

SANDIA REPORT

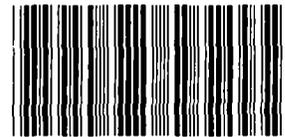
SAND92-2141 • UC-705
Unlimited Release
Printed May 1993

MICROFICHE

A General-Purpose Contact Detection Algorithm for Nonlinear Structural Analysis Codes

M. W. Heinstein, S. W. Attaway, J. W. Swegle, F. J. Mello

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789



8573079

SANDIA NATIONAL
LABORATORIES
TECHNICAL LIBRARY

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, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents 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
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

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

NTIS price codes
Printed copy: A05
Microfiche copy: A01

A General-Purpose Contact Detection Algorithm for Nonlinear Structural Analysis Codes*

M.W. Heinstein

Engineering Mechanics and Material Modeling Department

S.W. Attaway

Computational Mechanics and Visualization Department

J.W. Swegle

Experimental Impact Physics Department

F.J. Mello

Solid and Structural Mechanics Department

Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

A new contact detection algorithm has been developed to address difficulties associated with the numerical simulation of contact in nonlinear finite element structural analysis codes. Problems including accurate and efficient detection of contact for self-contacting surfaces, tearing and eroding surfaces, and multi-body impact are addressed. The proposed algorithm is portable between dynamic and quasi-static codes and can efficiently model contact between a variety of finite element types including shells, bricks, beams and particles. The algorithm is composed of (1) a location strategy that uses a global search to decide which slave nodes are in proximity to a master surface and (2) an accurate detailed contact check that uses the projected motions of both master surface and slave node. In this report, currently used contact detection algorithms and their associated difficulties are discussed. Then the proposed algorithm and how it addresses these problems is described. Finally, the capability of the new algorithm is illustrated with several example problems.

* This work performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract DE-AC04-76DP00789

Nomenclature

Δt_c	incremental time to contact
ξ, η	local surface coordinates of contact point
a	\hat{i} component of surface normal
B^i	bucket containing node i
b	\hat{j} component of surface normal
b_s	bucket size
C_{3D}	number of slave node-master nodes comparisons made in 3D bucket sorting
c	\hat{k} component of surface normal
\vec{d}_{ms}	vector from master surface node to slave node
dt	current time step
I_x, I_y, I_z	Index vector for the x , y , and z coordinate respectively
$ibox_{min}$	slice of data containing minimum x -coordinate of master surface capture box
$ibox_{max}$	slice of data containing maximum x -coordinate of master surface capture box
$jbox_{min}$	slice of data containing minimum y -coordinate of master surface capture box
$jbox_{max}$	slice of data containing maximum y -coordinate of master surface capture box
$kbox_{min}$	slice of data containing minimum z -coordinate of master surface capture box
$kbox_{max}$	slice of data containing maximum z -coordinate of master surface capture box
$(i_x)_{min}$	pointer into the Index vector corresponding to the first slave node inside a master surface capture box along the x -coordinate
$(i_x)_{max}$	pointer into the Index vector corresponding to the last slave node inside a master surface capture box along the x -coordinate
$(i_y)_{min}$	pointer into the Index vector corresponding to the first slave node inside a master surface capture box along the y -coordinate
$(i_y)_{max}$	pointer into the Index vector corresponding to the last slave node inside a master surface capture box along the y -coordinate
$(i_z)_{min}$	pointer into the Index vector corresponding to the first slave node inside a master surface capture box along the z -coordinate
$(i_z)_{max}$	pointer into the Index vector corresponding to the last slave node inside a master surface capture box along the z -coordinate
$lbox$	number of slave nodes in each bucket
m	number of master nodes

m_s	number of master surfaces
\hat{m}	unit normal of a master surface
\hat{m}_i	unit normal of master surface i
\hat{m}_p	unit normal of master surface at end of previous time step
\vec{N}_m	surface normal of master surface
n	number of slave nodes
\hat{n}	unit normal of the slave surface at a slave node
n_b	total number of buckets
n_{box}	bucket id for each slave node
$ndsort$	list of slave nodes sorted by ascending bucket id
$npoint$	pointer into vector $ndsort$ giving the starting location of the slave nodes in each bucket
\hat{p}_t	unit push-back direction at time t
p	penetration of slave node into master surface
R_x, R_y, R_z	Rank vector for the x, y, and z coordinate respectively
S_x, S_y, S_z	number of slices required along the x, y and z coordinates respectively
S_x^i, S_y^i, S_z^i	slice along x, y and z coordinate containing node i
t	current time
\vec{V}	velocity vector
\vec{V}_s	velocity of slave node
\vec{V}_m	velocity of master surface at contact point
v_x, v_y, v_z	x, y and z component of velocity
$(v_x)_{max}$	maximum x-component of velocity
$(v_y)_{max}$	maximum y-component of velocity
$(v_z)_{max}$	maximum z-component of velocity
x_i, y_i, z_i	x, y, and z coordinate of master surface nodal point i
xc_{min}	minimum x-coordinate of master surface capture box
xc_{max}	maximum x-coordinate of master surface capture box
yc_{min}	minimum y-coordinate of master surface capture box
yc_{max}	maximum y-coordinate of master surface capture box
zc_{min}	minimum z-coordinate of master surface capture box
zc_{max}	maximum z-coordinate of master surface capture box

Table of Contents

1. Introduction	7
2. Survey of Contact Detection Algorithms and Motivation for Current Work	8
2.1 Neighborhood Identification.....	9
2.1.1 Surface Side-Set Pairing	10
2.1.2 Surface Tracking.....	10
2.1.3 Bucket Searching	11
2.1.4 Pinball Contact.....	12
2.2 Detailed Contact Check	13
2.2.1 Ideal Contact Determination.....	13
2.2.2 Pushback inaccuracies	14
2.2.3 Overdetermined Contact	14
2.2.4 Undetermined Contact.....	16
2.3 Motivation for Current Work.....	17
3. New Contact Detection Algorithm	19
3.1 New Neighborhood Identification Strategy	19
3.1.1 Algorithm for vector architecture	19
3.1.2 Algorithm for parallel architecture	24
3.2 New Detailed Contact Check.....	29
3.2.1 Velocity Based Contact Check	29
3.2.2 Static Contact Check.....	32
3.3 Summary of New Contact Detection Algorithm	33
4. Surface Definition Algorithm	35
5. Example Problems	37
5.1 Contact of Elastic Blocks.....	37
5.2 Contact Chatter under High Normal Loads: Pressure Loading of Two Elastic Bodies.....	39
5.3 Self-Contacting Impact: Buckling of Shell-Like Structures.....	41
5.4 Automatic Contact Surface Redefinition: Cutting of a Steel Pipe.....	43
5.5 Multi-Body Impact: Elastic-Plastic Bar impacting Bricks	46
5.6 Large Sliding Contact: Elastic-Plastic Forging of a Copper Billet.....	48
6. Conclusions	50
7. References	51
A1. Appendix 1: Derivation of velocity based contact check	54
A2. Appendix 2: User instructions and example input files	60

List of Tables

1. Surface definition algorithm example.....	36
--	----

List of Figures

1. Master-slave relationship definition for contact enforcement	9
2. Surface tracking and side set pairing required for self-contacting structures.....	10
3. Detection of potential contacts for slave node 3 are influenced by bucket size	12
4. Calculation of the normal distance to contact point when contact is made.....	13
5. Calculation of the normal distance to contact point as point slides on surface	14
6. Resolving overdetermined contact by determining most opposed master surface.....	15
7. Determining contact point with consideration of slave node's movement.....	16
8. Resolving undetermined contact by identifying the closet point to the master surface	17
9. Slave node 2 tracking closest master node 14 results in a missed contact	18
10. Example of two blocks contacting each other	20
11. Bounding box and capture box for a moving master surface	21
12. Remaining penetration due to a partially enforced contact constraint.....	22
13. Bucket B^i defined by three slices of the data that contain node i	25
14. Master slave tracking using velocity and static contact check	29
15. Initial estimates for the local coordinates of a contact point	31
16. Push back direction for a concave surface based on minimum distance to master surface	32
17. Push back direction for convex surface based on previous master surface normal.....	33
18. Example mesh for surface definition algorithm	35
19. End-on impact of two blocks.....	37
20. Corner impact of two blocks.....	38
21. Two elastic bodies contacting under high normal load	39
22. Kinetic energy history and deformed shape (displacements magnified by 100x) using old and new contact detection algorithm	40
23. Finite element mesh of an elastic plate impacting an elastic-plastic can	41
24. Deformed meshes of the buckled can at various times.....	42
25. Finite element model for simulating the cutting of a 2 inch steel pipe.....	43
26. Pipe cutting simulation results at various times.....	44
27. Elastic-plastic bar impacting a stack of 17 elastic bricks	46
28. Multi-body impact simulation results at various times.....	47
29. Finite element mesh of a rivet forging process.....	48
30. Rivet forging of a copper billet.....	49
A1.1. Triangular master surface definition on a quadrilateral element face.....	54
A1.2. Initial estimates for the local coordinates of a contact point	58

1 Introduction

An increasingly important aspect of large-scale finite element structural simulations is the efficient and accurate determination of contact between deformable bodies. At Sandia National Laboratories, the PRONTO [1][2][3] transient dynamics codes and the SANTOS [4] and JAC [5][6] quasistatic codes have been used to solve a wide variety of problems involving contact and other nonlinearities. However in some cases, the range of their applicability could be increased by improving the efficiency and accuracy of contact detection. These improvements would be beneficial in problems involving: structures that buckle and fold onto themselves; structures that have materials that tear and create new surfaces; multiple body contact/impact; and structures that slide relatively large distances over other surfaces. This report deals with one part of the contact problem, namely the detection of contact, which is distinct from the procedure used to enforce contact constraints. In these codes, which use a master-slave approach to contact problems, contact detection includes identifying the time, location, and amount of slave node penetration through some portion of a master surface.

The benefits of reducing the time spent on contact algorithms can be significant. For iterative equation solvers, such as those used in the Sandia codes, inaccuracies in the detection of contact lead to an increase in the number of iterations required for convergence. These inaccuracies arise mainly from incorrectly determining the location of contact as a slave node slides across another surface. For large finite element simulations with large numbers of slave nodes and master surfaces, as much as 50 percent of the total CPU time is spent using currently available contact algorithms. Thus, improvements in the speed and efficiency of contact detection could significantly reduce the total computational cost. More importantly, these improvements are also expected to allow one to solve problems which cannot be solved using existing algorithms.

This report reviews the current contact detection techniques used in the Sandia structural analysis codes and outlines the difficulties associated with these algorithms. A new algorithm is proposed that circumvents these difficulties. The key points of the new algorithm are that it:

- i) uses a fast, memory-efficient global search to decide which slave nodes are in proximity to a master surface;
- ii) does a detailed contact check using projected movements of both the master surface and slave node to determine the location, magnitude and direction of slave node penetration of the master surface; and
- iii) automatically defines all surfaces given the mesh connectivity.

In Section 2 a short survey of the current contact detection algorithms and some difficulties associated with them is presented. Section 3 describes the proposed new contact detection algorithm, and in Section 4, an automatic surface definition algorithm is presented. Finally, in Section 5, example and verification problems using the new contact detection algorithm are discussed. Nomenclature used throughout the report is defined on the preceding pages. Two appendices are attached. In the first appendix, the velocity based contact check proposed in Section 3 is derived. The second appendix has a complete listing of all input files for the example problems presented in Section 5.

2 Survey of Contact Detection Algorithms and Motivation for Current Work

The contact detection algorithms described in this section are typical for many finite element codes, including the transient dynamic codes PRONTO [1][2][3], DYNA[7][8], and ABAQUS Explicit [9], and the quasistatic codes SANTOS [4], JAC [5][6] and NIKE [10][11]. Most of this section specifically reviews the contact detection algorithms used in the PRONTO, SANTOS, and JAC codes. Difficulties associated with the contact detection algorithms are presented as a way for motivating the current work. A recent survey of the contact-impact algorithm in the DYNA codes was reported in [12].

Contact detection algorithms of interest here define a set of nodes called slave nodes and a set of surface patches called master surfaces. A slave node is simply a nodal point on the surface of the mesh. A master surface is defined using the side of a finite element on the surface. The surface topology is then simply a set of nodal points on the surface connected by straight line segments, corresponding to linear surface elements in 2D and bi-linear, quadrilateral surface elements in 3D. The master surface patches need not have the same order of interpolation as the finite element side that they represent. A four-node quadrilateral finite element side, for example, might be subdivided into four triangular master surfaces with the central node position being the average of the four corner nodes [13].

Contact detection is accomplished by monitoring the displacements of the slave nodes throughout the calculation for possible penetration of a master surface. Following contact detection, a contact constraint is defined so that the slave node is “pushed back” to remain on the master surface. Based on this description, it is convenient to separate contact algorithms into a location phase and a restoration phase. The location phase consists of a neighborhood identification and a detailed contact check. The neighborhood identification matches a slave node to a set of master surfaces that it potentially could contact. The detailed contact check determines which of the candidate master surfaces is in contact with a slave node, the point of contact, the amount of penetration, and the direction of push-back. The point of contact, amount of penetration, and the direction of pushback define a contact constraint that is then enforced in the contact enforcement or restoration phase of the contact algorithm. This constraint is enforced in the following time step or possibly over several time steps.

Although this report does not deal directly with contact enforcement, several aspects of it will affect the contact detection algorithm. The first is that the contact constraint is not necessarily enforced exactly in one time step for transient dynamics analyses, or in one iteration for quasistatic analyses. In part, this is due to the necessity of determining contact based on an approximate or projected configuration. In many cases, an estimated amount of slave node penetration is too low. Consequently, in the following time step (or iteration) the slave node is still penetrating the master surface. It is also likely and even desirable that the contact constraint *intentionally* be enforced in several time steps to improve the stability of the nonlinear equations of motion (or in several iterations to improve the convergence of integrating the nonlinear equilibrium equations in the case of quasistatic codes). This implies that the location phase must allow for the possibility that the contact constraint is not exactly enforced.

Another aspect that will affect contact detection is that contact enforcement algorithms are based on a master-slave definition. For algorithms that require a strict master-slave treatment, the user must specify which surface is the master and which surface is the slave. This choice can have a significant effect on the resulting calculation and solution. For example, the coarser mesh should be designated as the master surface when the two contacting materials are the same, as shown in Figure 1a. If the master and slave surfaces are reversed as in Figure 1b, interpenetration that is not detected by the algorithm can result. The choice of master-slave roles is less clear when the materials of the contacting bodies are different. A more general contact enforcement approach requires that two passes be made through the contact detection algorithm, with master and slave surfaces exchanging roles. This two-pass approach, called a symmetric or partitioned contact algorithm, is more robust and often justifies the additional computational cost of the extra pass. The PRONTO codes also allow the user to specify a factor which partitions the master/slave roles of the contacting surfaces. For the contact detection algorithm, this implies that all nodes on the surfaces are slave nodes and all element faces on the surfaces are master surfaces.

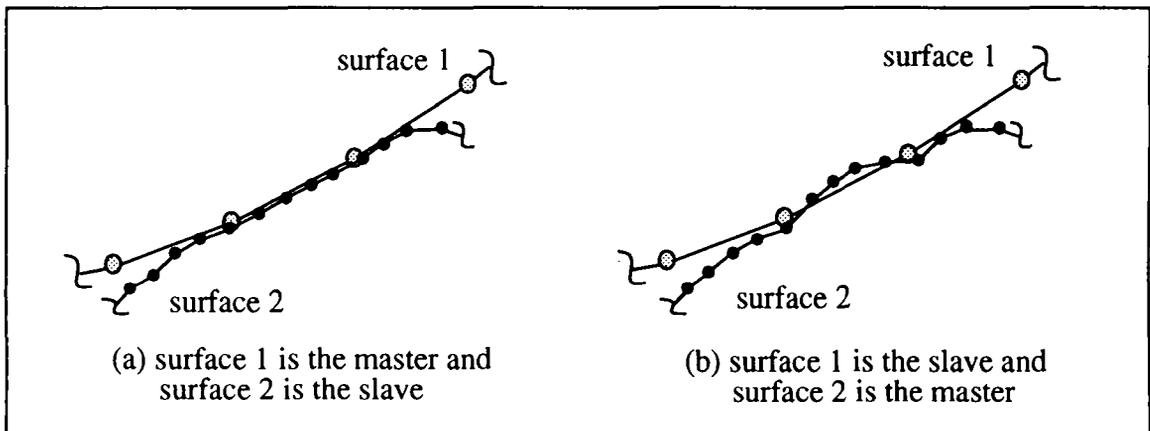


Figure 1. Master-slave relationship definition for contact enforcement.

2.1 Neighborhood Identification

Based on the description of contact detection, an algorithm called “neighborhood identification” is required to pair those slave nodes and master surfaces where potential contact is likely. The neighborhood identification phase is usually the most time consuming part of the contact detection algorithm. Obviously, the most robust approach would be to check every slave node against every master surface at every time step. Typically, the distances between each slave node and all master surface nodes are checked to find the closest master surface node. Master surfaces attached to this node are then considered as candidates for detailed contact checks. This exhaustive global searching approach requires nodal distance calculations on the order $n \times m$, where n is the number of slave nodes on the surface and m is the number of master nodes on the surface. Several algorithms have been proposed to speed up the neighborhood identification phase. These include surface side-set pairing, surface tracking, bucket searching, and pinball contact which are described below.

2.1.1 Surface Side-Set Pairing

One simple and widely used approach to speed up neighborhood identification is to define subsets of the surface and prescribe pairs of subsets that may be in contact. Using these contact pairs, a search may be restricted to only those slave nodes and master nodes which are included in the two subsets. All of the Sandia codes currently use surface side set pairing. A recent extension of this idea subdivides the contact side sets further and has been developed and implemented in DYNA3D [14]. This approach is effective when the contact surface pair has a very small number of slave nodes and master surfaces. For larger problems faster search techniques are needed.

2.1.2 Surface Tracking

To speed up the neighborhood identification further, some codes including PRONTO use a process called surface tracking [1,2,6,7]. Surface tracking requires the definition of contact surface pairs but does not require an exhaustive nxm search between all slave nodes and master nodes. The surface tracking algorithms currently in PRONTO and DYNA are based on two assumptions regarding the behavior of contacting surfaces. First, the spatially closest master surface node is assumed to be attached to the master surface that the slave node contacts. This assumption allows a very simple distance calculation to find what is called the tracked master surface node. Once the tracked master surface node is known, a detailed contact check for each master surface connected to the node can be made. The second assumption is that from one time step to another, the new tracked master surface node can be found near the currently-tracked node. This assumption allows the tracking to be updated by a *local* search for the nearest master surface node starting with the currently-tracked node and moving radially outward. To avoid searching for neighboring master surface nodes during the calculation, the tracking algorithm constructs a list of master surface neighbors at the start of the calculation and stores it for future reference. This requires a considerable amount of memory but is usually justified because a significant amount of computing time can be saved.

A surface tracking algorithm based on these assumptions cannot effectively model two classes of problems. The first is a structure buckling and folding onto itself as shown in Figure 2.

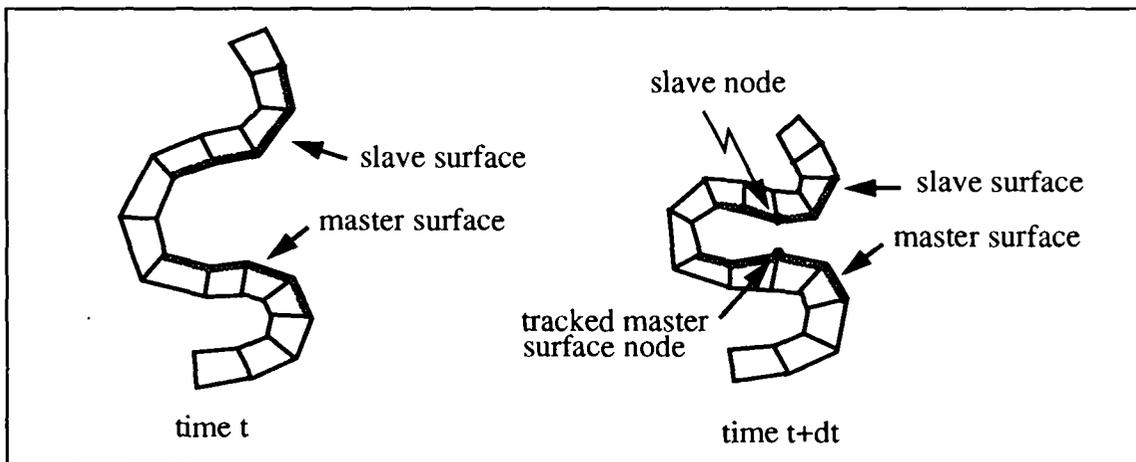


Figure 2. Surface tracking and side set pairing required for self-contacting structures

Since the portions of the surface that come into contact are not known *a priori*, the contact surface definition and pairing must be done intermittently throughout the analysis [15]. This extensive interaction by the user makes the solution of self contact difficult. The second class of problems involves structures with surfaces that tear or erode. Usually the newly created surface finds itself in contact with other parts of the structure. The fixed data structure required for surface tracking is an obstacle for redefining the surface and allowing further contact.

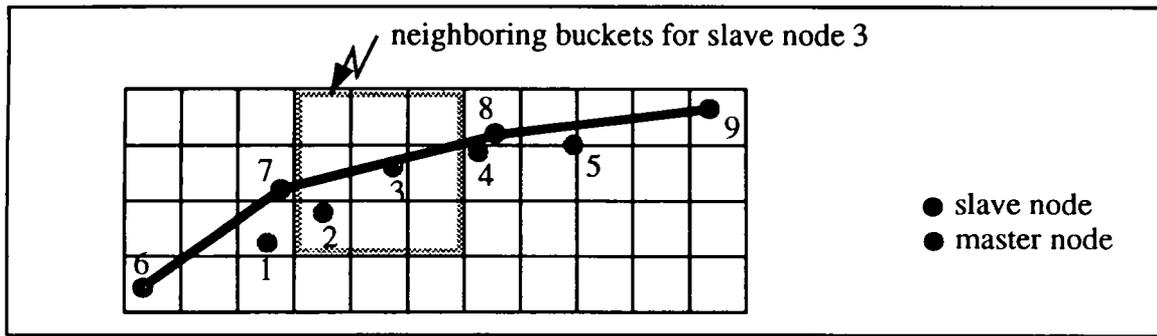
2.1.3 Bucket Searching

Recently several algorithms using a bucket search have been proposed for modelling self contacting surfaces [12][16]. These global search algorithms are referred to as bucket searches because the domain of the problem is broken down into buckets (or boxes) into which the slave nodes and master nodes are sorted. The domain of the problem is first subdivided into S_x slices along the x-coordinate, S_y slices along the y-coordinate, and S_z slices along the z-coordinate. The intersection of any three orthogonal slices defines a bucket. Potential interactions between nodes are determined by looping through the buckets and collecting the slave nodes and master nodes in neighboring buckets. Typically, a closest master node is determined for each slave node using a distance check. Assuming a uniform distribution of $n+m$ slave and master nodes throughout the domain, the total number of distance comparisons for a 3D problem (reported in [12]) is:

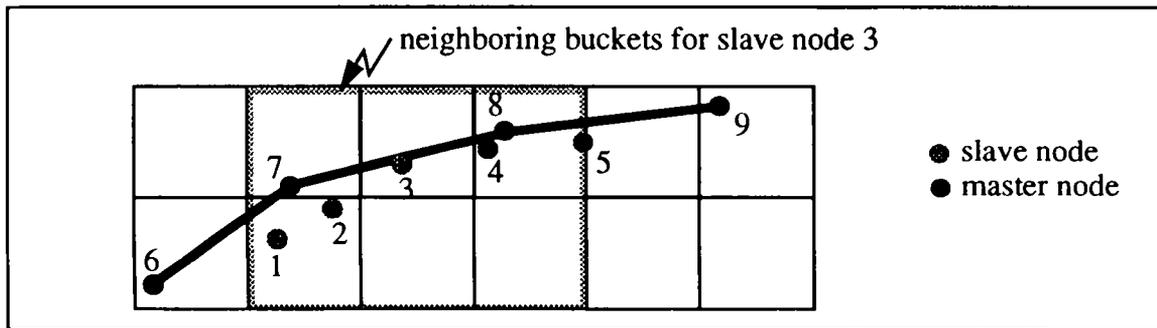
$$C_{3D} = (n + m) \left[\frac{27 (n + m)}{(S_x S_y S_z)} - 1 \right]$$

After the distance check, all master surfaces connected to the closest master node are then considered for potential contact with the slave node. This method for determining potential interactions implies that the bucket size should be chosen based on the largest master surface dimension to avoid missing a potential contact, as shown in Figure 3. The small bucket size in Figure 3a results in a missed contact because the master nodes 7 and 8 are not in the neighboring buckets of slave node 3. The larger bucket size in Figure 3b results in detecting the correct contact of slave node 3 with the master surface defined by master nodes 7 and 8. Based on this observation, a contact problem involving very dissimilar mesh sizes requires large buckets compared to many of the elements. Consequently, a large number of potential interactions may be found in each bucket.

A limitation of the bucket searching algorithm is the potential need for a large amount of memory. At first glance this algorithm appears to require $kS_x S_y S_z$ memory locations, where k is the maximum number of nodes likely to be found in a bucket. With some attention to memory management, these requirements can be significantly reduced for certain problems. For other problems the large amount of memory required remains an obstacle. If the problem domain expands greatly, either the number of buckets must increase (to maintain the same resolution) or the bucket size must increase (to maintain the same amount of memory). Either one adversely effects the bucket search efficiency. This dependency on the bucket size or, equivalently, on the geometry of the problem domain is considered as a limitation of current bucket searching techniques.



(a) a small bucket size results in a missed contact



(b) a large bucket size results in detecting correct contact

Figure 3. Detection of potential contacts for slave node 3 are influenced by bucket size

Even with these possible difficulties, the distinct advantage that bucket searching algorithms have is that they lend themselves to a parallel architecture, as discussed by Plimpton [20].

2.1.4 Pinball Contact

Another contact detection algorithm that uses a global search was proposed by Belytschko [17]. It is referred to as pinball contact since a circle is inscribed within each element on the surface of a two-dimensional mesh. (A sphere is inscribed within each element on the surface of a three-dimensional mesh). Potential contact is detected simply by overlap of any two spheres. The procedure vectorizes and therefore is suited to large-scale 3D computations where fast and efficient algorithms are imperative. To aid this overlap detection, a sorting or searching technique, such as bucket searching, may be necessary. After pinball overlap is detected, a more detailed contact check can be used to obtain a more accurate prediction of the contact constraint. The method was intended for problems where impact is the primary form of contact. Two dimensional [18] and three dimensional [17] problems have been successfully solved using the pinball algorithm. There is, however, a class of problems where the pinball algorithm does not work well. For problems where sliding contact and friction is encountered, the resolution of the mesh near the surface has a dramatic effect on the accurate prediction of contact forces. A mesh with varying resolution at the surface gives a poor prediction of contact pressures and forces. This is a result of modelling a geometrically flat surface with a series of inscribed circles and is a limitation of the pinball contact algorithm.

2.2 Detailed Contact Check

After gathering a list of potential interactions using any one of the searching techniques described above, a detailed contact check is done for each slave node - master surface pair. The following section presents what is typically done in a detailed contact check. This check determines: (1) which of the candidate master surfaces is in contact with the slave node, (2) the point of contact, (3) the amount of penetration, and (4) the direction the slave node should be pushed back. This overview is presented using 2D surfaces composed of lines for simplicity, but the theory applies equally to 3D surfaces composed of bilinear patches.

2.2.1 Ideal Contact Determination

The ideal condition for determining contact is shown in Figure 4. The slave node is unambiguously outside a master surface at time t and inside the same surface at time $t+dt$. At time $t+dt$, the distance, p , from the slave node to the master surface is calculated by projecting the vector \vec{d}_{ms} onto the surface normal \hat{m}_1 :

$$p = |\vec{d}_{ms} \cdot \hat{m}_1| \quad (1)$$

The vector \vec{d}_{ms} is defined, as shown in Figure 4, from the master surface node i to the slave node at time $t+dt$:

$$\vec{d}_{ms} = [x_s(t+dt) - x_i(t+dt)]\hat{i} + [y_s(t+dt) - y_i(t+dt)]\hat{j} \quad (2)$$

where $(x_s(t+dt), y_s(t+dt))$ and $(x_i(t+dt), y_i(t+dt))$ are the coordinates of the slave node and master surface node i at time $t+dt$, respectively. This normal distance, p , is called the amount of penetration and the unit vector $\hat{p}_t = \hat{m}_1$ is called the pushback direction, where \hat{p}_t is defined to pass through the slave node at time $t+dt$. The contact point, x , is determined by intersecting the master surface with the pushback direction:

$$(x_c, y_c) = (x_s, y_s) + (p(\hat{p}_t \cdot \hat{i}), p(\hat{p}_t \cdot \hat{j})) \quad (3)$$

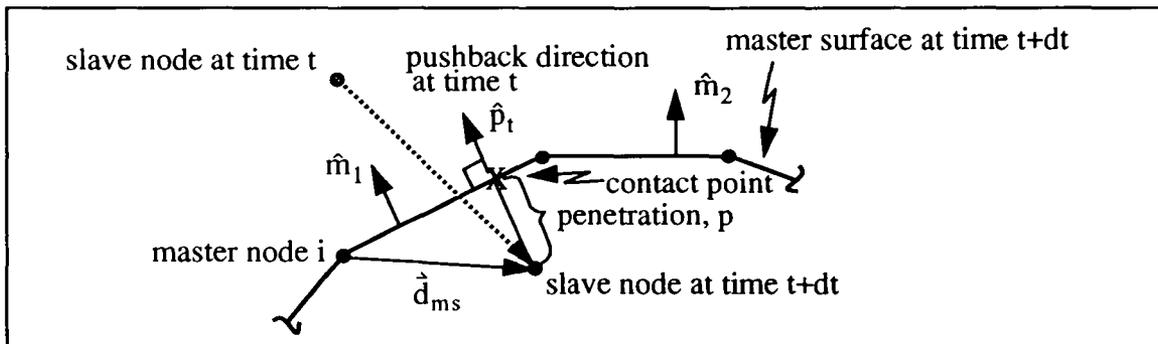


Figure 4. Calculation of the normal distance to contact point when contact is made

2.2.2 Pushback inaccuracies

Unfortunately, there are many ambiguous cases of contact that arise because of the discretization of the surface. The surface is discretized as a collection of C_0 continuous finite element sides in 2D (or faces in 3D). C_0 continuity in the geometric interpolation results in a surface normal that is not continuous. This can have adverse effects on the accuracy of the contact enforcement. Consider, for example, the case where a slave node is sliding across the surface shown in Figure 5. One must keep in mind that the enforcement of the contact constraint is not exact in one time step and could be enforced over several time steps. It is possible then, that the slave node be penetrating the master surface at time $t+dt$ (a time step dt after contact was detected). As the slave node slides from being in contact with master surface 1 to being in contact with master surface 2, the pushback direction is changing to coincide with the normal of master surface 2. This change in the pushback direction artificially introduces added slave node motion along its trajectory. This is a direct result of the inexact enforcement of the contact constraint in one time step. In calculations where friction is modelled, this velocity can add noise to the solution.

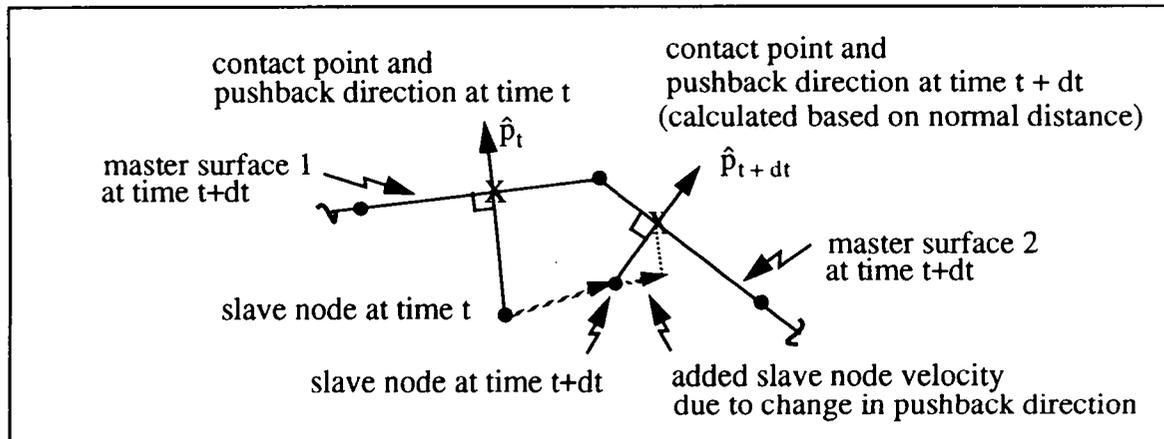
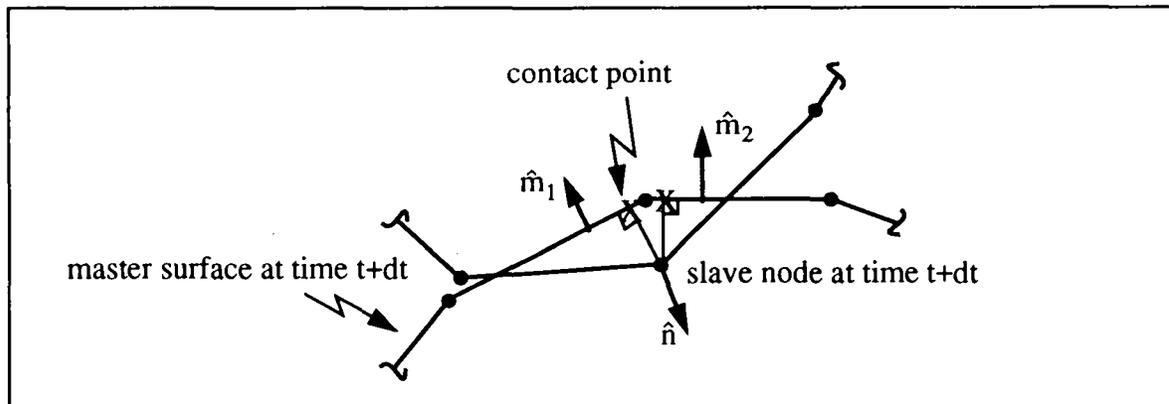


Figure 5. Calculation of the normal distance to contact point as point slides on surface

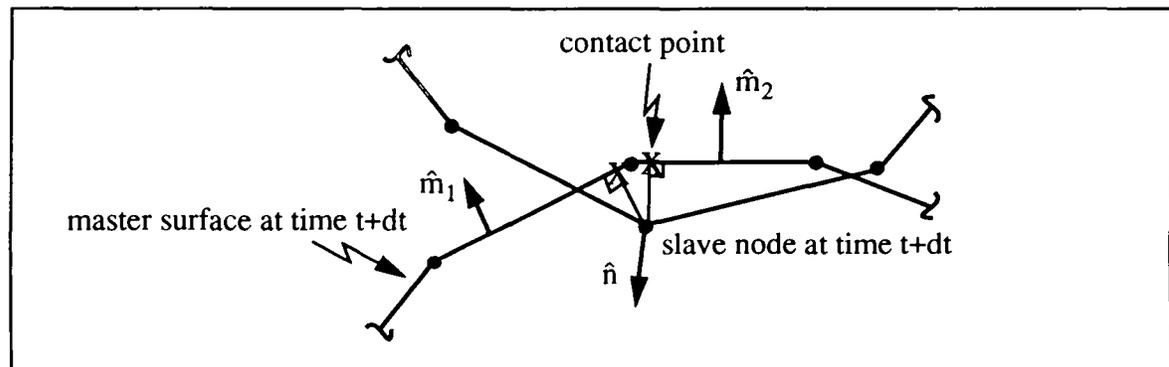
2.2.3 Overdetermined Contact

Further ambiguity is introduced when the contact algorithm performs the detailed contact checks based only on the estimated (deformed) configuration. For example, Figure 6 shows two different scenarios where a slave node can make contact with two master surfaces. Since the slave node motion is not used to determine how contact was made at time $t+dt$, two contact constraints can be defined (one with master surface 1 and the other with master surface 2). The algorithm, then, must resolve the overdetermined contact and decide which contact constraint to enforce. This is commonly done by determining which master surface is most opposed to the slave node surface normal [1][7][9]. The surface normal \hat{n} at a slave node is defined as the average of the normals of all the element faces connected to the slave node. A master surface is most opposed to the slave surface normal when the quantity $|\hat{n} \cdot \hat{m}|$ is largest. It is common to assume that the most opposed master surface is the one on which the

contact constraint should be enforced, so that in Figure 6a, the slave node is contacting master surface 1 and in Figure 6b the slave node is contacting master surface 2. The most opposed criterion is often called the strength of contact check.



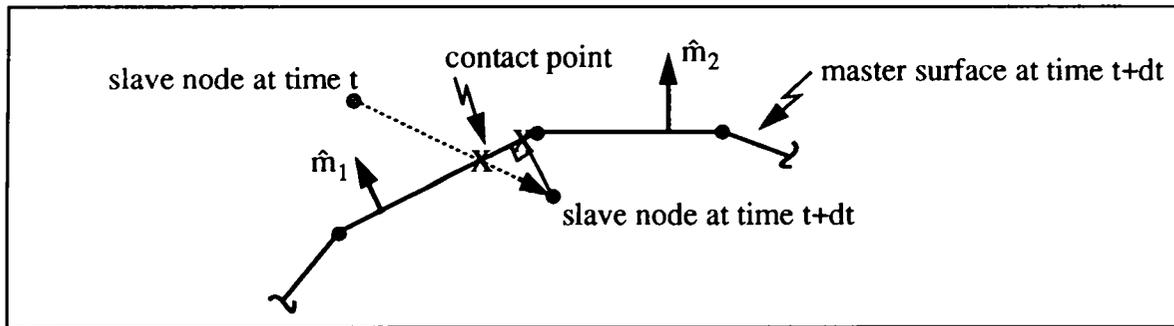
(a) slave node more opposed to master surface 1: $|\hat{n} \cdot \hat{m}_1| > |\hat{n} \cdot \hat{m}_2|$



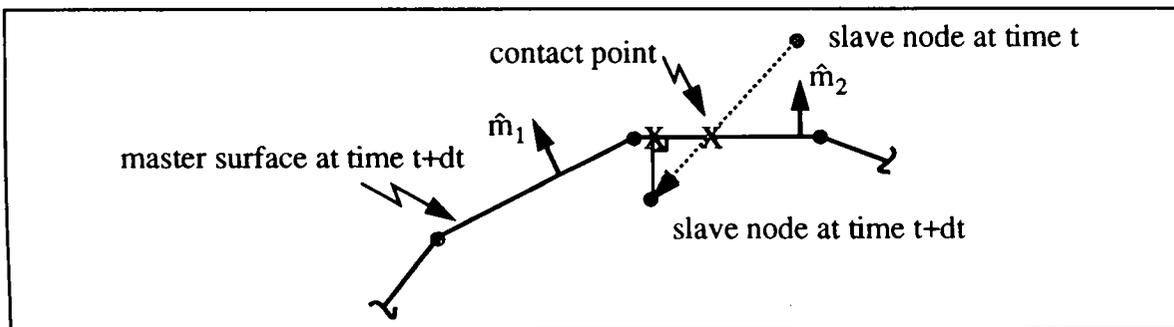
(b) slave node more opposed to master surface 2: $|\hat{n} \cdot \hat{m}_2| > |\hat{n} \cdot \hat{m}_1|$

Figure 6. Resolving overdetermined contact by determining most opposed master surface.

When two surfaces are already in contact the strength of contact check can be effective. However, when the surfaces are just coming into contact, as shown in Figure 7, this type of static contact check cannot consistently resolve the overdetermined contact and predict the correct contact constraint. In Figure 7a, the slave node makes contact with master surface 1. In Figure 7b, the slave node makes contact with master surface 2. The contacts are a result of the motion of the slave node and are not determined by the master surface which is most opposed.



(a) contact point on master surface 1



(b) contact point on master surface 2

Figure 7. Determining contact point with consideration of slave node's movement.

2.2.4 Undetermined Contact

Another ambiguity exists when the normal distance is undetermined. Figure 8 illustrates a situation when this can occur. The normal distance measured from the slave node to the master surface does not intersect any master segment. Intersection implies that the contact point, x , on the master surface lies between the master surface nodes, and not beyond them as shown in Figure 8a and b. Current algorithms typically allow an extension, ϵ , of the master surface so that contact can be detected. This extension means that either one contact is found or, in other cases, both contacts are detected and a choice between the two must be made. With either case, the contact point is still incorrectly defined. This inaccurate determination of the contact point causes "contact chatter", (see example 5.2).

In this situation, the detailed contact check should identify the vertex of the two master surfaces as the contact point since it is the closest point on the master surface, as shown in Figure 8c.

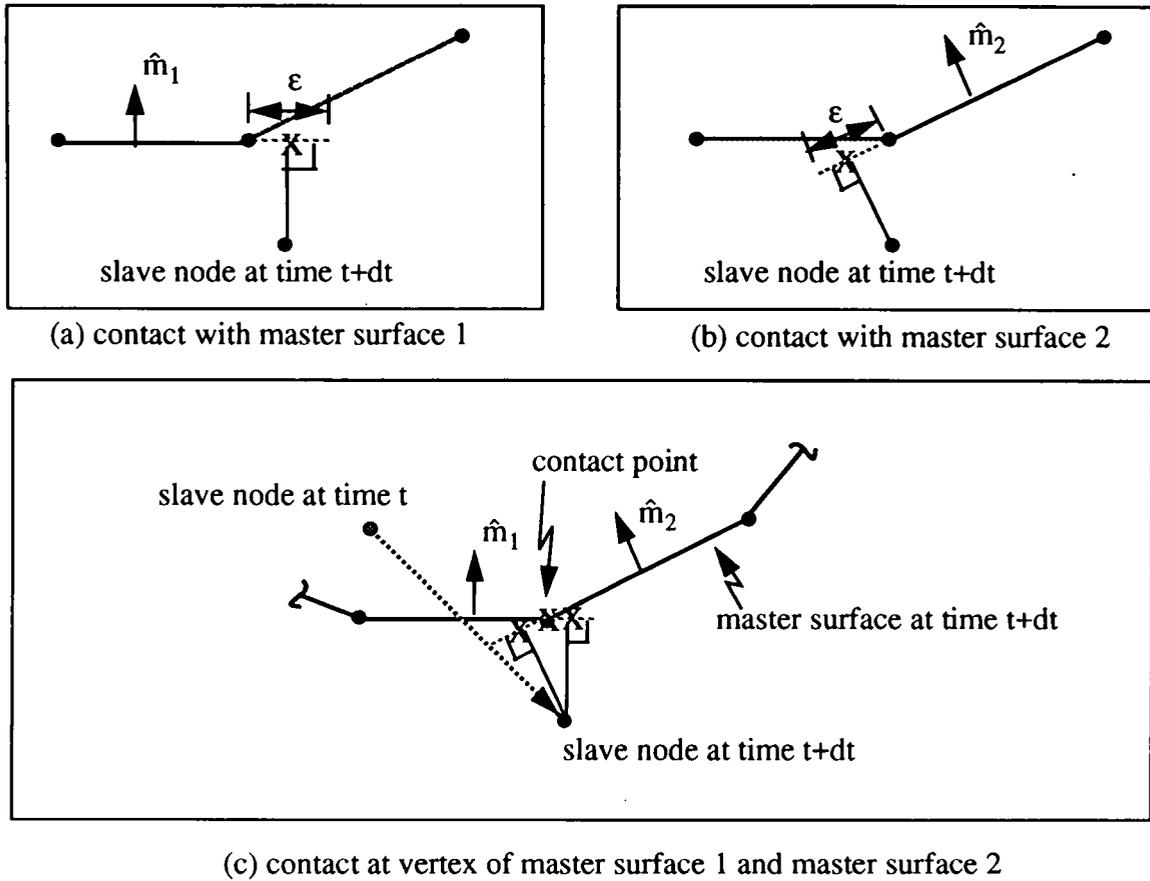


Figure 8. Resolving undetermined contact by identifying the closet point to the master surface

2.3 Motivation for Current Work

Several algorithms for efficient neighborhood identification have been summarized in this section. These included surface side-set pairing, surface tracking, bucket searching, and pinball contact. This current technology works successfully for a large variety of contact problems. There are some problems, however, where improvements in neighborhood identification and searching are required. Improvements would be beneficial in problems involving: structures that buckle and fold onto themselves; structures that have materials that tear and create new surfaces; multiple body contact/impact; and structures that slide relatively large distances over other surfaces.

In particular, improvements could address two distinct concerns with current search techniques. One concern is that nearly all neighborhood identification techniques are for nodes. For example, the result of a search is always the closest master *node* for a given slave node (with one exception in pinball overlap). A slave node does not always contact a master surface connected to the closest master node. Figure 9 shows a very simple shell self-contact

example demonstrating this fact. The slave node 2 is tracking master node 14, however it is actually contacting master surface 8-9 (defined by surface nodes 8-9). The tracked master node 14 implies that a detailed contact check would consider master surfaces 13-14 and 14-15 only and not master surface 8-9 as it should. Contact is always made between a slave node and a master *surface*, and an improved neighborhood identification should reflect this.

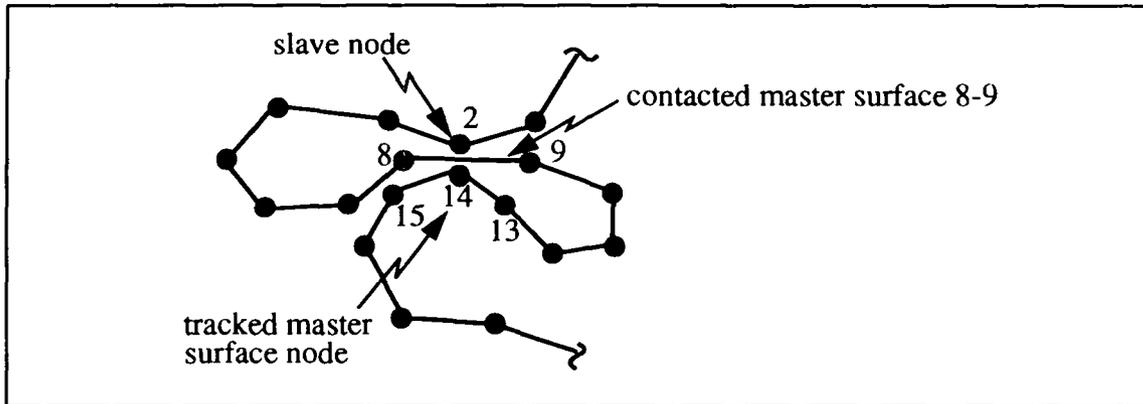


Figure 9. Slave node 2 tracking closest master node 14 results in a missed contact

Another concern is that current global sorting and searching routines depend on the problem domain. The efficiency of bucket searching is adversely affected by problems that grow or expand in spatial dimension. Significant improvements in the speed and efficiency of the search phase could be made if it depended only on the number of master surfaces and slave nodes in the problem, and *not* on the problem geometry.

The detailed contact check was also summarized in this section. The difficulties associated with detailed contact determination included (1) inaccurate pushback direction, (2) overdetermined (multiply defined) potential contact, and (3) undetermined contact. An improved contact detection algorithm should address these issues and offer improved accuracy in determining the contact point, penetration, and pushback direction.

3 New Contact Detection Algorithm

The proposed contact detection algorithm outlined in the following section offers improvements in the efficiency and accuracy of contact detection. The improvements are the result of a neighborhood identification strategy that uses a global contact search and an accurate detailed contact check that uses the projected motion of both master and slave surfaces. The algorithm does not require contact surface pairing and, therefore, easily handles self-contacting surfaces, eroding and tearing surfaces, and random contact between multiple bodies. The algorithm also resolves the ambiguities that arise because of the surface discretization and from using only the deformed configuration for detailed contact checks.

3.1 New Neighborhood Identification Strategy

The proposed neighborhood identification strategy uses a global search to determine what slave nodes are in close proximity to a master surface. It accumulates these potential interactions by constructing a local neighborhood around a master surface and globally searching for all slave nodes that are in the neighborhood. Two global search algorithms are presented, one that takes advantage of the characteristics of vector processors, and one that takes advantage of the characteristics of parallel processing. The efficiencies of each search algorithm are still under investigation. Both algorithms have been implemented on computers with scalar and vector processors. The algorithm for vector architecture uses a new search routine developed by Swegle [19]. The algorithm for parallel architecture is an extension of the state-of-the-art bucket searching technique [12], with a few significant exceptions.

To illustrate the two search algorithms, the example shown in Figure 10 of two blocks contacting each other will be used. Each block is a cube with dimensions 1" x 1" x 1". The centroid of block 1 is initially at the coordinates (0.5", 0.5", 0.5") and is moving with a velocity $v_y = 500$ in/s in the y-direction. The stable time step is approximately 1.1×10^{-6} s so that block 1 moves approximately 5.5×10^{-4} inches during one time step. Block 2 is initially at rest with its centroid located at the coordinates (0.5", 1.5", 0.5"). The symmetric contact enforcement being used implies that every node and element face on the surface is included in the contact surface. There are 52 slave nodes ($n=52$) and 48 master surfaces ($m_s=48$).

3.1.1 Algorithm for vector architecture

A search algorithm using $7n$ memory locations and requiring on the order of $m_s \log_2 n$ comparisons is utilized for the global location strategy. The algorithm is based on a particle search technique developed by Swegle [19]. It sorts the slave nodes by location and uses a binary search to construct a list of slave nodes that are in a master surface neighborhood. The search algorithm implemented here depends only on the number of slave nodes n and not on the geometry of the problem. It takes advantage of the known positions of the slave nodes and master surfaces as well as their predicted positions in the next time step.

The slave nodes are sorted based on their current coordinates, with each coordinate sorted independently. This sorting is done using an index vector so that the order of the slave nodes listed in the index vector corresponds to their coordinate value, with coordinates ordered from smallest to largest. For the current example, there are 52 slave nodes numbered as shown in Figure 10, resulting in the following index vectors:

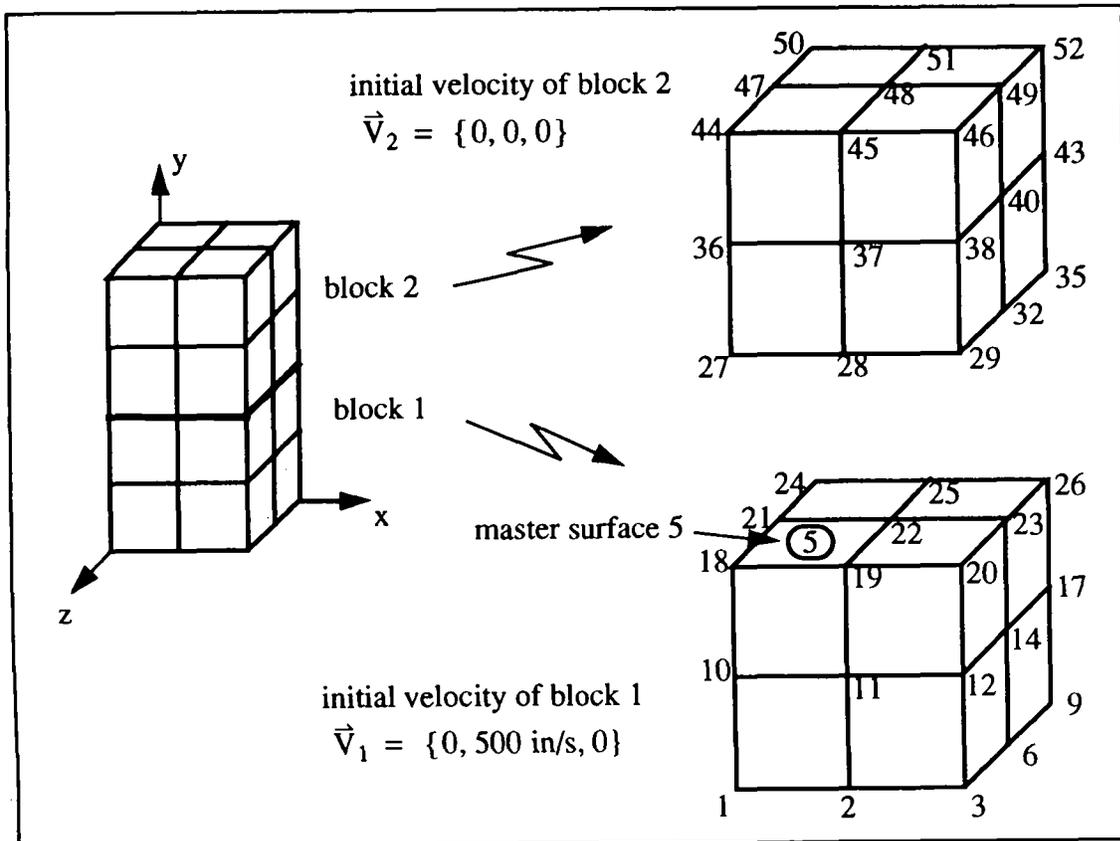


Figure 10. Example of two blocks contacting each other.

$$I_x = \{ 1,4,7,10,13,15,18,21,24,27,30,33,36,39,41,44,47,50,2,5,8,11,16,19, \\ 22,25,28,31,34,37,42,45,48,51,3,6,9,12,14,17,20,23,26,29,32,35,38, \\ 40,43,46,49,52 \}$$

$$I_y = \{ 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, \\ 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48, \\ 49,50,51,52 \}$$

$$I_z = \{ 7,8,9,15,16,17,24,25,26,33,34,35,41,42,43,50,51,52,4,5,6,13,14,21, \\ 22,23,30,31,32,39,40,47,48,49,1,2,3,10,11,12,18,19,20,27,28,29,36, \\ 37,38,44,45,46 \}$$

for the x, y, and z coordinates respectively. A rank vector is also constructed for the slave nodes. It gives the location of each slave node in the index vector and is required to avoid searching the index vector for a given slave node. It can be easily constructed by looping through the index vector. For example, suppose a slave node i is stored at position j in the index vector. The rank vector would then store the pointer j at its position i . For the current example, the rank vectors are:

$$R_x = \{ 1,19,35,2,20,36,3,21,37,4,22,38,5,39,6,23,40,7,24,41,8,25,42,9,26,43, \\ 10,27,44,11,28,45,12,29,46,13,30,47,14,48,15,31,49,16,32,50,17,33, \\ 51,18,34,52 \}$$

$$R_y = \{ 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, \\ 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48, \\ 49,50,51,52 \}$$

$$R_z = \{ 35,36,37,19,20,21,1,2,3,38,39,40,22,23,4,5,6,41,42,43,24,25,26,7,8,9, \\ 44,45,46,27,28,29,10,11,12,47,48,49,30,31,13,14,15,50,51,52,32,33, \\ 34,16,17,18 \}$$

After the slave nodes are sorted by x, y and z coordinate, the master surfaces are processed sequentially. This processing involves defining a local neighborhood for each master surface by bounding the space occupied by a master surface at its known location at time t and its predicted location at time $t+dt$. Figure 11 shows a bounding box for a master surface over one time step. Another box, called the capture box is also constructed to collect slave nodes that potentially contact the master surface.

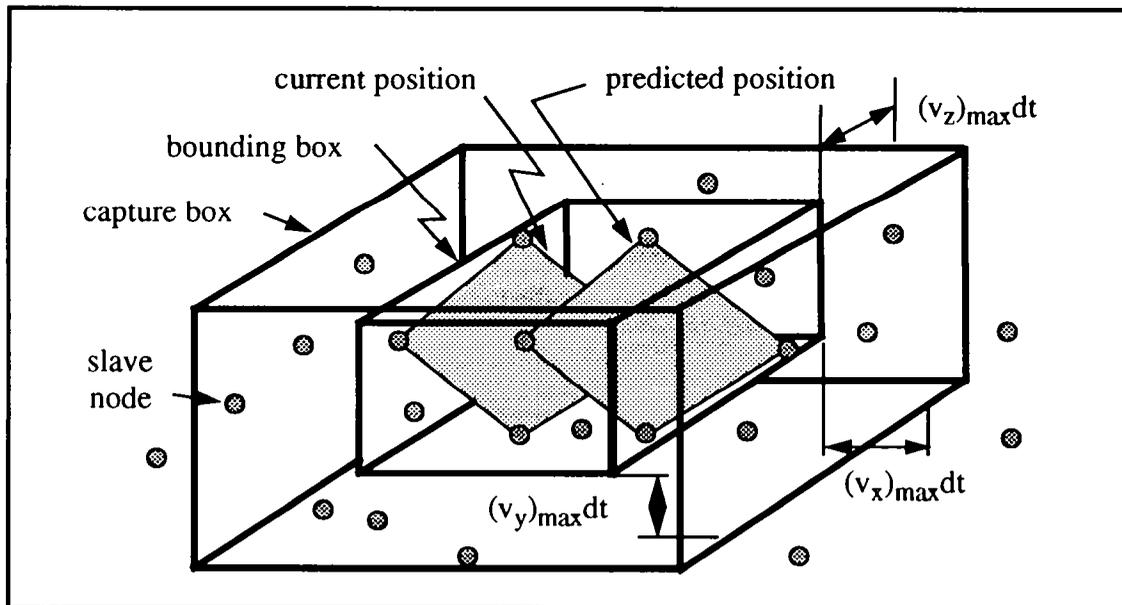


Figure 11. Bounding box and capture box for a moving master surface

For example, suppose the maximum distance any slave node moves in one time step is $(v_x)_{\max}dt$ in the x-direction. Then a slave node within a distance $(v_x)_{\max}dt$ from the bounding box could potentially contact the master surface. This holds for the y- and z-directions as well so that a capture box can be constructed, as shown in Figure 11. Any slave node inside the capture box should therefore be considered for potential contact with the master surface. In the example problem, $(v_x)_{\max} = 0$, $(v_y)_{\max} = 500$ in/s, $(v_z)_{\max} = 0$, and $dt = 1.1 \times 10^{-6}$ s, so the capture box for master surface 5 would have the corners:

$$x_{\min} = 0.0 \text{ in.}, x_{\max} = 0.5 \text{ in.}$$

$$y_{\min} = 1.0 - v_y dt = 0.99945 \text{ in.}, y_{\max} = 1.0 + v_y dt = 1.00055 \text{ in.}$$

$$z_{\min} = 0.5 \text{ in.}, z_{\max} = 1.0 \text{ in.}$$

At this point it is necessary to address one aspect of the contact enforcement algorithm that will affect contact detection. Recalling that a contact constraint is likely to be enforced over several time steps, the capture box must be enlarged to capture slave nodes with a partially enforced contact constraint. A distance, p_r , equivalent to the amount of penetration not enforced is easily computed for each slave node in contact with a master surface, as shown in Figure 12. The capture box is enlarged in all directions by this distance, p_r .

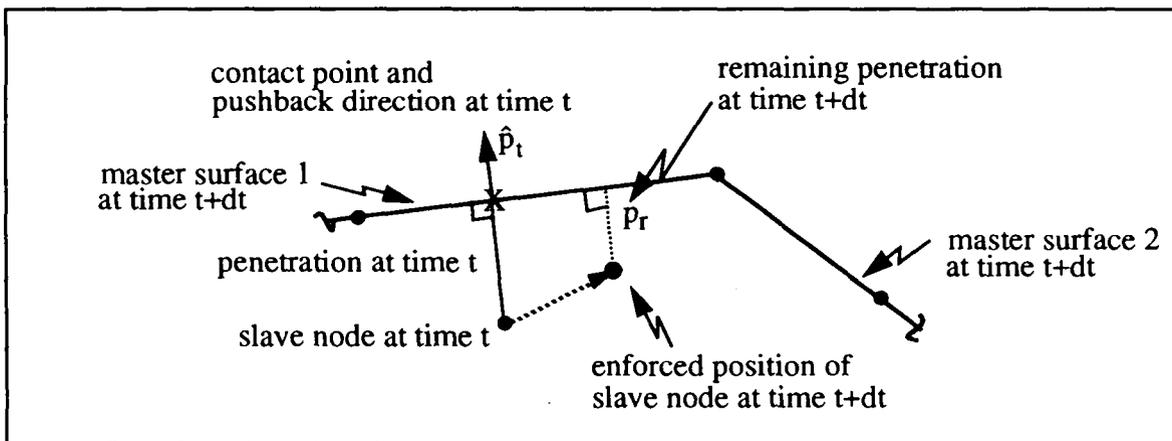


Figure 12. Remaining penetration due to a partially enforced contact constraint

After defining the capture box a binary search is done on the index vectors to find the slave nodes that are inside the capture box. The binary search results in two pointers for each coordinate direction. One is a lower pointer into the index vector that corresponds to the position in the index vector of the first slave node inside the capture box and the other is an upper pointer that corresponds to the position in the index vector of the last slave node inside the capture box.

For the example problem node 1 at position 1 in I_x is the first node in the box in the x direction, and node 51 at position 34 in I_x is the last node inside the capture box. In the y direction node 18 in position 18 of I_y and node 35 at position 35 in I_y are the first and last nodes just inside the capture box, respectively. Similarly, in the z direction node 4 in position 19 of I_z and node 46 at position 52 in I_z are the first and last nodes inside the capture box, respectively. Therefore, the binary search would give the following results:

$$(i_x)_{\min} = 1, (i_x)_{\max} = 34$$

$$(i_y)_{\min} = 18, (i_y)_{\max} = 35$$

$$(i_z)_{\min} = 19, (i_z)_{\max} = 52$$

This means, for example, that the subset of slave nodes from $I_x((i_x)_{\min})$ to $I_x((i_x)_{\max})$ are in the capture box of the master surface on the basis of the x-coordinate. For the example problem, the subset or list of slave nodes for each coordinate direction that are in the capture box of master surface 5 are:

$$L_x = \{ 1,4,7,10,13,15,18,21,24,27,30,33,36,39,41,44,47,50,2,5,8,11,16,19,22, \\ 25,28,31,34,37, 42,45,48,51 \}$$

$$L_y = \{ 18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35 \}$$

$$L_z = \{ 4,5,6,13,14,21,22,23,30,31,32,39,40,47,48,49,1,2,3,10,11,12,18,19,20, \\ 27,28,29,36, 37,38,44,45,46 \}$$

Finally, a contraction is done to find the slave nodes in the capture box of the master surface in all three coordinate directions simultaneously. To accomplish this, each of the slave nodes in one list is selected and then checked if its rank is between the lower and upper pointer in the other two coordinates. For computational efficiency the shortest list of slave nodes is selected, which can be determined by selecting the smallest of $[(i_w)_{\max} - (i_w)_{\min} + 1]$, $w = x, y, \text{ or } z$.

For the example problem, L_y is the list with the smallest number of slave nodes, so that slave nodes $i = I_y((i_y)_{\min}), I_y((i_y)_{\min} + 1), \dots, I_y((i_y)_{\max})$ are in the capture box if

$$(i_x)_{\min} \leq R_x(i) \leq (i_x)_{\max} \text{ and } (i_z)_{\min} \leq R_z(i) \leq (i_z)_{\max}.$$

To help illustrate this procedure, the first few slave nodes in list L_y are processed as follows:

$$1) i = I_y((i_y)_{\min}) = 18, R_x(18) = 7, \text{ and } R_z(18) = 41.$$

Since $(i_x)_{\min} \leq 7 \leq (i_x)_{\max}$ and $(i_z)_{\min} \leq 41 \leq (i_z)_{\max}$ slave node 18 is in the capture box of master surface 5.

$$2) i = I_y((i_y)_{\min} + 1) = 19, R_x(19) = 24, \text{ and } R_z(19) = 42.$$

Since $(i_x)_{\min} \leq 24 \leq (i_x)_{\max}$ and $(i_z)_{\min} \leq 42 \leq (i_z)_{\max}$ slave node 19 is also in the capture box of master surface 5.

$$3) i = I_y((i_y)_{\min} + 2) = 20, R_x(20) = 41, \text{ and } R_z(20) = 43.$$

Since $41 > (i_x)_{\max}$ slave node 20 is NOT in the capture box of master surface 5.

After processing all of the slave nodes in list L_y , it is found that only the slave nodes $\{18,19,21,22,27,28,30,31\}$ are within the bounding box of master surface 5.

For efficiency on a vector computer each slave node-master surface pair is stored in an array

for later processing after a vector length of pairs is accumulated. Note that only those pairs are added where the slave node is not also a node on the master surface. For the example problem only the following four pairs would be added to the array:

$$\{ (27, 5), (28, 5), (30, 5) \text{ and } (31, 5) \}$$

At this point an exhaustive search has been done and no assumptions have been made on the manner in which contact can or will be made. The processing that will determine contact is called the detailed contact check. In the current implementation of the algorithm, the binary sorting is implemented *every* time step.

3.1.2 Algorithm for parallel architecture

The following algorithm is essentially a conventional bucket search [12] with two significant exceptions. These exceptions are that (1) the bucket size b_s is based on the smallest master surface dimension and (2) a capture box is used to ensure all potential contacts are gathered for a detailed contact check. The capture box takes advantage of the known positions of the slave nodes and master surfaces as well as the predicted positions in the next time step. Both are described in detailed after the bucket search algorithm is reviewed.

The bucket search algorithm first sorts the slave nodes into buckets, then finds the buckets that a master surface occupies and pairs the master surface with the slave nodes in those buckets. It requires $(18n)$ memory locations plus two vectors with lengths equal to the number of buckets (n_b).

To sort the slave nodes into buckets, an integer coordinate system is constructed in each physical direction. Assuming that the bucket size b_s is based on the dimension of the smallest master surface, the number of slices (each with thickness b_s) required in the x, y, and z directions are:

$$S_x = \text{int} [(x_{\max} - x_{\min}) / b_s] + 1$$

$$S_y = \text{int} [(y_{\max} - y_{\min}) / b_s] + 1$$

$$S_z = \text{int} [(z_{\max} - z_{\min}) / b_s] + 1$$

The three slices containing any given node, i , can be calculated as follows:

$$S_x^i = \text{int} [(x_i - x_{\min}) / b_s] + 1$$

$$S_y^i = \text{int} [(y_i - y_{\min}) / b_s] + 1$$

$$S_z^i = \text{int} [(z_i - z_{\min}) / b_s] + 1$$

The intersection of the three orthogonal slices defines the bucket B^i containing node i , as shown in Figure 14.

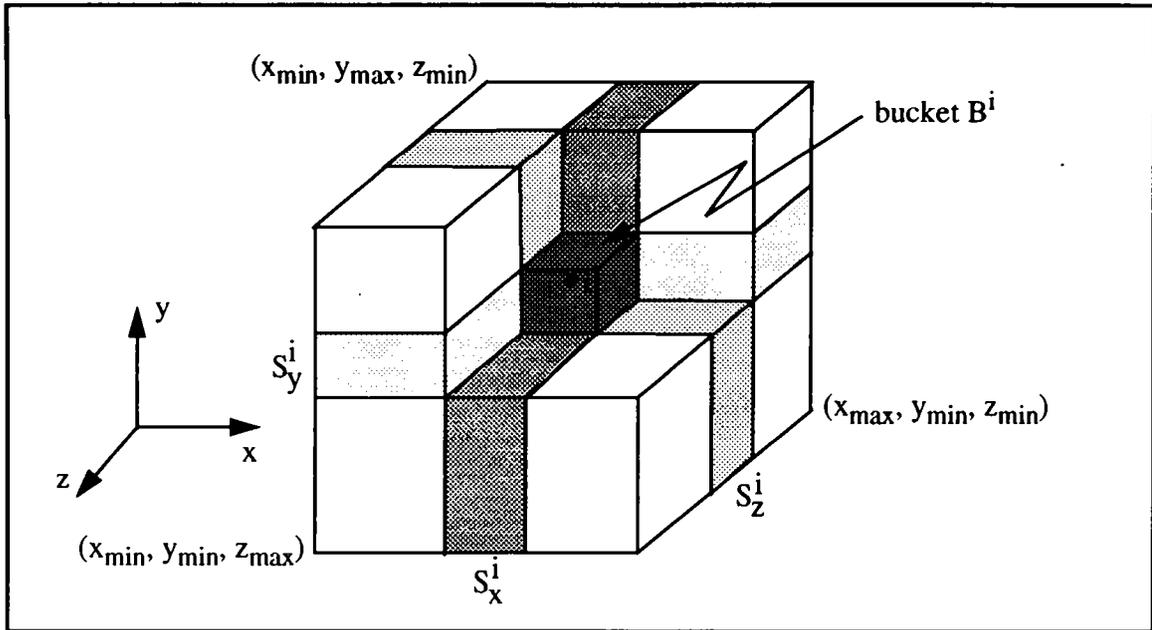


Figure 13. Bucket B^i defined by three slices of the data that contain node i .

If buckets are numbered sequentially, progressing first in the x direction, next in the y direction, and then in the z direction, the bucket id containing node i is given by;

$$B^i = (B_z^i - 1) S_x S_y + (B_y^i - 1) S_x + B_x^i$$

With the eventual aim of efficiently determining a list of slave nodes within any given bucket, two vectors of length n are created. One stores the bucket id for each slave node, $lbox$, and the other is a list of slave nodes sorted by bucket id, $ndsort$. Two additional vectors of length n_b are also constructed to efficiently access the slave nodes. The first contains the number of slave nodes occurring in each bucket, $nbox$, and the other is a pointer that identifies the first slave node in each bucket, $npoint$. The entire sorting procedure can be summarized as follows:

1. Zero the vector ($nbox$) containing the number of nodes in each bucket.
2. Find the bucket id (B^i) for each node.
3. Store the bucket id for node i in the vector: $lbox(i) = B^i$
4. Increment the counter for bucket B^i : $nbox(B^i) = nbox(B^i) + 1$
5. Calculate the pointer for each bucket j into a sorted list of nodes.

$$npoint(1) = 1 \qquad npoint(j) = npoint(j-1) + nbox(j-1)$$
6. Zero the vector $nbox$

7. Sort the slave nodes according to their bucket number into a list *ndsort*.

$$ndsort(nbox(lbox(i)) + npoint(lbox(i))) = i$$

$$nbox(lbox(i)) = nbox(lbox(i)) + 1$$

For the example problem, the bucket size is 0.5 inches corresponding to the dimension of the smallest master surface. This implies that $S_x = 3$, $S_y = 5$, and $S_z = 3$ and a total number of buckets $n_b = 45$. The bucket sort of all 52 slave nodes would give the following results:

Bucket id for each slave node:

$$lbox = \{31,32,33,16,17,18,1,2,3,34,35,36,19,21,4,5,6,37,38,39,22,23,24,7,8,9, \\ 37,38,39,22,23,24,7,8,9,40,41,42,25,27,10,11,12,43,44,45,28,29,30, \\ 13,14,15\}$$

Pointer into *ndsort* giving the starting location of the slave nodes in each bucket:

$$npoint = \{1,2,3,4,5,6,7,9,11,13,14,15,16,17,18,19,20,21,22,23,23,24,26,28,30,31, \\ 31,32,33,34,35,36,37,38,39,40,41,43,45,47,48,49,50,51,52\}$$

List of slave nodes sorted by their bucket id:

$$ndsort = \{7,8,9,15,16,17,24,33,25,34,26,35,41,42,43,50,51,52,4,5,6,13,14,21, \\ 30,22,31,23,32,39,40,47,48,49,1,2,3,10,11,12,18,27,19,28,20,29,36,37, \\ 38,44,45,46\}$$

Number of slave nodes in each bucket:

$$nbox = \{1,1,1,1,1,2,2,2,1,1,1,1,1,1,1,1,1,0,1,2,2,2,1,0,1,1,1,1,1,1,1,1,1,1,2,2,2, \\ 1,1,1,1,1,1\}$$

To illustrate how all slave nodes within, for example, bucket 23 are gathered, the sorted information described above is used as follows. The number of slave nodes in bucket 23 is given by: $nbox(23) = 2$. The slave nodes occupying bucket 23 can be found starting at location $npoint(23)=26$ in *ndsort*. This implies that slave nodes at location 26 and 27 in *ndsort* are in bucket 23: $ndsort(26-27) =$ slave nodes {22,31}.

The sorting algorithm above is identical to that described in [12][16]. Potential interactions between nodes in [12][16] are determined by looping through the buckets and collecting the slave nodes in neighboring buckets, using the pointer into the sorted list. This implies that the bucket size must be based on the largest master surface dimension to avoid missing potential contact, as was demonstrated in Figure 3.

To avoid these difficulties, the strategy that the proposed search algorithm employs is to collect all the buckets *occupied* by a master surface. Then, using the information obtained during sorting, the slave nodes in those buckets are collected. This ensures that all potential

interactions with a given master surface are found regardless of bucket size (and therefore problem geometry). In the current algorithm, the bucket size should be interpreted as the sorting refinement instead of a length measure of the largest master surface. A bucket size based on the *smallest* master surface assures that there will be a small number of slave nodes in each bucket and admits the possibility that a master surface could occupy many buckets.

The collection of slave nodes with which a master surface could potentially interact is determined in the following way:

1. A capture box is constructed for each master surface, as described previously
2. The buckets containing the extreme corners of the master surface capture box are determined

$$ibox_{min} = \min(S_x , \text{ifix}((xc_{min}-x_{min})/b_s) + 1)$$

$$jbox_{min} = \min(S_y , \text{ifix}((yc_{min}-y_{min})/b_s) + 1)$$

$$kbox_{min} = \min(S_z , \text{ifix}((zc_{min}-z_{min})/b_s) + 1)$$

$$ibox_{max} = \min(S_x , \text{ifix}((xc_{max}-x_{min})/b_s) + 1)$$

$$jbox_{max} = \min(S_y , \text{ifix}((yc_{max}-y_{min})/b_s) + 1)$$

$$kbox_{max} = \min(S_z , \text{ifix}((zc_{max}-z_{min})/b_s) + 1)$$

where xc_{min} and xc_{max} are the corners of the master surface capture box in the x direction (yc_{min} , yc_{max} , zc_{min} and zc_{max} have the same definitions in the y and z directions, respectively).

3. All buckets within the ranges calculated in step 2 are identified as follows:

```

Loop from  ibox = iboxmin  to  iboxmax
  Loop from  jbox = jboxmin  to  jboxmax
    Loop from  kbox = kboxmin  to  kboxmax
      Bi = (kbox-1)SxSy + (jbox-1)Sx + ibox
    Endloop
  Endloop
Endloop

```

4. All slave nodes in the buckets calculated in step 3 are considered for potential interaction with the master surface

In this approach, the algorithm takes advantage of the observation that the buckets that a master surface occupies can be easily determined. In the example problem, assume that master surface 5 is to be processed. The corners of the capture box of master surface 5 are:

$$x_{c_{\min}} = 0.0 \text{ in.}, \quad x_{c_{\max}} = 0.5 \text{ in.}$$

$$y_{c_{\min}} = 0.99945 \text{ in.}, \quad y_{c_{\max}} = 1.00055 \text{ in.}$$

$$z_{c_{\min}} = 0.5 \text{ in.}, \quad z_{c_{\max}} = 1.0 \text{ in.}$$

so that the capture box would occupy the range of buckets:

$$i_{\text{box}_{\min}} = 1, \quad i_{\text{box}_{\max}} = 2$$

$$j_{\text{box}_{\min}} = 3, \quad j_{\text{box}_{\max}} = 3$$

$$k_{\text{box}_{\min}} = 2, \quad k_{\text{box}_{\max}} = 3$$

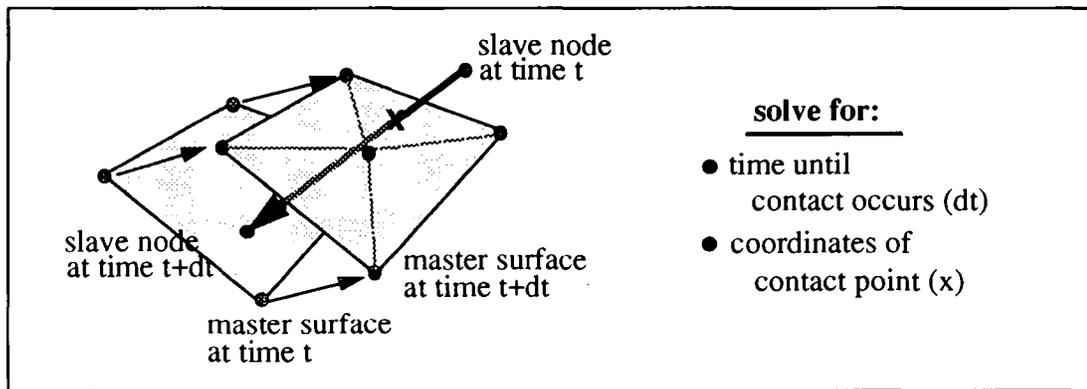
This identifies buckets 22, 23, 37, and 38. The slave nodes occupying these buckets would be considered for potential contact with master surface 5. Note that only the slave nodes which are not also nodes on master surface 5 are considered, so that only the following four pairs would be added to the vectorized list:

$$\{ (27, 5), (28, 5), (30, 5) \text{ and } (31, 5) \}$$

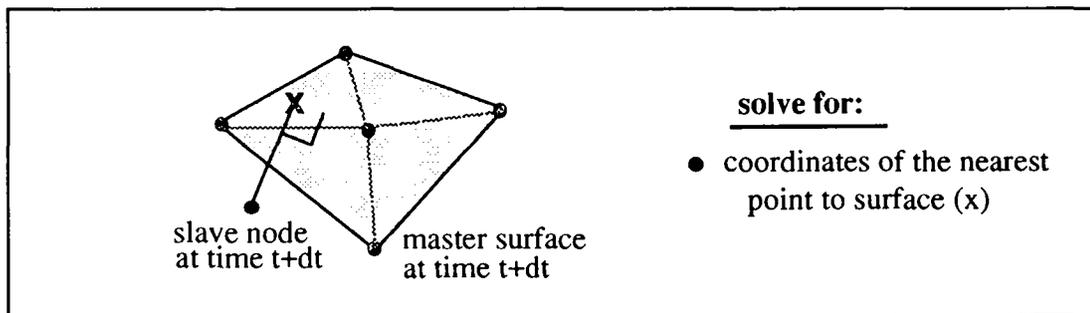
In the current implementation of the algorithm, the bucket sorting is implemented *every* time step in a vectorized mode. A study of the efficiencies of a bucket search algorithm was presented by Plimpton [20] for a computer with parallel architecture. For a certain class of spatially compact problems, the bucket searching seems to be faster than the vectorized global search. Further speedup might be anticipated by sorting every 5 to 10 time steps, suggested by [12], while storing the nearest neighbors for contact determination in the intermediate time steps.

3.2 New Detailed Contact Check

The proposed detailed contact check distinguishes between slave nodes that are not in contact and those that are already in contact. It does so to resolve the ambiguities that arise in each case. The ambiguity shown in Figure 7, for example, could be easily resolved by considering the velocity of the slave node. This idea of a velocity based contact check can be extended to include a moving slave node contacting a moving master surface, as shown in Figure 14a. For slave nodes just coming into contact the velocity based contact check identifies the point of contact (or impact) and the contact time. Figure 14b shows a static contact check that is used for slave nodes already in contact with a master surface. For these slave nodes, ambiguities arise because the surface normal is not continuous. This can result in not finding any contact when there should be one or finding multiple solutions to a single contact. The proposed velocity based and static based detailed contact checks resolve these ambiguities, as shown in the following sections.



(a) velocity based contact check based on position and velocity



(b) static based contact check based on position

Figure 14. Master slave tracking using velocity and static contact check

3.2.1 Velocity Based Contact Check

The velocity based contact check makes use of the current position of the surfaces and the estimated velocities in the following time step. The contact check is restricted to a moving triangular master surface and a moving slave node for 3D, as shown in Figure 14a.

The bi-linear quadrilateral master surface is subdivided into four triangular master surfaces. The four triangles are defined by the four corner nodes of the quadrilateral plus an added fifth node located at its centroid. In 2D, the master surface is a moving line, and is just a special case of the 3D moving contact presented below. A complete derivation of the velocity based contact check can be found in Appendix 1.

The equation of a triangular master surface (a plane) can be written as:

$$a(x - x_1) + b(y - y_1) + c(z - z_1) = 0 \quad (4)$$

The point (x_1, y_1, z_1) is a node on the master surface, and a , b , and c are components of the master surface normal $\vec{N}_m = a\vec{i} + b\vec{j} + c\vec{k}$, where:

$$\begin{aligned} a &= [(y_1 - y_3)(z_2 - z_1) - (y_2 - y_1)(z_1 - z_3)] \\ b &= [(x_2 - x_1)(z_1 - z_3) - (x_1 - x_3)(z_2 - z_1)] \\ c &= [(x_1 - x_3)(y_2 - y_1) - (x_2 - x_1)(y_1 - y_3)] \end{aligned} \quad (5)$$

Here, x_2, y_2, z_2 and x_3, y_3, z_3 are coordinates of the other nodes defining the triangular master surface. Now, note that the triangular surface is moving, so that:

$$\begin{aligned} [(x_i(t + \Delta t), y_i(t + \Delta t), z_i(t + \Delta t)) = \\ (x_i(t) + \dot{x}_i\Delta t, y_i(t) + \dot{y}_i\Delta t, z_i(t) + \dot{z}_i\Delta t)], i = 1, 3 \end{aligned} \quad (6)$$

and the slave node is also moving, so that:

$$\begin{aligned} (x_s(t + \Delta t), y_s(t + \Delta t), z_s(t + \Delta t)) = \\ (x_s(t) + \dot{x}_s\Delta t, y_s(t) + \dot{y}_s\Delta t, z_s(t) + \dot{z}_s\Delta t) \end{aligned} \quad (7)$$

Following these definitions, a time Δt is sought such that the slave node will be co-planar with the three nodes defining the triangular master surface. Such a time can be found (if it exists) by substituting equations (5), (6), and (7) into equation (4) and solving a cubic equation in Δt (see Appendix 1).

Suppose a solution $\Delta t = \Delta t_c$ is found. Then it also must satisfy two conditions for it to be considered a contact. The first is that the time Δt_c occurs in the next time step increment, $0 \leq \Delta t_c \leq dt$. The other condition is that the contact point:

$$(x_c, y_c, z_c) = (x_s + \dot{x}_s\Delta t_c, y_s + \dot{y}_s\Delta t_c, z_s + \dot{z}_s\Delta t_c) \quad (8)$$

must lie inside the edges of the quadrilateral master surface. This can be determined exactly by computing the local ξ, η coordinates of the contact point on the quadrilateral master surface. Figure 15 shows how an initial estimate ξ_0, η_0 of the local coordinates can be computed. Four triangles are constructed by connecting the contact point with the four corners of the quadrilateral master surface. The contact point is inside the master surface when all four areas $A_1, A_2, A_3,$ and A_4 are greater than or equal to zero and $A_1 + A_3 > 0$ and $A_2 + A_4 > 0$. If this condition is met, then an initial estimate of the local coordinates of the contact point can be found, as shown by the equations in Figure 15.

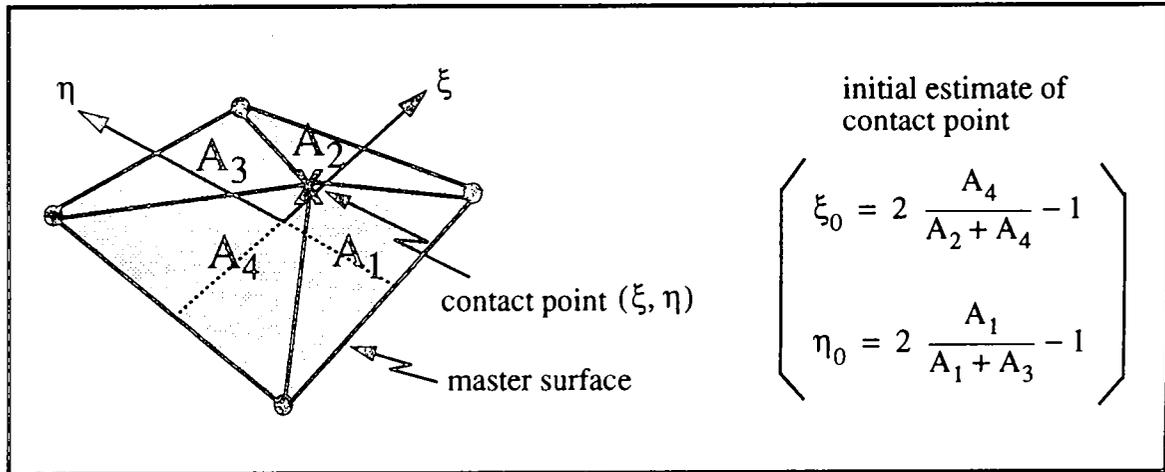


Figure 15. Initial estimates for the local coordinates of a contact point

The logic for these equations is based simply on the observation that $(\xi, \eta) = (\xi_0, \eta_0)$ is computed exactly for any rectangular quadrilateral and that the limiting values of ξ and $\eta = \pm 1$ are computed exactly for any shaped quadrilateral. If the quadrilateral surface is a rectangle, then the proof is simple. If it is not, then by observation ξ is computed exactly only when one of the areas A_2 or A_4 is zero. Likewise η is computed exactly only when one of the areas A_1 or A_3 is zero. For example, suppose A_2 is zero then the estimated coordinate $\xi = \xi_0 = 1$ is exact. If A_4 is zero then the estimated coordinate $\xi = \xi_0 = -1$ is also exact. If neither A_2 or A_4 are zero, then the equation gives a reasonable estimate of the contact point ξ . Improvements in the accuracy of the local coordinates can be achieved by performing Newton iterations on the nonlinear equations relating $\xi, \eta,$ and ζ to $x_c, y_c,$ and z_c (see Appendix 1).

In certain cases there may be multiple master surfaces where contact is possible. The example problem shown in Figure 10 illustrates such a case in which a slave node contacts multiple master surfaces. For example, the slave node 27 could contact any one of the three master surfaces connected to node 18. In these cases, a strength of contact check is used to determine the most opposed master surface where $\vec{N}_m \cdot (\vec{V}_s - \vec{V}_m)$ is minimized. In the example problem, master surface 5 is the most opposed master surface.

3.2.2 Static Contact Check

The static contact check is used for those slave nodes already in contact with a master surface. The contact check is also restricted to a slave node contacting a triangular master surface. The calculations are based on the predicted configuration of the surface if the contact forces were removed from the surface nodes. This predicted configuration would obviously have slave nodes penetrating master surface elements. In order to bring these interpenetrating surfaces back into compliance, the slave node must be pushed back to the master surface. In determining the push back direction, a distinction is made between concave and convex surfaces.

The push back direction for a concave surface is determined simply by the minimum distance to the master surface as shown in Figure 16. The push back direction \hat{p}_{t+dt} can be along the master surface normal as shown in Figure 16a or it may be defined by the minimum distance to a vertex as shown in Figure 16b. This choice in pushback direction is done simply to detect an undetermined contact.

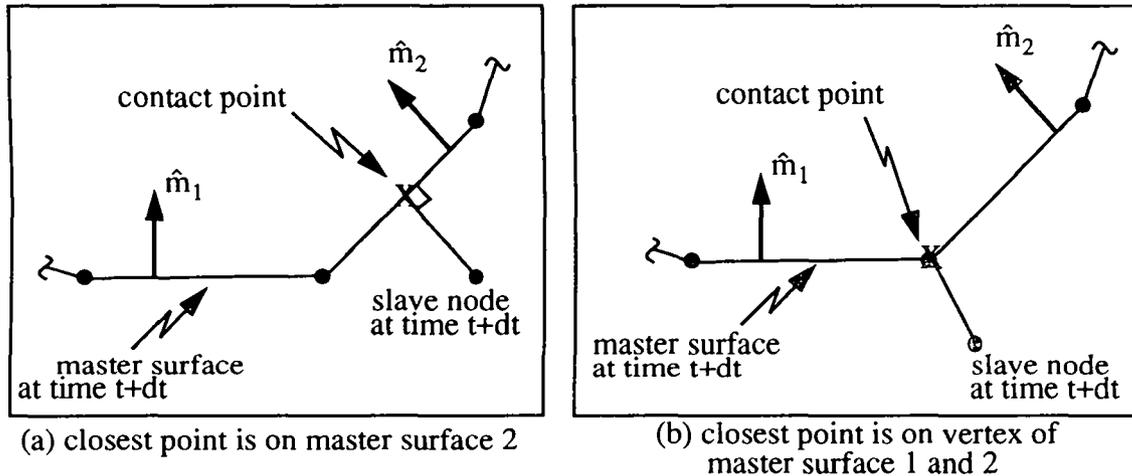


Figure 16. Push back direction for a concave surface based on minimum distance to master surface.

For a convex surface, the push back direction \hat{p}_{t+dt} is always along the normal of the master surface that the slave node was previously in contact with, i.e. \hat{m}_p in Figure 17. In Figure 17a the slave node was previously in contact with master surface 1 ($\hat{m}_p = \hat{m}_1$) so the pushback direction is $\hat{p}_{t+dt} = \hat{m}_p = \hat{m}_1$. Again, in Figure 17b the slave node was originally in contact with master surface 1 so $\hat{p}_{t+dt} = \hat{m}_p = \hat{m}_1$. After the slave node is pushed back to master surface 2, the pushback direction is updated to reflect that $\hat{p}_{t+2dt} = \hat{m}_2$. This avoids adding artificial slave node velocity due to a change in pushback direction, as illustrated in Figure 5.

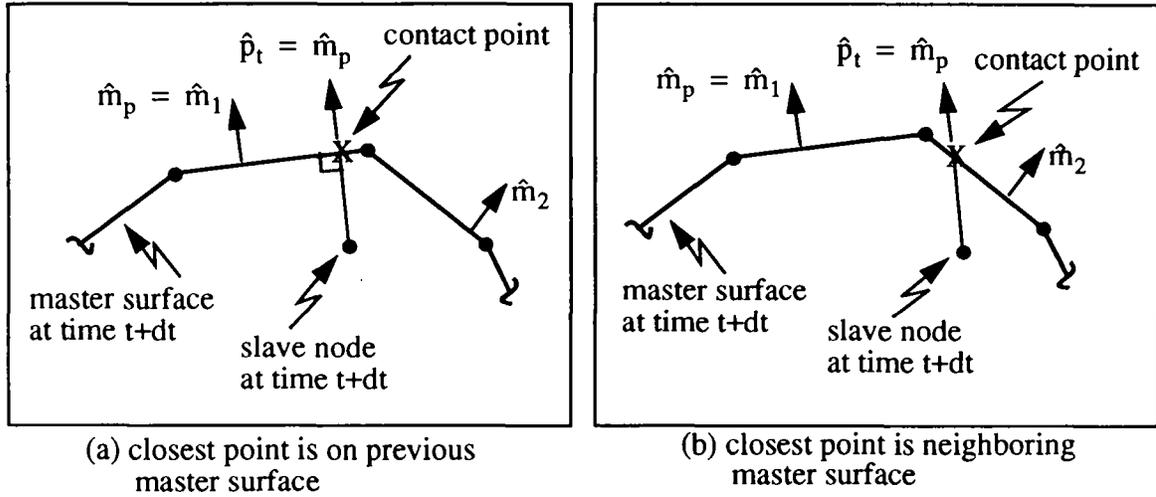


Figure 17. Push back direction for convex surface based on previous master surface normal.

The contact point, X , on a convex surface is found by the intersection of the triangular master surface plane, Equation (4), and a line defined by the parametric equations:

$$\begin{aligned}
 x &= x_s + s (\hat{m}_p \cdot \hat{i}) \\
 y &= y_s + s (\hat{m}_p \cdot \hat{j}) \\
 z &= z_s + s (\hat{m}_p \cdot \hat{k})
 \end{aligned} \tag{9}$$

Providing that the master surface and the line are not parallel, the parameter s can be found as:

$$s = \frac{a (x_1 - x_s) + b (y_1 - y_s) + c (z_1 - z_s)}{a (\hat{m}_p \cdot \hat{i}) + b (\hat{m}_p \cdot \hat{j}) + c (\hat{m}_p \cdot \hat{k})} \tag{10}$$

Just as in the case of a velocity based contact check, there may be some instances where contact with multiple master surfaces is possible according to the static contact check. Again a strength of contact check is used to determine the master surface where $\vec{N}_m \cdot \vec{N}_s$ is minimized.

3.3 Summary of Proposed Contact Detection Algorithm

The proposed contact detection algorithm outlined in this section offers some advantages over the currently used algorithm in the Sandia codes. It does so by separately considering a fast, memory efficient, global search for potential contacts, and a much more accurate detailed contact check:

- The efficient global search allows for global contact. This means that added capability for modelling self contacting structures and eroding or tearing surfaces is now available. Another benefit of the efficient global contact search is that fewer slave node - master surface pairs are found as potential contacts. This is possible by taking advantage of both the known location of the contacting surfaces and their velocities in constructing a master surface capture box. The capture box ensures that a minimum number of slave nodes are paired with the master surface.
- The detailed contact check is more accurate in determining the point of contact, amount of penetration, and the direction of pushback. This results in a physically correct determination of contact constraints. The improved accuracy has also reduced the number of iterations required for convergence in the iterative solvers in the Sandia quasistatic codes.

4 Surface Definition Algorithm

The proposed contact detection algorithm described in the previous section discussed the details of the neighborhood identification and global search, and the detailed contact check. Implied in the discussion was the definition of the entities involved in contact, namely the slave nodes and the master surfaces. Typically this is done using preprocessors such as FASTQ [21], GJOIN [22], GEN3D [23], and GENSHELL [24] to define meshes and side sets. Side sets are a collection of element faces and their corresponding nodal points. The contact algorithm would then require as input the listing of these side sets (without requiring side set pairing). In many applications, such as self contact, this kind of input is all that is required. For other applications such as tearing or eroding surfaces, the contact surface is changing during the execution of the problem. In these types of problems, a dynamically defined surface (composed of all master surfaces and slave nodes on the surface) is essential.

For this purpose, an algorithm for automatically determining the surface of an arbitrary mesh composed of hexahedral and shell elements is proposed. The surface definition algorithm uses a data structure that allows the initial surface definition and an incrementally updated surface when necessary. In the algorithm, shell elements are considered as a subset of hexahedral elements. For clarity, the following discussion is limited to a mesh composed of hexahedral elements. The algorithm requires $6 \times ne$ memory locations for a mesh composed of ne 8-node hexahedral elements and involves two simple steps. The first step is to construct a face id for every element face:

$$\text{faceid} = n_{\text{diag}} + n_{\text{min}} \times \text{nnod} \quad (11)$$

where $nnod$ is the total number of nodes in the problem, n_{min} is the smallest global node number defining the element face, and n_{diag} is the global node number that is diagonal to the smallest global node n_{min} . Step two is to search for all face id's for any non-repeated numbers which will correspond to the faces that are on the surface. The search is efficiently done using the CRAY UNICOS library routine **wheneq** [25]. Figure 18 shows an example of a 3D mesh composed of two hexahedral elements.

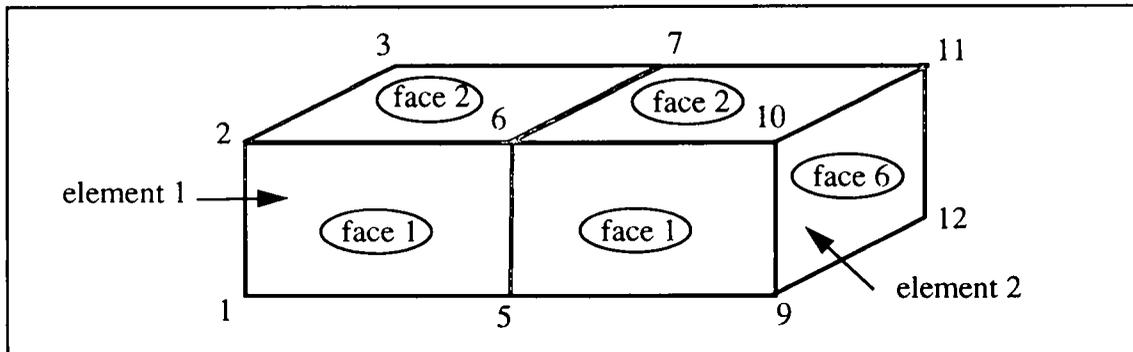


Figure 18. Example mesh for surface definition algorithm

Table 1 lists the 12 element faces in the mesh, their connectivity, n_{min} , n_{diag} , and their face id. In the example problem, all element faces are on the surface except element 1 - face 6 and element 2 - face 5 (which have a the same face id of 67).

element number (iele)	face number (iface)	connectivity	n_{min}	n_{diag}	<i>smap</i> (face id)
1	1	1 5 6 2	1	6	18
1	2	2 6 7 3	2	7	31
1	3	3 7 8 4	3	8	44
1	4	4 8 5 1	1	8	20
1	5	1 2 3 4	1	3	15
1	6	5 8 7 6	5	7	67
2	1	5 9 10 6	5	10	70
2	2	6 10 11 7	6	11	83
2	3	7 11 12 8	7	12	96
2	4	8 12 9 5	5	12	72
2	5	5 6 7 8	5	7	67
2	6	9 12 11 10	9	11	119

The result of collecting all the element faces with unique face ids is a list of the master surfaces. The slave nodes can be determined by looping through the master surface list and flagging the nodal points defined by the master surface connectivity.

The array called *smap* initially stores the faceid of each face, as shown in Table 1. For every element face on the surface (ones with a unique face id), a zero is over-written in $smap(6*(iele-1)+iface)$. For those interior element faces (ones without unique face ids), a pointer to the opposing face is stored in the array, *smap*:

$$smap(6(iele_i - 1) + iface_i) = 6(iele_j - 1) + iface_j$$

$$smap(6(iele_j - 1) + iface_j) = 6(iele_i - 1) + iface_i$$

Using this surface map array, the surface can be incrementally updated as elements are deleted. For the current example there are only two opposing faces, element 1 face 6 and element 2 face 5, so that $smap(6) = 11$, $smap(11) = 6$, and all other positions in *smap* would be zero.

This idea of collecting master surfaces and slave nodes into a heap allows modelling of contact between a variety of finite element types. For example, the nodal points of elements such as beams and trusses can be added to the slave node list. Also, the potential contacts in a problem coupling the finite element method with other methods can be modelled. For example, particle methods such as Smooth Particle Hydrodynamics (SPH) [26] or Particle-In-Cell (PIC) [27] can be easily coupled by adding the particles to the slave node list.

5 Example Problems

The examples considered in this section demonstrate both the improvements in the accuracy of contact detection and the added capabilities that are possible with the global search algorithm. The improvements in the accuracy of contact detection are demonstrated with an example of two elastic blocks contacting and an example of contact induced chatter or ringing. Some of the added capabilities include modelling (1) structures that buckle and fold onto themselves, (2) structures that have materials which tear and create new surfaces, (3) problems where multiple body contact/impact is occurring, and (4) problems where considerable sliding between bodies is occurring. (The necessary files for constructing the input for all example problems are in Appendix 2)

5.1 Contact of Elastic Blocks

The example of two blocks impacting one another is a simple demonstration of the need for an accurate detailed contact check. The new contact algorithm correctly chooses the most opposed master surface for each slave node in the two cases shown in Figures 19 and 20. The two different cases are described as follows:

Case 1. The elastic blocks already discussed in the previous sections and shown in Figure 10 are impacting at a velocity $v_y = 500$ in/s, as shown in Figure 19. Each corner node has the potential of contacting three different master surfaces, and each edge node has the potential of contacting four different master surfaces. The new contact algorithm duplicates the results of the old algorithm by correctly enforcing the most opposed contacts.

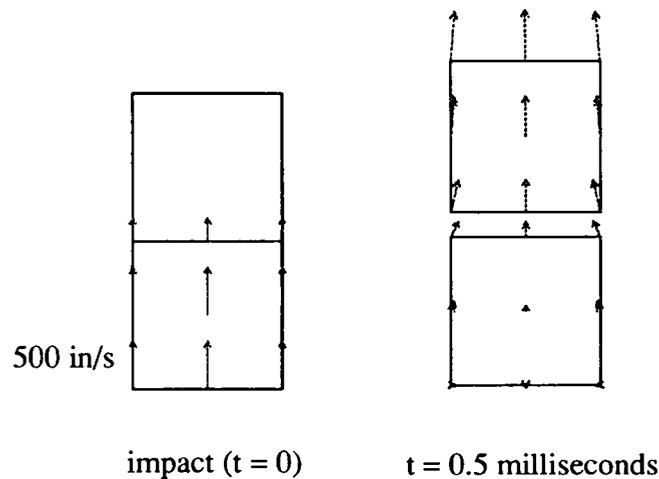


Figure 19. End-on impact of two blocks

Case 2. The same two blocks in Figure 10 are repositioned such that a corner of Block 1 is impacting a corner of Block 2, as shown in Figure 20. The old static contact check, which is based on surface normals, chooses the incorrect master surface causing both blocks to rotate clockwise. By incorrectly determining the pushback direction, a large increase in kinetic energy (from 92.5 in-lbs to 1300 in-lbs) results. The new contact detection algorithm correctly determines the initial contact using the velocity based check and the subsequent updated contact using the static based check. This results in the correct counterclockwise rotation of both blocks.

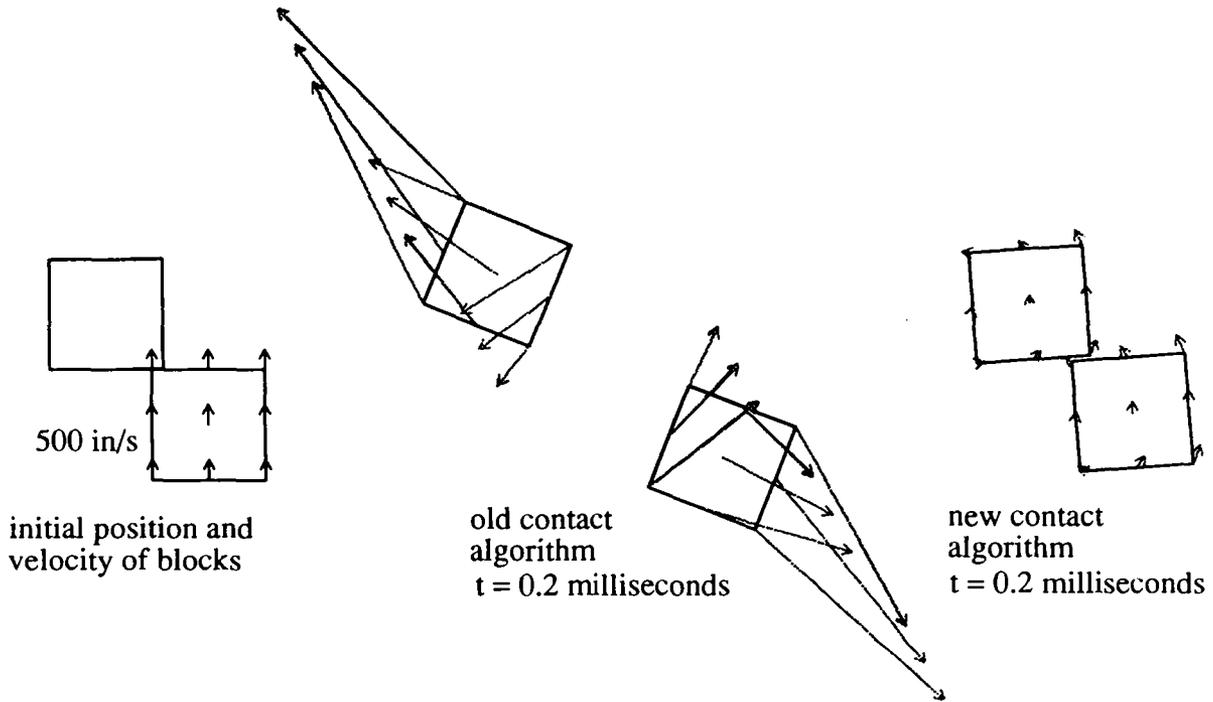


Figure 20. Corner impact of two blocks

5.2 Contact Chatter under High Normal Loads: Pressure Loading of Two Elastic Bodies

One of the difficulties with two curved surfaces contacting each other under high normal loads is the accurate determination of the push back direction. The example shown in Figure 21 has a semicircular rod that is pushed into a semicircular cavity. A pressure load is applied to the flat surface of the rod. All the nodes on each contact surface are initially aligned with each other, so that as the pressure is ramped up, the slave nodes on the semicircle (convex surface) must be pushed back to the vertices of the master surfaces on the cavity (concave surface). The old algorithm incorrectly determines the pushback direction and introduces noise (contact chatter) into the solution. This eventually accumulates over many time steps and results in mesh hourglassing and increased kinetic energy, as shown in Figure 22. The new contact detection algorithm correctly determines the pushback direction and does not introduce noise.

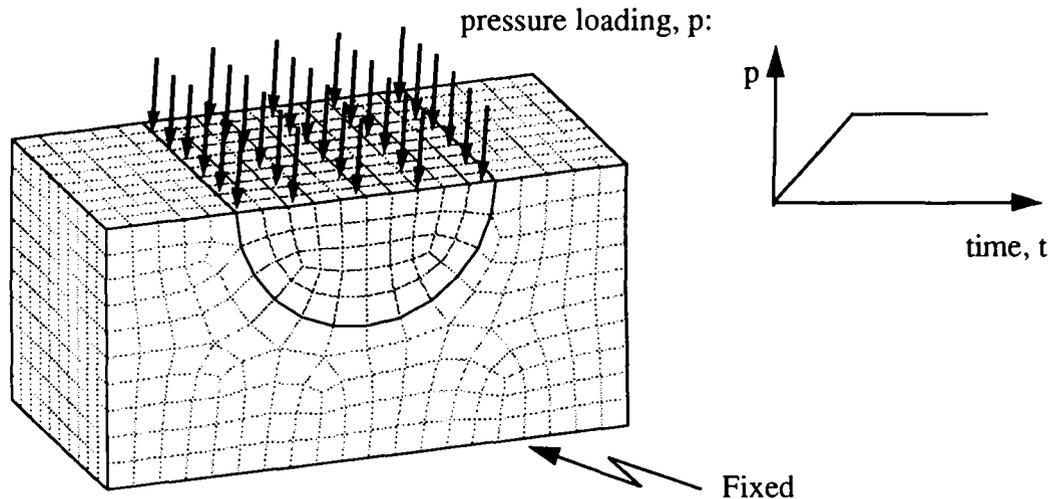
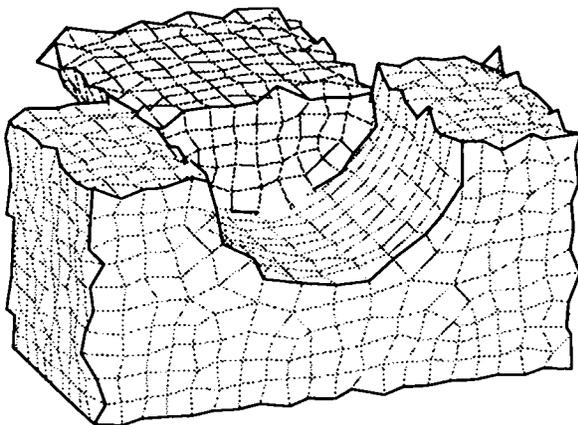
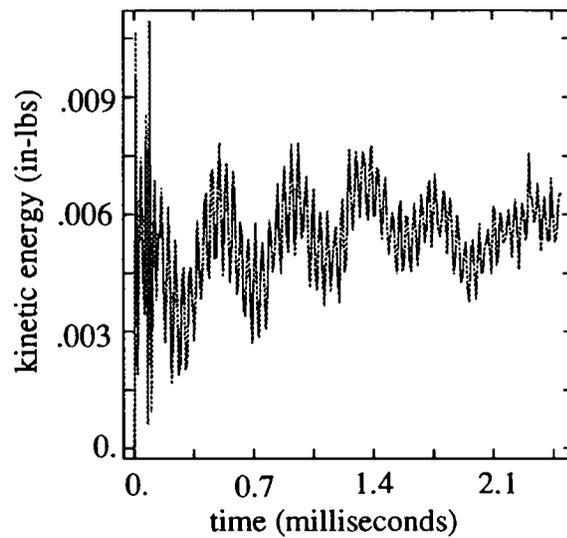
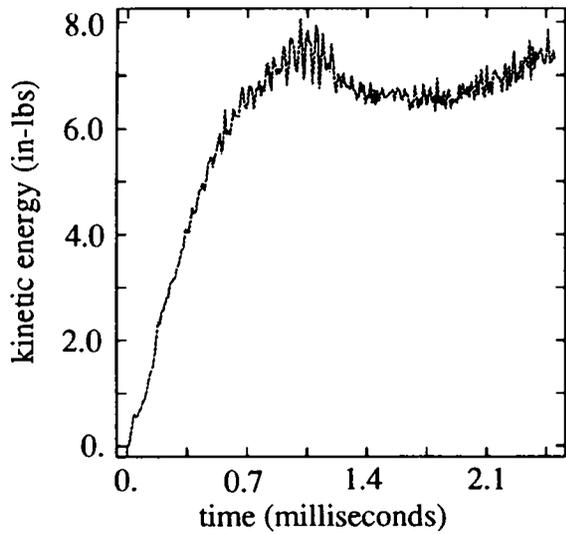
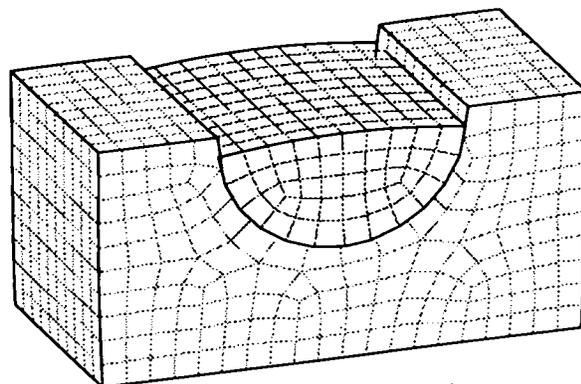


Figure 21. Two elastic bodies contacting under high normal load.



(a) old algorithm



(b) new algorithm

Figure 22. Kinetic energy history and deformed shape (displacements magnified by 100x) using old and new contact detection algorithm

5.3 Self-Contacting Impact: Buckling of Shell-Like Structures

The following example demonstrates the self contacting capability of the contact detection algorithm. This feature is important for modelling crash dynamics where buckling, tearing, and self contact is common. The elastic-plastic shell-like (can) structure shown in Figure 23 is impacted by an elastic-plastic plate. The can is 0.25 inches thick, has an inside radius of 5 inches, and is 15 inches long. The bottom of the can is constrained in all directions. The 22x11 in. plate is 2.5 inches thick and is initially tilted at a 10 degree angle as it impacts the can at 5000 in/s.

The contact detection algorithm currently in PRONTO3D is a surface tracking algorithm that requires surface pairing. The amount of user intervention required to intermittently define and pair many contact surfaces would make this problem impractical to solve using the current algorithm. In contrast, the new algorithm requires two surface ids (one for the plate and one for the can). Contact between any of the slave nodes and master surfaces on both surface sets are detected during the analysis. The deformed shapes at various times are shown in Figure 24. The self-contact of the buckled can is evident in several locations.

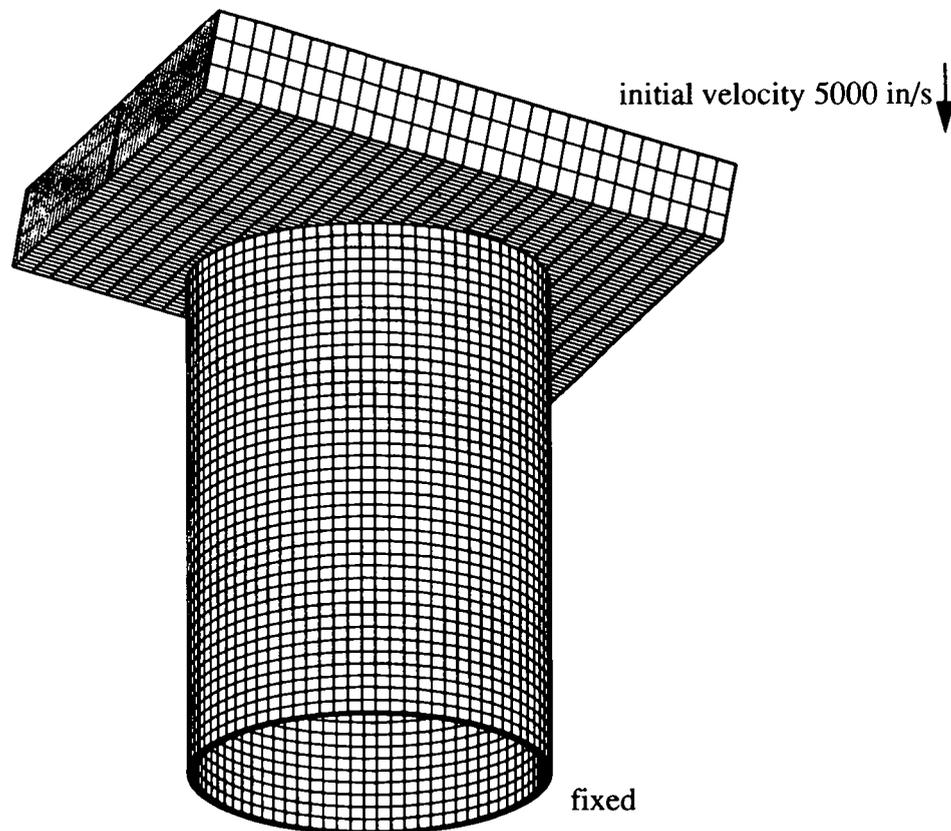
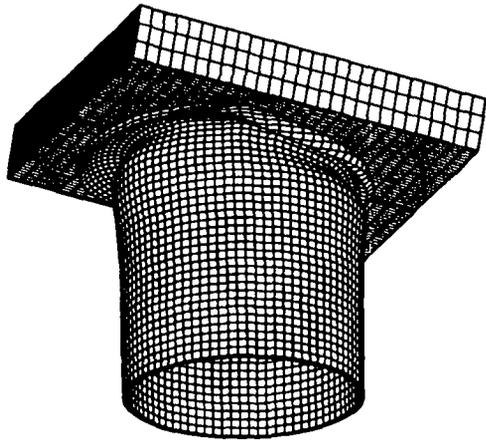
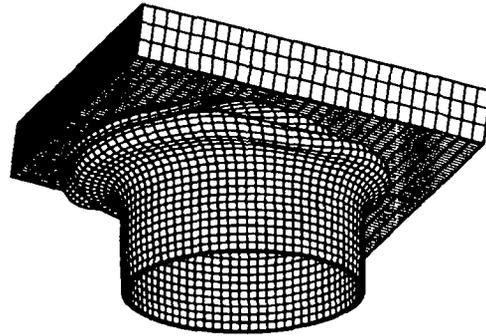


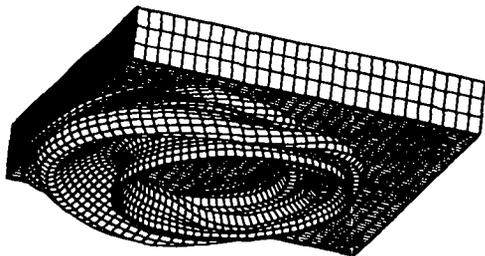
Figure 23. Finite element mesh of an elastic plate impacting an elastic-plastic can



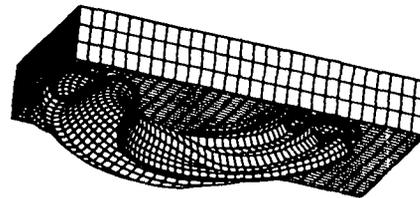
(a) $t = 1.1$ milliseconds



(b) $t = 2.2$ ms



(c) $t = 3.3$ ms



(d) half-section at $t = 3.3$ ms

Figure 24. Deformed meshes of the buckled can at various times

5.4 Automatic Contact Surface Redefinition: Cutting of a Steel Pipe

In the following example, the capability of automatically redefining the contact surface is exercised. For this problem, and others like it, the new surface that is generated as a result of tearing can find itself in contact with other surfaces of the body. The current capabilities of the Sandia codes do not include a periodic redefinition of the contact surface. The new algorithm automatically redefines the surface after any elements are deleted.

The algorithm is demonstrated with a simulation of a pipe cutting process. The process involves a hardened, 0.25 inch thick, wedged-shape cutting blade that is forced through a two inch, schedule 40 pipe that is resting on a support, as shown in Figure 25. The blade initially indents and punctures through one side of the pipe and then progressively tears the pipe walls, as shown in Figure 26. The tear is simulated by deleting elements in which the material damage has accumulated to 0.9. The power law hardening material accumulates damage when it exceeds a failure strain of 1.27 and is loaded under hydrostatic tension [28]. The newly created surface is automatically included in the contact algorithm by redefining the surface after any elements are deleted. As the simulation progresses, the edges of the tear are in contact with the moving cutting blade, as shown in Figure 26. The simulation continues until the blade cuts through the pipe at approximately 3 milliseconds.

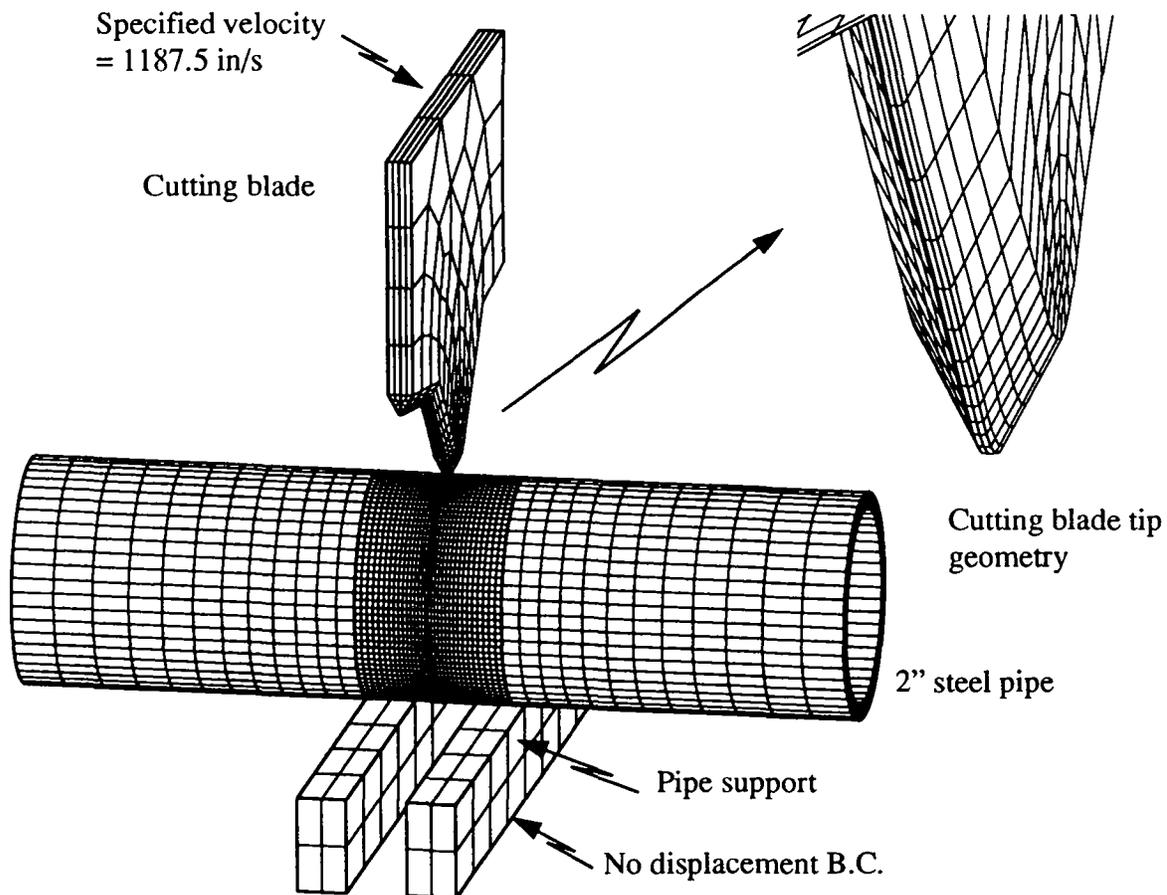


Figure 25. Finite element model for simulating the cutting of a 2 inch steel pipe.

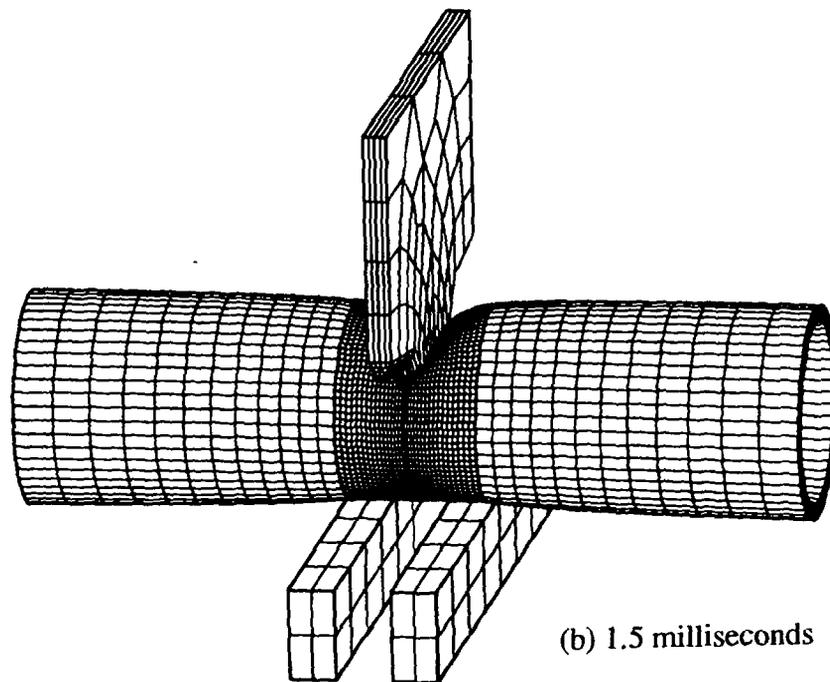
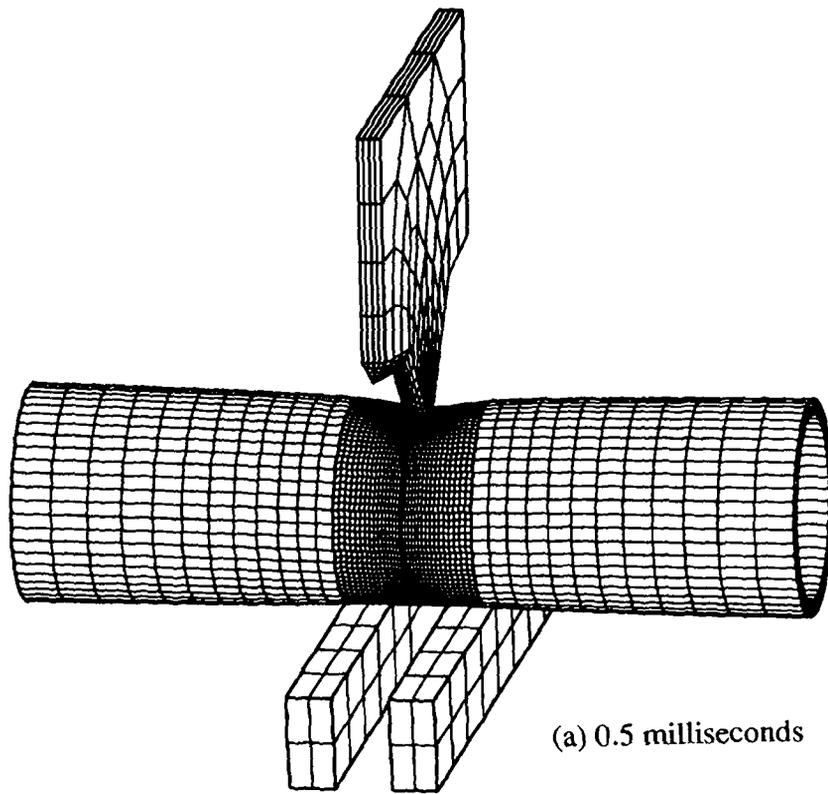


Figure 26. Pipe cutting simulation results at various times

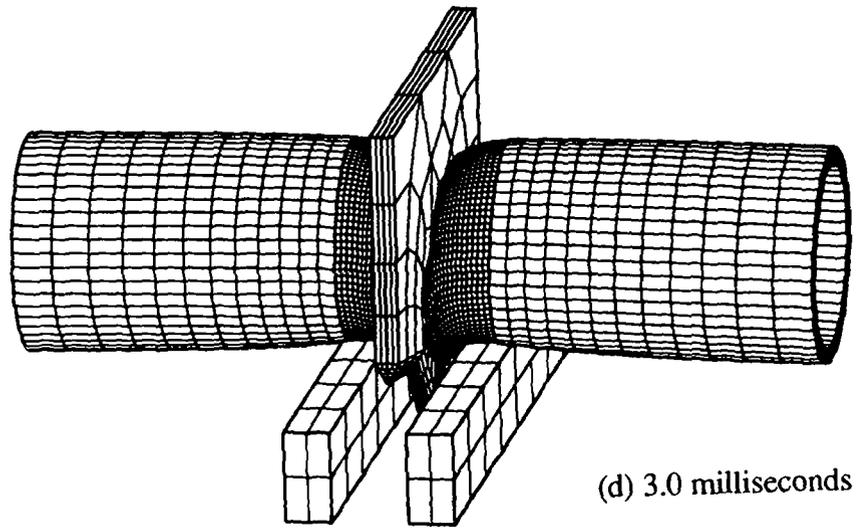
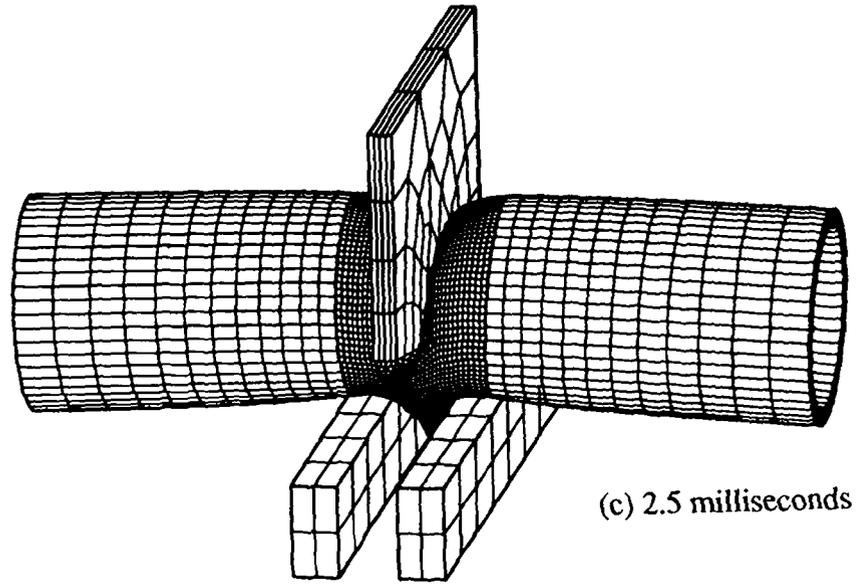


Figure 26 (cont'd). Pipe cutting simulation results at various times

5.5 Multi-Body Impact: Elastic-Plastic Bar impacting Bricks

One of the added capabilities of the new contact detection algorithm is the efficient modelling of multi-body impact without *a-priori* definition of contact surfaces. The example, shown in Figure 27, has an elastic-plastic bar impacting a stack of 17 elastic bricks. A stationary elastic-plastic wall is also resting against the stack of bricks. All slave nodes and master surfaces on the bodies were automatically defined using the algorithm described in Section 4.

For problems where random contact is anticipated, as in this example, each body could potentially impact any other body. For a contact-pairing algorithm, 19^2 contact pairs would be necessary, with each pair having $2n$ slave nodes. For the new global contact searching algorithm, one search with $19n$ slave nodes is necessary. Assuming that each block has $n \approx 50$ slave nodes, 19^2 pairs would require $19^2(2n \log(2n)) = 239,843$ comparisons, whereas the new global contact searching algorithm would require only $19n \log_2(19n) = 9397$ comparisons.

Another important feature of the contact detection algorithm is the spatial independence of the $n \log n$ vectorized sorting algorithm. The speed and efficiency of the algorithm is nearly independent of the location of the bricks. This becomes particularly important late in this example since the volume that the bodies occupy is increasing, as shown in Figure 28.

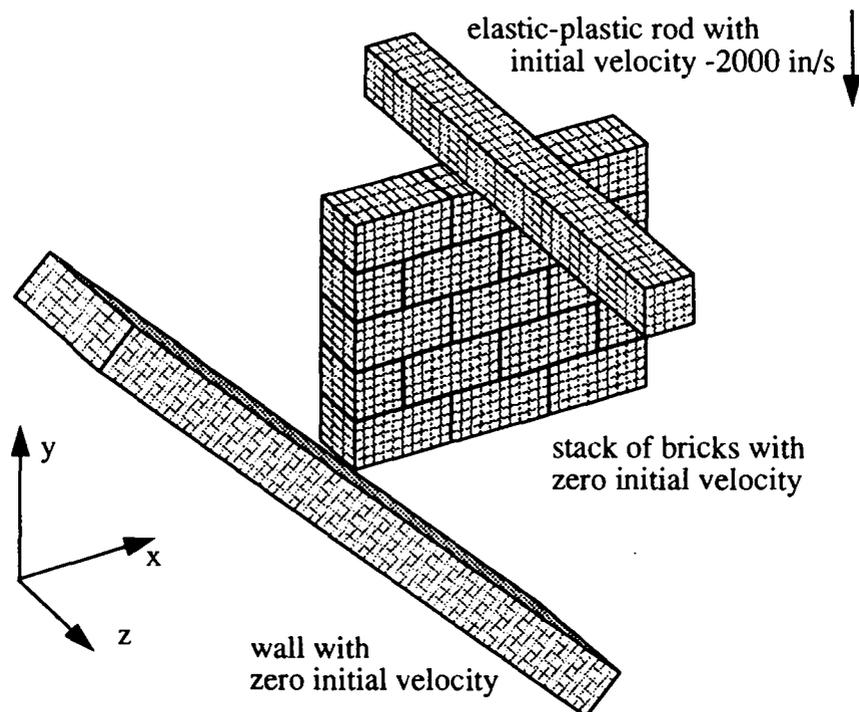


Figure 27. Elastic-plastic bar impacting a stack of 17 elastic bricks

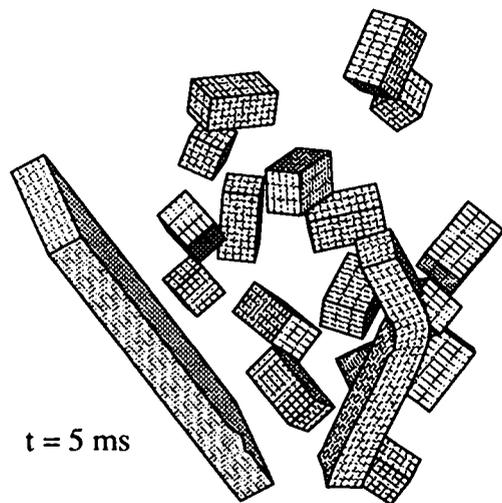
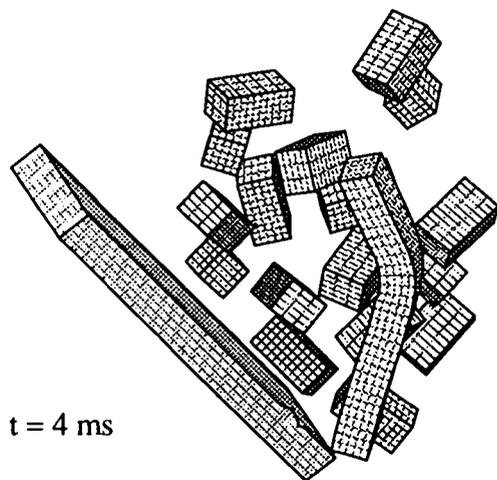
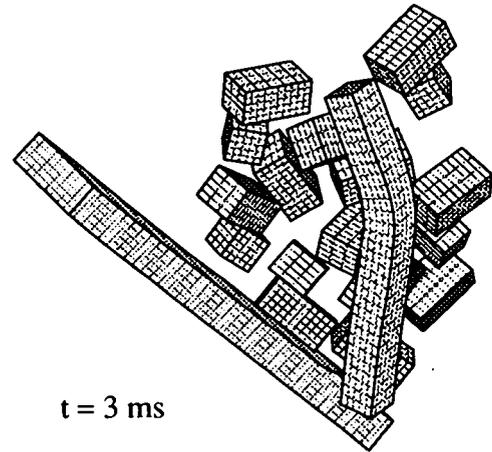
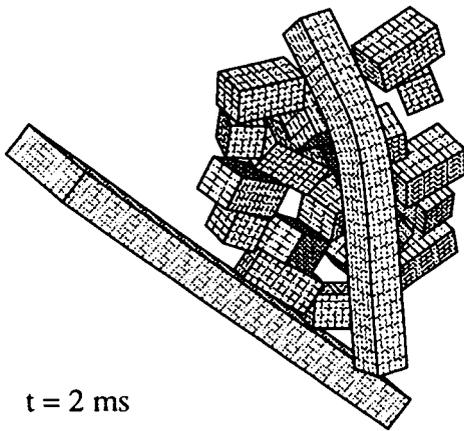
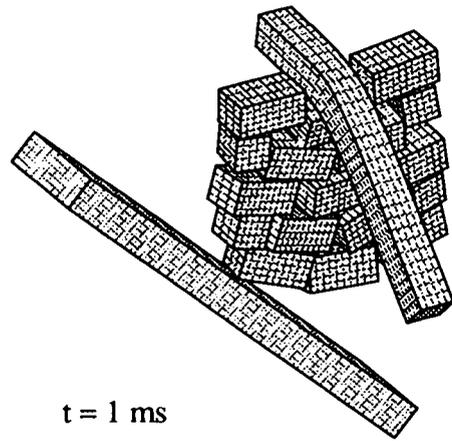
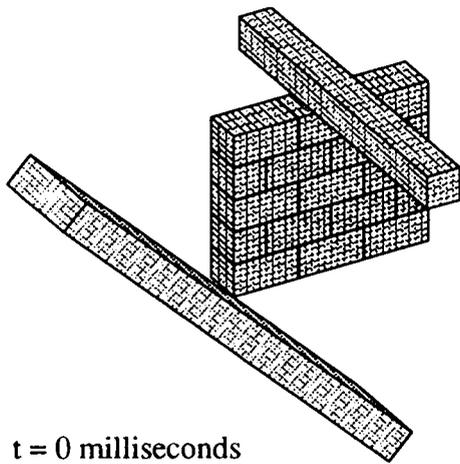


Figure 28. Multi-body impact simulation results at various times

5.6 Large Sliding Contact: Elastic-Plastic Forging of a Copper Billet

The following example demonstrates the capability of modelling large sliding contact typical in large deformation forming and forging. The quasistatic axisymmetric forging of a copper billet shown in Figure 29 was simulated with the program SANTOS. The billet has an initial radius of 2 inches and height of 5 inches. Figure 30 shows a series of deformed meshes as the billet is forged. As the billet is compressed, it slides horizontally until the die cavity is filled. After further compression, the billet is forged around the die corner forming the rivet head.

There are two difficulties related to contact detection in this example. The first is the accurate determination of the pushback direction of the slave node in the lower corner of the die. It is initially constrained on the horizontal master surface as the billet is compressed. Then, just as the billet fills the die cavity, the slave node is constrained on the vertical master surface. Upon further loading, gradual adjustments in the pushback direction are automatically made so that it is constrained to the vertex of the horizontal and vertical master surface. (The current tracking algorithms do not push the slave node back to the vertex of the two master surfaces. This results in contact chatter similar to that described in example 5.2.)

Another difficulty arises from the large sliding contact of the material in the billet around the die corner (shown enlarged in Figure 30). The new contact detection algorithm determines the updated contact point and pushback direction such that the slave nodes on the billet are not given any added artificial velocity.

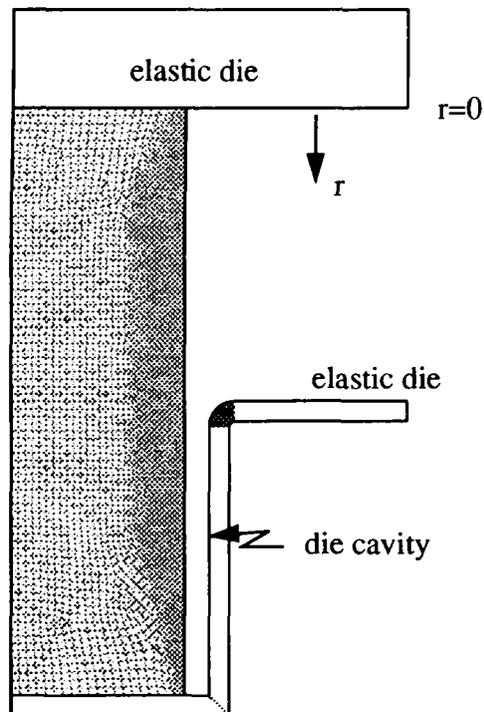


Figure 29. Finite element mesh of a rivet forging process

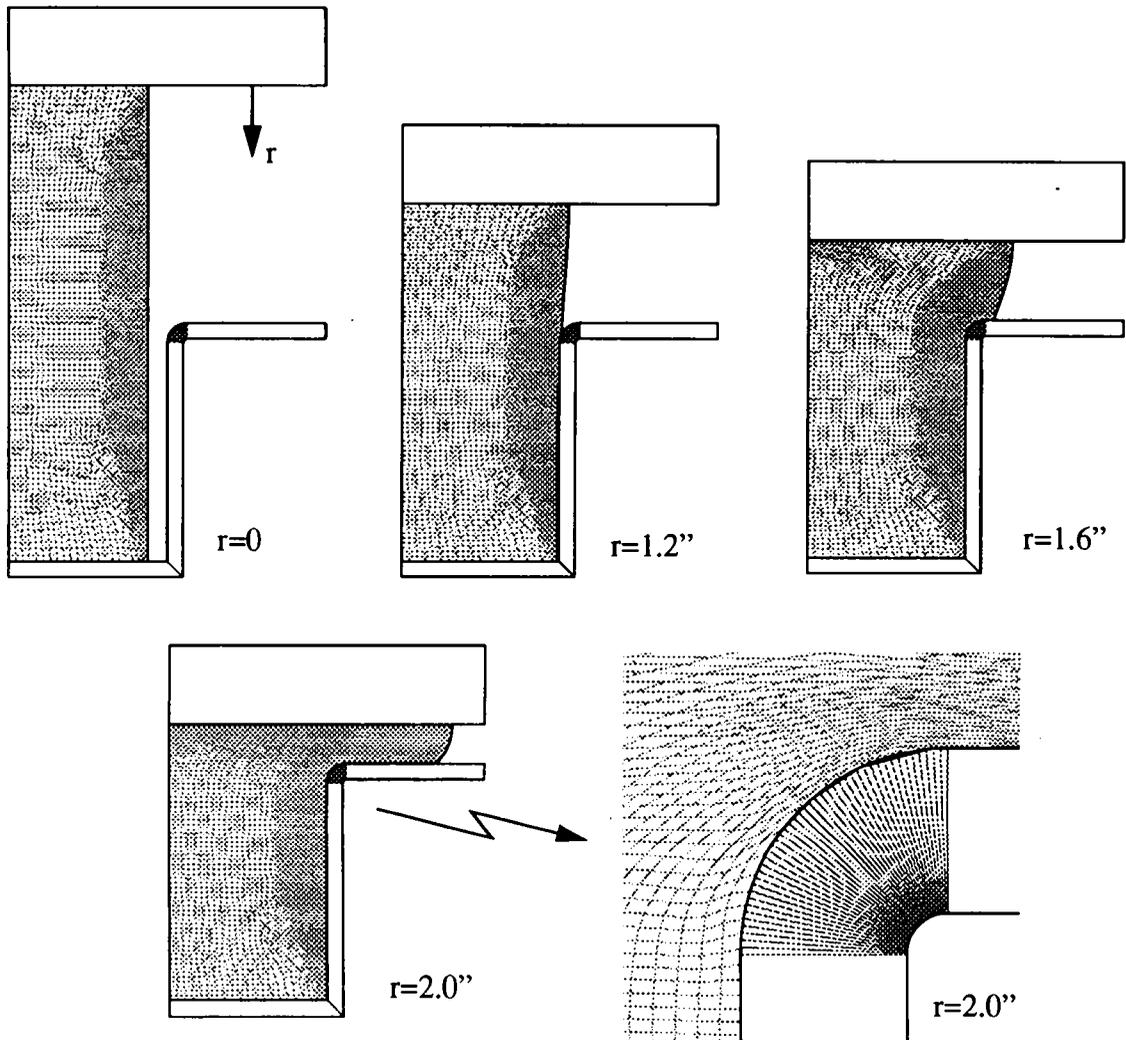


Figure 30. Rivet forging of a copper billet

6 Conclusions

The current contact detection techniques used in the Sandia structural analysis codes have been reviewed. The review has pointed out several areas where improvements are necessary in current contact detection techniques. These areas include neighborhood identification and detailed contact checking. The neighborhood search should not rely on a closest master node to a slave node, and it should be independent of the problem geometry. In several cases, the detailed contact check requires improved accuracy in determining the contact point, penetration, and the pushback direction.

A new contact detection algorithm has been presented that offers improvements in these areas. The improvements are a result of a neighborhood identification strategy that uses a global contact search and a detailed contact check that distinguishes between concave and convex surfaces. The key features of the new contact detection algorithm are:

- The efficient global search allows for global contact. This means that added capability for modelling self contacting structures and eroding or tearing surfaces is now available.
- The known locations of contacting surfaces and their velocities are used to construct a master surface capture box. This guarantees that only physically meaningful contacts are considered in the detailed contact check. The capture box also ensures that a minimum number of slave nodes are paired with the master surface.
- The position and velocity of both the slave node and master surface are considered in determining initial contact. This results in a physically correct determination of the contact location.
- A distinction between a concave and convex surface is made for slave nodes already in contact with a master surface. This results in a more accurate determination of the point of contact, amount of penetration, and the direction of pushback.
- An automatic surface definition algorithm allows for a simplified user input in many cases. One can, for example, include all surfaces of a body by specifying that the material be considered for global contact.

The capabilities of the new contact detection algorithm have been demonstrated with several example problems. Several of the example problems have demonstrated improvements in the accuracy and efficiency of contact determination. Other problems have demonstrated the added capability provided by the new algorithm.

7 References

- 1 Taylor, L.M. and Flanagan, D.P., *PRONTO2D: A Two-Dimensional Transient Solid Dynamics Program*, SAND86-0594, Sandia National Laboratories, Albuquerque, NM 87185, 1987.
- 2 Taylor, L.M. and Flanagan, D.P., *PRONTO3D: A Three-Dimensional Transient Solid Dynamics Program*, SAND89-1912, Sandia National Laboratories, Albuquerque, NM 87185, 1989.
- 3 Attaway, S.W., *Update of PRONTO2D and PRONTO3D Transient Solid Dynamics Programs*, SAND90-0102, Sandia National Laboratories, Albuquerque, NM 87185, 1990.
- 4 Stone, C.M., *SANTOS: A Two-Dimensional Finite Element Program for the Quasistatic Large Deformation, Inelastic Response of Solids*, SAND90-0543, Sandia National Laboratories, Albuquerque, NM 87185, in preparation.
- 5 Biffle, J.H., *JAC - A Two-Dimensional Finite Element Computer Program for the Nonlinear Quasi-Static Response of Solids with the Conjugate Gradient Method*, SAND81-0998, Sandia National Laboratories, Albuquerque, NM 87185, 1984.
- 6 Biffle, J.H., *JAC3D - A Three-Dimensional Finite Element Computer Program for the Nonlinear Quasi-Static Response of Solids with the Conjugate Gradient Method*, SAND87-1305, Sandia National Laboratories, Albuquerque, NM 87185, in preparation.
- 7 Hallquist, J.O., *User's Manual for DYNA2D: An Explicit Two-Dimensional Hydrodynamic Finite Element Code With Interactive Rezoning*, Rev. 2, UCID-18756, Lawrence Livermore National Laboratories, 1984.
- 8 Hallquist, J.O. and Benson, D.J., *User's Manual for DYNA3D: Nonlinear Dynamic Analysis of Structures*, Rev. 3, UCID-19592, Lawrence Livermore National Laboratories, 1987.
- 9 Hibbitt, Karlsson and Sorensen, Inc., *Contact Calculations with ABAQUS - ABAQUS Explicit Users Manual*, Hibbitt, Karlsson and Sorensen, Inc., 1992.
- 10 Hallquist, J.O., *NIKE2D: A Vectorized Implicit, Finite Deformation Finite Element Code for Analyzing the Static and Dynamic Response of 2-D Solids With Interactive Rezoning and Graphics*, UCID-19677, Lawrence Livermore National Laboratories, 1986.
- 11 Hallquist, J.O., *NIKE3D: An Implicit, Finite Deformation, Finite Element Code for Analyzing the Static and Dynamic Response of Three Dimensional Solids*, UCID-18822, Lawrence Livermore National Laboratories, 1984.

- 12 Benson, B.J. and Hallquist, J.O., "A Single Surface Contact Algorithm for the Post-Buckling Analysis of Structures," *Computer Methods in Applied Mechanics and Engineering*, Vol. 78, pp. 141-163, 1990.
- 13 Chaudhary, A.B. and Bathe, K.J., "A Solution Method for Static and Dynamic Analysis of Three-Dimensional Contact Problems with Friction," *Computers and Structures*, Vol. 24, No. 6, pp. 855-873, 1986.
- 14 Zhong, Z.H. and Nilsson, L., "A Contact Searching Algorithm for General 3D Contact-Impact Problems," *Computers and Structures*, Vol. 34, No. 2, pp. 327-335, 1990.
- 15 Harding, D.C., Attaway, S.W., Neilsen, J., Blacker, T.D., and Pierce, J., "Evaluation of Four Multiple Package Crush Environment to the Common Package, Model 1, Plutonium Air Transport Container," SAND92-0278, Sandia National Laboratories, Albuquerque, NM 87185, 1992.
- 16 Belytschko, T. and Lin, J.I., "A Three-Dimensional Impact-Penetration Algorithm with Erosion," *Computers and Structures*, Vol. 25, No. 1, pp. 95-104, 1987.
- 17 Belytschko, T. and Neal, M.O., "Contact-Impact by the Pinball Algorithm with Penalty and Lagrangian Methods," *Int. J. Numerical Methods Eng.*, Vol. 31, pp. 547-572, 1991.
- 18 Metzinger, K., Attaway, S., and Mello, F., "Bobbin Stresses Generated by Wire Winding," *First International Conference on Web Handling*, Oklahoma State University, Stillwater, OK 74078, May 19-22, 1991.
- 19 Swegle, J.W., "Search Algorithm," Memo to Distribution, Sandia National Laboratories, Albuquerque, NM 87185, May 25, 1992.
- 20 Plimpton, S.J., "Molecular Dynamics Simulations of Short-Range Force Systems on 1024-Node Hypercubes," Proceedings of the Fifth Distributed Memory Computing Conference, Charleston, South Carolina, April 8-12, 1990.
- 21 Blacker, T. D., "FASTQ Users Manual Version 1.2," SAND88-1326, Sandia National Laboratories, Albuquerque, NM 87185, June 1988.
- 22 Sjaardema, G. D., "GJOIN: A program for Merging Two or More GENESIS Databases Version 1.4," SAND92-2290, Sandia National Laboratories, Albuquerque, NM 87185, November 1992.
- 23 Gilkey, A. P. and Sjaardema, G. D., "GEN3D: A GENESIS Database 2D to 3D Transformation Program," SAND89-0485, Sandia National Laboratories, Albuquerque, NM 87185, March 1989.
- 24 Sjaardema, G. D., "GENSHELL: A GENESIS Database Shell Transformation Program," Sandia National Laboratories, Albuquerque, NM 87185, In preparation.
- 25 CRAY Research, Inc., Volume 3: UNICOS Math and Scientific Library Reference Manual, SR-2081 5.0, March 1989.

- 26 Monaghan, J.J., "An Introduction to SPH," *Comp. Phys. Comm.*, Vol. 48, pp. 89-98, 1988.
- 27 Harlow, F.H., "PIC and its Progeny," *Comp. Phys. Comm.*, Vol. 48, pp. 1-10, 1988.
- 28 Stone, C.M., Wellman, G.W., and Krieg, R.D., *A Vectorized Elastic/Plastic Power Law Hardening Material Model Including Luders Strain*, SAND90-0153, Sandia National Laboratories, Albuquerque, NM 87185, March 1990.

A1 Appendix 1: Derivation of velocity based contact check

This appendix derives the velocity based detailed contact check that determines the time and point of contact between a moving quadrilateral surface and a moving slave node in 3D.

The equation of a triangular master surface, shown in Figure A1.1, can be written as:

$$a(x - x_1) + b(y - y_1) + c(z - z_1) = 0 \quad (\text{A1.1})$$

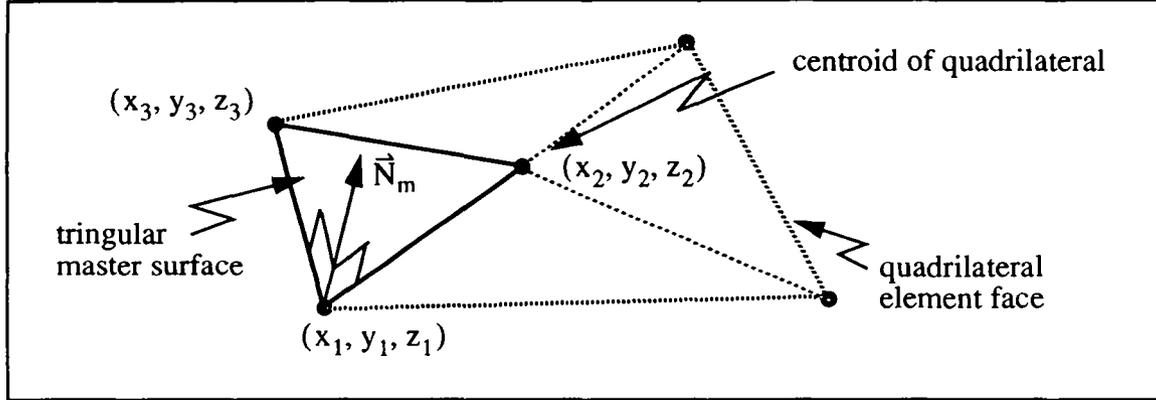


Figure A1.1 Triangular master surface definition on a quadrilateral element face

The point $x_1, y_1,$ and z_1 is a node on the master surface and $a, b,$ and c are components of the master surface normal $\vec{N}_m = a\hat{i} + b\hat{j} + c\hat{k}$:

$$\begin{aligned} a &= [(y_1 - y_3)(z_2 - z_1) - (y_2 - y_1)(z_1 - z_3)] \\ b &= [(x_2 - x_1)(z_1 - z_3) - (x_1 - x_3)(z_2 - z_1)] \\ c &= [(x_1 - x_3)(y_2 - y_1) - (x_2 - x_1)(y_1 - y_3)] \end{aligned} \quad (\text{A1.2})$$

Substituting equations (A1.2) in to equation (A1.1) and simplifying:

$$\begin{aligned} &\{x_1y_3z_2 + x_2y_1z_3 + x_3y_2z_1 - x_1y_2z_3 - x_2y_3z_1 - x_3y_1z_2\} \\ &+ \{ (y_1z_2 + y_2z_3 + y_3z_1 - y_1z_3 - y_2z_1 - y_3z_2) x \\ &\quad (x_1z_3 + x_2z_1 + x_3z_2 - x_1z_2 - x_2z_3 - x_3z_1) y \\ &\quad (x_1y_2 + x_2y_3 + x_3y_1 - x_1y_3 - x_2y_1 - x_3y_2) z \} = 0 \end{aligned} \quad (\text{A1.3})$$

Now, note that the nodal points of the triangular surface are moving, so that:

(4)

$$\begin{aligned} & [(x_i (t + \Delta t), y_i (t + \Delta t), z_i (t + \Delta t)) = \\ & (x_i (t) + \dot{x}_i \Delta t, y_i (t) + \dot{y}_i \Delta t, z_i (t) + \dot{z}_i \Delta t)], i = 1, 3 \end{aligned} \quad (\text{A1.5})$$

If equation (A1.4) is substituted into (A1.3), the result is an equation for a moving planar master surface, as follows:

$$a_0 + a_1 x + a_2 y + a_3 z = 0 \quad (\text{A1.6})$$

where:

$$\begin{aligned} a_0 = & (x_1 y_3 z_2) + (x_1 y_3 \dot{z}_2 + x_1 \dot{y}_3 z_2 + \dot{x}_1 y_3 z_2) \Delta t \\ & + (x_1 \dot{y}_3 \dot{z}_2 + \dot{x}_1 y_3 \dot{z}_2 + \dot{x}_1 \dot{y}_3 z_2) \Delta t^2 + (\dot{x}_1 \dot{y}_3 \dot{z}_2) \Delta t^3 \\ & + (x_2 y_1 z_3) + (x_2 y_1 \dot{z}_3 + x_2 \dot{y}_1 z_3 + \dot{x}_2 y_1 z_3) \Delta t \\ & + (x_2 \dot{y}_1 \dot{z}_3 + \dot{x}_2 y_1 \dot{z}_3 + \dot{x}_2 \dot{y}_1 z_3) \Delta t^2 + (\dot{x}_2 \dot{y}_1 \dot{z}_3) \Delta t^3 \\ & + (x_3 y_2 z_1) + (x_3 y_2 \dot{z}_1 + x_3 \dot{y}_2 z_1 + \dot{x}_3 y_2 z_1) \Delta t \\ & + (x_3 \dot{y}_2 \dot{z}_1 + \dot{x}_3 y_2 \dot{z}_1 + \dot{x}_3 \dot{y}_2 z_1) \Delta t^2 + (\dot{x}_3 \dot{y}_2 \dot{z}_1) \Delta t^3 \\ & + (x_1 y_2 z_3) + (x_1 y_2 \dot{z}_3 + x_1 \dot{y}_2 z_3 + \dot{x}_1 y_2 z_3) \Delta t \\ & + (x_1 \dot{y}_2 \dot{z}_3 + \dot{x}_1 y_2 \dot{z}_3 + \dot{x}_1 \dot{y}_2 z_3) \Delta t^2 + (\dot{x}_1 \dot{y}_2 \dot{z}_3) \Delta t^3 \\ & + (x_2 y_3 z_1) + (x_2 y_3 \dot{z}_1 + x_2 \dot{y}_3 z_1 + \dot{x}_2 y_3 z_1) \Delta t \\ & + (x_2 \dot{y}_3 \dot{z}_1 + \dot{x}_2 y_3 \dot{z}_1 + \dot{x}_2 \dot{y}_3 z_1) \Delta t^2 + (\dot{x}_2 \dot{y}_3 \dot{z}_1) \Delta t^3 \\ & + (x_3 y_1 z_2) + (x_3 y_1 \dot{z}_2 + x_3 \dot{y}_1 z_2 + \dot{x}_3 y_1 z_2) \Delta t \\ & + (x_3 \dot{y}_1 \dot{z}_2 + \dot{x}_3 y_1 \dot{z}_2 + \dot{x}_3 \dot{y}_1 z_2) \Delta t^2 + (\dot{x}_3 \dot{y}_1 \dot{z}_2) \Delta t^3 \end{aligned}$$

$$\begin{aligned} a_1 = & (y_1 z_2) + (y_1 \dot{z}_2 + \dot{y}_1 z_2) \Delta t + (\dot{y}_1 \dot{z}_2) \Delta t^2 \\ & + (y_2 z_3) + (y_2 \dot{z}_3 + \dot{y}_2 z_3) \Delta t + (\dot{y}_2 \dot{z}_3) \Delta t^2 \\ & + (y_3 z_1) + (y_3 \dot{z}_1 + \dot{y}_3 z_1) \Delta t + (\dot{y}_3 \dot{z}_1) \Delta t^2 \\ & + (y_1 z_3) + (y_1 \dot{z}_3 + \dot{y}_1 z_3) \Delta t + (\dot{y}_1 \dot{z}_3) \Delta t^2 \\ & + (y_2 z_1) + (y_2 \dot{z}_1 + \dot{y}_2 z_1) \Delta t + (\dot{y}_2 \dot{z}_1) \Delta t^2 \\ & + (y_3 z_2) + (y_3 \dot{z}_2 + \dot{y}_3 z_2) \Delta t + (\dot{y}_3 \dot{z}_2) \Delta t^2 \end{aligned}$$

$$\begin{aligned}
a_2 = & (x_1 z_3) + (x_1 \dot{z}_3 + \dot{x}_1 z_3) \Delta t + (\ddot{x}_1 \dot{z}_3) \Delta t^2 \\
& + (x_2 z_1) + (x_2 \dot{z}_1 + \dot{x}_2 z_1) \Delta t + (\ddot{x}_2 \dot{z}_1) \Delta t^2 \\
& + (x_3 z_2) + (x_3 \dot{z}_2 + \dot{x}_3 z_2) \Delta t + (\ddot{x}_3 \dot{z}_2) \Delta t^2 \\
& + (x_1 z_2) + (x_1 \dot{z}_2 + \dot{x}_1 z_2) \Delta t + (\ddot{x}_1 \dot{z}_2) \Delta t^2 \\
& + (x_2 z_3) + (x_2 \dot{z}_3 + \dot{x}_2 z_3) \Delta t + (\ddot{x}_2 \dot{z}_3) \Delta t^2 \\
& + (x_3 z_1) + (x_3 \dot{z}_1 + \dot{x}_3 z_1) \Delta t + (\ddot{x}_3 \dot{z}_1) \Delta t^2
\end{aligned}$$

$$\begin{aligned}
a_3 = & (x_1 y_2) + (x_1 \dot{y}_2 + \dot{x}_1 y_2) \Delta t + (\ddot{x}_1 \dot{y}_2) \Delta t^2 \\
& + (x_2 y_3) + (x_2 \dot{y}_3 + \dot{x}_2 y_3) \Delta t + (\ddot{x}_2 \dot{y}_3) \Delta t^2 \\
& + (x_3 y_1) + (x_3 \dot{y}_1 + \dot{x}_3 y_1) \Delta t + (\ddot{x}_3 \dot{y}_1) \Delta t^2 \\
& + (x_1 y_3) + (x_1 \dot{y}_3 + \dot{x}_1 y_3) \Delta t + (\ddot{x}_1 \dot{y}_3) \Delta t^2 \\
& + (x_2 y_1) + (x_2 \dot{y}_1 + \dot{x}_2 y_1) \Delta t + (\ddot{x}_2 \dot{y}_1) \Delta t^2 \\
& + (x_3 y_2) + (x_3 \dot{y}_2 + \dot{x}_3 y_2) \Delta t + (\ddot{x}_3 \dot{y}_2) \Delta t^2
\end{aligned}$$

The slave node is also moving, so that its position can be expressed as:

$$\begin{aligned}
& (x_s(t + \Delta t), y_s(t + \Delta t), z_s(t + \Delta t)) = \\
& (x_s(t) + \dot{x}_s \Delta t, y_s(t) + \dot{y}_s \Delta t, z_s(t) + \dot{z}_s \Delta t)
\end{aligned} \tag{A1.7}$$

Following these definitions, we seek a time Δt such that the slave node will be co-planar with the master surface. Such a time can be found (if it exists) by substituting equations (A1.6), into equation (A1.5) and solving the following cubic equation in Δt :

$$b_0 + b_1 \Delta t + b_2 \Delta t^2 + b_3 \Delta t^3 = 0 \tag{A1.8}$$

where:

$$\begin{aligned}
b_0 = & x_1 \{ y_2 (z_s - z_3) + y_3 (z_2 - z_s) + y_s (z_3 - z_2) \} \\
& + x_2 \{ y_1 (z_3 - z_s) + y_s (z_1 - z_3) + y_3 (z_s - z_1) \} \\
& + x_3 \{ y_s (z_2 - z_1) + y_2 (z_1 - z_s) + y_1 (z_s - z_2) \} \\
& + x_s \{ y_1 (z_2 - z_3) + y_2 (z_3 - z_1) + y_3 (z_1 - z_2) \}
\end{aligned}$$

$$\begin{aligned}
b_1 = & x_1 \{ y_2 (\dot{z}_s - \dot{z}_3) + y_3 (\dot{z}_2 - \dot{z}_s) + y_s (\dot{z}_3 - \dot{z}_2) \\
& \dot{y}_2 (z_s - z_3) + \dot{y}_3 (z_2 - z_s) + \dot{y}_s (z_3 - z_2) \} \\
& + x_2 \{ y_1 (\dot{z}_3 - \dot{z}_s) + y_s (\dot{z}_1 - \dot{z}_3) + y_3 (\dot{z}_s - \dot{z}_1) \\
& \dot{y}_1 (z_3 - z_s) + \dot{y}_s (z_1 - z_3) + \dot{y}_3 (z_s - z_1) \} \\
& + x_3 \{ y_s (\dot{z}_2 - \dot{z}_1) + y_2 (\dot{z}_1 - \dot{z}_s) + y_1 (\dot{z}_s - \dot{z}_2) \\
& \dot{y}_s (z_2 - z_1) + \dot{y}_2 (z_1 - z_s) + \dot{y}_1 (z_s - z_2) \} \\
& + x_s \{ y_1 (\dot{z}_2 - \dot{z}_3) + y_2 (\dot{z}_3 - \dot{z}_1) + y_3 (\dot{z}_1 - \dot{z}_2) \\
& \dot{y}_1 (z_2 - z_3) + \dot{y}_2 (z_3 - z_1) + \dot{y}_3 (z_1 - z_2) \} \\
& + \dot{x}_1 \{ y_2 (z_s - z_3) + y_3 (z_2 - z_s) + y_s (z_3 - z_2) \} \\
& + \dot{x}_2 \{ y_1 (z_3 - z_s) + y_s (z_1 - z_3) + y_3 (z_s - z_1) \} \\
& + \dot{x}_3 \{ y_s (z_2 - z_1) + y_2 (z_1 - z_s) + y_1 (z_s - z_2) \} \\
& + \dot{x}_s \{ y_1 (z_2 - z_3) + y_2 (z_3 - z_1) + y_3 (z_1 - z_2) \}
\end{aligned}$$

$$\begin{aligned}
b_2 = & \dot{x}_1 \{ y_2 (\dot{z}_s - \dot{z}_3) + y_3 (\dot{z}_2 - \dot{z}_s) + y_s (\dot{z}_3 - \dot{z}_2) \\
& \dot{y}_2 (z_s - z_3) + \dot{y}_3 (z_2 - z_s) + \dot{y}_s (z_3 - z_2) \} \\
& + \dot{x}_2 \{ y_1 (\dot{z}_3 - \dot{z}_s) + y_s (\dot{z}_1 - \dot{z}_3) + y_3 (\dot{z}_s - \dot{z}_1) \\
& \dot{y}_1 (z_3 - z_s) + \dot{y}_s (z_1 - z_3) + \dot{y}_3 (z_s - z_1) \} \\
& + \dot{x}_3 \{ y_s (\dot{z}_2 - \dot{z}_1) + y_2 (\dot{z}_1 - \dot{z}_s) + y_1 (\dot{z}_s - \dot{z}_2) \\
& \dot{y}_s (z_2 - z_1) + \dot{y}_2 (z_1 - z_s) + \dot{y}_1 (z_s - z_2) \} \\
& + \dot{x}_s \{ y_1 (\dot{z}_2 - \dot{z}_3) + y_2 (\dot{z}_3 - \dot{z}_1) + y_3 (\dot{z}_1 - \dot{z}_2) \\
& \dot{y}_1 (z_2 - z_3) + \dot{y}_2 (z_3 - z_1) + \dot{y}_3 (z_1 - z_2) \} \\
& + x_1 \{ \dot{y}_2 (\dot{z}_s - \dot{z}_3) + \dot{y}_3 (\dot{z}_2 - \dot{z}_s) + \dot{y}_s (\dot{z}_3 - \dot{z}_2) \} \\
& + x_2 \{ \dot{y}_1 (\dot{z}_3 - \dot{z}_s) + \dot{y}_s (\dot{z}_1 - \dot{z}_3) + \dot{y}_3 (\dot{z}_s - \dot{z}_1) \} \\
& + x_3 \{ \dot{y}_s (\dot{z}_2 - \dot{z}_1) + \dot{y}_2 (\dot{z}_1 - \dot{z}_s) + \dot{y}_1 (\dot{z}_s - \dot{z}_2) \} \\
& + x_s \{ \dot{y}_1 (\dot{z}_2 - \dot{z}_3) + \dot{y}_2 (\dot{z}_3 - \dot{z}_1) + \dot{y}_3 (\dot{z}_1 - \dot{z}_2) \}
\end{aligned}$$

$$\begin{aligned}
b_3 = & \dot{x}_1 \{ \dot{y}_2 (\dot{z}_s - \dot{z}_3) + \dot{y}_3 (\dot{z}_2 - \dot{z}_s) + \dot{y}_s (\dot{z}_3 - \dot{z}_2) \} \\
& + \dot{x}_2 \{ \dot{y}_1 (\dot{z}_3 - \dot{z}_s) + \dot{y}_s (\dot{z}_1 - \dot{z}_3) + \dot{y}_3 (\dot{z}_s - \dot{z}_1) \} \\
& + \dot{x}_3 \{ \dot{y}_s (\dot{z}_2 - \dot{z}_1) + \dot{y}_2 (\dot{z}_1 - \dot{z}_s) + \dot{y}_1 (\dot{z}_s - \dot{z}_2) \} \\
& + \dot{x}_s \{ \dot{y}_1 (\dot{z}_2 - \dot{z}_3) + \dot{y}_2 (\dot{z}_3 - \dot{z}_1) + \dot{y}_3 (\dot{z}_1 - \dot{z}_2) \}
\end{aligned}$$

and

Suppose a solution for $\Delta t = \Delta t_c$ is found. Then it also must satisfy two conditions for it to be considered a contact. The first is that the time Δt_c occurs in the next time step increment, $0 \leq \Delta t_c \leq dt$. The other condition is that the contact point:

$$(x_c, y_c, z_c) = (x_s + \dot{x}_s \Delta t_c, y_s + \dot{y}_s \Delta t_c, z_s + \dot{z}_s \Delta t_c) \quad (A1.9)$$

must lie inside the master surface edges. This can be determined exactly by computing the local ξ, η coordinates of the contact point on the master surface. Figure A1.2 shows how an initial estimate ξ_0, η_0 of the local coordinates can be computed. Four triangles are constructed by connecting the contact point with the four corners of the quadrilateral master surface. The contact point is inside the master surface when all four areas $A_1, A_2, A_3,$ and A_4 are all positive. If it is inside the master surface, then an initial estimate of the local coordinates of the contact point can be found, as shown by the equations in Figure A1.2.

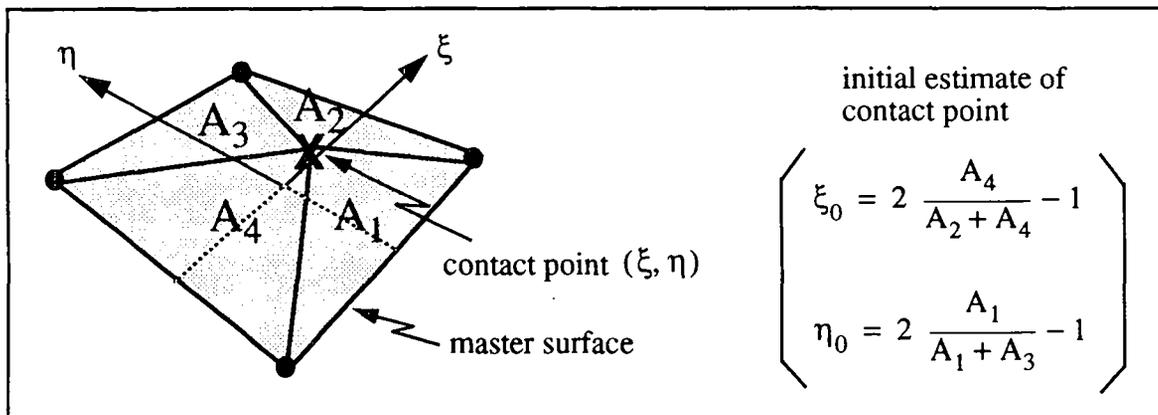


Figure A1.2 Initial estimates for the local coordinates of a contact point

The logic for these equations is based simply on the observation that $(\xi, \eta) = (\xi_0, \eta_0)$ is computed exactly for any rectangular quadrilateral and that the limiting values of ξ and $\eta = \pm 1$ are computed exactly for any shaped quadrilateral. If the quadrilateral surface is a rectangle, then the proof is simple. If it is not, then by observation ξ is computed exactly only when one of the areas A_2 or A_4 is zero. Likewise η is computed exactly only when one of the areas A_1 or A_3 is zero. For example, suppose A_2 is zero then the estimated coordinate $\xi = \xi_0 = 1$ is exact. If A_4 is zero then the estimated coordinate $\xi = \xi_0 = -1$ is also exact. If neither A_2 or A_4 are zero, then the equation gives a reasonable estimate of the contact point ξ . Improvements in the accuracy of the local coordinates can be achieved by performing Newton iterations on the nonlinear equations:

$$\begin{Bmatrix} \xi \\ \eta \end{Bmatrix} = \begin{bmatrix} \sum_{j=1}^4 x_j \xi_j & \sum_{j=1}^4 x_j \eta_j \\ \sum_{j=1}^4 y_j \xi_j & \sum_{j=1}^4 y_j \eta_j \end{bmatrix} \begin{Bmatrix} 4x - \sum_{j=1}^4 (1 + \xi_j \eta_j \xi \eta) x_j \\ 4y - \sum_{j=1}^4 (1 + \xi_j \eta_j \xi \eta) y_j \end{Bmatrix} \quad (\text{A1.10})$$

with the seed $(\xi, \eta) = (\xi_0, \eta_0)$. Upon converging on the local coordinates of the contact point ξ, η with $\zeta = (+ \text{ or } -) 1$, they should satisfy

$$x_c = \sum_{j=1}^8 N_j(\xi, \eta, \zeta) x_j, \quad y_c = \sum_{j=1}^8 N_j(\xi, \eta, \zeta) y_j, \quad z_c = \sum_{j=1}^8 N_j(\xi, \eta, \zeta) z_j \quad (\text{A1.11})$$

where $N_j(\xi, \eta, \zeta)$, $j=1,8$ are the eight shape functions for a hexahedral element.

A2 Appendix 2: User instructions and example input files

Reduced user input is a convenience that is an outcome of the global searching and automatic surface definition capability. One can, for example, include all surfaces of a body by specifying that the material be considered for global contact, or selectively include only a subset of the surface. Figure A2.1 shows the typical input required for a problem. Reducing the input required from the user minimizes the potential number of mistakes, which minimizes cost.

```
contact material 1
contact material 2
contact material 5
contact surface 100
contact data material 2, material 5, friction = .1
contact data material 1, surface 100, friction = 0.3, beta = 0.1
```

Surface 100, materials 1, 2 and 5 will be globally searched for contacts.

Contacts between material 2 and 5 will have a coefficient of friction equal to 0.1

Contacts between material 1 and contact surface 100 will have a coefficient of friction equal to 0.1 and a master slave partitioning of 0.1

Figure A2.1. Typical contact surface user input.

A2.1 PRONTO User Instructions

Listed below are updated and new keywords supported by the PRONTO 2D and 3D command language for using by the new contact detection algorithm. The uppercase letters represent the minimum abbreviation of each word.

- **CONtact SURface**
- **CONtact MATerial**
- **CONtact DATa**

The **CONtact SURface** command now has two distinct uses, a paired side set contact and a global contact. The paired side set contact is unchanged in its algorithm logic and usage. The global contact uses the new global contact detection algorithm and can be used to model a self contacting surface, for example.

A2.1.1 Paired Side Set Contact

CONTACT SURface, side 1 id, side 2 id, FIXED, β , toler

side 1 id	must match a side set on the GENESIS file
side 2 id	must match a side set on the GENESIS file
FIXED	-1 or the word FIXED will tie the contacts together
β	kinematic partition factor (default=0.5)
toler	tolerance for determining fixed contacts

CONTACT SURface, side 1 id, side 2 id, μ_0 , β , μ_1 , γ

side 1 id	must match a side set on the GENESIS file
side 2 id	must match a side set on the GENESIS file
μ_0	static coefficient of friction (default=0.0)
β	kinematic partition factor (default=0.5)
μ_1	high velocity coefficient of friction (default=0.0)
γ	velocity decay coefficient

A contact condition is enforced between the two surfaces defined by the respective side sets. The kinematic partition is a relative weighting of the master slave relationship of the two surfaces. A value of zero implies that the first surface (defined by side 1 id) acts only as a master and the second surface acts only as a slave. A value of one reverses these roles. The default value (0.5) gives a symmetric treatment of the contact. If one surface is much more massive than the other, this variable should be adjusted so that it is treated as the master. By massive, we mean that the surface either has a higher material density and/or a coarser mesh refinement.

A2.1.2 Global Contact

CONTACT SURface, side 1 id

side 1 id must match a side set on the GENESIS file

A contact condition is enforced between a surface contacting itself *and* other surfaces defined using the single side set **CONTACT SURface** keyword *and* those surfaces defined using the **CONTACT MATERIAL** keyword.

CONTACT MATERIAL, material id 1

material 1 id must match a material id on the GENESIS file. If no material id is specified then all materials are included.

All surfaces associated with the material id are automatically determined and considered for contact with itself *and* other surfaces defined by the single side set **CONTACT SURface** keyword

and any additional surfaces defined as a result of repeated use of **CONtact MATerial** keyword. If the material id has an element death option, the surface will be automatically redefined *as elements die*. Note that the automatic surface redefinition is only done for those surfaces defined by the **CONtact MATerial** keyword and not those defined by the **CONtact SURface** keyword.

A new command called **CONtact DATA** is added so that friction and partitioning factors can be defined between contact surfaces and materials if required. Note that default values are set.

CONtact DATA, id 1, id 2, μ_0 , β , μ_1 , γ

id 1	must be either “ SURface surface id” or “ MATerial material id”
id 2	must be either “ SURface surface id” or “ MATerial material id”
μ_0	static coefficient of friction (default=0.0)
β	kinematic partition factor (default=0.5)
μ_1	high velocity coefficient of friction (default=0.0)
γ	velocity decay coefficient

All surfaces associated with the surface id/material id pair use the contact friction conditions and kinematic partitioning factor specified. If a part of a surface is multiply defined by a material id and a surface id, the contact data specification using the surface id will override the material id specification.

A2.2 SANTOS and JAC User Instructions

Both SANTOS and JAC have the partitioned contact detection algorithm installed. At this moment contact surface pairs are still required (in the same manner outlined in the user instructions for the respective codes). The improvements in the accuracy of the detailed contact check are available in the updated versions of these codes. After the development of a partitioned contact enforcement algorithm, the benefit of using the global contact search can be available.

A2.3 Input files

This section provides a collection of the FASTQ, GEN3D, GREPOS, GJOIN, and PRONTO (or SANTOS) inputs for the example problems presented in this report.

A2.3.1 Contact of Elastic Blocks

makefile:

```
bricks.g2d: brick.fsq brick.size
fastq -aprepro -m bricks.g2d brick.fsq

brick.g: bricks.g2d brick.size bricks.gen3d
gen3d -aprepro bricks.g2d brick.g <bricks.gen3d

brick1.g: brick.g brick1.grp brick.size
grepos -aprepro brick.g brick1.g<brick1.grp

brick2.g: brick.g brick2.grp brick.size
grepos -aprepro brick.g brick2.g<brick2.grp

bricks.g: brick.g brick1.g brick2.g bricks.gjn
gjoin <bricks.gjn
```

FASTQ input:

```
title
contact test problem: two blocks impacting
(include(brick.size))
point 1 0. 0.
point 2 {xlen} 0.
point 3 {xlen} {ylen}
point 4 {0.} {ylen}
line 1 str 1 2 0 {ix}
line 2 str 2 3 0 {iy}
line 3 str 3 4 0 {ix}
line 4 str 4 1 0 {iy}
region 1 1 -1 -2 -3 -4
sidebc 300 1 2 3 4
exit
```

brick.size:

```
$ {xlen=1} {ylen=1} {zlen=1} {ix = 2} {iy=2} {iz = 2}
```

GEN3D file:

```
(include(brick.size))
translate {iz} {zlen}
ssets back 100
ssets front 200
exit
```

GREPOS file: brick 1:

```
{include(brick.size)}
offset 0. 0. 0.
change sideset 100 101
change sideset 200 201
change sideset 300 301
exit
```

GREPOS file: brick 2:

<u>Case 1</u>	<u>Case 2</u>
<pre>{include(brick.size)} offset (-0.9*xlen) (1.0*ylen) 0. change sideset 100 101 change sideset 200 201 change sideset 300 301 exit</pre>	<pre>{include(brick.size)} offset 0. (1.0*ylen) 0. change sideset 100 101 change sideset 200 201 change sideset 300 301 exit</pre>

GJOIN file:

```
brick1.g
brick2.g
N
SSETS
COMBINE 101 201 301
COMBINE 102 202 302
EXIT
EXIT
NO
bricks.g
```

PRONTO3D input file:

```
TITLE
  FIXED CONTACT TEST PROB
TERMINATION TIME = .5E-3
PLOT TIME = .1e-5
OUTPUT TIME = .1e-5

MATERIAL, 1, ELASTIC, .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3333
END
MATERIAL, 2, ELASTIC, .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3333
END

INITIAL VELOCITY MATERIAL 1 0., 500., 0.
INITIAL VELOCITY MATERIAL 2 0., 0., 0.

PLOT ELEMENT =
PLOT NODAL DISPLACEMENT, REACTION, VELOCITY, ACCELERATION

CONTACT SURFACE 101
CONTACT SURFACE 102

EXIT
```

A2.3.2 Pressure Loading of Two Elastic Bodies

FASTQ input:

```
title
contact chatter
$ (r=.5) (hx=1) (hy=1)

point 1 0. 0.

point 2 {r} 0.
point 3 {-r} 0.

line 1 str 1 2 0 5
line 2 circ 2 3 1 12
line 3 str 3 1 0 5

region 1 1 -1 -2 -3

point 10 {r} {0}
point 11 {hx} {0}
point 12 {hx} {hy}
point 13 {-hx} {hy}
point 14 {-hx} {0}
point 15 {-r} {0}

line 10 str 10 11 0 5
line 11 str 11 12 0 10
line 12 str 12 13 0 20
line 13 str 13 14 0 10
line 14 str 14 15 0 5
line 15 circ 10 15 1 12

region 2 2 -10 -11 -12 -13 -14 -15

scheme 1 c6s
scheme 2 x

sidebc 1 15
sidebc 2 2

sidebc 100 3 1

linebc 11 12

exit
```

GEN3D file:

```
translate 10 1.
exit
```

PRONTO3D input file:

```
ttitle
  chatter test problem
termination time .0025
plot time .00001
output time .00001
write restart .001

material 1 elastic .00074
  youngs modulus 30e6
  poissons ratio .3
end

material 2 elastic .00074
  youngs modulus 30e6
  poissons ratio .3
end

contact surface 1 2

pressure 100 50 1.

function 50
  0. 0.
  .00015 20000.
  .01 20000.
end

no displacement y 11
no displacement x 11

exit
```

A2.3.3 Buckling of a Shell-like Structures

makefile:

```
can.g2d: can.fsq can.size
fastq -aprepro -m can.g2d can.fsq

can.g: can.g2d can.gen3d can.size
gen3d -aprepro can.g2d can.g <can.gen3d

block.g2d: block.fsq can.size
fastq -aprepro -m block.g2d block.fsq

block.g: block.g2d block.gen3d can.size
gen3d -aprepro block.g2d block.g < block.gen3d

can_block.g: can.g block.g can_block.gjn
gjoin < can_block.gjn
```

FASTQ input for can:

```
title
self contact test
(include(can.size))
point 1 0 0
point 2 (rad) 0.
point 3 (rad+t) 0.

point 4 (-rad) 0.
point 5 (-rad-t) 0.

line 1 str 2 3 0 {ithick}
line 2 circ 2 4 1 {irad}
line 3 str 4 5 0 {ithick}
line 4 circ 3 5 1 {irad}
region 1 1 -1 -2 -3 -4

line 5 circ 4 2 1 {irad}
line 6 circ 5 3 1 {irad}

$ half can
region 2 3 -1 -5 -3 -6

sidebc 1 2 5
sidebc 2 4 6
```

FASTQ input for block:

```
title
self contact test problem
(include(can.size))
point 1 {rad+e} {-(rad+e)}
point 2 {-(rad+e)} {-(rad+e)}
point 3 {-(rad+e)} {rad+e}
point 4 {rad+e} {rad+e}

line 1 str 1 2 0 {isq}
line 2 str 2 3 0 {isq}
line 3 str 3 4 0 {isq}
line 4 str 4 1 0 {isq}

region 1 2 -1 -2 -3 -4
sidebc 1000 1 2 3 4
exit
```

can.size:

```
$ $ cylinder rad = {rad=5} t= {t=.2} int = {ithick=3} {irad=40}
$ length of cylinder {cylten=15}
$ number of elements in cylinder {icyl=40}
$ block {e = 2} int= {isq = 28}
$ number of elements in block thick {iblkt =5 } block thickness {blkt=2.5}
$ block angle {angle=10}
```

GEN3D file for can:

```
{include(can.size)}
translate {icyl} {cylten}
sideset front 4
nodeset back 100
exit
```

GEN3D file for block:

```
{include(can.size)}
translate {iblkt} {blkt}
revolve y {180+angle}
revcen 0. 0. 0.

offset 0. 0. {tand(angle)*(rad+t)+.01}
sideset front 3
exit
```

GJOIN file:

```
can.g
block.g
no
ssets
combine 1 2 4
combine 3 1000
exit
exit
no
can_block.g
```

PRONTO3D input file:

```
title
  self contact test problem
hourglass stiffness .01 .03
TERMINATION TIME = .004
READ RESTART = .002
WRITE RESTART = .00005
PLOT TIME = .0001
OUTPUT TIME = .00001
MATERIAL, 1, ELASTIC plastic, .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3333
hardening modulus 0.
yields stress 30000
beta = 1
END
MATERIAL, 2, ELASTIC plastic, .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3333
hardening modulus 0.
yields stress 30000
beta = 1
END
MATERIAL, 3, ELASTIC plastic, .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3333
hardening modulus 0.
yields stress 30000
beta = 1
END
contact surface 1
contact surface 3

INITIAL VELOCITY MATERIAL 2 0., 0. -5000.

no displacem x 100
no displacem y 100
no displacem z 100

PLOT ELEMENT =
PLOT NODAL DISPLACEMENT, VELOCITY, ACCELERATION
plot state = eqps

EXIT
```

A2.3.4 Cutting of a Steel Pipe

makefile:

```
bladel.g2d: bladel.fsq blade.size
fastq -aprepro -m bladel.g2d bladel.fsq

bladel.g: bladel.g2d blade.size bladel.gen3d
gen3d -aprepro bladel.g2d bladel.g <bladel.gen3d

edge.g2d: edge.fsq blade.size
fastq -aprepro -m edge.g2d edge.fsq

edge1.g3: edge.g2d blade.size edge1.gen3d
gen3d -aprepro edge.g2d edge1.g3 < edge1.gen3d

edge1.g31: edge1.g3 edge1.grp blade.size
grepos -aprepro edge1.g3 edge1.g31 < edge1.grp

edge1.g: edge1.g31 edge11.grp blade.size
grepos -aprepro edge1.g31 edge1.g < edge11.grp

edge2.g3: edge.g2d blade.size edge2.gen3d
gen3d -aprepro edge.g2d edge2.g3 < edge2.gen3d

edge2.g31: edge2.g3 edge2.grp blade.size
grepos -aprepro edge2.g3 edge2.g31 < edge2.grp

edge2.g: edge2.g31 edge21.grp blade.size
grepos -aprepro edge2.g31 edge2.g < edge21.grp

edge3.g3: edge.g2d blade.size edge3.gen3d
gen3d -aprepro edge.g2d edge3.g3 < edge3.gen3d

edge3.g31: edge3.g3 edge3.grp blade.size
grepos -aprepro edge3.g3 edge3.g31 < edge3.grp

edge3.g: edge3.g31 edge31.grp blade.size
grepos -aprepro edge3.g31 edge3.g < edge31.grp

edge4.g3: edge.g2d blade.size edge4.gen3d
gen3d -aprepro edge.g2d edge4.g3 < edge4.gen3d

edge4.g31: edge4.g3 edge4.grp blade.size
grepos -aprepro edge4.g3 edge4.g31 < edge4.grp

edge4.g: edge4.g31 edge41.grp blade.size
grepos -aprepro edge4.g31 edge4.g < edge41.grp

edge5.g3: edge.g2d blade.size edge5.gen3d
gen3d -aprepro edge.g2d edge5.g3 < edge5.gen3d

edge5.g31: edge5.g3 edge5.grp blade.size
grepos -aprepro edge5.g3 edge5.g31 < edge5.grp

edge5.g: edge5.g31 edge51.grp blade.size
grepos -aprepro edge5.g31 edge5.g < edge51.grp
```

makefile (contd):

```
blade.g3: blade1.g edge1.g edge2.g edge3.g edge4.g edge5.g blade.gjn
gjoin < blade.gjn
```

```
blade.g: blade.g3 blade.grp
grepos -aprepro blade.g3 blade.g < blade.grp
```

```
pipe1.g2d: pipe1.fsq blade.size pipe.size
fastq -aprepro -m pipe1.g2d pipe1.fsq
```

```
pipe1.g3: pipe1.g2d blade.size pipe.size pipe1.gen3d
gen3d -aprepro pipe1.g2d pipe1.g3 <pipe1.gen3d
```

```
pipe1.g: pipe1.g3 pipe1.grp
grepos -aprepro pipe1.g3 pipe1.g < pipe1.grp
```

```
pipe2.g2d: pipe2.fsq blade.size pipe.size
fastq -aprepro -m pipe2.g2d pipe2.fsq
```

```
pipe2.g3: pipe2.g2d blade.size pipe.size pipe2.gen3d
gen3d -aprepro pipe2.g2d pipe2.g3 <pipe2.gen3d
```

```
pipe2.g: pipe2.g3 pipe2.grp
grepos -aprepro pipe2.g3 pipe2.g < pipe2.grp
```

```
clamp.g2d: clamp.fsq blade.size pipe.size clamp.size
fastq -aprepro -m clamp.g2d clamp.fsq
```

```
clamp.g3: clamp.g2d blade.size pipe.size clamp.size clamp.gen3d
gen3d -aprepro clamp.g2d clamp.g3 <clamp.gen3d
```

```
clamp.g: clamp.g3 blade.size clamp.size clamp.grp
grepos -aprepro clamp.g3 clamp.g < clamp.grp
```

```
pcut.g: blade.g pipe1.g pipe2.g clamp.g
gjoin < pcut.gjn
```

FASTQ input for blade1:

```
title
  pipe cutter blade

(include(blade.size))

point 1 -2.594 0.312
point 2 2.716 0.312
point 3 {x3} {y3}
point 4 {x4} {y4}
point 5 {x5} {y5}
point 6 {x6} {y6}
point 7 {x7} {y7}
point 8 {x8} {y8}
point 9 {x9} {y9}
point 10 {0.5*(x8+x9)} {y8+0.7*abs(x8-x9)}
point 11 {0.5*(x8+x9)} {y8-0.068}

line 1 str 1 2 0 {3*i1}
line 2 str 2 3 0 {4*i1} 0.9
line 3 str 3 4 0 {2*i1}
line 4 str 4 8 0 {10*i1} 0.8
line 5 circ 11 8 10 {3*i1} 1.17647
line 6 circ 9 11 10 {3*i1} 0.85
line 7 str 9 6 0 {10*i1} 1.25
line 8 str 6 7 0 {3*i1}
line 9 str 1 7 0 {4*i1} 0.9

region 1 1 -1 -2 -3 -4 -5 -6 -7 -8 -9

scheme 1 X6s

nodebc 200 1

exit
```

GEN3D file for blade1:

```
(include(blade.size))

translate {i2} {bthick/2.+0.002}
nsets back 100
ssets front 20
exit
```

blade.size:

```
$ {i1=1} {i2=2} {bthick=0.25}
$ {x3=2.716} {y3=-2.1715}
$ {x4=1.5} {y4=-2.286}
$ {x5=0.156} {y5=-3.63}
$ {x6=-1.0409} {y6=-2.4331}
$ {x7=-2.594} {y7=-2.175}
$ {x8=0.288583} {y8=-3.497417}
$ {x9=0.023417} {y9=-3.497417}
```

FASTQ input for edge:

```
title
  pipe cutter blade edge

(include(blade.size))

point 1 0.002 0.0
point 2 -0.125 0.0
point 3 -0.02 -0.2
point 4 0.002 -0.2165
point 5 0.002 -0.205

line 1 str 1 2 0 {i2}
line 2 str 2 3 0 {2*i2} 0.8
line 3 circ 3 4 5 2
line 4 str 4 1 0 {2*i2} 1.4

region 1 1 -1 -2 -3 -4

scheme 1 X6

nodebc 100 4
sidebc 10 2 3

exit
```

GEN3D file for edge1:

```
(include(blade.size))
translate (2*i1), {x3-x4}
exit
```

GREPOS files for edge1:

```
(include(blade.size))
offset spline
slope left {atan(abs(y4-y3)/abs(x4-x3))}
slope right {atan(abs(y4-y3)/abs(x4-x3))}
0.0 0. 0.
{-abs(x4-x3)} 0.0 {-abs(y3-y4)}
end
exit
```

```
(include(blade.size))
revcen 0.0 0.0 0.0
revolve y 90.
offset x {x3} y {y3} z {-bthick/2.0}
exit
```

GEN3D file for edge2:

```
(include(blade.size))
translate (10*i1), {x4-x8} 0.8
exit
```

GREPOS files for edge2:

```
{include(blade.size)}  
offset spline  
slope left {atan(abs(y5-y4)/abs(x5-x4))}  
slope right {atan(abs(y5-y4)/abs(x5-x4))}  
0.0 0. 0.  
{-abs(x5-x4)} 0.0 {-abs(y4-y5)}  
end  
exit
```

```
{include(blade.size)}  
revcen 0.0 0.0 0.0  
revolve y 90.  
offset x {x4} y {y4} z {-bthick/2.0}  
exit
```

GEN3D file for edge3:

```
{include(blade.size)}  
translate {10*i1}, {x9-x6} 1.25  
exit
```

GREPOS files for edge3:

```
{include(blade.size)}  
offset spline  
slope left {atan(abs(y6-y5)/abs(x6-x5))}  
slope right {atan(abs(y6-y5)/abs(x6-x5))}  
0.0 0. 0.  
{-abs(x6-x5)} 0.0 {abs(y5-y6)}  
end  
exit
```

```
{include(blade.size)}  
revcen 0.0 0.0 0.0  
revolve y 90.  
offset x {x9} y {y9} z {-bthick/2.0}  
exit
```

GEN3D file for edge4:

```
{include(blade.size)}  
translate {3*i1}, {x6-x7}  
exit
```

GREPOS files for edge4:

```
{include(blade.size)}
offset spline
slope left {atan(abs(y7-y6)/abs(x7-x6))}
slope right {atan(abs(y7-y6)/abs(x7-x6))}
0.0 0. 0.
{-abs(x7-x6)} 0.0 {abs(y6-y7)}
end
exit
```

```
{include(blade.size)}
revcen 0.0 0.0 0.0
revolve y 90.
offset x {x6} y {y6} z {-bthick/2.0}
exit
```

GEN3D file for edge5:

```
{include(blade.size)}
translate {6*i1}, {abs(x8-x9)}
exit
```

GREPOS files for edge5:

```
{include(blade.size)}

offset spline
slope top 0.0 {atan(abs(y5-y4)/abs(x5-x4))}
slope bottom 0.0 {-atan(abs(y5-y4)/abs(x5-x4))}
0.0 0.0 0.0
$ {abs(x8-x9)/4.0} 0.0 -0.048083261
{-abs(x8-x9)/2.0} 0.0 -0.068
$ {3.0*abs(x8-x9)/4.0} 0.0 -0.048083261
{-abs(x8-x9)} 0.0 0.0
end
exit
```

```
{include(blade.size)}
revcen 0.0 0.0 0.0
revolve y 90.
offset x {x8} y {y8} z {-bthick/2.0}
exit
```

GJOIN file for blade:

```
blade1.g
edge1.g
combine .000001
combine exit
exit
yes
edge2.g
combine .000001
combine exit
exit
yes
edge3.g
combine .000001
combine exit
exit
yes
edge4.g
combine .000001
combine exit
exit
yes
edge5.g
combine .01
combine exit
exit
no
blade.g3
```

GREPOS file for blade:

```
offset y -0.064
exit
```

FASTQ input for pipe1:

```
title
pipe

(include(pipe.size))
(include(blade.size))

point 1 (x5) (y5 - (bthick/2.)/tan(3.14159*30./180.) - pr - pt)
point 2 (x5) (y5 - (bthick/2.)/tan(3.14159*30./180.) - pt )
point 3 (x5) (y5 - (bthick/2.)/tan(3.14159*30./180.) )
point 4 (x5) (y5 - (bthick/2.)/tan(3.14159*30./180.) - 2*pr - pt )
point 5 (x5) (y5 - (bthick/2.)/tan(3.14159*30./180.) - 2*pr -2*pt)

line 1 str 2 3 0 {ithick}
line 2 circ 2 4 1 {irad}
line 3 str 4 5 0 {ithick}
line 4 circ 3 5 1 {irad}
line 5 circ 4 2 1 {irad}
line 6 circ 5 3 1 {irad}

region 1 2 -1 -2 -3 -4
region 2 3 -1 -5 -3 -6

exit
```

pipe.size:

```
$ {ithick=4} {irad=50} {pr=1.0335} {pt=0.154}
$ {ithick2=2} {irad2=25}
$ {plength=1.5} {i3=16}
$ {plength2=3.5} {i4=12}
```

GEN3D file for pipe1:

```
(include(blade.size))
(include(pipe.size))

translate {i3} {plength/2.} 1.05
nsets front 100
ssets back 400
exit
```

GREPOS file for pipe1:FASTQ input for pipe2:

```
{include(blade.size)}
revcen {x5} 0.0 0.0
revolve y 180
offset x 0.00001 z {-bthick/2.+0.0001}
exit
```

```
title
pipe

{include(pipe.size)}
{include(blade.size)}

point 1 {x5} {y5 - (bthick/2.)/tan(3.14159*30./180.) - pr - pt}
point 2 {x5} {y5 - (bthick/2.)/tan(3.14159*30./180.) - pt }
point 3 {x5} {y5 - (bthick/2.)/tan(3.14159*30./180.) }
point 4 {x5} {y5 - (bthick/2.)/tan(3.14159*30./180.) - 2*pr - pt }
point 5 {x5} {y5 - (bthick/2.)/tan(3.14159*30./180.) - 2*pr -2*pt}

line 1 str 2 3 0 {ithick2}
line 2 circ 2 4 1 {irad2}
line 3 str 4 5 0 {ithick2}
line 4 circ 3 5 1 {irad2}
line 5 circ 4 2 1 {irad2}
line 6 circ 5 3 1 {irad2}

region 1 4 -1 -2 -3 -4
region 2 5 -1 -5 -3 -6

exit
```

GEN3D file for pipe2:

```
{include(blade.size)}
{include(pipe.size)}

translate {i4} {plength2} 1.08
ssets front 401
exit
```

GREPOS file for pipe2:

```
{include(blade.size)}
{include(pipe.size)}

revcen {x5} 0.0 0.0
revolve y 180
offset x 0.00001 z {-bthick/2.+0.0001+plength/2.}
exit
```

FASTQ input for clamp:

```
title
clamp

(include(pipe.size))
(include(blade.size))
(include(clamp.size))

point 1 -4.0 (y5 - (bthick/2.)/tan(3.14159*30./180.) - 2*pr -2*pt)
point 2 4.0 (y5 - (bthick/2.)/tan(3.14159*30./180.) - 2*pr -2*pt)
point 3 4.0 (y5 - (bthick/2.)/tan(3.14159*30./180.) - 2*pr -2*pt - cheight)
point 4 -4.0 (y5 - (bthick/2.)/tan(3.14159*30./180.) - 2*pr -2*pt - cheight)

line 1 str 1 2 0 8
line 2 str 2 3 0 2
line 3 str 3 4 0 8
line 4 str 4 1 0 2

region 1 1 -1 -2 -3 -4

nodebc 300 3
sidebc 30 1
exit
```

clamp.size:

```
{cthickness=0.5} {coffset=0.31}
{cheight=1.0}
```

GEN3D file for clamp:

```
(include(blade.size))
(include(pipe.size))
(include(clamp.size))

translate 2 {cthickness}
sset back 31
exit
```

GREPOS file for clamp:

```
(include(blade.size))
(include(clamp.size))

offset z {cthickness+coffset-bthickness/2.}
exit
```

GJOIN file for blade, pipe, and clamp:

```
blade.g
pipe1.g
no
exit
yes
pipe2.g
no
exit
yes
clamp.g
no
exit
no
pcut.g
```

PRONTO3D input file:

```
TITLE
  Pipe Cutting Simulation

$hourglass stiffness 0.01 0.03
TERMINATION TIME = 4.E-3
PLOT TIME = 4.e-5
OUTPUT TIME = 4.e-5
write restart = 8.e-5

function 1
0.0 1.0
1.0 1.0
end

MATERIAL, 1, ELASTIC, .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3333
END

MATERIAL, 2, plh strength , .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3
yield stress, 38.8e3
hardening constant 93.6e3
hardening exponent 0.4539
luders strain 0.021
failure value 1.27
decay constant 0.7
END

MATERIAL, 3, plh strength , .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3
yield stress, 38.8e3
hardening constant 93.6e3
hardening exponent 0.4539
luders strain 0.021
failure value 1.27
decay constant 0.7
END
```

PRONTO3D input file (contd):

```
MATERIAL, 4, plh strength , .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3
yield stress, 38.8e3
hardening constant 93.6e3
hardening exponent 0.4539
luders strain 0.021
failure value 1.27
decay constant 0.7
END

MATERIAL, 5, plh strength , .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3
yield stress, 38.8e3
hardening constant 93.6e3
hardening exponent 0.4539
luders strain 0.021
failure value 1.27
decay constant 0.7
END

death 2 decay min 0.1
death 3 decay min 0.1

prescribed VELOCITY y 200 1 -1187.5
INITIAL VELOCITY MATERIAL 1 0. -1187.5 0.

no displacement z 100
no displacement y 300

PLOT ELEMENT = vonmises, pressure
plot state = eqps
PLOT NODAL DISPLACEMENT, REACTION, VELOCITY, ACCELERATION

contact surface 400 401 fixed
CONTACT material 1
CONTACT material 2
CONTACT material 3

EXIT
```

A2.3.5 Elastic-Plastic Bar Impacting Bricks

makefile:

```
bricks.g2d: bricks.fsq
    fastq -aprepro -m bricks.g2d -- bricks.fsq

bricks.g: bricks.g2d bricks.gen3d
    gen3d bricks.g2d bricks.g <bricks.gen3d

bricks.gx: bricks.g
    exoxdr bricks.g bricks.gx

wall.g2d: wall.fsq
    fastq -m wall.g2d -- wall.fsq

wall.g: wall.g2d wall.gen3d
    gen3d wall.g2d wall.g <wall.gen3d

brick_wall.gx: brick_wall.g
    exoxdr brick_wall.g brick_wall.gx

rod.g2d: rod.fsq
    fastq -m rod.g2d -- rod.fsq

rod.g: rod.g2d rod.gen3d
    gen3d rod.g2d rod.g <rod.gen3d

brick_wall.g: bricks.g wall.g rod.g
    gjoin < brick_wall.gjn
```

FASTQ file for bricks:

```

TITLE
STEEL ROD HITTING BRICKS
$ (e = .005)
POINT 1 {0.+e} {0.+e}
POINT 2 {1.-e} {0.+e}
POINT 3 {1.-e} {.5-e}
POINT 4 {0.+e} {.5-e}

LINE 1 STR 1 2 0 10
LINE 2 STR 2 3 0 5
LINE 3 STR 3 4 0 10
LINE 4 STR 4 1 0 5

REGION 1 1 -1 -2 -3 -4

POINT 11 {1.+e} {0.+e}
POINT 12 {2.-e} {0.+e}
POINT 13 {2.-e} {.5-e}
POINT 14 {1.+e} {.5-e}

LINE 11 STR 11 12 0 10
LINE 12 STR 12 13 0 5
LINE 13 STR 13 14 0 10
LINE 14 STR 14 11 0 5

REGION 11 1 -11 -12 -13 -14

POINT 21 {2.+e} {0.+e}
POINT 22 {3.-e} {0.+e}
POINT 23 {3.-e} {.5-e}
POINT 24 {2.+e} {.5-e}

LINE 21 STR 21 22 0 10
LINE 22 STR 22 23 0 5
LINE 23 STR 23 24 0 10
LINE 24 STR 24 21 0 5

REGION 21 1 -21 -22 -23 -24

$ SECOND ROW

POINT 101 {0.+e} {.5+e}
POINT 102 {.5-e} {0.5+e}
POINT 103 {.5-e} {1.-e}
POINT 104 {0.+e} {1.-e}

LINE 101 STR 101 102 0 5
LINE 102 STR 102 103 0 5
LINE 103 STR 103 104 0 5
LINE 104 STR 104 101 0 5

REGION 101 1 -101 -102 -103 -104

POINT 111 {.5+e} {.5+e}
POINT 112 {1.5-e} {.5+e}
POINT 113 {1.5-e} {1.-e}
POINT 114 {.5+e} {1.-e}

LINE 111 STR 111 112 0 10
LINE 112 STR 112 113 0 5
LINE 113 STR 113 114 0 10
LINE 114 STR 114 111 0 5

REGION 111 1 -111 -112 -113 -114

POINT 121 {1.5+e} {.5+e}
POINT 122 {2.5-e} {0.5+e}
POINT 123 {2.5-e} {1.-e}
POINT 124 {1.5+e} {1.-e}

LINE 121 STR 121 122 0 10
LINE 122 STR 122 123 0 5
LINE 123 STR 123 124 0 10
LINE 124 STR 124 121 0 5

REGION 121 1 -121 -122 -123 -124

POINT 131 {2.5+e} {.5+e}
POINT 132 {3.0-e} {0.5+e}
POINT 133 {3.0-e} {1.-e}
POINT 134 {2.5+e} {1.-e}

LINE 131 STR 131 132 0 5
LINE 132 STR 132 133 0 5
LINE 133 STR 133 134 0 5
LINE 134 STR 134 131 0 5

REGION 131 1 -131 -132 -133 -134

$ THIRD ROW

POINT 201 {0.+e} {1.+e}
POINT 202 {1.-e} {1.+e}
POINT 203 {1.-e} {1.5-e}
POINT 204 {0.+e} {1.5-e}

LINE 201 STR 201 202 0 10
LINE 202 STR 202 203 0 5
LINE 203 STR 203 204 0 10
LINE 204 STR 204 201 0 5

REGION 201 1 -201 -202 -203 -204

POINT 211 {1.+e} {1.+e}
POINT 212 {2.-e} {1.+e}
POINT 213 {2.-e} {1.5-e}
POINT 214 {1.+e} {1.5-e}

```

FASTQ file for bricks (contd):

```

LINE 211 STR 211 212 0 10
LINE 212 STR 212 213 0 5
LINE 213 STR 213 214 0 10
LINE 214 STR 214 211 0 5

REGION 211 1 -211 -212 -213 -214

POINT 221 {2.+e} {1. +e}
POINT 222 {3.-e} {1. +e}
POINT 223 {3.-e} {1.5-e}
POINT 224 {2.+e} {1.5-e}

LINE 221 STR 221 222 0 10
LINE 222 STR 222 223 0 5
LINE 223 STR 223 224 0 10
LINE 224 STR 224 221 0 5

REGION 221 1 -221 -222 -223 -224

$ FOURTH ROW

POINT 301 {0.+e} {1.5+e}
POINT 302 {.5-e} {1.5+e}
POINT 303 {.5-e} {2.0-e}
POINT 304 {0.+e} {2.0-e}

LINE 301 STR 301 302 0 5
LINE 302 STR 302 303 0 5
LINE 303 STR 303 304 0 5
LINE 304 STR 304 301 0 5

REGION 301 1 -301 -302 -303 -304

POINT 311 {.5+e} {1.5+e}
POINT 312 {1.5-e} {1.5+e}
POINT 313 {1.5-e} {2.-e}
POINT 314 {.5+e} {2.-e}

LINE 311 STR 311 312 0 10
LINE 312 STR 312 313 0 5
LINE 313 STR 313 314 0 10
LINE 314 STR 314 311 0 5

REGION 311 1 -311 -312 -313 -314

POINT 321 {1.5+e} {1.5+e}
POINT 322 {2.5-e} {1.5+e}
POINT 323 {2.5-e} {2.-e}
POINT 324 {1.5+e} {2.-e}

LINE 321 STR 321 322 0 10
LINE 322 STR 322 323 0 5
LINE 323 STR 323 324 0 10
LINE 324 STR 324 321 0 5

REGION 211 1 -321 -322 -323 -324

POINT 331 {2.5+e} {1.5+e}
POINT 332 {3.0-e} {1.5+e}
POINT 333 {3.0-e} {2.-e}
POINT 334 {2.5+e} {2.-e}

LINE 331 STR 331 332 0 5
LINE 332 STR 332 333 0 5
LINE 333 STR 333 334 0 5
LINE 334 STR 334 331 0 5

REGION 331 1 -331 -332 -333 -334

$ FIFTH ROW

POINT 401 {0.+e} {2.+e}
POINT 402 {1.-e} {2. +e}
POINT 403 {1.-e} {2.5-e}
POINT 404 {0.+e} {2.5-e}

LINE 401 STR 401 402 0 10
LINE 402 STR 402 403 0 5
LINE 403 STR 403 404 0 10
LINE 404 STR 404 401 0 5

REGION 401 1 -401 -402 -403 -404

POINT 411 {1.+e} {2.+e}
POINT 412 {2.-e} {2. +e}
POINT 413 {2.-e} {2.5-e}
POINT 414 {1.+e} {2.5-e}

LINE 411 STR 411 412 0 10
LINE 412 STR 412 413 0 5
LINE 413 STR 413 414 0 10
LINE 414 STR 414 411 0 5

REGION 411 1 -411 -412 -413 -414

POINT 421 {2.+e} {2. +e}
POINT 422 {3.-e} {2. +e}
POINT 423 {3.-e} {2.5-e}
POINT 424 {2.+e} {2.5-e}

LINE 421 STR 421 422 0 10
LINE 422 STR 422 423 0 5
LINE 423 STR 423 424 0 10
LINE 424 STR 424 421 0 5

REGION 421 1 -421 -422 -423 -424

EXIT

```

GEN3D file for bricks:

```
translate 3 .6
exit
```

FASTQ file for wall:

```
title
wall
point 1 5.5 0.
point 2 5.85355 .35355
point 4 .5 5.
point 3 .85355 5.35355

line 1 str 1 2 0 5
line 2 str 2 3 0 70
line 3 str 3 4 0 5
line 4 str 4 1 0 70

region 1 3 -1 -2 -3 -4

exit
```

GEN3D file for wall:

```
translate 9 1.5
offset 0.05 0.05 .5
exit
```

FASTQ file for rod:

```
title
brick wall example
$ STEEL ROD

POINT 500 1.25 0.
POINT 501 1.75 0.
POINT 502 1.75 -.5
POINT 503 1.25 -.5

LINE 501 STR 500 501 0 5
LINE 502 STR 501 502 0 5
LINE 503 STR 502 503 0 5
LINE 504 STR 503 500 0 5

REGION 500 2 -501 -502 -503 -504

EXIT
```

GEN3D file for rod:

```
translate 30 5
offset 0. 0. 3.
exit
```

GREPOS file for bricks, wall, and rod:

```
bricks.g
wall.g
no
exit
yes
rod.g
no
exit
no
brick_wall.g
```

PRONTO3D input file:

```
title
  MULTI-BOBY CONTACT test problem
TERMINATION TIME = .0005
PLOT TIME = .00001
OUTPUT TIME = .00001

MATERIAL, 1, ELASTIC, .00074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3333
END
MATERIAL, 2, ELASTIC, .0074
YOUNGS MODULUS, 30E6
POISONS RATIO, .3333
END

MATERIAL, 3, ELASTIC plastic, .0005
YOUNGS MODULUS, 16E6
POISONS RATIO, .3333
hardening modulus 1000.
beta = 1
yield stress = 10000.
END

INITIAL VELOCITY MATERIAL 2 0. , 2500. , 0.

PLOT ELEMENT =
PLOT NODAL DISPLACEMENT, VELOCITY, ACCELERATION

CONTACT MATERIAL 1
CONTACT MATERIAL 2
CONTACT MATERIAL 3

EXIT
```

A2.3.6 Forging of a Copper Billet

FASTQ file:

```
TITLE
RIVET PROBLEM FOR CONTACT SURFACE CHECK
POINT 1 0.0 0.0
POINT 2 2.0 0.0
POINT 3 2.2 0.0
POINT 4 0.0 0.2
POINT 5 2.0 0.2
POINT 6 2.2 0.2
POINT 7 2.0 2.95
POINT 8 2.2 3.0
POINT 9 4.0 3.0
POINT 10 2.0 3.2
POINT 11 2.25 3.2
POINT 12 4.0 3.2
POINT 13 0.0 0.2
POINT 14 1.75 0.2
POINT 15 1.75 6.2
POINT 16 0.0 6.2
POINT 17 0.0 6.2
POINT 18 4.0 6.2
POINT 19 4.0 7.2
POINT 20 0.0 7.2
POINT 21 2.25 3.0
POINT 22 2.20 2.95
POINT 23 2.25 2.95
LINE 1 STR 1 3 0 1
LINE 2 STR 3 22 0 1
LINE 3 STR 21 9 0 1
LINE 4 STR 4 5 0 1
LINE 5 STR 5 7 0 1
LINE 6 STR 11 12 0 1
LINE 7 STR 1 4 0 1
LINE 8 STR 5 3 0 1
LINE 9 STR 8 10 0 1
LINE 10 STR 9 12 0 1
LINE 13 STR 13 14 0 20
LINE 14 STR 14 15 0 80
LINE 15 STR 16 15 0 20
LINE 16 STR 13 16 0 80
LINE 17 STR 17 18 0 1
LINE 18 STR 18 19 0 1
LINE 19 STR 20 19 0 1
LINE 20 STR 17 20 0 1
LINE 21 CIRC 21 22 23 45
LINE 22 CIRC 11 7 23 45
LINE 23 STR 7 22 0 1
LINE 24 STR 21 11 0 1
REGION 1 1 -1 -8 -4 -7
REGION 2 1 -2 -23 -5 -8
REGION 3 1 -3 -10 -6 -24
REGION 4 2 -13 -14 -15 -16
REGION 5 3 -17 -18 -19 -20
REGION 6 1 -21 -24 -22 -23
NODEBC 3 1 2 3 4 5 6 21 22
NODEBC 1 16 17 18 19 20
NODEBC 2 17 18 19 20
SIDEBC 100 4 5 6 22
SIDEBC 200 13 14 15
SIDEBC 300 17
SIDEBC 400 14 15
EXIT
```

SANTOS2D input file:

```
TITLE
RIVET PROBLEM - CHECK FOR CONTACT SURFACES
AXISYMMETRIC

STEP CONTROL
500 1.
END
OUTPUT TIME
10 1.
END
PLOT TIME
5 1.
END

TIME STEP SCALE = 0.75
MINIMUM DAMPING FACTOR = .8
RESIDUAL TOLERANCE, 0.5
MAXIMUM TOLERANCE, 100.
MAXIMUM ITERATIONS, 2000

INTERMEDIATE PRINT, 10
PLOT NODAL = DISPLACEMENT
PLOT STATE = EQPS
PLOT ELEMENT = VONMISES, PRESSURE, STRESS, STRAIN

FUNCTION, 1
0.0,0.0
1.0,1.0
END

MATERIAL, 1, ELASTIC, 1.
YOUNGS MODULUS = 30.E6
POISSONS RATIO = 0.3
END

MATERIAL, 2, EP POWER HARD, 1.
YOUNGS MODULUS = 17.0E+6
POISSONS RATIO = 0.35
HARDENING CONSTANT = 50.0E+3
HARDENING EXPONENT = .31
YIELD STRESS = 15.5E+3
LUDER STRAIN = 0.0
END

MATERIAL, 3, ELASTIC, 1.
YOUNGS MODULUS = 30.E+6,
POISSONS RATIO = 0.3
END

CONTACT SURFACE, 100, 200, 0., 1.E-10
CONTACT SURFACE, 300, 400, 0., 1.E-10

NO DISPLACEMENT, X, 3
NO DISPLACEMENT, Y, 3
NO DISPLACEMENT, X, 1
PRESCRIBED DISPLACEMENT, Y, 2, 1, -2.5

EXIT
```

DISTRIBUTION:

1 Dr. R. T. Allen
Pacifica Technology
P.O. Box 148
Del Mar, CA 92014

3 Anatech International Corp.
5435 Oberlin Drive
San Diego, CA 92121
Attn: Dr. R. A. Dameron
Dr. R. S. Dunham
Dr. Joe Rashid

Prof. S. Atluri
Center for the Advancement of
Computational Mechanics
School of Civil Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Dr. Ali S. Argon
Dept. of Mechanical Engineering
Massachusetts Institute of
Technology
Cambridge, MA 02139

Dr. William E. Bachrach
Areojet Research Propulsion Inst.
P. O. Box 13502
Sacramento, CA 95853-4502

Mr. Ken Bannister
USA Ballistic Research Lab
SLCBBR-IB-M
Aberdeen Proving Grounds, MD
21005-5066

3 Battelle
505 King Avenue
Columbus, OH 43201-2693
Attn: Mr. R. Douglas Everhart
Dr. Michael L. Fisher
Mr. Charles R. Hargreaves

Prof. T. Belytschko
Department of Civil Engineering
Northwestern University
Evanston, IL 60201

Mr. Willy Benz
University of Arizona
Steward Observatory
Tucson, AZ 85321

Mr. Naury K. Birnbaum
Century Dynamics Incorporated
7700 Edgewater Dr., Suite 626
Oakland CA 94621

Mr. Akif O. Bolukbasi
McDonnell Douglas Helicopter
Company
5000 East McDowell Road
Mesa, AZ 85205-9797

Dr. Kenneth W. Brown
Computer Aided Engineering
Associates, Inc.
398 Old Sherman Hill Rd.
Woodbury, CT 06798

Mr. Malcolm Burgess
AEA Transport Technology
205/B71 Winfrith Technology Centre
Dorchester, Dorset DT2 8DH
United Kingdom

Mr. Mark Campbell
PASTDCO
1248 Princeton N.E.
Albuquerque, NM 87106

Dr. Tom Canfield
Argonne National Laboratories
9700 S. Cass Ave CTD/221
Argonne, IL 60439-4844

Mr. Ted Carney
Advanced Sciences Inc.
6739 Academy Road N.E.
Albuquerque, NM 87109

Mr. Tien S. Chou
EG&G Mound
P.O. Box 3000
Miamisburg, OH 45343

Mr. Chuck Charman
GA Technologies
P.O. Box 81608
San Diego, CA 92138

Mr. Ken K. Chipley
Martin Marietta Energy Systems
P.O. Box 2009
Oak Ridge, TN 37831-8053

Mr. Ken P. Chong
Dept. of Civil Engineering
University of Wyoming
Laramie, WY 82071

Dr. S. C. (Tony) Chou
U.S. Army Materials Technology Lab
SLCMT-BM
Watertown, MA 02172-0001

Mr. Dwight Clark
Mail Stop 281
Morton Thiokol Corp.
P. O. Box 524
Brigham City, UT 84302

Mr. Gregory Clifford
Cray Research Park
655E Lone Oak Drive
Eagan, MN 55121

Mr. Gerald Collingwood
Morton Thiokol, Inc.
Huntsville, AL 35807-7501

Mr. David L. Conover
Swanson Analysis Systems, Inc.
P. O. Box 65
Houston, PA 15342-0065

Prof. Steven M. Cramer
University of Wisconsin
2266 Engineering Building
1415 Johnson Drive
Madison, WI 53706

Mr. Steven Crouch
GeoLogic Research, Inc.
1313 Fifth St. SE, Suite 226
Minneapolis, MN 55414

Dr. Ian Cullis
XTZ Division
Royal Armament R&D
Establishment
Fort Halstead
Sevenoaks, Kent
United Kingdom

Mr. Peter Cundall
ITASCA Consulting Group, Inc.
1313 Fifth Street, S.E.
Minneapolis, MN 55414

Mr. Richard E. Danell
Research Officer
Central Research Laboratories
BHP Research & New Technology
P.O. Box 188
Wallsend NSW 2287
Australia

Mr. William A. Danne
Strategic Systems Division, MS#50
Teledyne Brown Engineering
P.O. Box 07007
Huntsville, AL 35807-7007

Dr. C. S. Desai
Dept. of Civil Eng. & Eng. Mech.
The University of Arizona
Tucson, AZ 85721

Mr. Ramji Digumarthi
Org. 8111, Bldg. 157
Lockheed MSD
P.O. Box 3504
Sunnyvale, CA 94088-3504

Prof. Robert Dodds, Jr.
Department of Civil Engineering
3140 Newmark Laboratory, MC-250
University of Illinois at Urbana
Urbana, IL 61801-2397

Dr. Arlo Fossum
RE/SPEC Inc.
Box 725
Rapid City, SD 57709

Dr. Russel Garnsworthy
CRA Advanced Tech Development
G.P.O. Box 384D
Melbourne 3001, Australia

Prof. Lorna Gibson
Department of Civil Engineering
Room 1-232
Massachusetts Institute of
Technology
Cambridge, MA 02139

- 2 Goodyear Technical Center
P.O. Box 3531
Akron, OH 44309-3531
Attn: Mr. Loren K. Miller
Mr. David Wismer, D/410F

Dr. Gerry Goudreau
Methods Development Group
Mechanical Engineering Department
Lawrence Livermore National Lab
Livermore, CA 94550

Prof. O. Hayden Griffin, Jr.
Dept. of Eng. Science & Mechanics
Virginia Polytechnic Institute and
State University
Blacksburg, VA 24061-0219

- 2 Grumman Corporate Technology
Bethpage, NY 11714-3580
Attn: Dr. John M. Papazian
Dr. Allan B. Pifko

Mr. H. L. Hassenpflug
B&W Fuel Company
P.O. Box 10935
3315 Old Forest Rd
Lynchburg, VA 24501

- 5 Hibbitt, Karlsson & Sorrensen, Inc.
100 Medway St.
Providence, RI 02906
Attn: Dr. David Hibbitt
Dr. Joop Nagtegaal
Dr. D. P. Flanagan
Dr. L. M. Taylor
Dr. W. C. Mills-Curran

Mr. Richard Hilson
M/S 4G09
GTE Government Systems
Corporation
P.O. Box 7188
Mountain View, CA 94039

Mr. Douglas Holzhauer
Rome Air Development Center
Griffiss AFB, NY 13441

Dr. William Hufferd
United Technologies
Chemical Systems Division
P.O. Box 50015
San Jose, CA 95150-0015

17
Prof. T. J. R. Hughes
Dept. of Mechanical Engineering
Stanford University
Palo Alto, CA 94306

Mr. James P. Johnson
Rm L120, CPC Analysis Dept.
General Motors Corp. Engineering
Center
30003 Van Dyke Avenue
Warren, MI 48090-9060

Mr. Jerome B. Johnson
USACRREL
Building 4070
Ft. Wainwright, AK 99703

Mr. Ken Johnson
Theoretical and Applied Mechanics
Group
Battelle Pacific Northwest
Laboratories
P.O. Box 999
Richland, WA 99352

Dr. Gordon R. Johnson
Honeywell, Inc.
5901 S. County Rd. 18
Edina, MN 55436

Mr. James W. Jones
Swanson Service Corporation
18700 Beach Blvd.
Suite 200-210
Huntington Beach, CA 92648

Mr. Sheldon Jones
Kaman Sciences
P.O. Box 7463
Colorado Springs, CO 80933-7463

Mr. G. A. Kaupp
Ford Motor Company
P.O. Box 2053, Room 2019
Dearborn, MI 48124

Dr. David W. Keck
CONVEX Computer Corporation
P.O. Box 833851 M.S. MAR
Richardson, TX 75083-3851

Mr. Gary Ketner
Applied Mechanics and Structures
Battelle Pacific Northwest
Laboratories
P.O. Box 999
Richland, WA 99352

Dr. Sam Key
RE/SPEC Inc.
4775 Indian School NE, Suite 300
Albuquerque, NM 87110-3827

Prof. Raymond D. Krieg
Engineering Science and Mechanics
301 Perkins Hall
University of Tennessee
Knoxville, TN 37996-2030

Mr. Don D. Kunard
Analytical Systems Engineering
Corp.
1725 Jefferson Davis Hwy, Suite 212
Arlington, VA 22202

Mr. Brett Lewis
APTEK
1257 Lake Plaza Drive
Colorado Springs, CO 80906-3578

Norman A. Lindsey
MCAE Technical Marketing
CONVEX Computer Corporation
P. O. Box 833851 M.S. MAR
Richardson, TX 75083-3851

Mr. Trent R. Logan
Rockwell International Corp.
P.O. Box 92098
Los Angeles, CA 90009

34 Los Alamos National Laboratory
Los Alamos, NM 87545

Attn:

J. Hopson, T3, MS B216
R. Hill, P15, MS D44
J. P. Hill, WX-11, MS C931
D. J. Sandstorm, MST-DO, MSG756
K. A. Meyer, X-3 MS F663
W. A. Cook, N-6, MS K557
P. T. Maulden, N-6, MS K557
J. J. Ruminer, WX-11, MS C931
S. P. Girrens, MEE-13, MS J576
J. L. Fales, MEE-13, MS J575
J. D. Allen, MEE-4, MS G787
D. A. Rabern, MEE-4, MS G787
M. W. Burkett, MEE-4, MS G787
J. H. Fu, MEE-4, MS G787
P. R. Romero, MEE-4, MS G787
P. S. Follansbee, MST-DO, MSG756
D. Mandell, X-3, MS F663
R. F. Davidson, N-6, MS K557
J. N. Johnson, N-6, MS K557
J. K. Dienes, N-6, MS K557
S. Marsh, N-6, MS K557
L. H. Sullivan, N-6, MS K557
D. L. Jaeger, WX-11, MS K557
C. A. Anderson, MEE-13, MS J576
J. G. Bennett, MEE-13, MS J576
T. A. Butler, MEE-13, MS J576
D. C. Nelson, MEE-4, MS G787
R. B. Parker, MEE-4, MS G787
M. W. Lewis, MEE-4, MS G787
E. S. Idar, MEE-4, MS G787
B. M. Wheat, MEE-4, MS G787
F. Guerra, WX-11, MS C931
C. Wingate, MS F645
B. Stellingwerf, MS F645

Dr. Jack Maison
Engineering Cybernetics, Inc.
1856 Lockhill Selma Rd, Suite 105
San Antonio, TX 78213

Mr. Joseph Marti
Hamilton Standard Division of
United Technologies
M/S 1-3-BC52
One Hamilton Road
Windsor Locks, CT 06096-1010

Mr. Darin McKinnis
NASA Pyrotechnics Group, MS EP5
LBJ Space Center
Houston, TX 77058

Mr. Craig Miller
Unit 973
Neutron Devices Department
General Electric Company
P.O. Box 2908
Largo, FL 34294-2908

2 Lockheed Missiles and Space Co.
P. O. Box 3504
Sunnyvale, CA 94088-3504
Attn:
Mr. J. J. Murphy, 59-22 B/580
Mr. Brian M. Cuthbert, 81-12 B/157

Prof. V. D. Murty
School of Engineering
University of Portland
5000 N. Willamette Blvd.
Portland, OR 97203

2 Naval Surface Warfare Center
10901 New Hampshire Ave.
Silver Spring, MD 20903-5000
Attn:
Mr. Hans Mair, Code R14
Mr. Andrew Wardlaw Jr. Code R44

2 Naval Research Lab
Materials Science & Technology
Building 28, Code 6386
4555 Overlook Avenue SW
Washington, DC 20375-5000
Attn: Mr. Luther D. Flippen
Dr. Carl Dyka

Dr. R. E. Nickell
c/o Anatech International Corp.
5435 Oberlin Drive
San Diego, CA 92121

Mr. Dean Norman
Waterways Experiment Station
P.O. Box 631
Vicksburg, MS 39180

- 2 Office of Naval Research
Structural Mechanics Div. (Code 434)
800 N. Quincy Street
Arlington, VA 22217
Attn: Dr. Rembert Jones
Dr. Alan S. Kushner

Mr. Shane R. Page
Albuquerque Valve & Fitting
Company
2451 Alamo S.E.
Albuquerque, NM 87106

Dr. Robert Pardue
Martin Marietta, MS 2
Y-12 Plant, Bldg. 9998
Oak Ridge, TN 37831

Dr. T. Kim Parnell
Failure Analysis Associates, Inc.
P.O. Box 3015
Menlo Park, CA 94025

Dr. Philip A. Pfund
Babcock & Wilcox
P.O. Box 271
Barberton, OH 44203

Mr. Mitchell R. Phillabaum
Monsanto Research Corp.
MRC-MOUND
Miamisburg, OH 45342

- 4 Phillips Laboratory (AFSC)
Kirtland AFB, NM 87117-6008
Attn:
Firooz Allahdadi, PL/WSSD
David Amdahl, PL/WSSD
David Medina, PL/WSSD
David H. Hilland, PL/WSSH

- 3 POD Associates, Inc.
2309 Renard Pl, Suite 201
Albuquerque, NM 87106
Attn: Mr. Dale R. Atkinson
Mr. Steven F. Rieco
Dr. Alan J. Watts

- 2 Pratt & Whitney Aircraft
400 Main St.
East Hartford, CT 06108
Attn: John Cowles, MS 118-38
Mick Bruskotter, MS 114-38

Dr. Harold E. Read
S-Cubed
P.O. Box 1620
La Jolla, CA 92038-1620

Dr. Douglas Reeder
Hardening Technology Dept.
General Research Corp.
P.O. Box 6770
Santa Barbara, CA 93160-6770

Prof. J. A. Reuscher
Texas A & M
Dept. of Nuclear Engineering
College Station, Texas 77843

- 2 Reynolds Metals Company
1941 Reymet Road
Richmond, VA 23237
Attn: Mr. Stephen P. Sunday
Mr. Armand Beaudoin

Mr. J. S. (Gus) Rice
Caterpillar Inc. Technical Center
Division 927
P.O. Box 1875
Peoria, IL 61656-1875

Mr. Samit Roy
Dept. of Engineering Mechanics
Southwest Research Institute
P.O. Drawer 28510
San Antonio, TX 78284

R. G. Sauvé
Applied Mechanics Section
Mechanical Research Department
Ontario Hydro
700 University Avenue
C26
Toronto, Ontario M5G 1X6
Canada

Mr. Donald W. Sandidge
AMSMI-RLA
U.S. Army Missile Command
Redstone Arsenal, AZ 35898-5247

Mr. Steven Sauer
Ktech Corporation
901 Pennsylvania Ave NE
Albuquerque, NM 87110

Mr. Martin Schmidt, M/S 4G09
WL/MNSA
Eglin AFB, FL 32542-5434

Mr. Luka Serdar, Jr.
Kaman Sciences Corporation
AviDyne Office
83 Second Ave
Burlington, MA 01803-4479

Mr. Harvey Singer
Science Applications International
P.O. Box 1303
McLean, VA 22102-1303

Mr. Mark E. Smith
Arvin Calspan Corp.
AEDC Division, M/S 440
Arnold AFB, TN 37389-9998

Mr. Ray Stoudt
Lawrence Livermore National Lab
P.O. Box 808, L200
Livermore, CA 94550

Prof. D. V. Swenson
Mechanical Engineering Department
Kansas State University
Manhattan, KS 66506

Mr. David W. Sykora
U.S. Army Corps of Engineers
Waterways Experimental Station
P.O. Box 631
Vicksburg, MS 39180

Mr. Sing C. Tang
Rm 3039 Scientific Lab
P. O. Box 2053
Dearborn, MI 48121-2053

2 Spokane Research Center
U.S. Bureau of Mines
315 Montgomery Avenue
Spokane, WA 99207-2291
Attn: Mr. J. Donald Dixon
Dr. Hamid Maleki

2 TRW Ballistic Missiles Division
Bldg 527, Rm 709
P.O. Box 1310
San Bernadino, CA 92402
Attn: Dr. Mike Katona
Mr. Richard Lung

4 United Technologies Research Center
411 Silver Lane
East Hartford, CT 06108
Attn: Dave Edwards, MS 129-13
Robert LaBarre, MS 129-20
Tony Giamei, MS 129-22
Tom Vasko

2 Department of Applied Mechanics
and Engineering Sciences
University of California San Diego
La Jolla, CA 92093
Attn: Prof. S. Nemat-Nasser
Prof. Dave Benson

3 Department of Aerospace
Engineering and Engineering
Mechanics
The University of Texas at Austin
Austin, TX 78712-1085
Attn: Prof. E. B. Becker
Prof. J. T. Oden
Prof. M. Stern

Mr. David Wade, 36E
Bettis Atomic Power Laboratory
P.O. Box 79
West Miffland, PA 15122

Dr. Krishan K. Wahi
Gram, Inc.
1709 Moon NE
Albuquerque, NM 87112

Dr. Paul T. Wang
Fabricating Technology Division
Alcoa Technical Center
Alcoa Center, PA 15069

Dr. Ted B. Wertheimer
MARC Analysis Research
Corporation
260 Sheridan Ave, Suite 309
Palo Alto, CA 94306

4 Westinghouse Electric Corporation
Bettis Atomic Power Laboratory
P.O. Box 79
West Mifflin, PA 15122-0079
Attn: Todd Hoover
Claire Knolle
Dan Kotcher
Wayne Long

Prof. Tomasz Wierzbicki
Dept. of Ocean Engineering
Massachusetts Institute of
Technology
Cambridge, MA 02139

Prof. John Wilson
Department of Geoscience
NM Institute of Mining &
Technology
Socorro, NM 87801

Mr. Philip J. Winters
Chicago Bridge & Iron
1501 North Division Street
Plainfield, IL 60544

Dr. John F. Wohler
Land Systems Division
General Dynamics
P.O. Box 2074
Warren, MI 48090-2074

Dr. Albert Yao
SMCRI-SEE-A
Rock Island Arsenal
Rock Island, IL 61299-5000

Mr. Jerry Zimmerlee
Manager, Engineering Analysis
Johnson Controls, Inc.
P. O. Box 8010
Plymouth, MI 48170

	Mr. J. A. Zukas		6313	John Holland
	Computational Mechanics		6313	Joseph Jung
	Consultants, Inc.		6313	John Pott
	8600 La Salle Road, Suite 614		6313	Alex Treadway
	Towson, MD 21204		6316	Mike Wernig
			6500	James Rice
			6514	Jim Fisk
			6515	Mike Rightley
			6522	Joel Miller
			6642	Douglas Ammerman
	1200 George Allshouse		5 7141	Technical Library
	1239 Frank Dempsey		7151	Technical Publications
	1400 Ed Barsis		10 7613-2	Document Processing for DOE/OSTI
	1425 Johnny Biffle		8523-2	Central Technical Files
10	1425 Stephen Attaway		8702	Bill Robinson
	1425 Mark Blanford		8712	Kim Mahin
	1425 William Bohnhoff		8240	George Johnson
50	1425 Marilyn Smith		8741	Michael Chiesa
	1431 Michael McGlaun		8742	Bruce Kistler
	1431 James Peery		8743	Melvin Callabresi
	1434 David Martinez		8743	Douglas Bammann
	1500 D. J. McCloskey		8743	Juanita Benson
	1501 Carl Peterson		8743	Lee Bertram
	1502 Paul Hommert		8743	Mark Horstemeyer
	1511 J. S. Rottler		8743	James Lathrop
	1512 A. C. Ratzel		8744	Arthur Ortega
	1513 R. D. Skocypec		8745	William Winters
	1551 W. P. Wolfe			
	1552 C. E. Hailey			
	1553 W. L. Hermina			
	1554 W. H. Rutledge			
15	1561 Harold Morgan and Staff			
10	1561 Martin Heinstein			
14	1562 Robert Thomas and Staff			
2	1562 Frank Mello			
2	1562 Jeff Swegle			
	1832 Jeanne Ramage			
	1912 William Mason			
	2565 Stephen Montgomery			
	2814 Randall Lober			
	5600 Dennis Hayes			
	5941 John Schamaun			
	6000 Dan Hartley			
	6112 Dale Preece			
	6112 Norman Warpinski			
	6113 Stephen Bauer			
	6113 Brian Ehgartner			

THIS PAGE INTENTIONALLY LEFT BLANK