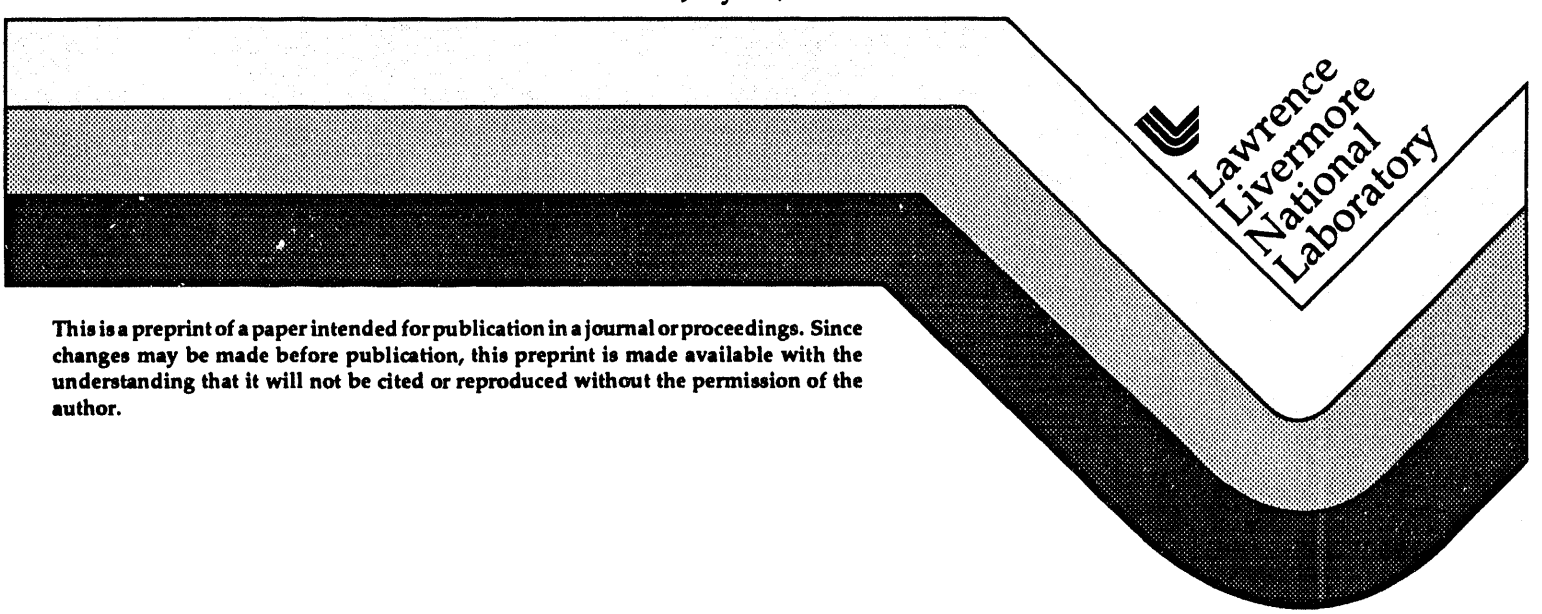
UCRL-JC-114598
PREPRINT

Automatic Contact in DYNA3D for Vehicle Crashworthiness

R. G. Whirley
B. E. Engelmann

This paper was prepared for submittal to the
1993 ASME Winter Meeting
New Orleans, LA
November 28 - December 3, 1993

July 15, 1993



DISCLAIMER

Automatic Contact in DYNA3D for Vehicle Crashworthiness

Robert G. Whirley
Bruce E. Engelmann

Methods Development Group, L-122
University of California
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, California 94550

ABSTRACT

This paper presents a new formulation for the automatic definition and treatment of mechanical contact in explicit nonlinear finite element analysis. Automatic contact offers the benefits of significantly reduced model construction time and fewer opportunities for user error, but faces significant challenges in reliability and computational costs. This paper discusses in detail a new four-step automatic contact algorithm. Key aspects of the proposed method include automatic identification of adjacent and opposite surfaces in the global search phase, and the use of a smoothly varying surface normal which allows a consistent treatment of shell intersection and corner contact conditions without ad-hoc rules. The paper concludes with three examples which illustrate the performance of the newly proposed algorithm in the public DYNA3D code.

1. INTRODUCTION AND MOTIVATION

Almost all finite element models of vehicle crashworthiness involve contact to some extent. This contact may arise from many sources, including impact of some part of the vehicle with a barrier or internal contact between two objects within a crushing cavity. Self-contact may arise between two surfaces of a single body due to local buckling and folding deformation. The accurate treatment of these contact problems is clearly an important aspect of vehicle crashworthiness analyses. The dynamic nature of vehicle crashworthiness contact problems, combined with their complex geometry and highly nonlinear material behavior, makes them well suited for analysis using an explicit nonlinear finite element code such as DYNA3D (Whirley and Hallquist, 1991) from Lawrence Livermore National Laboratory.

Contact problems are traditionally modeled within nonlinear finite element codes by defining a list of nodes or segments (element faces) comprising a contact surface, and then defining which contact surfaces should be checked against each other for contact detection. This approach has the advantage that the analyst can use intuition and insight to minimize the size of the contact surface area. Since contact represents a significant fraction of the cost of a crashworthiness analysis, this is an important factor in minimizing the overall cost of running the analysis.

Recent trends toward vastly increased computer power have motivated a re-examination of this paradigm. Today's computers are capable of handling much larger problems than the machines of just a few years ago, and now the cost of the analyst's time in preparing the analysis model is becoming the dominant cost in performing the analysis. Since the manual definition of contact surfaces in a complex model can require a substantial amount of an analyst's time,

the total cost of performing an analysis could be reduced if automatic contact could be used. In addition, automatic contact can lead to improved robustness and reliability arising from fewer errors in complex models constructed by less-experienced analysts. Also, automatic contact may be more robust than defined-surface contact in complex crashworthiness models because objects dynamically deform in ways not anticipated by even an experienced analyst, requiring that the model be augmented with new contact definitions and the analysis rerun. Considering these benefits, it now seems natural to examine the question of automating the treatment of contact within a complex crashworthiness model.

There are two major obstacles to widespread use and acceptance of automatic contact treatments. First, concerns exist regarding the robustness of automatic contact on complex crashworthiness problems. These concerns are well founded, since large vehicle models contain some of the most difficult contact situations. However, it is these same large vehicle models which would yield the most time and money savings in reduced model preparation time if automatic contact could be confidently used. The other major concern is that automatic contact will be sufficiently more expensive than defined-surface contact that any savings in manpower costs would be eaten away by increased computing costs. In estimating this impact, it should be recognized that once a large model is constructed, it is usually analyzed several times under differing conditions. Thus, increased computer costs accumulated over the life of the analysis model could potentially outweigh savings in reduced model generation time. In this situation automatic contact may still yield a benefit in reduced analysis turnaround time, even if the analysis cost is marginally increased.

The remainder of this paper presents a new formulation for the automatic treatment of contact in an explicit nonlinear finite element code. The next section presents an overview of the new algorithm, and the subsequent four sections describe the details of each step of the method. The following section presents three numerical examples which demonstrate the performance of the new algorithm on several demanding contact problems. The paper concludes with a summary of current research directions for automatic contact in DYNA3D.

2. OVERVIEW OF NEW CONTACT ALGORITHM

A successful automatic contact algorithm is more than just a surface generation procedure used in conjunction with standard single-surface or two-surface contact treatments. Automatic contact generates much more complicated surface geometries, and mixtures of shell and solid element contact segments in complex surface intersections are much more common than in defined surface contact. This characteristic of automatic contact means that the contact search methodologies must be especially robust to correctly handle these cases, and this requirement has motivated the development of the approach described below.

The contact algorithm presented in this paper is based on the widely used node-on-segment concept. In this approach, the two surfaces to be considered are arbitrarily labeled as a slave surface and a master surface, and slave nodes are prevented from penetrating through the master surface. In the case of automatic contact, each node in the contact surface plays the role of a slave node, and is tested against a set of "master surface regions" defined as the subset of the entire contact surface which is spatially close to the slave node. Variations of this node-on-segment approach have been used with success in defined-surface contact implementations in many nonlinear finite element codes for crashworthiness analysis.

In the following, it is useful to distinguish between the contact *segment*, which is the bilinear isoparametric surface containing the four nodes, and the contact *surface*, which is the actual surface of the material containing any thickness offsets from the segment. A contact surface is therefore coincident with the corresponding contact segment for a solid (continuum) element, but differs by a half-thickness for a shell element.

The complete automatic contact algorithm proposed herein consists of four main steps: contact surface construction, global search, local search, and constraint enforcement. The contact surface construction step generates contact segments for the entire exterior boundary of each

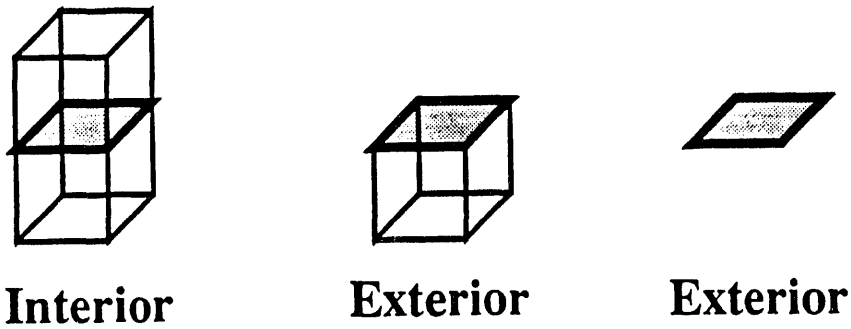


Figure 1: The exterior boundary of a body is found as the set of segments with zero or one continuum elements attached.

body in the problem. The global search step then locates, for each (slave) node, a set of master surface regions which are spatially close and likely to be penetrated. For each slave node, the local search then carefully searches the set of master surface regions to find the contact point. Finally, the constraint enforcement step performs the calculations to transmit interface pressure and prevent interpenetration. Each of these steps is further discussed below.

3. GENERATION OF CONTACT SURFACE

The first step in automatic contact, performed during initialization, is the generation of a contact surface for the entire outer boundary of the model. The outer boundary is determined using the procedure described for adaptive slidesurfaces in (Whirley and Engelmann, 1992). In summary, all segments (element faces) in the model which comprise possible contact segments are first given unique numbers (i.e, segments which differ only in the ordering of their bounding nodes are considered the same face). Next, the number of elements attached to each segment is determined; there is a maximum of two 8-node continuum elements and one 4-node shell element which may be attached to a given segment. The exterior segments are then quickly determined as the set of segments with zero or one continuum elements attached (see Fig. 1). For each exterior segment, a thickness is computed based on the maximum nodal thickness of any node in the segment. An effective stiffness for use in penalty contact calculations is also computed and stored for each segment.

4. GLOBAL SEARCH

The objective of the global search is to identify and store a region of the body where a slave node may potentially come into contact. This is accomplished by defining a set of "master nodes" which are close to a given slave node using the procedure detailed below. The master surface region is then defined as the total area of all contact segments connected to master nodes, as depicted in Fig. 2.

It is not normally necessary to perform the global search at every time step of an explicit transient analysis. Time steps are usually small due to stability limits, and since nodes undergo only small relative motions within a time step, data from a global search remains valid for several steps thereafter. Experience to date indicates that a fixed global search interval of ten steps works reasonably well, but clearly this interval should be chosen based on problem parameters such as peak velocities and element characteristic lengths. General algorithms for selecting the global search interval will be presented in future publications.

The first step in finding the set of "close nodes" is to compute a characteristic length for each node on the contact surface. This nodal characteristic length is found as the maximum

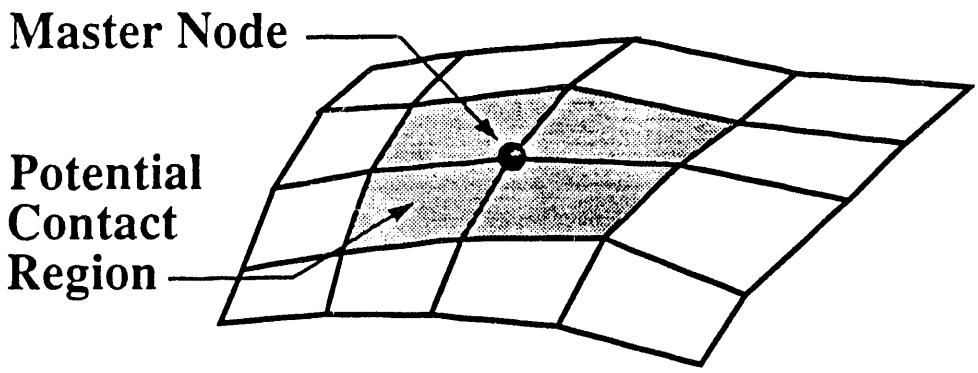


Figure 2: Master surface region in segments around a master node.

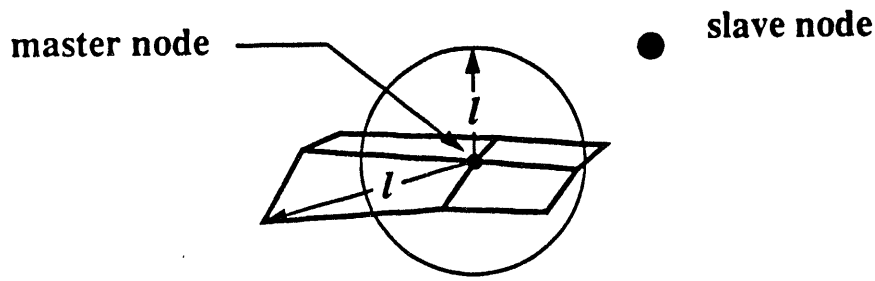


Figure 3: Nodal characteristic lengths are used to limit the number of segments considered for potential contact with a slave node. This figure shows how the master node excluded by this test cannot be connected to a segment in contact with the slave node.

diagonal of any contact segment connected to a node. Slave nodes which are farther away than this characteristic length from a master node cannot possibly be in contact with any segment connected to the master node.

The next step is to perform a set of three nested one-dimensional bucket sorts, as proposed in (Benson and Hallquist, 1990). The nodes from the bucket containing the slave node along with neighboring buckets are then combined to form the "node pool" for that slave node. Next, a "reduced node pool" is formed by eliminating from the node pool any nodes which either share a contact segment with the slave node or which are farther than their characteristic length away from the slave node. as shown in Fig. 3. This second test was motivated by the pinball contact algorithm proposed in (Belytschko and Neal, 1989).

This reduced node pool still contains more nodes than are needed from the global search to define regions of potential contact. Several approaches may be taken to construct the final "master node list" from the reduced node pool. Perhaps the simplest approach, which is economical but lacks robustness, is to simply choose from the reduced node pool some fixed number of nodes which are closest to the slave node under consideration. Experience has indicated that keeping only the one closest node works in some simple cases, and that keeping the three closest nodes works for many problems but may also fail in complex geometries.

The most robust approach which is still cost-competitive is to select for the master node list a minimal set of nodes in the reduced node pool which identifies all potential contact regions. In order to construct this set, it is useful to introduce the concept of *adjacent segment* and *opposite segment* with respect to the slave node. An adjacent segment is a segment from which

the path along the surface of the body to reach the slave node results in a monotonically decreasing distance to the slave node. An opposite segment is a segment which either: (a) lies on a completely separate body from the slave node, or (b) lies on the same body as the slave node but from which the path along the surface to reach the slave node does not result in a monotonically decreasing distance function. It is also useful to introduce the concept of nodal tracking, wherein if contact is not found within a given master segment, then all surrounding segments are evaluated to see if one of them is potentially better. If one appears promising, then that segment is checked, and if nothing is found then its most promising neighbor segment is checked, and so on. This tracking algorithm has the advantage of letting the master segment "track" the slave node motion, but will fail if it can reach the segment containing the slave node (i.e., if applied to an adjacent region). Tracking is particularly useful in extending the interval between global searches without sacrificing robustness of the overall contact algorithm. Thus, it is appropriate to store only the one closest node on each opposite contact region, and use the tracking algorithm to locate the contact point if it lies elsewhere in that region. All reduced pool nodes on an adjacent contact region which are within their characteristic length of the slave node are included in the master node list since tracking cannot be used on these nodes.

For each node in the contact surface (i.e., each slave node), the algorithm used to build the master node list from the reduced pool node list is summarized below:

A. Loop over all nodes which share a segment with the slave node; for each of these nodes use the list of segments connected to this node and flag all nodes of each segment as nontrackable. This prevents the tracking algorithm from finding the slave node in contact with itself.

B. Initialize number of master nodes (for current slave node) to zero.

C. Loop over all nodes in the reduced master node pool

1. Choose the node from the reduced master pool which is closest to the slave node, and mark it as taken.
2. If this node is marked as nontrackable for this slave node and its distance from the slave node is greater than its characteristic length, then discard it and get the next node from the reduced master node pool.
3. If this node is marked as nontrackable for this slave node but within its characteristic length, then store it as a valid nontrackable master node. Then, mark all nodes of all segments connected to this reduced pool node as nontrackable.
4. If this node is marked as trackable for this slave node, then just mark all nodes of all segments connected to this reduced pool node as trackable. This means that a closer node has already been stored for this opposite surface so it is not necessary to store this node.
5. If this node has not yet been marked for this slave node, then this must be the closest node on an opposite surface, so add it to the master node list as a trackable master node. Then, mark as trackable all nodes of all segments connected to this reduced pool node.

The action of this algorithm is illustrated (in two dimensions for clarity) in Fig. 4. The slave node is represented by the large black circle, and reduced pool nodes which are assumed to have passed the characteristic length check are shown as small black circles. First, the algorithm eliminates from consideration reduced pool nodes which share a segment with the slave node, as indicated by the "X" over the two nodes in the figure. Next, reduced pool nodes are considered in order of increasing distance from the slave node, and in Fig. 4 are sequentially labeled m_1 through m_8. Node m_1 is considered first, and has been marked as a nontrackable node in step A of the algorithm because of its proximity to the slave node. Node m_1 is therefore stored as a

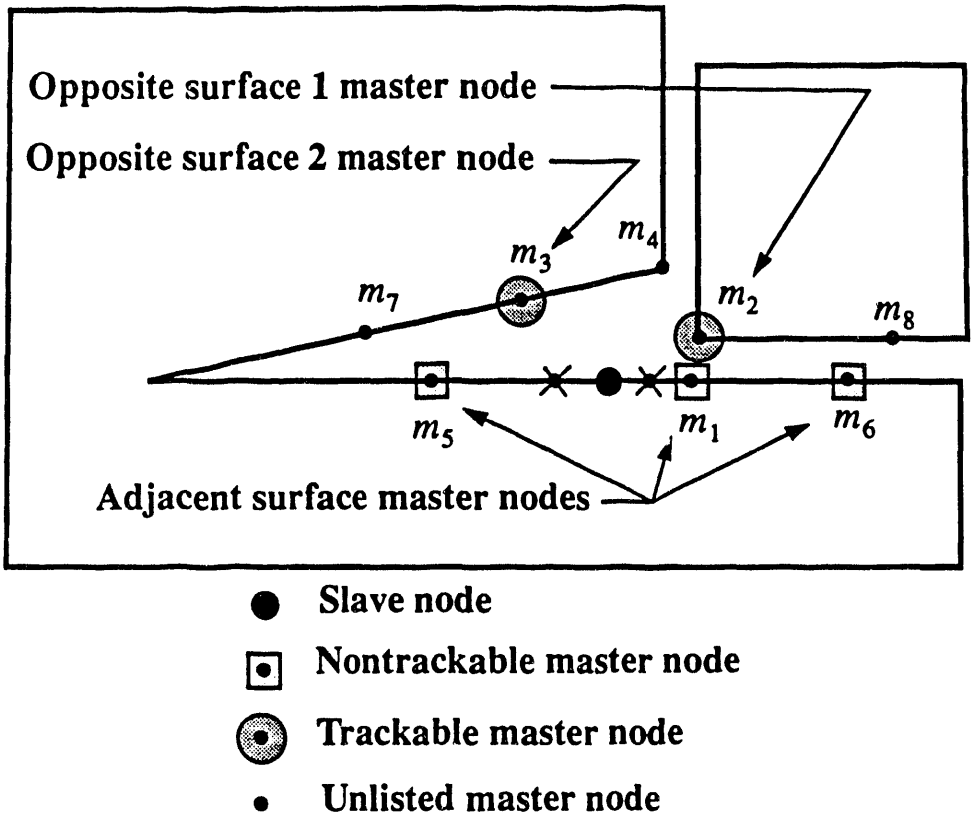


Figure 4: Action of the global search algorithm to construct the master node list from element surfaces with uneven spacing in both self-contact and two-surface contact (shown in 2-D for clarity). This global search procedure also identifies opposite and adjacent contact regions, and trackable and nontrackable master nodes. Nodes included in the final master node list are enclosed by shaded circles for trackable nodes and shaded squares for nontrackable nodes.

nontrackable master node, and all nodes which share a segment with m_1 (only m_6 in this case) are marked as nontrackable. Node m_2 is considered next, and has not yet been marked. Thus, node m_2 must be the closest node on an opposite surface, so it is stored as a trackable master node, and its neighbor nodes (m_8) are marked as trackable. Node m_3 is now considered, and is also found unmarked. Thus, m_3 represents the closest node on another opposite surface, so it is stored as a trackable master node, and its neighbor nodes (m_4 and m_7) are marked as trackable. The global search now considers node m_4, and it is found to have been marked as trackable so it is not stored and its neighbors are marked as trackable (no neighbors of node m_4 are within the characteristic length in this example, so no action is taken). Node m_5 is considered next, and is found marked as a nontrackable node from step A, so it is stored as a nontrackable master node. Node m_6 is treated similarly. Node m_7 is found marked as trackable (by operations on node m_3), and has no neighbors so no action is taken. Node m_8 is also already marked as trackable and has no neighbors, so no action is taken. Thus, the final master node list consists of nontrackable nodes m_1, m_5, and m_6 and trackable nodes m_2 and m_3.

5. LOCAL SEARCH

The objective of the local search is to determine if a slave node is in contact. If contact is found, then the local search algorithm also determines the precise location of the contact point within

a contact segment.

If a slave node was in contact the previous step, then the local search begins by checking the master segment in which contact was previously found. If no contact is found, then all segments connected to the best node of the old contact segment are also checked. It may be desirable to apply an inexpensive screening criterion to these nodes to determine that they are really potential contact regions before performing the detailed penetration check in each of the surrounding segments. If a slave node was not in contact the previous step, or was in contact but the above procedure did not detect contact this step, then the master node list from the most recent global search is used as the starting point. Nodes are selected from the master node list in order of increasing distance from the slave node. Once a master node is chosen, two inexpensive tests are first applied to verify that the slave node could potentially be in contact with one of the neighboring segments. If these tests indicate contact is possible, then all segments connected to the master node are checked for contact with the slave node. If no contact is found anywhere in this process, then the slave node is determined to not be in contact at all during this step. If contact is found, then one of the constraint enforcement algorithms described in a subsequent section is used to satisfy the contact conditions.

The remainder of this section first presents the local segment search procedure. The detailed contact check and contact point calculation procedure used for candidate master segments is then described.

5.1 Local Segment Search

For a slave node which was in contact during the previous time step, the local search begins with a segment contact check in the master segment where contact was found last step. If a contact point is found on the interior of this segment, then the search is terminated unless the segment lies on a shell element and the contact point is within a half-thickness of the segment boundary. For shell segments with contact points within a half-thickness of the boundary, the search procedure must be continued to assure that the correct contact point is located for shell intersections and corners. If a contact point is found and the search terminated, then the selected constraint enforcement procedure is applied.

If contact is not found, the local search is expanded by performing a segment contact check on all segments connected to the closest node of the old contact segment; segments containing the slave node are not considered. At this point it is possible in general to detect multiple contacts, especially if this region of the structure contains corners or shell-solid element intersections, so a hierarchy must be developed to determine the "best" segment to use. This hierarchy is:

1. any shell segment interior contact,
2. a shell segment tube/pinball contact or solid segment interior contact,
3. a solid segment tube/pinball contact.

Segment tube/pinball contact indicates that a contact point was found on a segment boundary, as described in the "Segment Contact Check" section below. If more than one contact segment is found at the same level of the hierarchy, then the segment is chosen containing the contact point closest to the slave node. This corresponds to choosing the least penetrated solid segment or the most penetrated shell segment, and yields the correct behavior in all shell-solid intersections examined thus far. If the best segment found is only a solid segment tube/pinball contact and other solid contact segments were found which were not in contact, then this situation is treated as no contact. If an acceptable contact segment has been found, then the selected constraint enforcement procedure is applied, otherwise the local search is continued as described below.

If the slave node was not in contact last step, or if it was in contact but the above local search procedure did not locate a contact segment for this step, then the general local search is used. The master nodes found in the global search are examined in order of increasing distance from the slave node.

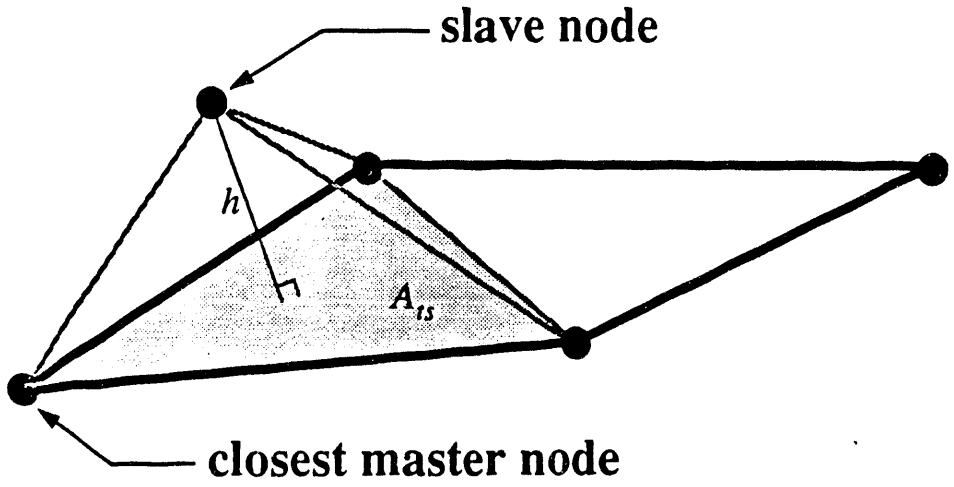


Figure 5: Tetrahedron test used to screen candidate master nodes taken from the global search before applying the segment contact check.

Before applying the detailed segment contact check described below, a master node from the global search is first subjected to two inexpensive screening tests to verify that it does locate a potential contact region. First a "tetrahedron test" is applied to each segment containing the master node by evaluating the volume of a tetrahedron containing the slave node, the master node, and two adjacent nodes from the master segment, as shown in Fig. 5. In the tetrahedron test, a "characteristic height" h is computed from the tetrahedron using

$$h = 3\frac{V_t}{A_{ts}}, \tag{1}$$

where V_t is the volume of the tetrahedron and A_{ts} is the area of the tetrahedron base which is part of the master segment (note that h can be negative). The tetrahedron test is passed (i.e., contact is possible) if, for a master segment on a shell element

$$t_m + t_s \geq h \geq -t_m - t_s, \tag{2}$$

or for a master segment on a solid element

$$t_s \geq h \geq t_s - t_m. \tag{3}$$

Thus, this test is passed if any tetrahedron yields a sufficiently small positive volume or negative volume to indicate that penetration of the slave node through the master surface is possible.

Next, a "bounding box" test is performed by finding the minimum and maximum coordinates in each direction of the master node and its two adjacent nodes in the master segment and adjusting outward using the slave thicknesses. This test is passed if the slave node is contained within any bounding box constructed using the master node.

If a master node passes both of the screening tests, then all segments connected to that master node are evaluated using the segment contact check described below. Once again, it is possible (and likely in complex structures) that multiple contacts will be found, and the hierarchy described above is used to choose the best contact. If more than one contact is found at the same level of the hierarchy, a slightly different resolution procedure is required. First, a comparison vector $\boldsymbol{v}$ is constructed: if the slave node was in contact the previous step, $\boldsymbol{v}$ is defined as the total assembled contact force on the slave node; if the slave node was not in

contact the previous step, then $\boldsymbol{v}$ is defined as the relative velocity between the slave node and the master node,

$$\boldsymbol{v} = \boldsymbol{v}_s - \boldsymbol{v}_m. \tag{4}$$

The appropriate contact segment is then chosen which produces the most negative result of $min(\boldsymbol{v} \cdot \hat{\mathbf{n}})$.

If contact is found, then the master segment number containing the contact point is stored for the current slave node for use in future steps. The selected constraint enforcement algorithm may then be applied.

5.2 Segment Contact Check

The first step in the segment contact check is to compute the local isoparametric coordinates (ξ, η) of the closest point projection of the slave node onto the master segment. Let $\boldsymbol{L}(\xi, \eta)$ represent the parametric equation of the master surface, and let $\boldsymbol{x}_s$ be a position vector to the slave node, then for a closest-point projection the contact point (ξ_c, η_c) must satisfy

$$\frac{\partial \boldsymbol{L}}{\partial \xi}(\xi_c, \eta_c) \cdot [\boldsymbol{x}_s - \boldsymbol{L}(\xi_c, \eta_c)] = 0, \tag{5}$$

$$\frac{\partial \boldsymbol{L}}{\partial \eta}(\xi_c, \eta_c) \cdot [\boldsymbol{x}_s - \boldsymbol{L}(\xi_c, \eta_c)] = 0. \tag{6}$$

These simultaneous equations are solved numerically using Newton iteration.

If the slave node is directly above or penetrated through the master segment, then the above algorithm will yield a contact point (ξ_c, η_c) that is inside the master segment. If the slave node is not directly above or below the master segment, then the above procedure may return a contact point outside the segment. If this happens, then the contact point is returned to the nearest edge of the segment using a closest-point projection, and a flag is set to indicate potential "line contact" if only one coordinate was out of the segment, or potential "point contact" if both coordinates were out of the segment.

Once a contact point has been located, then the return direction is found and a contact check made. First, the thickness of the master segment at the contact point is evaluated based on the isoparametric shape functions and the nodal segment thicknesses. Experience has indicated that accurate calculation of this thickness is essential to the performance of this method. Next, the return direction vector $\hat{\mathbf{n}}$ is calculated. If the contact point was found to be in the interior of the master segment, then the return direction vector is simply the normal to the segment at the contact point. If the contact point was found outside the segment, then the return direction vector is taken from the appropriate "tube" or "pinball" placed on the segment edges and corners as shown in Fig. 6. and a flag is set. The use of the tube and pinball for evaluation of the return direction vector is an essential component of this algorithm. It assures that the normal varies continuously across shell edges, and greatly improves the basis for treatment of complex corner contact geometries without ad-hoc procedures.

The next step in the penetration check is to define a vector $\boldsymbol{t}$ from the contact point on the master segment to the slave node,

$$\boldsymbol{t} = \boldsymbol{x}_s - \boldsymbol{x}_c, \tag{7}$$

where $\boldsymbol{x}_c$ is the position vector to the contact point. Next, compute the distance from the contact point to the slave node, with the sign convention taken from the normal $\boldsymbol{n}$ to the master segment (note that this normal may be arbitrarily oriented and is used only as a reference direction),

$$d = ||\boldsymbol{t}|| \frac{\boldsymbol{n} \cdot \boldsymbol{t}}{||\boldsymbol{n} \cdot \boldsymbol{t}||}. \tag{8}$$

The return direction vector may now be computed by adjusting the orientation of $\boldsymbol{t}$ to be the same as that of $\boldsymbol{n}$.

$$\hat{\mathbf{n}} = \boldsymbol{t} \frac{\boldsymbol{n} \cdot \boldsymbol{t}}{||\boldsymbol{n} \cdot \boldsymbol{t}||}. \tag{9}$$

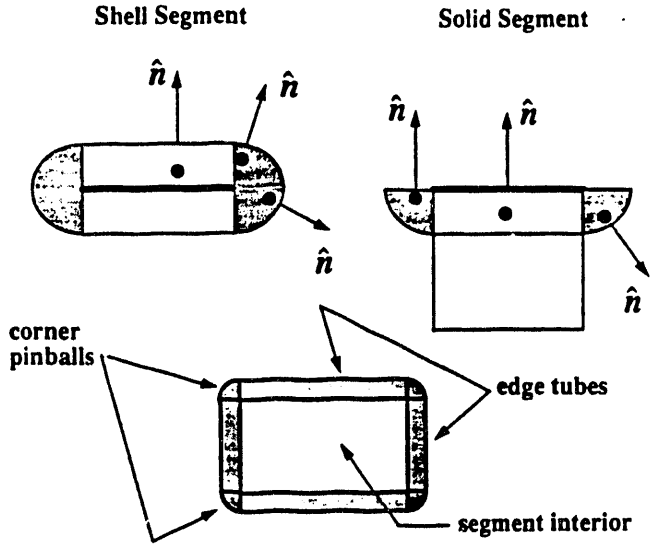


Figure 6: Contact segment showing the use of edge tubes and corner pinballs to compute the return direction vector $\hat{\mathbf{n}}$ if line or point contact is detected.

Note that if $\boldsymbol{n} \cdot \boldsymbol{t} = 0$, then the slave node lies on the master segment, $d = 0$, and Eq. (9) cannot be be used to determine the return direction vector. In this (rare) case, set $\hat{\mathbf{n}} = \boldsymbol{n}$.

The contact surface penetration check may now be performed. Let t_m denote the half-thickness of the master surface at the contact point, and let t_s denote the half-thickness of the slave node. If a contact surface lies on shell elements, then the physical thickness is used, and for solids a representative thickness is chosen as the minimum of 60% of the smallest segment diagonal and the ratio of the element volume to the contact segment area.

If the master segment lies on a continuum element, then contact is detected if $d \leq t_s$ and $d \geq t_s - t_m$. In this case, the penetration depth a considering all surface thicknesses may be computed as

$$a = d - t_s, \tag{10}$$

as shown in Fig. 7. If the master segment lies on a shell element, then contact is detected if $d < t_s + t_m$ and $d > -(t_s + t_m)$. The penetration depth a may now be found from

$$a = d - \frac{d}{|d|}(t_s + t_m), \tag{11}$$

as illustrated in Fig. 8. Finally, an integer flag is computed which indicates which of the four nodes of the contact segment is closest to the contact point, and whether the contact point is within t_m of the segment border. This flag is used in the local segment search when checking old segments as previously described.

6. CONSTRAINT ENFORCEMENT

Once the contact point has been located, a constraint enforcement algorithm must be used to enforce the contact conditions. The proposed formulation has been evaluated in DYNA3D with two constraint enforcement approaches: the penalty method, and a Lagrangian projection method. Each of these is briefly described below.

The penalty method enforces contact conditions by effectively placing a spring between a penetrating slave node and the corresponding master segment. A restoring force $\boldsymbol{f}_s$ is placed on the penetrating slave node to return it to the surface, and an equal and opposite force is

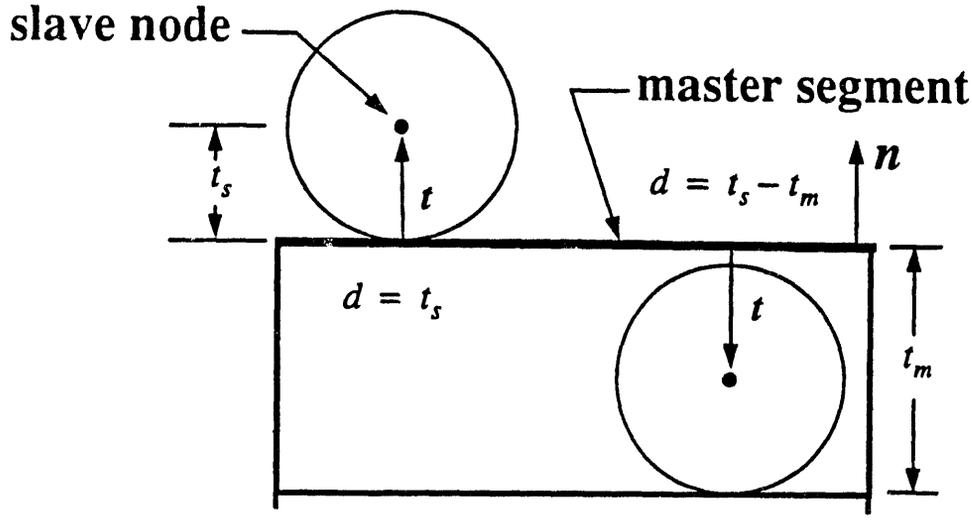


Figure 7: Contact detection test for master surface which lies over a continuum element. Limiting cases of $d = t_s$ (first contact) and $d = t_s - t_m$ (too penetrated to be in contact) are shown.

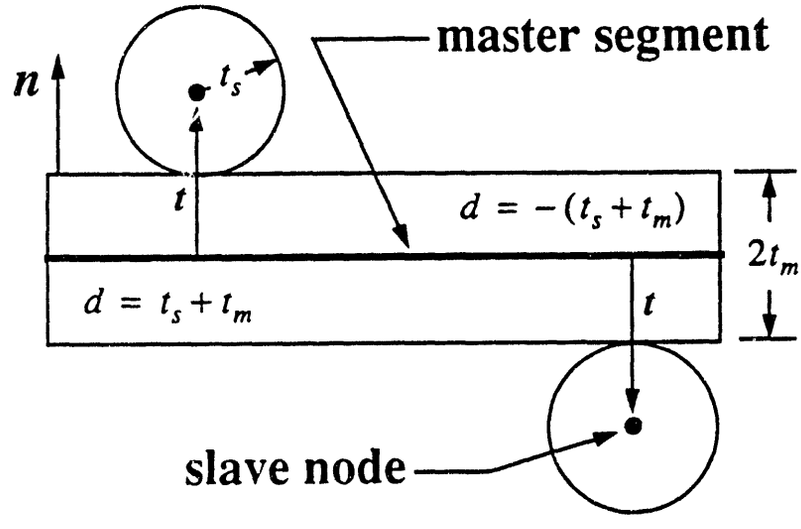


Figure 8: Contact detection and penetration test for master surface which lies over a shell element. Limiting cases of $d = t_s + t_m$ (just in contact with top surface) and $d = -(t_s + t_m)$ (just in contact with bottom surface) are shown.

placed on the master segment at the contact point. Nodal forces on the master segment are then computed by interpolation using the bilinear shape functions of the contact segment.

The slave node restoring force is found from

$$f_s = K a \hat{n}, \tag{12}$$

where K is a penalty stiffness, a is the penetration depth, and $\hat{n}$ is the return direction vector discussed above. The penalty stiffness is computed from the bulk modulus of the master segment material and the effective element thickness.

A Lagrangian projection method has been recently implemented for comparison with the penalty method. In contrast to the penalty method, the Lagrangian projection method gives exact results (zero interpenetration) for node-on-node contact, and has been found to generally yield more accurate pressure distributions than the penalty method. This approach is adapted from the forward-Lagrange method of (Carpenter, Taylor, and Katona, 1991), and is similar to the method proposed in (Belytschko and Neal, 1989). More experience on production DYNA3D analysis problems will be required to fully evaluate the strengths and weaknesses of each method.

7. NUMERICAL EXAMPLES

7.1 Crush of a Wide Flange Beam

This example illustrates the modeling of complex self-contact conditions which occur during the buckling and collapse of a wide flange beam. Although generated somewhat artifically in this example by a pressure load on the upper surface, this type of deformation involving local buckling and sequential contact with several different surfaces separated by corners in a T-intersection is quite typical of the behavior found in large-scale vehicle crashworthiness models. Clearly, the successful solution of this problem requires a robust self-contact capability along with proper treatment of the corner condition at the intersection of the web and lower flange.

A time sequence of deformed geometry plots is shown in Fig. 9. The top left image shows the undeformed geometry, and the top right image shows initial folding of the upper flanges due to a plastic hinge near the web, and initial buckling of the web itself. The lower left image shows a more developed buckle with self-contact in the web, and multiple points of contact between the web and the folded flanges. The upper left flange is about to strike the top surface of the bottom left flange near the T-intersection with the web. The lower right image shows complete collapse of the web and resulting contact with both the lower flange surface and the folded upper flanges in addition to complex self-contact. The folded upper flanges have impacted the lower flange and are sliding outward along its upper surface.

7.2 Wall of Blocks

To illustrate the advantages of automatic contact and the robustness of the proposed algorithm, the impact of a stiff elastic block onto the end of a wall of rigid blocks was simulated in DYNA3D. This problem would be very difficult to solve using defined-surface contact because of the large number of disjoint surfaces and potential contact pairs. In addition, this problem illustrates the importance of keeping an arbitrary number of nodes in the global search since many slave nodes can have several potential master surface regions on opposite surfaces for contact. This problem also exercises most branches of the decision tree for contact at intersections.

The wall of rigid blocks was constructed with a small initial space between blocks to prevent contact from being detected at initialization, and thus forcing the run-time search algorithm to handle the complete contact detection problem. Although the very coarse mesh also causes some inaccuracy in the contact enforcement due to edge-to-edge penetration, this could be removed in a more realistic problem by simply refining the mesh. The coarse mesh is retained in this example, however, because it forces a larger number of master surfaces to be considered for each slave node and increases the number of corner and edge conditions, and is therefore a more demanding test on the search algorithm.

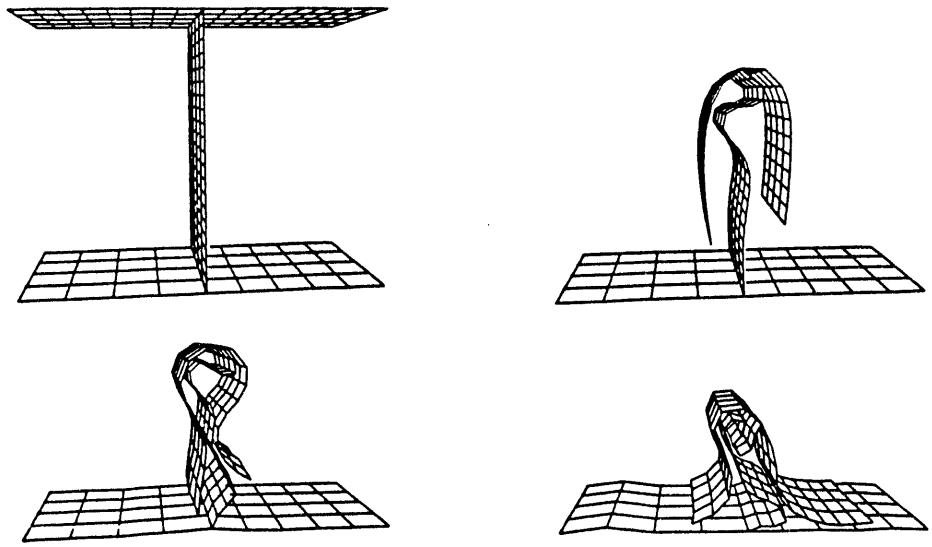


Figure 9: Time sequence of deformed meshes showing collapse of a wide flange beam including buckling. self-contact, edge contact, and sliding surface contact.

A time sequence of deformed geometry plots is shown in Fig. 10. The upper left image shows the initial geometry, and the upper right image shows that the first large motions occur in the lower blocks on the free end, as expected, with small motions evident in the impact area. The lower left and lower right images show the propagation of these disturbances to the first and last columns of blocks, and then to the entire wall. Note that contact conditions are correctly enforced in each of the figures.

7.3 Vehicle Frontal Impact

As a final example, the frontal impact of a vehicle front frame model was simulated using the proposed automatic contact algorithm in DYNA3D. The impact barrier was modeled using a rigid wall unilateral contact option, and the entire model was selected for automatic contact treatment. The original and deformed meshes of the half-symmetry model are compared in Fig. 11. The two front and middle crossmembers are constructed of steel channel sections, with a plate attached between the upper and lower members. The front channel sections collapse upon impact with the barrier and the lower flange of the upper channel folds under and comes into self-contact. A similar self-contact occurs in the upper flange of the lower channel. Numerous two-surface contacts occur during the impact, including those between the upper and lower energy absorbers and the crossmembers, and between the rigid engine block and the plate between the aft crossmembers.

Although this model contains only 2,000 shell elements and is quite coarse, it contains a number of complex shell intersections. The deformed mesh shows that the proposed automatic contact algorithm detected and correctly handled both the self-contact and two-body contact conditions. This analysis required 916 seconds of CPU time on an IBM RS/6000 model 550 workstation, of which approximately one-third was consumed by the automatic contact calculations.

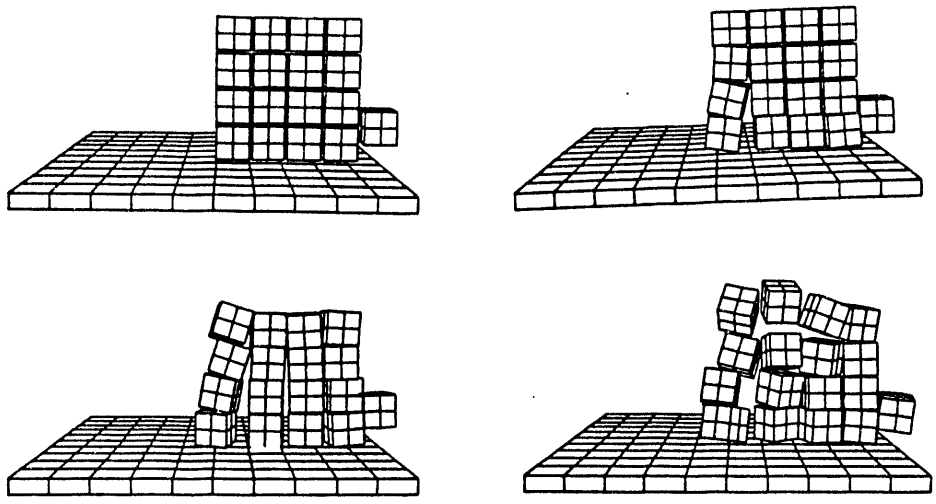


Figure 10: Time sequence showing the impact of a stiff elastic block on a wall of rigid blocks. This problem has a large number of potential contact surface pairs and complex contact conditions on edges and corners, and would be much more difficult to solve using defined-surface contact.

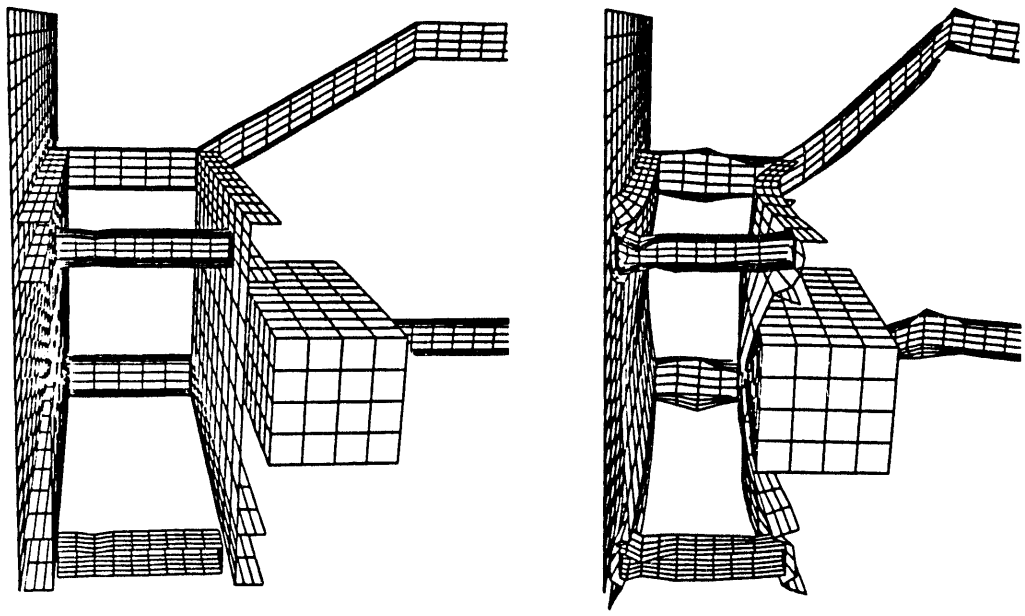


Figure 11: Comparison of original and deformed geometry for a vehicle front-end frame in a frontal impact using automatic contact in DYNA3D. Note the combination of self-contact and two-surface contact in various regions of the model.

8. FUTURE RESEARCH DIRECTIONS

Ongoing research is focused on refinement of several aspects of the algorithm. The minimization of the number of nodes stored by the global search is one area of continuing study. Other efforts concern the proper treatment of hypervelocity impact in the segment contact check. Developments are also underway to permit automatic contact to be activated by material groups in DYNA3D to improve efficiency and reduce costs when only a small part of a model is involved in contact. These developments will soon be available in the production version of DYNA3D. Experience and feedback from DYNA3D outside collaborators will no doubt highlight other areas where improvements can be made and are an important part of this research.

Implementation efforts are well underway to complete the vectorization of the algorithm to fully exploit hardware capabilities on CRAY and other vector machines. Strategies are also being developed to implement the automatic contact algorithm on massively parallel computers using a message-passing programming model which accommodates vector hardware on each processor, if available. These vector and parallel implementations in DYNA3D should allow very high performance computing resources to be successfully applied to complex vehicle crashworthiness problems.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U. S. Department of Energy by the Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

REFERENCES

1. Belytschko, T., and Neal, M. O., "Contact-Impact by the Pinball Algorithm with Penalty, Projection, and Lagrangian Methods," *Proceedings of the Symposium on Computational Techniques for Impact, Penetration, and Perforation of Solids,* ASME AMD Vol. 103, pp. 97-140, 1989.

2. Benson, D. J., and Hallquist, J. O., "A Single-Surface Contact Algorithm for the Post-Buckling Analysis of Shell Structures," *Computer Methods in Applied Mechanics and Engineering,* Vol. 78, pp. 141-163, 1990.

3. Carpenter, N. J., Taylor, R. L., and Katona, M. G., "Lagrange Constraints for Transient Finite Element Surface Contact," *International Journal for Numerical Methods in Engineering,* Vol. 32, pp. 103-128, 1991.

4. Whirley, R. G., and Hallquist, J. O., "DYNA3D: A Nonlinear, Explicit, Three-Dimensional Finite Element Code for Solid and Structural Mechanics - User Manual, University of California, Lawrence Livermore National Laboratory, Report UCRL-MA-107254, May, 1991.

5. Whirley, R. G., and Engelmann, B. E., "Slidesurfaces with Adaptive New Definitions (SAND) for Transient Analysis, *Proceedings of the Symposium on New Methods in Transient Analysis,* ASME AMD Vol. 143, pp. 65-71.

END

DATE
FILMED
3/28/94