SAND88-0737
Unlimited Release
Printed March 1989

Distribution
UC-32

NUMBERS:
A Collection of Utilities for Pre- and Postprocessing Two- and Three-Dimensional EXODUS Finite Element Models

NOTE: This document is semi-automatically converted from TeX to FrameMaker to PDF. The cleanup is not complete!!!

Gregory D. Sjaardema
Applied Mechanics Division I
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

NUMBERS is a shell program which reads and stores data from a finite element model described in the EXODUS database format. Within this shell program are several utility routines which calculate information about the finite element model. The utilities currently implemented in *NUMBERS* allow the analyst to determine information such as (1) the volume and coordinate limits of each of the materials in the model; (2) the mass properties of the model; (3) the minimum, maximum, and average element volumes for each material; (4) the volume and change in volume of a cavity; (5) the nodes or elements that are within a specified distance from a user-defined point, line, or plane; (6) an estimate of the explicit central-difference timestep for each material; (7) the validity of contact surfaces or slidelines, that is, whether two surfaces overlap at any point; and (8) the distance between two surfaces.

Contents

1 Introduction

The *NUMBERS* program is a shell program which reads and stores data from a finite element model described in the EXODUS database format [EXODUS]. Within this shell program are several utility routines which calculate information about the finite element model. The utilities currently implemented in *NUMBERS* allow the analyst to determine:

- the volume and coordinate limits of each of the materials in the model;
- the mass properties of the model;
- the minimum, maximum, and average element volumes for each material;
- the volume and change in volume of a cavity;
- the nodes or elements that are within a specified distance from a user-defined point, line, or plane;
- an estimate of the explicit central-difference timestep for each material;
- the validity of contact surfaces or slidelines, that is, whether two surfaces overlap at any point; and
- the distance between two surfaces.

These utilities have been developed to automate and simplify some of the tasks normally performed during an analysis. The *NUMBERS* program reads the finite element model and results from a file written in the EXODUS binary file format which is used in the Engineering Analysis Department at SNLA.

The capabilities of *NUMBERS* have evolved during the past eighteen months. Originally, it was written solely to calculate the mass properties of a body. However, once the basic function of reading and storing an EXODUS database was in place, it was realized that several tasks that were usually performed manually could easily be implemented in *NUMBERS* . Tasks such as determining node and element numbers, verifying contact surfaces, and others, are now performed more efficiently and, hopefully, more accurately since the code performs the repetitive calculations automatically.

Although the original reason for developing *NUMBERS* was to simply calculate mass properties, the code now functions as an EXODUS shell that can be easily extended by analysts who require specific calculations or need to create information not currently available. The analyst can simply write a subroutine to perform their function, and insert it into *NUMBERS* without worrying about the details of reading an EXODUS database and providing a user interface. For most cases, adding a function to *NUMBERS* requires only writing the function subroutine, adding the command name to the table of valid commands, and adding a few statements to call the routine.

The remainder of this report is organized as follows. Chapter [Ref: c:numerics] describes the numerical algorithms used by the utility functions in *NUMBERS* . A list of the commands and the command syntax are presented in Chapter [Ref: c:commands] . Chapter

[Ref: c:examples] gives several examples of the use of the utilities, and Chapter [Ref: c:conclude] concludes the report. Three appendixes are included. Appendix [Ref: a:cmdsum] is a summary of the command syntax for each of the commands, Appendix [Ref: a:site] lists the specifics of running *NUMBERS* on the computer systems at SNLA, and Appendix [Ref: a:exodus] is a description of the EXODUS file format.

The descriptions in the following chapters assume that the reader is familiar with the GENESIS and EXODUS file formats and with the analysis, preprocessing, and postprocessing codes used in the Engineering Analysis Department at SNLA. Readers not familiar with these can check the references at the end of this report for a list of the documentation for these codes and file formats.

2 Description of Utilities in *NUMBERS*

Each utility and the numerical algorithm used in the utility are described in this chapter. Although most of the algorithms are well-known and documented in several references, it is important to understand the actual methods used in each of the utilities to avoid misinterpretation of the output and to understand the limitations and assumptions used to calculate the properties and values.

Another reason for describing the algorithms is that *NUMBERS* is written as a basic shell program for reading EXODUS files. If additional utilities are developed for *NUMBERS* , it is likely that several of the algorithms already implemented can be modified to perform the new function.

This chapter is organized into several sections; each utility is described in a separate section. The descriptions include the function of the utility and the algorithm used in the utility including its assumptions and limitations. The complete command syntax for each utility is given in Section [Ref: sec:oper] ; a summary of the command syntax is given in Section [Ref: sec:opsum] . Examples of the use of most of the utilities are presented in Chapter [Ref: c:examples] .

2.1 Mass Properties Utility

The mass properties utility calculates the volume, mass, mass moments of inertia, and centroid location of the body. These values are often required in a finite element analysis to ensure that the finite element idealization correctly approximates the actual body. This utility also lists the minimum, maximum, and average element volume for each material block. A summary of the equations used to calculate the mass properties is given in this section.

The moments of inertia are originally calculated about the origin of the coordinate system and then transferred to the centroid using the parallel-axis theorem. The mass and the moments of inertia about the origin of the coordinate system are given by the following integrals:

$$M = \int_V \rho \, dV \tag{1}$$

$$I_x = \int_V \rho(y^2 + z^2) dV \tag{2}$$

$$I_y = \int_V \rho(x^2 + z^2) dV \tag{3}$$

$$I_z = \int_V \rho(x^2 + y^2) dV \tag{4}$$

$$I_{xy} = \int_V \rho \, xy dV \tag{5}$$

$$I_{xz} = \int_V \rho \, xz dV \tag{6}$$

$$I_{yz} = \int_V \rho\, yz dV \tag{7}$$

where M is the mass, ρ is the density, I is the mass moment of inertia, and the subscripts x, y, and z denote the axes about which the moment is measured. The double subscripts indicate products of inertia. Note that the product of inertia with respect to any two orthogonal axes is zero if either of the axes is an axis of symmetry.

2.1.0.1 Axisymmetric Two-dimensional Body:

For an axisymmetric two-dimensional body, the integrals are written in cylindrical coordinates using the radius r and the angle θ, where $x = r\cos\theta$ and $z = r\sin\theta$. The y-axis is assumed to be the axis of revolution. The infinitesimal volume dV is equal to $rdAd\theta$ where dA is an infinitesimal area equal to $drdy$. Rewriting Equations \eref V through \eref IZ in terms of r and θ, and performing the theta integration from $-\pi$ to π gives:

$$M = 2\pi\rho\int_A r dA \tag{8}$$

$$I_y = 2\pi\rho\int_A r^3 dA \tag{9}$$

$$I_x = I_z = 2\pi\rho\int_A r\, y^2 dA + \pi\rho \int_A r^3 dA \tag{10}$$

$$= 2\pi\rho\int_A r\, y^2 dA + I_y \backslash \text{over } 2 \tag{11}$$

The products of inertia I_{xy}, I_{xz}, and I_{yz} are zero since all three axes are symmetry axes.

2.1.0.2 Two-dimensional Planar Body:

For a two-dimensional planar body, area moments of inertia are calculated; the depth in the z-direction is ignored. The integrals simplify as follows:

$$I_x = \rho\int_A y^2 dA \tag{12}$$

$$I_y = \rho\int_A x^2 dA \tag{13}$$

$$I_z = \rho\int_A (x^2 + y^2) dA = I_x + I_y \tag{14}$$

$$I_{xy} = \rho\int_A xy dA \tag{15}$$

The products of inertia I_{xz} and I_{yz} are both zero.

Evaluation of Integrals:

The integrals are evaluated using the isoparametric element formulation and Gaussian Quadrature. Details of this process can be found in most books on finite element methodology. In *NUMBERS*, two options are available for integration of the equations. The two-dimensional equations can be integrated with either 1- or 4-point quadrature, and the three-dimensional equations with either 1- or 8-point quadrature. The mass moments of inertia for bodies with non-rectangular elements are calculated more accurately with the

higher-order integration; volumes and areas are integrated exactly with either integration option. The second-order quadrature rule is also useful for calculating the section properties of a non-standard shape. For this purpose, the discretization of the body is only refined enough to capture the essential details of the shape since a structural analysis will not be performed.

2.1.0.3 Calculation of Centroid Location:

The location of the centroid is calculated using the first mass moments about each of the three coordinate axes. The mass moments are summed in each of the three coordinate directions over the range of the elements. The centroid location is then given by the quotient of the mass moment sums divided by the total mass of the body. For a two-dimensional axisymmetric body, x_c and z_c (the x and z coordinates of the centroid) are zero; for a two-dimensional planar body, z_c is zero.

2.2 Locate Utility

The locate utility outputs the numbers of all nodes or elements that are within a specified distance from a user-defined point, line, or plane. This information is often required for plotting results at a specified location in the model, for example, the acceleration or velocity at the centerline of a body during an impact analysis. The equations used to calculate the distances from a node or element centroid to the point, line, or plane are defined in the following subsections. All equations are defined in terms of a three-dimensional body; for a two-dimensional body the z coordinates are deleted.

Note that the point location procedure is also a circle (2D) and sphere (3D) location procedure and the line location procedure is also a cylinder (3D) location procedure since the search interval distance from the point and line can be specified.

In the location utility, the distance for an element is defined to be the distance to the element geometric center. The location of the element center is calculated by summing the four or eight nodal coordinates in each coordinate direction and dividing by the number of nodes.

2.2.1 Point Location

The point location utility outputs all nodes or elements that are a specified distance from a user-specified point. The distance d_r from the node or element to the point (x_0, y_0, z_0) is given by:

$$d_r = \sqrt{(x_0-x_n)^2 + (y_0-y_n)^2 + (z_0-z_n)^2} \tag{16}$$

where x_n, y_n, and z_n are the coordinates of the node or element. The distance is calculated for each node or element and compared to the user-specified distance and tolerance. If the node or element is within this range, its number and coordinates are output. The angles θ and ϕ are also calculated. The angle θ is the angle between the x axis and the projection of the line from the point to the node onto the x–z plane. The angle ϕ is the angle between the y axis and the line from the point to the node. In two dimensions, θ is the angle between the line and the x axis. Figure [Ref: f:theta] illustrates the definitions of θ and ϕ.

2.2.2 Line Location

The line location utility outputs all nodes or elements that are within a specified distance from a user-specified line. The distance is measured normal to the line. The parametric representation of the line from $P_1(x_1, y_1, z_1)$ to $P_2(x_2, y_2, z_2)$ is $P(t) = P_1 + (P_2 - P_1)t$. The components of this line are:

$$x = x_1 + (x_2 - x_1)\, t = x_1 + at \tag{17}$$

$$y = y_1 + (y_2 - y_1)\, t = y_1 + bt \tag{18}$$

$$z = z_1 + (z_2 - z_1)\, t = z_1 + ct \tag{19}$$

The minimum distance d_t from the line to the point (x_n, y_n, z_n) is

$$d_t^2 = (x_1 + at - x_n)^2 + (y_1 + bt - y_n)^2 + (z_1 + ct - z_n)^2 \tag{20}$$

where the parameter t is

$$t = -\text{\frac } a(x_1 - x_n) + b(y_1 - y_n) + c(z_1 - z_n) a^2 + b^2 + c^2 \tag{21}$$

2.2.3 Plane Location

The plane location utility outputs all nodes or elements that are within a specified distance from a user-specified plane. The distance is measured normal to the plane. A unique plane can be defined by a specified point and a normal vector to the plane. Given the point (x_0, y_0, z_0) and the unit vector \v{n} = a\v{i} + b\v{j} + c\v{k}, the equation of the plane is:

$$0 = ax + by + cz + d \tag{22}$$

$$d = -(ax_0 + by_0 + cz_0) \tag{23}$$

The normal distance d_n from the plane to a node or element center is:

$$d_n = \text{\frac } | a\, x_n + b\, y_n + c\, z_n - d | \sqrt{a^2 + b^2 + c^2} \tag{24}$$

where the subscript n refers to the coordinates of the node or element. The normal distance is calculated for each node or element and compared to the user-specified distance and tolerance. If the node or element is within this range, its number, coordinates, normal distance, and radial distance are output. The radial distance is the same distance calculated in the point location utility.

2.2.4 Sort Algorithm

Although sort is not a location option, it is used in the location utility to order the output. *NUMBERS* uses the *heapsort* routine which has been recommended in Reference [Press:nr] as a good routine for a variety of sorting applications. It is an “in-place” sort requiring no auxiliary storage. It is an $N\log_{2N}$ process, not only on average, but also for worst-case order of input data. A FORTRAN listing of the sort subroutine is given in

Appendix [Ref: a:sort] . One disadvantage of the sort routine is that it is not “stable.” This means that sorting the data a second time on a different field will destroy the order of the first sort. A method for sorting on two or more fields simultaneously is being investigated and will be implemented in a later version of *NUMBERS* .

2.2.5 Sort Fields

The output from the location utility can be sorted on any of the calculated quantities or “fields.” The table below lists the fields that are defined for each of the location options; these are defined following the table.

Option	2D/ 3D	Valid *sort_fields*							
POINT	2D	X	Y			DISTANCE		THETA	
POINT	3D	X	Y	Z		DISTANCE		THETA	PHI
LINE	2D	X	Y		T	DISTANCE	PARAMETRIC		
LINE	3D	X	Y	Z	T	DISTANCE	PARAMETRIC		
PLANE	3D	X	Y	Z		DISTANCE		RADIUS	

X, Y, or Z The x, y, or z coordinate of the located node or element center.

T The parametric distance from the initial point of the line to the located node or element center as defined in Equation \eref parametric. The initial point of the line is located at T = 0; the final point is located at T = 1.

PARAMETRIC The same quantity as T.

DISTANCE The distance from the node or element center to the location point, line, or plane. See the respective sections above for the definition of distance for each of the location options.

RADIUS The shortest distance from the plane definition point (see Section [Ref: sec:slocate]) to the located node or element center.

THETA For three-dimensional bodies, THETA is the angle between the *x* axis and the projection of the line from the point to the node or element center onto the *x*–*z* plane (see Section [Ref: sec:plocate]). For two-dimensional bodies, theta is the angle between the line from the point to the node or element center and the *x* axis.

PHI The angle between the *y* axis and the line from the point to the located node or element center.

2.3 Cavity Volume Utility

The cavity volume utility calculates the volume and change in volume of a cavity or hole in a body. The boundary of the cavity is defined by side set flags in the EXODUS file. Two separate calculations are involved in this utility; the first is the calculation of the cavity volume, and the second is the calculation of the change in volume.

The cavity volume is calculated by forming triangles (2D) or pentahedrons (3D) for each segment of the cavity boundary. The apex of the triangles or pentahedrons are at a user-specified point. The bases of the triangles or pentahedrons are the segments of the cavity boundary side set. The segments are element faces; in two-dimensions, the faces are lines; in three-dimensions, the faces are four-node quadrilaterals. The faces are assumed to be planar. The volume of each triangle or pentahedron is calculated and the sum of the volumes is, for certain geometries, the volume of the cavity.

Figure [Ref: cvol2d] is an example of this process for a two-dimensional cavity with three boundary segments: 1–2, 2–3, and 3–4.

The area is calculated by summing the areas of the three triangles 012, 023, and 034, where the three numbers refer to the points defining the triangle and point 0 is the apex or center point. A similar process is used for three-dimensional cavities except that pentahedral volumes are calculated instead of triangular areas. The volume V of a pentahedron with the apex at the point (x_c,y_c,z_c) and the base formed by a boundary segment is [dpfeigen]:

$$V = \sfrac{1}{12}\{ ((2y_c - y_3)\, z_{42} + y_2\, (z_{c3} + z_{c4}) - y_4\, (z_{c3} + z_{c2}))\, x_1 \quad (25)$$

$$+ ((y_4 - 2y_c)\, z_{31} + y_3\, (z_{c4} + z_{c1}) - y_1\, (z_{c4} + z_{c3}))\, x_2 \quad (26)$$

$$+ ((y_1 - 2y_c)\, z_{42} + y_4\, (z_{c1} + z_{c2}) - y_2\, (z_{c4} + z_{c1}))\, x_3 \quad (27)$$

$$+ ((2y_c - y_2)\, z_{31} + y_1\, (z_{c2} + z_{c3}) - y_3\, (z_{c2} + z_{c1}))\, x_4 \quad (28)$$

$$+ (y_{24}\, z_{31} + y_{31}\, z_{42})\, 2\, x_c\} \quad (29)$$

where the numerical subscript refers to the sequence of the node of the boundary segment, the c subscript refers to the center location, and the double subscript z_{ij} is defined as z_i-z_j.

The planar area V_P of a triangle with the apex at the point (x_c,y_c) and the base formed by a boundary segment is:

$$V_P = \sfrac{1}{2}[(y_1-y_c)\, (x_2-x_c) - (y_2-y_c)\, (x_1-x_c)] \quad (30)$$

where the subscript refers to the first and second node of the boundary segment, and y_c is the approximate vertical geometric center of the cavity. It is calculated by summing the y coordinates and dividing by the total number of nodes on the boundary.

If the body is axisymmetric, the volume V_A is calculated as:

$$x_c = \sfrac 1/3(x_1 + x_2) \tag{31}$$

$$V_A = 2\pi\, x_c\, V_P \tag{32}$$

where V_P is the area calculated for the plane strain cavity.

The above method correctly calculates the volume of cavities defined by a closed boundary. However, many cavities are bounded on one or more sides by symmetry planes which are not included in the cavity boundary definition. In two dimensions, the correct volume will be calculated if the triangle apex is on the axis of symmetry. In three dimensions, the correct volume will be calculated if the apex of the pentahedron is on the symmetry plane, or if it is on the intersection of two or more symmetry planes. If the apex point is not specified by the user, it is set to the point (0,0,0) for three-dimensional bodies, and the point $(0,y_c)$ for two-dimensional bodies, where y_c is the approximate vertical geometric center of the cavity.

Cavity Volume Change:
The change in cavity volume is calculated by calculating the volume of the hexahedron formed by the original element face and the displaced element face. This is repeated for each element face on the cavity boundary and the total volume change is the sum of the element face volume changes. Note that this calculation is correct for all cavities even if they are bounded by symmetry planes.

For a three-dimensional cavity, the volume change ΔV for each element on the cavity boundary is given by the following sequence of calculations [PRONTO3D].

$$\Delta V = \sfrac 1/12\sum_{i=1}^{8} x_i B_i \tag{33}$$

where x_i for i\le 4 is the x coordinate of a node on the element face and x_i for 5 \le i\le 8 is the displaced coordinate of node i-4. The B_i values are given by:

$$B_1 = y_2 (z_{63}-z_{45}) + y_3 z_{24} + y_4 (z_{38}-z_{52}) + y_5 (z_{86}-z_{24}) + y_6 z_{52} + y_8 z_{45} \tag{34}$$

$$B_2 = y_3 (z_{74}-z_{16}) + y_4 z_{31} + y_1 (z_{45}-z_{63}) + y_6 (z_{57}-z_{31}) + y_7 z_{63} + y_5 z_{16} \tag{35}$$

$$B_3 = y_4 (z_{81}-z_{27}) + y_1 z_{42} + y_2 (z_{16}-z_{74}) + y_7 (z_{68}-z_{42}) + y_8 z_{74} + y_6 z_{27} \tag{36}$$

$$B_4 = y_1 (z_{52}-z_{38}) + y_2 z_{13} + y_3 (z_{27}-z_{81}) + y_8 (z_{75}-z_{13}) + y_5 z_{81} + y_7 z_{38} \tag{37}$$

$$B_5 = y_8 (z_{47}-z_{61}) + y_7 z_{86} + y_6 (z_{72}-z_{18}) + y_1 (z_{24}-z_{86}) + y_4 z_{18} + y_2 z_{61} \tag{38}$$

$$B_6 = y_5 (z_{18}-z_{72}) + y_8 z_{57} + y_7 (z_{83}-z_{25}) + y_2 (z_{31}-z_{57}) + y_1 z_{25} + y_3 z_{72} \tag{39}$$

$$B_7 = y_6 (z_{25}-z_{83}) + y_5 z_{68} + y_8 (z_{54}-z_{36}) + y_3 (z_{42}-z_{68}) + y_2 z_{36} + y_4 z_{83} \tag{40}$$

$$B_8 = y_7 (z_{36} - z_{54}) + y_6 z_{75} + y_5 (z_{61} - z_{47}) + y_4 (z_{13} - z_{75}) + y_3 z_{47} + y_1 z_{54} \quad (41)$$

where y_i and z_i are defined in the same way as x_i, and z_{ij} is equal to $z_i - z_j$.

For a two-dimensional, planar cavity, the volume change ΔV_P for each element on the cavity boundary is equal to

$$\Delta V_P = \sfrac 1/2[x_{12} (\Delta y_2 + \Delta y_1) - y_{12} (\Delta x_2 + \Delta x_1) + \Delta x_1 \Delta y_2 - \Delta x_2 \Delta y_1] \quad (42)$$

where x and y are the original coordinates of the face nodes and Δx and Δy are the x and y displacements of the nodes.

For an axisymmetric cavity, the volume change ΔV_A for each element on the cavity boundary is equal to

$$\Delta V_A = 2\pi x_c \Delta V_P \quad (43)$$

where x_c is the x coordinate of the geometric center which is equal to

$$x_c = \frac (\Delta x_2 + \Delta x_1)4 + \frac (x_2 + x_1)2 \quad (44)$$

2.4 Overlap Utility

The overlap utility is intended to assist the analyst in the generation of a mesh with valid contact surfaces. One of the most difficult tasks in generating a valid contact surface occurs when the contact surface is curved with different discretizations on the two surfaces. In this case, a small gap must be inserted between the surfaces to prevent penetration. Figure [Ref: f:overlap] illustrates the problem that can occur if the gap is not large enough or nonexistent. Note that the nodes on the more refined side of the contact surface penetrate the elements on the other side of the surface. This is a somewhat contrived example to illustrate the problem; in actual finite element meshes it is very difficult to notice the overlap unless the mesh is examined element by element. However, if an analysis is run with overlapping contact surfaces, the problem manifests itself in a sudden increase in kinetic energy or by excessive deformations and velocities in the overlapped portion of the mesh since the code will separate the contact surface in one timestep. Currently, the utility only checks for penetration of the master surface by the slave surface; however, other checks for valid contact surfaces, such as continuity or other requirements imposed by analysis codes, can be added if there is a need.

Many times, the most efficient method to determine whether the contact surfaces overlap is to run the analysis for a very short period of time, and then examine the results to see if the kinetic energy has increased suddenly, or if the mesh has been “blown” apart where the analysis code detects an overlapping contact surface and separates it in one timestep—both symptoms indicate that a slideline overlap exists in the original mesh.

The overlap utility was written to provide an efficient means of determining whether an overlap exists prior to running an analysis. The algorithm is logically broken into three separate steps. In the first step, a "bounding box" is defined for each element on the master surface. The bounding box contains the coordinate ranges in each of the coordinate directions. Secondly, for each element on the master surface, each node on the slave surface is tested to determine if it is within the bounding box. This test is a simple comparison of the slave node coordinates with the coordinates of the bounding box. Finally, if a node is within an element's bounding box, the more computationally expensive calculation of determining whether the node penetrates the element is performed. To determine if the node penetrates the element, four triangles (two-dimensional body) or six pentahedrons (three-dimensional body) are formed with the slave node as the apex and each element face as the base. The volume of each triangle or pentahedron is calculated; if all of the volumes are positive, the node is inside the element; if a volume is equal to zero, the node is on the face. If the node is inside the element, its node number and the number and connectivity of the penetrated element are output. At the end of the calculation, the total number of nodes penetrating the surface and the number of nodes on the surface are output.

The volume V_P^i of the three-dimensional pentahedron formed by slave node *s* and the element face *i* is equal to [dpfeigen]:

$$12V_P^i = x_A[(2y_S - y_C)\, z_{DB} + y_B\,(z_{SC} + z_{SD}) - y_D\,(z_{SC} + z_{SB})] \quad (45)$$

$$+ x_B[(y_D - 2y_S)\, z_{CA} + y_C\,(z_{SD} + z_{SA}) - y_A\,(z_{SD} + z_{SC})] \quad (46)$$

$$+ x_C[(y_A - 2y_S)\, z_{DB} + y_D\,(z_{SA} + z_{SB}) - y_B\,(z_{SD} + z_{SA})] \quad (47)$$

$$+ x_D[(2y_S - y_B)\, z_{CA} + y_A\,(z_{SB} + z_{SC}) - y_C\,(z_{SB} + z_{SA})] \quad (48)$$

$$+ 2x_S[y_{BD}\, z_{CA} + y_{CA}\, z_{DB}] \quad (49)$$

where the subscript *S* refers to the slave node, and the subscripts *A*, *B*, *C*, and *D* refer to the nodes on an element face (assumed to be planar) as given in Table 1.]

Figure 1: Numbering of Face Nodes on an Eight-Node Hexahedral Element

Hexahedron	Face					
	1	*2*	*3*	*4*	*5*	*6*
A	1	6	6	5	4	1
B	2	7	5	1	3	5
C	3	3	8	4	7	6
D	4	2	7	8	8	2

The volume V_T^i of the two-dimensional triangle formed by the slave node *s* and the element face *i* is equal to:

$$2V_T^i = x_A (y_B - y_S) + x_B (y_S - y_A) + x_S (y_A - y_B) \quad (50)$$

where the subscript *S* refers to the slave node, and the subscripts *A* and *B* refer to the nodes on an element face as given in Table 2.

Figure 2: Numbering of Face Nodes on an Four-Node Quadrilateral Element

Quadrilateral	Face			
	1	*2*	*3*	*4*
A	1	2	3	4
B	2	3	4	1

2.5 Time Step Estimation Utility

Explicit integration is used in most large-deformation, nonlinear, transient dynamics finite element codes, for example \code PRONTO [PRONTO2D,PRONTO3D]. This utility provides an estimate of the time step size which will be used in the explicit integration in these codes.

The stable time step Δt for the central difference operator commonly used in transient dynamic analysis codes is given by [Cook]

$$\Delta t \leq 2\over \omega_{max} (\sqrt{1+\varepsilon^2} - \varepsilon) \quad (51)$$

where ω_{max} is the maximum frequency of the mesh, and ε is the fraction of critical damping in the highest element mode. In a explicit integration finite element code, the linear bulk viscosity term is an estimate of ε [PRONTO2D]. The default value of ε is 0.06 which is the default value of the linear bulk viscosity used in \code PRONTO2D and \code PRONTO3D.

Flanagan and Belytschko [dpfeigen] have derived simple formulas for bounding the maximum eigenvalues for the uniform strain eight-node hexahedron and four-node quadrilateral which can be used to provide conservative estimates of the maximum frequency. The maximum frequency estimate for a rectangular quadrilateral element is

$$\hat{\omega}_{max}^2 = 4(\lambda + 2\mu)\over \rho(1\over s_1^2 + 1\over s_2^2) \quad (52)$$

where s_1 and s_2 are the lengths of the sides of the rectangle, λ and μ are Lame's constants, ρ is the density, and $\hat{\omega}_{max}$ is the predicted value for the maximum frequency. Similarly, for a rectangular parallelepiped hexahedron,

$$\hat{\omega}_{max}^2 = 4(\lambda + 2\mu)\over \rho(1\over s_1^2 + 1\over s_2^2+1\over s_3^2) \qquad (53)$$

Substituting the maximum frequency equation into the stable time step equation gives the following estimate for the stable time step size:

$$\Delta\hat{t} \le \sqrt{\frac{\rho}{\lambda+2\mu}} \left(\sum_{i=1}^{n_D} \frac{1}{s_i^2}\right)^{-1/2} \left(\sqrt{1+\varepsilon^2} - \varepsilon\right) \qquad (54)$$

where $\Delta\hat{t}$ is the estimate of the stable time step, and n_D is the number of dimensions. The first quantity on the right-hand side of the inequality is the inverse of the dilatational wave speed which is input by the user. The second quantity is calculated by *NUMBERS* with the assumption that the element is rectangular.

The output from this utility includes the calculated time step, the element with the minimum time step, and the number of time steps per millisecond of analysis time for each element block.

The number of time steps per millisecond can be used to estimate the computer time required to perform an analysis. Most explicit transient dynamics codes output the average CPU time required to perform the calculations for one element for one time step. Although this quantity varies for different constitutive models and the number of contact surfaces (slidelines), the average value is usually relatively constant and well known. The CPU time per millisecond of analysis time can be estimated using the formula

$$CPU\over ms = (1\E -3/ \over \Delta\hat{t}) (Speed)(NUMEL) \qquad (55)$$

where NUMEL is the number of elements, Speed is the CPU time per element per timestep, and (1\E -3/ / $\Delta\hat{t}$) is the number of time steps per millisecond.

2.6 Gap Utility

The gap utility is used to determine the distance between nodes on two surfaces. One surface is called the “master” surface and the other surface is the “slave” surface. For each node on the master surface, a normal vector is calculated as shown below. A matching process is then performed to determine the “closest” slave node to each master node, where closeness can be defined either as closest to the master node normal vector, or closest in absolute distance. Figure [Ref: f:gap] illustrates the two closeness measures. In this figure, Node M is the master node and nodes S_1 and S_2 are two slave nodes. If the absolute distance is used, node S_1 will be the matching node; if the distance to the normal vector is used, node S_2 will be the matching node.

After all of the nodes on the master surface have been matched to a node on the slave surface, the normal and tangential distances, measured in the coordinate frame of the normal vector, are determined for the undeformed geometry and for each of the selected timesteps in the database. This utility is normally used to calculate the change in distance between two surfaces, for example, the closure of a drift in a geomechanics problem or the slip (tangential movement) along a contact surface.

The normal for a node is defined to be the average normal of the element faces for all elements connected to the node. For a three-dimensional body, a technique developed by Newell [Rogers:pefcg] gives the exact normal for planar faces and a "best" approximation for almost planar faces. The coefficients *a*, *b*, and *c* of the normal vector for an element face \v{n} = *a*\v{i} + *b*\v{j} + *c*\v{k} are given by:

$$a = \sum_{i=1}^{n} (y_i - y_j)(z_i + z_j) \tag{56}$$

$$b = \sum_{i=1}^{n} (z_i - z_j)(x_i + x_j) \tag{57}$$

$$c = \sum_{i=1}^{n} (x_i - x_j)(y_i + y_j) \tag{58}$$

where *n* is the number of nodes per face, and *j* = 1 if *i*=*n*, or *j*=*i*+1 if *i*\ne *n*. The vector is then normalized and added to the direction cosine matrix for each node on the face. After all of the element normals have been computed and added to the direction cosine matrix, the average direction cosine unit vector for each node is determined by normalizing each entry in the direction cosine matrix.

The procedure is similar for a two-dimensional body, except that the normal vector for an element face \v{n} = *a*\v{i} + *b*\v{j} is given by:

$$r = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{59}$$

$$a = (x_i - x_j) / r \tag{60}$$

$$b = (y_i - y_j) / r \tag{61}$$

where x_i and x_j are the *x* coordinates of the two face nodes, and y_i and y_j are the *y* coordinates of the two face nodes.

After the average nodal normals have been calculated, the node matching process is performed. For each node on the master surface, the *closest* node on the slave surface is determined. There are two criteria for determining closeness: radial distance and tangential distance. The radial distance d_r between two nodes is simply

$$d_r = \sqrt{(x_s - x_m)^2 + (y_s - y_m)^2 + (z_s - z_m)^2} \tag{62}$$

where *x*, *y*, and *z* are the nodal coordinates and the subscripts *m* and *s* refer to the master and slave nodes, respectively.

The tangential distance is the distance from the master node's normal vector to the slave node. The distance is given by [Rogers:pefcg]:

$$d_t = \sqrt{d_r^2 - d_n^2} \tag{63}$$

$$d_n = -(a(x_m - x_s) + b(y_m - y_s) + c(z_m - z_s)) \quad (64)$$

where d_r is the radial distance given in Equation \eref distr, d_n is the distance to the slave node measured in the direction of the normal vector, and a, b, and c are the components of the unit normal vector determined in Equations \eref 3dnorm or \eref 2dnorm.

2.7 Limits Utility

The limits utility provides the basic function of determining the minimum, maximum, and range of the coordinates for each of the material blocks. No special algorithms are used in this utility and there are no inherent limitations or assumptions in its implementation. The utility simply examines the coordinates of each element in each element block and saves the minimum and maximum values. After all of the elements have been processed, the minimum and maximum values are subtracted to determine the coordinate range of the data.

3 Command Input

Each command in *NUMBERS* is described below. The commands are divided into four categories: (1) operational commands, (2) parameter-setting commands, (3) database display commands, and (4) miscellaneous commands. Operational commands are the commands that actually perform the utility functions of *NUMBERS* , such as calculating the mass properties, or verifying the correct definition of contact surfaces. The parameter-setting commands are used to initialize or modify the parameters required by the operational commands such as the material density which is required to calculate the mass properties. The database display commands echo information contained on the database to the terminal. Miscellaneous commands are simply commands that do not fit into any of the previous categories.

3.1 Command Syntax

The commands are in free-format and must adhere to the following syntax rules.

- Valid delimiters are a comma or one or more blanks.
- Either lowercase or uppercase letters are acceptable, but lowercase letters are converted to uppercase.
- A "$" character in any command line starts a comment. The "$" and any characters following it on the line are ignored.
- A command may be continued over several lines with an "*" character. The "*" and any characters following it on the current line are ignored and the next line is appended to the current line.

Each command has an action keyword or "verb" followed by a variable number of parameters.

The command verb is a character string. It may be abbreviated, as long as enough characters are given to distinguish it from other commands.

The meaning and type of the parameters is dependent on the command verb. Below is a description of the valid entries for parameters.

- A numeric parameter may be a real number or an integer. A real number may be in any legal FORTRAN numeric format (e.g., 1, 0.2, -1E-2). An integer parameter may be in any legal integer format.
- A string parameter is a literal character string. Most string parameters may be abbreviated.
- Variable names must be fully specified. The blank delimiter creates a problem with database variable names with embedded blanks. The program handles this by deleting all embedded blanks from the input database names. For example, the variable name "SIG R" must be entered as "SIGR". The blank must be deleted in any references to the variable.

- Some parameters allow a range of values. A range is in one of the following forms:
- "n_1" selects value n_1,
- "n_1 TO n_2" selects all values from n_1 to n_2,
- "n_1 TO n_2 BY n_3" selects all values from n_1 to n_2 stepping by n_3, where n_3 may be positive or negative.

If the upper limit of the range is greater than the maximum allowable value, the upper limit is changed to the maximum without a message.

The notation conventions used in the command descriptions are:

- The command verb is in **bold** type.
- A literal string is in all uppercase SANSERIF type and should be entered as shown (or abbreviated).
- The value of a parameter is represented by the parameter name in *italics*.
- A literal string in square brackets ("[ ]") represents a parameter option which is omitted entirely (including any following comma) if not appropriate. These parameters are distinct from most parameters in that they do not require a comma as a place holder to request the default value.
- The default value of a parameter is in angle brackets ("< >"). The initial value of a parameter set by a command is usually the default parameter value. If not, the initial setting is given with the default or in the command description.
- Two or more literal strings separated by a vertical bar ("|") represents a list of valid options. Only one of the options is allowed. If the strings are in square brackets ("[ ]"), they represent a parameter option which can be omitted entirely; if the strings are in braces ("{ }"), one of the strings must be entered.

3.2 Operational Commands

MASS *nquad* <1 or previous value,> \optparam density
PROPERTIES *nquad* <1 or previous value,> \optparam density

mass or properties calculates several mass properties of the body. The calculated properties are the volume and mass of each element block; the mass or area moments of inertia; the location of the center of gravity; the minimum, maximum, and average element volume for each element block; and the number of elements in each element block.

The parameter *nquad* selects the number of quadrature (numerical integration) points used to calculate the mass properties. It must be 1 or 4 for a two-dimensional mesh, and 1 or 8 for a three-dimensional mesh. If it is not entered, it will be set to the last entered value, or to 1 if it has never been set. If all of the material densities are equal, they can be set by entering a positive value for the optional parameter *density*. If the densities are not equal, they must be input through the density command.

If the exodus switch is ON [Footnote: See command EXODUS] , the mass properties will be calculated for the deformed geometry of the body at each selected time step; if exodus is OFF, values will be calculated for the undeformed geometry. Section [Ref: sec:mass] describes the algorithms used in this calculation, and the resulting output.

LOCATE {NODES|ELEMENTS} *locate_option*

LOCATE is used to locate NODES or ELEMENTS that are within a specified distance from a user-defined point, line, or plane. The available options are defined below. The algorithms used are described in Section [Ref: sec:locate] . If ELEMENTS is specified, distance is measured to the element centroid. In the descriptions of the *locate_options* below, the *z* coordinates are omitted for two-dimensional bodies. LOCATE calculates distances only for the undeformed geometry irregardless of the setting of the EXODUS switch.

LOCATE {NODES|ELEMENTS} POINT x_0, y_0, z_0, [distance, *toler]*

outputs all nodes or elements located a distance of *distance*±*toler* from the point \param x_0, y_0, z_0. If no nodes or elements are located in the specified range, the minimum calculated distance is printed. The output includes the node or element number, the coordinates of the node or the element center, the distance to the point, and the angles θ and ϕ [Footnote: ϕ is defined for three-dimensional bodies only] for each node or element in the specified range. The angles are defined in Section [Ref: sec:plocate] .

LOCATE {NODES|ELEMENTS} LINE {x_0, y_0, z_0}, {x_1, y_1, z_1}, [distance, *toler*], [BOUNDED|UNBOUNDED] <UNBOUNDED>

outputs all nodes or elements located a distance of *distance*±*toler* from the line defined by the two points \param x_0, y_0, z_0, and \param x_1, y_1, z_1. If bounded is specified, the node or element center must lie between the two end points. If no nodes or element centers are located in the specified range, the minimum calculated distance is printed. The output includes the node or element number, the coordinates of the node or the element center, and the normal and parametric distances from the node or element center to the line for each node or element in the specified range. The distances are defined in Section [Ref: sec:llocate] .

LOCATE {NODES|ELEMENTS} PLANE {x_0, y_0, z_0}, {*i*, *j*, *k}*,[distance, *toler]*

outputs all nodes or elements located a distance of *distance*±*toler* from the plane that passes through the point \param x_0, y_0, z_0 and has a normal defined by the vector \param *i*, *j*, *k*. This command is only defined for three-dimensional bodies. If no nodes or elements are located in the specified range, the minimum calculated distance is printed. The output includes the node or element number, the coordinates of the node or the element center, and the normal and radial distances to the plane for each node or element in the specified range. The distances are defined in Section [Ref: sec:slocate] .

CAVITY *sset*$_1$, *sset*$_2$, ..., *sset*$_n$, [CENTER, *xcen, ycen, zcen*]

CAVITY calculates the volume of a cavity defined by the element sidesets *sset*$_1$ to *sset*$_n$. If the literal string CENTER is entered, the following values are taken as the center point in determining the cavity volume. The cavity volume algorithm, the effect of the center point, the resulting output, and the rules for defining a valid cavity are described in Section [Ref: sec:cavity] .

If the EXODUS switch is ON [Footnote: See command EXODUS] , the cavity volume, the total change in volume, the volume change over the last time step, and the rate of volume change are calculated for each of the selected time steps, else only the undeformed cavity volume is calculated.

LIMITS ALLTIMES

limits determines the minimum, maximum, and range of the X, Y, and Z coordinates for each of the selected material blocks. If ALLTIMES is specified, the limits are determined for each of the selected time steps. Note that the exodus switch status does not affect this command.

OVERLAP *flag*$_m$, *flag*$_s$

overlap is used to verify contact surfaces or slidelines. The command determines whether any of the nodes on the surface defined by the sideset *flag*$_s$ penetrate the element faces on the surface defined by the sideset *flag*$_m$. If a node on the *flag*$_s$ surface penetrates the *flag*$_m$ surface, the node and element numbers are output along with the connectivity array for the penetrated element. The algorithm and assumptions used in this calculation are described in Section [Ref: sec:overlap] . overlap checks only the undeformed geometry irregardless of the setting of the EXODUS switch.

TIMESTEP *wave_speed* <1.0,> [*damping_fraction*] <0.06>

TIMESTEP estimates the critical time step which will be used in an explicit transient dynamics finite element program. If all of the materials have the same dilitational wavespeed, it is entered as *wave_speed*; else, command WAVESPEED is used to enter the different wavespeeds for each material. The optional parameter *damping_fraction* is the fraction of critical damping in the highest element.

The output from this command lists the dilitational wavespeed, the minimum stable time step, the element number, and the number of timesteps per millisecond of analysis time for each material block. Section [Ref: sec:timestep] explains each of these values and the method used to estimate the timestep.

GAP *flag*$_m$, *flag*$_s$, *d*$_{max}$, [DISTANCE|NORMAL]

gap calculates the distance between nodes on the surfaces defined by the sideset flags *flag*$_m$ and *flag*$_s$. For each node on the sideset *flag*$_m$, the program searches for

a matching node on the sideset $flag_s$. A matching node is the closest node measured according to the method specified by DISTANCE|NORMAL. If DISTANCE is specified, the distance is simply the actual distance between the two nodes; if normal is specified, the distance is the tangential distance from the node on the $flag_s$ surface to a line passing through the node on the $flag_m$ surface and normal to the surface. The normal vector is the average of the face normals of the elements connected to the node. If the distance to the closest $flag_s$ node is greater than d_{max}, then the $flag_m$ node has no match and is not shown in the output.

For each $flag_m$ node with a match, the output shows the matching $flag_s$ node, the direction cosines of the surface normal, and the normal and tangential distance from the $flag_m$ node to the $flag_s$ node.

If the EXODUS switch is ON, the above information is output for each time step. The node matching process and the determination of the surface normal are only performed on the undeformed geometry; therefore, the same nodes are monitored during deformation and the distances output for subsequent time steps are measured relative to the undeformed normal.

This command is normally used to calculate the change in distance between two surfaces, for example, the closure of a drift in a geomechanics problem. Note that unless d_{max} is specified, a match will be found for every node on the $flag_m$ surface. The output from this command can be very large; it is recommended that echo be turned off prior to executing a GAP command. The algorithm and assumptions used for this command are described in Section [Ref: sec:gap] .

3.3 Parameter-Setting Commands

AXISYMMETRIC

AXISYMMETRIC informs *NUMBERS* that the two-dimensional body described in the \EXO database is axisymmetric. All of the relevant operational commands will calculate values based on an axisymmetric geometry. This is the default for two-dimensional bodies.

PLANAR or PLANE

PLANAR informs *NUMBERS* that the two-dimensional body described in the \EXO database is planar or plane strain. All of the relevant operational commands will calculate values based on a planar geometry.

DENSITY

DENSITY prompts the user for the mass density of each of the element blocks. The density is used in the mass properties calculation. If densities are not entered prior to entering a MASS or PROPERTIES command, the density command will be executed automatically. An optional 16-character label can be entered following the density. The label will be used to identify the material blocks in the mass properties output. The label cannot contain any blanks.

WAVESPEED

WAVESPEED prompts the user for the dilitational wavespeed of each of the element blocks. The wavespeed is used in the critical time step estimation calculation. If wavespeeds are not entered prior to entering a TIMESTEP command, the WAVESPEED command will be executed automatically. An optional 16-character label can be entered following the dilitational wavespeed. The label will be used to identify the material blocks in the time step estimation output. The label cannot contain any blanks.

SORT *sort_field* [*sort_order*]

SORT is used with the LOCATE command to specify which quantity in the locate output is sorted (*sort_field*) and in which order (*sort_order*). The valid entries for *sort_field* are shown in the following table.

Option	2D/3D	Valid *sort_fields*							
POINT	2D	X	Y			DISTANCE		THETA	
POINT	3D	X	Y	Z		DISTANCE		THETA	PHI
LINE	2D	X	Y		T	DISTANCE	PARAMETRIC		
LINE	3D	X	Y	Z	T	DISTANCE	PARAMETRIC		
PLANE	3D	X	Y	Z		DISTANCE		RADIUS	

The *sort_order* can be UP, ASCENDING, or INCREASING for a sort in increasing order; or DOWN, DESCENDING, or DECREASING for a sort in decreasing order. INCREASING is the default. The fields are defined in Section [Ref: sortfields]

SELECT *option*

SELECT selects the materials or material blocks for the operational commands.

SELECT BLOCKS [ALL], $block\_id_1$, $block\_id_2$, ...

<all element blocks>

selects the element blocks. The *block_id* is the element block identifier displayed by the LIST BLOCKS command. The relevant operational commands will only calculate values for the selected element blocks.

\cmdoption SELECT MATERIALS \optparam ALL, $material\_id_1$, $material\_id_2$, ...

<all materials>

selects the element blocks with the material number corresponding to *material_id*.

The relevant operational commands will only calculate values for the selected materials.

EXODUS \optparam ON|OFF <ON if timesteps present, else OFF>

EXODUS sets the EXODUS switch to ON which indicates that the operational commands should calculate values for each of the selected time steps, or OFF which indicates that the operational commands should only calculate values for the basic undeformed geometry.

This command is used with the time selection commands defined below. There is a subtle difference between calculations performed with the EXODUS switch set to off and the calculations performed at the first time step since the first time step may have nonzero displacements that will cause the calculations to be performed on a deformed geometry.

TMIN *tmin* <minimum database time>

TMIN sets the minimum selected time *tmin* to the specified parameter value.

TMAX *tmax* <maximum database time>

TMAX sets the maximum selected time *tmax* to the specified parameter value.

DELTIME \param Δ *t* <0.0>

DELTIME sets the selected time interval \param Δ *t* to the specified parameter value.

TIMES *tmin* TO *tmax* BY \param Δ *t*

TIMES sets *tmin*, *tmax*, and \param Δ *t* with a single command.

3.4 Database Display Commands

LIST *option*

LIST displays the database information specified by *option* on the user's terminal. The "selected" items are specified with the SELECT command.

\cmdoption LIST {VARS|VARIABLES}

displays a summary of the database. The summary includes the database title; the number of nodes, elements, and element blocks; the number of node sets and side sets; and the number of each type of variable.

\cmdoption LIST {BLOCKS|MATERIALS}

displays a summary of the element blocks. The summary includes the block identifier, the number of elements in the block, the element number of the first and last element in the block, and the block status, either selected or not selected.

\cmdoption LIST {NSETS|NODESETS}

displays a summary of the node sets. The summary includes the set identifier and the number of nodes in the set.

\cmdoption LIST {SSETS|SIDESETS}

displays a summary of the side sets. The summary includes the set identifier, the number of elements in the set, and the number of nodes in the set.

\cmdoption LIST STEPS

displays the number of time steps and the minimum and maximum time step times.

\cmdoption LIST TIMES

displays the step numbers and times for all time steps on the database.

HELP *command* <no parameter>

HELP displays information about the program command given as the parameter. If no parameter is given, all the command verbs are displayed. HELP is system-dependent and may not be available on all systems.

3.5 Miscellaneous Commands

ECHO \optparam ON|OFF <ON>

ECHO controls the output of results to the terminal. If ON, the results are output to the screen. If OFF, PRINT is automatically set to ON and results are output to the output file only. Error messages and information from a LIST command are always output to the terminal.

PRINT \optparam ON|OFF <ON>

PRINT controls the output of results to the output file. If ON, the results are printed to the output file. If OFF, ECHO is automatically set to ON and results are output to the terminal only.

COMMENT \optparam PAGE, *ncom*<1>

COMMENT prompts the user for *ncom* comment lines which are written to the output file. If the optional parameter PAGE is present, a page eject will be written to the output file prior to writing the comment line(s).

EXIT

EXIT exits the program and closes all files.

4 Utility Usage Examples

Examples of the usage of the utilities described in the report will be given in this chapter. A two-dimensional finite element model of a portion of a transportation container will be used for most of the examples. The outline of the model is shown in Figure [Ref: exmodel] .

The model is composed of nine materials. Material 1 is the outer shell which is a \sfrac 3/ 8-inch thick stainless steel wall with four elements through the thickness. Material 2 is the inner shell which is a \sfrac 1/4-inch thick stainless steel wall with three elements through the thickness. The energy-absorbing foam between the inner and outer shells is Material 3. Above the inner shell is another layer of energy-absorbing foam which is Material 7; on top of the foam is a thin 0.120-inch thick steel sheet (Material 9), and the cargo material (Material 10). Materials 11 and 22 are the same thickness as Materials 1 and 2, respectively, except that the density has been increased to account for the top half of the container. The 6-inch diameter punch at the bottom of the model is Material 8. There are five contact surfaces in this model that are defined by ten side sets. The contact surfaces and the corresponding side set identifications and locations are defined in the following table.

Contact Surface	Side Set Number	Location
1	100	Bottom of outer shell
	999	Top and side of punch
2	200	Top of outer shell
	300	Bottom of energy-absorbing foam
3	400	Top of energy-absorbing foam
	500	Bottom of inner shell
4	600	Top of inner shell
	700	Bottom of foam beneath cargo
5	900	Top of sheet beneath cargo
	1000	Bottom of cargo

Several of the utilities in *NUMBERS* were used during the development of this model. Examples of the mass properties, overlap, locate, and timestep utilities will be shown below using the above model.

4.1 mass properties Utility Example

The mass properties command was used to determine the pseudo-densities of Materials 10, 11, and 22. These densities are called “pseudo-densities” since the density is modified to account for the mass of the body that was omitted from the finite element model. Material 10 is the cargo material that has a total mass of 22.0 \mass (8,500 pounds), and Materials 11 and 22 are the tops of the inner and outer shells which are used to simulate the mass of the top half of the container. The mass utility was used to determine the volume of each of the materials in the model. The density of the cargo (Material 10) was then calculated by dividing the cargo mass by the cargo volume. Materials 11 and 22 are intended to simulate the mass of the upper half of the container. The density ρ_{11} of Material 11 is calculated as:

$$\rho_{11} = \rho_1 \, (V_1 + 2V_{11}) \, / \, V_{11} \tag{65}$$

where ρ_1 and V_1 are the density and volume of Material 1, and V_{11} is the volume of Material 11. The input required to calculate the mass properties for an axisymmetric body with distinct densities using four-point quadrature is:

```
AXISYMMETRIC
DENSITY
7.324E-04  OUTER_SHELL
3.297E-03  MASS_OUTER_SHELL
1.035E-05  FOAM
7.324E-04  INNER_SHELL
2.901E-03  MASS_INNER_SHELL
1.603E-05  CARGO_FOAM
7.324E-04  PUNCH
7.324E-04  THIN_SHEET
7.781E-04  CARGO_MASS
MASS 4
EXIT
```

Figure [Ref: exmassout] shows the resulting output. Note that the material labels cannot contain spaces; the character “_” was used to simulate a space. Although only the masses and volumes were needed for this example, the output also shows the location of the centroid, the mass moments of inertia, and the minimum, maximum, and average element volumes for each material block.

The mass properties utility was also used during the documentation of this analysis to estimate the weight savings due to using a thinner wall. A separate analysis of this body was performed with a \sfrac 3/8-inch thick inner wall and a \sfrac 1/2-inch thick outer wall. *NUMBERS* was then run to determine the mass of the inner and outer shells for both models so that the weight differences could be calculated.

4.2 overlap Utility Example

This example will show the input and output for the overlap utility. Each of the contact surfaces in the model will be checked to see if they are penetrated. Since this is a model that has already been tested, none of the contact surfaces overlap. The input for the command is:

```
OVERLAP  100   999
OVERLAP  200   300
OVERLAP  400   500
OVERLAP  600   700
OVERLAP  900  1000
EXIT
```

where the first flag is the master surface and the second flag is the slave surface. Figure [Ref: exoverlapout] shows the resulting output. Normally, the contact surfaces should be checked in both directions, that is, check first with one flag as the master and then check again with that flag as the slave. Figure [Ref: exoverlap2] shows the output from the overlap utility when the contact surfaces do overlap. The output shows the number of the penetrating node, the number of the penetrated element, and the connectivity of the element.

4.3 locate Utility Example

The analysis performed on the example finite element model was concerned with the response of the outer steel shell due to the impact with the punch. The relatively thin outer shell of this model makes it very difficult to examine the results of the finite element analysis since it is not possible to plot contours of the stresses or strains in the steel shell. However, the \code SPLOT module in the postprocessing code *BLOT* [BLOT] can be used to plot the stresses and strains as a function of the element number. Although it is possible to determine the element numbers of the elements in the outer shell by looking at plots of the model, an easier and more efficient method is to use the locate command. The outer shell has a inner radius of 95.75 inches and a thickness of \sfrac 3/8 inch. Since there are four elements through the thickness, each element is \sfrac 3/32 inch thick. Using the locate point command with the center point at the center of curvature (0.0, 24.2) and a distance of 95.75 + \sfrac 3/64 will locate all elements in the innermost element row; a distance of 95.75 + \sfrac 21/64 will locate all elements in the outermost element row. The input used to locate all elements in the outermost element row is:

```
SORT THETA
LOCATE ELEMENTS POINT   0  24.2     96.078125   .001
EXIT
```

Figure [Ref: exlocateout] is the resulting output. The output is sorted on the theta field so that the elements are ordered from the centerline outward.

4.4 limits Utility Example

The limits utility is executed simply by entering the command `LIMITS`. The resulting output is shown in Figure [Ref: exlimitsout] .

4.5 timestep Estimation Example

The final example using this model will be the estimation of the timestep using the timestep command. The input is:

```
WAVESPEED
2.273e+05  OUTER_SHELL
1.071e+05  MASS_OUTER_SHELL
2.003e+04  FOAM
2.273e+05  INNER_SHELL
1.142e+05  MASS_INNER_SHELL
2.731e+04  CARGO_FOAM
2.273e+05  PUNCH
2.273e+05  THIN_SHEET
1.525e+05  CARGO_MASS
TIMESTEP
EXIT
```

The resulting output is shown in Figure [Ref: extimestepout] .

The minimum timestep is 2.273\E -7/ seconds for Material 2 (the inner shell). This is exactly equal to the timestep that was used in the \code PRONTO2D analysis. For this problem, the average CPU time per element per timestep in \code PRONTO2D was 13.3 μ *s*. Since there are 6,825 elements and 4,399 timesteps per millisecond of analysis time, the estimated CPU time per millisecond is 400 seconds. The actual CPU time per millisecond was 395 seconds. This example shows that it is possible to get a very accurate estimate of the time required to perform an analysis prior to running the analysis. The timestep estimation is also useful in determining which material has the smallest or controlling timestep; if the material with the controlling timestep is not in the area of interest, reducing the refinement in that material will give a larger timestep and less analysis time.

5 Summary and Conclusions

The *NUMBERS* program is a shell program which reads and stores data from a finite element model described in the EXODUS database format [EXODUS]. Within this shell program are several utility routines which calculate information about the finite element model. The utilities currently implemented in *NUMBERS* allow the analyst to determine:

- the volume and coordinate limits of each of the materials in the model;
- the mass properties of the model;
- the minimum, maximum, and average element volumes for each material;
- the volume and change in volume of a cavity;
- the nodes or elements that are within a specified distance from a user-defined point, line, or plane
- an estimate of the explicit central-difference timestep for each material;
- the validity of contact surfaces or slidelines, that is, whether two surfaces overlap at any point; and
- the distance between two surfaces.

Since it is relatively easy to add a new utility to *NUMBERS* , its capabilities should increase in the future. Utilities that may be added in the future include the calculation of element distortion parameters which would be useful for validating automatically generated finite element discretizations, determination of the surface area of side sets, and additional verification of contact surfaces. Although *NUMBERS* does not currently read any of the variables, except for the displacements, from an EXODUS file, the code is structured such that this capability could be easily added if needed. \raggedright \bibliographystyle 1520

6 Command Summary

6.1 Operational Commands (page 22)

\cmdsum MASS *nquad*, \optparam density PROPERTIES *nquad* \optparam density

calculates several mass properties of the body.

\cmdsum LOCATE {NODES|ELEMENTS} *locate_option*

locates NODES or ELEMENTS that are a specified distance from a user-defined point, line, or plane.

LOCATE	NODES|ELEMENTS	POINT	x_0		y_0	z_0
			distance		toler	
LOCATE	NODES|ELEMENTS	LINE	x_0		y_0	z_0
x_1	y_1	z_1	distance		toler	
LOCATE	NODES|ELEMENTS	PLANE	x_0		y_0	z_0
i	j	k	distance		toler	

\cmdsum CAVITY $sset_1$, $sset_2$, ..., $sset_n$, \optparam CENTER, \optparam xcen, \optparam ycen, \optparam zcen

calculates the volume of a cavity defined by the element sidesets $sset_1$ to $sset_n$.

\cmdsum LIMITS \optparam ALLTIMES

determines the minimum, maximum, and range of the X, Y, and Z coordinates for each of the selected material blocks.

\cmdsum OVERLAP $flag_m$, $flag_s$

determines whether any of the nodes on the surface defined by the sideset $flag_s$ penetrate the element faces on the surface defined by the sideset $flag_m$.

\cmdsum TIMESTEP *wave_speed* \optparam damping_fraction

estimates the maximum stable time step for an explicit transient dynamics finite element program.

\cmdsum GAP $flag_m$, $flag_s$, $d_{\max}$, \optparam DISTANCE|NORMAL

calculates the distance between nodes on the surfaces defined by the sideset flags $flag_m$ and $flag_s$.

6.2 Parameter Setting Commands (page 25)

\cmdsum AXISYMMETRIC

the two-dimensional body is axisymmetric.

\cmdsum PLANAR or PLANE

the two-dimensional body described is planar or plane strain.

\cmdsum EXODUS \optparam ON|OFF

indicates that the operational commands should calculate values based on the undeformed geometry (off), or for every time step (on).

\cmdsum DENSITY

prompts the user for the density of each of the element blocks

\cmdsum SORT *sort_field* \optparam sort_order

orders the output of a LOCATE command according to *sort_field* and *sort_order*.

\cmdsum SELECT *option*

selects the materials or material blocks for the operational commands.

\cmdsum SELECT BLOCKS \optparam ALL, $block\_id_1$, $block\_id_2$, ...

selects the element blocks.

\cmdsum SELECT MATERIALS \optparam ALL, $material\_id_1$, $material\_id_2$, ...

selects the element blocks with the material number corresponding to *material_id*.

\cmdsum TMIN *tmin*

sets the minimum selected time to *tmin*.

\cmdsum TMAX *tmax*

sets the maximum selected time to *tmax*.

\cmdsum DELTIME *delt*

sets the selected time interval to *delt*.

\cmdsum TIMES *tmin* TO *tmax* BY *delt*

sets *tmin*, *tmax*, and *delt* with a single command.

6.3 Database Display Commands (page 27)

\cmdsum LIST *option*

displays the database information specified by *option*.
LIST {VARS|VARIABLES} LIST {BLOCKS|MATERIALS} LIST {NSETS|NODESETS} LIST {SSETS|SIDESETS} LIST STEPS LIST TIMES

\cmdsum HELP *command*

displays information about a program command.

6.4 Miscellaneous Commands (page 28)

\cmdsum ECHO \optparam ON|OFF

controls the output of results to the terminal.

\cmdsum PRINT \optparam ON|OFF

controls the output of results to the output file.

\cmdsum COMMENT \optparam PAGE, *ncom*

prompts the user for *ncom* comment lines which are written to the output file.

\cmdsum EXIT

exits the program and closes all files.

7 Site Supplements

NOTE: This information is no longer valid or applicable (2/1/97)

7.1 VAX VMS

The command to execute *NUMBERS* on the \SNL \es VAXCluster running the VMS operating system is:

numbers *database* \optparam user_input

database is the filename of the input EXODUS database. A prompt appears if *database* is omitted.

If *user_input* is given, the user input is read from this file. Otherwise it is read from SYS$INPUT (the terminal keyboard). User output is directed to SYS$OUTPUT (the terminal).

NUMBERS operates in either interactive or batch modes.

7.2 CRAY CTSS

To execute *NUMBERS* , the user must have selected the Engineering Analysis Department's acclib library and be running ccl.

The command to execute *NUMBERS* on CTSS is:

numbers \param database i=\param input o=\param output

database is the filename of the input EXODUS database. The default is tape9.

User input is read from *input*, which defaults to tty (the terminal). User output is directed to *output*, which defaults to tty (the terminal).

7.3 Execution Files

The table below summarizes the files used by *NUMBERS* .

Description	Unit Number	Type	File Format
User input	standard input	input	Section [Ref: c:commands]
User output	standard output	output	ASCII
Output file	7	output	ASCII
EXODUS database	9	input	Appendix [Ref: a:exodus]

All files must be connected to the appropriate unit before *NUMBERS* is run. Each file (except standard input and output) is opened with the name retrieved by the EXNAME routine of the SUPES library [SUPES].

7.4 Special Software

NUMBERS is written in ANSI FORTRAN-77 [F77] with the exception of the following system-dependent features:

- the VAX VMS help facility and
- the OPEN options for the files.

NUMBERS uses the following software package:

- the SUPES package [SUPES] which includes dynamic memory allocation, a free-field reader, and FORTRAN extensions.

8 The EXODUS Database Format

The following code segment reads an EXODUS database. The first segment is the GENESIS database format.

```
C   --Open the EXODUS database file
      NDB = 9
      OPEN (UNIT=NDB, ..., STATUS='OLD', FORM='UNFORMATTED')
C   --Read the title
      READ (NDB) TITLE
C      --TITLE - the title of the database (CHARACTER*80)
C   --Read the database sizing parameters
      READ (NDB) NUMNP, NDIM, NUMEL, NELBLK,
     &   NUMNPS, LNPSNL, NUMESS, LESSEL, LESSNL, NVERSN
C      --NUMNP - the number of nodes
C      --NDIM - the number of coordinates per node
C      --NUMEL - the number of elements
C      --NELBLK - the number of element blocks
C      --NUMNPS - the number of node sets
C      --LNPSNL - the length of the node sets node list
C      --NUMESS - the number of side sets
C      --LESSEL - the length of the side sets element list
C      --LESSNL - the length of the side sets node list
C      --NVERSN - the file format version number
C   --Read the nodal coordinates
      READ (NDB) ((CORD(INP,I), INP=1,NUMNP), I=1,NDIM)
C   --Read the element order map (each element must be listed once)
      READ (NDB) (MAPEL(IEL), IEL=1,NUMEL)
C   --Read the element blocks
      DO 100 IEB = 1, NELBLK
C      --Read the sizing parameters for this element block
         READ (NDB) IDELB, NUMELB, NUMLNK, NATRIB
C         --IDELB - the element block identification (must be unique)
C         --NUMELB - the number of elements in this block
C         --   (the sum of NUMELB for all blocks must equal NUMEL)
C         --NUMLNK - the number of nodes defining the connectivity
C         --   for an element in this block
C         --NATRIB - the number of element attributes for an element
C         --   in this block
C      --Read the connectivity for all elements in this block
         READ (NDB) ((LINK(J,I), J=1,NUMLNK, I=1,NUMELB)
C      --Read the attributes for all elements in this block
         READ (NDB) ((ATRIB(J,I), J=1,NATRIB, I=1,NUMELB)
  100 CONTINUE
C   --Read the node sets
      READ (NDB) (IDNPS(I), I=1,NUMNPS)
C      --IDNPS - the ID of each node set
      READ (NDB) (NNNPS(I), I=1,NUMNPS)
C      --NNNPS - the number of nodes in each node set
      READ (NDB) (IXNNPS(I), I=1,NUMNPS)
C      --IXNNPS - the index of the first node in each node set
C      --   (in LTNNPS and FACNPS)
      READ (NDB) (LTNNPS(I), I=1,LNPSNL)
```

```
C      --LTNNPS - the nodes in all the node sets
       READ (NDB) (FACNPS(I), I=1,LNPSNL)
C      --FACNPS - the factor for each node in LTNNPS
C   --Read the side sets
       READ (NDB) (IDESS(I), I=1,NUMESS)
C      --IDESS - the ID of each side set
       READ (NDB) (NEESS(I), I=1,NUMESS)
C      --NEESS - the number of elements in each side set
       READ (NDB) (NNESS(I), I=1,NUMESS)
C      --NNESS - the number of nodes in each side set
       READ (NDB) (IXEESS(I), I=1,NUMESS)
C      --IXEESS - the index of the first element in each side set
C      --   (in LTEESS)
       READ (NDB) (IXNESS(I), I=1,NUMESS)
C      --IXNESS - the index of the first node in each side set
C      --   (in LTNESS and FACESS)
       READ (NDB) (LTEESS(I), I=1,LESSEL)
C      --LTEESS - the elements in all the side sets
       READ (NDB) (LTNESS(I), I=1,LESSNL)
C      --LTNESS - the nodes in all the side sets
       READ (NDB) (FACESS(I), I=1,LESSNL)
C      --FACESS - the factor for each node in LTNESS
```

A valid GENESIS database may end at this point or at any point until the number of variables is read.

```
C   --Read the QA header information
       READ (NDB, END=900) NQAREC
C      --NQAREC - the number of QA records (must be at least 1)
       DO 110 IQA = 1, MAX(1,NQAREC)
          READ (NDB) (QATITL(I,IQA), I=1,4)
C         --QATITL - the QA title records; each record contains:
C         --   1) analysis code name (CHARACTER*8)
C         --   2) analysis code qa descriptor (CHARACTER*8)
C         --   3) analysis date (CHARACTER*8)
C         --   4) analysis time (CHARACTER*8)
  110  CONTINUE
C   --Read the optional header text
       READ (NDB, END=900) NINFO
C      --NINFO - the number of information records
       DO 120 I = 1, NINFO
          READ (NDB) INFO(I)
C         --INFO - extra information records (optional) that contain
C         --   any supportive documentation that the analysis code
C         --   developer wishes (CHARACTER*80)
  120  CONTINUE
C   --Read the coordinate names
       READ (NDB, END=900) (NAMECO(I), I=1,NDIM)
C      --NAMECO - the coordinate names (CHARACTER*8)
C   --Read the element type names
       READ (NDB, END=900) (NAMELB(I), I=1,NELBLK)
C      --NAMELB - the element type names (CHARACTER*8)
```

The GENESIS section of the database ends at this point.

```
C   --Read the history, global, nodal, and element variable information
      READ (NDB, END=900) NVARHI, NVARGL, NVARNP, NVAREL
C      --NVARHI - the number of history variables
C      --NVARGL - the number of global variables
C      --NVARNP - the number of nodal variables
C      --NVAREL - the number of element variables
      READ (NDB)
     &   (NAMEHV(I), I=1,NVARHI),
     &   (NAMEGV(I), I=1,NVARGL),
     &   (NAMENV(I), I=1,NVARNP),
     &   (NAMEEV(I), I=1,NVAREL)
C      --NAMEHI - the history variable names (CHARACTER*8)
C      --NAMEGV - the global variable names (CHARACTER*8)
C      --NAMENV - the nodal variable names (CHARACTER*8)
C      --NAMEEV - the element variable names (CHARACTER*8)
      READ (NDB) ((ISEVOK(I,J), I=1,NVAREL), J=1,NELBLK)
C      --ISEVOK - the name truth table for the element blocks;
C      --   ISEVOK(i,j) refers to variable i of element block j;
C      --   the value is 0 if and only if data will NOT be output for
C      --   variable i for element block j (otherwise the value is 1)
C   --Read the time steps
  130 CONTINUE
        READ (NDB, END=900) TIME, HISTFL
C         --TIME - the time step value
C         --HISTFL - the time step type flag:
C         --   0.0 for all variables output ("whole" time step) else
C         --   only history variables output ("history-only" time step)
C
        READ (NDB) (VALHV(IVAR), IVAR=1,NVARGL)
C         --VALHV - the history values for the current time step
        IF (HISTFL .EQ. 0.0) THEN
           READ (NDB) (VALGV(IVAR), IVAR=1,NVARGL)
C            --VALGV - the global values for the current time step
           DO 140 IVAR = 1, NVARNP
              READ (NDB) (VALNV(INP,IVAR), INP=1,NUMNP)
C               --VALNV - the nodal variables at each node
C               --   for the current time step
  140       CONTINUE
           DO 160 IBLK = 1, NELBLK
              DO 150 IVAR = 1, NVAREL
                 IF (ISEVOK(IVAR,IBLK) .NE. 0) THEN
                    READ (NDB) (VALEV(IEL,IVAR,IBLK),
    &                  IEL=1,NUMELB(IBLK))
C                     --VALEV - the element variables at each element
C                     --   for the current time step
                 END IF
  150         CONTINUE
  160       CONTINUE
        END IF
C      --Handle time step data
      GOTO 130
  900 CONTINUE
C   --Handle end of file on database
```

9 Listing of Heapsort Routine

\listing [-]indexx.for

\graphicsfigure 5nullIllustration of the Angles θ and ϕ for the Locate Point Utility

Figure 3 Illustration of Cavity Volume Determination for a Two-Dimensional Cavity

Figure 4 Illustration of Contact Surface Overlap due to Discretization Mismatch on Curved Contact Surfaces

Figure 5: Numbering of Face Nodes on an Eight-Node Hexahedral Element

Hexahedron	Face					
	1	*2*	*3*	*4*	*5*	*6*
A	1	6	6	5	4	1
B	2	7	5	1	3	5
C	3	3	8	4	7	6
D	4	2	7	8	8	2

\hfil

Figure 6: Numbering of Face Nodes on an Four-Node Quadrilateral Element

Quadrilateral	Face			
	1	*2*	*3*	*4*
A	1	2	3	4
B	2	3	4	1

Table 1Numbering of Face Nodes on an Eight-Node Hexahedral Element and a Four-Node Quadrilateral Element

\graphicsfigure 5nullIllustration of Closeness Measures Used in the Gap Utility

Figure 7 Finite Element Model for *NUMBERS* Examples

```
              *** NUMBERS Experimental Version 0.20 ***
                          Revised 00/00/00
                 A GENESIS/EXODUS DATABASE INFORMATION PROGRAM
                          Run on 00/00/00 at 00:00:00
Genesis: XX:[XXXXXXX.XXXXXXX.XXXXXXX.XXXX]2DP.GEN;1
Title:   2D MESH - NUMBERS EXAMPLE PROBLEM
Number of Nodes:      7755
Number of Elements:   6825
Two-Dimensional Mesh, Axisymmetric
Mass Properties Calculation
4-Point Quadrature, Time =  0.000E+00
Material    Density        Volume          Mass        Label
   1     7.324E-04     1.110E+04     8.128E+00     OUTER_SHELL
  11     3.297E-03     2.831E+03     9.334E+00     MASS_OUTER_SHELL
   3     1.035E-05     1.320E+05     1.366E+00     FOAM
   2     7.324E-04     6.035E+03     4.420E+00     INNER_SHELL
  22     2.901E-03     1.419E+03     4.115E+00     MASS_INNER_SHELL
   7     1.603E-05     2.998E+04     4.806E-01     CARGO_FOAM
   8     7.324E-04     2.824E+02     2.068E-01     PUNCH
   9     7.324E-04     4.879E+02     3.573E-01     THIN_SHEET
  10     7.781E-04     2.827E+04     2.200E+01     CARGO_MASS
                        ----------     ----------
            Total:       2.124E+05      5.041E+01
MASS PROPERTIES--- (Inertias at centroid)
 Xc =  0.000E+00   Yc = -3.682E+01   Zc =  0.000E+00
Ixx =  4.478E+04  Iyy =  6.133E+04  Izz =  4.478E+04
Mat  Minimum   Elem   Maximum   Elem   Average   Num
Num    Area    Num      Area    Num      Area    Elem
 1   1.17E-02   927   2.40E-01  1431   3.16E-02  1444
11   2.34E-01  1451   2.34E-01  1471   2.34E-01    40
 3   1.85E-01  3392   6.86E-01  3408   3.92E-01  1926
 2   7.15E-03  4011   1.02E-01  4822   2.15E-02  1432
22   1.56E-01  4845   1.56E-01  4849   1.56E-01    40
 7   5.64E-02  5885   4.67E-01  4958   3.21E-01  1008
 8   1.15E-02  6371   1.68E-01  5922   4.91E-02   611
 9   2.23E-02  6592   2.52E-02  6545   2.48E-02   174
10   2.00E+00  6765   2.00E+00  6734   2.00E+00   150
```

Figure 8 Output for Mass Properties Example

```
                  *** NUMBERS Experimental Version 0.20 ***
                              Revised 00/00/00
                 A GENESIS/EXODUS DATABASE INFORMATION PROGRAM
                           Run on 00/00/00 at 00:00:00
    Genesis: XX:[XXXXXXX.XXXXXXX.XXXXXXX.XXXX]2DP.GEN;1
    Title:   2D MESH - NUMBERS EXAMPLE PROBLEM
    Number of Nodes:      7755
    Number of Elements:   6825
    Two-Dimensional Mesh, Axisymmetric
Checking Master Surface   100 Versus Slave Surface   999
No nodes found penetrating master surface.
Checking Master Surface   200 Versus Slave Surface   300
No nodes found penetrating master surface.
   26 Nodes on an element face.
Checking Master Surface   400 Versus Slave Surface   500
No nodes found penetrating master surface.
   54 Nodes on an element face.
Checking Master Surface   600 Versus Slave Surface   700
No nodes found penetrating master surface.
    1 Nodes on an element face.
Checking Master Surface   900 Versus Slave Surface  1000
No nodes found penetrating master surface.
   31 Nodes on an element face.
```

Figure 9 Output for Overlap Utility Example–No Penetration

```
Checking Master Surface   100 Versus Slave Surface   200
Node:    26 Element:     3      3      4     15     14
Node:    27 Element:     4      4      5     16     15
Node:    28 Element:     5      5      6     17     16
Node:    29 Element:     7      7      8     19     18
Node:    30 Element:     8      8      9     20     19
Warning -    5 Nodes penetrate master surface.
```

Figure 10 Output for Overlap Utility Example–Penetration

```
Locating all Elements at a distance  9.608E+01 plus/minus  1.000E-03
from the POINT
(  0.000E+00  2.420E+01)
Sorted on field THETA
 Elements    X         Y         DISTANCE     THETA
        1    0.0780  -71.8781   9.608E+01  -8.995E+01
        2    0.2344  -71.8778   9.608E+01  -8.986E+01
        3    0.3915  -71.8773   9.608E+01  -8.977E+01
        4    0.5491  -71.8765   9.608E+01  -8.967E+01
        5    0.7074  -71.8755   9.608E+01  -8.958E+01
        6    0.8663  -71.8742   9.608E+01  -8.948E+01
        7    1.0259  -71.8726   9.608E+01  -8.939E+01
        8    1.1861  -71.8708   9.608E+01  -8.929E+01
        9    1.3469  -71.8687   9.608E+01  -8.920E+01
       10    1.5083  -71.8663   9.608E+01  -8.910E+01
```

```
----- lines 11 to 189 have been omitted
       190   42.5913  -61.9219   9.608E+01  -6.369E+01
       191   42.8891  -61.7739   9.608E+01  -6.349E+01
       192   43.1877  -61.6243   9.608E+01  -6.329E+01
       193   43.4869  -61.4731   9.608E+01  -6.309E+01
       194   43.7868  -61.3202   9.608E+01  -6.289E+01
       195   44.0873  -61.1657   9.608E+01  -6.269E+01
       196   44.3885  -61.0094   9.608E+01  -6.248E+01
       197   44.6904  -60.8515   9.608E+01  -6.228E+01
       198   44.9929  -60.6919   9.608E+01  -6.208E+01
       199   45.2960  -60.5305   9.608E+01  -6.187E+01
       200   45.5996  -60.3675   9.608E+01  -6.167E+01
```

Figure 11 Output for Locate Utility Example–No Penetration

```
     Material        X             Y
         1      0.00000E+00  -7.19250E+01  Minimum
                5.05000E+01  -2.50000E+01  Maximum
                5.05000E+01   4.69250E+01  Range
        11      4.78750E+01  -2.50000E+01  Minimum
                4.82500E+01   0.00000E+00  Maximum
                3.75000E-01   2.50000E+01  Range
         3      0.00000E+00  -7.15000E+01  Minimum
                4.78750E+01  -3.50000E+01  Maximum
                4.78750E+01   3.65000E+01  Range
         2      0.00000E+00  -5.95500E+01  Minimum
                3.86250E+01  -2.50000E+01  Maximum
                3.86250E+01   3.45500E+01  Range
        22      3.60000E+01  -2.50000E+01  Minimum
                3.62500E+01   0.00000E+00  Maximum
                2.50000E-01   2.50000E+01  Range
         7      0.00000E+00  -5.93000E+01  Minimum
                3.59484E+01  -4.71200E+01  Maximum
                3.59484E+01   1.21800E+01  Range
         8      0.00000E+00  -8.19300E+01  Minimum
                3.00000E+00  -7.19300E+01  Maximum
                3.00000E+00   1.00000E+01  Range
         9      0.00000E+00  -4.71200E+01  Minimum
                3.60000E+01  -4.70000E+01  Maximum
                3.60000E+01   1.19999E-01  Range
        10      0.00000E+00  -4.70000E+01  Minimum
                3.00000E+01  -3.70000E+01  Maximum
                3.00000E+01   1.00000E+01  Range
     Limits for Total Body:
                0.00000E+00  -8.19300E+01  Minimum
                5.05000E+01   0.00000E+00  Maximum
                5.05000E+01   8.19300E+01  Range
```

Figure 12 Output for Limits Utility Example

```
     Time Step Estimates:
```

```
   (Damping = 0.06)
 Mat  Wave Speed  Element     Time Step    Steps/ms
  1   2.2730E+05      601     3.328E-07        3005    OUTER_SHELL
 11   1.0710E+05     1449    1.631E-06         613    MASS_OUTER_SHELL
  3   2.0030E+04     3304     1.478E-05          67    FOAM
  2   2.2730E+05     4011     2.273E-07        4399    INNER_SHELL
 22   1.1420E+05     4845    1.026E-06         974    MASS_INNER_SHELL
  7   2.7310E+04     5885     5.804E-06         172    CARGO_FOAM
  8   2.2730E+05     6371     3.141E-07        3183    PUNCH
  9   2.2730E+05     6600     2.456E-07        4072    THIN_SHEET
 10   1.5250E+05     6795     5.524E-06         181    CARGO_MASS
 Minimum Time Step in Material 2
      2.2730E+05     4011     2.273E-07        4399    INNER_SHELL
```

Figure 13 Output for Timestep Estimation Utility Example