'}(f)$ is the complex cross-correlation matrix between loads a and a', and the superscript '*' denotes complex conjugate. The subscripts m and n are applicable to the 3 degrees of freedom at a single location.

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

$$G_{mn}(f) = \sum_{i,j} \phi_{im} \phi_{jn} \mathcal{G}_{ij}(f) \tag{1.71}$$

where

$$\mathcal{G}_{ij}(f) = q^*_i(f) q_j(f) \sum_{a,a'} Z^i_a Z^j_{a'} S^{a,a'}(f) \tag{1.72}$$

Note that with the exception of the Z^i_a (which may be computed only once and are a fairly small matrix), all the terms in equation 1.72 are completely known on each subdomain.

1.8.3 RMS Output

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

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

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

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 just as 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 just $G_{mm}(\omega)\omega^4$. Subsection 2.21.5 provides details about transforming power spectra to an output coordinate system.

1.8.4 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

1.8.5 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. Therefore, the `svd()` must be computed using complex arithmetic. We use the `zgesvd` routine from `arpack`. The results from the `svd` of an hermitian matrix are real eigenvalues (stored in X), and complex vectors, stored in Q.

At the element level another `svd` must be performed. In this case we are computing the singular values of the matrix C.

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

where,

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

Obviously, 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 it's 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.

1.8.6 model truncation

The svd calculations provide the information needed for model truncation. In general, if the size of the model grows, the number of modes required for an analysis also grows. The relationship is very model dependent. However, the computational time for calculating the svd varies as the cube of the dimension of the matrix. Since the svd(Γ) is only computed once, it is not terribly important. 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 5^3 one might expect). Clearly, this is unacceptable especially as 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, $X[(\texttt{Cdim})]/\texttt{X}[0] > \texttt{10}^{-14}$. This restricts the dimension of C to a fairly 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.

1.9 Modal Frequency Response Methods

The Salinas 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.

1.9.1 No Rigid Body Modes

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

$$(-\omega^2 M + j\omega C + K)\hat{u} = \hat{f} \tag{1.73}$$

Consider the modal approximation

$$\hat{u} \approx \sum_{i=1}^{N} \phi_i q_i \tag{1.74}$$

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

$$(-\omega^2 m_i + j\omega c_i + k_i)q_i = \phi_i^T \hat{f} \tag{1.75}$$

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

$$m_i = \phi_i^T M \phi_i \tag{1.76}$$
$$c_i = \phi_i^T C \phi_i \tag{1.77}$$
$$k_i = \phi_i^T K \phi_i \tag{1.78}$$
$$\tag{1.79}$$

are the modal mass, modal damping, and modal stiffness of the i'th mode. Solving equation 1.75 for q_i leads to

$$q_i = (\phi_i^T \hat{f})/(-\omega^2 m_i + j\omega c_i + k_i) \tag{1.80}$$

Replacing $(-\omega^2 M + j\omega C)\hat{u}$ in equation 1.73 with the modal approximation

$$(-\omega^2 M + j\omega C) \sum_{i=1}^{N} \phi_i q_i \tag{1.81}$$

leads to

$$K\hat{u} = \hat{f} + (\omega^2 M - j\omega C) \sum_{i=1}^{N} \phi_i q_i \tag{1.82}$$

Recall that the mode shapes satisfy the eigenproblem

$$K\phi_i = \omega_i^2 M\phi_i \tag{1.83}$$

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

$$K^{-1} M\phi_i = \phi_i/\omega_i^2 \tag{1.84}$$

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

$$C = \sum_{i=1}^{N} \left(\frac{2\zeta_i \omega_i}{m_i} \right) (M\phi_i)(M\phi_i)^T \tag{1.85}$$

where ζ_i is the damping ratio of the i'th mode. Substituting equations 1.84 and 1.85 into equation 1.82 and solving for $\hat{u}$ leads to

$$\hat{u} = K^{-1}\hat{f} + \sum_{i=1}^{N} (\omega^2/\omega_i^2 - 2\zeta_i j\omega/\omega_i)\phi_i q_i \tag{1.86}$$

The acceleration frequency response, $\hat{a}$, can be obtained by multiplying equation 1.86 by $-\omega^2$.

1.9.2 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[9] (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 such as FETI 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 Salinas are unable to manage a singular stiffness matrix.

The equations of interest are the frequency domain equations of motion

$$-\omega^2 Mu + j\omega Cu + Ku = f \tag{1.87}$$

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

$$u(\omega) = u_R(\omega) + u_E(\omega) \tag{1.88}$$
$$= \Phi_R q_R(\omega) + \Phi_E q_E(\omega) \tag{1.89}$$

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 NxN_R matrix of rigid body eigenvectors, Φ_E is an NxN_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 now substitute equation 1.89 into equation 1.87, and premultiply both sides by Φ_R^T and Φ_E^T. This yields two sets of equations, after using orthogonality and the fact that $K\Phi_R = 0$.

$$-\omega^2 q_R + j\omega C_R q_R = \Phi_R^T f \tag{1.90}$$
$$-\omega^2 q_E + j\omega C_E q_E + K_E q_E = \Phi_E^T f \tag{1.91}$$

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 ith diagonal entry of C_E is $2\omega_i\zeta_{E_i}$, and the ith 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

$$q_{R_i} = \frac{\Phi_{R_i}^T f}{-\omega^2 + j\omega C_{R_i}} \tag{1.92}$$
$$q_{E_i} = \frac{\Phi_{E_i}^T f}{-\omega^2 + j\omega C_{E_i} + \omega_{E_i}^2} \tag{1.93}$$

Equation 1.91 can be solved for q_E, and substituting this into equation 1.89, we obtain

$$u = \Phi_R q_R + \Phi_E K_E^{-1} \Phi_E^T f + \omega^2 \Phi_E K_E^{-1} q_E - j\omega \Phi_E K_E^{-1} C_E q_E \tag{1.94}$$

The first term in equation 1.94 is known. The third and fourth terms of equation 1.94 can be computed by modal truncation, and in fact these are the same as the second and third terms of equation 1.86. The second term in equation 1.94 is the static correction, and is not readily computable in the present form since all of the flexible modes would have to be known to compute it.

In order to compute the second term in equation 1.94, 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 in

order 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 FETI 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. We note that FETI can solve problems of the form $Ku = f$ even if K is singular, provided that the right hand side f is orthogonal to the rigid body modes.

The procedure is to first apply Gram Schmidt orthogonalization to obtain f_E. Then, we use FETI to solve the system $Ku_E = f_E$, where K is singular. Finally, to be sure u_E is orthogonal to the rigid body modes, we apply Gram Schmidt one more time to u_E. Though in theory u_E is already orthogonal to the rigid body modes after the FETI solve, numerical roundoff may result in a small loss of orthogonality (especially if the solver tolerance is loose), and thus we apply this final orthogonalization to u_E to be on the safe side. The resulting solution we again denote by u_E. Then,

$$u_E = \Phi_E K_E^{-1} \Phi_E^T f \tag{1.95}$$

and thus all of the terms in equation 1.94 are known. Thus the modal frequency response can be computed using equation 1.94.

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

$$\alpha = \frac{\phi_i^T f}{\phi_i^T \phi_i} \tag{1.96}$$

and then

$$f_E = f - \alpha\phi \tag{1.97}$$

The dot products appearing in these expressions do not involve the mass matrix. They are the standard dot products.

1.9.3 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, 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.

1.10 Fast Modal Solutions

Because modal based solutions such as **modaltransient** do not require a linear solve, they can greatly accelerate 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

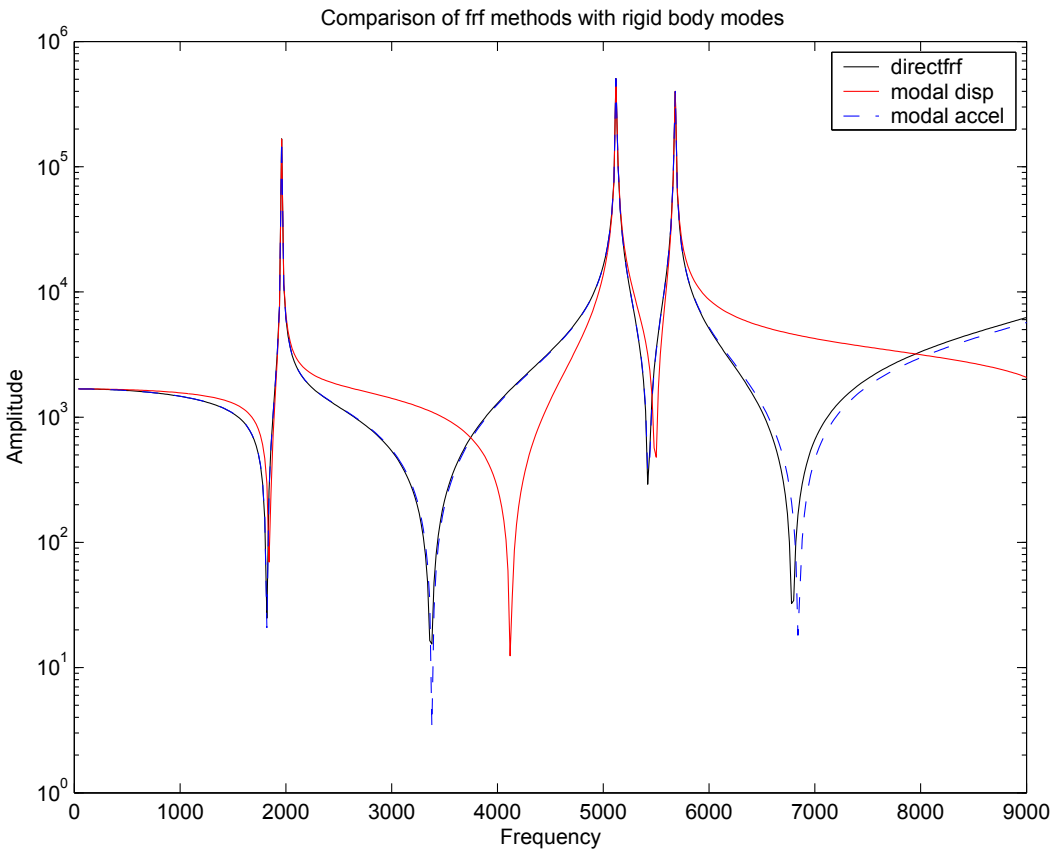


Figure 2. A comparison of the modal displacement, modal acceleration, and direct frequency response approaches. The modal acceleration method gives a better approximation to the direct approach than the modal displacement method.

1. Compute the full eigen problem, $(K - \lambda M)\Phi = 0$
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 1.101.
4. Expand from modal to *full* physical space.

$$X_{n+1}^k = \sum_i^{Nmodes} q_{n+1}^i \Phi_{ki}$$

5. Collapse the physical space to the output degrees of freedom.

$$\tilde{x} = \text{subset}(X)$$

The parallel data (matrices and vectors Φ and X) are partitioned by processor.

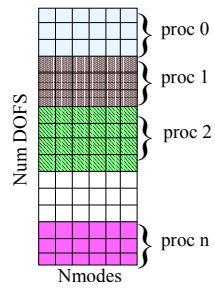


Figure 3. Standard Modal Transient Algorithm. Note that while the output is required on only a small part of the model, a calculation of data on all degrees of freedom is performed first, and results are then collapsed back to the reduced model.

manipulates a lot of data when the model size is large, see Figure 3. We here address a method for much higher performance provided that output is required on a very limited data set and that the force is simple.

1.10.1 Modal Solution Summary

Using the trapezoidal rule, Newmark-Beta integrator[2] equation 1.6 may be condensed to,

$$\left[\frac{4}{\Delta t^2}M + \frac{2}{\Delta t}\hat{C} + K\right] d_{n+1} = F_{n+1}^{ext} + \hat{C}\left[v_n + \frac{2}{\Delta t}d_n\right] + M\left[\frac{4}{\Delta t^2}d_n + \frac{4}{\Delta t}v_n + a_n\right] \tag{1.98}$$

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

Also,

$$v_{n+1} = -v_n + \frac{2}{\Delta t}(d_{n+1} - d_n) \tag{1.99}$$

$$a_{n+1} = -a_n + \frac{4}{\Delta t^2}(d_{n+1} - d_n) - \frac{4}{\Delta t} v_n \tag{1.100}$$

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.

$$a_i q_{n+1}^i = q_n^i + f_{n+1}^i + \tilde{f}^i \tag{1.101}$$

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 equations are now uncoupled, i.e. the solution for each modal coordinate is independent of any other.

1.10.2 Parallel Fast Modal

In many cases the analyst is interested only in the data in a very reduced set (such as data in the history file). In these cases, large amounts of data are processed, only to reduce the data at each time step to a the reduced system. The parallel computer processing is being expended to process large vectors that are not really 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 4 for transient dynamics, and in Figure 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.

1.10.3 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.

1. Begin with eigenvalues, λ, and *reduced* eigen vectors, 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 1.101 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 4. Fast Modal Transient Algorithm

1. Begin with eigenvalues, λ, and *reduced* eigen vectors, 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 5. Fast Modal Frequency Response Algorithm

However, in most cases,[3] the force in the physical coordinates is computed as a sum of spatial and temporal terms.[4]

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

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,

$$\begin{aligned} f^i(t) &= \sum_k \Phi_{ki} F^{ext}(x_k,t) && (1.102) \\ &= \sum_k \Phi_{ki} \sum_s^{Nsets} \hat{F}_s(x)\delta^s(t) \\ &= \sum_s^{Nsets} \zeta_s^i \delta^s(t) && (1.103) \end{aligned}$$

where,

$$\zeta_s^i = \sum_k \Phi_{ki} \hat{F}_k^s \tag{1.104}$$

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

1.11 Complex Eigen Analysis - Modal Analysis of Damped Structures

1.11.1 Modal Analysis of Damped Structures

Salinas will solve the eigenvalue problems for structures with some types of damping. The algorithms are designed for internally damped structures such as from viscoelastic materials. The package is called `Ceigen`, and the parameters to be aware of are `eig_tol`, `nmodes`, and `viscofreq`. The first two parameters, `eig_tol` and `nmodes` will be familiar to Salinas 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 complex eigen value problem which we solve is also known as the quadratic eigenvalue equation.

$$\left[K + \lambda D + \lambda^2 M\right] \phi = 0 \tag{1.105}$$

where,

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

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

[4] What is described here for time applies equally well for functions in the frequency domain. They are products of spatial and frequency components.

All of the matrices are independent of frequency. Note that we are solving for $\lambda = i\omega + \gamma$, not ω^2.

1.11.2 Input File Specification

The Salinas input file specification is similar to the specification for transient simulations. To change a working Salinas input file for a transient problem into a Salinas 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 ceig
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 was needed for the transient simulation, and is still needed for modal analyses.

1.11.3 Output File Format

The output is very 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 (1.106). Pure real eigenvalues are not written to the Results file.[5] 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 less than or equal to `nmodes`.

As is the case with the undamped eigenvalue problem, Salinas 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 actually 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.

[5] Real modes correspond to an overdamped mode with no oscillatory component. These are usually generated from numerical artifacts discussed below, and are seldom of practical value

1.11.4 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. Salinas returns the frequencies $\omega/(2\pi)$. `Ceigen` solves a similar problem. `Ceigen` solves the quadratic eigenvalue problem

$$[\mathbf{M}\lambda^2 + \mathbf{D}\lambda + \mathbf{K}]u = 0, \quad u^T u = 1. \tag{1.106}$$

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 of the solutions. For matrix eigenvalue problems the key idea is deflation. One big subspace is used to compute all of 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.

1.11.5 Viscoelasticity

The eigenvalue problem for viscoelastic problems[10] in the most simple case (one term Prony series) has the form

$$[\mathbf{M}s^2 + \mathbf{D}(s)s + \mathbf{K}]u = 0. \tag{1.107}$$

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

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

Prony series damping in the time domain[10] creates a frequency domain problem with real eigenvalues that are not physical.[10] 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.

1.11.6 Viscofreq

The eigenvalue problem in equation (1.107) 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`$= \omega$ 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 Salinas 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.[10] 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 (1.106) that are far from `viscofreq` are over-damped. If for a given mode shape, s_o is closer to the real eigenvalue of equation (1.107) than either complex conjugate pair, then `Ceigen` may return the real eigenvalue. For example equation (1.107) has many real eigenvalues clustered left of $-a$.

1.11.7 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 (qevp). If the k-th mode does not satisfy the convergence tolerance, and $k \leq$`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 very 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).

1.11.8 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 since the complex-valued damping matrix is somewhat difficult to deal with in quadratic eigensolvers. 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.

$$\left[K_\infty + sD(s) - s^2 M\right] u = f(s) \tag{1.108}$$

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

$$D(s) = (E_G - E_\infty)\frac{1}{s+1/\tau}B \tag{1.109}$$

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.

We now return to equation 1.108, and consider different ways of linearizing the relation, since for the quadratic eigenvalue problem, we may only solve equations of the form in equation 1.105, i.e. quadratic in λ or s.

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 $\frac{(r+1)-i\omega\tau}{(r+1)-i\omega\tau}$.

$$\begin{aligned} D(s) &= (E_G - E_\infty)\frac{1}{s+1/\tau}B && (1.110)\\ &= (E_G - E_\infty)\frac{\tau}{i\omega\tau + (r\tau+1)}B && (1.111)\\ &= \frac{\tau((r\tau+1) - i\omega\tau)}{(r\tau+1)^2 + \omega^2\tau^2}(E_G - E_\infty)B && (1.112) \end{aligned}$$

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

$$\begin{aligned} sD(s) &= (i\omega + r)D(i\omega + r) && (1.113)\\ &= \frac{\tau(i\omega+r) + \omega^2\tau^2 + r^2\tau^2}{(r+1)^2 + \omega^2\tau^2}(E_G - E_\infty)B && (1.114) \end{aligned}$$

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

$$\left[\left(E_\infty + (E_G - E_\infty)\frac{\omega^2\tau^2 + r^2\tau^2}{(r+1)^2 + \omega^2\tau^2}\right)B + s\left(\frac{\tau}{(r+1)^2 + \omega^2\tau^2}\right)(E_G - E_\infty)B + s^2 M\right]\phi = 0 \tag{1.115}$$

Thus we see that the damping matrix is purely 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 a number of Prony terms. Equation 1.115 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.

$$\tilde{K}(r + i\omega) = \sum_{elem} \tilde{K}_{elem}(r + i\omega) \tag{1.116}$$

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,

$$\tilde{D}_{elem}(r+i\omega) = \text{real}(D(r+i\omega)) \tag{1.117}$$

We now solve the *linearized* eigenvalue equation for λ,

$$\left[\tilde{K}(r+i\omega) + i\lambda\tilde{D}(r+i\omega) - \lambda^2 M\right]\phi = 0 \tag{1.118}$$

A Simple Error Estimate This question is now how well the eigenvalues computed from equation 1.115 approximate the true eigenvalues of equation 1.108.

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

$$\delta = s_c - s_\omega \tag{1.119}$$

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 1.108.

$$\left(s_c^2 M + s_c D(s_c) + K\right)\phi_c = res \tag{1.120}$$

The residual, as defined in equation 1.120, is a computable quantity. Obviously, 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.

1.12 SA_eigen

The quadratic eigenvalue problem which we address in this solution method is given by the equation below.

$$\left(K + \lambda C + \lambda^2 M\right)\phi = 0 \tag{1.121}$$

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.

$$\left(\begin{bmatrix} K_s & 0 \\ 0 & K_a \end{bmatrix} + \lambda \begin{bmatrix} C_s & L \\ -\rho_a L^T & C_a \end{bmatrix} + \lambda^2 \begin{bmatrix} M_s & 0 \\ 0 & M_a \end{bmatrix}\right)\begin{bmatrix} \phi_s \\ \phi_a \end{bmatrix} = 0 \tag{1.122}$$

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, $\phi_a = \nabla\dot{u}_a$.

Table 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.

The matrix C will be completely asymmetric if it contains only coupling terms. In this case it is called gyroscopic, and it can be shown that the system is Hermitian, and has real eigenvalues. 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.

While various methods are available for solving the generalized, linear eigenvalue problem,[6] 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 project back out 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.

$$\phi = Tq \tag{1.123}$$

$$T^{\dagger}\left(K+\lambda C+\lambda^2 M\right)Tq = 0 \tag{1.124}$$

$$\left(\tilde{k}+\lambda\tilde{c}+\lambda^2\tilde{m}\right)q = 0 \tag{1.125}$$

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 1.122 are complex conjugates of each other.

The structural/acoustics problem may be viewed as a two subdomain problem.[7] There are a variety of basis functions that have been examined for connecting such subdomains. Two common sets are listed in Table 2.

[6] The generalized linear eigenvalue problem is $(K-\lambda M)\phi = 0$.

[7] There is no requirement that each of these subdomains be topologically connected in any special way.

We here investigate only the free-free method. Though this method has proved to converge rather slowly for structure/structure problems, the coupling between the structural and acoustic domains is often rather weak, so 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.
- 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 some of these as well. Perhaps some methods could be used to identify a subset of modes that would best aid in model completeness and convergence.
- Spline or boundary expansions are possible.

1.13 Quadratic Modal Superposition

Consider the system

$$M\ddot{u} + C\dot{u} + Ku = f(t) \tag{1.126}$$

where M, C, and K are the mass, damping, and stiffness matrices. Standard methods may be used to solve the eigenvalue equation derived from 1.126 only in the case where the eigenvectors of K and M also diagonalize C (as in proportional damping for example). Unfortunately, such cases are usually not physical, and are rare in practice. For a general damping matrix, no procedures are available to directly solve the eigen equation. For an excellent survey article on quadratic eigenvalue systems, see the article by Tisseur.[11]

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,

$$w = \begin{bmatrix} u \\ \dot{u} \end{bmatrix} \tag{1.127}$$

If we consider the eigenvalue problem corresponding to equation 1.126, 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

$$\begin{bmatrix} -K & \mathbf{0} \\ \mathbf{0} & M \end{bmatrix} w = \begin{bmatrix} C & M \\ M & \mathbf{0} \end{bmatrix} \dot{w} \tag{1.128}$$

We assume a solution of the form $w = \phi e^{\lambda t}$, and arrive at the eigenvalue problem,

$$A\phi = \lambda B\phi \tag{1.129}$$

where

$$A = \begin{bmatrix} -K & \mathbf{0} \\ \mathbf{0} & M \end{bmatrix}, \tag{1.130}$$

and

$$B = \begin{bmatrix} C & M \\ M & \mathbf{0} \end{bmatrix} \tag{1.131}$$

Equation 1.129 yields the "right" eigenvectors. As is seen later, we also need the "left" eigenvectors, which correspond to the eigenvalue problem,

$$\psi^{\dagger} A = \lambda \psi^{\dagger} B \tag{1.132}$$

We denote the left eigenvectors as ψ_i to distinguish them from the right eigenvectors ϕ_i.

1.13.1 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 1.128, 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,[11, 12, 13] one case where the matrices A and B are diagonalizable is if all of 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, as follows:[14]

- 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 quite simple to prove.[15] If the entries in a matrix are all real-valued, then any complex eigenvalues or eigenvectors that arise must come in complex conjugate pairs. In order 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 of the eigenvalues are complex, the only way that the product

$$det(A) = \lambda_1 \lambda_2 ... \lambda_n \tag{1.133}$$

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

$$\lambda_n \lambda_{n+1} = (\lambda_n^r + j\lambda_n^i) * (\lambda_n^r - j\lambda_n^i) = [\lambda_n^r]^2 + \left[\lambda_n^i\right]^2 \tag{1.134}$$

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 1.129.

$$W = [\phi_1 \phi_2 ... \phi_{2n}] \tag{1.135}$$

We can also define a matrix corresponding to the left-eigenvectors from equation 1.132.

$$U = [\psi_1 \psi_2 ... \psi_{2n}] \tag{1.136}$$

Representing the solution as $w = \sum_{i=1}^{2n} z_i \phi_i$, and the loading as,

$$g(t) = \begin{bmatrix} f(t) \\ 0 \end{bmatrix} \tag{1.137}$$

we have[11]

$$-\alpha_i z_i(t) + \beta_i \dot{z}_i(t) = \psi_i^\dagger g(t) \tag{1.138}$$

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, and not just 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 note that this is a first order system in time, rather than second order, and thus different methods are required for the numerical integration than are used for real-valued modal superposition. Superposition must be performed on the linearized system, as we have no general solution of the original second order system.

Time Domain Superposition

Equation 1.138 can be integrated numerically, using first-order time integrators. However, another approach is to use the analytical solution.

$$z_i(t) = \int_0^t \psi_i^* g(\tau) e^{-\lambda_i (t-\tau)} d\tau \tag{1.139}$$

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 1.138 implies that z_i also comes in conjugate pairs. We note that

$$w = \sum_{i=1}^{2n} z_i \phi_i = \sum_{i=1}^{n} \left[z_i \phi_i + \bar{z}_i \bar{\phi}_i \right] \tag{1.140}$$

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.

$$g(t) = g_0 e^{i\omega_{ex} t} \tag{1.141}$$
$$z_i(t) = z_i e^{i\omega_{ex} t} \tag{1.142}$$
$$\tag{1.143}$$

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 1.138, we obtain the equations for complex modal frequency response

$$[-\alpha_i + i\omega\beta_i] z_i = \psi_i^\dagger g_0 \tag{1.144}$$

This can also be written as,

$$z_i = \frac{\psi_i^\dagger g_0}{-\alpha_i + i\omega\beta_i} \tag{1.145}$$

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[16] of stabelizing 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 1.145. Thus, when $\alpha_i = i\omega\beta_i$ this additional term would prevent a singular response. This additional real term takes the form

$$z_i = \frac{\psi_i^\dagger g_0}{\gamma_i - \alpha_i + i\omega\beta_i} \tag{1.146}$$

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.

$$w(\omega) = \sum_{i=1}^{2n} z_i(\omega)\phi_i \tag{1.147}$$
$$= \sum_{i=1}^{2n} \frac{\phi_i \psi_i^\dagger g_0}{\gamma_i - \alpha_i + i\omega\beta_i} \tag{1.148}$$

1.13.2 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 1.126, and the transformation $u = T\hat{u}$, we can transform equation 1.126 into a reduced space as

$$\hat{m}\ddot{\hat{u}} + \hat{c}\dot{\hat{u}} + \hat{k}\hat{u} = \hat{f} \tag{1.149}$$

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

$$\hat{q} = \begin{bmatrix} \hat{u} \\ \dot{\hat{u}} \end{bmatrix} \tag{1.150}$$

As was done for the full system for the QEVP method, we project this into the first order system[8].

$$\hat{A}\hat{q} - \hat{B}\hat{\dot{q}} = g(\hat{t}) \tag{1.151}$$

where

$$\hat{A} = \begin{bmatrix} 0 & I \\ -\hat{k} & -\hat{c} \end{bmatrix} \tag{1.152}$$

$$\hat{B} = \begin{bmatrix} I & 0 \\ 0 & \hat{m} \end{bmatrix} \tag{1.153}$$

$$\hat{g} = \begin{bmatrix} 0 \\ -\hat{f} \end{bmatrix} \tag{1.154}$$

Assuming a solution of the form $\hat{q} = \hat{\phi}e^{\lambda t}$, we arrive at the eigenvalue problem

$$\hat{A}\hat{\phi} = \lambda\hat{B}\hat{\phi} \tag{1.155}$$

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 1.155 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.

We transform equation 1.151 into the frequency domain.

$$\hat{A}\hat{q} - i\omega_{ex}\hat{B}\hat{q} = \hat{g}(\omega) \tag{1.156}$$

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 1.156, and premultiplying by the left eigenvectors $\hat{\psi}_i$, we obtain

$$\hat{\alpha}_i\hat{z}_i - i\hat{\beta}_i\omega_{ex}z_i = \hat{\psi}_i{}^{\dagger}\hat{g} \tag{1.157}$$

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, 1.157 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 1.150. Finally, once $\hat{u}$ is known, the original solution u can be computed from the relation $u = T\hat{u}$.

[8] also known as a state space solution

1.13.3 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, 1.126. 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.
- Both the left and the right eigenvectors of the linearized system can be written as displacement and velocity components, where the velocity component is just just rescaled. Specifically,

$$\phi_j = \begin{bmatrix} \phi_j^u \\ \lambda_j \phi_j^u \end{bmatrix}, \tag{1.158}$$

  where ϕ^u is the displacment component of the eigenvector.
- 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.

1.13.4 Complex Eigenvector Orthogonalization

When the eigenvalues of a system are redundant, the eigenvectors are not fully defined, but can be arbitrary linear combinations. Some solvers, such as DGGEV don't guarantee orthogonality of these vectors. If such orthogonalization is required, the procedure in Figure 6 may be followed to orthogonalize two eigenvectors with a common eigen value.

1.14 Component Mode Synthesis

Component mode synthesis in Salinas 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's book, (Chapter 19 of 9). It is summarized below. Note that in Salinas 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 solution methods as well. Here we describe only the eigen analysis application. Within Salinas only a subset of the standard CMS method is available. Salinas may reduce *an entire model* to a set of interface degrees of freedom with the corresponding system matrices and transfer matrices. Salinas 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.

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.

$$\psi_1^\dagger B\phi_1 = \beta_{11} \tag{1.159}$$
$$\psi_1^\dagger B\phi_2 = \beta_{12} \tag{1.160}$$
$$\psi_2^\dagger B\phi_1 = \beta_{21} \tag{1.161}$$

We modify ψ_2 and ϕ_2 to ensure that $\beta_{12} = \beta_{21} = 0$. Let $\hat{\psi}$ be the corrected eigenvector.

$$\hat{\psi}_2 = \psi_2 - \varepsilon\psi_1$$

We require that $\hat{\psi}_2^\dagger B\phi_1 = 0$. Then,

$$0 = \hat{\psi}_2^\dagger B\phi_1 \tag{1.162}$$
$$= (\psi_2 - \varepsilon\psi_1)^\dagger B\phi_1 \tag{1.163}$$
$$= \beta_{21} - \varepsilon\beta_{11} \tag{1.164}$$

Thus,

$$\hat{\psi}_2 = \psi_2 - \frac{\beta_{21}}{\beta_{11}}\psi_1 \tag{1.165}$$

For the right eigenvector,

$$\hat{\phi}_2 = \phi_2 - \frac{\beta_{12}}{\beta_{11}}\phi_1 \tag{1.166}$$

Figure 6. Complex EigenVector orthogonalization

1.14.1 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, V.

$$\left(\begin{bmatrix} K_{vv} & K_{vc} \\ K_{cv} & K_{cc} \end{bmatrix} - \lambda \begin{bmatrix} M_{vv} & M_{vc} \\ M_{cv} & M_{cc} \end{bmatrix} \right) \begin{bmatrix} u_v \\ u_c \end{bmatrix} = 0 \tag{1.167}$$

Within Salinas we consider only the cases where K_{vv} is nonsingular. For the Craig-Bampton method this implies that clamping the interface degrees of freedom removes all zero energy modes from the structure.

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

$$u = \Psi p \tag{1.168}$$

The component modes are selected as follows. We let $\Psi = [\Phi\ \psi]$, where Φ is a set of eigen modes of the fixed interface, i.e.,

$$(K_{vv} - \lambda M_{vv})\,\Phi = 0$$

We retain only a subset of the modes in this system. In addition, we define the constraint modes, ψ, as 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 Craig,

$$\psi = -K_{vv}^{-1} K_{vc} \tag{1.169}$$

Note that since we require that K_{vv} be positive definite, all these solutions are well defined. The matrix need be factored only once for all the modes.

Reduced System

As shown in *Craig*, the reduced system matrices can be written as follows.

$$\mu = \begin{bmatrix} \mu_{kk} & \mu_{kc} \\ \mu_{ck} & \mu_{cc} \end{bmatrix} \tag{1.170}$$

and,

$$\kappa = \begin{bmatrix} \kappa_{kk} & \kappa_{kc} \\ \kappa_{ck} & \kappa_{cc} \end{bmatrix} \tag{1.171}$$

where,

$$
\begin{aligned}
\mu_{kk} &= I_{kk} \\
\mu_{kc} &= \mu_{ck}^T = \phi^T (M_{vv}\psi + M_{vc}) \\
&= \phi^T M_{vv}\psi + (M_{cv}\phi)^T \\
\mu_{cc} &= \psi^T (M_{vv}\psi + M_{vc}) + M_{cv}\psi + M_{cc} \\
&= \psi^T M_{vv}\psi + (M_{cv}\psi)^T + M_{cv}\psi + M_{cc}
\end{aligned} \tag{1.172}
$$

and,

$$
\begin{aligned}
\kappa_{kk} &= \Lambda_{kk} \\
\kappa_{kc} &= \kappa_{ck} = 0 \\
\kappa_{cc} &= K_{cc} - K_{cv}K_{vv}^{-1}K_{vc} \\
&= K_{cc} + K_{cv}\psi
\end{aligned} \tag{1.173}
$$

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 simply 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 eigensystem 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.

$$
\begin{bmatrix} K_{vv} & K_{vc} \\ K_{cv} & K_{cc} \end{bmatrix} \begin{bmatrix} u_v \\ u_c \end{bmatrix} = \begin{bmatrix} 0 \\ R \end{bmatrix} \tag{1.174}
$$

Equation 1.169 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.

$$
\begin{bmatrix} K_{1vv} & K_{1vc} & 0 & 0 & C_{1v}^T \\ K_{1cv} & K_{1cc} & 0 & 0 & C_{1c}^T \\ 0 & 0 & K_{2vv} & K_{2vc} & C_{2v}^T \\ 0 & 0 & K_{2cv} & K_{2cc} & C_{2c}^T \\ C_{1v} & C_{1c} & C_{2v} & C_{2c} & 0 \end{bmatrix} \begin{bmatrix} u_{1v} \\ u_{1c} \\ u_{2v} \\ u_{2c} \\ \mu \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ R \end{bmatrix} \tag{1.175}
$$

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

$$\begin{bmatrix} K_{1vv} & 0 & C_{1v}^T \\ 0 & K_{2vv} & C_{2v}^T \end{bmatrix} \begin{bmatrix} u_{1v} \\ u_{2v} \\ \mu \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \tag{1.176}$$

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 just 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 1.170 through 1.173 are all of the form, $a^T Bc$, 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.

   The issue of multipoint constraints is a little trickier. The system is now divided using lagrange multipiers, χ. Equation 1.167 may be so expressed.

$$\left(\begin{bmatrix} K_{vv} & K_{vc} & C_v^T \\ K_{cv} & K_{cc} & C_c^T \\ C_v & C_c & 0 \end{bmatrix} - \lambda \begin{bmatrix} M_{vv} & M_{vc} & 0 \\ M_{cv} & M_{cc} & 0 \\ 0 & & 0 \end{bmatrix} \right) \begin{bmatrix} u_v \\ u_c \\ \chi \end{bmatrix} = 0 \tag{1.177}$$

   where χ are the lagrange multipliers. But, we want these multipiers 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.

$$\left(\begin{bmatrix} \tilde{K}_{vv} & \tilde{K}_{vc} \\ \tilde{K}_{cv} & K_{cc} \end{bmatrix} - \lambda \begin{bmatrix} \tilde{M}_{vv} & \tilde{M}_{vc} \\ \tilde{M}_{cv} & M_{cc} \end{bmatrix} \right) \begin{bmatrix} \tilde{u}_v \\ u_c \end{bmatrix} = 0 \tag{1.178}$$

   where,

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

   and,

$$\tilde{u}_v = \begin{bmatrix} u_v \\ \chi \end{bmatrix}$$

   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, all of 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. **Accuracy Issues.** The accuracy of the null space is determined by the sum of two large quantities (see equation 1.173). With iterative solvers, this may not be determined accurately enough to insure 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.

   Our experience has shown that inaccurate solves lead to corruption of the zero energy modes with little impact on the remaining elastic modes. Thus, it seems reasonable to eliminate the error in a post processing step. 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 use the following algorithm.

   (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 a little more involved. Details of the algorithm are presented in Figure 7. Most of the operations in the algorithm operate on matrices of order 12 or smaller, so the computational cost is fairly minimal. The method does require very accurate determination of the zero energy modes.

1.15 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,[17],[18] and the second is a quantity of interest approach.[19] The explicit approaches are described in chapter 2 of,[20] 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,[20] though there are many similarities. Both the explicit and the quantity of interest approaches

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 eigenspectrum 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 7. Eigenvalue and Eigenvector corrections of Craig-Bampton reduced models

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, explicit estimators are more frequently used as element indicators to drive adaptivity algorithms, rather than as error estimators. The quantity of interest approach avoids the unknown constants, but is more work in terms of implementation.

1.15.1 Preliminaries

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

$$-\rho\lambda u = (\Lambda+\mu)\nabla(\nabla\cdot u)+\mu\nabla^2 u = \nabla\cdot\sigma(u) \tag{1.179}$$

or

$$A_1(u) = -\lambda A_2(u) \tag{1.180}$$

where where $A_1(u)$ and $A_2(u)$ are the partial differential operators implied by equation 1.179, 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 1.179 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 1.179 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

$$B(u,v) = \lambda M(u,v) \quad \forall v \in V \tag{1.181}$$

where

$$B(u,v) = \int_\Omega \sigma(u)\varepsilon(v)dx \tag{1.182}$$

and

$$M(u,v) = \int_\Omega \rho uv dx \tag{1.183}$$

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

$$Ku = \lambda Mu \tag{1.184}$$

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.

1.15.2 Approach I - explicit error estimator

In *Larsen*[17] and *Rannacher*,[18] two independently derived error estimates are presented for the Laplace equation. While the two estimates differ slightly, 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[17] and[18] respectively. In what follows we use Larsen's results (equation 1.185) exclusively. [9]

[9]Equation 1.185 applies to elements with linear shape functions. The more general expression may be found in equation 1.235 or the reference.

$$|\lambda - \lambda_h| \leq c\lambda C_{e,0} \left(\sum_{e=1}^{N_e} h_e^4 \bar{\rho}(u_h, \lambda_h)^2 \right)^{\frac{1}{2}} \tag{1.185}$$

$$|\lambda - \lambda_h| \leq C_2 \sum_{e=1}^{N_e} h_e^2 \bar{\rho}(u_h, \lambda_h)^2 \tag{1.186}$$

where h_e is the element diameter, and

$$\bar{\rho}(u_h, \lambda_h)^2 = \int_{\Omega_e} \left(|A_1 u_h + \lambda_h A_2 u_h| + R_{flux} \right)^2 d\Omega_e \tag{1.187}$$

The first term on the right hand side is the interior element residual, which is the differential stiffness operator A_1, defined in equation 1.180, applied to the computed element displacement combined with the computed eigenvalue times the differential mass operator A_2, also defined in equation 1.180, applied to the computed element displacement. This term is computed by representing the eigenvector as a summation

$$u_h(x) = \sum_{i=1}^{N} a_i N_i(x) \tag{1.188}$$

where a_i is the ith entry in the eigenvector, and $N_i(x)$ is the ith shape function, and then simply applying the gradient and divergence operators from equation 1.179 to the summation in equation 1.188.

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 $A_1 u_h$

$$A_1 u_h = \nabla \cdot \sigma(u_h) \tag{1.189}$$

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.

The second term is the boundary or flux residual.

$$R_{flux} = (h_e vol(e))^{-1/2} \left[\int_{\Gamma_e} R^2 d\Gamma_e \right]^{1/2} \tag{1.190}$$

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,

$$R = g - \sigma_{ij} n_j \tag{1.191}$$

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

$$R = [\sigma_{ij} n_j] = \sigma_{ij}^a n_j - \sigma_{ij}^b n_j \tag{1.192}$$

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 1.185. 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 the extension to elasticity. The second is the extension to multiple materials. These are covered in the following sections.

1.15.3 Extension of Estimators to 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 17, and uses quantities defined there.

We consider the eigenvalue problem

$$-\mu\Delta\mathbf{u} - (\Lambda+\mu)\nabla(\nabla\cdot\mathbf{u}) = -\nabla\cdot\sigma(\mathbf{u}) = \theta\rho\mathbf{u} \quad \text{in } \Omega \tag{1.193}$$

$$\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega \tag{1.194}$$

where the Lamé constants Λ and μ satisfy

$$\Lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)} \tag{1.195}$$

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

$$a(\mathbf{u},\mathbf{v}) = \theta b(\mathbf{u},\mathbf{v}), \quad \forall\, \mathbf{v}\in\mathbb{V} \tag{1.196}$$

$$b(\mathbf{u},\mathbf{u}) = 1 \tag{1.197}$$

where

$$a(\mathbf{u},\mathbf{v}) = \int_\Omega \sigma(\mathbf{u})\cdot\varepsilon(\mathbf{v})dx \tag{1.198}$$

and

$$b(\mathbf{u},\mathbf{v}) = \int_\Omega \rho\mathbf{u}\cdot\mathbf{v}dx \tag{1.199}$$

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

Residual

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

$$R(\mathbf{u}_h,\theta_h)_{|\tau} = \frac{1}{\sqrt{\rho}}|\nabla\cdot\sigma(\mathbf{u}_h)+\theta_h\rho\mathbf{u}_h| + \sqrt{\frac{1}{h\ vol(\tau)}\int_{\partial\tau\setminus\partial\Omega}\left(\mathbf{n}\cdot\left[\frac{\sigma(\mathbf{u}_h)}{2\sqrt{\rho}}\right]\right)^2} \tag{1.200}$$

Note that we have

$$R(\mathbf{u}_h,\theta_h) \equiv R(\mathbf{u}_h,\theta_h,\rho,E,\nu) \tag{1.201}$$

and that R satisfies the following scaling properties

$$R(\frac{\mathbf{u}_h}{\sqrt{\alpha}},\frac{\theta_h}{\alpha},\alpha\rho,E,\nu) = \frac{1}{\alpha}R(\mathbf{u}_h,\theta_h,\rho,E,\nu) \tag{1.202}$$

$$R(\mathbf{u}_h,\alpha\theta_h,\rho,\alpha E,\nu) = \alpha R(\mathbf{u}_h,\theta_h,\rho,E,\nu) \tag{1.203}$$

Stability estimates

The equation (3.10) becomes

$$||D^{2+s}\mathbf{v}|| \leq C_{e,s}\sqrt{b\left(\left(\frac{-1}{\rho}\nabla\cdot\sigma\right)^{1+s/2}(\mathbf{v}),\left(\frac{-1}{\rho}\nabla\cdot\sigma\right)^{1+s/2}(\mathbf{v})\right)} \tag{1.204}$$

Note that

$$\Lambda+\mu = \frac{E}{2(1+\nu)(1-2\nu)} \quad , \quad \frac{\mu}{\Lambda+\mu} = 1-2\nu \tag{1.205}$$

Then, we get

$$C_{e,s} = c\frac{\rho^{(1+s)/2}}{(\Lambda+\mu)^{(2+s)/2}} \tag{1.206}$$

Note that we have

$$C_{e,s} \equiv C_{e,s}(\rho,E,\nu) \tag{1.207}$$

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

$$C_{e,s}(\alpha\rho,E,\nu) = \alpha^{(1+s)/2}C_{e,s}(\rho,E,\nu) \tag{1.208}$$

$$C_{e,s}(\rho,\alpha E,\nu) = \frac{1}{\alpha^{(2+s)/2}}C_{e,s}(\rho,E,\nu) \tag{1.209}$$

A posteriori estimates

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

$$\max_{\theta_i\notin\Theta}\left|\frac{\theta_h-\theta}{\theta_i-\theta}\right| \leq \delta \quad , \quad ||Q_\Theta\mathbf{u}_h||^2 \leq \delta \tag{1.210}$$

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

$$\frac{\theta_h - \theta}{\theta} \leq \frac{c}{1-\delta} C_{e,0} \sqrt{\rho} ||h^2 R(\mathbf{u}_h, \theta_h)|| \tag{1.211}$$

The estimates on the error in the discrete eigenvector are now

$$\sqrt{b(\mathbf{e}_\Theta, \mathbf{e}_\Theta)} \leq \frac{c}{1-\delta} C_{e,0} (1 + \max_{\theta_i \notin \Theta} \frac{\theta}{|\theta_i - \theta|}) \sqrt{\rho} ||h^2 R(\mathbf{u}_h, \theta_h)|| \tag{1.212}$$

$$\sqrt{a(\mathbf{e}_\Theta, \mathbf{e}_\Theta)} \leq \frac{c\sqrt{\rho}}{1-\delta} (C_c + C_{e,0} \max_{\theta_i \notin \Theta} \frac{\theta \theta_i^{1/2}}{|\theta_i - \theta|} h_{max}) ||hR(\mathbf{u}_h, \theta_h)|| \tag{1.213}$$

where C_c is related to the coercivity constant

$$||D\mathbf{v}|| \leq C_c \sqrt{a(\mathbf{v}, \mathbf{v})} \tag{1.214}$$

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

$$a(\mathbf{v}, \mathbf{v}) \geq 2\mu ||D\mathbf{v}|| \quad \Rightarrow \quad C_c = \frac{c}{\sqrt{2\mu}} \tag{1.215}$$

1.15.4 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 Salinas 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 affect the fractional error in the solution. It is an essential part of both Larsen's development and Ulrich's extension to elasticity. See equation 1.197.
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 1.187). Both the internal and the flux terms of the residuals are unaffected by most scaling operations provided that materials remain constant within an element. Note that the evaluation of the flux jump (equation 1.190) is unaffected by 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 Salinas are mass normalized, i.e. $u^T M u = I$. We renormalize for error estimation in the following manner.

1. A unitless mass matrix, $\bar{M}$ is computed using unit density material.

2. We compute a scale factor

$$m_\alpha = u^T \bar{M} u \tag{1.216}$$

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 1.206 inside the summation of equation 1.185. This maintains the proper scaling with respect the element stiffness terms.

A recent paper by Bernardi and Verfurth[21] has shown that explicit estimators 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 explicit estimators.

$$\|\mathbf{u_h} - \mathbf{u}\|_\alpha \leq c \sum_K \left(h \|R_i(\mathbf{u_h}, \theta_h)\|^2_{L^2(K)} + \sqrt{h} \| \frac{[\sigma_n(\mathbf{u_h})]}{2} \|^2_{L^2(\partial K)} \right)^{\frac{1}{2}} \tag{1.217}$$

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 fully account for discontinuous 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 1.217 in the form

$$\|u_h - u\|_\alpha \leq c \sum_K \left(\| h R_i(u_h, \theta_h) + \frac{\sqrt{h}}{Vol(K)} \frac{[\sigma_n(\mathbf{u_h})]}{2} \|^2_{L^2(K)} \right)^{\frac{1}{2}} \tag{1.218}$$

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 1.218.

The approach in Bernardi is to replace the estimator in equation 1.217 by

$$\|\mathbf{u_h} - \mathbf{u}\|_\alpha \leq c \sum_K \left({\mu_K}^2 \|R_i(\mathbf{u_h}, \theta_h)\|^2_{L^2(K)} + \mu_e \| \frac{[\sigma_n(\mathbf{u_h})]}{2} \|^2_{L^2(\partial K)} \right)^{\frac{1}{2}} \tag{1.219}$$

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

$$\frac{-\nabla \cdot \sigma}{\rho} = \theta \mathbf{u} \tag{1.220}$$

the corresponding bilinear forms as

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

and the corresponding interior residual as

$$R_i(\mathbf{u_h}, \theta_h) = |\frac{\nabla \cdot \sigma(\mathbf{u}_h)}{\rho} + \theta_h \mathbf{u}_h| \tag{1.221}$$

By dividing through by ρ, we include the density in the energy norm. This will be important later on when the coefficients in equation 1.219 are selected.

As in Bernardi, we need the following identities, which follow from equation 1.181

$$a(\mathbf{u} - \mathbf{u_h}, \mathbf{v}) \quad = \quad \theta b(\mathbf{u},\mathbf{v}) - a(\mathbf{u_h},\mathbf{v}) \tag{1.222}$$

$$\theta b(\mathbf{u},\mathbf{v}) - a(\mathbf{u_h},\mathbf{v}) = \sum_K \int_K \left(\theta \mathbf{u} + \frac{1}{\rho}\nabla \cdot \sigma(\mathbf{u}_h)\right) \mathbf{v} d\mathbf{x} - \sum_e \int_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot \mathbf{v} d\tau \tag{1.223}$$

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

$$a(\mathbf{u},\mathbf{v})_K \quad \leq \quad \|\mathbf{u}\|_\alpha \|\mathbf{v}\|_\alpha \tag{1.224}$$
$$\leq \quad \alpha_K \|\mathbf{u}\|_1 \|\mathbf{v}\|_1 \tag{1.225}$$

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

$$
\begin{aligned}
\|R_i(u_h,\theta_h)\|^2_{L^2(K)} &\leq \gamma_1^2 \int_K \left[\frac{1}{\rho}\nabla\cdot\sigma(\mathbf{u_h}) + \theta_h \mathbf{u_h}\right]\cdot \mathbf{w_K} d\mathbf{x} \\
&= -\gamma_1^2 \int_K \frac{1}{\rho}\sigma(\mathbf{u_h})\cdot\varepsilon(\mathbf{w_K}) d\mathbf{x} + \gamma_1^2 \int_K \theta_h \mathbf{u_h}\cdot\mathbf{w_K} \\
&= \gamma_1^2 a(\mathbf{u}-\mathbf{u_h},\mathbf{w_K})_K - \gamma_1^2\theta\int_K \mathbf{u}\cdot\mathbf{w_K} d\mathbf{x} + \gamma_1^2\theta_h\int_K \mathbf{u_h}\cdot\mathbf{w_K} d\mathbf{x} \\
&\leq \gamma_1^2\left[\|\mathbf{u}-\mathbf{u_h}\|_{\alpha(K)}\gamma_2 h_K^{-1}\alpha_K^{\frac{1}{2}} + \|\theta_h\mathbf{u_h}-\mathbf{u}\theta\|_{L^2(K)}\right] \\
&\times \|R_i(u_h,\theta_h)\|_{L^2(K)}
\end{aligned}
\tag{1.226}
$$

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,

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

Now is where a critical assumption comes into play. We assume here that the computed eigenvalue θ_h and eigenvector $\mathbf{u_h}$ are closer to the exact solution θ and $\mathbf{u}$ than any other eigenvalue/eigenvector pair. 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.[22] 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 1.226 we have

$$
\begin{aligned}
\| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|^2_{L^2(e)} &\leq \gamma_3^2 \int_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot \mathbf{w_e} d\tau \\
&= \gamma_3^2 \sum_K \int_K \left(\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta_h \mathbf{u}_h\right) \cdot \mathbf{w_e} - \gamma_3^2 \sum_K a(\mathbf{u} - \mathbf{u_h}, \mathbf{w_e}) \\
&+ \gamma_3^2 \sum_K \int_K (\theta \mathbf{u} - \theta_h \mathbf{u_h}) \cdot \mathbf{w_e} \\
&\leq c\gamma_3^2 \left(\sum_K \gamma_5 h_e^{\frac{1}{2}} \|R_i(u_h, \theta_h)\|_{L^2(K)} + \sum_K \gamma_4 h_e^{-\frac{1}{2}} \alpha_K^{\frac{1}{2}} \|\mathbf{u} - \mathbf{u_h}\|_\alpha \right. \\
&+ \left. \gamma_5 h_e^{\frac{1}{2}} \sum_K \|\mathbf{u}\theta - \mathbf{u_h}\theta_h\|_{L^2(K)} \right) \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)} \\
&\leq c\gamma_3^2 \left[\sum_K h_e^{-\frac{1}{2}} \alpha_K^{\frac{1}{2}} \|\mathbf{u} - \mathbf{u_h}\|_\alpha + \sum_K h_e^{\frac{1}{2}} \|\theta_h \mathbf{u_h} - \theta \mathbf{u}\|_{L^2(K)} \right] \\
&\times \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)}
\end{aligned} \tag{1.227}
$$

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 $\mu_e = h_e \max(\alpha_{K1}, \alpha_{K2})^{-1}$, 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.

$$
\begin{aligned}
\int_\Omega \left(\frac{1}{\rho}\nabla \cdot \sigma(\mathbf{u}_h) + \theta \mathbf{u}\right) \cdot (\mathbf{w} - \mathbf{w_h}) = -a(\mathbf{u_h}, \mathbf{w} - \mathbf{w_h}) + \\
\sum_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot (\mathbf{w} - \mathbf{w_h}) + \int_\Omega \theta \mathbf{u}(\mathbf{w} - \mathbf{w_h})
\end{aligned} \tag{1.228}
$$

This implies that

$$a(\mathbf{u_h}, \mathbf{w}-\mathbf{w_h}) = \sum_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot (\mathbf{w}-\mathbf{w_h})$$
$$+ \int_\Omega \theta \mathbf{u} \cdot (\mathbf{w}-\mathbf{w_h}) - \int_\Omega \left(\frac{1}{\rho}\nabla \cdot \sigma(\mathbf{u}_h) + \theta\rho\mathbf{u}\right) \cdot (\mathbf{w}-\mathbf{w_h}) \tag{1.229}$$

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

$$\|\mathbf{u}-\mathbf{u_h}\|_\alpha^2 = a(\mathbf{u}-\mathbf{u_h}, \mathbf{w}) = a(\mathbf{u}-\mathbf{u_h}, \mathbf{w}-\mathbf{w_h}) \tag{1.230}$$

where the last equality follows from Galerkin orthogonality. Breaking the previous expression into element-wise quantities, and using equation 1.229, we obtain

$$\begin{aligned}
\|\mathbf{u}-\mathbf{u_h}\|_\alpha^2 &= \sum_K a(\mathbf{u}-\mathbf{u_h}, \mathbf{w}-\mathbf{w_h}) \qquad (1.231)\\
&= \sum_K a(\mathbf{u}, \mathbf{w}-\mathbf{w_h}) - \sum_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot (\mathbf{w}-\mathbf{w_h})\\
&- \sum_K \int_K \theta\mathbf{u} \cdot (\mathbf{w}-\mathbf{w_h}) + \sum_K \int_K \left(\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\mathbf{u}\right) \cdot (\mathbf{w}-\mathbf{w_h})\\
&= \sum_K \int_K \left(\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\mathbf{u}\right) \cdot \mathbf{w}-\mathbf{w_h} - \sum_e \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \cdot (\mathbf{w}-\mathbf{w_h})\\
&\leq \sum_K \mu_K \|\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\mathbf{u}\|_{L^2(K)} \mu_K^{-1} \|\mathbf{w}-\mathbf{w_h}\|_{L^2(K)}\\
&+ \sum_e \mu_e^{\frac{1}{2}} \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)} \mu_e^{\frac{1}{2}} \|\mathbf{w}-\mathbf{w_h}\|_{L^2(e)}\\
&\leq \left[\sum_K \mu_K^2 \|\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\mathbf{u}\|_{L^2(K)}^2 + \sum_e \mu_e \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)}^2\right]^{\frac{1}{2}}\\
&\times \left[\sum_K \mu_K^{-2} \|\mathbf{w}-\mathbf{w_h}\|_{L^2(K)}^2 + \sum_e \mu_e^{-1} \|\mathbf{w}-\mathbf{w_h}\|_{L^2(e)}^2\right]^{\frac{1}{2}}
\end{aligned}$$

We now use equation 2.16 in Bernardi's paper, which shows that

$$\left[\sum_K \mu_K^{-2} \|\mathbf{w}-\mathbf{w_h}\|_{L^2(K)}^2 + \sum_e \mu_e^{-1} \|\mathbf{w}-\mathbf{w_h}\|_{L^2(e)}^2\right]^{\frac{1}{2}} \leq c\|\mathbf{w}\|_\alpha \tag{1.232}$$

With this result, we have

$$\|\mathbf{u}-\mathbf{u_h}\|_\alpha \leq c\left[\sum_K \mu_K^2 \|\nabla \cdot \frac{1}{\rho}\sigma(\mathbf{u}_h) + \theta\rho\mathbf{u}\|_{L^2(K)}^2 + \sum_e \mu_e \| \left[\frac{1}{\rho}\sigma_n(\mathbf{u_h})\right] \|_{L^2(e)}^2\right]^{\frac{1}{2}} \tag{1.233}$$

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.

1.15.5 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 1.15.4, equation 1.216.
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 1.192. Integrate over all faces (by summing at surface gauss points) to determine R_{flux} (eq 1.190).
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,
   $$\begin{aligned} \bar{\rho}^2 &= \int_{\Omega_e} (a_1 + R_{flux})^2 d\Omega_e \\ &\approx \sum_i^{N_{gauss}} w_i (a_1(x_i) + R_{flux})^2 \end{aligned}$$
4. Compute the global contribution to the error. For elements with linear shape functions, this may be written,
   $$\frac{|\lambda - \lambda_h|}{\lambda} \leq c \left(\sum_{e=1}^{N_e} (C_{e,0} h_e^2 \bar{\rho})^2 \right)^{\frac{1}{2}}. \tag{1.234}$$
   Where (as shown in section 1.15.3, equation 1.206),
   $$C_{e,0}^2 = \frac{\rho}{(\Lambda + \mu)^2}$$
   and ρ, Λ and μ are the material density and the Lamé constants respectively. The more general expression for elements of order p is,
   $$\frac{|\lambda - \lambda_h|}{\lambda^{(p+1)/2}} \leq c \left(\sum_{e=1}^{N_e} (C_{e,p-1} h_e^{(p+1)} \bar{\rho})^2 \right)^{\frac{1}{2}}. \tag{1.235}$$
   We note that although the constant, c, in equation 1.234 is not known completely, it is usually 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.

1.15.6 Approach II - quantity of interest estimator

In,[19] an error estimator is derived for the elasticity equation, using the eigenvalues as the quantity of interest. The estimate is of the form

$$\eta^{\lambda}_{low} = -\eta^{2}_{upp} \tag{1.236}$$
$$\eta^{\lambda}_{upp} = -\eta^{2}_{low} \tag{1.237}$$

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 so-called *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 residual method involves solving the following linear system on each element

$$-B(\Phi_K, v) = R(v,0) + \int_{\partial K} g_{\gamma,K} v ds \quad \forall v \in W_K \tag{1.238}$$

or

$$K_b a = f \tag{1.239}$$

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 ith 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 1.239 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 simply

$$R(v,0) = B(u_h, v) - \lambda_h M(u_h, v) \tag{1.240}$$

Equation 1.240 can be most efficiently evaluated using the following method.[23] We evaluate the first term first.

$$B(u_h, v) = \int_K B^T_{bubble} \sigma(x) dx \tag{1.241}$$

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 1.241 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 1.241, 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.

[10] for consistent mass only.

The second term can be evaluated in a similar way.

$$M(u_h, v) = \int_K u_h(x)v(x)dx \tag{1.242}$$

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 ith bubble shape function are as follows

$$\int_K u_{1h}(x)\phi_i(x)dx \tag{1.243}$$

$$\int_K u_{2h}(x)\phi_i(x)dx \tag{1.244}$$

$$\int_K u_{3h}(x)\phi_i(x)dx \tag{1.245}$$

$$\tag{1.246}$$

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

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

$$\int_{\partial K} g_{\gamma,K} v ds \quad = \quad \int_{\partial K} [\sigma_{ij} n_j] v ds \tag{1.247}$$

where $[\sigma_{ij} n_j]$ 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.

$$[\sigma_{ij} n_j] = \frac{1}{2}\left(\sigma^a_{ij} n_j + \sigma^b_{ij} n_j\right) \tag{1.248}$$

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 1.238 and equation 1.247 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

$$\int_{\partial K} g_{\gamma,K} v ds = \sum_{i=1}^{N_{faces}} \int_{\partial K_i} g_{\gamma,K} v ds \tag{1.249}$$

where ∂K_i is the ith 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

$$\int_{\partial K_i} g_{\gamma,K} \phi_{bubble} ds \tag{1.250}$$

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 1.239 are solved on each element, the upper bound, η_{up} from equation 1.237 can be computed as follows

$$\eta_{upp} = \sqrt{\sum_K B(\Phi_K, \Phi_K)} \tag{1.251}$$

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,

$$\Phi_K = \sum_{i=1}^{Nshape_{bubble}} a_i N_i \tag{1.252}$$

then

$$\eta_{upp} = \sqrt{\sum_K B(\Phi_K, \Phi_K)} = \sqrt{\sum_K a^T K_b a} \tag{1.253}$$

Finally, using equation 1.237, 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,[19] 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

$$\eta_{low} = \frac{|R_p(\chi, 0)|}{\sqrt{B(\chi, \chi)}} \tag{1.254}$$

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 1.240), and then summing globally.

Summarizing, in order 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 1.239, 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 1.239. This consists of the interior part, equation 1.240, and the boundary part, equation 1.247.
3. Loop over all elements and solve the linear systems 1.239, to obtain the error functions Φ_K.
4. Compute the upper bound on the error in the eigenvalue using equation 1.253.
5. Compute the lower bound on the error in the eigenvalue using equation 1.254.

2 Elements

Structural dynamics is a rich and extensive field. Finite element tools such as **Salinas** 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 1.

2.1 Isoparametric Solid Elements. Selective Integration

The following applies to any solid isoparametric element, but is implemented in the code on elements with linear shape functions (such as hex8 or wedge6). This discussion addresses calculation of relevant operators on the shape functions and eventual integration into the stiffness matrices. [11]

2.1.1 Derivation

We begin with the separation of the strain into deviatoric and dilitational parts so that their contributions to the stiffness matrix can be computed separately. This is part of the strategy for avoiding overstiffness with respect to bending.

The strain energy density in the case of an isotropic, linearly elastic material is:

$$p = \frac{1}{2}(2G\varepsilon + \lambda tr(\varepsilon)I) \bullet \varepsilon \tag{2.1}$$

with some re-arrangement, this can be shown to be:

$$p = G\hat{\varepsilon} \bullet \hat{\varepsilon} + \frac{1}{2}\beta(tr(\varepsilon))^2 \tag{2.2}$$

where $\hat{\varepsilon} = \varepsilon - \frac{1}{3}tr(\varepsilon)I$.

Having separated the part of the strain energy density due to deviatoric part of the strain from the part of the strain energy density due to the dilitational part of the strain, we shall integrate them separately. First, we must determine how to express the strains in terms of nodal degrees of freedom.

We know that the deformation field is linear in the nodal degrees of freedom and that the displacement gradient is also, so we should be able to expand each of those quantities 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:

$$\vec{u}(x) = \tilde{N}^{P_j}(x) u_{s_j}^{P_j} \vec{e}_{s_j} \tag{2.3}$$

where summation takes place over the degree of freedom j.

[11] This development is based on work by Dan Segalman.

Similarly, the displacement gradient is:

$$\vec{\nabla}\vec{u}(x) = (\frac{\partial}{\partial x_k})\tilde{N}^{P_j}(x)u_{s_j}^{P_j}\vec{e}_{s_j}\vec{e}_k \tag{2.4}$$

We now define the shape deformation tensor W^j corresponding to the j th nodal degree of freedom:

$$W^j(x) = (\frac{\partial}{\partial u_{s_j}^{P_j}})\vec{\nabla}\vec{u}(x) \tag{2.5}$$

which, with Equation 2.4 yields:

$$W^j(x) = (\frac{\partial}{\partial x_k})\tilde{N}^{P_j}(x)\vec{e}_{s_j}\vec{e}_k \tag{2.6}$$

The symmetric part of this tensor is:

$$S^j(x) = \frac{1}{2}(W^j(x) + W^j(x)^T) \tag{2.7}$$

and the strain tensor is

$$\varepsilon(x) = S^j(x)u_{s_j}^{P_j} \tag{2.8}$$

From the above, we construct the dilitational and deviatoric portions of the strain in terms of the nodal displacement components:

$$tr(\varepsilon(x)) = b^j(x)u_{s_j}^{P_j} \tag{2.9}$$

where

$$b^j(x) = tr(S^j(x)) \tag{2.10}$$

Similarly,

$$\hat{\varepsilon}(x) = \hat{B}^j(x)u_{s_j}^{P_j} \tag{2.11}$$

where

$$\hat{B}^j(x) = S^j(x) - \frac{1}{3}b^j(x)I \tag{2.12}$$

The stiffness matrix is evaluated using the constitutive equation (Equation 2.2) and the following definition:

$$K_{m,n} = \frac{\partial^2}{\partial u_{s_m}^{P_m}\partial u_{s_n}^{P_n}}\int_{volume} p(x)dV(x) \tag{2.13}$$

This plus our expressions for strain in terms of the nodal degrees of freedom yield us the following expression for element stiffness:

$$\begin{aligned} K_{m,n} = G\int_{volume}(\hat{B}^m(x))^T \bullet \hat{B}^n(x)dV(x) \\ +\beta\int_{volume} b^m(x)b^n(x)dV(x) \end{aligned} \tag{2.14}$$

2.2 Implementation

From the above it is seen that once the shape deformation tensor W^j is found, the rest of the calculation follows naturally. The calculation of the components of that tensor is presented here. The components of W^j are

$$W^j_{mn} = \vec{e}_m \cdot W^j \cdot \vec{e}_n \tag{2.15}$$

$$= \delta_{m,s_j}(\frac{\partial}{\partial x_n})\tilde{N}^{P_j}(x) \tag{2.16}$$

The partial derivative $(\frac{\partial}{\partial x_n})\tilde{N}^{P_j}(x)$ is calculated from

$$(\frac{\partial}{\partial x_n})\tilde{N}^{P_j}(x(\xi)) = (\frac{\partial}{\partial \xi_\alpha})N^{P_j}(\xi)J^{-1}_{\alpha,n} \tag{2.17}$$

where

$$J_{m,\gamma} = \frac{\partial}{\partial \xi_\gamma}x_m(\xi) \tag{2.18}$$

and

$$N(\xi) = \tilde{N}(x(\xi)) \tag{2.19}$$

The issue of selective integration in the elements is discussed in Appendix B. The formulation discussed there applies to all the isoparametric solid elements.

2.3 Mean Quadrature Element with Selective Deviatoric Control

In this section we discuss the implementation of the mean quadrature element in Salinas. This work is a result of a collaboration with Sam Key.[24]

We first examine the element stiffness matrix resulting from a fully integrated element

$$K = \int_V B^T C B dV \tag{2.20}$$

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 very 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, $\tilde{B}$,

$$\tilde{B} = \int_V B dV \tag{2.21}$$

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. In the approach described

here, we use the latter method. We note that both B and $\tilde{B}$ can be decomposed into their volumetric and deviatoric components, i.e.

$$\begin{aligned} \tilde{B} &= \tilde{B}_V + \tilde{B}_D \\ B &= B_V + B_D \end{aligned} \tag{2.22}$$

With these decompositions, we define

$$\hat{B} = \tilde{B}_V + \tilde{B}_D + sd(B_D - \tilde{B}_D) \tag{2.23}$$

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

$$K = \int_V \hat{B}^T C \hat{B} dV \tag{2.24}$$

2.4 Bubble Element

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,[25],[26].[27] 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

$$\mathbf{u} = \sum_{\mathbf{i=1}}^{\mathbf{8}} \mathbf{u_i N_i}(\xi) = \mathbf{u^T N} \tag{2.25}$$

where u is the element displacement, N_i is the ith 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

$$\mathbf{u} = \sum_{\mathbf{i=1}}^{\mathbf{8}} \mathbf{u_i N_i}(\xi) + \sum_{\mathbf{i=1}}^{\mathbf{3}} \mathbf{a_i P_i}(\xi) = \mathbf{u^T N} + \mathbf{a^T P} \tag{2.26}$$

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

$$\varepsilon = \mathbf{Bu} + \mathbf{Ga} \tag{2.27}$$

where B and G are the appropriate derivatives of the shape functions. We note that B is a $6x24$ matrix, whereas G is a $6x9$ matrix. See,[25][26] for the exact forms of these matrices.

The corresponding element stiffness and load terms can be assembled into a 2x2 system

$$\begin{bmatrix} K & E^T \\ E & H \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{2.28}$$

where $K = \int_e B^T CBdV$ is the 24x24 element stiffness matrix corresponding to standard element shape functions, $H = \int_e G^T CGdV$ is the 9x9 stiffness matrix corresponding to bubble shape functions, $E = \int_e G^T CBdV$ is the 9x24 matrix corresponding to products of bubble and standard shape functions, and f is the element load vector. Since the bubble unknowns a are local to each element, they can be condensed out, which yields a modified element stiffness matrix

$$\hat{K} = K - E^T H^{-1} E \tag{2.29}$$

Note that K is still a 24x24 matrix.

It has been shown that the bubble hex element does not pass the patch test unless a correction is made to the element formulation. There are two options for this correction. The first[25] involves evaluating the matrix G at the centroid of the element rather than at the Gauss points. The second approach[26] consists of subtracting from the matrix G its average value. Both approaches yield an element that passes the patch test, and thus convergence is assured.

In Salinas, 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.

$$\hat{G} = G - \frac{1}{V_e} \int_e G dV \tag{2.30}$$

Then, we simply replace G with $\hat{G}$ in the above equations. We note that, in the implementation of this element in Salinas, it was found that after implementing the correction described above, the element passed the patch test. Without the correction, the element failed all of the patch tests.

With the bubble element, the stresses vary through the thickness. In order to compute the stresses at any particular point within the element, we need to recover the strains. These are given in equation 2.27. However, an additional task is to compute the bubble degrees of freedom, since only the displacement degrees of freedom are calculated during the solution procedure. From equation 2.28, the bubble degrees of freedom can be computed from the displacements as

$$\mathbf{a} = \mathbf{H^{-1}Eu} \tag{2.31}$$

where $\mathbf{u}$ is the element displacement vector. Given $\mathbf{a}$, we can then compute the strains from equation 2.27, and then the stresses can be computed in the standard way.

2.4.1 Nonlinear analysis with bubble element

The bubble element can be used in nonlinear analysis. A brief description of the procedure is given in.[26] More details will be given here. In,[26] an assumed strain approach was used rather than the assumed displacement method, but the two reduce to the same procedure.

We will give the necessary modifications for a nonlinear static analysis. The equations that need to be satisfied are

$$F^{int}(\mathbf{u}, \alpha) = \mathbf{F^{ext}} \tag{2.32}$$

More specifically, this breaks down to two separate equations

$$F_1^{int} = \int_\Omega B^T \sigma d\Omega = F^{ext} \tag{2.33}$$

$$F_2^{int} = \int_\Omega G^T \sigma d\Omega = 0 \tag{2.34}$$

$$\tag{2.35}$$

The stress is given by $\sigma = C\varepsilon$, where ε is given by equation 2.27.

Next, we expand the expressions for internal force in a Taylor series, and truncate after the first two terms. In the following, 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.

$$F_1^{int}(\mathbf{u},\alpha) \approx \mathbf{F_1^{int}}(\hat{\mathbf{u}},\hat{\alpha}) + \frac{\partial \mathbf{F_1^{int}}}{\partial \mathbf{u}} \blacksquare \mathbf{u} + \frac{\partial \mathbf{F_1^{int}}}{\partial \alpha} \blacksquare \alpha \tag{2.36}$$

$$F_2^{int}(\mathbf{u},\alpha) \approx \mathbf{F_2^{int}}(\hat{\mathbf{u}},\hat{\alpha}) + \frac{\partial \mathbf{F_2^{int}}}{\partial \mathbf{u}} \blacksquare \mathbf{u} + \frac{\partial \mathbf{F_2^{int}}}{\partial \alpha} \blacksquare \alpha \tag{2.37}$$

$$\tag{2.38}$$

We define

$$K_T = \frac{\partial F_1^{int}}{\partial \mathbf{u}} \tag{2.39}$$

$$E_T = \frac{\partial F_1^{int}}{\partial \alpha} \tag{2.40}$$

$$H_T = \frac{\partial F_2^{int}}{\partial \alpha} \tag{2.41}$$

$$\tag{2.42}$$

where the subscript T denotes tangent matrices that are computed at the current configuration. Using these definitions and substituting equations 2.38 into equations 2.35, we obtain

$$\begin{bmatrix} K_T & (E^T)_T \\ E_T & H_T \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{a} \end{bmatrix} = \begin{bmatrix} Res_{\mathbf{u}} \\ Res_\alpha \end{bmatrix} \tag{2.43}$$

where

$$Res_{\mathbf{u}} = F^{ext} - F_1^{int}(\hat{\mathbf{u}},\hat{\alpha}) \tag{2.44}$$

$$Res_\alpha = -F_2^{int}(\hat{\mathbf{u}},\hat{\alpha}) \tag{2.45}$$

$$\tag{2.46}$$

More detailed expressions for the tangent matrices will now be given. We note that, for example, in equation 2.35, both σ and the matrix B depend on displacement u and bubble unknowns α. Thus,

the chain rule is needed to compute the following expressions.

$$K_T = \frac{\partial \int_\Omega B^T \sigma d\Omega}{\partial \mathbf{u}} = \int_\Omega \frac{\partial B^T}{\partial \mathbf{u}} \sigma d\Omega + \int_\Omega B^T \frac{\partial \sigma}{\partial \mathbf{u}} d\Omega \tag{2.47}$$

$$E_T = \frac{\partial \int_\Omega B^T \sigma d\Omega}{\partial \alpha} = \int_\Omega \frac{\partial B^T}{\partial \alpha} \sigma d\Omega + \int_\Omega B^T \frac{\partial \sigma}{\partial \alpha} d\Omega \tag{2.48}$$

$$H_T = \frac{\partial \int_\Omega G^T \sigma d\Omega}{\partial \alpha} = \int_\Omega \frac{\partial G^T}{\partial \alpha} \sigma d\Omega + \int_\Omega G^T \frac{\partial \sigma}{\partial \alpha} d\Omega \tag{2.49}$$

$$\tag{2.50}$$

In each of these expressions, the first term on the right hand side represents a geometric stiffness term, whereas the second term represents the material stiffness term. Next, in order to evaluate terms like $\frac{\partial B^T}{\partial \mathbf{u}}$ and $\frac{\partial B^T}{\partial \alpha}$, we use the deformation gradient. We use the notation $\mathbf{x} = \mathbf{u} + \mathbf{X}$, where $\mathbf{x}$ is the current configuration, $\mathbf{u}$ is the displacement, and $\mathbf{X}$ is the initial configuration.

$$e = \frac{1}{2}(F^T F - I) \tag{2.51}$$

$$B = \frac{\partial \varepsilon}{\partial u} = F \frac{\partial F}{\partial u} \tag{2.52}$$

$$\frac{\partial B}{\partial u} = F \frac{\partial^2 F}{\partial u^2} + \frac{\partial F}{\partial u} \frac{\partial F}{\partial u} = \frac{\partial F}{\partial u} \frac{\partial F}{\partial u} \tag{2.53}$$

$$\tag{2.54}$$

where the last identity follows from the fact that $\frac{\partial^2 F}{\partial u^2} = 0$. This can be seen from the following relations.

$$F = \frac{\partial x}{\partial X} = I + \frac{\partial u}{\partial X} = I + u^T \frac{DN}{DX} + \alpha^T \frac{DP}{DX} \tag{2.55}$$

$$\frac{\partial F}{\partial u} = \frac{DN}{DX} \tag{2.56}$$

$$\frac{\partial^2 F}{\partial u^2} = 0 \tag{2.57}$$

$$\tag{2.58}$$

Similarly, we can construct these equations for the bubble functions

$$e = \frac{1}{2}(F^T F - I) \tag{2.59}$$

$$G = \frac{\partial \varepsilon}{\partial \alpha} = F \frac{\partial F}{\partial \alpha} \tag{2.60}$$

$$\frac{\partial G}{\partial \alpha} = F \frac{\partial^2 F}{\partial \alpha^2} + \frac{\partial F}{\partial \alpha} \frac{\partial F}{\partial \alpha} = \frac{\partial F}{\partial \alpha} \frac{\partial F}{\partial \alpha} \tag{2.61}$$

$$\tag{2.62}$$

where similar identities have been used

$$F = \frac{\partial x}{\partial X} = I + \frac{\partial u}{\partial X} = I + u^T \frac{DN}{DX} + \alpha^T \frac{DP}{DX} \tag{2.63}$$

$$\frac{\partial F}{\partial \alpha} = \frac{DP}{DX} \tag{2.64}$$

$$\frac{\partial^2 F}{\partial \alpha^2} = 0 \tag{2.65}$$

$$\tag{2.66}$$

For the cross terms, we have

$$e = \frac{1}{2}(F^T F - I) \tag{2.67}$$

$$B = \frac{\partial \varepsilon}{\partial u} = F \frac{\partial F}{\partial u} \tag{2.68}$$

$$\frac{\partial B}{\partial \alpha} = F \frac{\partial^2 F}{\partial u \partial \alpha} + \frac{\partial F}{\partial u}\frac{\partial F}{\partial \alpha} = \frac{\partial F}{\partial u}\frac{\partial F}{\partial \alpha} \tag{2.69}$$

$$\tag{2.70}$$

where, again we justify that the second term vanishes as follows

$$F = \frac{\partial x}{\partial X} = I + \frac{\partial u}{\partial X} = I + u^T \frac{DN}{DX} + \alpha^T \frac{DP}{DX} \tag{2.71}$$

$$\frac{\partial F}{\partial u} = \frac{DN}{DX} \tag{2.72}$$

$$\frac{\partial^2 F}{\partial u \partial \alpha} = 0 \tag{2.73}$$

$$\tag{2.74}$$

In a similar manner as was done for the linear element, the bubble degrees of freedom can be condensed from equations 2.46. This results in the equation

$$(K_T - E_T^T H_T^{-1} E_T)\Delta \mathbf{u} = \mathbf{Res_u} - \mathbf{E_T^T H_T^{-1} Res}_\alpha \tag{2.75}$$

Thus, the full tangent operator for the bubble element is given by

$$K_T - E_T^T H_T^{-1} E_T \tag{2.76}$$

the internal force is given by

$$F_1^{int}(\hat{\mathbf{u}}, \hat{\alpha}) - E_T^T H_T^{-1} F_2^{int}(\hat{\mathbf{u}}, \hat{\alpha}) \tag{2.77}$$

and the residual is given by two terms

$$Res_{\mathbf{u}} - E_T^T H_T^{-1} Res_\alpha \tag{2.78}$$

These equations fully describe the nonlinear analysis of the bubble element.

2.5 Quadratic Isoparametric Solid Elements

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 section 2.1 and Appendix B for definitions of G and β.) Therefore, G=0 and β=0 are good values for such elements.

2.6 Wedge elements

2.6.1 Shape Functions

The shape functions are given explicitly in *Hughes*, (ref. 28). 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:

$$N_k = A_0^k + A_1^k r + A_2^k s + A_3^k \xi + A_4^k r\xi + A_5^k s\xi \tag{2.79}$$

The shape functions and the coefficients are given in the following table:

Shape Function	A_0	A_1	A_2	A_3	A_4	A_5
$N_1 = \frac{1}{2}(1-\xi)r$		$\frac{1}{2}$			-$\frac{1}{2}$	
$N_2 = \frac{1}{2}(1-\xi)s$			$\frac{1}{2}$			$-\frac{1}{2}$
$N_3 = \frac{1}{2}(1-\xi)t$	$\frac{1}{2}$	-$\frac{1}{2}$	-$\frac{1}{2}$	-$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$N_4 = \frac{1}{2}(1+\xi)r$		$\frac{1}{2}$			$\frac{1}{2}$	
$N_5 = \frac{1}{2}(1+\xi)s$			$\frac{1}{2}$			$\frac{1}{2}$
$N_6 = \frac{1}{2}(1+\xi)t$	$\frac{1}{2}$	-$\frac{1}{2}$	-$\frac{1}{2}$	$\frac{1}{2}$	-$\frac{1}{2}$	-$\frac{1}{2}$

2.6.2 Quadrature

Three reasonable quadratures for wedges that come to mind are indicated in the following table:

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/$\sqrt{3}$
	1/3	1/6	-1/$\sqrt{3}$
	1/6	1/3	-1/$\sqrt{3}$
	1/6	1/6	1/$\sqrt{3}$
	1/3	1/6	1/$\sqrt{3}$
	1/6	1/3	1/$\sqrt{3}$

2.7 Tet10 elements

The 4-point integration is given in *Hughes* (see 29), and the 16-point integration is given in *Jinyun*. It is believed that a higher order integration is needed for the mass matrix than the stiffness matrix and that the reason is that the mass matrix involves higher degree polynomials. (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.)

2.8 Calculating shape functions and gradients of the Hex20 element

Using a 3D Pascal's triangle, we can construct 20 polynomials of the form,

$$p_i = \varepsilon_1^{r_i}\varepsilon_2^{s_i}\varepsilon_3^{t_i}$$

where the r_i, s_i and t_i $(i = 1,\ldots,20)$ are integers satisfying,

$$r_i^2 + s_i^2 + t_i^2 \leq 7$$

These terms may be constructed with the following loop.[12]

```
count=0
for I = 0 to 7
  for J = 0 to 7
    for K = 0 to 7
      if I^2 + J^2 + K^2 <= 7
        count = count + 1
          r(count) = I
          s(count) = J
          t(count) = K
      endif
    endfor
  endfor
endfor
```

We require 20 shape functions N_i, with $i = 1,\ldots,20$, that satisfy the conditions that $N_i = 1$ at node i and $N_i = 0$ at every other node. This results in 20 equations at each node. Expressing the N_i as linear combinations of the p_i, we can write,

$$\vec{N} = A\vec{p} \tag{2.80}$$

where A is a 20x20 matrix. We want to find the 400 term $A-$matrix values. For each node, we have 20 equations and there are 20 nodes; so, there are 400 equations for the 400 unknowns. 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, $A\vec{p}(\vec{\varepsilon}_i) = \vec{e}_i$. So,

$$[\vec{\varepsilon}_1, \vec{\varepsilon}_2, \ldots, \vec{\varepsilon}_{20}] = [A][\vec{p}(\vec{\varepsilon}_1), \vec{p}(\vec{\varepsilon}_2), \ldots, \vec{p}(\vec{\varepsilon}_{20})]$$

[12] This is how the `rst` matrix in `Hex20.C` was created.

or,

$$I = AP$$

or,

$$A = P^{-1}$$

This matrix A is the matrix *"hc20"* in `Hex20.C`.

Not only can the shape functions be expressed as a linear combination of the p_i, but so can the derivatives, $\frac{\partial \vec{N}}{\partial \varepsilon_j}$, $(j = 1,2,3)$. Differentiating equation 2.80, we have

$$\frac{\partial \vec{N}}{\partial \varepsilon_j} = A \frac{\partial \vec{p}}{\partial \varepsilon_j}$$

but the $\partial \vec{p}/\partial \varepsilon_j$ may be written as a linear combination of the p_k via the following three equations.

$$\frac{\partial p_i}{\partial \varepsilon_1} = r_i \varepsilon_1^{r_i - 1} \varepsilon_2^{s_i} \varepsilon_3^{t_i} \tag{2.81}$$

$$\frac{\partial p_i}{\partial \varepsilon_2} = s_i \varepsilon_1^{r_i} \varepsilon_2^{s_i - 1} \varepsilon_3^{t_i} \tag{2.82}$$

$$\frac{\partial p_i}{\partial \varepsilon_3} = t_i \varepsilon_1^{r_i} \varepsilon_2^{s_i} \varepsilon_3^{t_i - 1} \tag{2.83}$$

while noting that equations 2.81, 2.82 and 2.83 are zero if r_i, s_i, or t_i is zero, respectively. We would like to find the matrix B_j with $j = 1,2,3$ 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,

$$\left[\frac{\partial \vec{N}}{\partial \varepsilon_j}(\vec{\varepsilon}_1), \frac{\partial \vec{N}}{\partial \varepsilon_j}(\vec{\varepsilon}_2), \ldots, \frac{\partial \vec{N}}{\partial \varepsilon_j}(\vec{\varepsilon}_{20}) \right] = B_j \left[\vec{p}(\vec{\varepsilon}_1), \vec{p}(\vec{\varepsilon}_2), \ldots, \vec{p}(\vec{\varepsilon}_{20}) \right] \tag{2.84}$$

Matrix equation 2.84 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.

2.9 Anisotropic Elasticity

Anisotropic elasticity requires 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. A derivation of the formulae for rotating those matrices is given in A.

2.10 Triangular Shell Element

The triangular shell element (TriaShell) is derived as follows. The bending d.o.f. (w, θ_x, θ_y) and the membrane d.o.f. (u, v, θ_z) are decoupled. The idea is to obtain the membrane response using Allman's triangle and the bending response using the discrete Kirchoff triangular (DKT) element.

2.10.1 Allman's Triangular Element

Using the formulation given in Ref. 30 and replacing $\cos(\gamma_{ij}) = \frac{y_{ji}}{l_{ij}}$ and $\sin(\gamma_{ij}) = \frac{-x_{ji}}{l_{ij}}$, we get

$$u = u_1\psi_1 + u_2\psi_2 + u_3\psi_3 + \frac{1}{2}y_{21}(\omega_2 - \omega_1)\psi_1\psi_2 + \frac{1}{2}y_{32}(\omega_3 - \omega_2)\psi_2\psi_3 + \frac{1}{2}y_{13}(\omega_1 - \omega_3)\psi_3\psi_1 \tag{2.85}$$

$$v = v_1\psi_1 + v_2\psi_2 + v_3\psi_3 + \frac{1}{2}x_{21}(\omega_2 - \omega_1)\psi_1\psi_2 - \frac{1}{2}x_{32}(\omega_3 - \omega_2)\psi_2\psi_3 - \frac{1}{2}x_{13}(\omega_1 - \omega_3)\psi_3\psi_1 \tag{2.86}$$

The stiffness and mass matrices ($[K]_{AT}, [M]_{AT}$) are found using general finite element procedures. Unfortunately, a mechanism exists for this element if the deformations are all zero and the rotations are all the same value. Cook *et al.*[5] have a "fix" for this which has been implemented to avoid undesirable low energy modes produced by this mechanism.

2.10.2 Discrete Kirchoff Element

As for the DKT[31] element, things are not so simple. The nine d.o.f. element is obtained by transforming a twelve d.o.f. element with mid-side nodes to a triangle with the nodes at the vertices only. This is obtained as follows. Using Kirchoff 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 d.o.f. 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 *at al.*. Using a nine d.o.f. 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 is taken from Ref. 32, as

$$w = \alpha_1\psi_1 + \alpha_2\psi_2 + \alpha_3\psi_3 + \alpha_4\psi_1\psi_2 + \alpha_5\psi_2\psi_3 + \alpha_6\psi_3\psi_1 + \alpha_7{\psi_1}^2\psi_2 + \alpha_8{\psi_2}^2\psi_3 + \alpha_9{\psi_3}^2\psi_1 \tag{2.87}$$

For the AT and DKT elements, the stiffness and mass matrices are derived with the help of Maple. The consistent mass matrix is derived using "normal" finite element procedures. If a lumped mass matrix is requested then the mass matrix terms associated with the translation d.o.f. are found in the "normal" sense. However, mass matrix terms for the rotational d.o.f. are set to $\frac{1}{125}$ of the translation terms.

In summary, the code has been written which uses the AT and DKT element use in combination as a shell element. The stiffness matrices are calculated without complication. The mass matrix for the AT element is also derived without complication. The mass matrix for the DKT element is derived using an incomplete polynomial, but the results obtained should not be effected very much.

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 3. 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 4. Comparison of deflections at Node 3

2.10.3 Verification and Validation

The AT element is verified by comparing calculated results with the results published by Allman in Ref. 30. The square plate in pure bending and a cantilevered beam with a parabolic tip load are used as verification examples. The mass matrix is not verified except to note that the mass is conserved in the u, v directions.

The DKT element is validated by using the experimental data published by Batoz *et al.* in Ref. 31 for a triangular fin. 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. 33. Tables 3 and 4 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 SALINAS using this DKT/AT element.

2.11 Triangular Shell - Tria3

The triangular shell used most in Salinas is the **Tria3** element developed by Carlos Felippa of the University of Colorado in Boulder. This element is very similar to the **TriaShell** element presented in section 2.10. Full details of the theory behind the element is out of the scope of this document, but details may be found in references 34, 35 and 36.

2.12 Two Node Beam

This is the definition for a Beam element based on Cook's development (see pp 113-115 of reference 5).

The beam uses underintegrated cubic shape functions. Only isotropic material models are supported. Torsional affects are accounted for in the axis of the beam. The beam is uniform in area and bending moments, i.e. they are not a function of position in the beam.

The following parameters are read from the exodus file.

1. The cross subsectional area of the beam (Attribute 1)

2. The orientation of the beam (Attributes 2, 3 and 4)

   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 completely 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.

3. The first bending moment, I1. (Attribute 5).

4. The second bending moment. I2. (Attribute 6).

5. The torsional moment, J. (Attribute 7).

2.13 Truss

This is the definition for a Truss element based on pages 214-216 of Cook (ref 5).

The truss uses linear shape functions. Unlike the truss elements used by Nastran, there is no torsional stiffness. The truss is uniform in area, i.e. the area is not a function of position in the truss.

The following parameters are read from the exodus file.

1. The cross subsectional area of the truss (Attribute 1)

2.14 Springs

The *Spring* element is the simplest one dimensional element. It has no mass. Entries in the stiffness matrix are added by hand. Note the following.

- The force generated in a *Spring* element should be colinear with the the nodes. Typically spring elements connect coincident nodes so that no torques are generated.
- *Springs* attach 3 degrees of freedom. In the event that some of the spring constants are zero, there is no effective stiffness for that associated degree of freedom. However, the degree of freedom will remain in the A-set matrices. This will be a problem if the other degrees of freedom are not attached to other elements which provide stiffness entries connecting them to the remainder of the model. For an understanding of the various solution spaces (such as the A-set), see section 4.1.

The data for spring elements is entered in the input file. Three values are given, Kx, Ky, and Kz. This results in a 6x6 element stiffness matrix,

$$K' = \begin{pmatrix} K_x & 0 & 0 & -K_x & 0 & 0 \\ 0 & K_y & 0 & 0 & -K_y & 0 \\ 0 & 0 & K_z & 0 & 0 & -K_z \\ -K_x & 0 & 0 & K_x & 0 & 0 \\ 0 & -K_y & 0 & 0 & K_y & 0 \\ 0 & 0 & -K_z & 0 & 0 & K_z \end{pmatrix} \tag{2.88}$$

Notice that K' is blocked. It could be written more simply,

$$K' = \begin{pmatrix} K'_{11} & -K'_{11} \\ -K'_{11} & K'_{11} \end{pmatrix}$$

The rotation matrix for the two endpoints is block diagonal.[13] As a result, the stiffness matrix in the basic coordinate system can be written,

$$K = \begin{pmatrix} K_{11} & K_{12} \\ K_{12} & K_{11} \end{pmatrix}$$

where,

$$K_{ij} = R^T K'_{ij} R$$

and R is the 3x3 rotation matrix of subsection 2.20.

[13] In other words, the displacments in a rotated frame are related to the unrotated frame by a transformation matrix of the form,

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = [T] \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix}$$

where,

$$T = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$$

Here, R_i is a 3x3 rotation matrix, and because the two nodes of the spring must rotate together, $R_1 = R_2$

2.15 Gap Elements

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 2.88 with the following 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 $KL \gg KU$.

Also, like the spring, the two nodes of the gap element must rotate together and the matrix transforms exactly as the matrix for a spring element.

2.16 Multi-Point Constraints, MPCs

A description of *MPC*s is contained in the users manual. This subsection discusses the coordinate system dependencies.

*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.

Salinas 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.

2.16.1 Constraint Transforms

Constraints may be eliminated using the constraint transform method. This is described in detail in Cook, chapter 9 (ref 5). 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, Salinas 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 insure that the constrained dofs are independent and never specified more than once.

For constraint elimination we have a constraint matrix $C = C_c C_r$, where C_c is a square, nonsingular 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 gauss 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 salinas 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 just 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.)

2.17 Rigid Elements

Salinas supports standard *pseudo*elements for rigid bodies. These include,

- RRODs - a rigid truss like element, infinitely stiff in extension, but with no coupling to bending degrees of freedom.
- RBARS - a rigid beam, 6 degrees of freedom deleted

- RBE2 - a rigid solid. $6(n-1)$ degrees of freedom deleted, where n is the number of nodes
- RBE3 - an averaging type solid. This connects to many nodes, but removes only 6 dofs.

All of the rigid elements are stored and applied internally as MPC equations. The RBE2 is a special case of RBAR (actually just multiple instances). Note, that unlike MPC equations, these rigid elements do activate (or touch) degrees of freedom. In general, an MPC equation will not activate a degree of freedom. In the case of a rigid element however, it is necessary to activate the degrees of freedom before constraining them. Otherwise the rigid elements do not act like real elements.

Rigid elements are input into Salinas using exodus beam elements. A block entry is then provided in the input file indicating what type of rigid element is required. There is no stiffness or mass matrix entry for any type of rigid elements (only the MPC entries described above).

2.17.1 RROD

An RROD is a *pseudo*element which is infinitely stiff in the extension direction. The constraints for an RROD may be conveniently stated that the dot product of the translation and the beam axial direction for a RROD is zero. There is one constraint equation per RROD.

2.17.2 RBAR

An RBAR is a *pseudo*element which is infinitely stiff in all the directions. The constraints for an RBAR may be summarized as follows.

1. the rotations at either end of the RBAR are identical,
2. there is no extension of the bar, and
3. translations at one end of the bar are consistent with rotations.

It is apparent that 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.

2.17.3 RBE3

The RBE3 element behavior is taken from Nastran's element of the same name. Earlier implementations of the RBE3 differed significantly from the MSC/Nastran implementations (see appendix C). The revised element should act like a nastran RBE3 for most applications[14]. The element is used to apply distributed forces to many nodes while not stiffening the structure as an RBE2 or RBAR does. The RBE3 uses the concept of a slave node. The theory follows the MSC documentation included in the appendix.

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

$$L_{i,x} = x_i - x_q \tag{2.89}$$
$$L_{i,y} = y_i - y_q \tag{2.90}$$
$$L_{i,z} = z_i - z_q \tag{2.91}$$
$$L_i = \sqrt{L_{i,x}^2 + L_{i,y}^2 + L_{i,z}^2} \tag{2.92}$$

The characteristic length of the element is the average of these lengths,

$$L_c = \sum_{i=1}^{N_c} |L_i| / N_c, \tag{2.93}$$

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.

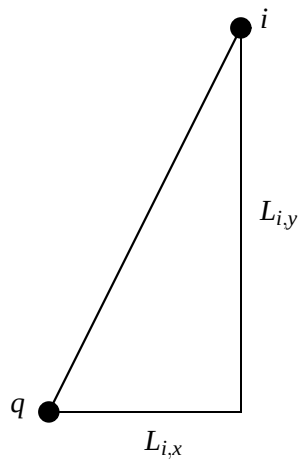
To insure that the element is invariant to a change of scale, the weighting functions `w1` through `w6` provided by the user are modified to produce a connected grid point's weighting matrix.

$$W = \begin{bmatrix} w_1 & & & & & \\ & w_2 & & & 0 & \\ & & w_3 & & & \\ & & & w_4 L_c^2 & & \\ & & 0 & & w_5 L_c^2 & \\ & & & & & w_6 L_c^2 \end{bmatrix} \tag{2.94}$$

That is, the rotational DOF coefficients are scaled by the square of the characteristic length.

[14]The Salinas element is not as flexible as the nastran element in all respects. In particular, there is no flexibility to apply node specific weighting. Weights may be applied by degree of freedom, but these weights are applied uniformly to all nodes in the pseudo element.

Figure 8. 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.

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 slave node as illustrated in Figure 8. The loads at the connection point, i, relate to the loads at the slave point.

$$P_q = S'_{iq} P_i \tag{2.95}$$

Where,

$$S_{iq} = \begin{bmatrix} 1 & 0 & 0 & 0 & L_{i,z} & -L_{i,y} \\ & 1 & 0 & -L_{i,z} & 0 & L_x \\ & & 1 & L_{i,y} & -L_{i,x} & 0 \\ & & & 1 & 0 & 0 \\ & 0 & & & 1 & 0 \\ & & & & & 1 \end{bmatrix} \tag{2.96}$$

Assembled Constraint. As shown in Appendix C (equation C.1), the loads on the set of all connection nodes may be computed from the load on the slave node.

$$P_i = G'_{qi} P_q \tag{2.97}$$

Where,

$$G_{qi} = A^{-1} \mathcal{S}' \mathcal{W} \tag{2.98}$$

here $\mathcal{S}$ is a concatenation of the individual S_{iq},

$$\mathcal{S} = \begin{bmatrix} S_{1,q} \\ S_{2,q} \\ \dots \\ S_{N_c,q} \end{bmatrix}, \tag{2.99}$$

Similarly,

$$\mathcal{W} = \begin{bmatrix} W_1 & & & \\ & W_2 & & \\ & & \dots & \\ & & & W_c \end{bmatrix} \tag{2.100}$$

and A is a 6 by 6 weightings matrix.

$$A = \mathcal{S}' W \mathcal{S} \tag{2.101}$$

We require that A be nonsingular, 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,

$$\mathcal{C} = \left[\begin{array}{cc} -I_{qq} & G_{qi} \end{array} \right] \tag{2.102}$$

and apply the constraint,

$$\mathcal{C} \left[\begin{array}{c} u_q \\ u_i \end{array} \right] = 0. \tag{2.103}$$

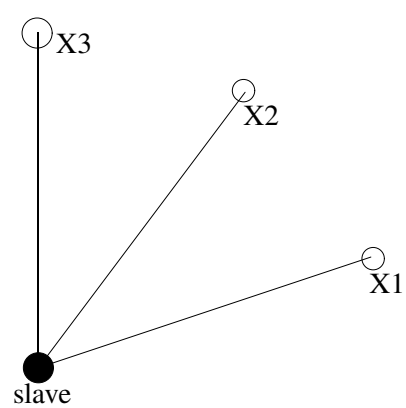
Each row of $\mathcal{C}$ contains the constraint coefficients for one of the six possible constraints in the RBE3.

2.17.4 RBE3 – old version

The RBE3-old elements behavior is taken from Nastran's element of the same name. Note however, that the precise mathematical framework of the Nastran RBE3 element is not specified in the open literature. This element should act like an RBE3 for most applications. The element is used to apply distributed forces to many nodes while not stiffening the structure as an RBE2 or RBAR would. The RBE3-old uses the concept of a slave node. Constraints are specified as follows.

1. The translation of the slave node is the sum of translations of all the other nodes in the element.
2. The rotation of the slave node is the weighted average of all the other nodes about it. This is determined by the nodal translations, not by their rotations.

While the first of these constraints is easy enough to apply using multi-point constraints, the second is a little more difficult. We seek a least squares type solution.



Let $\vec{D}_i = \vec{U}_i - \vec{U}_{slave}$,

$\vec{L}_i = \vec{X}_i - \vec{X}_{slave}$

The L represent a vector from the "origin" to the point i, while the D_i represent the differential displacement of the same points. Note that the origin is at the location of the slave node, and will not in general be at the centroid of the structure.

We will use least squares to compute the rotational vector of the slave node. This is equivalent to computing a rotational inertial term and requiring a similar net rotation for the centroid.

The displacement at the centroid should be given by,

$$\vec{D}_i = \vec{R} \times \vec{L}_i$$

or, in the least squares sense we seek to minimize E.

$$E = \sum_i (\vec{D}_i - \vec{R} \times \vec{L}_i) \cdot (\vec{D}_i - \vec{R} \times \vec{L}_i)$$

Take the derivative of E with respect to a component of R, r_k.

$$\frac{dE}{dr_k} = 0 = 2\sum_i (\hat{e}_k \times \vec{L}_i) \cdot (\vec{R} \times \vec{L}_i) - \vec{D}_i \cdot (\hat{e}_k \times \vec{L}_i)$$

Now, let $R = \sum_m r_m \hat{e}_m$. We substitute for R in the previous equation to obtain,

$$\sum_m \sum_i r_m (\hat{e}_k \times \vec{L}_i) \cdot (\hat{e}_m \times \vec{L}_i) - \vec{D}_i \cdot (\hat{e}_k \times \vec{L}_i) = 0$$

Now, if we write L_i as a column vector then the expression $(\hat{e}_k \times \vec{L}_i) \cdot (\hat{e}_m \times \vec{L}_i)$ can be written as $L_i^T L_i \cdot I - L_i L_i^T$. If the sum on i is performed for the first term, we may write,

$$\sum_m r_m A_{mk} - \sum_i \hat{e}_k \cdot (\vec{L}_i \times \vec{D}_i) = 0$$

where

$$A_{mk} = \left(\sum_i^n |L_i|^2 \right) \delta_{mk} - L_i^m L_i^k$$

This provides three equations (one for each k) in the 3 unknowns, r_m. Note that L_i^m represents the m component (1-3) of the vector L_i.

The solution is found by looping once through all i to fill in the A matrix, and simultaneously to fill out the coefficients for the three equations involving D_i. Once the loop has been completed, the coefficients of R are known, and the three components of r_m can be added for each of the three equations. Each equation has 3 components of R, $2n$ components of U_i and 2 components of U_{slave} for a total of $2n+5$ equations.

2.18 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.

Define a coordinate system at the node, with variables u. On the offset beam the coordinate system is $\tilde{u}$.

Now, u is related simply 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. Notice that the rotation is the same at both nodes.

Thus we can write,

$$\begin{pmatrix} \tilde{u} \\ \kappa \end{pmatrix} = [L] \begin{pmatrix} u \\ \kappa \end{pmatrix} \tag{2.104}$$

where L is a constant matrix which depends only on the geometry. We can use this transformation matrix to eliminate the degrees of freedom associated with $\tilde{u}$. The energy of the shell can be written,

$$E_{strain} = 0.5 \begin{Bmatrix} \tilde{u} \\ \kappa \end{Bmatrix}^T \left[\tilde{K}\right] \begin{Bmatrix} \tilde{u} \\ \kappa \end{Bmatrix} \tag{2.105}$$

But with this substitution,

$$E_{strain} = 0.5 \begin{Bmatrix} u \\ \kappa \end{Bmatrix}^T \left[L^T \tilde{K} L\right] \begin{Bmatrix} u \\ \kappa \end{Bmatrix} \tag{2.106}$$

If we let $K = L^T \tilde{K} L$, then,

$$E_{strain} = 0.5 \begin{Bmatrix} u \\ \kappa \end{Bmatrix}^T [K] \begin{Bmatrix} u \\ \kappa \end{Bmatrix} \tag{2.107}$$

Thus, $\tilde{u}$ has been eliminated, and the equations may be rather simply put in terms of the output variables.

2.19 Notes on Consistent Loads Calculations

Starting with equation 4.1-6 from *Concepts and Applications of Finite Element Analysis* by Cook *et al.*,

$$\{r_e\} = \int_{V_e} [B]^T [E] \{\varepsilon_0\} dV - \int_{V_e} [B]^T \{\sigma_0\} dV + \int_{V_e} [N]^T \{F\} dV + \int_{S_e} [N]^T \{\Phi\} dS \tag{2.108}$$

where each of these terms are defined in Subsection 4.1 of the above mentioned reference. The load vector, $\{r_e\}$, is composed of four parts in Eqn. 2.108. In this document, only the last part, which is the contribution of the surface tractions to the load vector, will be considered. Rewriting,

$$\{r_e\} = \int_{S_e} [N]^T \{\Phi\} dS \tag{2.109}$$

Here, the integral is calculated over the surface of the element on which the surface traction, $\{\Phi\}$, is applied. Therefore,

$$\{\Phi\} = [\Phi_x \Phi_y \Phi_z]^T \tag{2.110}$$

and $[N]$ is the shape function matrix of the element on which the surface tractions, $\{\Phi\}$, are applied. In Salinas, $\{\Phi\}$ can be applied within PATRAN by applying a spatial field to a specified side set. As a result, when calculating the load vector, this field must be accounted for. In Salinas however, this spatial field values will be available only at the nodes of the element. Using the nodal values of this surface traction, the value inside must be defined 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 Salinas, a surface traction may be applied only in one direction at a time. Therefore, $\{\Phi\}$ will be defined as

$$\{\Phi\} = \left\{ \begin{array}{c} n_x \\ n_y \\ n_z \end{array} \right\} \Phi(x,y,z) \tag{2.111}$$

2.19.1 Salinas Element Types

The following 3-D and 2-D elements have consistent loads implemented:

- Hex8
- Hex20
- Wedge6
- Tet4
- Tet10
- Tria3
- TriaShell
- Tria6 (four Tria3s)
- QuadT (two Tria3s)
- Quad8T (1 QuadT and 4 Tria3s)

2.19.2 Pressure Loading

Here, we will consider only pressure loads on 3-D elements, such that

$$\{\Phi\} = \left\{ \begin{array}{c} n_x \\ n_y \\ n_z \end{array} \right\} \Phi(x,y,z) \tag{2.112}$$

where $[n_x, n_y, n_z]^T$ is the normal to the element face. Hence, the consistent loads can be calculated as,

$$\{r_e\} = \int_{S_e} [N]^T \{\Phi\} dS = \int_{S_e} [N]^T \Phi(x,y,z)(\vec{a} \times \vec{b}) dS_e \tag{2.113}$$

Here,

$$\vec{a} = [\frac{\partial x}{\partial r}, \frac{\partial y}{\partial r}, \frac{\partial z}{\partial r}]^T \tag{2.114}$$

$$\vec{b} = [\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s}]^T \tag{2.115}$$

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. Notice, taking the cross-product of $\vec{a}$ and $\vec{b}$, the normal is obtained.

2.19.3 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. If the shape functions for the 3-D elements are used, it will reduce code and "fit" 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 will 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 will have a local coordinate system (r,s). The "mapping" will define how (r,s) are related to (η_1, η_2, η_3). This will also help defined how 2-D Gauss points are mapped to the 3-D face. These mappings are done for all the 3-D elements.

2.19.4 Shell Elements - consistent loads

All the 2-D elements (shell elements) are based on the Tria3. The consistent loads calculations for the Tria3 can be "copied" to the TriaShell. This way all the shell elements will 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

$$f^1{}_s = \frac{1}{20}(7q_{s1} + 3q_{s2})L_{21} \qquad f^2{}_s = \frac{1}{20}(3q_{s1} + 7q_{s2})L_{21} \tag{2.116}$$

$$f^1{}_n = \frac{1}{20}(7q_{n1} + 3q_{n2})L_{21} \qquad f^2{}_n = \frac{1}{20}(3q_{n1} + 7q_{n2})L_{21} \tag{2.117}$$

$$f^1{}_t = \frac{1}{20}(7q_{t1} + 3q_{t2})L_{21} \qquad f^2{}_t = \frac{1}{20}(3q_{t1} + 7q_{t2})L_{21} \tag{2.118}$$

$$m^1{}_s = m^2{}_s = 0 \tag{2.119}$$

$$m^1{}_n = -\frac{1}{60}(3q_{t1} + 2q_{t2})L^2{}_{21} \qquad m^2{}_n = \frac{1}{60}(2q_{t1} + 3q_{t2})L^2{}_{21} \tag{2.120}$$

$$m^1{}_t = -\frac{1}{40}(3q_{n1} + 2q_{n2})L^2{}_{21} \qquad m^2{}_t = \frac{1}{40}(2q_{n1} + 3q_{n2})L^2{}_{21} \tag{2.121}$$

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 Salinas, by the thickness of the triangle.

Now if the pressure load is on the face of the Tria3, the equations become,

$$f^1{}_x = f^1{}_y = m^1{}_z = f^2{}_x = f^2{}_y = m^2{}_z = f^3{}_x = f^3{}_y = m^3{}_z = 0 \tag{2.122}$$

$$f^1{}_z = (\frac{8}{45}p_1 + \frac{7}{90}p_2 + \frac{7}{90}p_3)A \tag{2.123}$$

$$f^2{}_z = (\frac{7}{90}p_1 + \frac{8}{45}p_2 + \frac{7}{90}p_3)A \tag{2.124}$$

$$f^3{}_z = (\frac{7}{90}p_1 + \frac{7}{90}p_2 + \frac{8}{45}p_3)A \tag{2.125}$$

$$m^1{}_x = \frac{A}{360}[7(y_{31} + y_{21})p_1 + (3y_{31} + 5y_{21})p_2 + (5y_{31} + 3y_{21})p_3] \tag{2.126}$$

$$m^1{}_y = \frac{A}{360}[7(x_{13} + x_{12})p_1 + (3x_{13} + 5x_{12})p_2 + (5x_{13} + 3x_{12})p_3] \tag{2.127}$$

$$m^2{}_x = \frac{A}{360}[(5y_{12} + 3y_{32})p_1 + 7(y_{12} + y_{32})p_2 + (3y_{12} + 5y_{32})p_3] \tag{2.128}$$

$$m^2{}_y = \frac{A}{360}[(5x_{21} + 3x_{23})p_1 + 7(x_{21} + x_{23})p_2 + (3x_{21} + 5x_{23})p_3] \tag{2.129}$$

$$m^3{}_x = \frac{A}{360}[(3y_{23} + 5y_{13})p_1 + (5y_{23} + 3y_{13})p_2 + 7(y_{23} + y_{13})p_3] \tag{2.130}$$

$$m^3{}_x = \frac{A}{360}[(3x_{32} + 5x_{31})p_1 + (5x_{32} + 3x_{31})p_2 + 7(x_{32} + x_{31})p_3] \tag{2.131}$$

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 Tria3s require a little extra overhead. For example, the Quad8T is composed of 1 QuadT and 4 Tria3s. However, since it is defined as a Quad8T, it will have 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 will be 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.

2.20 Coordinate Systems

Coordinate systems are provided for a number of 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 2.14).
5. specification of output coordinate systems (in history files only).

There are some applications for coordinate systems which we do NOT intend to support. These include,

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 noncartesian systems, the XZ plane is used for defining the origin of the θ direction only.

Each coordinate system carries with it a rotation matrix. It is important to clarify the meaning of that matrix. Specifically,

$$X' = RX$$

Where X' is the new system of coordinates, R is the rotation matrix and X is the basic coordinate system. For cartesian systems, the rotation matrix is static. Curvilinear systems will require computation of a new rotation matrix at each location in space.

The usual identity on rotation matrices applies, namely:

$$X = R^T X' \tag{2.132}$$

and

$$R^T R = RR^T = I$$

As an example, consider a cartesian system as shown in Figure 9.

The new system (marked by primes) is rotated θ from the old system with the new X' axis in the first quadrant of the old system. The rotation matrix is,

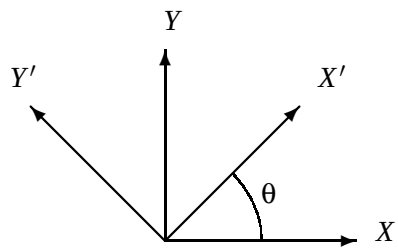


Figure 9. Original, and rotated coordinate frames

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2.21 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 Salinas are also outlined.

Consider a constraint equation,

$$C'u' = Q \tag{2.133}$$

where the primes indicate a generalized coordinate frame. The frame may be transformed to the basic coordinate system using equation 2.132, and

$$u' = Ru \tag{2.134}$$

We can now rewrite equation 2.133,

$$\begin{aligned} C'Ru &= Q \\ Cu &= Q \end{aligned} \tag{2.135}$$

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.

2.21.1 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.

$$u = \begin{bmatrix} u_r \\ u_c \end{bmatrix} \tag{2.136}$$

The whole constraint equation may be similarly partitioned.

$$\begin{bmatrix} C_r & C_c \end{bmatrix} \begin{bmatrix} u_r \\ u_c \end{bmatrix} = [Q] \tag{2.137}$$

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,

$$u_c = C_c^{-1} Q - C_c^{-1} C_r u_r \tag{2.138}$$

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.

$$\begin{bmatrix} u_r \\ u_c \end{bmatrix} = [T]\,[u_r] + \tilde{Q} \tag{2.139}$$

where

$$T = \begin{bmatrix} 1 \\ C_{rc} \end{bmatrix} \tag{2.140}$$

$$C_{rc} = -C_c^{-1} C_r \tag{2.141}$$

and

$$\tilde{Q} = \begin{bmatrix} 0 \\ C_c^{-1} Q \end{bmatrix} = \begin{bmatrix} 0 \\ \breve{Q} \end{bmatrix} \tag{2.142}$$

2.21.2 Transformation of Stiffness Matrix

We assume a similar partition of the stiffness matrix. The equations for statics are then,

$$\begin{bmatrix} K_{rr} & K_{rc} \\ K_{cr} & K_{cc} \end{bmatrix} \begin{bmatrix} u_r \\ u_c \end{bmatrix} = \begin{bmatrix} R_r \\ R_c \end{bmatrix} \tag{2.143}$$

or,

$$[K]\,[T]\,u_r + [K]\left[\tilde{Q}\right] = R \tag{2.144}$$

and

$$T^T K T u_r = T^T \left\{ R - K\tilde{Q} \right\} = \tilde{R} \tag{2.145}$$

We can define the reduced equations,

$$\tilde{K} = T^T K T = K_{rr} + K_{rc} C_{rc} + C_{rc}^T K_{cr} + C_{rc}^T K_{cc} C_{rc} \tag{2.146}$$

and,

$$\begin{aligned} \tilde{R} &= T^T R - T^T \begin{bmatrix} K_{rc}\breve{Q} \\ K_{cc}\breve{Q} \end{bmatrix} \\ &= R_r + C_{rc}^T R_c - K_{rc}\breve{Q} - C_{rc}^T K_{cc}\breve{Q} \end{aligned} \tag{2.147}$$

The solution in the retained system is

$$\tilde{K} u_r = \tilde{R} \tag{2.148}$$

The system may now be solved using the reduced equations, and the constrained degrees of freedom may be solved using equation 2.138. Much of this is detailed in Cook, but without the constrained right hand side.

For eigen analysis the mass matrix may be transformed exactly as the stiffness matrix in equation 2.146. 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 Salinas.

2.21.3 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 2.146 and 2.147 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 2.133).
2. Transform the constraint equation to the basic coordinate system (equation 2.134).
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 2.137 and 2.141.

5. Compute the corrections to the force vector from equation 2.147. 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 2.146.

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 2.138).

A few words about post processing could also prove useful. In the first implementation of Salinas, constraints were applied only in the basic coordinate system. The degree of freedom to eliminate was obvious from the exodus file, and it's 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,

$$u_{\text{spc}} = \text{constant} + \sum_{i}^{\text{retained dofs}} u_i C_i \tag{2.149}$$

2.21.4 Multi Point Constraints

The application to multipoint constraints is very 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 now be stored in sparse format. However, the steps that are applicable for single point constraints apply here as well. Subsection 2.16 deals more explicitly with MPCs.

2.21.5 Transformation of Power Spectral Densities

Note: The following is taken almost verbatim from Paez's book [37]. We identify how to transform output PDS.

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,

$$\tilde{H}_1(f) = \sum_{k=1}^{3} c_{ki} H_k(f) \tag{2.150}$$

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,

$$G_{ij}(f) = \alpha H_i^*(f) H_j(f) \tag{2.151}$$

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 2.150 to express $\tilde{G}$ in terms of the H_i. We may then use the spectral definition in equation 2.151 to provide the transformation of spectral densities.

$$\begin{aligned} \tilde{G}_{ij}(f) &= \alpha \left(\sum_{k=1}^{3} c_{ki} H_k^*(f) \right) \left(\sum_{m=1}^{3} c_{mj} H_m(f) \right) \\ &= \sum_{k=1}^{3} \sum_{m=1}^{3} c_{ki} c_{mj} G_{km} \end{aligned} \tag{2.152}$$

This can be expressed in matrix notation as $\tilde{G} = C^T G C$.

2.22 HexShells

Hexshells are provided to give the analyst an element with performance similar to a standard shell, but with the mesh topography of a brick. Thus, thin regions of the model can be meshed with hexshells, without concern for the bad aspect ratio of the elements, and with topography consistent with a solid mesh.

The element is documented extensively in the description by Carlos Felippa (see reference 38). The paragraphs in this document summarize the limitations of the shells and the possible usage.

Because hexshells have an inherent thickness direction, it is important to be able to identify that direction. There are (at least) four methods to accomplish this.

natural The *natural* ordering of the nodes in the element can determine the thickness direction. This is the method used by Carlos in developing 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 Usually the topology can be used to identify the thickness direction. The hexshell should be used in a sheet. If the hexshells are considered alone, only the free surfaces of the sheet are candidates for the thickness direction. Further, once the thickness direction is established for one element, it must propagate to the neighbors. (Note that this implies that we can't have a self intersecting sheet).

projection The thickness direction could be determined by the closest projection to a coordinate direction.

We will try to support all of 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 code development team. 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. Only one of the four keywords above can be entered. If no keyword is entered, then *topology* is assumed.

```
Block 9
   HexShell
   orientation sideset='1,2'
   material=9
end
```

or,

```
Block 10
   HexShell
   orientation topology
   material=9
end
```

2.23 Membrane

In this section we provide the theory behind the tangent stiffness matrix for the quad membrane element in Salinas. This element has stiffness in the in-plane directions, but has no stiffness out-of-plane. Also, it has no rotational degrees of freedom. We note that the formulation given here is identical to the membrane used in Abaqus.[39]

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. Then, we consider the weak formulation of the internal force term for the membrane in the deformed configuration[1]

$$\delta W_{int} = \int_{\Omega} \delta D : \sigma d\Omega \tag{2.153}$$

where W_{int} is the virtual work, Ω is the domain of the membrane, σ is the stress tensor, and $L = \frac{\partial u}{\partial x} = D + W$ is the deformation gradient. The rate-of-deformation D and spin tensors W are defined as

$$D = \frac{1}{2}\left[\left(\frac{\partial u}{\partial x}\right) + \left(\frac{\partial u}{\partial x}\right)^T\right] \tag{2.154}$$

$$W = \frac{1}{2}\left[\left(\frac{\partial u}{\partial x}\right) - \left(\frac{\partial u}{\partial x}\right)^T\right] \tag{2.155}$$

Note that we are using an updated Lagrangian formulation here, and thus the integral in equation 2.153 is over the current (deformed) configuration of the membrane.

We note that we can also write equation 2.153 as

$$\delta W_{int} = \int_\Omega \delta L : \sigma d\Omega \tag{2.156}$$

since W is a skew-symmetric tensor, and the tensor product of a skew-symmetric tensor with a symmetric tensor (i.e. σ) is zero.

Equation 2.156 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 2.153 in terms of the (l,m,n) coordinate system of the membrane, we can eliminate the out-of-plane terms and write as

$$\delta W_{int} = \int_\Omega \delta L_{lm} : \sigma_{lm} d\Omega \tag{2.157}$$

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 $2x2$, 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 2.157 are in terms of global coordinates. We can express the in-plane displacement in terms of the out-of-plane displacement as

$$u_l = u\dot{l} \tag{2.158}$$

$$u_m = u\dot{m} \tag{2.159}$$

$$u_n = u\dot{n} \tag{2.160}$$

$$\tag{2.161}$$

Then, the relationship between the derivatives can be computed

$$\frac{\partial u}{\partial x_l} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial x_l} = \frac{\partial u}{\partial x} e_l \tag{2.162}$$

where 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, e_m, we arrive at

$$\frac{\partial u_m}{\partial x_l} = e_m \frac{\partial u}{\partial x} e_l \tag{2.163}$$

Next, we consider the expression given for the tangent operator in[39]

$$\int_{\Omega} \delta D : C : dD + \sigma : \left(\delta L^T \cdot dL - 2\delta D \cdot dD\right) d\Omega \tag{2.164}$$

Since there is no stress in the out-of-plane direction, and nothing varies through the thickness, the thickness can be pulled out, and this can be written simply as an area integral

$$t \int_{A} \delta D : C : dD + \sigma : \left(\delta L^T \cdot dL - 2\delta D \cdot dD\right) dA \tag{2.165}$$

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 now it is restricted to two dimensions. The geometric stiffness term is more involved, and so we elaborate some more on that.

First, we consider the deformation gradient in the plane of the element

$$L_{lm} = e_l \frac{\partial u}{\partial x_m} \tag{2.166}$$

Then, we have

$$\delta L_{lm} = e_l \frac{\partial \delta u}{\partial x_m} \tag{2.167}$$

$$\delta L_{lm}^T = \left(\frac{\partial \delta u}{\partial x_m}\right)^T e_l^T \tag{2.168}$$

$$\tag{2.169}$$

We also note that

$$L^T L = \left(\frac{\partial u}{\partial x_m}\right)^T e_l^T e_m \frac{\partial u}{\partial x_l} = \left(\frac{\partial u}{\partial x_m}\right)^T \frac{\partial u}{\partial x_l} \tag{2.170}$$

since $e_l^T e_m = \delta_{lm}$.

The rate of deformation D is simply the symmtric part of L. Thus, we can write

$$D_{lm} = \frac{1}{2}\left(e_l \frac{\partial u}{\partial x_m} + e_m \frac{\partial u}{\partial x_l}\right) \tag{2.171}$$

With these relations, we can expand the expression for the geometric stiffness, as

$$t \int_{A} \sigma_{lm} \left(\frac{\partial \delta u}{\partial x_m}\right)^T \frac{\partial u}{\partial x_l} - \frac{1}{2}\left(e_l \frac{\partial \delta u}{\partial x_m} + e_m \frac{\partial u}{\partial x_l}\right)\left(e_l \frac{\partial \delta u}{\partial x_m} + e_m \frac{\partial u}{\partial x_l}\right) \tag{2.172}$$

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 2.165 can be written as

$$K_{mat} = \int_{V} B^T C B dV \tag{2.173}$$

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,

$$\tilde{B} = \int_V B dV \tag{2.174}$$

We note that both B and $\tilde{B}$ can be decomposed into their volumetric and deviatoric components, i.e.

$$\begin{aligned} \tilde{B} &= \tilde{B}_V + \tilde{B}_D \\ B &= B_V + B_D \end{aligned} \tag{2.175}$$

With these decompositions, we define

$$\hat{B} = \tilde{B}_V + \tilde{B}_D + sd(B_D - \tilde{B}_D) \tag{2.176}$$

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

$$K = \int_V \hat{B}^T C \hat{B} dV \tag{2.177}$$

This is the approach taken for integrating the material stiffness term in equation 2.165

3 Loadings

3.1 Matrices from Applied Forces

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. Coriolis matrix in rotating structures. This is a skew-symmetric term that becomes part of the damping matrix.

3.2 Theory of Modal Analysis of Rotating Structures

The finite element modal analysis of rotating structures has been studied by many authors. There are two different approaches to this problem, with each approach being limited to certain applications. In the first approach, a rotating coordinate system is constructed that rotates with the structure.[40,41,42] Then, relative deformations about that rotating coordinate system are sought. In the second approach, an Eulerian (ALE) formulation is used, in which the structure rotates through an Eulerian mesh, and then Lagrangian deformations are considered about the Eulerian configuration.[43,44]

The first approach is not appropriate for modal analysis when contact surfaces are present, since the boundary conditions in the contact patch would change with time. On the other hand, the second approach is applicable to modal analysis with contact, but requires the structure to have a radial symmetry. In either case, the formulation leads to a gyroscopic eigenvalue problem, which can then be solved using a quadratic eigenvalue solver.

In these notes, we derive the finite element eigenvalue formulation corresponding to three-dimensional finite elements.

We begin by considering the homogeneous equations of motion of a solid body in three dimensions

$$\rho \ddot{u} - \nabla \cdot \sigma = 0 \tag{3.1}$$

where $\ddot{u}$ is the particle acceleration, ρ is the material density, and σ is the stress tensor. We only consider the homogeneous (no forcing) equation here, since we are mainly interested in eigenvalue analysis. This equation holds relative to a fixed, inertial reference frame. The term *inertial reference frame* is typically used to describe a reference frame that is not accelerating.

We now consider a reference frame that has the same origin as the inertial one described above, but is rotating at some angular velocity $\Omega = (\omega_1, \omega_2, \omega_3)$. We wish to formulate the eigenvalue

problem in an Eulerian framework, in which the displacement, velocity, and acceleration are all written as *relative* quantities, i.e. relative to the rotating coordinate system. Once the equations are written in terms of these relative quantities, we will be able to consider the small deformation eigenvalue problem about this rotating state.

We first note that the position vector of a point on the structure can written in terms of both the stationary coordinate system, i.e. $r = (x, y, z)$, or the rotating (relative) coordinate system, i.e. $r_{rel} = (x_{rel}, y_{rel}, z_{rel})$. It is clear that $r = r_{rel}$, even though the individual components in these vectors are different.

The relationships between the velocities and accelerations in the two coordinate systems are a bit more complex. Standard textbooks on rigid body dynamics[45] give the following expressions for the velocity $\dot{u}$ and acceleration $\ddot{u}$ in terms of the relative velocity $\dot{u}_{rel}$ and relative acceleration $\ddot{u}_{rel}$

$$\dot{u} = \dot{u}_{rel} + \Omega \times r \tag{3.2}$$

and

$$\ddot{u} = \ddot{u}_{rel} + 2\Omega \times \dot{u}_{rel} + \frac{d\Omega}{dt} \times r + \Omega \times \Omega \times r \tag{3.3}$$

where $r = x + u$ and x are the position vector and coordinates of a point.

We can now rewrite the first term in equation 3.1 as

$$\rho \ddot{u}_{tt} = \rho \left[\ddot{u}_{rel} + 2\Omega \times \dot{u}_{rel} + \frac{d\Omega}{dt} \times r + \Omega \times \Omega \times r \right] \tag{3.4}$$

In our case, we are only interested in the case where the structure is rotating at a fixed angular velocity, and thus $\frac{d\Omega}{dt} = 0$.

Having the equations of motion in the rotating coordinate system, we now proceed to construct the weak formulation. This can be done by multiplying equation 3.1 by a test function v and integrating by parts

$$\rho \left[\int_V \ddot{u}_{rel} \cdot v dV + 2 \int_V (\Omega \times \dot{u}_{rel}) \cdot v dV + \int_V (\Omega \times \Omega \times r) \cdot v dV \right] + \int_V \sigma : \nabla v dV + \int_S \sigma_n v dS = 0 \tag{3.5}$$

We note that since $r = x + u$, the term involving x will simply become part of the load vector. Since we are interested in eigenvalue analysis only, we can drop this term. Also, we will subsequently drop the *rel* subscripts from the above equation, since all quantities are now in the relative (rotating) coordinate system. Thus, the weak formulation becomes

$$\rho \left[\int_V \ddot{u} \cdot v dV + 2 \int_V (\Omega \times \dot{u}) \cdot v dV + \int_V (\Omega \times \Omega \times u) \cdot v dV \right] + \int_V \sigma : \nabla v dV + \int_S \sigma_n v dS = 0 \tag{3.6}$$

For the purposes of eigenvalue analysis, we can also drop the boundary term $\int_S \sigma_n v dS$, since it will contribute to the load vector. Thus, we have

$$\left[\int_V \ddot{u} \cdot v dV + \int_V (\Omega \times \dot{u}) \cdot v dV + \int_V (\Omega \times \Omega \times u) \cdot v dV \right] + \int_V \sigma : \nabla v dV = 0 \tag{3.7}$$

The first and last terms in the above equations correspond to the mass and stiffness matrices, respectively. The second term is the skew-symmetric coriolis term, and the third term is the symmetric

centrifugal term. We note that the stiffness term includes both the initial (material) stiffness associated with the material properties, as well as the geometric stiffness associated with the stresses. This stress state comes from the solution of the steady-state rolling problem, which includes the additional stresses associated with the inertial forces.

It is easy to show that the centrifugal term is symmetric, whereas the coriolis term is skew-symmetric. For the centrifugal term, we note the following identity for the triple cross product

$$a \times (b \times c) = b(a \cdot c) - c(a \cdot b) \tag{3.8}$$

Using this for examining the coriolis term, we have

$$\int_V (\Omega \times \Omega \times u) \cdot v dV = \int_V (\Omega \cdot v)(\Omega \cdot u) - (u \cdot v)(\Omega \cdot \Omega) dV \tag{3.9}$$

By switching u and v in the above expression, the same result is obtained, since the dot product is commutative. Thus, this term is symmetric.

For the coriolis term, we first write out the cross product term in terms of its components

$$\Omega \times \dot{u} = (-\Omega_3 \dot{u}_2 + \Omega_2 \dot{u}_3, \Omega_3 \dot{u}_1 - \Omega_1 \dot{u}_3, -\Omega_2 \dot{u}_1 + \Omega_1 \dot{u}_2) \tag{3.10}$$

Then, we have

$$(\Omega \times \dot{u}) \cdot v = v_1(-\Omega_3 \dot{u}_2 + \Omega_2 \dot{u}_3) + v_2(\Omega_3 \dot{u}_1 - \Omega_1 \dot{u}_3) + v_3(-\Omega_2 \dot{u}_1 + \Omega_1 \dot{u}_2) \tag{3.11}$$

Similarly, we can show that

$$(\Omega \times v) \cdot \dot{u} = \dot{u}_1(-\Omega_3 v_2 + \Omega_2 v_3) + \dot{u}_2(\Omega_3 v_1 - \Omega_1 v_3) + \dot{u}_3(-\Omega_2 v_1 + \Omega_1 v_2) \tag{3.12}$$

Comparing terms, we see that equation 3.11 is precisely the negative of equation 3.12. Thus, the coriolis term

$$\int_V (\Omega \times \dot{u}) \cdot v dV \tag{3.13}$$

is skew-symmetric.

We can now construct the finite element discretization of this equation by adopting the usual expansions, $u = N_i u_i$, $\dot{u} = N_i \dot{u}_i$, and $\ddot{u} = N_i \ddot{u}_i$. We will generate the forms of the matrices corresponding to the interactions a single node (node i) with another single node (node j). These will be 3×3 matrices, which thencan be projected into the global matrices. First, we note the form of the expansion for displacement

$$u = N_i u_i \tag{3.14}$$

Since the displacement is a vector of dimension 3, each shape function can be represented as a dimension-3 vector of the form

$$N_i = (\phi_i, 0, 0) \tag{3.15}$$

where ϕ_i is the ith shape function. Although we write the shape function in the first entry of the 3-vector N_i, it is actually placed in the k entry, where $k = mod(i, 3)$. With this notation, the 3×3 coriolis submatrix corresponding to the interaction between nodes i and j can be evaluated by setting $u = N_i u_i$, and $v = N_j$. Then, the (i, j)submatrix is given by

$$\int_V (\Omega \times N_i) \cdot N_j dV \tag{3.16}$$

After doing some simplifications, we find that the 3×3 matrix corresponding tonodes i and j is given by

$$\int_V \phi_i \phi_j \begin{bmatrix} 0 & \Omega_3 & \Omega_2 \\ -\Omega_3 & 0 & \Omega_1 \\ -\Omega_2 & -\Omega_1 & 0 \end{bmatrix} dV \tag{3.17}$$

As observed earlier, this matrix is skew-symmetric.

Next, we derive the form of the 3×3 submatrix corresponding to the centrifugal term. Again, using the expansion $u = N_i u_i$ and setting $v = N_j$, we have the 3×3 matrix

$$\int_V (\Omega \times \Omega \times N_i) \cdot N_j dV = \tag{3.18}$$

$$\int_V (\Omega \cdot N_j)(\Omega \cdot N_i) - (N_i \cdot N_j)(\Omega \cdot \Omega) dV = \tag{3.19}$$

$$\int_V \phi_i \phi_j (\Omega_k \Omega_m - \delta_{km}) dV \tag{3.20}$$

where δ_{km} is the 3×3 identity matrix, $k = mod(i,3)$, and $m = mod(j,3)$. Switching k and m, we see that the matrix is the same. Hence, we conclude that the centrifugal term is symmetric.

4 Linear Algebra Issues

4.1 Solution Spaces

There are a number of different dimensions in Salinas. These will be summarized here with a focus on using the data within the matlab framework. Examples of how to convert data from one dimensionality to another will be given.

The subject of matrix dimensions is an important one. Salinas has a fairly simple set of dimensions compared to more complex systems like Nastran. However, it is critical that these be well understood if we wish to manipulate the data.

As an example, I consider an eigen analysis of a structure with 9938 nodes. This structure is made of shells and solids. There are no boundary conditions, but there are 9 mpcs applied. I look at only the serial file sizes.

To get the required maps and other m-files, we must select 'mfiles' in the output section. To get the eigenvector data, we must also write the exodus file with 'disp' selected in the output section.

For this model, we have the following important dimensions.

1. #nodes=9938
2. external set= #nodes * 6 dofs/node = 59628
3. `G-set` = # active dofs before boundary conditions = 42708
4. `A-set` = analysis set = # equations to be solved = 42699
5. reduced external set = #nodes * 3 = 29814

There are 3 dofs/node for solid elements, but shells and beams have 6. In aggregate, the total dofs is 42708 before boundary conditions and mpcs are applied. There are no BCs in the model, but there are 9 MPC equations, each of which eliminates 1 dof, so the Aset is reduced to 42699.

Unfortunately, the `eigen_disp*.m` files are written in the reduced external set since this is what the analysts typically want. The bad news is that these m-files are useless to us. The good news is that all the data is available in either `m-files` or in the `exodus` output.

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 + Kssr' - speye(size(Kssr)).*Kssr;
```

The full eigenvectors (in the external 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);
>>> tmp = (0:nnodes-1)*6;
>>> phi(tmp+1,:)=nvar01;
>>> phi(tmp+2,:)=nvar02;
>>> phi(tmp+3,:)=nvar03;
>>> phi(tmp+4,:)=nvar04;
>>> phi(tmp+5,:)=nvar05;
>>> phi(tmp+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. We clearly can't multiply phi by K for example - the dimensions don't match. To do this we need a map.

We have two maps in our directory. FetiMap_a.m is the map from the external set to the A set. Thus we can reduce phi to the A-set by combining it with Fetimap_a. If the G-set is desired instead of the A-set, replace FetiMap_a with FetiMap.

```
>>> p2=zeros(max(max(FetiMap_a)),nsteps);
>>> for j=1:nnodes*6
>>>  i=FetiMap_a(j);
>>>  if ( i > 0 )
>>>    p2(i,:)=phi(j,:);
>>>  end
>>> end
```

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

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

Now we can do all the neat things like p2'*K*p2.

To get back to the external 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 ( FetiMap_a(i)>0 )
>>>     XX(i)=x(FetiMap_a(i));
>>>   end
>>> end
```

Obviously, similar shortcuts can be made to make this more efficient. One that appears to work is shown here.

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

4.2 Matrix Dimensions: Revision

The previous section is pretty confusing, and worse than this, it does not correspond well with other documentation. Let me make another stab at it. The various *spaces* are listed in Table 5. A discussion of each follows.

Space	Description
Full-set	biggest possible set. 8 * number of nodes
Output-set	6 * number of nodes This is the space that is 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
M-set	degrees of freedom eliminated by MPC
A-set	Analysis set.

Table 5. Salinas solution spaces

Full-set This space is referenced by many of our solvers. We then provide a map from this space to the A-set using Feti-map. Every node has 8 degrees of freedom (3 translations, 3 rotations, acoustic and generalized).

Output-set This is identical to the *full-set* except that acoustic and generalized dofs have been eliminated. It is used for output to exodus files, and contains all the structural dofs of the model.

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

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

Currently the only map to the assembly set is found in the `NodeArray`.

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

M-set This is the list of degrees of freedom that are eliminated using multipoint constraints (or MPCs). This space is non-null *only* if we are using constraint elimination in serial.

A-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*.

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, output or assembly set. This confusion spreads throughout the documentation and the comments in the notes.

4.3 Rotational Degrees of Freedom

In addition to the three translational degrees of freedom common in solid elements, beams, shells and some other specialty elements use rotational degrees of freedom. These degrees of freedom permit direct application of moments and allow efficient computations of structural element response such as bending. Rotational degrees of freedom are also important for management of rigid bodies. There are two methods of managing rotational degrees of freedom in our applications. Full rotation tensors are used for large deformation nonlinear response, while infinitismal rotation angles are typically used for small strain, linear response such as eigen analyis.

4.3.1 Euler Angles

In standard texts on classical mechanics, the rotation of a rigid body is often described using a rotation tensor complete with Euler angles. However, there are a variety of definitions of these angles, and the order by which they are applied does matter. From the wikipedia:

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 lower case (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.

Whatever definition is used, 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 rotion. Rotation of a vector by this angle is a tensor product with this matrix. i.e. $v' = Rv$.

4.3.2 Infinitismal Rotational Angles

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, $sin(\theta) = \theta$, 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:

$$[\mathbf{R}] = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix}$$

This formulation is extremely convenient, because the coordinates are completely 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.

4.3.3 Quaternions

The Euler angles of the previous sections can properly define the rotations of a body. However, the three ordered matrix operations required are not very convenient from a computational point of view. The quaternion provides an alternate form of algebra which is equivalent to the full Euler rotations, and is much more elegant (and efficient) for this type of computation. Within Salinas, we use the full rotation tensor, while other sierra solid mechanics codes use quaternions. They are mathematically equivalent.

4.3.4 Salinas Implementations

Linear vs. Nonlinear Solutions

Very simply put, all linear solutions use the infinitesmal 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 quite adequate for use in some parts of non-linear 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, Salinas allows the user to specify that certain material blocks in a model are linear, even in a nonlinear analysis. This 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, $< \alpha, \beta, \gamma >$. From the rotation increment, we compute a rotation increment quaternion as follows.

1. $\theta = \sqrt{(\alpha^2 + \beta^2 + \gamma^2)}$
2. $q_1 = \cos(\theta/2)$
3. $c = \sin(\theta/2)/\theta$

4. $q_2 = c\alpha$
5. $q_3 = c\beta$
6. $q_4 = c\gamma$
7. The quaternion is normalized.

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.

4.3.5 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 infinitismal rotations are signficant, 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.

4.4 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 kind of 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. For example, an inconsistent tied constraint may be constructed with a node on face approach. Alternatively a mortar method can accomplish the same thing with a different set of constraints. The weights for these systems may differ, but all must allow the body to freely rotate. It is clear that each constraint equation must satisfy this orthogonality independently.

4.4.1 Beam Example

I take an an example a simple two node beam to which a single node is constrained as is illustrated in Figure 10. The beam is of unit length, in the X direction. Point 3 is located a distance ε from point 1.

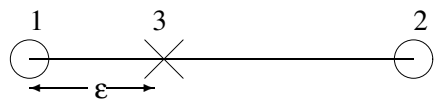


Figure 10. Node Constrained Directly to Beam.

The displacement vector is defined as,

$$U = [u_{1x}\, u_{1y}\, u_{2x}\, u_{2y}\, u_{3x}\, u_{3y}] \tag{4.1}$$

With a linear shape function, the typical constraints required to keep point 3 on the line are,

$$[(1-\varepsilon) \quad 0 \quad \varepsilon \quad 0 \quad -1 \quad 0] \tag{4.2}$$
$$[0 \quad (1-\varepsilon) \quad 0 \quad \varepsilon \quad 0 \quad -1] \tag{4.3}$$

The three orthogonal rigid body vectors are,[15]

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & -\theta & 0 & \theta & 0 & (2\varepsilon-1)\theta \end{bmatrix} \tag{4.4}$$

It can be seen that the constraints are orthogonal to the rigid body vectors.

4.4.2 Offset Example

With an offset, such as that shown in Figure 11, the rigid body vectors change. They become,

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & -\theta & 0 & \theta & -L\theta & (2\varepsilon-1)\theta \end{bmatrix} \tag{4.5}$$

What is important here is that the rotation rigid body mode gains an extra term. Rotation of this X axis beam about the Z axis now has a term in X. These rotational rigid body modes are no longer orthogonal to the original constraints, 4.3.

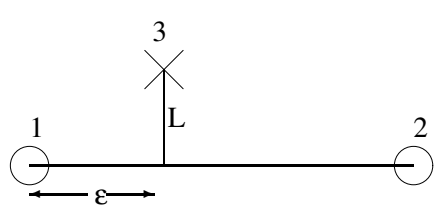


Figure 11. Node Constrained Offset to Beam.

[15] We are using infitismal rotations where $\sin(\theta) = \theta$.

4.4.3 Orthogonalization

A simple orthogonalization step can make the constraint weights once again orthogonal. We compute,

$$n = \vec{C} \cdot \vec{R}_i / ||\vec{R}_i||^2 \tag{4.6}$$
$$\vec{C} \leftarrow \vec{C} - n\vec{R}_i \tag{4.7}$$

where $\vec{C}$ represents the constraint equation, and $\vec{R}_i$ represents one of the *orthogonalized* rigid body modes. Without loss of generality, we can restrict $\vec{R}$ to the nodes in the constraint interaction. In general, this operation must be performed for all rigid body modes on each constraint.

4.5 Mass Properties

Mass properties are computed using the method of Baruch and Zemel.[46] 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 slightly different procedure. Both methods are discussed below.

4.5.1 Mass Property Calculations for Most Element Types

The mass properties are computed using rigid-body vectors. At a node, the translational rigid-body vectors are

$$\{R_x\} = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad \{R_y\} = \begin{Bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad \{R_z\} = \begin{Bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{4.8}$$

and the rotational rigid-body vectors are

$$\{R_{rx}\} = \begin{Bmatrix} 0 \\ -z \\ y \\ 1 \\ 0 \\ 0 \end{Bmatrix} \quad \{R_{ry}\} = \begin{Bmatrix} z \\ 0 \\ -x \\ 0 \\ 1 \\ 0 \end{Bmatrix} \quad \{R_{rz}\} = \begin{Bmatrix} -y \\ x \\ 0 \\ 0 \\ 0 \\ 1 \end{Bmatrix} \tag{4.9}$$

where x, y, and z are the location of the node in the global coordinate system. These vectors are actually assembled on an element level. As an example, for a three-node triangle element, $\{R_{rx}\}$ takes the form

$$\{R_{rx}\}^T = \left\{ \begin{array}{cccccccccccccccccc} 0 & -z_1 & y_1 & 1 & 0 & 0 & 0 & -z_2 & y_2 & 1 & 0 & 0 & 0 & -z_3 & y_3 & 1 & 0 & 0 \end{array} \right\}. \tag{4.10}$$

The total mass for an element can be computed as

$$M_{element} = \{R_x\}^T [M_e] \{R_x\} \tag{4.11}$$
$$= \{R_y\}^T [M_e] \{R_y\} \tag{4.12}$$
$$= \{R_z\}^T [M_e] \{R_z\} \tag{4.13}$$

where $[M_e]$ is the mass matrix for the element. The total mass for the model is computed by summing over all the elements

$$M_{total} = \sum_{i=1}^{Nel} \{R_x\}^T [M_e] \{R_x\}. \tag{4.14}$$

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

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

$$x_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nel} \{R_{rz}\}^T [M_e] \{R_y\}, \tag{4.15}$$

$$y_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nel} \{R_{rx}\}^T [M_e] \{R_z\}, \tag{4.16}$$

$$z_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nel} \{R_{ry}\}^T [M_e] \{R_x\}. \tag{4.17}$$

The components of the inertia tensor are computed as

$$I_{xx} = \sum_{i=1}^{Nel} \{R_{rx}\}^T [M_e] \{R_{rx}\}, \tag{4.18}$$

$$I_{yy} = \sum_{i=1}^{Nel} \{R_{ry}\}^T [M_e] \{R_{ry}\}, \tag{4.19}$$

$$I_{zz} = \sum_{i=1}^{Nel} \{R_{rz}\}^T [M_e] \{R_{rz}\}, \tag{4.20}$$

$$I_{xy} = \sum_{i=1}^{Nel} \{R_{rx}\}^T [M_e] \{R_{ry}\}, \tag{4.21}$$

$$I_{xz} = \sum_{i=1}^{Nel} \{R_{rx}\}^T [M_e] \{R_{rz}\}, \tag{4.22}$$

$$I_{yz} = \sum_{i=1}^{Nel} \{R_{ry}\}^T [M_e] \{R_{rz}\}. \tag{4.23}$$

This procedure for computing mass properties applies to hex8, hex20, wedge6, wedge15, tet4, tet10, beam2, Obeam, Nbeam, truss, tri3, tri6, tria, quad4, quad8, quadM, and conmass elements.

4.5.2 Mass Property Calculations for Acoustic Elements 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 **Salinas** .

The total mass is

$$M_{total} = \sum_{i=1}^{Nnode} M_i \tag{4.24}$$

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

$$x_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nnode} M_i x_i, \tag{4.25}$$

$$y_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nnode} M_i y_i, \tag{4.26}$$

$$z_{cg} = \frac{1}{M_{total}} \sum_{i=1}^{Nnode} M_i z_i \tag{4.27}$$

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

$$I_{xx} = \sum_{i=1}^{Nnode} M_i (y_i^2 + z_i^2), \tag{4.28}$$

$$I_{yy} = \sum_{i=1}^{Nnode} M_i (x_i^2 + z_i^2), \tag{4.29}$$

$$I_{zz} = \sum_{i=1}^{Nnode} M_i (x_i^2 + y_i^2), \tag{4.30}$$

$$I_{xy} = -\sum_{i=1}^{Nnode} M_i x_i y_i, \tag{4.31}$$

$$I_{xz} = -\sum_{i=1}^{Nnode} M_i x_i z_i, \tag{4.32}$$

$$I_{yz} = -\sum_{i=1}^{Nnode} M_i y_i z_i. \tag{4.33}$$

5 Constraints and Contact

5.1 Tied Friction

The work on tied surfaces with friction is under development. Details are maintained in our design documentation.

5.2 Mortar Methods

Mortar methods are an active area of research. Details are part of our design documentation (including theory and implementation).

References

[1] Belytschko, T., Liu, W. K., and Moran, B., *Nonlinear Finite Elements for Continua and Structures*, John Wiley & Sons, first edn., 2000.

[2] Koteras, J. R., Gullerud, A. S., Crane, N. K., and Hales, J. D., "Presto User's Guide," Tech. Rep. SAND2006-6093, Sandia National Laboratories, Albuquerque, NM, 2006.

[3] Krieg, R. D. and Key, S. W., "Transient Shell Response by Numerical Time Integration," *International Journal for Numerical Methods in Engineering*, **vol. 7**, 1973, pp. 273–286.

[4] Brown, K. and Voth, T., "ACME: Algorithms for Contact in a Multiphysics Environment, API Version 1.3," *SAND Report 2003-1470, Sandia National Laboratories*, 2003.

[5] Cook, R. D. and D. S. Malkaus, M. E. P., *Concepts and Applications of Finite Element Analysis*, John Wiley & Sons, third edn., 1989.

[6] Arben, P., Hetmaniuk, U. L., Lehoucq, R. B., and Tuminaro, R. S., "A Comparison of Eigensolvers for Large-scale 3D Modal Analysis using AMG-Preconditioned Iterative Methods," *International Journal for Numerical Methods in Engineering*, **vol. 1**, 2003, pp. 1–21.

[7] Lehoucq, R. B., Sorensen, D., and Yang, C., *ARPACK Users' Guide*, SIAM, Philadelphia, PA, USA, 1998.

[8] Reese, G., Field, R., and Segalman, D. J., "A Tutorial on Design Analysis Using von Mises Stress in Random Vibration Environments," *Shock and Vibration. Digest*, **vol. 32**, no. 6, 2000.

[9] Craig, R. R., *Structural Dynamics: An Introduction to Computer Methods*, John Wiley & Sons, 1981.

[10] Day, D. M. and Walsh, T., "Damped Structural Dynamics," Tech. Rep. SalinasDocuments/SandReport/qevp - Draft, Sandia National Laboratories, 2003.

[11] Tisseur, F. and Meerbergen, K., "The Quadratic Eigenvalue Problem," *SIAM Review*, **vol. 43**, no. 2, 2001, pp. 235–286.

[12] Saad, Y., *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press UK, 1992.

[13] Lang, G. F., "Demystifying Complex Modes," *Sound and Vibration Magazine*, **vol. 28.8**, 1989, pp. 36–40.

[14] Meyer, C. D., *Matrix Analysis and Applied Linear Algebra*, 2001.

[15] Mahan, G., *Applied Mathematics*, Kluwer Academic Publishers, 2002.

[16] Brinkmeier, M., Nackenhorst, U., Petersen, S., and von Estorff, O., "A Finite Element Approach for the Simulation of Tire Rolling Noise," *Journal of Sound and Vibration*, **vol. 309**, no. 1-2, 2008, pp. 20–39.

[17] Larsen, M., "A Posteriori and a Priori Error Analysis for Finite Element Approximations of Self-Adjoint Elliptic Eigenvalue Problems," *SIAM Journal of Numerical Analysis*, **vol. 38**, no. 2, 2000, pp. 608–625.

[18] Heuveline, V. and Rannacher, R., "A Posteriori Error Control for Finite Element Approximations of Elliptic Eigenvalue Problems," *Advances in Computational Mathematics*, **vol. 15**, 2001, pp. 107–138.

[19] Oden, J. T. and Prudhomme, S., "Error Estimation of Eigenfrequencies for Elasticity and Shell Problems," *Mathematical Models and Methods in Applied Sciences*, **vol. 13**, no. 3, 2003, pp. 323–344.

[20] Ainsworth, M. and Oden, J. T., *A Posteriori Error Estimation in Finite Element Analysis*, John Wiley & Sons, Inc., first edn., 2000.

[21] Bernardi, C. and Verfurth, R., "Adaptive finite element methods for elliptic equations with non-smoooth coefficients," *Numerische Mathematik*, **vol. 85**, 2000, pp. 579–608.

[22] Duran, R., Padra, C., and Rodriguez, R., "A Posteriori Error Estimates for the Finite Element Approximation of Eigenvalue Problems," *Mathematical Models and Methods in Applied Sciences*, **vol. 13**, no. 8, 2003, pp. 1219–1229.

[23] Prudhomme, S., "personal communication," March 2004.

[24] Key, S. W., "personal communication," December 2003.

[25] Taylor, R. L., Beresford, P. J., and Wilson, E. L., "A Non-conforming Element for Stress Analysis," *International Journal for Numerical Methods in Engineering*, **vol. 10**, 1976, pp. 1211–1219.

[26] Ibrahimbegovic, A. and Wilson, E. L., "A Modified Method of Incompatible Modes," *Communications in Applied Numerical Methods*, **vol. 7**, 1991, pp. 187–194.

[27] MacNeal, R. H., *Finite Elements: Their Design and Performance*, Marcel Dekker, 1994.

[28] Hughes, T. J. R., *The Finite Element Method–Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Inc, 1987.

[29] Hughes, T. J. R., *The Finite Element Method–Linear Static and Dynamic Finite Element Analysis*, chap. Appendix 3.1, Prentice-Hall, Inc, 1987, p. 170.

[30] Allman, D. J., "A Compatible Triangular Element Including Vertex Rotations for Plane Elasticity Problems," *Computers and Structures*, **vol. 19**, no. 1-2, 1996, pp. 1–8.

[31] Batoz, J.-L., Bathe, K.-J., and Ho, L.-W., "A Study of Three-Node Triangular Plate Bending Elements," *International Journal for Numerical Methods in Engineering*, **vol. 15**, 1980, pp. 1771–1812.

[32] Zienkiewicz, O. C. and Taylor, R. L., *The Finite Element Method*, vol. 2, chap. 1, McGraw-Hill Book Company Limited, fourth edn., 1991, pp. 23–26.

[33] Ertas, A., Krafcik, J. T., and Ekwaro-Osire, S., "Explicit Formulation of an Anisotropic Allman/DKT 3-Node Thin Triangular Flat Shell Elements," *Composite Material Technololgy*, **vol. 37**, 1991, pp. 249–255.

[34] Alvin, K., de la Fuente, H. M., Haugen, B., and Felippa, C. A., "Membrane triangles with corner drilling freedoms – I. The EFF element," *Finite Elements in Analysis and Design*, **vol. 12**, 1992, pp. 163–187.

[35] Felippa, C. A. and Militello, C., "Membrane triangles with corner drilling freedoms – II. The ANDES element," *Finite Elements in Analysis and Design*, **vol. 12**, 1992, pp. 189–201.

[36] Felippa, C. A. and Alexander, S., "Membrane triangles with corner drilling freedoms – III. Implementation and performance evaluation," *Finite Elements in Analysis and Design*, **vol. 12**, 1992, pp. 203–239.

[37] Wirsching, P. H., Paez, T. L., and Ortiz, K., *Random Vibrations, theory and practice*, John Wiley and Sons, Inc, 1995.

[38] Felippa, C. A., "The SS8 Solid-Shell Element: Formulation and a Mathematica Implementation," Tech. Rep. CU-CAS-02-03, Univ. Colo. at Boulder, 2002.

[39] Abaqus, I., *Abaqus Theory Manual, Version 6.5*, 2005.

[40] Carne, T., Lobitz, D., Nord, A., and Watson, R., "Finite Element Analysis and Modal Testing of a Rotating Wind Turbine," Tech. Rep. SAND82-0345, Sandia National Laboratories, 1982.

[41] Laurenson, R., "Modal Analysis of Rotating Flexible Structures," *AIAA Journal*, **vol. 14**, no. 10, 1976, pp. 1444–1450.

[42] ENDO, M., HATAMURA, K., SAKATA, M., and TANIGUCHI, ., "Flexible Vibration of a Thin Rotating Ring," *Journal of Sound and Vibration*, **vol. 92(2)**, 1984, pp. 261–272.

[43] Nackenhorst, U., "The ALE-formulation of bodies in rolling contact. Theoretical foundations and finite element approach," *Computer Methods in Applied Mechanics and Engineering*, **vol. 193**, 2004, pp. 4299–4322.

[44] Boman, R. and Ponthot, J., "Finite element simulation of lubricated contact in rolling using the arbitrary Lagrangian-Eulerian formulation," *Computer Methods in Applied Mechanics and Engineering*, **vol. 193**, 2004, pp. 4323–4353.

[45] Wilson, C., Sadler, J., and Michaels, W., *Kinematics and Dynamics of Machinery*, Harper and Row, 1983.

[46] Baruch, M. and Zemel, Y., "Mass Conservation in the Identification of Space Structures," *AIAA*, , no. 89-1239-CP, 1988, pp. 710–712.

[47] Auld, B. A., *Acoustic Fields and Waves in Solids, Second Edition*, vol. I, Robert E. Krieger Publishing Company, 1990.

[48] "MSC support," http://support.mscsoftware.com/.

[49] Harari, I., Grosh, K., Hughes, T., Malhotra, M., Pinsky, P., Stewart, J., and Thompson, L., "Recent Developments in Finite Element Methods for Structural Acoustics," *Archives of Computational Methods in Engineering*, **vol. 3**, 1996, pp. 132–311.

[50] Everstine, G. C., "Finite Element Formulations of Structural Acoustics Problems," *Computers and Structures*, **vol. 65**, no. 3, 1997, pp. 307–321.

[51] Aminpour, M., Ransom, J., and McCleary, S., "A coupled analysis method for structures with independently modelled finite element subdomains," *IJNME*, **vol. 38**, 1995, pp. 3695–3718.

[52] Dohrmann, C., Key, S., and Heinstein, M., "Methods for Connecting Dissimilar Three-Dimensional Finite Element Meshes," *IJNME*, **vol. 47**, 2000, pp. 1057–1080.

[53] Laursen, T. and Heinstein, M., "Consistent mesh tying methods for topologically distinct discretized surfaces in nonlinear solid mechanics," *IJNME*, **vol. 57**, 2003, pp. 1197–1242.

[54] Puso, M. A., "A 3D Mortar Method for Solid Mechanics," *Int. J. Numer. Meth. Engng.*, **vol. 59**, 2004, pp. 315–336.

[55] Mandel, J., "An Iterative Substructuring Method for Coupled Fluid-Solid Acoustic Problems,," *J. Comp. Phys.*, **vol. 177**, 2002, pp. 95–116.

[56] Flemisch, B., Kaltenbacher, M., and Wohlmuth, B., "Elasto-acoustic and acoustic-acoustic coupling on non-matching grids," *IJNME*, 2006, pp. (in press).

[57] Alonzo, A., Russo, A., Souto, C., Padra, C., and Rodriguez, R., "An Adaptive Finite Element Scheme to Solve Fluid-Structure Vibration Problems on Non-Matching Grids," *Computing and Visualization in Science*, **vol. 4**, 2001, pp. 67–78.

[58] Bermudez, A., Gamallo, P., and Rodriguez, R., "A Hexahedral Face Element for Elastoacoustic Vibration Problems," *JASA*, **vol. 109**, no. 1, 2001, pp. 422–425.

[59] Cook, R. D. and D. S. Malkaus, M. E. P., *Concepts and Applications of Finite Element Analysis: Chapter 9*, John Wiley & Sons, third edn., 1989.

[60] Chung, J. and Hulbert, G. M., "A Time Integration Algorithm for Structural Dynamics with Improved Numerical Dissipation - The Generalized Alpha Method," *JAM*, **vol. 60**, no. 2, 1993, pp. 371–375.

A Anisotropic Materials

Here we discuss how anisotropic elasticity is implemented in Salinas.[16] The approach is reasonably standard, but a documentation here is necessary to specify which of the many conventions of material parameter numbering is used in Salinas. Further, it is useful to present the theoretical development for those who may do maintenance on this part of the code.

A.1 Linear Anisotropic Elasticity

Linear elasticity asserts that the stress is a linear function of the strain:

$$\sigma_{ij} = C^4_{ijkl}\varepsilon_{kl} \tag{A.1}$$

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.

A.2 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.

$$s = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{Bmatrix} \tag{A.2}$$

and,

$$e = \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \\ 2\varepsilon_{12} \end{Bmatrix}.$$

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 seems to be popular in the materials science world, but it differs from the numbering used in NASTRAN and from the numbering in ABAQUS. Further, note that though the above are usually referred to as "stress vectors" and "strain vectors", they are not vectors in the sense that they map from one coordinate system to another as true vectors do. How that mapping is done is discussed in a later section.

[16] This is a transcription of Dan Segalman's framemaker document, "aniosConst.frm".

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.

$$e_n = Enij\varepsilon_{ij} \tag{A.3}$$

and

$$\varepsilon_{ij} = e_n F_{nij} \tag{A.4}$$

where

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad E_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad E_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$E_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad E_5 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad E_6 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{A.5}$$

and

$$F_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad F_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad F_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$F_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1/2 \\ 0 & 1/2 & 0 \end{bmatrix} \quad F_5 = \begin{bmatrix} 0 & 0 & 1/2 \\ 0 & 0 & 0 \\ 1/2 & 0 & 0 \end{bmatrix} \quad F_6 = \begin{bmatrix} 0 & 1/2 & 0 \\ 0 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} \tag{A.6}$$

We note that the stress mappings are also achieved with the above third order quantities:

$$s_n = F_{nij}\sigma_{ij} \tag{A.7}$$

and

$$\sigma_{ij} = snE_{nij} \tag{A.8}$$

From Equations A.3 and A.4 or Equations A.7 and A.8 we see that,

$$E_{mij}F_{nij} = \delta_{mn} \tag{A.9}$$

Substituting Equations A.4 and A.8 into Equation A.1 and simplifying with Equation A.9, we find

$$s_m = C_{mn}e_n \tag{A.10}$$

where

$$C_{mn} = F_{mij}C^4_{ijkl}F_{nkl} \tag{A.11}$$

Though above shows how to find the 6x6 matrix C_{ij} in terms of the fourth order tensor components C^4_{ijkl}, the material description is usually provided directly in terms of the components of C_{ij}.

A.3 Strain Energy and Orientation

We now address the situation where the matrix of material parameters of are provide in a Cartesian coordinate system different from the coordinate system (usually the global system) in which strains are calculated. Because stress and strain are tensors, they transfer from one coordinate system to another by:

$$\sigma_{ij} = R_{ai}\hat{\sigma}_{ab}R_{bj} \tag{A.12}$$

and

$$\varepsilon_{ij} = R_{ai}\hat{\varepsilon}_{ab}R_{bj} \tag{A.13}$$

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:

$$\hat{b}_a = R_{ai}b_i \tag{A.14}$$

where b_1, b_2, and b_3 are the basis vectors of the global frame.

From Equations A.7, A.8, and A.11, we find following

$$s_m = (F_{mij}E_{nab}R_{ai}R_{bj})\hat{s}_n. \tag{A.15}$$

From Equations A.3, A.4, and A.13, we find the more useful relationship

$$e_m = (E_{mij}F_{nab}R_{ai}R_{bj})\hat{e}_n. \tag{A.16}$$

The above two transformations are simplified:

$$s = T^T\hat{s} \tag{A.17}$$

and

$$e = T\hat{e} \tag{A.18}$$

where the 6x6 transformation matrix, T, is defined

$$T_{nk} = E_{nij}F_{kab}R_{ai}R_{bj} = tr\left(E_n^T RF_k R^T\right) \tag{A.19}$$

Noting that

$$s = \hat{C}\hat{e}, \tag{A.20}$$

and substituting Equations A.17 and A.18 into Equation A.20, we further find

$$s = T^T\hat{C}Te. \tag{A.21}$$

Comparing the above with Equation A.10, we finally find that

$$C = T^T\hat{C}T \tag{A.22}$$

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 $1/2$. After using Maple to simplify the product matrix,we find that T has a fairly simple form.

$$T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T22 \end{bmatrix} \tag{A.23}$$

where

$$T_{11} = \begin{bmatrix} R_{11}^2 & R_{12}^2 & R_{13}^2 \\ R_{21}^2 & R_{22}^2 & R_{23}^2 \\ R_{31}^2 & R_{32}^2 & R_{33}^2 \end{bmatrix}, \tag{A.24}$$

$$T_{12} = \begin{bmatrix} R_{13}R_{12} & R_{13}R_{11} & R_{13}R_{11} \\ R_{23}R_{22} & R_{23}R_{21} & R_{23}R_{21} \\ R_{33}R_{32} & R_{33}R_{31} & R_{33}R_{31} \end{bmatrix}, \tag{A.25}$$

$$T_{21} = \begin{bmatrix} 2R_{21}R_{31} & R_{22}R_{32} & R_{23}R_{33} \\ 2R_{11}R_{31} & R_{12}R_{32} & R_{13}R_{33} \\ 2R_{11}R_{21} & R_{12}R_{22} & R_{13}R_{23} \end{bmatrix}, \tag{A.26}$$

and

$$T_{22} = \begin{bmatrix} R_{23}R_{32}+R_{22}R_{33} & R_{23}R_{31}+R_{21}R_{33} & R_{22}R_{31}+R_{21}R32 \\ R_{13}R_{32}+R_{12}R_{33} & R_{13}R_{31}+R_{11}R_{33} & R_{12}R_{31}+R_{11}R32 \\ R_{13}R_{22}+R_{12}R_{23} & R_{13}R_{21}+R_{11}R_{23} & R_{12}R_{21}+R_{11}R22 \end{bmatrix}. \tag{A.27}$$

Note that T defined above is the transformation matrix N in of Equation 3.34 in Auld's *"Acoustic Waves in Solids, Volume I"* (reference 47), which is used in the same way.

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))) ;
```

B Integration of Isoparametric Solids

We show below how one achieves effective selective integration of isoparametric solids in a manner that satisfies the standard conditions (such as the patch test) and also accommodates anisotropic materials.[17]

We begin with the definition of the strain vector. For computational convenience defines the stress and strain vectors:

$$s = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{Bmatrix} \tag{B.1}$$

and,

$$\nu = \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \\ 2\varepsilon_{12} \end{Bmatrix}. \tag{B.2}$$

These are related through the matrix of elastic constants.

$$s = C\nu \tag{B.3}$$

We now take a look at virtual work, since it is from virtual work that the stiffness matrix is derived.

$$\delta W = \int_V s^T \delta\nu dV = \int_V \nu^T C \delta\nu dV \tag{B.4}$$

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,

$$\nu(x) = \sum_j B_j(x) u_j \tag{B.5}$$

where u_j is the jth nodal degree of freedom, the virtual work becomes

$$\delta W = u_j \delta u_k \int_V B_j(x)^T C B_k(x) dV \tag{B.6}$$

Since the element stiffness matrix is defined by

$$\delta W = u_j \delta K_{ij} \tag{B.7}$$

[17] This is a transcription of Dan Segalman's framemaker document, "IsoInt.frm".

we conclude that

$$K_{ij} = \int_V B_j(x)^T C B_k(x) dV \tag{B.8}$$

The next step is to decompose the strain-displacement vectors into deviatoric and dilatational components.

$$B_j(x) = B_j^D(x) + B_j^V(x) \tag{B.9}$$

where,

$$B_j^V(x) = d_j(x) \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{B.10}$$

and $3d_j(x)$ is the sum of the first three rows of $B_j(x)$. $B_j^D(x)$ is defined by Equation B.9. Substitution of Equation B.9 into Equation B.8 yields:

$$\begin{aligned} K_{ij} &= \int_V B_j^D(x)^T C B_k^D(x) dV + \int_V B_j^V(x)^T C B_k^V(x) dV + \cdots \\ &\quad + \int_V B_j^V(x)^T C B_k^D(x) dV + \int_V B_j^D(x)^T C B_k^V(x) dV \end{aligned} \tag{B.11}$$

For isotropic materials, the deviatoric and dilatational portions of the strain are orthogonal with respect to the matrix of material constants, so the last two integrals in the above equation are zero. It is sometimes common to integrate the contributions of each to the stiffness matrix using separate strategies. Such approaches can produce elements with slightly less susceptibility to parasitic shear. Such an approach does not work for elements of anisotropic material, so the following approach has been developed.

B.1 Uniform Strain-Displacement Matrices

At this point it is useful to define the element averaged strain displacement matrices.

$$\bar{B}_k = \frac{1}{V} \int_V B_k(x) dV \tag{B.12}$$

For hex elements, these are the strain-displacement matrices of the Flanagan and Belytschko, and are known as "uniform strain" elements. Elements formed by the above strain/displacement matrices are very "soft", having properties similar to elements formed by single point integration. Hex elements of this sort display extraneous zero-energy modes. In what follows, we consider linear combinations of this strain-displacement matrix formulation with the consistent formulation presented in Equation B.5.

The uniform strain matrices are also separable into dilatational and deviatoric parts.

$$\bar{B}_k = \bar{B}_k^V + \bar{B}_k^D \tag{B.13}$$

B.2 Mixed Integration

The approach presented here builds on one presented by Hughes.[28] 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.

$$\hat{B}_k(x) = \alpha \bar{B}_k^V + (1-\alpha) B_k^V(x) + \beta \bar{B}_k^D + (1-\beta) B_k^D(x) \tag{B.14}$$

(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 almost as good as that of the Flanagan and Belytschko element but without admitting hour-glass modes.

C MSC documentation of Nastran's RBE3 element

This documentation is provided by MSC from their web page.[48] It has been reformatted for math type formatting in TEX.

C.1 Abstract: Mathematical Specification of the Modern RBE3 Element

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

C.1.1 Extended Description

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. The reference is not available in machine-readable format. A fax copy may be requested from the MSC/NASTRAN Development Secretary, Jan.McLaughlin@MSCSOFTWARE.COM. It is primarily of historical interest now.

REFERENCE: Mathematical Specification for the RBE3 Element, MAG-4, 15 April 1975 (Also known as MAG-81).[18]

C.1.2 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.

The weighting functions `w1` through `w6` provided by the user are modified for reasons to be motivated later to produce a connected grid point's weighting matrix, a diagonal matrix shown here as a vector. Let $\tilde{w}_i = w_i L_c^2$. Then,

$$W = [w_1\, w_2\, w_3\, \tilde{w}_4\, \tilde{w}_5\, \tilde{w}_6]$$

That is, the rotation DOF coefficients are scaled by the characteristic length squared, but not the translation DOF coefficients.

[18] This TAN is known in MSC' s internal filing system as MAG-102.

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,

$$P_q = S'_{iq} P_i$$

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

$$Pq_{out} = \begin{bmatrix} S'_1 & S'_2 & \dots & S'_c \end{bmatrix} P_i = S'_{iq} P_i$$

Then

$$Pq_{out} = S'_{iq} WS_{iq} XPq_{in}$$

If $Pq_{out} = Pq_{in}$, then

$$X = [S'_{iq} WS_{iq}]^{-1} = A^{-1}$$

and,

$$P_i = WSXPq = G'_{qi} Pq$$

Where for convenience we define,

$$G'_{qi} = WSX \tag{C.1}$$

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

$$S_{iq} = \begin{bmatrix} T_1 S_1 \\ T_2 S_2 \\ \ldots \\ T_c S_c \end{bmatrix}$$

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,

$$R_{qi} = \begin{bmatrix} -I_{qq} & G_{qi} \end{bmatrix}$$

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.

C.1.3 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 of them to be discarded, as described in the next section.

C.1.4 EQUATION SELECTION

The total R_{qi} is generated above. It has 6 rows. Six or less 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.

C.1.5 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. That is, the weighting factors, whatever they are, 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 mainly 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.

C.1.6 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 all of 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.

C.1.7 UPWARD COMPATIBILITY

The RBE3 element prior to V70.7 had a more primitive theory that does not provide dimension independence. Its theory is identical to that above if a value of 1.0 is substituted for the characteristic length L_c. A system cell is provided to obtain this theory in V70.7. Its use allows computation of the same answers that were provided in earlier systems.

System Cell 310 Value	Action
0 (default)	Use new theory.
1	Use old theory.

The name of this system cell is OLDRBE3. For example, either entry below will cause the old theory to be used:

```
NASTRAN OLDRBE3=1 $  or
NASTRAN SYSTEM(310)=1 $
```

Changes to the RBE3 element for V70.7 are summarized in TAN 4155.

C.1.8 TOPICS FOR FUTURE WORK

The present order of operations requires that at first six equations be generated that allow meeting equilibrium conditions, then some equations (rows of R_{qi}) may be discarded, at the user's option. This makes modeling of planar elements, for example, awkward. There are now enough numerical tools such as Singular Value Decomposition (SVD) that would allow a different order of operations where only the equations required would be generated. There would then be no requirement to make the element stable for 6 DOFs, then, only for the number of equations actually used.

At present all dependent DOFs must either be totally on the reference grid point(default action), or on the connected grid points (UM data). There have been some unsolvable modeling problems due to singular R_{mg} matrices uncovered by clients having to do with interconnected RBE3 elements in a field of very regular geometry. If the geometry is perturbed slightly the equations are solvable, a disquieting feature when small changes in the model move it from a stable to an unstable state. It was shown in breadboard work that the problem is solvable if some of the reference point DOFs and some of the connected DOFs can both be in the Um data. The present rule that dependent DOFs must all be on the reference point or all on connected DOFs was done merely for programming convenience. The rule could readily be changed to allow mixed sets of dependent freedoms.

There are now enough mathematical tools to allow the dependent set for all MPC equations to be picked automatically, without the requirement for user input. There have been some unsuccessful attempts to do so in the past, but the lessons learned there, and the new mathematical tools available today, (particularly the SVD) offer promise for successful research in this area.

C.2 Abstract: RBE3 ELEMENT CHANGES IN VERSION 70.7

Solution#:	4155	Last Modified Date:	04/17/00 02:50:26 PM
Product Line:	MSC.Nastran	Product Name:	MSC.NASTRAN Basic (1003)
Product Version:	70.7	Product Feature:	ELEM
Article Type:	FAQ	Publish:	Y

C.2.1 Extended Description

1. The theory used for the RBE3 element has been modified so that the element is now independent of the units of measure. For example, a structure modeled in centimeters will now provide the same results when modeled in millimeters. This was not true for certain cases in systems prior to Version 70.7. A system cell provides the capability available prior to Version 70.7.

Ref. Tan 3280 for Version 70.6

2. 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 all of 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 theory for the element is rather involved. The derivation is given in TAN 4494. Some of the 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 GIDi and GIDq 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 instructive once in one's lifetime, if tedious, to work out a simple example by hand, for a simple geometry. We will instead just look at typical terms, to avoid some of the tedium.

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 insures that there will be no internal SPC forces in the element.

D Theory Notes for Acoustics and Structural Acoustics

Finite element analysis of acoustic and structural acoustic phenomena has become a common practice in both academia and industry. Excellent review articles can be found in,[49].[50]

In this section, we present the theory and equations behind the acoustic implementations in Salinas. This section is a work in progress, but still is a good start. We start with linear acoustics, on conforming and nonconforming structural acoustic meshes. Then, we describe the nonlinear acoustic work in Salinas.

D.1 Conforming and Nonconforming Linear Structural Acoustics

Having the same mesh density in the acoustic fluid and solid may be very inefficient, since the two domains typically require significantly different mesh densities to achieve a given level of discretization accuracy. Perhaps more importantly, 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 very difficult, if not impossible, even given the most sophisticated mesh generation software. Excellent 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 very different mesh density requirements. Joining them into a single, monolithic mesh is usually impractical.

Although methods for joining dissimilar meshes are well-known in structural mechanics,[51,52,53,54] very few papers exist in the area of dissimilar structural acoustic meshes. Mandel[55] 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.[56] 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[57] 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[58] 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, Raviert-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.

Here we present a new technique for acoustic and structural acoustic analysis in the case of nonconforming fluid/solid interface meshes. We first construct a simple method for coupling mis-

matched 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 on the structural side of the interface, and that extra ("ghost") acoustic degrees of freedom are introduced on the structural side of the wet interface. With this arrangement, the structural acoustic coupling resembles a conforming method on the structural side of the wet interface, and then the fluid degrees of freedom on both sides are coupled with the same approach that was used for the nonconforming fluid-fluid meshes. The coupling operators ensure a weak continuity of particle velocity and stress between the structural degrees of freedom and the ghost acoustic degrees of freedom, and then the linear constraints ensure continuity of acoustic pressure between the two sets of acoustic degrees of freedom.

Although we do not consider more sophisticated methods for nonconforming acoustic/acoustic meshes, such as mortar methods, our approach allows such methods to be readily applied to nonconforming structural acoustic meshes, since the wet interface coupling involves only acoustic degrees of freedom. Also, in the case that the fluid/solid meshes are conforming, our approach reduces to standard methods for conformal structural acoustic coupling.

D.2 The Governing 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 will begin with the case when all meshes are fully conforming, and then we will extend this to the nonconforming case.

D.3 Conforming Structural Acoustics

We will begin by constructing a weak formulation of the linear acoustic wave equation for conforming meshes. Subsequently, we will consider conforming structural acoustics.

The linear acoustic wave equation is given by

$$\frac{1}{c^2}\frac{\partial^2\psi}{\partial t^2} - \Delta\psi = 0, \tag{D.1}$$

where ψ is the velocity potential ($\psi = \nabla\dot{u}$, where $\dot{u}$ is the particle velocity), and c is the speed of sound. Note that this implies we neglect volume (body) forces on the fluid.

A weak formulation of equation D.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. $\psi(x,0) = \dot{\psi}(x,0) = 0$, 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 mapping $\psi: \quad [0,T] \rightarrow V_f(\Omega_f)$ such that

$$\frac{1}{c^2}\int_\Omega \ddot{\psi}\phi dx + \int_\Omega \nabla\psi\cdot\nabla\phi dx = -\int_{\partial\Omega_n} \rho_f \dot{u}_n \phi ds$$

$\forall\phi \in V_f(\Omega_f)$, where $\dot{u}_n$ is the prescribed velocity on the Neumann portion of the fluid boundary.

Inserting a finite element discretization $\phi(x) = \sum_{i=1}^{N}\phi_i N_i(x)$ into equation D.2 results in the system of equations

$$M\ddot{\psi} + K\psi = f_a, \tag{D.2}$$

where N is the vector of shape functions, $M = \int_{\Omega_f}\frac{1}{c^2}NN^T dx$ is the mass matrix, $K = \int_{\Omega_f}\nabla N\cdot\nabla N^T dx$ is the stiffness matrix, and $f_a = \int_{\partial\Omega_n}\rho_f\dot{u}_n N^T dx$ 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

$$\begin{aligned}
&\rho_s u_{tt} - \nabla\cdot\sigma = f,\\
&\frac{1}{c^2}\frac{\partial^2\psi}{\partial t^2} - \Delta\psi = 0.
\end{aligned} \tag{D.3}$$

Here u corresponds to the 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

$$\begin{aligned}
\frac{\partial\psi}{\partial n} &= -\rho_f\dot{u}_n,\\
\sigma_n &= -\dot{\psi}\hat{n},
\end{aligned} \tag{D.4}$$

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.

The weak formulation of the coupled problem is constructed by multiplying the two partial differential equations in equation D.3 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 $(u,\psi): \quad [0,T] \rightarrow V_s(\Omega_s) \times V_f(\Omega_f)$ such that

$$\begin{aligned}\int_{\Omega_s} \rho_s \ddot{u} w dx + \int_{\Omega_s} \sigma : \nabla^s w dx - \int_{\partial\Omega_{wet}} \sigma_n w ds &= \int_{\Omega_s} f w dx + \int_{\partial\Omega_n} \sigma_n w ds, \\ \frac{1}{c^2}\int_{\Omega_f} \ddot{\psi}\phi dx + \int_{\Omega_f} \nabla\psi \cdot \nabla\phi dx + \int_{\partial\Omega_{wet}} \frac{\partial\psi}{\partial n}\phi ds &= \\ &= \int_{\partial\Omega_n} \frac{\partial\psi}{\partial n}\phi ds\end{aligned} \tag{D.5}$$

$\forall w \in V_s(\Omega_s)$ 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 D.4, and we define $\sigma_n = g$ on the solid portion of $\partial\Omega_n$, and $\frac{\partial\psi}{\partial n} = -\rho_f \dot{u}_n$ on the fluid portion of $\partial\Omega_n$. This leads to the following weak formulation. Find the mapping $(u,\psi) \quad [0,T] \rightarrow V_s(\Omega_s) \times V_f(\Omega_f)$ such that

$$\begin{aligned}\int_{\Omega_s} \rho_s \ddot{u} w dx + \int_{\Omega_s} \sigma : \nabla^s w dx + \int_{\partial\Omega_{wet}} \dot{\psi}\hat{n} w ds &= \int_{\Omega_s} f w dx + \int_{\partial\Omega_n} g w ds, \\ \frac{1}{c^2}\int_{\Omega_f} \ddot{\psi}\phi dx + \int_{\Omega_f} \nabla\psi \cdot \nabla\phi dx - \rho_f \int_{\partial\Omega_{wet}} \dot{u}_n \phi ds &= \\ -\rho_f \int_{\partial\Omega_n} \dot{u}_n \phi ds\end{aligned} \tag{D.6}$$

$\forall w \in V_s(\Omega_s)$ and $\forall \psi \in V_f(\Omega_f)$.

Assuming a linear constitutive model for the solid, and inserting the spatial discretizations $u = \sum u_i N_i$ and $\phi = \sum \phi_i N_i$ into equation D.6 yields the following semidiscrete system of linear ordinary differential equations in time

$$\begin{bmatrix} M_s & 0 \\ 0 & M_f \end{bmatrix} \begin{bmatrix} \ddot{u} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} C_s & L \\ -\rho_f L^T & C_f \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} K_s & 0 \\ 0 & K_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} = \begin{bmatrix} f_s \\ f_f \end{bmatrix}, \tag{D.7}$$

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, as well as 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.

D.4 Nonconforming Structural Acoustics

In the case of nonconforming fluid/solid discretizations, equations D.5 and D.6 contain some extra technicalities. In this section we will first describe a simple procedure for coupling two acoustic

domains which share a common boundary, but with nonconforming discretizations. This method will then serve as a stepping stone to the case of nonconforming structural acoustics.

In order 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 package[4] has been developed as a tool 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-to-face interactions between the opposing surfaces, based on user-defined normal and tangential 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 user-specified search tolerance. The ACME package can compute contact conditions between most of the standard three-dimensional finite elements, including hexahedral, tetrahedral, and prismatic elements. Once these interactions are defined, one can devise enforcement algorithms to enforce continuity of the appropriate field variables. For the purposes of our work, we use ACME only to determine the node-to-face interactions on the wet interface. Once these are known, we derive our own enforcement algorithms, as explained below.

We consider the situation shown in Figure D.1. Here there are 2 interacting acoustic domains, and two contact surfaces. We adopt a master-slave approach, where one of the two interacting surfaces is designated as a master, and the other as the slave. We denote surface 1 as master, and surface 2 as slave. For a transient acoustic simulation involving the two meshes shown in Figure D.1, we would have to solve the system of equations given in D.2, 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.

In Figure D.2, node x from surface 1 is impinging on element face y of surface 2. If ACME 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

$$\psi^a = \sum_{i=1}^{4} c_i \psi_i^b, \tag{D.8}$$

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. 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 simply an average. We use this approach, commonly referred to as inconsistent tied contact, for all of 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. These constraint equations can be written mathematically as follows

$$C\Phi = 0, \tag{D.9}$$

where C is a matrix that contains all of the constraint coefficients from all of the node-face interactions, and vector Φ contains nodal velocity potentials from all of the nodes involved in the constraint

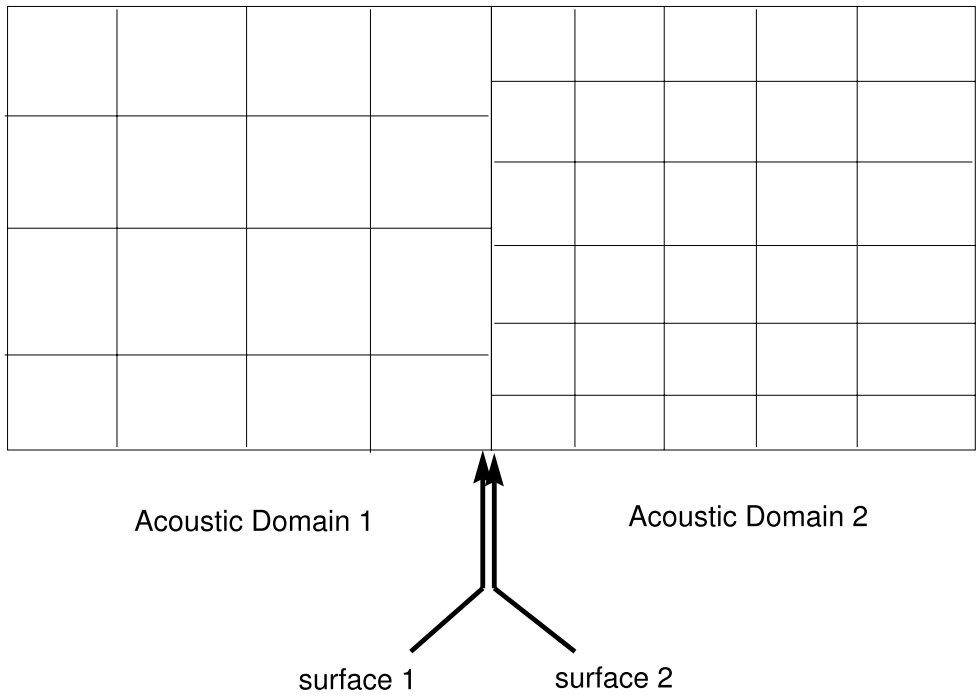


Figure D.1. Two interacting acoustic domains, with nonconforming meshes at the common interface. In this case surface 1 is defined to be the master surface, and surface 2 is the slave.

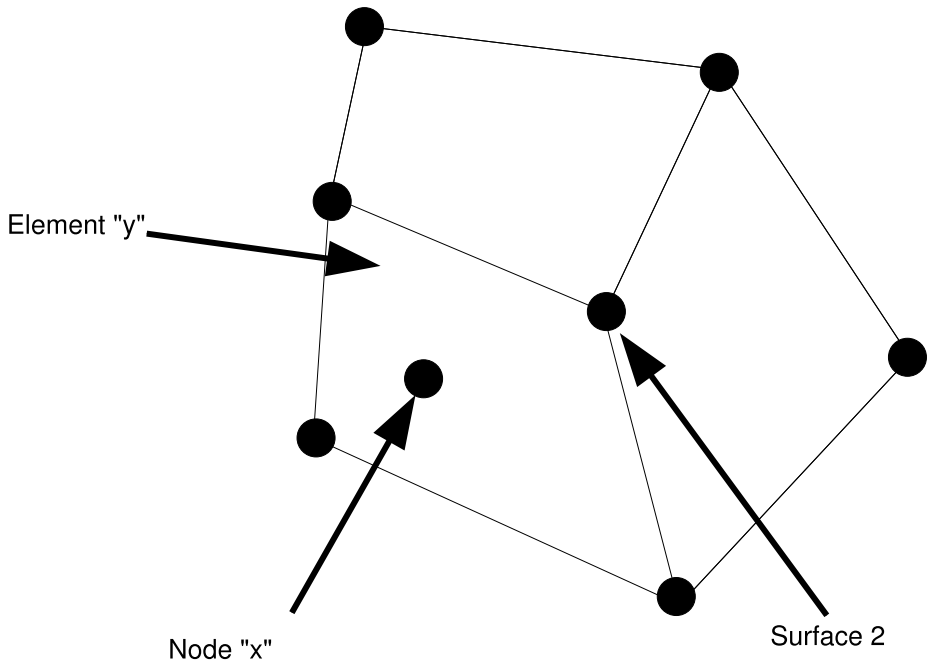


Figure D.2. A node-face interaction on the structural acoustic interface.

equations. Φ can be partitioned into master and slave components

$$\Phi = \begin{bmatrix} \Phi_m \\ \Phi_s \end{bmatrix}. \tag{D.10}$$

With this partition, the matrix C can also be split into master and slave components, and equation D.9 can be written as

$$[C_m \ , C_s] \begin{bmatrix} \Phi_m \\ \Phi_s \end{bmatrix} = 0. \tag{D.11}$$

Also, if we define $C_{ms} = -C_s^{-1} C_m$, we can condense the slave degrees of freedom from the stiffness matrix in equation D.2.

$$\tilde{K} = K_{mm} + K_{ms} C_{ms} + C_{ms}^T K_{sm} + C_{ms}^T K_{ss} C_{ms} \tag{D.12}$$

Similar condensation expressions hold for the mass and damping matrices.[59] After condensing out the slave acoustic degrees of freedom in equation D.2, we obtain a modified system of equations

$$\tilde{M}\ddot{\psi} + \tilde{K}\psi = \tilde{f}_a, \tag{D.13}$$

where the tilde superscripts indicate that the slave constraints have been condensed out. Note that the vector ψ now only contains the interior degrees of freedom (corresponding to nodes that are not on the interacting surfaces), and the master degrees of freedom on the contact surface, since the slave degrees of freedom have been eliminated. Equations D.13 can also be solved in the frequency domain, as follows

$$\left[s^2 \tilde{M} + \tilde{K}\right] \psi = \tilde{f}_a, \tag{D.14}$$

where s is the frequency parameter that comes from the Laplace transform.

In the case of structural acoustics, the algorithm just described for the nonconforming fluid/fluid meshes can be used as a stepping stone to the nonconforming solid/fluid meshes. In this approach, acoustic degrees of freedom are added to the nodes on the adjacent structural side of the wet interface. We subsequently refer to these as the ghost acoustic degrees of freedom. Subsquently, the acoustic and structural meshes are matching on the structural side of the wet interface, and the nodes on that side have four degrees of freedom instead of three. Next, the two surface integrals in equation D.6, i.e. $\int_{\partial\Omega_{wet}} \ddot{\psi} \hat{n} w ds$ and $\rho_f \int_{\partial\Omega_{wet}} \dot{u}_n \phi ds$, are both evaluated on the structural side of the wet interface. Finally, the mismatched acoustic meshes (the "true" acoustic surface nodes and their ghost counterparts) are tied together using the same set of linear constraint equations that was developed for the nonconforming acoustic/acoustic case.

In addition to equations D.7, we have a set of linear constraint equations that couple acoustic variables across the wet interface. As in the fluid/fluid case, these constraint equations represent the relations between the master and slave acoustic degrees of freedom, and they take the form

$$C\Phi = 0. \tag{D.15}$$

Upon condensing these constraints out of the system of equations, D.7, we obtain a modified system of equations

$$\begin{bmatrix} \tilde{M}_s & 0 \\ 0 & \tilde{M}_f \end{bmatrix} \begin{bmatrix} \ddot{u} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \tilde{C}_s & \tilde{L} \\ -\rho_f \tilde{L}^T & \tilde{C}_f \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \tilde{K}_s & 0 \\ 0 & \tilde{K}_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} = \begin{bmatrix} \tilde{f}_s \\ \tilde{f}_f \end{bmatrix}, \tag{D.16}$$

where again the tilde superscripts represent the matrices with constraints condensed out. Note that, in this case, even the structural matrices (and coupling matrices) must be modified during the constraint removal process, even though the constraints involve only acoustic degrees of freedom. This is because of the coupling matrices L and L^T, which couple the acoustic and structural degrees of freedom on the structural side of the wet interface. The fact that these other matrices are also modified is an essential part of the overall fluid/solid coupling scheme. To solve this system of equations, we use the generalized alpha time integration method,[60] 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. This can be seen simply by transforming equation D.16 to the frequency domain.

$$s^2 \begin{bmatrix} \tilde{M}_s & 0 \\ 0 & \tilde{M}_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} + s \begin{bmatrix} \tilde{C}_s & \tilde{L} \\ -\rho_f \tilde{L}^T & \tilde{C}_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} + \begin{bmatrix} \tilde{K}_s & 0 \\ 0 & \tilde{K}_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} = \begin{bmatrix} \tilde{f}_s \\ \tilde{f}_f \end{bmatrix}, \tag{D.17}$$

where s is the frequency parameter that comes from the Laplace transform. The same constraint equations that were used for the transient problem would also apply to the eigenvalue problem. Equations D.17 constitute a quadratic eigenvalue problem, which could be solved for the coupled modes. Note that the forcing terms would need to be set to zero in that case. Alternatively, if the frequency response (Helmholtz) problem was of interest, it could be obtained simply by setting $s = i\omega$ in the above equations, where ω is the frequency of interest. This would result in following complex-valued system of equations

$$-\omega^2 \begin{bmatrix} \tilde{M}_s & 0 \\ 0 & \tilde{M}_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} + i\omega \begin{bmatrix} \tilde{C}_s & \tilde{L} \\ -\rho_f \tilde{L}^T & \tilde{C}_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} + \begin{bmatrix} \tilde{K}_s & 0 \\ 0 & \tilde{K}_f \end{bmatrix} \begin{bmatrix} u \\ \psi \end{bmatrix} = \begin{bmatrix} \tilde{f}_s \\ \tilde{f}_f \end{bmatrix}. \tag{D.18}$$

Our method can be summarized by the diagram in Figure D.3. As shown the structural nodes on the wet interface are augmented with acoustic degrees of freedom. Consequently, these nodes each have four degrees of freedom. These "ghosted" acoustic degrees of freedom are then constrained by the acoustic degrees of freedom on the adjacent side of the wet interface. The structural acoustic coupling operators, which come from the weak formulation, are both evaluated on the structural side of the wet interface.

D.5 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

$$\psi^{tot} = \psi^{in} + \psi^{sc} \tag{D.19}$$

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

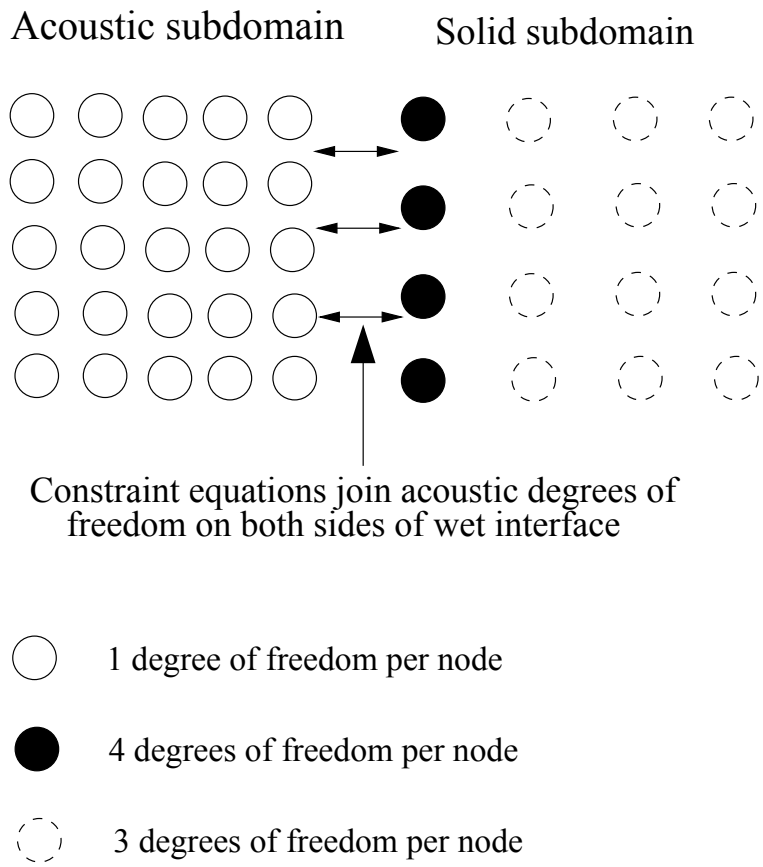


Figure D.3. Illustration of our method for structural acoustic meshes with nonconforming interfaces. Ghost acoustic degrees of freedom are added to the structural side of the wet interface, and then connected to the adjacent acoustic surface with constraint equations.

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.

The incident potential satisfies the wave equation, and for a plane wave takes the form

$$\psi^{in} = Ae^{i[\omega t - k\cdot\mathbf{x}]} \tag{D.20}$$

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 simply 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 D.20 drops out, and we are left with

$$\psi^{in} = Ae^{ik\cdot\mathbf{x}} \tag{D.21}$$

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 in terms of equations D.18. The left hand side of these equations will remain unchanged, but the right hand side will need special attention to impose the plane wave conditions on both the fluid and the solid.

First, we recall that the portion of the acoustic load f_a that comes from Neumann boundary conditions can be written in the form (see equations D.5)

$$f_a = -\rho_f \int_{\partial\Omega_n} \dot{u}_n \phi ds = \int_{\partial\Omega_n} \frac{\partial \psi^{in}}{\partial n} \phi ds \tag{D.22}$$

where $\rho_f \dot{u}_n = \frac{\partial \psi^{in}}{\partial n}$. Given equation D.21, we define $n = (n_x, n_y, n_z)$ to be the surface normal of the solid body. We also let $k \cdot x = kdotx$, $k = \frac{\omega}{c}(dir_x, dir_y, dir_z)$, where (dir_x, dir_y, dir_z) define the direction cosines of the direction of propagation of the indicent plane wave, this can be computed as

$$\frac{\partial \psi^{in}}{\partial n} = \nabla \psi^{in} \cdot n = i\frac{\omega}{c}[n_x dir_x + n_y dir_y + n_z dir_z] Ae^{ikdotx} \tag{D.23}$$

Inserting this expression into equation D.22, 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 (see equations D.5)

$$f_s = \int_{\partial\Omega_n} \sigma_n w ds \tag{D.24}$$

where $\sigma_n = n\psi^{in} = in\omega\psi^{in} = in\omega Ae^{i(kdotx)}$. Inserting this expression into equation D.24, and integrating, we obtain the loading on the solid body due to scattering. Inserting the expressions for f_a and f_s from equations D.22 and D.24 into equations D.18, we can solve for the responses of the acoustic fluid and solid body to incident plane waves. Note that the solution to equations D.18 will

give the scattered acosutic potential. In order to compute the total acoustic potential, we would need to add the incident and scattered potentials together, as in equation D.19.

Finally, we examine the complex-valued loads presented in equations D.22 and D.24. 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, Salinas 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 Salinas.

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 coefficent $A(\omega)$. Thus, we would have an inconsistency with the current approach in Salinas of adding the spatial parts together before multiplying by the time function (which in this case would be $A(\omega)$).

D.6 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 absobing boundary conditions. We describe below the simple absorbing boundary conditions that have been implemented in Salinas. A description of infinite elements will be added at a later time. We describe the cases of an acoustic space and an elastic space separately.

D.6.1 Acoustic Space

The implementation of absorbing boundary conditions begins by considering the weak formulation of the equations of motion, in equations D.5. On an absorbing boundary, one needs to consider the term

$$\int_{\partial\Omega_n} \frac{\partial\psi}{\partial n}\phi ds, \tag{D.25}$$

which arises from the integration by parts on the acoustic space. Absorbing boundary conditions are typically derived by applying impedance matching conditions to equation D.25, in such a way that the boundary absorbs waves of a given form exactly. For example, the simplest absorbing boundary conditions consist of plane wave and spherical wave conditions, which can be written as follows[50]

$$\frac{\partial\psi}{\partial n} = \frac{-1}{c_f}\frac{\partial\psi}{\partial t} \tag{D.26}$$

$$\frac{\partial \psi}{\partial n} = \frac{-1}{c_f}\frac{\partial \psi}{\partial t} - \frac{1}{R}\psi \tag{D.27}$$

where R is the radius of the absorbing spherical boundary.

Inserting equation D.26 into equation D.25, we obtain a term propotional to $\dot{\psi}$, which becomes a damping matrix. Inserting equation D.27 into equation D.25, 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 D.26 and D.27 are implemented in Salinas.

D.6.2 Elastic Space

In the case of an elastic space, very similar absorbing boundary conditions can be applied as were in the acoustic space, except now 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 slightly more complicated.

The equation of motion for an elastic space can be written as

$$\rho u_{tt} - \nabla \cdot \sigma = f \tag{D.28}$$

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.

$$\int_V \rho u_{tt} w dV + \int_V \sigma : \nabla w dV - \int_{\partial V} \sigma_s w dS = \int_V f \cdot w dV \tag{D.29}$$

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 exactly the same.

Considering the term

$$\int_{\partial V} \sigma_s w dS \tag{D.30}$$

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

$$\begin{aligned} \sigma_n &= \rho c_L \frac{\partial v_n}{\partial t} \\ \sigma_t &= \rho c_T \frac{\partial v_t}{\partial t} \end{aligned} \tag{D.31}$$

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

D.30 yields two absorbing boundary matrices. Since these matrices involve the velocities, they become part of the overall damping matrix of the structure.

DISTRIBUTION:

1	MS 0380	Garth M. Reese, 01542
1	MS 0380	Timothy F. Walsh, 01542
1	MS 0380	Manoj K. Bhardwaj, 01542
2	MS 9018	Central Technical Files, 8944
2	MS 0899	Technical Library, 4536

Sandia National Laboratories