DEVELOPMENT AND VALIDATION OF NEW ALGORITHMS TO IMPROVE CONTACT DETECTION AND ROBUSTNESS IN FINITE ELEMENT SIMULATIONS

by

Umashankar Mahadevaiah

B.E. in Mechanical Engineering, August 1999, Bangalore University
M.S. in Civil & Environmental Engineering, August 2003, The George Washington University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University in partial fulfillment
of the requirements for the degree of Doctor of Philosophy

May 17, 2009

Dissertation directed by

Dhafer Marzougui
Assistant Research Professor of Civil Engineering

and

Azim Eskandarian
Professor of Engineering and Applied Science

www.manaraa.com

UMI Number: 3344428

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI Microform 3344428
Copyright 2009 by ProQuest LLC
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

The School of Engineering and Applied Science of The George Washington University certifies that Umashankar Mahadevaiah has passed the Final Examination for the degree of Doctor of Philosophy as of January 23, 2009. This is the final and approved form of the dissertation.

DEVELOPMENT AND VALIDATION OF NEW ALGORITHMS TO IMPROVE CONTACT DETECTION AND ROBUSTNESS IN FINITE ELEMENT SIMULATIONS

Umashankar Mahadevaiah

Dissertation Research Committee:

Dhafer Marzougui, Assistant Research Professor of Civil Engineering,

Co- Director

Azim Eskandarian, Professor of Engineering and Applied Science,

Co-Director

Majid Taghizadeh Manzari, Professor of Civil Engineering,

Committee Member

Sameh S. Badie, Associate Professor of Civil Engineering,

Committee Member

Cing-Dao (Steve) Kan, Associate Research Professor of Engineering,

Committee Member

www.manaraa.com

Acknowledgments

First of all, I would like to express my sincere gratitude to Dr. Dhafer Marzougui, who has been my research advisor and supervisor since the beginning of my study. He has been a friend, a teacher and a colleague and provided me with many helpful suggestions, advice and constant encouragement during the course of this work.

I also would like to express my appreciation to my advisor Prof. Azim Eskandarian for his academic supervision, personal support and constant motivation throughout all my years in graduate school.

Special thanks are due to Dr. Cing-Dao Kan for his support throughout my graduate studies. I would like to acknowledge the financial, academic and technical support he provided me as the Director of the National Crash Analysis Center.

My keen appreciation goes to Prof. Majid Manzari, Prof. Samie Badie and Dr. Kenneth Opeila for taking time from their busy schedule to serve as my committee members.

Sincere thanks are extended to Jason Mader, Pradeep Mohan, Murat Buyuk and all my other colleagues and friends at the National Crash Analysis Center for their assistance and willingness to share their knowledge and skills.

Lastly, but most importantly, my special appreciation goes to my family for the love and support they provided me through my entire life. My gratitude to them is beyond words.

www.manaraa.com

Abstract of Dissertation

Development And Validation Of New Algorithms To Improve Contact Detection And Robustness In Finite Element Simulations

Approximately half of all numerical problems in crashworthiness analysis involve impact dynamics, and accurate contact algorithms are critical to capture the structures' behavior. Conventional contact algorithms use the principle of preventing 'slave' nodes from penetrating 'master' segments. Only nodes are checked in these contact algorithms and the connectivity of the nodes (in the slave side) are not considered. Additionally, to achieve efficiency, the conventional contact algorithms use different methods to eliminate element pairs that would unlikely come in contact and simplify the geometry while searching for penetration between the contact pairs. These eliminations and simplifications, sometimes, cause inaccuracy in the results.

In this research, a new contact algorithm has been developed and implemented in an explicit nonlinear large displacement finite element code (DYNA3D). A new global search method and a new local search method for contact search have been implemented in the algorithm. The new global search method uses the concept of enclosing spheres around nodes combined with bucket-sorting. Unlike in the current algorithms where bucket-sort checks for presence of nodes in the buckets, bucket-sort in the new global search check for intersections of enclosed spheres with the buckets. In the new local search method, effort is made to represent accurate geometry of the contact surface. The element surfaces are offset by their thickness and, edges and corners are represented

www.manaraa.com

using beams of circular cross-section and spheres respectively. Using this configuration, problems associated in finding penetration in a skewed mesh are eliminated.

Constant stiffness that is used in computing contact force in current contact algorithms is replaced by exponentially varying stiffness in the new contact algorithm. When compared to the constant stiffness, the varying stiffness applies significantly higher forces when the penetration becomes large.

The new contact algorithm has been implemented in DYNA3D and validated. Element level and component level examples have been used to check accuracy of the contact algorithm. Using these examples, gap between the contact surfaces and stress variation along the contact surface are checked. Using the new contact algorithm, the gap distance was found to be accurate and stress variation was found to be minimal.

The new contact algorithm has few limitations which need to be addressed before it can be used to solve general three dimensional problems. Provisions should be made to the contact algorithm to include segments from solid elements, and rectangular & varying cross-sectional beam elements in the contact and to delete failed elements from the contact. Care should be taken not to include severely warped elements and initially penetrated elements in the contact definition.

www.manaraa.com

Table of Contents

www.manaraa.com

www.manaraa.com

www.manaraa.com

List of Figures

www.manaraa.com

www.manaraa.com

www.manaraa.com

List of Tables

www.manaraa.com

1. INTRODUCTION

1.1 Background

Automobile crashworthiness and highway safety have been receiving significant attention in the past several years. The term 'crashworthiness' is understood to denote the ability of a vehicle structure and any of its components to deform plastically and yet maintain a sufficient survival space and thus protect the occupants in survivable crashes. Not only good design of automobiles, but also better design of highway and roadside hardware are required. Roadside hardware such as effective traffic barrier system, crash cushions, end terminals, break-away devices, truck-mounted attenuators and others must be used to achieve the highest levels of highway safety.

Crashworthiness of a vehicle and performance of roadside hardware have been studied mainly by physical testing. However, physical testing along with analytical or numerical simulations are preferred since it is cost-effective and can obtain much more detailed information on the involved phenomena. Three types of analytical models are used to simulate vehicle structures – Lumped Parameter (LP) models, Finite Element (FE) models and hybrid models. Over the years, these models progressed from simple analytical model, tuned one or more parameters to fit a specific test, to a complex model with great geometric details and material properties. Even then, the most detailed models (LP or FE) developed to date are considered approximations of a highly complex non-linear system that is often subject to large non-linear elastic-plastic deformations. Hence,

www.manaraa.com

advances in understanding complex system performance such as crashworthiness can be achieved by increasingly including more details and making the analytical models represent as close as possible the physical components of the actual structures in geometry, material characteristics, connections, and contact interactions.

It is important to note that numerical simulations are not aimed at lowering the normal workload of test laboratories. Testing will still be needed for the verification and certification of vehicle prototypes or safety systems for many years to come. The contribution of simulation lies in that it complements the testing by making the design and analysis process more efficient and cost effective. The strength of simulation lies in rapidly performing simulations in the form of parametric studies that allow quick elimination from prototyping those designs which have a high probability of not satisfying the testing criteria. The ideal process is one of a design, heavily supported by analysis, resulting in building of only those prototypes or systems that are almost certain to pass final verification testing. When a safety-related problem appears in a prototype/safety-system during a test, it is simulation that allows for diagnosis of the cause of the problem and selection of an appropriate structural modification in a minimal amount of time. Extensive use of numerical simulation has enabled the safety engineers and motor vehicle industry to make increasingly safer roads and automobiles in less time without a significant increase in testing costs.

1.2 FEM in Engineering Analysis

The finite element analysis (FEA) is firmly established as a powerful and popular analysis tool. It provides solutions to problems that would be difficult to solve by classical analytical methods. At present, it is applied to many different problems of

www.manaraa.com

continua but is most widely used in engineering analysis of solids and structures. Testing of prototypes is increasingly replaced by simulation with nonlinear finite element analysis because this provides a more rapid and less expensive way to evaluate design concepts and design details. FEA uses numerical technique called finite element method (FEM).

The object or system to be analyzed is represented by a geometrically similar model consisting of multiple, linked, simplified representations of discrete regions called "elements". The elements are connected to one another at points called nodes or nodal points. Each node has various degrees of freedom (d.o.f) depending on the type of analysis and physical constraints applied on it. Equations of equilibrium, in conjunction with applicable physical constraints are applied to each element/node, and a system of simultaneous equations is constructed. These equations are solved to determine unknown values of d.o.f using appropriate numerical techniques.

Analysis can be classified into linear analysis or nonlinear analysis depending on the problem. Nonlinearity in the problem can be due to material nonlinearity or geometric nonlinearity. Analysis can also be classified into static analysis or dynamic analysis depending on the loading conditions.

Finite element analyses of vehicle crashworthiness and evaluations safety performance of systems are among the most challenging nonlinear problems in structural mechanics. The solution obtained from a finite element simulation is an approximation of the exact solution. A finite element simulation can be seen as a chain with two objects. The first link is the numerical model, essentially a complicated spring-mass system

www.manaraa.com

whose dynamic behavior is an approximation of the continuum that is to be modeled. The second link is the software or the numerical algorithm that has to perform a numerical time integration of ordinary differential equations that govern the behavior of the model.

1.3 Problem

Approximately half of all numerical problems in crashworthiness analyses involve impact dynamics and accurate contact algorithms are critical to capture the structures' behavior. Conventional contact algorithms use the principle of preventing 'slave' nodes from penetrating 'master' segments. In other words, a well-defined set of nodes is not allowed to penetrate an equally well-defined set of segments. If the nodes and segments are on different surfaces, a master-slave contact definition is used. If they are on the same surface, a single surface contact definition, where each node of the surface is not allowed to penetrate any segment on the same surface that is not connected to that node, is used. Only nodes are checked in conventional contact algorithms and the connectivity of the nodes (in the slave side) are not considered in the contact algorithm.

Edge-to-edge penetration was not a significant problem in early simulations because contacts between convex surfaces with low curvature were considered and consequently node-to-segment contact algorithms have ability to detect all the occurring penetrations. This is no longer the case for certain structures such as automotive structures. These structures have complex surfaces with high curvatures resulting in surfaces with several kinks and edges. In these situations, preventing the nodes from penetrating the segments is not sufficient to keep the surfaces from penetrating each other. Edge-to-edge penetrations can occur and go undetected causing nodes to move to the opposite side of the segments. Further movements of the penetrated nodes often lead

www.manaraa.com

to high contact forces and consequently unrealistic response of the structure. Figure 1-1 shows some of the problems with conventional contact algorithms.

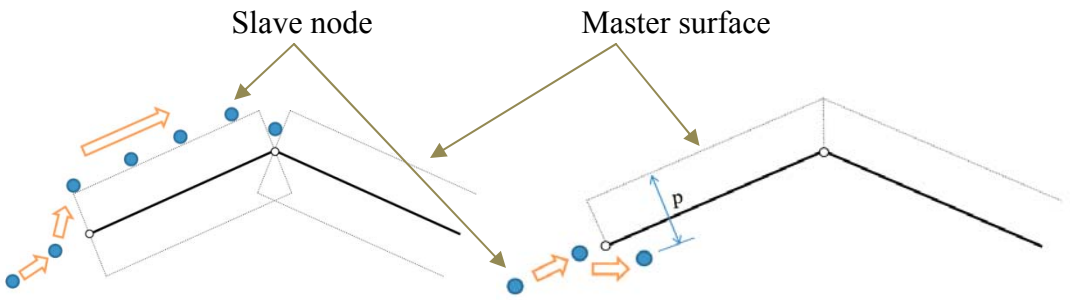


Figure 1-1 Problems with conventional contact algorithms

Additionally, to achieve efficiency, current contact algorithms simplify the geometry while searching for contact pairs and eliminate element pairs from local search that would unlikely come in contact. Depending on the elimination process used by the contact algorithms, sometimes element pairs that would come in contact are eliminated. The simplification of geometry and elimination of contact element pairs cause inaccurate results.

Hence to make numerical algorithms much more robust, it is necessary to have an accurate contact algorithm which (i) identifies all contact possibilities, (ii) accurately checks for contact and penetration between the segments (surface to edge as well as edge to edge) at all times and (iii) eliminates numerical instability due to high contact forces.

Although much has been done in the development of contact algorithms, there is still scope for improvement regarding both the efficiency and the reliability of the algorithms. An ideal contact algorithm is one which is as accurate as the brute search

www.manaraa.com

method (brute search method is one in which every node/element is checked for penetration against every other element in every cycle) with a reasonable amount of computation time.

1.4 Contributions of this research

The objective of this study is to develop a contact algorithm that addresses some the deficiencies mentioned above. The algorithm is implemented by adding several FORTRAN subroutines in the general finite element code DYNA3D (Whirley, 1993). New methods for global search and local search that resolve the contact issues are developed. In the new global search method, efforts are made to identify every element pair that could come in contact. In the new local search method, accurate representation of geometry is considered while searching for penetration. In this research, only 1-dimensional beam elements and 2-dimensional shell elements are considered in the contact. The algorithm is however coded in such a way that the 3-dimensional solid elements can be easily included in the contact. Additionally, in this research, more focus is given to the accuracy of the algorithm than its efficiency. The developed subroutines can be further optimized, by software developers, to minimize computational costs.

1.5 Research overview

Different steps are involved in developing the new contact algorithm and implementing it in DYNA3D and these steps are presented in the following chapters. In Chapter 2, a brief overview of finite element method theories and contact-impact are given. An extensive review of literature on contact mechanics and the recent work in the same area is presented. The concept behind various contact algorithms that are currently

www.manaraa.com

available is presented along with their advantages and limitations. Theory behind different contact enforcing mechanics is also presented.

In Chapter 3, an overview of the non-linear finite element code "DYNA3D" that is used in this study is presented. Some of the theories used to implement the various features in DYNA3D are explained along with a summary of DYNA3D code.

In Chapter 4, the concept used in the proposed contact algorithm is explained. The methods incorporated for the global and local searches are explained in detail. The subroutines that are added and subroutines that are modified to implement the contact algorithm in DYNA3D are presented.

Next step in developing the algorithm involved validation by simulating problems that have analytical solutions. In Chapter 5, problems that are used in the validation process and results from the simulations are presented.

Several examples were simulated using the proposed contact algorithm to show the differences and improvements over current DYNA3D contact algorithms. The results from these comparisons are presented in Chapter 6.

Finally, concluding remarks and recommendations for future research are presented in Chapter 7.

www.manaraa.com

2. LITERATURE REVIEW

Accurate and efficient contact algorithms play an important role in structural analyses and vehicle crashworthiness simulations. Hence improvement of existing contact algorithms and development of new ones have been given importance in the field of finite element analysis.

Contact algorithms have seen significant improvements since the beginning of finite element computer programs in late 1960s and early 1970s. There has been numerous published works on numerical methods of analysis of contact interactions. All relevant work has been reviewed and analysis of various contact search methods is presented in this chapter.

2.1 Review of contact algorithms

Contact algorithms can be broadly classified according to the concept utilized for the description of motion of a continuous medium: Lagrangian and non-Lagrangian. In Langrangian contact algorithms, the nodes move with the velocity of the material medium. In non-Lagrangian, the nodes either are fixed (Eulerian algorithms) or move independently of the material medium (Arbitrary Lagrangian-Eulerian, ALE, algorithms).

The contact algorithm may be considered as consisting of two parts. The first part is a search algorithm which is used to detect and measure overlap or interpenetration of regions of the structure. The second part accounts for the mechanics of contact by

www.manaraa.com

applying appropriate interface tractions between surfaces in contact and, depending on the algorithm in use, may modify displacements, velocities, and/or accelerations to be consistent with the current contact constraints. Contact algorithms can also be classified into two main categories based on the procedure they use to prevent interpenetration: Lagrange multiplier method and penalty method. Two more methods are derived from these methods and they are augmented Lagrangian method and perturbed Lagrangian method. These methods are explained later in section 2.1.3.

The current contact algorithms use the principle of preventing "slave" nodes from penetrating "master" segments. Slave and master are designations given to distinguish two bodies or entities or elements. A segment corresponds to a 4-node shell, a 3-node shell or a face of a brick element. The number of operations required for searching for contact pairs, "node-segment", is proportional to the square of the number of contact segments or nodes. For problems with large number of nodes, this process requires significant computational effort and may lead to non-practical times for solving the problem. Hence, the search process is divided into two or more levels to accelerate the process of detection of contact pairs. The two levels are usually referred to as the global search and the local search.

On the global search level, the regions of possible contact are searched among groups of neighboring nodes. The groups of nodes that lie far away from the region of possible contact and, therefore, not involved in the contact are rapidly discarded. On the local level, contact pairs are identified by the violation of impenetrability constraint or by a sufficient proximity criterion.

www.manaraa.com

Few of the well-known global search algorithms are, the bucket sorting algorithm (Belytschko, 1987; Benson, 1990; Hallquist 2005), the spherical sorting algorithm (Papadopoulos, 1993), the hierarchy-territory algorithm (Zhong, 1996), and the linear position code algorithm (Oldenburg, 1994). Several local search algorithms that have been proposed are pinball algorithm (Belytschko, 1989; Hallquist, 1985), node-to-segment algorithm (Belytschko, 1991; Hallquist 2005), and inside-outside algorithm (Wang, 1997). Based on the similar concepts as these local search algorithms, several other algorithms have been developed to improve accuracy and efficiency of the search. To name a few, free-formed-surface (FFS) algorithm (Wang, 2001), no-binary-search (NBS) algorithm (Munjiza, 1998), algorithm using space-filling curve (SPC) (Diekmann, 2000), direct localization using pinball algorithm (Petkevicius, 2003), splitting pinball algorithm (Belytschko, 1993) and parallel contact algorithm (Malone 1994). Each of these contact algorithms is described briefly with their strengths and weaknesses.

2.1.1 Global Search Algorithms

2.1.1.1 Bucket-Sort Algorithm

Bucket-sort algorithm (Hallquist, 2005) is the most commonly used global search algorithm and it generates a reasonable neighborhood definition. The contact surface is divided into number of 'buckets'. The term 'bucket' is used instead of 'neighborhood' in computer science literature. Each node is assigned a bucket number and all nodes with the same number define a neighborhood. The buckets are sized such that if a node is compared to all of the segments having nodes in its bucket to the left or the right, all possible overlap pairs are obtained. In two and three dimensions buckets are nested. In

www.manaraa.com

two dimensions, each bucket is a square, and is surrounded by eight neighbors. Each node is compared against all the nodes in its bucket and the eight neighboring buckets. In three dimensions, each bucket is a cube and surrounded by 26 buckets, forming a 3 x 3 x 3 cube. The dimensions of the buckets do not have to be the same in all directions. Figure 2-1 shows one-dimensional, two-dimensional and three-dimensional bucket sorting.

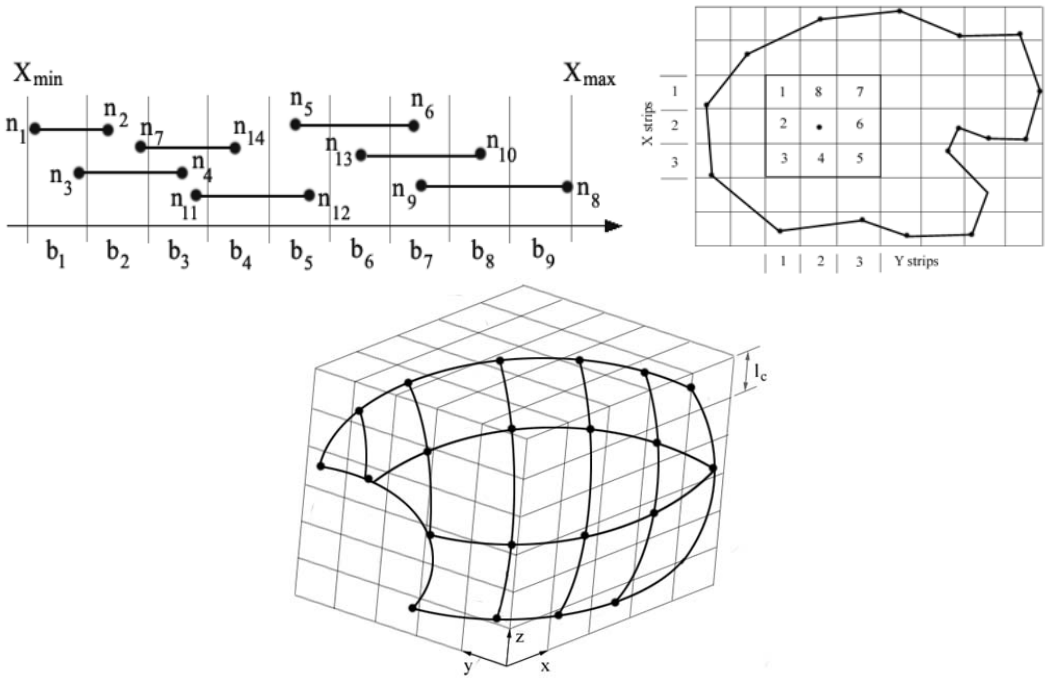


Figure 2-1 One-, two- and three-dimensional bucket sorting

The smaller the bucket-size the fewer nodes in each bucket, and consequently the smaller the number of segment comparisons. A smaller bucket size, however, allows the possibility of missing an overlapping/intersecting pair.

Sorting is not done at every time step in explicit analysis as it becomes computationally very expensive. The results of a sort are used for several time steps

www.manaraa.com

because the incremental displacements over a time step are small relative to the mesh dimensions. But in implicit analysis, where the time step is significantly larger and the displacements are greater over each time step, a complete sort is performed at the beginning of every time step. In an explicit calculation, the three nearest neighbors are stored after each sort, and the closest node is calculated from them each time step. No local search is performed if the nearest neighbor is greater than a bucket away. By adopting this strategy, a complete sort is done every five to twenty time steps.

The main advantage of this algorithm is that it is about 150 to 800 times faster than a brute force global search when it is three-dimensional analysis. A drawback of this algorithm is that it is likely to fail if the contact surfaces become highly distorted compared to their initial configurations, due to the restriction of the search to the closest neighborhood of each contacting node.

2.1.1.2 Spherical-Sorting Algorithm

Spherical-sorting algorithm (Papadopoulos, 1993) uses the idea of a pinball algorithm (Belytschko, 1989) to decide whether a local search is needed between a contact pair. Pinball algorithm is explained in detail later in this chapter. In spherical-sorting algorithm every element face is superscribed in an imaginary sphere of the smallest possible radius. Then every pair of faces from the two surfaces is checked for intersection of corresponding spheres. Non-intersecting pairs are discarded and intersecting pairs are checked for local contact. Figure 2-2 shows example of a non-intersecting pair and an intersecting pair.

www.manaraa.com

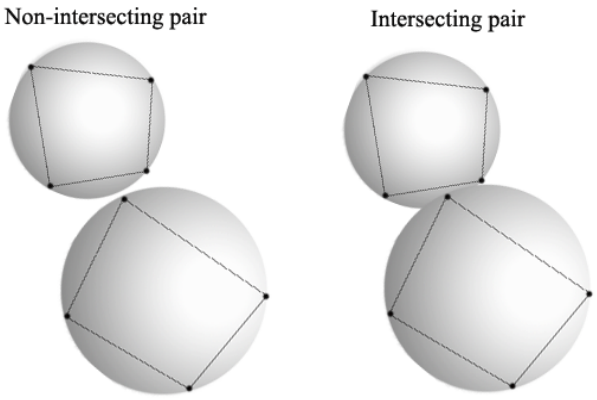


Figure 2-2 Non-intersecting and intersecting pair

The advantage of this algorithm is that it is quick to decide if a local search is needed between a contact pair once the radii of enclosed spheres and their centers are computed. But the disadvantage is that every pair is checked for intersection at every cycle. Even though it is simple to check a pair just by finding distance between centers and comparing it with sum of radii of the pair, a large number of elements in the contact will substantially increase the time required for each cycle compared to bucket-sort.

2.1.1.3 Hierarchy-Territory Algorithm

In Hierarchy-territory algorithm (HITA) (Zhong, 1996; Belytschko, 1993), hierarchical relation in a contact system along with contact territory is used. Contact hierarchy and contact territory are defined.

A contact system may contain one or more contact bodies, a contact body may contain one or more boundary surfaces, a contact boundary may be divided into several contact segments, and a contact segment may contain three or more contact edges. A contact segment is defined using two or more contact nodes. In each system multi level hierarchical is obtained which contains contact bodies, contact boundaries, contact

www.manaraa.com

segments, contact edges and contact nodes. In the pyramid of hierarchy, the contact system is at the top and the contact nodes are at the bottom.

Contact territory of segments, edges and nodes are defined. Contact territory of a segment is defined using the contact edges, outward unit normal vector and thickness. For an edge, outward normal vectors and thicknesses of the segments it shares are considered. For a node, all the possible unit normal vectors that belong to the edges that share the node, and thickness, are used to define the contact territory. Figure 2-3 shows the contact territories of a node, an edge and a segment.

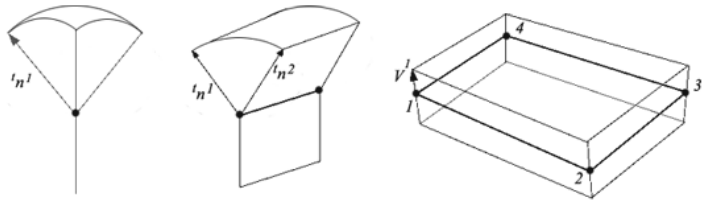


Figure 2-3 Contact territories of a node, an edge and a segment

To find whether or not a given contact node lies outside the contact territory of a contact object, the hierarchy territory is used. Hierarchy territory of a segment is a box with its upper and lower limits are obtained by maximum and minimum values of the contact territories of entities in the hierarchy below it. For a reliable search, hierarchy is expanded by a small amount, and it is known as expanded territory. When territories at the same hierarchical level are tested, the absence of a common territory means that further testing on lower-level hierarchical elements can be trivially rejected. When a common territory is detected, the search proceeds with testing between elements on the lower levels of the hierarchy, which are enclosed or intersected by the common territory. A contact node is said to form a test pair with a contact segment or a contact edge or

www.manaraa.com

another contact node if the contact node lies within the expanded territory of the contact segment or the contact edge or that other contact node. Once a contact pair is formed, local search is performed at each time step, in which the matched nodes are checked for contact with the actual segment and the segments in the closest neighborhood of the matched segment.

The advantage of this algorithm is that it is efficient when there are multiple contact bodies in a contact system which may get in contact with each other only occasionally during the time domain of interest. The drawback of this algorithm is that it cannot generate reasonable contact territory when outward normal vectors of segments are not in the same direction. This algorithm is well suited for simulations with contact in small parts of the mesh and where the contact areas are not overlapping too much (Diekmann, 2000). But in the case of large overlapping areas or nested objects, this algorithm does not perform as good as the position code algorithms which are discussed later in this chapter.

2.1.1.4 Linear Position Code Algorithm

Linear position code algorithm (Oldenburg, 1994) uses an idea similar to that of the bucket sort algorithm and hierarchy algorithm. In this algorithm, each segment is checked for the presence of contact nodes situated within the segment territory. Segment territory is defined as the smallest cubic box that encloses the contact territory of the segment, and contact territory of the segment is defined using outward normal and thickness of the segment. Contact territory and segment territory of a two dimensional segment is shown in Figure 2-4.

www.manaraa.com

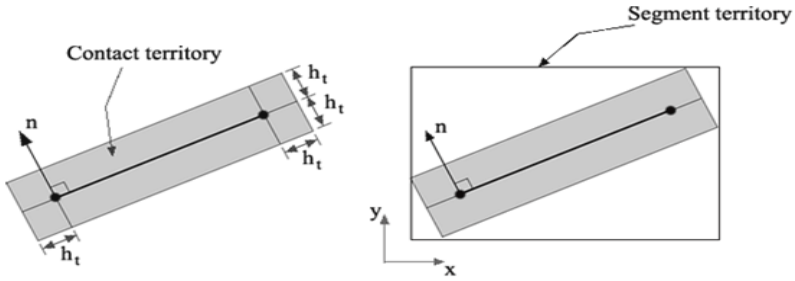


Figure 2-4 Contact- and segment-territory of a segment

Once the contact and segment territories have been constructed, the detection of contact nodes within the segment territories is performed with an algorithm based on sorting and searching in one dimension. The mapping from three dimensions to one dimension is achieved by the definition of a discrete position code. The three-dimensional space containing the contact surfaces of the model is divided into cubic boxes and each box is assigned a number relative to its position in the global co-ordinate system. All contact nodes are assigned a position code corresponding to the position box where they are currently situated. The expression for the position code is given by

$$p_c = B_y B_z b_x + B_x b_y + b_z \tag{2.1}$$

where p_c is the position code and b_x, b_y, b_z are the box numbers in x, y and z dimension respectively, and B_x, B_y and B_z are their maximum numbers respectively. Three-level of hierarchy is defined which consists of contact surfaces, contact segments and contact nodes. If several contact surfaces are defined in the system, the position codes for the contact nodes are stored in a position code vector defined for each surface. The position code vector is processed to find out position code numbers that correspond to position

www.manaraa.com

boxes which are intersected by the segment territory. All contacting nodes in those position boxes are checked for the contact with the segment.

The efficiency of this algorithm is influenced by the choice of two parameters, the expansion of the territories and the size of the position boxes. Larger territory expansion decreases searching frequency with increase in local search procedure. Larger position boxes increases local search with decrease in binary search operations. Hence it depends on the user to choose an optimal combination of the two.

The computation cost of this algorithm is n log n, where n is the number of nodes. This algorithm is input sensitive; i.e., the cost function is only related to the input of the system, which in this case is the number of nodes. Even when the two contact bodies are far from each other, meaning that there are no contact element interactions to be found, the searching algorithm will still require time of order (n log n).

2.1.1.5 Space-filling Curve Algorithm

Space-filling curve algorithm (SPC) (Diekmann, 2000) is a variant of the linear position code algorithm, which is used for global contact search. Instead of row-wise ordering like in the position code algorithm, SPC uses numbering technique that follows a space-filling curve.

The row-wise ordering implies one major disadvantage: Segments which are oriented vertically are more expensive than horizontal ones (Diekmann, 2000). To overcome this drawback, SPC algorithm uses a curve which visits all boxes in some kind of N-like order (Lebesgue curve). This curve is defined in a recursive manner and preserves high locality.

www.manaraa.com

SPC algorithm divides search area into quarters. These boxes are numbered depending on their position with the binary codes 00 (bottom left), 01 (top left), 10 (bottom right) and 11 (top right). The division is continued recursively up to certain pre-defined level. Each child box gets assigned a code which is the code of its parent box concatenated with 00, 01, 10, or 11. This gives a unique key for each box mapping the Lebesgue curve into the searching area. A position code is then assigned to each node. The position code is the box number the node is placed in, and all the nodes are sorted according to it.

Two additional values, number of boxes of the lowest level in x- and y- direction are stored for each node in order to update the position code of the nodes efficiently. Based on the position codes, local search for contact penetration or gap is carried out.

2.1.2 Local Search Algorithms

2.1.2.1 Node-to-segment Algorithm

Node-to-segment algorithm (Hallquist, 1985) is the most common local contact search used in general finite element codes. It uses the principle of preventing slave nodes from penetrating master segments. A segment corresponds to a 4-node shell, a 3-node shell or a face of a brick element. This interface treatment may be outlined as: 1. For each slave node, locate closest master node, and check the master segments that include the master node to identify the segment, if any, containing the slave node. 2. Locate the position of the slave node on the master surface. 3. Determine if slave node has penetrated the master segment.

www.manaraa.com

Determination of master segment containing slave node

Consider a slave node, n_s, sliding on a smooth master surface and assume that search of the master surface has located and stored the master node, n_m, lying closest to n_s, as shown in Figure 2-5. To minimize the operation count, the search for the closest node only includes the closest node from the previous time step, n_m^{old} and its surrounding nodes which are available in the connectivity of the segment that contain n_m^{old} .

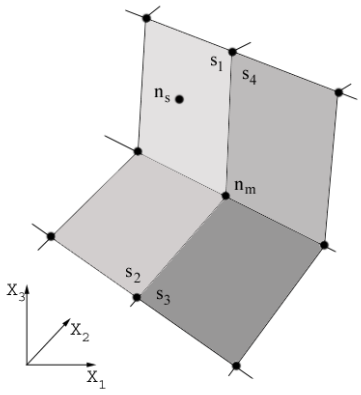


Figure 2-5 Determination of nearest master node

If n_m and n_s do not coincide, n_s can usually be shown to lie in a segment s_i via the following tests:

$$\begin{aligned}(c_i \times s) \bullet (c_i \times c_{i+1}) > 0,\\ (c_i \times s) \bullet (s \times c_{i+1}) > 0.\end{aligned} \tag{2.2}$$

Vectors c_i and c_{i+1} are along edges of s_i and point outward from n_m, and vector s is the projection of the vector beginning at n_m, ending at n_s and denoted by g, on to the plane being examined as shown in Figure 2-6.

www.manaraa.com

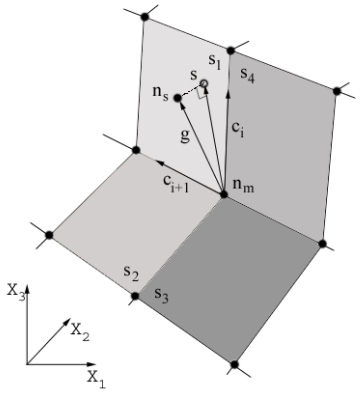


Figure 2-6 Projection of slave node on to nearest master segment

$$s = g - (g \cdot m)m \tag{2.3}$$

$$\text{Where } m = \frac{c_i \times c_{i+1}}{|c_i \times c_{i+1}|}. \tag{2.4}$$

Determination of the contact point

Once the master segment has been located for slave node n_s, the 'contact point' on master segment is identified. Contact point is defined as the point on the master segment which is closest to n_s.

The contact point coordinates (ζ_c, η_c) on s_i is identified by using bilinear parametric representation,

$$\begin{aligned} r &= f_1(\zeta,\eta)i_1 + f_2(\zeta,\eta)i_2 + f_3(\zeta,\eta)i_3, \\ f_i(\zeta,\eta) &= \sum_{j=1}^{4} \phi_j x_i^j, \quad \phi_j(\zeta,\eta) = \frac{1}{4}(1+\zeta\zeta_j)(1+\eta\eta_j) \end{aligned} \tag{2.5}$$

ζ_j, η_j take on their nodal values at $(\pm 1, \pm 1)$, and x_i^j is the nodal coordinate of the j^{th} node in the i^{th} direction.

www.manaraa.com

If t is a position vector drawn to slave node n_s, and r is the vector drawn to the contact point, as shown in Figure 2-6 then following equations are satisfied.

$$\begin{aligned} \frac{\partial r}{\partial \zeta}(\zeta_c, \eta_c) \cdot [t - r(\zeta_c, \eta_c)] = 0, \\ \frac{\partial r}{\partial \eta}(\zeta_c, \eta_c) \cdot [t - r(\zeta_c, \eta_c)] = 0. \end{aligned} \tag{2.6}$$

Once the contact point is determined, each slave node is checked for penetration through its master segment. If the slave node does not penetrate, nothing is done, but if it does, an interface force is applied between the slave node and its contact point. The magnitude of this force is proportional to the amount of penetration. Equal and opposite forces are applied to the slave node and master segment

Magnitude of the penetration is calculated using

$$l = n_i \cdot [t - r(\zeta_c, \eta_c)] \tag{2.7}$$

where n_i is the normal vector of the master segment s_i. Interface force is applied if l is negative.

The above method of contact search is very effective when the contact surfaces are smooth and convex, and the mesh quality is good. When the elements have poor aspect ratios, as shown in Figure 2-7, this method fails to identify the correct nearest master segment. Also when the contact system has elements from different parts that are not connected to each other, this method fails as it checks for new nearest node among the nodes that are connected to the master segment from previous time step. There is no notion of connectivity on the slave side in this contact algorithm which leads to undetected penetration as shown in Figure 2-8.

www.manaraa.com

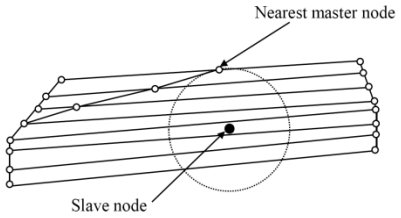


Figure 2-7 Incorrectly identifying nearest master node in a severely deformed mesh

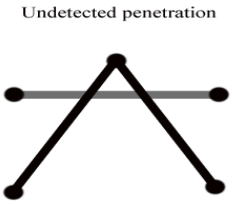


Figure 2-8 Undetected penetration

2.1.2.2 Pinball Algorithm

The concept behind the pinball algorithm (Belytschko, 1989) is to enforce the impenetrability condition and define the interpenetration via a set of spheres, or 'pinballs' which are embedded in the finite elements as shown in Figure 2-9. Each element is embedded in different pinball, regardless of whether it is a shell or solid element. Contact constraint is enforced on the spheres rather than the elements, and thus reducing the time required by the contact algorithm. Checking contact and penetration between two elements becomes computing the distance between two pinballs embedded in the elements.

www.manaraa.com

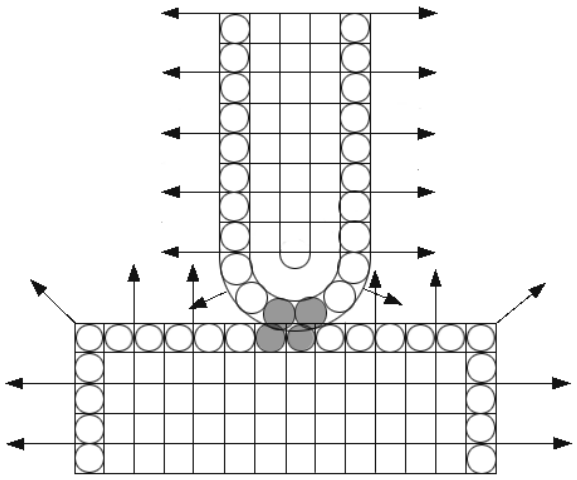


Figure 2-9 Elements embedded in pinballs

In three-dimensional analysis, a sphere or pinball, is embedded in each of the hexahedral elements of the mesh and the radius is determined by setting the volume of the sphere equal to the volume of the element. The center of each sphere is the average of its nodal coordinates. They are given by

$$R = \sqrt[3]{\frac{3V^e}{4\pi}}$$
$$C_i = \frac{1}{8}\sum_{I=1}^{8} x_{Ii}^e \qquad (2.8)$$

where R is the radius of the pinball and V^e is the volume of element e, C_i are the coordinates of the center of the sphere, x_{Ii}^e are the co-ordinates of node I of element e. Radius of the sphere for each element is kept constant throughout the simulation assuming the volume of the element doesn't change.

The detection of the impacting pairs is, computationally, a very simple procedure. The distance between the centers of each slave pinball and each master pinball is

www.manaraa.com

calculated and then compared with the sum of radii of the two elements. Interpenetration is said to have occurred when the distance is less then sum of radii i.e.,

$$d < R_1 + R_2 \tag{2.9}$$

where d is the distance between the centers of elements 1 and 2 and R_1, R_2 are the radii of elements 1 and 2.

In the penalty form of the algorithm, whenever overlap of pinballs is detected, equal and opposite forces proportional to the magnitude of the interpenetration are then applied to the centers of the pinballs. These forces are then transferred to the nodes of the elements in which the pinball embedded.

The advantage of this algorithm is that, it is simple and identical regardless of what type of contact is involved. When combined with a penalty method of treatment, it involves almost no iterative calculations or conditional statements; hence it is amenable to vectorization. Shortcoming of this method is that it cannot be used for problems where sliding and friction are crucial such as in crashworthiness. Inaccurate geometrical representation and non-usage of compressible materials make this algorithm less desirable in complicated finite element analysis. When the shell elements are thin compared to their length, accuracy of this algorithm deteriorates. Also when two thin shell elements are initially in contact, this method fails.

2.1.2.3 Inside-Outside Algorithm

Inside-outside algorithm (Wang, 1997) uses position vector and normal vector of a contact point to check whether there is any penetration of this point on the contact

www.manaraa.com

surface. Once a contact node and contact surface pair has been identified by global search, this algorithm determines the contact point on the contact surface and the distance—gap or penetration—from the node to the contact point.

First the position of the contact node, either inside or outside, with respect to the segment is defined. This can be done using either of the two methods. The first method is based on 'mesh-normal' of the node. Mesh normal of a node is evaluated by averaging all surface normal vectors of the connective elements, as shown in Figure 2-10. When the projected point of the node along its mesh normal direction is located inside the polygon, the node is regarded as inside the segment, else it is considered outside the segment. The second method consists of checking relative position of the node with respect to edges of the segment. A contact segment need is not only a 3-node or a 4-node element; it can be a polygon of any number of sides.

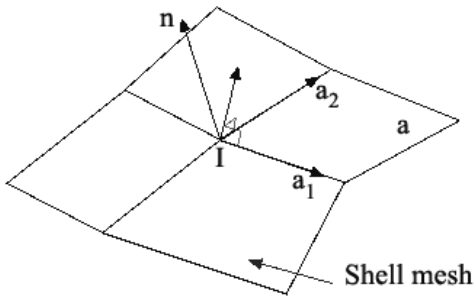


Figure 2-10 Surface mesh normal of node I

For triangular segments, edge inside-outside status detection is performed three times. Similarly, for quadrilateral segments, the detection is checked four times. Any convex polygon can be checked using the same concept. Figure 2-11 shows the inside-outside status check of a node on a 4-node segment.

www.manaraa.com

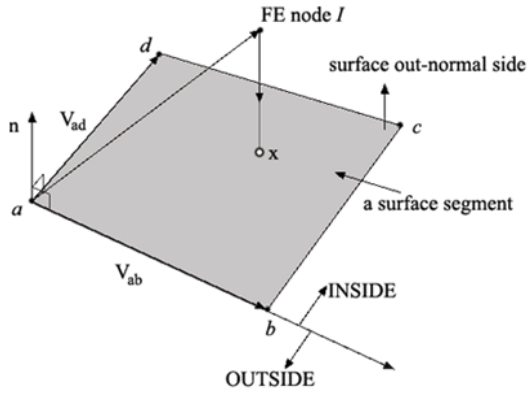


Figure 2-11 Inside-Outside check on a 4-node segment

The inside-outside status of a node is checked by considering all the edges, assuming the segment has counterclockwise connectivity: if all edges are found "inside" or all edges are found "outside" then the node projects inside the segment, else, the node projects outside the segment. In the outside situation, the node is contacting the segment from its opposite direction of the projection vector.

When the projection of node is found to be inside the polygon, the distance between the node and the surface segment is calculated along the mesh normal vector. Using the position vector x of a projection point, the penetration (or gap) is obtained as:

$$g_t = \mathrm{n} \cdot (\mathrm{x} - \mathrm{x}^I) \tag{2.10}$$

where $g_t > 0$ means a gap (no penetration) and $g_t < 0$ means penetration. The case where $g_t = 0$ indicate that the node lies on the surface segment, x^I is the position vector of node and "n" is normal vector of surface.

Important features of this algorithm are that it requires no iteration, and eliminates multiple contact possibilities of the nodes with contact surfaces. A unique contact point is obtained using the mesh normal of a node. Limitation of this algorithm is that surface

www.manaraa.com

normal of the master and slave elements should face each other. Even though polygon of any number of sides can be used, the segment has to be convex. This algorithm may result in error near the intersection edge of two bilinear contact segment surfaces if the mesh of the segment surface is not fine enough.

2.1.2.4 Free-formed-surface Algorithm

Free-formed-surface (FFS) algorithm (Wang, 2001) is developed to improve the accuracy of the contact searching by reducing inaccuracies generated during the finite element meshing. In this algorithm, the three-dimensional contact area is approximated with free-formed-surface patch. This is to make the geometry smooth and accurate for contact searching and contact stress analysis.

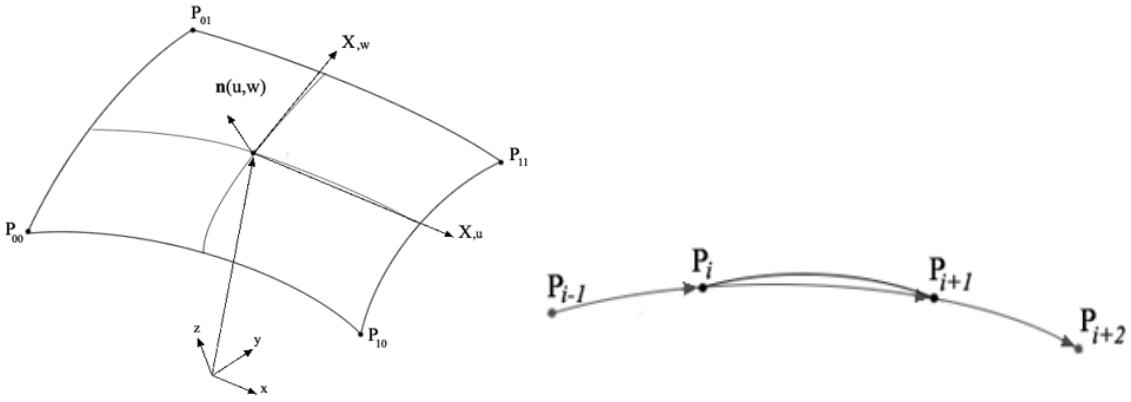


Figure 2-12 Parametric surface patch and surface patch

A parametric surface patches are constructed to describe the continuous body surface using the nodal coordinates, as shown in Figure 2-12. A surface patch is constructed between two nodes using a smooth curve as shown in Figure 2-12. Each curve segment is joined to its neighbors in a continuous fashion. Surface patches are tied together in such a way that continuity with the neighboring patches is assured, at least to the first-order gradient. Considering the case of four nodes of a quadrilateral segment and

www.manaraa.com

other eight nodes surrounding the segment, the parametric equation of the free-formed-surface patch is given by:

$$\mathrm{x}(u,w)=\sum_{i=0}^{3}\sum_{j=0}^{3}c_{ij}u^{i}(1-u)^{3-i}w^{j}(1-w)^{3-j} \tag{2.11}$$

where $\mathrm{x}(u,w)$ is the Cartesian representation of the surface patch, u and w are two independent parameter variants in the range [0,1], and $c_{ij}(i,j=0,1,2,3)$ is the coefficient vector.

Subdivision of the surface patch is done by dividing the patch along its curves. Smaller sub-patches are more closed to planes than the previous sub-patch. The more the subdivision of levels, the higher the accuracy of contact searching will be.

For each contact pair, which consists of a slave node and a master segment formed at global searching level, a free-formed-surface patch is constructed using the master segment nodes and the neighboring nodes. The patch is subdivided into sub-patches depending on the accuracy level needed and the slave node is projected on to these sub-patches in the direction normal to the sub-patches. This projection point is checked if it lies on any of the sub-patch. If it is found on the sub-patch, then penetration (or gap) is calculated, which is given by:

$$g_n=(\mathrm{x}-\mathrm{x}_s)\cdot\mathrm{n} \tag{2.12}$$

where n is the unit normal vector at the contact point x.

Although it is an advantage to have no numerical iteration in this algorithm, it gets more accurate only when the sub-patch subdivision gets smaller. About 5 subdivision levels are

www.manaraa.com

required to obtain acceptable accuracy. This algorithm cannot reach satisfactory results for systems containing sharp corners or highly deformed elements. When elements deform severely compared to their initial configuration, un-accommodated FFS patches may be obtained by which accuracy of contact search decreases rapidly. When sharp corners are necessary, they have to be treated differently, as two or more elements sharing a common edge on the sharp corner as separate ones. All discrete elements are approximated with identical spheres/discs.

2.1.2.5 Splitting Pinball Algorithm

The accuracy of the pinball algorithm deteriorates when it is applied to shell elements that are relatively thin compared to their length. Also pinball algorithm fails when two thin shells are initially in contact. The splitting pinball algorithm (Belytschko, 1993) is a variation of the pinball approach where the developers tried to overcome some of the shortcomings of pinball algorithm.

As in the original pinball method, in the splitting pinball method, a pinball is associated with each element. Unlike in original pinball method where the radius of the pinball is calculated such that volume of pinball is equal to volume of element, in splitting pinball method the radius is always chosen large enough so that it completely envelopes the element. This large pinball is called the parent pinball.

Interpenetration of parent pinballs now indicates the possibility of interpenetration of elements. Whenever overlap of parent pinballs is detected, another level of smaller pinballs is constructed, in which the diameters of the last pinballs in hierarchy are of the order of the thickness of the shell. Figure 2-13 shows few examples of pinball hierarchy

www.manaraa.com

and Figure 2-14 shows an example of interpenetration. If penetration is detected on this level, penalty forces are applied to these pinballs. Subsequently these pinball forces are transferred to the nodes of the associated element.

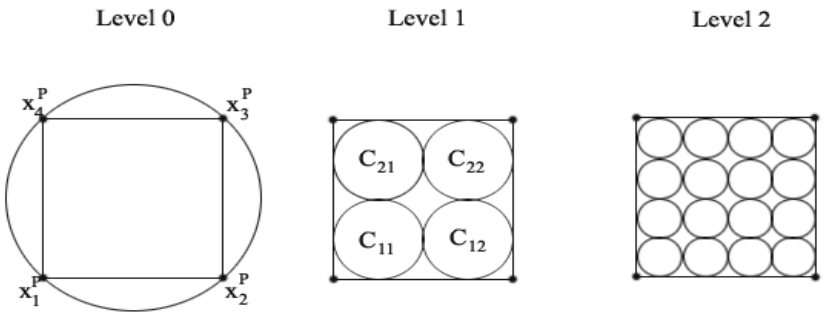


Figure 2-13 Pinball hierarchy of 4-node shell element

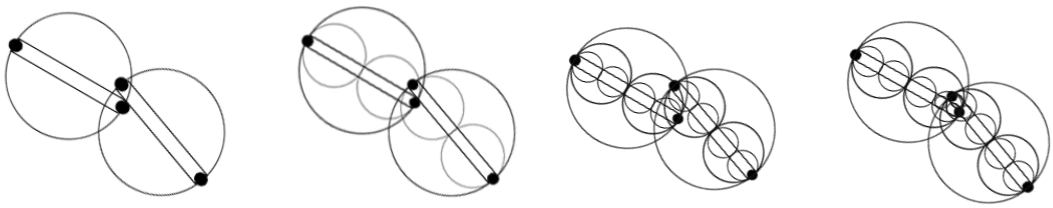


Figure 2-14 Detection of penetration using pinballs

The splitting pinball algorithm possesses the advantage that it is not necessary to distinguish surface-to-surface contact from edge-to-surface contact. It doesn't require any special input from the user. The disadvantage of this algorithm is that the contact surfaces are represented only by a coarse approximation. When higher order elements are used and sliding contact and friction are very important these approximations induce error.

2.1.2.6 Direct Localization Algorithm

Direct localization algorithm (Petkevicius, 2003) uses the concept of splitting pinball algorithm but avoiding the splitting process which decreases the efficiency.

www.manaraa.com

Instead of checking for penetration from the parent level pinball and keep splitting pinballs until the diameter becomes the thickness of shell, this algorithm checks the nodal position of one quadrilateral to that of the other quadrilateral element.

In direct localization algorithm, slave and master elements called projectile and target elements, several pinballs are generated in the quadrilaterals of the projectile. The higher the number of pinballs, the higher the precision is. At the lowest level, only one pinball is generated which is located in the center of a quadrilateral finite element. When one pinball per element does not satisfy the precision requirements, then the number of pinballs is increased as shown in Figure 2-15.

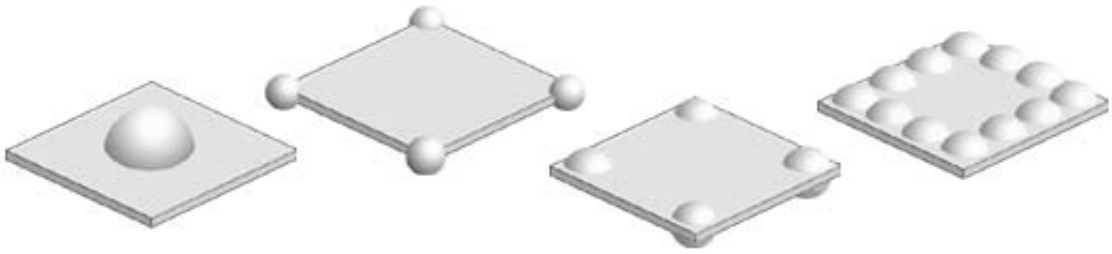


Figure 2-15 Different splittings of quadrilateral element

The pinballs are projected onto the target element, and the distance between the center of the pinball and its projection is calculated using equation (2.13).

$$\left.\begin{aligned} u\left(P_{2x}-P_{1x}\right)+v\left(P_{4x}-P_{1x}\right)+n_x D &= Q_x-P_{1x} \\ u\left(P_{2y}-P_{1y}\right)+v\left(P_{4y}-P_{1y}\right)+n_y D &= Q_y-P_{1y} \\ u\left(P_{2z}-P_{1z}\right)+v\left(P_{4z}-P_{1z}\right)+n_z D &= Q_z-P_{1z} \end{aligned}\right\} \tag{2.13}$$

where D is the distance between the pinball center and the middle surface of the target element; u and v are local coordinates on the target element; n_x, n_y, n_z are projections of

www.manaraa.com

the vector connecting the pinball center and the target element; $P_{1x}, P_{1y}, P_{1z}, \ldots\ldots P_{4x}, P_{4y}, P_{4z}$ are nodal coordinates of the target elements; and Q_x, Q_y, Q_z are coordinates of the center of the pinball.

If the projectile pinball projects onto the target element and the distance D is less than half of the sum of the thickness of the projectile and target elements, penalty forces are computed and applied to the nodes of the projectile and target.

Although the direct localization approach has the advantage of a significant decrease in computational time compared to the splitting pinball algorithm, it has the disadvantage, similar to other pinball algorithms, that it doesn't represent the actual geometry for contact search. This becomes critical if sliding contact and friction are important in the analysis. User has to do several simulations to identify the number of pinballs required to get a reasonable degree of accuracy in contact search.

2.1.3 Contact Mechanics

Once the contact search is complete and the depths of penetrations are computed, the next step is to remove the penetrations by applying appropriate forces. The forces should be accurate in magnitude, direction, and location. The process is known as contact mechanics and the forces are called contact constraints.

For the most part, numerical methods used for implementing contact mechanics in transient finite element analysis may be broadly classified into either Lagrange multiplier or penalty function methods. However, for high-velocity impact problems in which transfer of momentum, rather than structural deformation is the dominant effect at the

www.manaraa.com

contact interface, a different methods based on momentum conservation have been developed. Lagrange multiplier methods are alternatively referred to as mixed or hybrid variational methods by some authors and penalty methods are commonly referred to as 'contact', 'gap', or 'joint' element methods. For transient analyses by explicit integration, penalty methods have received the most attention in the literature and in commercial finite element programs.

2.1.3.1 Lagrange Multiplier Method

A brief review of the classical Lagrange multiplier method is presented below (Fortin 1983).

The finite element semi-discretized equation of motion is expressed in general form as:

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{F}\left(\mathbf{U}, \dot{\mathbf{U}}\right) = \mathbf{R} \tag{2.14}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{U}$ is the vector of displacement degrees of freedom, $\dot{\mathbf{U}}$ is the velocity, $\ddot{\mathbf{U}}$ is acceleration, $\mathbf{F}$ is the internal force vector, and $\mathbf{R}$ is the external force vector. In addition to the usual prescribed boundary conditions, it is assumed that the solution of equation (2.14) is also subject to surface contact displacement constraints. These constraints may be expressed as:

$$\mathbf{G}\{\mathbf{U}+\mathbf{X}\} = 0 \tag{2.15}$$

where $\mathbf{X}$ is the material coordinate vector, the sum of $\mathbf{U}$ and $\mathbf{X}$ is the spatial coordinate vector and $\mathbf{G}$ is a surface contact displacement constraint matrix. The components of $\mathbf{G}$ are typically unknown in the beginning and generally change as displacement and deformation occur. Motion of slave and master nodes is tracked and as contact occurs, displacement constraint components are introduced in $\mathbf{G}$. During contact, the

www.manaraa.com

components of $\mathbf{G}$ may change with time as required to ensure that the associated contact force reactions satisfy contact force conditions.

The Lagrange multipliers are introduced into the equation of motion to give:

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{F}\left(\mathbf{U}, \dot{\mathbf{U}}\right) + \mathbf{G}^T \lambda = \mathbf{R} \tag{2.16}$$

where the components of the Lagrange multiplier vector λ are the surface contact forces. The Lagrange multiplier method proceeds by treating λ as unknown and solving equations (2.15) and (2.16) simultaneously.

To summarize, Lagrange multiplier method, which solves for the unknown Lagrange multipliers, exactly enforces the kinematic impenetrability constraints on displacements and, when necessary, applies special impact and release conditions to correctly determine velocities, accelerations, and tractions over the contact interface.

Since for each constraint condition a Lagrange parameter is introduced and it appears in the list of unknowns, the dimension of the resulting system of equations will increase. In addition, the associated tangent matrix is indefinite and has zero diagonal entries that pose some difficulties in the solution step. In this approach, the system of equations cannot be solved in an explicit way. The Lagrange multiplier matrix has to be inverted at each cycle of computation. In the case of auto-contact, the number of points in the contact can become significant and this formulation then becomes quite expensive.

Based on classical Lagrange parameter procedure, few other versions of contact procedures are developed such as augmented Lagrangian (Fortin, 1983; Malone, 1994) and perturbed Lagranian procedure (Simo, 1985).

www.manaraa.com

2.1.3.2 Penalty Methods

Unlike the Lagrange multiplier methods where addition of Lagrange parameters increases the dimensions of the solution, the penalty methods enable one to transform the constrained problem into an unconstrained one without introducing additional variables. The constraint condition is now satisfied only approximately for finite values of the penalty parameter.

The penalty method assumes from the outset that the impenetrability condition will be violated. This results in solutions that satisfy the contact conditions only approximately. Nodal contact forces normal to the contact surface are essentially computed by multiplying the amount of penetration by an arbitrarily defined penalty parameter. The accuracy of the solution depends strongly on the penalty parameter. The choice of penalty parameters for both the normal and tangential forces is primarily determined from numerical experience. In explicit time integration schemes, large penalty parameters may cause numerical instability. Therefore, penalty parameters are typically chosen conservatively at the expense of allowing greater amounts of penetration. However, in some analyses, the penalty number is defined by the problem itself using a specific formulation.

2.2 Need for a new contact algorithm

Although several contact-impact search methods are available, none of them are suitable for all situations. Each of them has its own advantages and disadvantages, as explained in the previous sections, and users have to choose among them depending on his/her need. This can lead to the need of several simulations before Factors to consider while choosing a contact-impact algorithm are: whether the analysis is implicit or

www.manaraa.com

explicit, Eulerian or Lagrangian, whether the contact surfaces are convex or concave, type of elements used and whether the contact search is between surface-to-surface, edge-to-surface or edge-to-edge.

Although significant progress has been achieved in the development of contact algorithms, there is still a need for improvement in both the efficiency and the reliability of the algorithms. The primary concerns for contact searching are computational cost, accuracy and robustness. Conventional algorithms use the principle of preventing 'slave' nodes from penetrating 'master' segments. There is no notion of connectivity on the slave side in a contact algorithm.

In summary, the drawbacks of the currently available algorithms:

- The determination of the interpenetration requires iteration and consequently does not vectorize well;
- Because only interpenetration between finite elements nodes and elements are checked, the surface-to-surface and edge-to-surface contacts shown in Figure 2-16 cannot be detected;

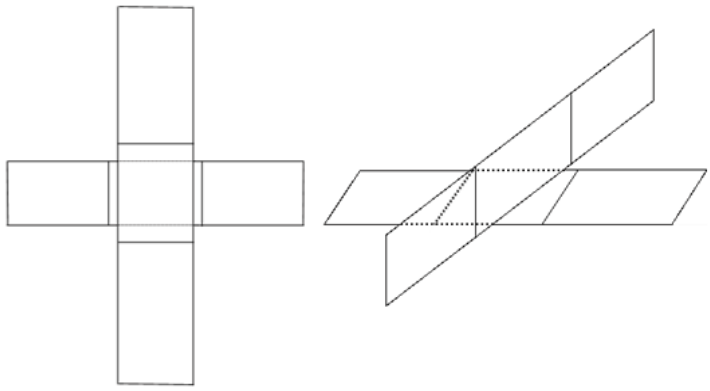


Figure 2-16 Surface-to-surface and edge-to-surface failure

www.manaraa.com

- The algorithm is ambiguous in situations such as shown in Figure 2-17. A slave node which do not see the edges and has come to the other side of master surface will be interpreted as a spurious penetration. To avoid such ambiguities, master segments have to be carefully defined, i.e. a prior knowledge of the master elements which are likely to be penetrated by slave nodes is required;

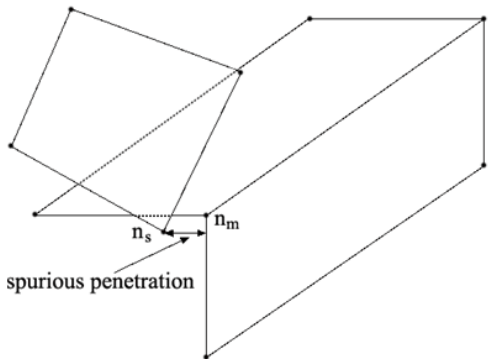


Figure 2-17 Ambiguous situation during nodes-to-surface contact

- Multiple contacts between a slave node and more than one master segment such as shown in Figure 2-18 can cause difficulties.

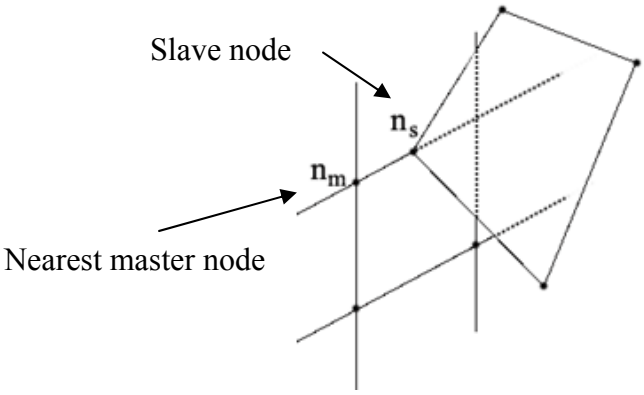


Figure 2-18 Multiple contacts

- Concept of using pinballs for checking interpenetration decreases accuracy of the geometry and hence diminishing the robustness of the contact search. This causes

www.manaraa.com

serious problems especially if significant sliding between the segments occurs and also when friction between components is important.

- In complex simulations, the users might need to do several simulations before correctly identifying all contacts. In case of pinball algorithms, they might need more simulations to identify the level or number of pinballs to obtain the required accuracy.

2.3 Contact Validation Tests

To address some of the problems mentioned in the previous section, a new contact algorithm is developed and implemented in DYNA3D public code. Process of development and implementation is explained in Chapters 4 and 5. Before the new contact algorithm can be used with confidence, it should be assessed for accuracy and stability. Two commonly used tests to assess the robustness of contact algorithms are: the Hertz contact test and the contact patch test. The next sections present a summary of these two tests.

2.3.1 Hertz Contact Test

Very few problems involving contact can be solved analytically. Contact between the two bodies occurs over many small areas, each of which constitutes a single asperity contact (Hertz, 1882). Well known theoretical solution on single asperity or single point was developed in the late nineteenth century by Hertz (Adams, 2000) which has become to known as Hertz contact problem. Hertz investigated the elastic contact of two spheres and derived the pressure distribution in the contact area as well as the approach of the spheres under compression. The assumptions of this problem are: (1) the contact area is elliptical; (2) each body is approximated by an elastic half-space loaded over the plane

www.manaraa.com

elliptical contact area; (3) the dimensions of the contact area must be small compared to the dimensions of each body and the radii of curvature of surfaces; (4) the strains are sufficiently small for linear elasticity to be valid and (5) the contact is frictionless, so that only a normal pressure is transmitted.

Figure 2-19 shows partial view of two bodies, 1 and 2, in contact. If there is no pressure between the bodies, the contact is at one point '*O*'. When load *P* is applied between the bodies, the contact area becomes elliptical.

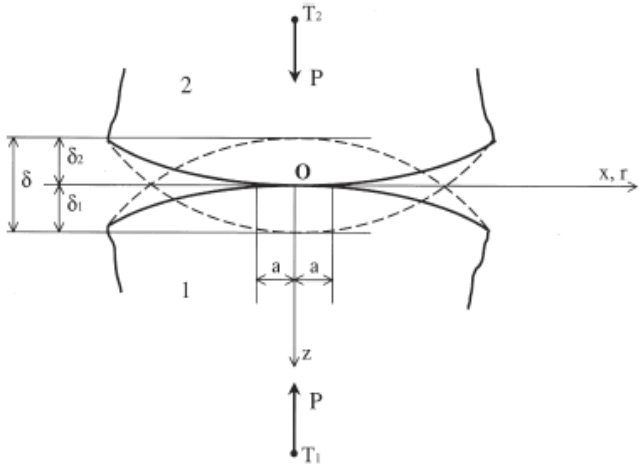


Figure 2-19 Hertz contact of two nonconforming elastic bodies

For the case of solids of revolution, the contact area is circular. The interference 'δ', contact radius '*a*' and maximum contact pressure 'p_0' are given by

$$a = \left(\frac{3PR}{4E^*}\right)^{1/3}, \qquad \text{where} \quad \frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}, \qquad \frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

$$\delta = \left(\frac{9P^2}{16RE^{*2}}\right)^{1/3}, \qquad p_0 = \left(\frac{6PE^{*2}}{\pi^3 R^2}\right)^{1/3}$$

www.manaraa.com

In the above equations, p_0 is the maximum contact pressure (which occurs at $r = 0$), E^* is the composite Young's modulus, E_1, E_2 and ν_1, ν_2 are the Young's modulii and Poisson's ratios for the lower and upper body respectively, P is the normal load, R is the composite radius of curvature and R_1, R_2 are the radii of curvature of the lower and upper bodies respectively.

Analogous expressions may be written for the contact of two cylindrical bodies whose long axes are parallel to the *y*-axis. The results for half-width of the contact strip and the maximum contact pressure are

$a = \left(\frac{4P'R}{\pi E^*}\right)^{1/2}$, $p_0 = \left(\frac{P'E^*}{\pi R}\right)^{1/2}$ where P' is the applied load per unit length of y-direction.

For a sphere on a flat plate, $R_2 \to \infty$, so $\frac{1}{R} = \frac{1}{R_1}$

And for a sphere in a spherical cup, $\frac{1}{R} = \frac{1}{R_2} - \frac{1}{R_1}$

Figure 2-20 shows the configuration of sphere on a flat plate and sphere on a spherical cup.

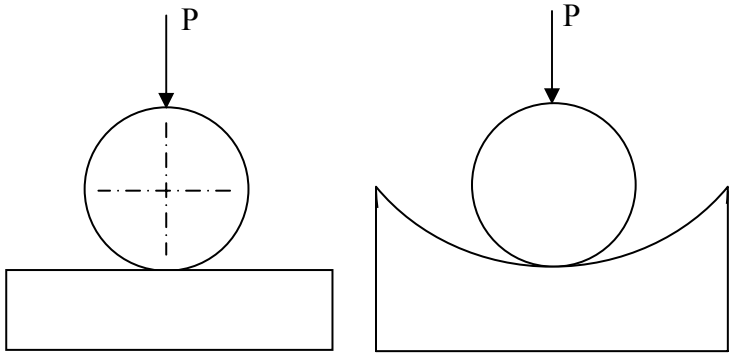


Figure 2-20 Sphere on a flat plate and sphere in a spherical cup

www.manaraa.com

2.3.2 Contact Patch Test

When it comes to an element, it is known that every 'element', whether it is from a meshed approach or a meshless one, must possess certain properties to guarantee its validity, i.e. it must be consistent and convergent (Timoshenko, 1970). Zero stress condition during rigid body movement and constant stress condition when subjected to a linear displacement field are the two conditions that are usually used to verify. For an 'element' formed in slide-line (or mesh matching) procedures, similar requirements on its quality must hold, particularly, the preservation of uniform displacements (stress distribution) across the interfaces under uniform loading. This in the contact context is often called contact patch test (El-Abbbasi, 2001; Taylor, 1991; Sacco, 1995). If a contact formulation fails this test, fictitious localized stresses will occur across the contact surfaces. Sometimes, they become substantially large so as to undermine the prediction of stress distributions on the interfaces.

Different researchers use different contact patch test problems to verify consistency and stability of the contact algorithms.Figure 2-21 shows a simple contact patch test problem used in this research. Two rectangular plates that are on the same plane were made to contact each other on their edges. Two edges of the bottom plate and one edge of top plate are constrained as shown in the Figure 2-21. A uniform load is applied on the top edge of the top plate and stresses and displacements along the contact surfaces are observed.

www.manaraa.com

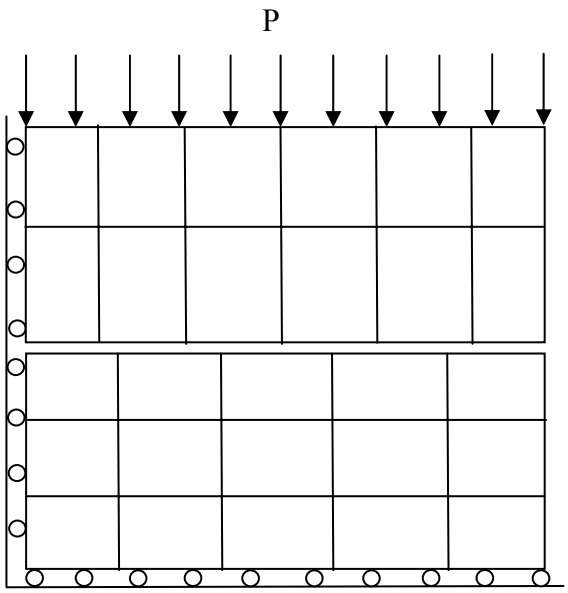


Figure 2-21 Simple contact patch test problem

www.manaraa.com

3. DYNA3D OVERVIEW

DYNA3D is a part of set of public codes developed at the Lawrence Livermore National Laboratory. It is a general-purpose finite element code based on explicit time integration for analyzing the transient response of three dimensional, non-linear, dynamic, large displacement problems. The first version of DYNA3D was released in 1976 and since then it has come a long way in its capability and user convenience. Even though initial focus was in military applications, DYNA3D has been used in various fields including automotive crashworthiness and roadside hardware structure analysis.

DYNA3D has numerous features that allow for the analysis of several nonlinear dynamic problems. It has several element formulations that include one-dimensional truss and beam elements, two-dimensional quadrilateral and triangular shell elements, two-dimensional delamination and cohesive interface elements, and three-dimensional continuum elements. Various material models are available to represent, a wide range of material behavior including elasticity and plasticity, composites, thermal effects and rate dependence. The most advantageous capability of DYNA3D over other finite element codes is its contact algorithm. It has sophisticated contact interface capability including frictional sliding and single surface contact.

In this chapter, a brief overview of DYNA3D features and the theory behind these features are presented. Only the features that are relevant to this study are covered. These include the explicit finite element method, the time step, and the contact interface.

www.manaraa.com

3.1 Explicit Finite Element Method

DYNA3D uses a displacement-based, Lagrangian, central-difference finite element formulation to solve for the dynamic response of nonlinear structural problems. The formulation makes use of Cauchy's first law of motion and principle of virtual work to determine the potential energy equation. The potential energy equation is then discretized in space through the finite element mesh and shape functions. It is then discretized in time through the explicit central difference method to derive the dynamic equations of motion. In this section, the DYNA3D explicit finite element formulations are derived and the theories behind these formulations are presented.

3.1.1 Principle of virtual work

In this section, first, a general three dimensional problem to be solved is stated. Then, principle of virtual displacements, which is used as basis of finite element solutions, is discussed and the finite element equations are derived.

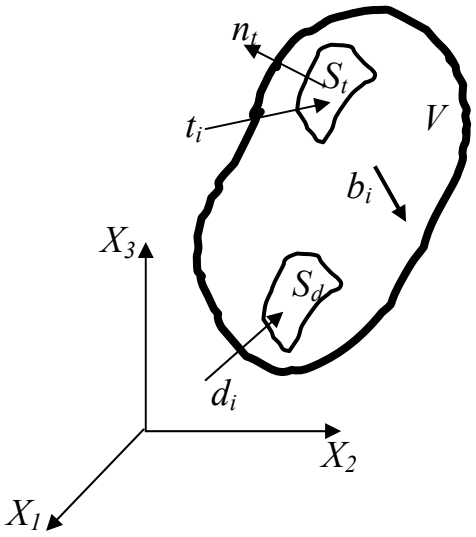


Figure 3-1 A general three-dimensional body

www.manaraa.com

A three-dimensional body, as shown in Figure 3-1, is located in a fixed (Lagrangian) space. The body is subjected to traction forces t_i *(t)* (forces per unit area) over a portion of its outer surface S_t, prescribed displacements d_i *(t)* over the surface S_d, and external body forces b_i *(t)* (forces per unit volume) over its entire volume V.

The solution to this problem must satisfy the following differential equations:

$$\sigma_{ij,j} + \rho b_i - \rho \ddot{x}_i = 0 \qquad \text{over the volume of the body } V \qquad (3.1)$$

$$\sigma_{ij} n_j = t_i \qquad \text{over the traction surface } S_t \qquad (3.2)$$

$$x_i = d_i \qquad \text{over the displacement boundary } S_d \qquad (3.3)$$

where σ_{ij} denotes Cauchy's stress tensor, ρ is the material current density, and n_i is the outward normal unit vector to the traction surface S_t.

Equation (3.1) is Cauchy's first law of motion, which ensures that the linear momentum of the system is conserved. Equation (3.2) is the traction boundary condition which must be satisfied at each particle on the surface S_t. Equation (3.3) is the displacement boundary condition equation which must be satisfied over the surface S_d. These equations are said to state the problem in the strong form, meaning the differential equations have to be satisfied at every point in the body or in the surface. While solving the problem numerically using the finite element method, the problem is defined in the weak form. In the weak form, the conditions do not have to be satisfied at every point in the body, but only on an average or integral sense.

The weak form equation to the general problem in Figure 3.1 is derived from the principle of virtual work. An arbitrary virtual displacement δx_i, that satisfies the

www.manaraa.com

displacement boundary condition in S_d, is introduced. Multiplying equation (3.1) by the virtual displacement and integrating over the volume of the body leads to,

$$\int_v (\sigma_{ij,j} + \rho b_i - \rho \ddot{x}_i)\delta x_i \, dV = 0 \tag{3.4}$$

For equation (3.4) to be valid for any arbitrary virtual displacement, the term between the brackets should be equal to zero, which is equivalent to equation (3.1).

Using one of the mathematical properties of differentiation,

$$(\sigma_{ij}\delta x_i)_{,j} = \sigma_{ij,j}\delta x_i + \sigma_{ij}\delta x_{i,j} \tag{3.5}$$

and substituting for the first term in equation (3.4) leads to,

$$\int_V \left((\sigma_{ij}\delta x_i)_{,j} - \sigma_{ij}\delta x_{i,j} + \rho \, b_i \, \delta x_i - \rho \, \ddot{x}_i \, \delta x_i \right) dV = 0 \tag{3.6}$$

From the divergence theorem, the first term of the equation (3.6) can be expressed as,

$$\int_V (\sigma_{ij} \, \delta x_i)_{,j} \, dV = \int_{S_t} (\sigma_{ij} \, \delta x_i) \, n_i \, dS \tag{3.7}$$

Using equation (3.2), equation (3.7) can be written as,

$$\int_V (\sigma_{ij} \, \delta x_i)_{,j} \, dV = \int_{S_t} t_i \, \delta x_i \, dS \tag{3.8}$$

From the symmetry of the stress tensor, the second term in equation (3.4) can be expressed as,

$$\int_V \sigma_{ij} \, \delta x_{i,j} \, dV = \int_V \frac{1}{2} (\sigma_{ij} \, \delta x_{i,j} + \sigma_{ji} \, \delta x_{j,i}) dV$$

$$= \int_V \sigma_{ij} \, \delta \varepsilon_{ij} \, dV \tag{3.9}$$

www.manaraa.com

where $\delta\varepsilon_{ij}$ is the virtual strain tensor attributed to the virtual displacement δx_i. Substituting equations (3.8) and (3.9) into equation (3.6), we get

$$-\int_V \rho\, \ddot{x}_i\, \delta x_i\, dV - \int_V \sigma_{ij}\, \delta\varepsilon_{ij}\, dV + \int_V \rho\, b_i\, \delta x_i\, dV + \int_{s_t} t_i\, \delta x_i\, dS = 0 \qquad (3.10)$$

Equation (3.10) is a statement of the principle of virtual work for the general three dimensional problem defined in Figure 3-1.

The next step in deriving the finite element equation is spatial discretization. This is achieved by subdividing the complex geometry of the body into small simpler shapes called elements. The elements are interconnected at the corners through nodal points. To establish continuity of the displacement field throughout the finite element mesh, interpolation function, also known as shape functions, are introduced. These shape functions establish a relationship between the displacements at inner points in the elements and the displacements at the nodal points. Using shape functions, the displacement at any point can be expressed as,

$$\delta x_i = \sum_{\alpha=1}^{n} N_\alpha\, \delta x_{\alpha i} \qquad (3.11)$$

where δx_i are the displacements at any point inside the element, n is the number of nodes in the element, N_α is the shape function at node α, and $\delta x_{\alpha i}$ are the displacements at node α. Similar expressions can also be written for the coordinates, velocities and acceleration of a point inside the element.

The finite element equations are derived by discretizing the virtual work equation (3.10) in space. This is achieved by first writing an approximation of the virtual work

www.manaraa.com

equation as the sum of the potential energy at each element in the system. Equation (3.10) can be written as,

$$\sum_{m=1}^{M}\left\{\int_{V_m}\rho\,\ddot{x}_i\,\delta x_i\,dV_m+\int_{V_m}\sigma_{ij}\,\delta x_{i,j}dV_m-\int_{V_m}\rho\,b_i\,\delta x_i\,dV_m-\int_{S_t}t_i\,\delta x_i\,dS_m\right\}=0 \tag{3.13}$$

where M is the total number of elements in the system and Vm is the volume of the elements. Replacing δx_i and $\ddot{x}_i$ with the equations using shape functions, we get,

$$\sum_{m=1}^{M}\left\{\int_{V_m}\rho\left(N_\beta\ddot{x}_{\beta i}\right)\left(N_\alpha\delta x_{\alpha i}\right)dV_m+\int_{V_m}\sigma_{ij}\left(N_{\alpha,j}\delta x_{\alpha i}\right)dV_m-\int_{V_m}\rho\,b_i\left(N_\alpha\delta x_{\alpha i}\right)dV_m-\int_{S_t}t_i\left(N_\alpha\delta x_{\alpha i}\right)dS_m\right\}=0 \tag{3.14}$$

where $\delta x_{\alpha i}$ and $\ddot{x}_{\alpha i}$ are the virtual displacement and the accelerations at the nodal points respectively. Equation (3.14) can be simplified and rewritten as,

$$\sum_{m=1}^{M}\left\{\int_{V_m}\rho\,N_\alpha\,N_\beta\,dV_m\right\}\ddot{x}_{\beta i}=\sum_{m=1}^{M}\int_{V_m}N_\alpha\,\rho\,b_i\,dV_m+\sum_{m=1}^{M}\int_{S_t}N_\alpha\,t_i\,dS_m-\sum_{m=1}^{M}\int_{V_m}N_{\alpha,j}\,\sigma_{ij}\,dV_m \tag{3.15}$$

In matrix form, equation (3.15) reduces to,

$$[\mathrm{M}]\{\ddot{\mathrm{x}}\}=\{\mathrm{F}\} \tag{3.16}$$

where [M] is the mass matrix, $\{\ddot{\mathrm{x}}\}$ is the acceleration vector and {F} is the vector sum of all internal and external forces. Equation (3.16) is the finite element equation that needs to be solved in time.

www.manaraa.com

3.1.2 Time discretization

If the applied forces vary with time, the equilibrium equation (3.16) is a statement of equilibrium for any specific point in time. Hence the problem discretized in time domain and the finite element equations are satisfied at discrete points in time rather than at all points in time within the interval of the solution. The time interval between two successive points in time, t_n and t_{n+1} is known as the time step $\Delta t_n \left(\Delta t_n = t_{n+1} - t_n\right)$. The time step has significant influence on the accuracy on the explicit finite element solution.

Several direct integration methods have been developed and are classified into implicit and explicit methods. The average acceleration (trapezoidal rule), the Fox-Goodwin (royal road), and the linear acceleration are examples of implicit method and central difference method is example of explicit.

DYNA3D uses central difference method for discretizing the finite element equation in time. In this method, the velocity vector is lagged by half the time step. In other words, the displacement and acceleration vectors are computed at times $t_1,...,t_n,t_{n+1},...,t_f$ (where t_f is the final problem time) and the velocity vector is computed at times $t_{1/2},...,t_{n-1/2},t_{n+1/2},..t_{f-1/2}$. To advance to next time step, following equations are used:

$$v^{n+1/2} = v^{n-1/2} + a^n \Delta t^n \tag{3.17}$$

$$u^{n+1} = u^n + v^{n+1/2} \Delta t^{n+1/2} \tag{3.18}$$

$$x^{n+1} = x^0 + u^{n+1} \tag{3.19}$$

$$\Delta t^{n+1/2} = \frac{\left(\Delta t^n + \Delta t^{n+1}\right)}{2} \tag{3.20}$$

www.manaraa.com

3.2 Time Step Criteria

The choice of time step is critical in explicit finite element analysis. A large time step can make the solution unstable while a small time step can make the computation cost expensive. To ensure stability of time integration, the global critical time step has to be small enough such that the stress wave does not travel across more than one element at each time increment cycle. Therefore, it is important to compute critical time step accurately. It can be achieved using Courant criteria

$$\Delta t_e = \frac{L_s}{c} \tag{3.21}$$

where Δt_e is the critical time step of the element, L_s is the characteristic length of the element and c is the sound speed.

For one dimensional elements, L_s is the length of the element and c is given by

$$c = \sqrt{\frac{E}{\rho}} \tag{3.22}$$

where E and ρ are the Young's modulus and density of the material respectively. For two-dimensional elements c is given by

$$c = \sqrt{\frac{E}{\rho\left(1-\nu^2\right)}} \tag{3.23}$$

where ν is Poisson's ratio. The global time step is the minimum value over all elements.

$$\Delta t^{n+1} = \alpha * \min\left(\Delta t_1, \Delta t_2, ..., \Delta t_N\right) \tag{3.24}$$

www.manaraa.com

where α is a scale factor typically set some value smaller than 1 and N is the number of elements.

3.3 Contact Interface Equations

Contact-impact algorithms in general purpose FE codes can treat interaction of many bodies. Although the two bodies are interchangeable with respect to their mechanics, in some equations and algorithms the bodies are distinguished as master and slave. Nodes lying on those surfaces are referred to as master and slave nodes respectively.

Consider two-body problem shown in Figure 3-2. Let Ω^A and Ω^B be current configurations of the two bodies and S^A and S^B be their surface boundaries respectively.

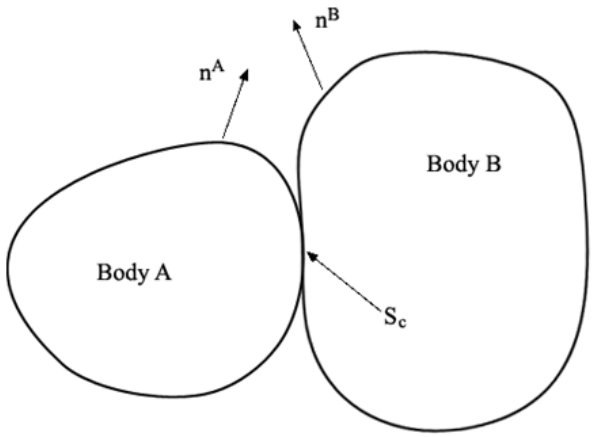


Figure 3-2 Notations of two bodies in contact

The common contact surface (interface) S^C between two bodies is defined by

$$S^c = S^A \cap S^B \tag{3.25}$$

www.manaraa.com

The contact interface is a function of time, and its determination is an important part of the solution of the contact-impact problem. Designating body A as master and body B as slave, the normal for master surface at any point is given by

$$\mathbf{n}^A = \hat{\mathbf{e}}_1{}^A \times \hat{\mathbf{e}}_2{}^B \tag{3.26}$$

where $\hat{\mathbf{e}}_1{}^A$ and $\hat{\mathbf{e}}_2{}^A$ are unit vectors in local coordinate system tangent to the surface.

On the contact surface

$$\mathbf{n}^A = -\mathbf{n}^B \tag{3.27}$$

i.e. the normals of the two bodies are in opposite directions. The velocity fields are expressed in terms of local components by

$$\begin{aligned} \mathbf{v}^A &= v_N{}^A \mathbf{n}^A + \mathbf{v}_T{}^A \\ \mathbf{v}^B &= v_N{}^B \mathbf{n}^B + \mathbf{v}_T{}^B \end{aligned} \tag{3.28}$$

where $\mathbf{v}_T$ are tangential velocities. The range of tangential velocities is 2 in three-dimensional problems and single tangential vector in case of two-dimensional problem. The normal velocities are given by:

$$\begin{aligned} v_N{}^A &= \mathbf{v}^A \cdot \mathbf{n}^A \\ v_N{}^B &= \mathbf{v}^B \cdot \mathbf{n}^B \end{aligned} \tag{3.29}$$

The contact adds the following conditions to the standard governing field equations: the bodies cannot interpenetrate and the tractions must satisfy the momentum conservation on the interface. Furthermore, the normal traction across the contact interface cannot be tensile.

www.manaraa.com

3.3.1 Impenetrability condition

In a multi-body problem, the bodies must observe the impenetrability condition. The impenetrability condition for a pair of bodies can be stated as

$$\Omega^A \cap \Omega^B = 0 \tag{3.30}$$

that is, the intersection of the two bodies is the null set. In other words, the two bodies are not allowed to overlap, which can also be viewed as a compatibility condition. The impenetrability condition is highly nonlinear for large displacement problems, and in general cannot be expressed as an algebraic or differential equation in terms of the displacements. The difficulty arises because in an arbitrary motion it is impossible to anticipate which points of the two bodies will contact.

Because it is not feasible to express impenetrability condition in terms of displacements, it is convenient to express the equations in rate form or incremental form in each stage of the process. The rate form of the impenetrability condition is applied to those portions of bodies which are already in contact, i.e. to those points which are on the contact surface S^C. It can be written as

$$\gamma_N = \mathbf{v}^A \cdot \mathbf{n}^A + \mathbf{v}^B \cdot \mathbf{n}^B \equiv v_N{}^A - v_N{}^B \leq 0 \text{ on } S^C \tag{3.31}$$

The impenetrability condition restricts the interpenetration rate for any points on the contact surface to be negative, i.e. when the two bodies are in contact they must either remain in contact or they must separate.

www.manaraa.com

3.3.2 Traction conditions

The tractions must observe the balance of momentum across the contact interface. Since the interface has no mass, this requires that the sum of tractions on the two bodies vanishes:

$$\mathbf{t}^A + \mathbf{t}^B = 0 \tag{3.32}$$

By Cauchy's law, tractions on the surfaces of bodies are defined as,

$$\begin{aligned} \mathbf{t}^A &= \sigma^A \cdot \mathbf{n}^A \\ \mathbf{t}^B &= \sigma^B \cdot \mathbf{n}^B \end{aligned} \tag{3.33}$$

where σ is the Cauchy stress tensor. The normal tractions are defined by

$$\begin{aligned} t_N{}^A &= \mathbf{t}^A \cdot \mathbf{n}^A \\ t_N{}^B &= \mathbf{t}^B \cdot \mathbf{n}^A \end{aligned} \tag{3.34}$$

Normal component of momentum balance is obtained by taking dot product of equation (3.32) with the normal vector $\mathbf{n}^A$, which gives

$$t_N{}^A + t_N{}^B = 0 \tag{3.35}$$

Since adhesion between the contact surfaces in the normal direction is not considered, the normal tractions cannot be tensile. This condition can be stated as

$$t_N = t_N{}^A = -t_N{}^B \leq 0 \tag{3.36}$$

The tangential tractions are defined by

$$\begin{aligned} \mathbf{t}_T{}^A &= \mathbf{t}^A - t_N{}^A \mathbf{n}^A \\ \mathbf{t}_T{}^B &= \mathbf{t}^B - t_N{}^B \mathbf{n}^B \end{aligned} \tag{3.37}$$

www.manaraa.com

so the tangential tractions are the total tractions projected on the master contact surface. Momentum balance requires that

$$\mathbf{t}_T{}^A + \mathbf{t}_T{}^B = 0 \tag{3.38}$$

3.4 DYNA3D Source Code

The source code for DYNA3D is written in FORTRAN. It consists of over 800 subroutines and more than 87,000 lines and constantly being updated and enhanced in its capabilities. The code is vectorized to take advantage of vector registers and reduce computation time. Since vector registers are generally some multiple of 64-bit words, vector lengths of 64 or its multiples are used. In DYNA3D, elements are sorted by their number, material type and connectivity and arranged them into groups of 128. Vector processing can lead to significant jumps in execution efficiency, especially when large amount of data has to be processed. The elements in each group are processed simultaneously instead of computing one element at a time. Unlike a scalar processor which process one instruction at a time, a vector processor processes multiple instructions simultaneously as long as the operations are independent of each other, thus reducing computation time. Vectorizing the code also improves parallel computation efficiency in which more than one processor is used to do the computation. The vectorization process adds complexity to the source code. Furthermore, the memory usage in DYNA3D is minimized by storing all information in one large array. This method is very efficient because memory is allocated only to the variables that are used in the model. This method of sorting information adds further complexity to the source code.

www.manaraa.com

Because the source code is quite large, fully vectorized, and uses one array to store all information; the task of understanding the DYNA3D source code requires a considerable amount of time. This task, however, is made easier because the program is well structured. Flowchart for DYNA3D source code is shown in Figure 3-3. The flowchart is simplified for clarity.

The source code consists of three main subroutines: the input phase, the initiation phase, and the solution phase subroutines. In the input phase, the finite element model is read in from the input file and checked for errors. In the initiation phase, all variables are set to their initial values and the boundary conditions such as initial velocities and nodal constraints are imposed. These first two phases are performed once in the beginning of the simulation and usually require very little computation time when compared to the solution phase. In solution phase, a time integration loop based on the explicit central difference method is executed. A flow chart of the solution phase is shown in Figure 3-4. After each cycle of the solution phase loop, the simulation time is incremented by the time step and the next cycle is performed. This process is repeated until the final simulation time is reached. In a typical simulation, thousands of cycles are computed before the final simulation time is reached. Hence, most of the computation time is consumed in the solution phase.

During each cycle of the solution phase several steps are performed as shown in Figure 3-4. In the first step, the accelerations are computed by dividing the nodal force by the corresponding nodal mass. These accelerations are then modified to reflect the imposed boundary conditions and nodal constraints. In the next step, the velocities and displacements are updated using the central difference method equations. The pressures,

www.manaraa.com

concentrated loads, and body forces are then computed and added to the nodal force vector. Next, the solid, beam, thin shell, thick shell, and discrete elements are processed and the resulting internal forces are added to the nodal force vector. The contact forces are then computed and incorporated in the nodal force vector. In the final step, the simulation time is incremented by the time step and the next cycle is started.

Significant portion of the computation time in DYNA3D is spent in checking elements for contact and updating contact force vector. Different contact interfaces are treated differently. The most popular and commonly used contact interface is 'single surface contact'. The first step in this algorithm is to sort all the slave nodes and find out the minimum and maximum coordinates in X, Y, and Z directions. These extremum coordinates form the contact space, and this space is divided into 'buckets'. Next, all the slave nodes are sorted among the buckets and each node belong to one or the other bucket. This is called bucket sorting. Next, for each node, distances between the node and all other nodes in its bucket as well as its neighboring buckets are calculated and a nearest master-node is identified. Among the segments connected to the master-node, the nearest master-segment containing the slave-node is identified. And in the final stage, a slave-node—master-segment pair is checked for penetration and forces are calculated if penetration is found. This force is added to the nodal force vector.

www.manaraa.com

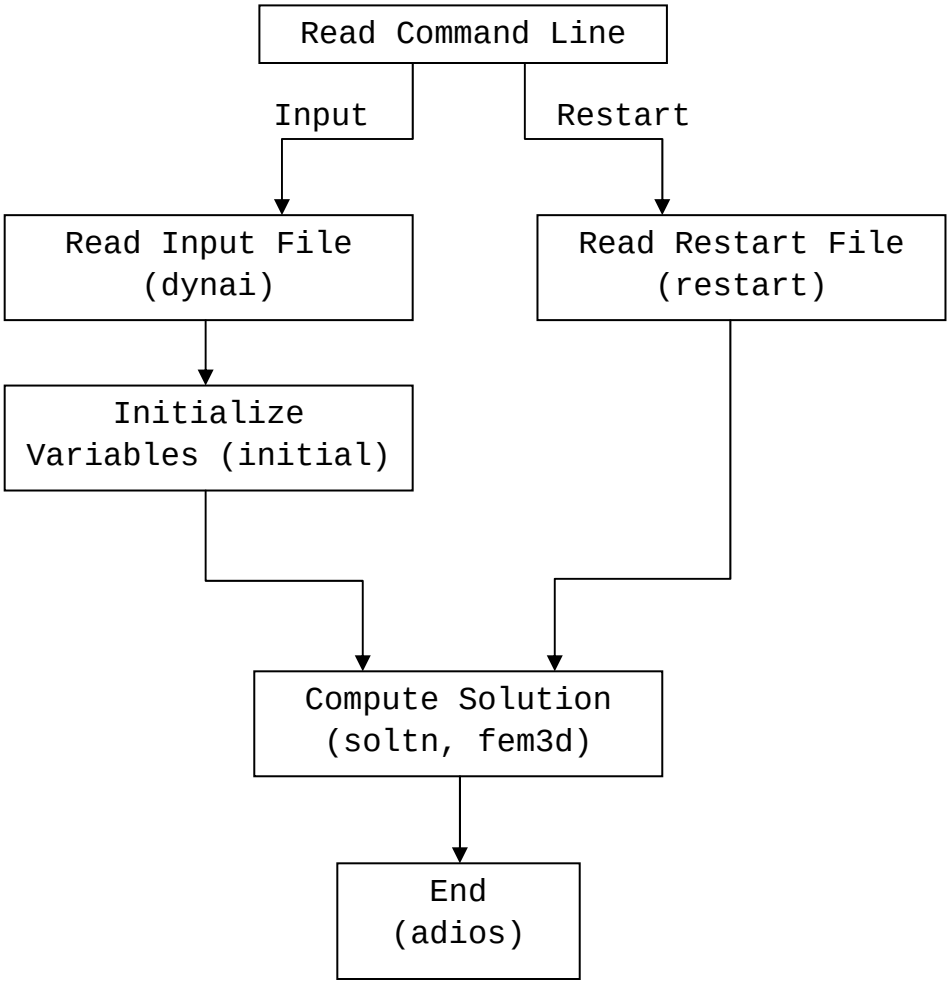


Figure 3-3 Simplified flow chart for DYNA3D

www.manaraa.com

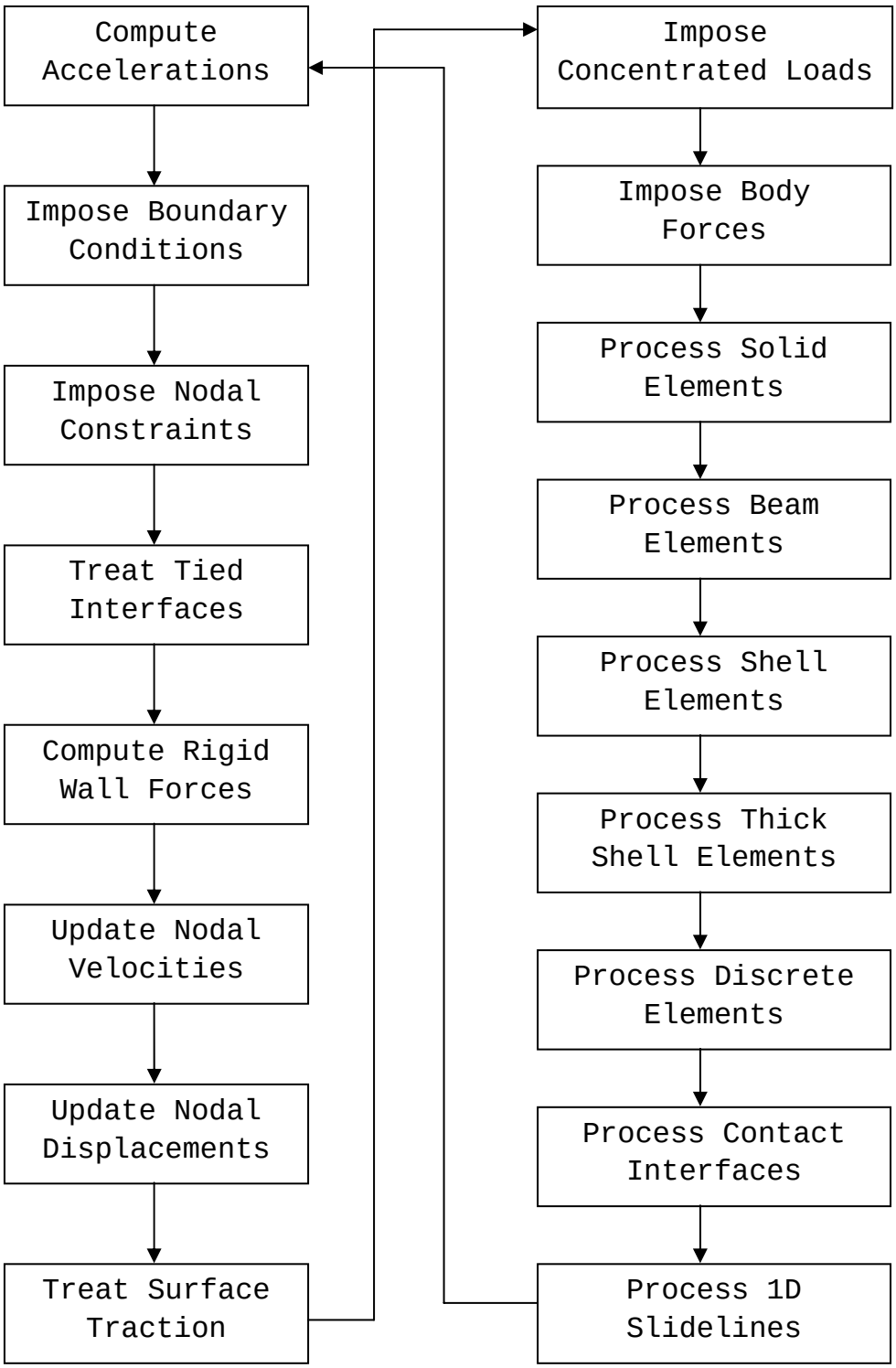


Figure 3-4 Flow chart for solution phase in DYNA3D

www.manaraa.com

4. CONTACT ALGORITHM IMPLEMENTATION

Approximately half of all the problems associated with numerical simulations of crashworthiness analysis are caused by contact search. The main problems in current contact algorithms originate in the node-to-segment nature of the model definition and in the search algorithms that define which nodes are in contact with which segments. In particular, treating node-to-segment conditions only leads to a systematic failure of detecting edge-to-edge or edge-to-segment penetrations. Early search algorithms detected a nearest master node for each slave node and selected a single nearest master segment from all segments connected to the nearest master node. This is an algorithm that works very well for the simulation of the contact of two smooth convex surfaces, but fails in many situations that occur in the high curvature failure modes such as in automotive structures. In particular, multiple impacts may occur simultaneously, and high curvatures and irregularity in the meshes may easily lead to the detection of a wrong neighbor segment, allowing numerous penetrations to remain undetected.

Most of the recent research on contact algorithms is aimed at improving the efficiency (making them faster in terms of computation). Hence there is a need for a contact algorithm which is more accurate in detecting penetrations and applying forces than the existing ones. In this chapter, new global and local search techniques along with an improved force-calculation method are introduced. The algorithms that are modified

www.manaraa.com

and algorithms that are added to implement new contact algorithm in DYNA3D are presented.

4.1 Contact algorithm considerations

To improve the contact search accuracy in DYNA3D, a new contact algorithm based on penalty formulation is added to the code. In implementing the contact algorithm several considerations were taken into account to ensure its generality and practicality. The main considerations are:

- The algorithm should provide accurate results. Even though importance was not given to the efficiency, the algorithm should not take unrealistic computational time. DYNA3D is an explicit finite element code and therefore the computation time is directly related to the time step size. The element size is predominant contributor to the time step size. Consequently, the contact algorithm should not do drastic modification to the size and shape of elements.
- The new contact algorithm should be implemented without restricting or limiting the original capabilities of the code. The DYNA3D code is a general-purpose finite element code. It has several features such as, different constraints, different connection options, different loading options, different material models and different type of element formulations. The new contact algorithm should not hinder functionality of any of the other features in the code.
- The contact algorithm should find penetration between all types of elements and apply appropriate penalty force to remove the penetration. The penalty force should take into consideration the types of elements, material properties of the

www.manaraa.com

elements and of course the magnitude of penetration. The new contact algorithm should be able to distinguish different types of elements; modify the search accordingly and compute the right amount of force.

- The contact algorithm should be formulated such a way that future improvements and new approaches can be easily added without major modifications. Since computation speed is constantly increasing and becoming less expensive, new ways to improve the accuracy of finite element analysis is constantly being found and codes are updated. The new contact algorithm should be structured such a way that if necessary these changes are easily incorporated.

4.2 Contact algorithm limitations

Implementing a new contact algorithm in DYNA3D that would incorporate all the features for all element types and material models involve significant amount of work and time. Hence some limitations are put on this contact algorithm to make the task achievable in reasonable amount of time. These limitations are only attributed to lack of time, not feasibility.

- The new contact algorithm considers only shell and beam elements. Segments of solid elements are not considered in the contact. Little modifications are necessary to make the new contact algorithm consider the segments with negligible or zero thickness such as segments of solid elements.

- The new contact algorithm development is limited to elasto-plastic material models. As rigid material model has infinite stiffness, material stiffness method cannot be used to calculate the contact penalty force. In order to determine

www.manaraa.com

penalty forces, the contact algorithms should be able to accept force vs. deflection curve as input. This feature is currently not available in the new algorithm and can be easily added later.

- The new contact algorithm assumes frictionless condition and does not compute frictional forces. With some additional effort, frictional forces using different friction models can be implemented and computed in the algorithm.

- The new contact algorithm assumes that there is no initial penetration at the beginning of simulation. In case of initial penetration, a large force will be applied to remove the penetration and this might lead to inaccurate results.

New Contact Algorithm

The proposed new algorithm, similar to other contact algorithms, is divided into three phases: global-search, local-search and penalty-calculation phases. Each of the phases is explained in detail in the following sections.

4.3 Global search (sphere-bucket-sort algorithm)

Sphere-bucket-sort uses the concepts of spherical-sorting and bucket sorting techniques. This algorithm uses advantages of both techniques while abridging their drawbacks.

In spherical-sorting algorithm, each and every element in the contact interface is superscribed in separate imaginary spheres of the smallest possible radius. Then every pair of spheres from the two contact surfaces is checked for intersection. Intersection

www.manaraa.com

check is a simple process of finding distance between the centers of spheres and comparing it with sum of the two radii of the pair. Non-intersecting pairs are discarded and intersecting pairs are checked for local contact.

In the bucket-sorting algorithm, the reference space is divided into "buckets" and every node in the contact is assigned to one of the bucket. The shape of the buckets depends on the number of dimensions of the problem. In one-dimensional problem, shape of the buckets is column of spaces; in two-dimensional problem it is a square or a rectangle; and in three-dimensional problem, it is a cube or a cuboid. The number of buckets depends on the element size as well as the size of the model. Once each node is assigned to a bucket, the local search process begins. In the local search, each slave node is associated with a nearest master node. The nearest master node is selected from the nodes in the bucket containing the slave node or the immediate neighboring buckets. The nearest master segment is then selected from the group of segments that contain the nearest master node.

Sphere-bucket-sort algorithm takes the following advantages from bucket-sort and spherical-sort algorithms.

1. Bucket-sorting eliminates local-search between the elements which are more than one bucket length away from the slave node.
2. Bucket-sorting is done once every 10 to 15 cycles to minimize the computation time.

www.manaraa.com

3. Spherical-sorting encloses every element inside an imaginary sphere to find intersecting pair. By enclosing each element inside a sphere, every possible combination of intersecting pairs is found and local search is done.
4. Spherical-sorting uses distance check to eliminate the non-intersecting pair of elements which is a simple and fast procedure.

Sphere-bucket-sort algorithm eliminates the drawbacks of bucket-sort and spherical-sort algorithms:

1. Bucket-sort updates the nearest master node by checking only the nodes which are connected to the element from the previous nearest master segment. This will fail to identify the correct master node when the elements are distorted severely compared to the initial configuration.
2. Spherical-sort algorithm checks distance between every pair of elements at every cycle and this will substantially increase the computation time if the number of elements in contact is large.

The proposed new global search, sphere-bucket-sort algorithm, instead of circumscribing each element in a sphere similar to spherical-sorting, it creates an imaginary sphere around each slave node with the center of sphere being the slave node itself. The radius of each sphere around the node is just big enough to circumscribe all the elements that are attached to that particular node. Centroid calculation of the elements is eliminated since the nodes are the centers of the spheres.

For a node which is connected to just beam elements, the radius of the sphere enclosing it will be the length of the longest element it is attached to. The radius of the

www.manaraa.com

sphere enclosing a node that is connected to only triangular elements is equal to the longest of the edges that the node is sharing. In case of a node connected to quadrilateral elements, the radius of the sphere is the longest of all the edges and diagonals that it is connected to. Figure 4-1 shows the concept of spheres enclosing nodes.

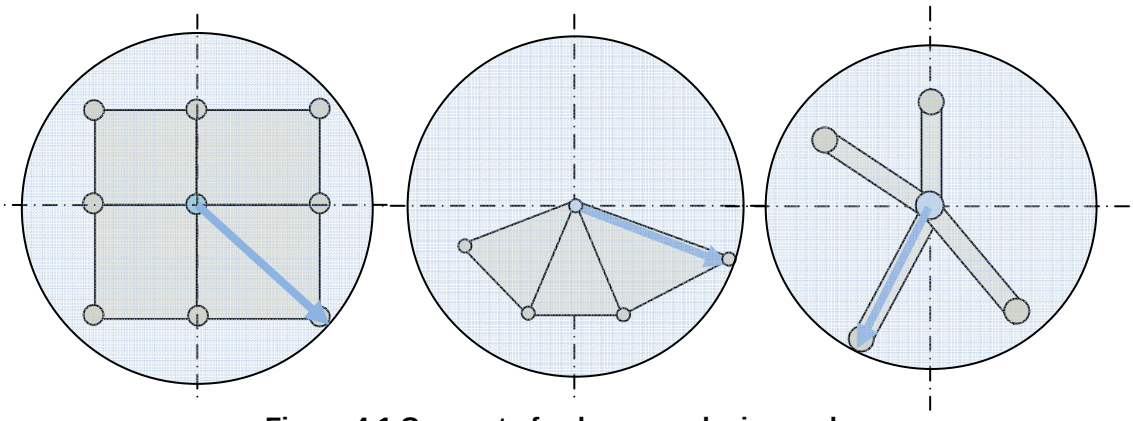


Figure 4-1 Concept of spheres enclosing nodes

Once all the slave nodes are enclosed inside the imaginary spheres, penetrating pairs of spheres are identified. This is done by computing distance between the centers of a pair of spheres and comparing it with the sum of their radii. A comprehensive global search simply takes each sphere and checks to see if it penetrates any other sphere enclosing other node. This will be an exhaustive search and the cost of search goes up with the number of slave nodes. Hence a neighborhood search (bucket-sort) is done after enclosing nodes inside imaginary spheres.

The bucket-sort is an algorithm that generates a reasonable neighborhood definition. The idea behind the bucket sort is to perform some grouping of the spheres so that the sort operation needs only to calculate the distance of the spheres in the nearest groups. Bucket-sort eliminates most of the non-intersecting spheres without having to find distance between them.

www.manaraa.com

The bucket-sort is a simple process of sorting. Like most sorting algorithms, it sorts in one dimension; and to sort across two or three dimensions, one-dimensional sorts are performed in nested loops. The number of buckets depends on the size of the elements as well as the extent of contact surface. It is determined by finding the maximum and minimum coordinates of the nodes in each direction and their difference is then divided by characteristic length. The characteristic length is calculated by taking a fraction of the longest segment diagonal in the surface definition. The number of buckets in the x, y, and z coordinate directions are given by

$$NX = \frac{x_{\max} - x_{\min}}{LMAX} \tag{4.1}$$

$$NY = \frac{y_{\max} - y_{\min}}{LMAX} \tag{4.2}$$

$$NZ = \frac{z_{\max} - z_{\min}}{LMAX} \tag{4.3}$$

where coordinates $(x_{\max}, x_{\min})$ $(y_{\max}, y_{\min})$ and $(z_{\max}, z_{\min})$ define the extent of the contact surface and are updated each time the bucket searching is performed, and *LMAX* is the longest characteristic length.

The bucket pointers in *x, y,* and *z* directions, of a node whose coordinates are *x, y* and *z*, are given by

$$PX = NX \cdot \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} + 1 \tag{4.4}$$

$$PY = NY \cdot \frac{(y - y_{\min})}{(y_{\max} - y_{\min})} + 1 \tag{4.5}$$

www.manaraa.com

$$PZ = NZ \cdot \frac{(z - z_{\min})}{(z_{\max} - z_{\min})} + 1 \tag{4.6}$$

If it were to store bucket pointers in each direction for all the nodes, then memory requirement would be very large. To reduce the memory requirements, sorting is nested in each direction and three pointers are collapsed into a single index. This single-index bucket-number is given by

$$NB = PX + (PY - 1) \cdot NX + (PZ - 1) \cdot NX \cdot NY \tag{4.7}$$

Figure 4-2 shows bucket pointers and single-index bucket-number.

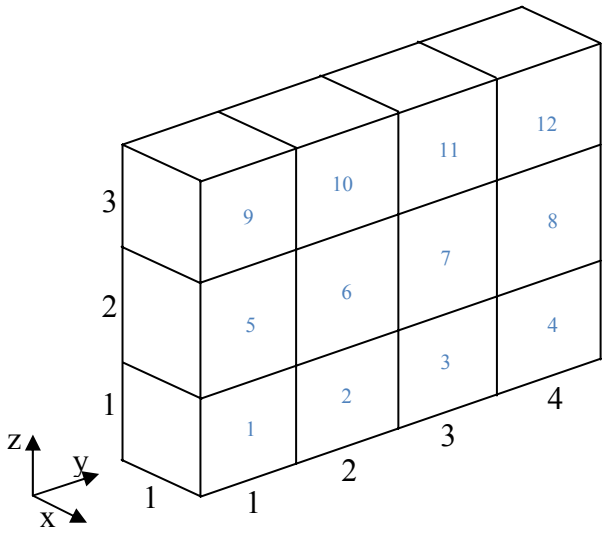


Figure 4-2 Bucket pointers and single-index bucket numbers

Once the contact surface is divided into buckets, every sphere is checked for all the buckets it intersects with. Each sphere could be a part of one, few or all the buckets. This is done using the following bucket finding algorithm.

- Loop over all the nodes

www.manaraa.com

- Find maximum and minimum dimension of the sphere in each direction using radius and center of the sphere.
- In each direction, find maximum and minimum bucket-pointer that sphere will intersect viz., *bkx1, bkx2, bky1, bky2, bkz1* and *bkz2*
- Calculate bucket number using

$$Bucket\text{-}number = i + (j-1) * NX + (k-1) * NY * NZ$$

- Update the *bucket-number* list with the node number.

Once every bucket has been updated by nodes which it intersects, every sphere in the bucket is checked for intersection with other spheres in the same bucket. Non-intersecting pairs are disregarded and intersecting spheres are saved for local search.

4.4 Sorting frequency

If sphere-bucket-sort is to be done every time step, the cost of computation increases significantly. In explicit finite element analysis, there is no need for sorting every time step as the critical time step is very small and the incremental displacements over a time step are very small relative to the mesh dimensions. In typical explicit finite element crash analysis the element size is approximately 5 to 7 mm and critical time-step is approximately one microsecond. At this element size and time-step, elements moving at 35 mph will displace about 0.0156 mm in one cycle (time-step). At this rate, it takes about 320 to 625 cycles for two elements which do not intersect in one sort to penetrate in next sort. At present, to be on safer side, sorting is done every 100 cycles. If the contact is used in high-speed impact simulation, the sorting frequency should be increased.

www.manaraa.com

4.5 Local Search

When the sorting frequency is reduced, the cost of local search dominates the contact search cost. The local search presented here doesn't distinguish between different shell elements. A 4-node quadrilateral element is treated as two 3-node triangular elements. This section briefly explains the algorithm used in the local search. The local search is divided into two parts: beam-to-beam penetration check and beam-to-triangle penetration check. Assumptions and computations involved in both parts are explained.

The local search is performed only if two spheres S1 and S2 around nodes n1 and n2, as shown in Figure 4-3, are identified as intersecting pairs by the global sphere-bucket-sort. In this case, all elements connected to node n1 are checked against all the elements connected to node n2. Each element is treated once as slave and once as master while checking against another element. From this treatment, the solution is identical at all times.

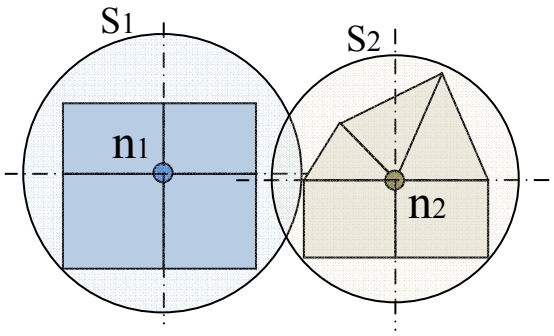


Figure 4-3 An example of intersecting pair

www.manaraa.com

4.5.1 Beam-to-beam penetration check

While checking for penetration between two elements E1 and E2, the edges of both elements are checked against one another. The edges are treated as beams of circular cross section and radius equal to the thickness of the corresponding element. Each corner is treated as a sphere of radius equal to the thickness of the element. The final shape of the beam as seen during local search is as shown in Figure 4-4.

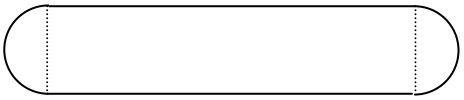


Figure 4-4 Geometric surface of a beam seen by contact algorithm

The beam-to-beam penetration check is done by calculating the shortest distance between two beams and comparing it with the sum of their radii. If the shortest distance between the two is less than the sum of the radii, then magnitude of penetration is given by the taking the difference of sum of radii and shortest distance.

Figure 4-5 shows notations used in a beam-to-beam check. The first step is to find if the beams are parallel to each other. Let $\vec{u}$ be the vector from node 1 to node 2, $\vec{v}$ from node 3 to node 4, and $\vec{w}$ from node 1 to node 3.

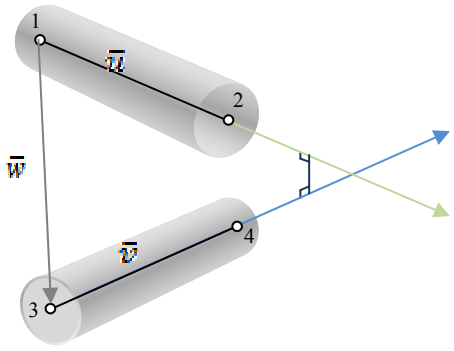


Figure 4-5 Notations used in beam-to-beam check algorithm

www.manaraa.com

The two beams are said to be parallel if the following condition is satisfied:

$$AC - B^2 \leq 0 \tag{4.8}$$

where

$A = \vec{u} \cdot \vec{u}$, $B = \vec{u} \cdot \vec{v}$ and $C = \vec{v} \cdot \vec{v}$.

If the lines are found to be parallel, the perpendicular distance between two infinite lines along $\vec{u}$ and $\vec{v}$ is calculated using

$$Pdist = \left(N_3 + T\vec{v}\right) - N_1 \tag{4.9}$$

where N_3 is position of node 3, N_1 is position of node 1 and $T = \frac{\vec{u} \cdot \vec{w}}{\vec{u} \cdot \vec{v}}$.

If *Pdist* is greater than the sum of the radii of the two beams, then these beams do not intersect. If it is less, then further checks are done to see if any portions of these beams overlap. If they overlap, more calculations are done to find out how much of the beams overlap and midpoints of their overlap. These midpoints of overlap are the points where forces are applied.

If the beams are found to be not parallel, the point of intersection of two infinite lines along vectors $\vec{u}$ and $\vec{v}$ are found. This point of intersection can be on one of the beams, on both the beams, or on neither of the beams. Depending on the point of intersection and position of the beams, the shortest distance is calculated using different approaches. A few of the various possibilities of beams coming into contact are shown in Figure 4-6.

www.manaraa.com

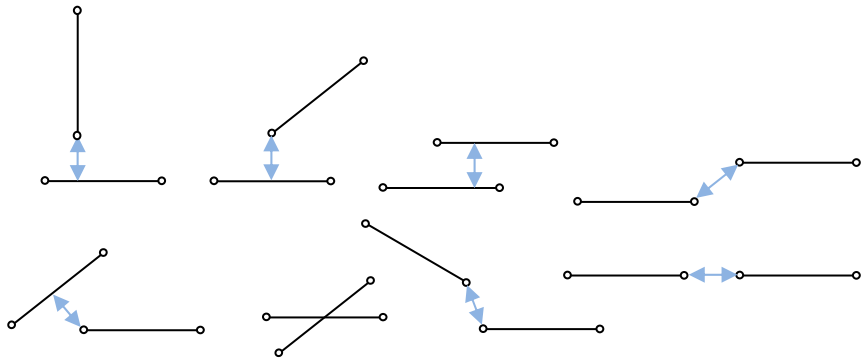


Figure 4-6 Various configurations of beams' contact

4.5.2 Beam-to-triangle penetration check

The shape of the triangular element is represented by a combination of a triangular prism, three cylinders whose axes are edges of element, and spheres whose centers are the corners of the element. The coordinates of the three nodes and thickness are used to create the prism. This triangular prism is checked with the cylinder shaped beam.

First, the cylinder is extended infinitely in both the directions of the beam to check if it intersects the plane of triangle. If the beam doesn't intersect the plane, then the beam is parallel to the plane of triangle. In this case, distance between the plane of triangle and axis of beam is calculated. If the distance is more than half of sum of thickness of triangle and radius of cylinder, the beam does not intersect the triangle. If the distance is less, further checks are done to see if any portion of beam overlaps the triangle.

www.manaraa.com

If beam is not parallel to the plane of the triangle, the intersection point of infinite long beam and plane of triangle is found. Figure 4-7 shows the notations used in beam-to-triangle check.

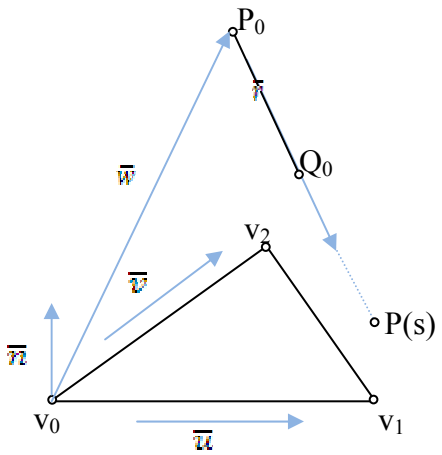


Figure 4-7 Notations used in beam-to-triangle check

The point of intersection between beam and the plane of triangle is found by using

$$P(s) = P_0 + f(Q_0 - P_0) \tag{4.10}$$

where $f = -\dfrac{\vec{n} \cdot \vec{w}}{\vec{n} \cdot \vec{r}}$

This point will be on the beam if $0 \le f \le 1$.

The intersection point is checked if it is inside the boundary of the triangle.

$$\begin{aligned} V(s,t) &= V_0 + s(V_1 - V_0) + t(V_2 - V_0) \\ &= V_0 + s\,\vec{u} + t\,\vec{v} \end{aligned} \tag{4.11}$$

where s and t are given by

www.manaraa.com

$$s = \frac{\vec{w} \cdot \left(\vec{n} \times \vec{v}\right)}{\vec{u} \cdot \left(\vec{n} \times \vec{v}\right)}$$
$$t = \frac{\vec{w} \cdot \left(\vec{n} \times \vec{u}\right)}{\vec{v} \cdot \left(\vec{n} \times \vec{u}\right)} \qquad (4.12)$$

This point is in the triangle if $s \geq 0,\ t \geq 0$ and $s + t \leq 1$

If the point of intersection is not in the triangle or on the beam, then there is no penetration between the beam and the triangle. But if the intersection point is on the beam and in the triangle, then proximities of this point with respect to all the nodes on the beam and the triangle are found and forces are calculated and distributed.

The depth of penetration is given by taking the difference of $\frac{\left(thk_1 + thk_2\right)}{2}$ and $\left(\vec{n} \cdot \vec{u}\right)$, where thk_1 and thk_2 are thickness of triangle and radius of beam respectively.

4.6 Penalty calculations

Once the depth of penetration and point of intersection are found by the local search method, as explained in the previous section, a massless spring is introduced to remove the penetration. The stiffness of the contact spring depends on the material properties and type and size of the elements involved.

The interface stiffness K is determined using both the master and slave surfaces. To ensure the stability of the solution, the stiffness is multiplied by a scaling factor which can be varied depending on the problem type. The overall contact spring stiffness is determined by having stiffness of both the surfaces in series as shown in Figure 4-8.

www.manaraa.com

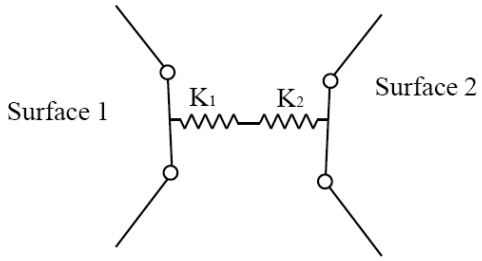


Figure 4-8 Contact surface stiffness in series

The equation for overall contact spring stiffness is

$$K_0 = s\frac{K_1 K_2}{K_1 + K_2} \tag{4.13}$$

where s is the stiffness scaling factor. Generally a scale factor of unity or more is used for high speed contact problems and a scale factor of less than unity for lower speed contact problems.

K_1= surface-1 stiffness

K_2= surface-2 stiffness, and

K_0 = overall contact stiffness.

If the contact element is a shell element, then surface stiffness for that element is given by

$$K_i = \frac{E_i A_i}{\max(shell\ diagonal)} \tag{4.14}$$

where E = Modulus of elasticity

A_i= Segment area

www.manaraa.com

and if the contact element is a brick element, the stiffness is calculated by

$$K_i = \frac{B_i A_i^2}{V_i} \tag{4.15}$$

where B_i = Bulk modulus

A_i = Segment area

V_i = Element volume

The contact force on master surface is computed using

$$F_c = K_s P n_i \tag{4.16}$$

where F_c is the total contact force, P is magnitude of penetration, n_i is the normal to the master segment and K_s is stiffness which given by

$$K_s = K_0 \left(\frac{Gap}{Gap - P} \right) \tag{4.17}$$

where Gap is smallest of the thicknesses of the two segments. The stiffness is nonlinear and varies with the amount of penetration. It increases exponentially as the penetration increases. Figure 4-9 shows the difference between forces using constant stiffness and variable stiffness. Force using constant stiffness linearly increases as the depth of penetration increases whereas force using variable stiffness increases exponentially. It can be noted that both the forces are nearly identical when penetration is small, but for any reason if penetration could not be removed, the variable stiffness method applies significantly higher force compared to the constant stiffness method.

www.manaraa.com

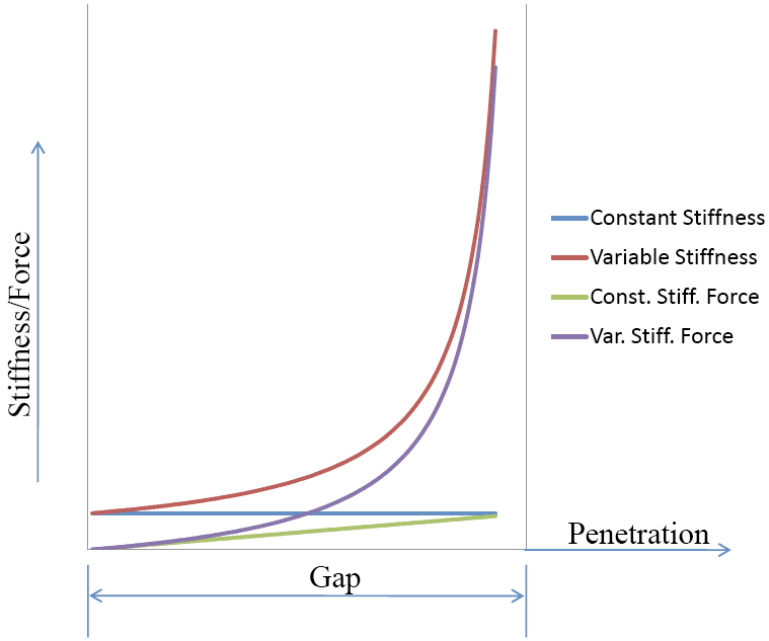


Figure 4-9 Penetration-Force curve

Once the magnitude and direction of contact force on master surface is computed, same magnitude of force is applied in opposite direction on the slave segment. Depending on the location of the penetration point relative to the segment nodes, the forces are distributed using shape functions.

Shape functions for a beam segment are given by

$$\begin{matrix} f \\ 1-f \end{matrix} \tag{4.18}$$

and shape functions for a triangular segment are given by

$$\begin{matrix} 1-s-t \\ s \\ t \end{matrix} \tag{4.19}$$

www.manaraa.com

where f, s and t are as shown in Figure 4-10.

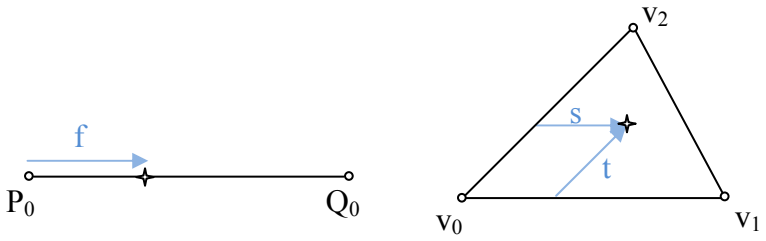


Figure 4-10 Shape functions while applying force

In summary, the new contact algorithm which includes a new global search method, a new local search method and improved contact mechanics has the following advantages:

1. It identifies all possible contact element pairs accurately
2. It considers accurate geometry of the contact surfaces
3. Solves the issues with edge-to-edge and edge-to-surface contact checks
4. Applies exponentially increasing constraint forces so that penetrations are removed at all times.

4.7 New contact algorithm implementation

The new contact algorithm has been implemented in the DYNA3D finite element program. It consists of three stages: global search, local search, and force (penalty) calculation. Algorithms for these stages are developed and implemented. The algorithm is developed for one-dimensional beam elements with constant circular cross-section and two-dimensional shell elements with constant thickness. Extending this algorithm to treat

www.manaraa.com

solid elements involves additional steps of transforming surfaces of solid elements into two-dimensional shell elements and performing contact search between those shell elements.

The new algorithm is aimed at improving the accuracy of the current DYNA3D contact algorithms. The current algorithms use bucket-sort and incremental-search for global search which fails when the contact surface has high curvature or when the slave nodes move at high speed. These algorithms work well for most of the general cases, but when it involves significant sliding between the elements, the chances of failing to detect the penetration increases. In the local search, the current algorithms use 'avoiding penetration of slave-nodes on master-surface' approach. In this approach, a slave-node has only one master-element at a given cycle and in the next cycle, the master-element for this cycle will be the same element or one of the elements attached to it. This approach fails when a node approaches two elements symmetrically. These drawbacks of the current contact algorithms in DYNA3D are overcome in the new contact algorithm.

The new algorithm uses spherical-bucket-sort which considers all possible pairs of slave-node—master-segment with a reasonable increase in computation time. For local search, it uses beam v/s beam and beam v/s triangle approach instead of node v/s segment. This approach eliminates undetected penetrations.

The implementation of the new contact algorithm in DYNA3D involved modifying several subroutines and adding several other subroutines to the code. In this section, the modifications made to the original code and the subroutines added to the code are described.

www.manaraa.com

4.7.1 Modifications to the source code

The first modification to the DYNA3D code was to allocate extra memory space during initialization process for storing the variables required during contact search. This is a finite amount of memory space which depends on the number of slave nodes and the smallest characteristic length in the model. For ease of implementation, this space is kept separate from the other variables space in the code and made user defined. The user needs to be careful while defining this space since allocating less than the minimum required may result in error termination. Allocating a larger amount of space than required is a good practice and it does not affect computational time required for the simulation.

The next modification to the code was in the input phase subroutines. The input format for the new subroutine was kept similar to the original 'single-surface' contact input. Few subroutines were added to the input phase to read, store and sort the contact segments and nodes. This information is stored in the extra memory space that is allocated by the user.

Several other modifications were made to the DYNA3D to incorporate the new contact algorithm. A total of 17 new subroutines were added and two existing subroutines were modified.

4.7.2 Added subroutines

The subroutines that are modified and new ones that are added to the DYNA3D code are listed in the Table 4-1. The subroutines uminit, umconnec, umsrtsn and

www.manaraa.com

umupdthk constitute addition to the initialization phase. The subroutines umchkbeam, umfndbks, uminitbks, uminithk, umradii, umslmrlist constitute global search phase. The subroutines umchkdist, umb2b, umb2t constitute local search; and the subroutines umbforce and umtforce constitute force calculation phase. The flow chart of this contact algorithm is shown in Figure 4-11. The initialization phase subroutines are executed once in the beginning of the computation. The global search phase subroutines are executed every 100 cycles. The local-search and force-calculation phase subroutines are executed at every time increment cycle.

In the initialization phase, extra memory storage space is allocated for the contact calculations. After DYNA3D reads the input file, slave nodes are extracted from the slave segments definition and are sorted. The thicknesses of these nodes are extracted from the corresponding elements and stored. The slave segments that connected to each of the slave nodes are stored.

The global search phase is performed once every 100 cycles to minimize the computation time. In this phase, the minimum volume that is orthogonal to the global coordinate system and required to enclose the contact surfaces is computed by spanning maximum and minimum value in each dimension. Using the minimum characteristic length, this volume is then divided into a number of buckets and each bucket is given a unique number. Imaginary spheres around each slave node are constructed with radii computed based on the size of elements that the slave nodes are connected to. From the intersections of all spheres and buckets, master-slave segment pairs are identified and stored in the memory.

www.manaraa.com

In the local search phase, each master-slave segment pair is checked for penetration. Depending on the type of element, combination of 'beam-to-beam' and 'beam-to-triangle' checks are performed and if a penetration is found, the necessary force is applied, in the force-calculation phase, on the nodes to overcome penetration.

Once the new contact algorithm is implemented and validated, several problems were simulated using both the new contact algorithm and the current contact algorithm in DYNA3D to show the differences and improvements. The results and comparisons from these simulations are presented in chapter 5.

www.manaraa.com

Table 4-1 New and modified subroutines

New Subroutines	***Description***
umb2b	checks penetration between two beams/cylinders
umb2t	checks penetration between a beam and a triangle
umbforce	calculates penalty forces on beams and updates force vector
umchkbeam	checks if the given element is a beam or not
umchkdist	checks if the elements of given nodes have possibility of interpenetration
umconnec	makes the list of all the slave segments connected to the given node
umfndbks	finds all the buckets to which the given node belongs
uminit	initializes the extra memory storage
uminitbks	initializes the buckets and bucket sizes
uminithk	initializes radii of slave nodes
umlcsrch	does local search between given two elements
umradii	calculates and stores radii of slave nodes
umsl4	main contact subroutine
umslmrlist	makes list of slave and master elements in the given bucket
umsrtsn	sorts slave nodes
umtforce	calculates penalty forces on triangle and beam, and updates force vector
umupdthk	update thickness of nodes
Modified Subroutines	***Description***
dynai	reads input and initializes variables
soltn	computes solution phase

www.manaraa.com

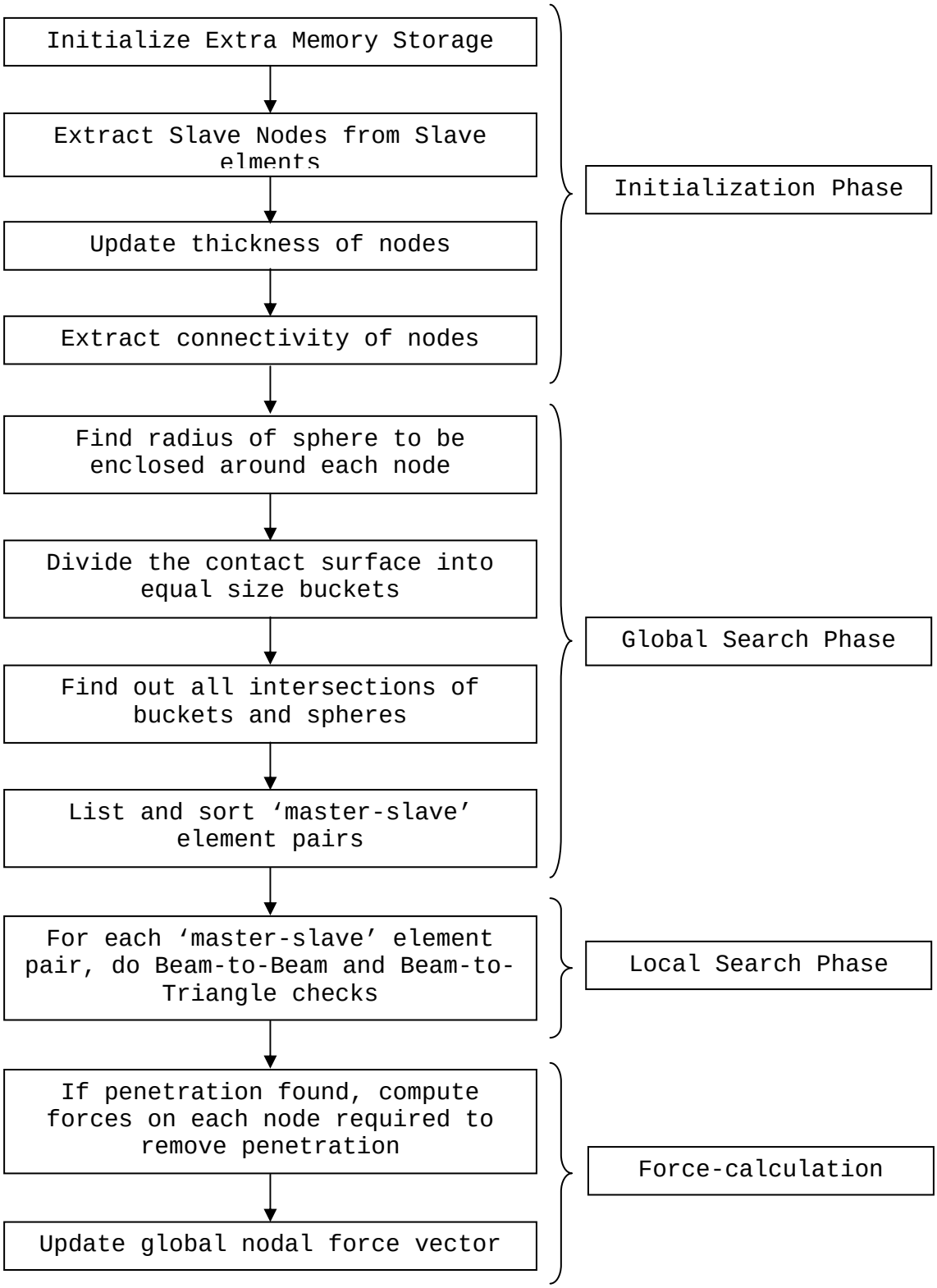


Figure 4-11 Flow chart for the new contact algorithm

www.manaraa.com

5. VALIDATION OF NEW CONTACT ALGORITHM

A new contact algorithm that consists of new global and local search and improved force-calculation methods is implemented in DYNA3D. To check validity and to demonstrate the general behavior of the new contact algorithm, few element level and component level examples are simulated and the results are presented in this chapter. The element level examples focus on the cases where DYNA3D fails to detect contacts and penetrations using the existing 'single-surface' and 'nodes-to-surface' contact interfaces. The component level examples are: Hertz contact problem, contact patch test, impact between two tubes of different mesh densities and crush of a symmetric tube between two rigid walls. The Hertz contact problem was simulated to validate maximum stress induced in the model. The contact patch test gives an indication of stability and stress propagation during a contact impact. An impact between two tubes of different mesh densities and symmetric tube crush were simulated to examine the compliance and to demonstrate the self-contact search capability of the new contact algorithm.

5.1 Element level validation

The element level examples are simple problems in which an element contacts one or two elements in different configurations. All elements were assigned elastic material properties and the contact was assumed frictionless. All the examples were also simulated using currently available contact algorithms in DYNA3D and the results were compared with results from new contact algorithm.

www.manaraa.com

5.1.1 Example 1 (Nodes to surface)

In this example, a small plate was impacted on one end of a big plate. The small plate was created using 9 equal size shell elements and the big plate using one shell element. A thickness of 3 mm and elastic-plastic material property were assigned to both plates. A contact was defined between the plates and the distance between the plates was measured. Figure 5-1 shows the initial configuration and velocity vector.

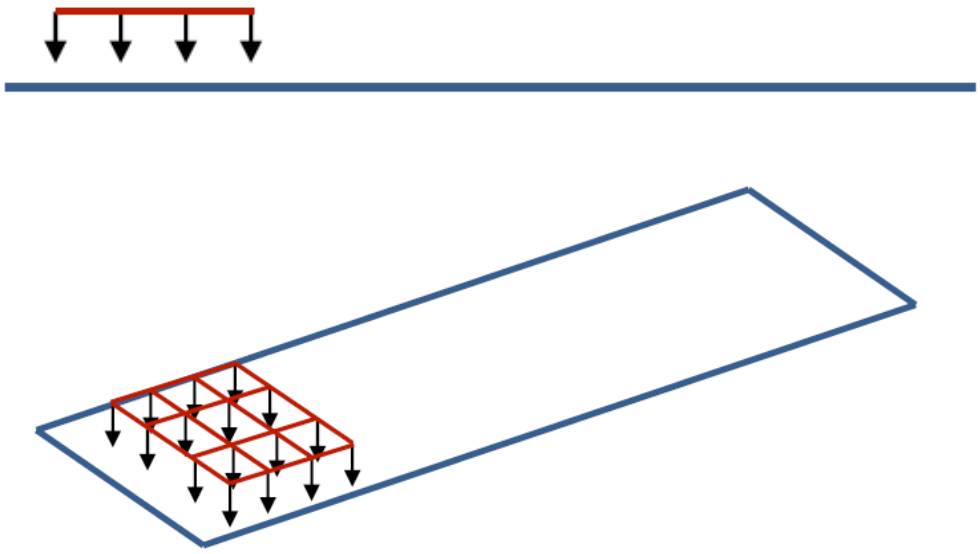


Figure 5-1 Initial configuration and velocity vector of Example 1

Three different simulations were run using the same configuration. First – using the original 'single-surface' contact in DYNA3D, second – using 'nodes-to-surface' contact in DYNA3D and third – using the (new contact algorithm) in DYNA3D. In each case the distance between the plates was monitored and the results are shown in Figure 5-2.

In the first case, when using 'single-surface' contact, the contact fails to detect penetration of few of the nodes. With this contact algorithm, in a pool of slave segments,

www.manaraa.com

when one slave segment is found penetrating a master segment, the same slave segment is not checked with other master segment. This phenomenon can be observed in Figure 5-3 in which it can be seen that the contact algorithm prevents penetration of node 'N_b' but not node 'N_a'. When the slave segments were defined in a different order, a different behavior was observed. The distance between node 'N_a' and the large plate is represented by curve 'A' in Figure 5-2.

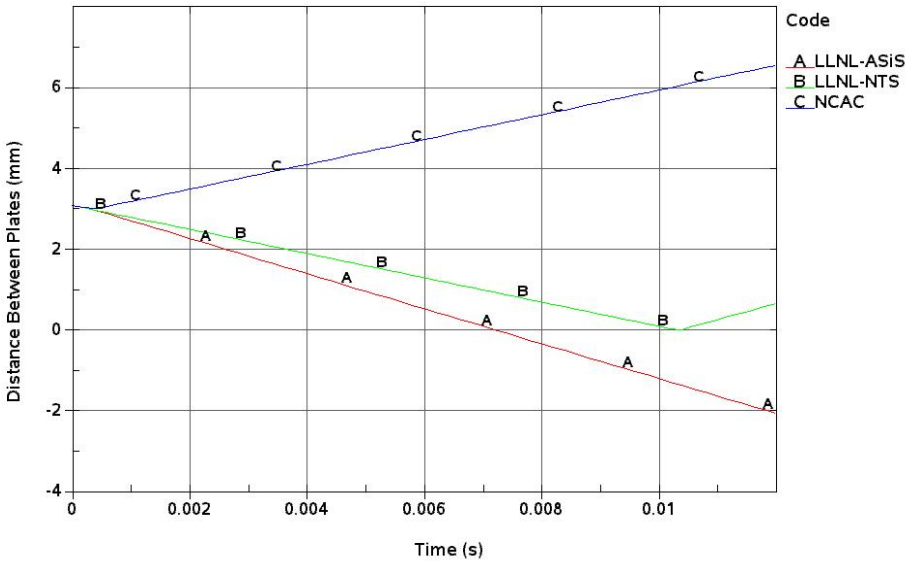


Figure 5-2 Distance between two plates

Figure 5-3 Failure to detect penetration using single surface contact

In the second case, using the original 'nodes-to-surface' contact algorithm, penetration of all slave nodes were detected by the master segment, but the thickness of

www.manaraa.com

the segments were not taken into account. The distance between the plates decreases to zero before it starts increasing. This is shown as curve 'B' in Figure 5-2.

In the third case, using the new contact algorithm, all penetrations are detected and the true thickness of each segment was taken into consideration. Curve C in Figure 5-2 represents the distance between the small and large plates, and it can be seen that the distance between the plates starts increasing after it reaches 3 mm.

5.1.2 Example 2 (Surface to surface)

In this example, two equal sized rectangular elements whose widths are smaller than their lengths were placed in such a way that their lengths were perpendicular to each other. Both elements are assigned elastic-plastic material property and a thickness of 3 mm. An initial velocity was assigned to the top element such that the face of top element impacts the face of bottom element. Three cases were simulated using DYNA3D with different contact algorithms. The configuration and velocity vector is shown in Figure 5-4.

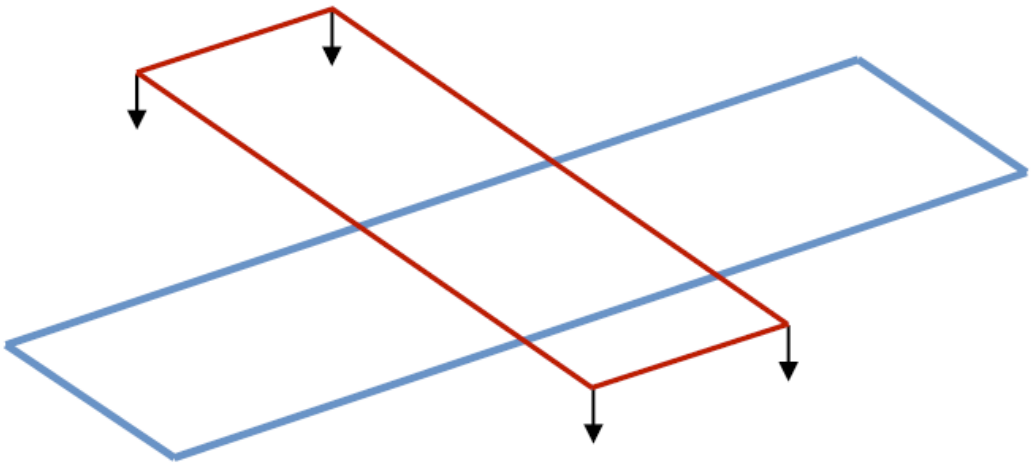


Figure 5-4 Initial configuration and velocity vector of Example 2

www.manaraa.com

The first and second cases in which the original 'single-surface' and 'nodes-to-surface' contact algorithms are used respectively fail to detect penetration. Since both algorithms use the same concept; preventing nodes from penetrating the surface, and no node is penetrating any surface in this example, the contacts fail. Curves 'A' and 'B' in Figure 5-5 represents the distance between the two elements. It can be seen that the two elements traverse each other without any resisting force.

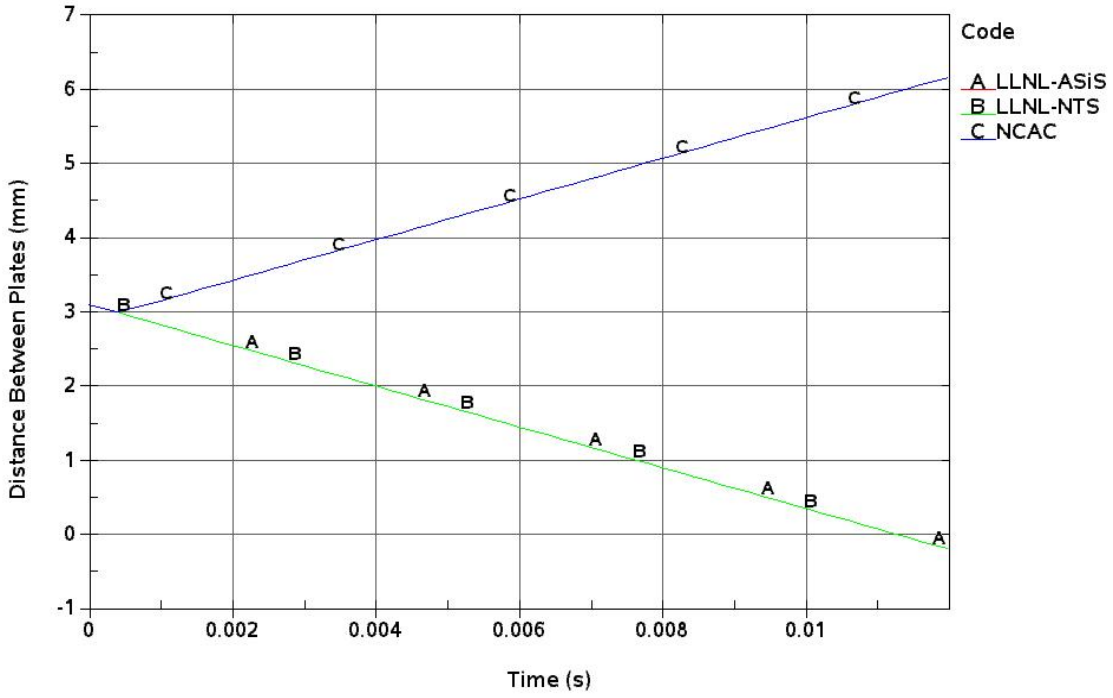


Figure 5-5 Distance between the elements

The third case, in which the new contact algorithm was used, detects the penetration at the defined thickness and applies appropriate forces on nodes to remove the penetrations. Curve 'C' in Figure 5-5 shows the distance between the two elements for the case of new algorithm. It can be seen that the distance between the elements decreases to 3 mm and then increases. By comparing the curves 'A' and 'C', it can be

www.manaraa.com

noticed that the rate of increase of distance between the elements during the rebound phase is equal to the approach speed. It can be inferred that the amount of force applied to remove the penetration is accurate.

5.1.3 Example 3 (Edge to surface)

This example is similar to the example 2 except that the faces of elements are perpendicular to each other. Both elements were assigned elastic-plastic material property and a thickness of 3 mm. The problem configuration and initial velocity is shown in Figure 5-6. Element 2 was given an initial velocity such that face of the element 2 impacts an edge of element 1.

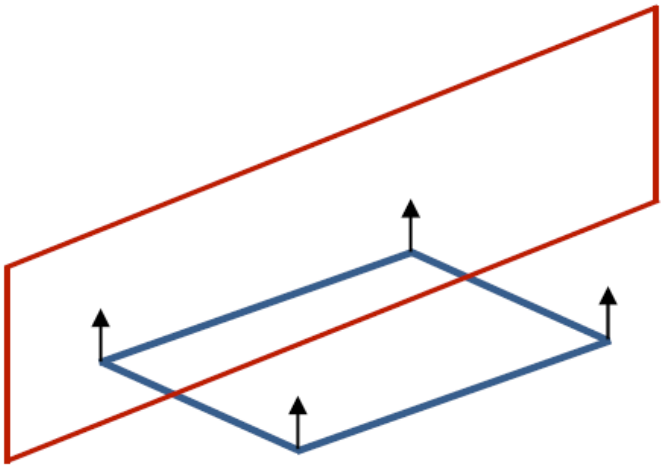


Figure 5-6 Initial configuration and velocity vector of Example 3

Similar to examples 1 and 2, three cases were simulated using DYNA3D with different contact algorithms. In each case the distance between elements was measured to see the accuracy of contact algorithms. The results are shown in Figure 5-7. Curve 'A' and 'B' represent distance between the elements when using 'single-surface' and 'nodes-to-surface' algorithms respectively. In these cases, the contacts do not detect elements

www.manaraa.com

crossing each other and apply no reaction forces. Element E1 penetrates element E2 without any resistance and the distance between them becomes negative.

The distance between the elements for the third case is represented by curve 'C' in Figure 5-7. In this case the new contact algorithm detects the penetration and applies forces on nodes to remove the penetration. It is evident from comparing curves 'A', 'B' and 'C' that the forces applied in the third case is accurate as the rate of approach of elements in case 'A' and 'B' are equal to the rate of departure in case of 'C'.

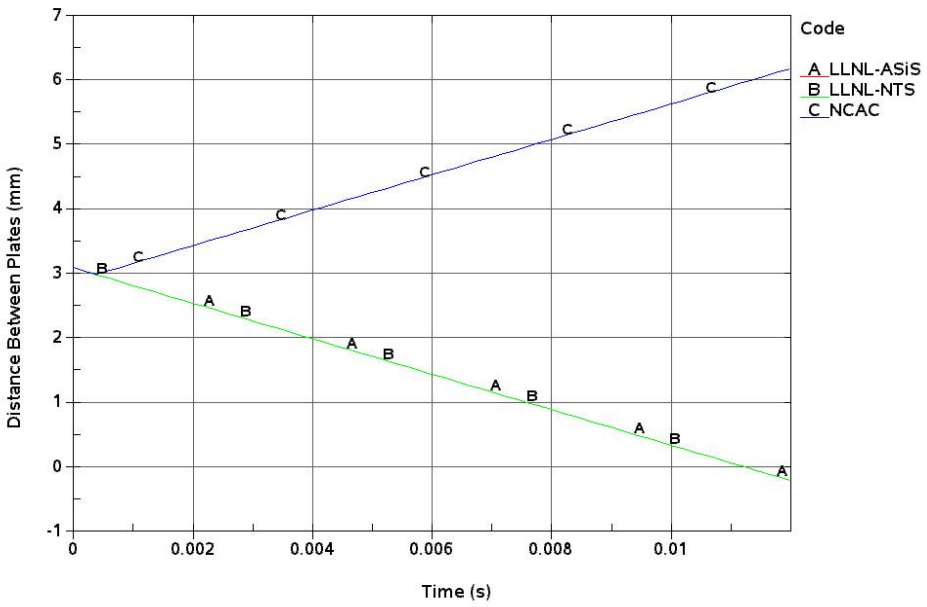


Figure 5-7 Distance between the elements

5.1.4 Example 4 (Edge to edge)

In this example, two parts; one with two-rectangular elements forming a shape of 'V' and another with one-element in the shape of a rectangle, were made to impact each other such that an edge from each part comes in contact. The problem configuration and

www.manaraa.com

velocity vector are shown in Figure 5-8. Both parts were assigned the elastic-plastic material property and the elements were assigned a thickness of 2 mm. Three cases were simulated using DYNA3D with three different contact algorithms. Similar to the previous examples, the original 'single-surface', the original 'nodes-to-surface' and the new contact algorithms were used.

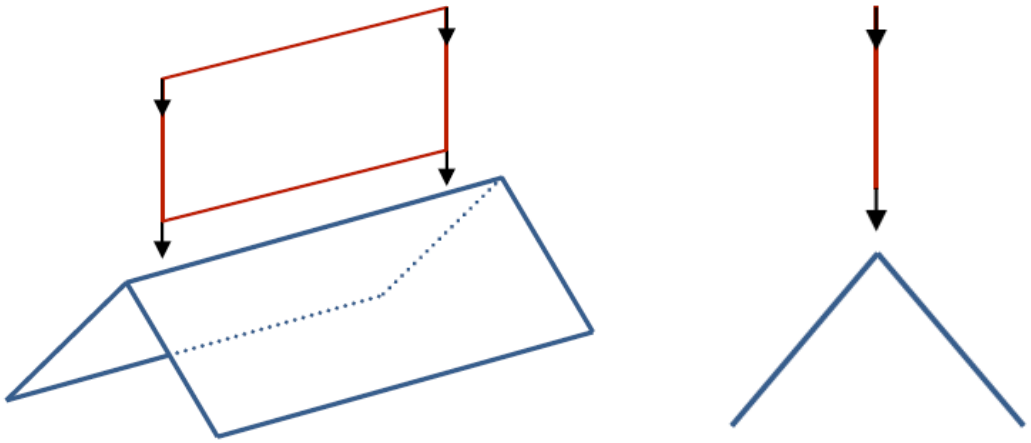


Figure 5-8 Initial configuration and velocity vector of Example 4

Figure 5-9 shows the comparison of minimum distances between the edges in the different cases. Curve 'A' gives the distance between the edges when the 'single-surface' contact was used. Curves 'B' and 'C' show the distances when the 'nodes-to-surface' contact algorithm was used (two cases with different master surface definitions were simulated). Curve 'D' shows the distance when using the new contact algorithm.

www.manaraa.com

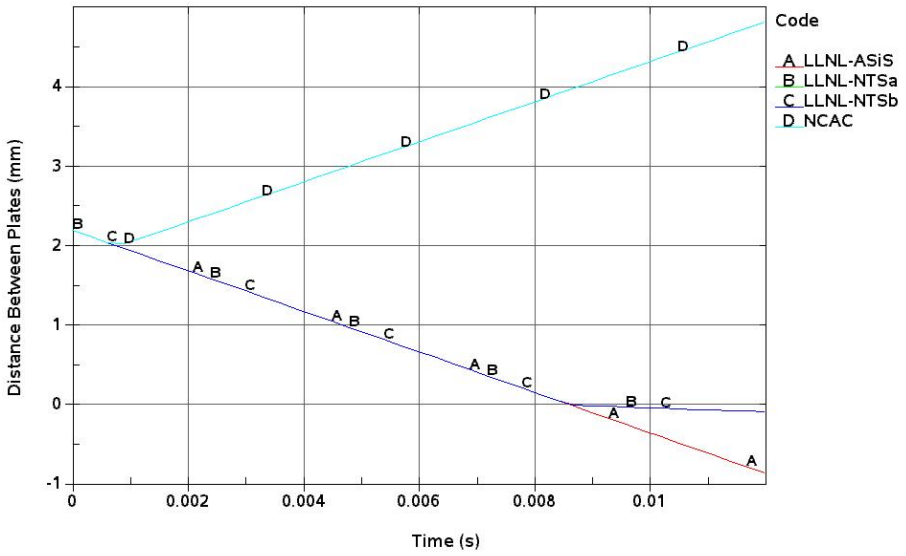


Figure 5-9 Distance between the contacting edges

It can be seen that in case 1 (curve 'A'), the edge to edge contact was not detected and the nodes penetrate without any contact force applied on them.

Simulation using nodes-to-surface contact algorithm was further divided into two sub cases in which the master segments were defined differently. Curves 'B' and 'C' in Figure 5-9 represent these sub-cases. Even though the distances between the edges were the same in both the cases, the slave nodes' behavior was completely different. Figure 5-10 shows two different behaviors of slave nodes with two different master segment definitions. The contact detects the penetrations of slave nodes with master segment E1 but not E2. Depending on which element is defined as E1, the slave nodes move away from the element E1. Slave nodes are checked for penetration through the first master segment and if penetration is found forces are applied and no further checks are done with the other master segment. In both cases, the normals of master segments were

www.manaraa.com

pointing towards slave nodes. The behavior of slave nodes depends on the order in which master segments are defined.

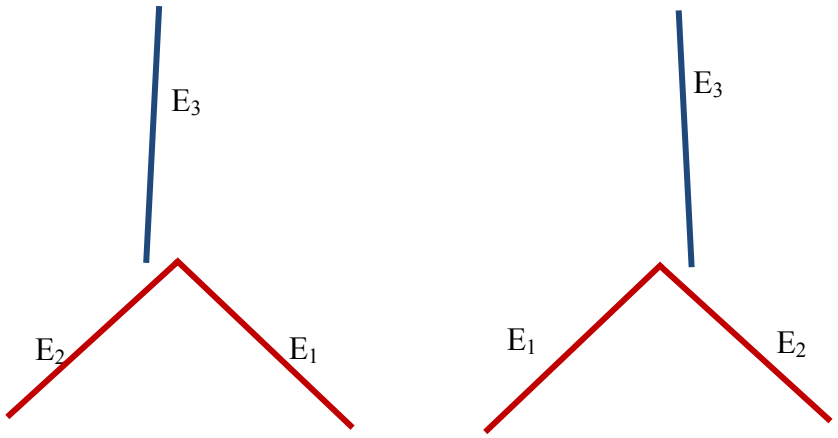


Figure 5-10 Different behaviors of slave nodes in nodes-to- surface contact

Case 3, in which the new contact algorithm was used, the penetration is detected at the right distance and applies appropriate reaction force. Curve 'D' in Figure 5-9 shows the distance between contacting edges during the simulation. It can be seen that the distance decreases to 2 mm, which is thickness of both the parts, and then increases.

5.1.5 Example 5 (Multiple contacts)

Example 5 is similar to example 4 except that the rectangular element is set up such that its contacting the 'V' shaped elements from inside. Schematic diagram of the problem along with initial velocity vector is shown in Figure 5-11. Both parts were assigned elastic-plastic material properties and a thickness of 2 mm. Three cases were simulated using DYNA3D with the original 'single-surface' , the original 'nodes-to-surface' and the new contact algorithms.

www.manaraa.com

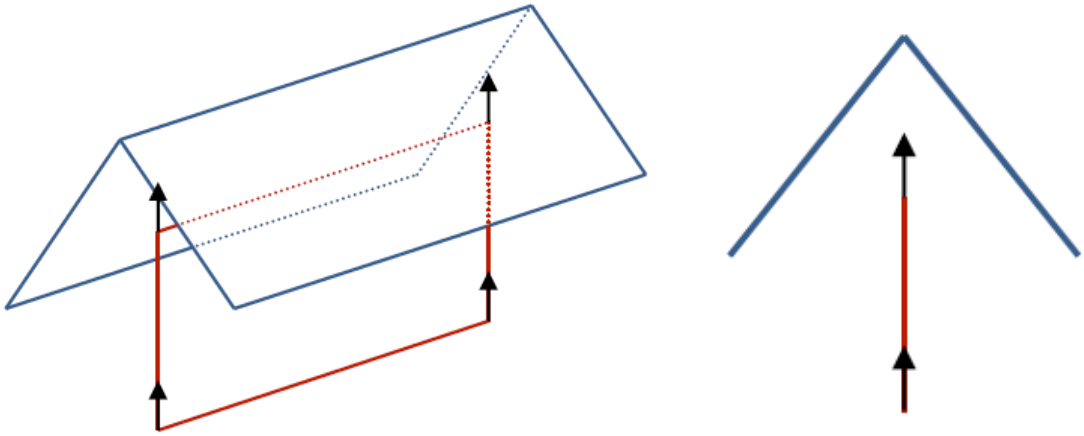


Figure 5-11 Initial configuration and velocity of Example 5

In the first case, where 'single-surface' contact algorithm is used, the penetrations are identified but not at right distances. Figure 5-12 shows the two different behaviors when the order of slave segments is changed. The numbering represents the order in which slave segments are defined.

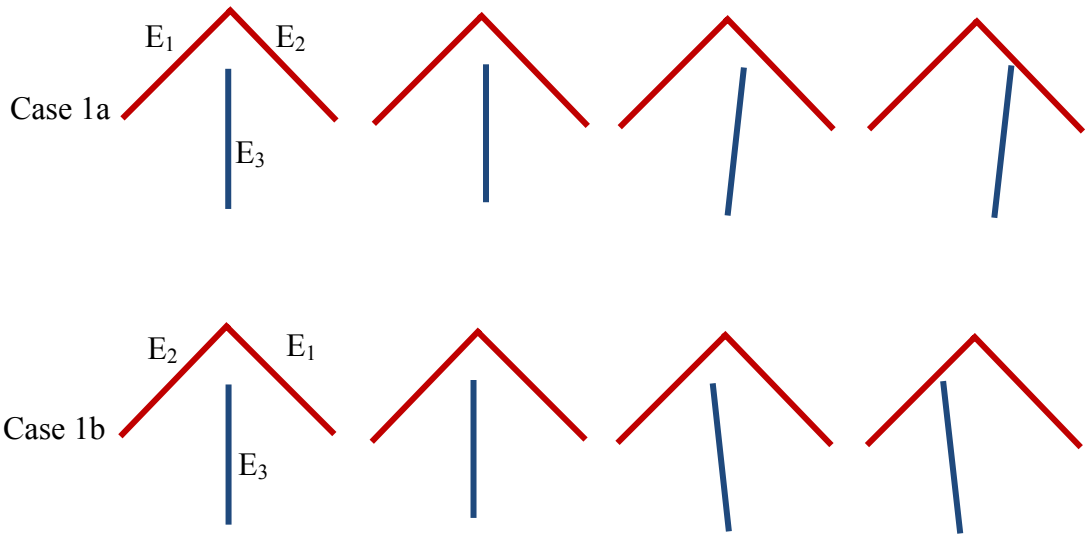


Figure 5-12 Two different behaviors of slave nodes when using single-surface contact

www.manaraa.com

Contact forces are applied on the nodes when slave nodes from E3 are located at a distance of 1.23 mm from E1. Then the nodes from E3 are pushed towards E2 till the distance between E2 and E3 reaches zero. At this point, a large force is applied on the nodes of E2 and E3. Depending on what order the slave segments are defined, the behavior of the plates change. Figure 5-13 shows the distance between the edges of two plates during the simulation. It can be seen that even though the thicknesses of the elements are 2 mm, the distance between the edges becomes less than 2 mm.

In the second case, where the 'nodes-to-surface' contact algorithm is used, the penetrations are identified but the distances at which the forces are applied are not accurate. The element thicknesses are not taken into consideration while calculating penetration. For this case, Figure 5-14 shows the distance between the edges.

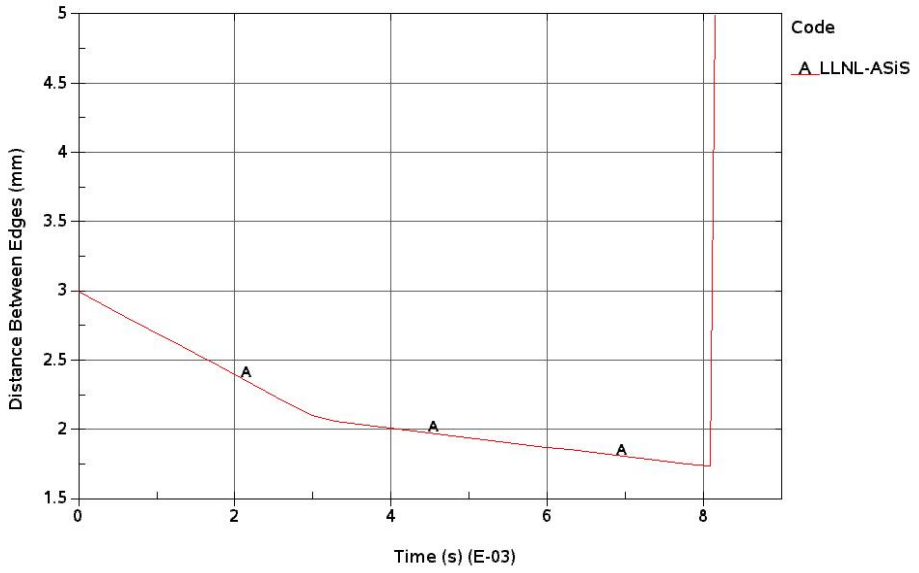


Figure 5-13 Distance between edges when using single-surface contact

www.manaraa.com

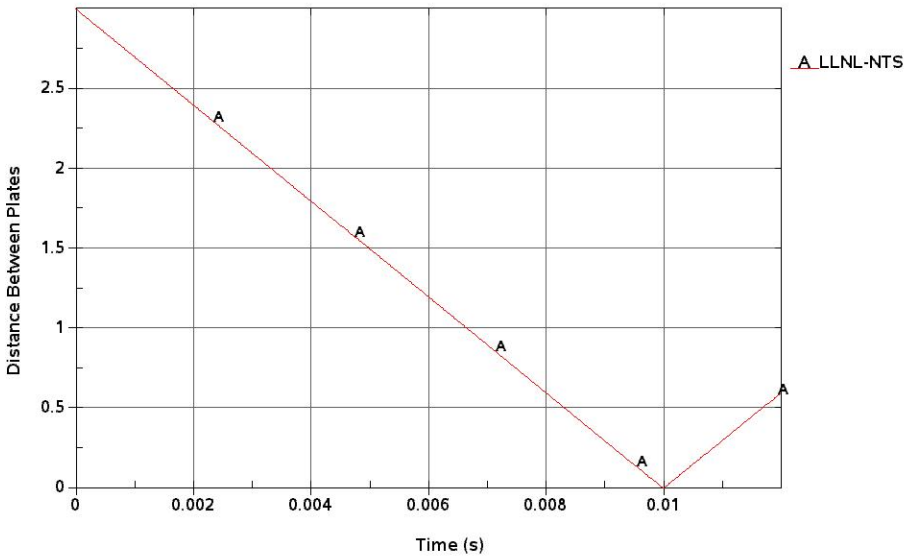


Figure 5-14 Distance between edges when using nodes-to-surface contact

Figure 5-15 shows the distance between the edges during the simulation using new contact algorithm. Element E3 comes in contact with elements E1 and E2 simultaneously and symmetric forces are applied on the nodes. Figure 5-16 shows the geometry of the parts when they come in contact with each other. The dotted lines represent the boundary of the elements. The smallest distance measured between the edges is 2.828 mm.

www.manaraa.com

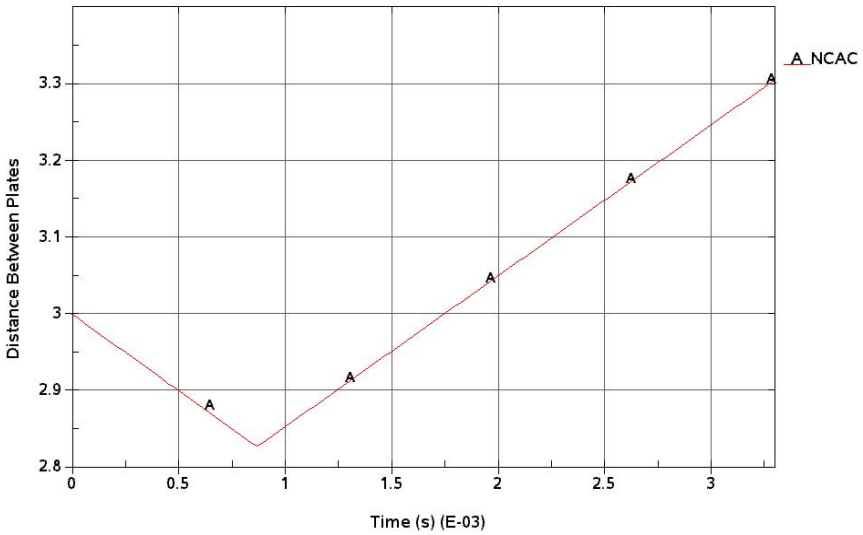


Figure 5-15 Distance between the edges when using new contact algorithm

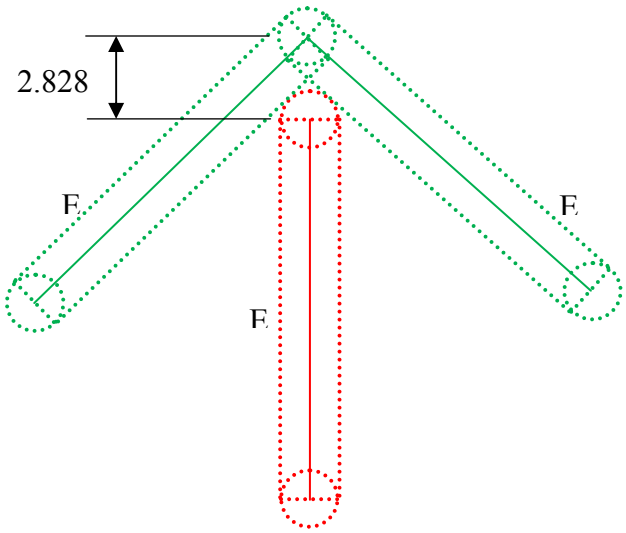


Figure 5-16 Configuration when elements are in contact

5.2 Component level validation

In these examples, several capabilities of contact algorithm are validated: ability to check contact and penetrations between different size of elements, ability to perform self-

www.manaraa.com

contact search, ability to apply right magnitude of force such that accurate stress is seen in the elements, and. ability to apply constant stress along the contact surface, In all these examples, contact is assumed frictionless.

5.2.1 Example 6 (Impact between two tubes)

In this example, the new contact algorithm is tested for its ability to search penetration between different size meshes. The results from the simulation are compared with the results from the DYNA3D's single-surface contact algorithm. In this example, two cylindrical tubes with their axes perpendicular to each other having an approach velocity of 35 m/s were made to impact each other and their general behavior was observed. Each tube was 150 mm long, 3 mm thick and has 100 mm diameter. Elastic-plastic with failure material model was chosen for both tubes and tube 1 was more coarsely meshed than tube 2. Tube 1 was given an initial velocity of 35m/s and a column of nodes in tube 2 were constrained in all directions. The geometry and the initial conditions of the tubes are shown in the Figure 5-17 and the finite element configuration is shown in Figure 5-18.

The results from the finite element simulations are shown in Figure 5-19. It shows comparison of effective von-mises stress between the single-surface contact algorithm using DYNA3D and the new contact algorithm at different stages of simulation. It can be seen that the deformed configurations from the two simulations are nearly identical. Maximum stress in each case reached 43.1 N/mm2.

www.manaraa.com

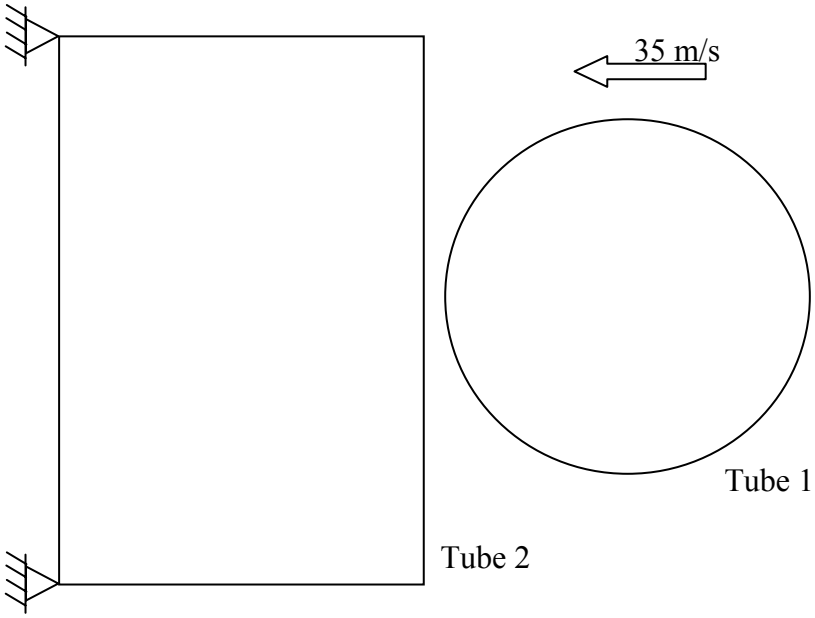


Figure 5-17 Contact-impact between two tubes -- Geometry and initial conditions

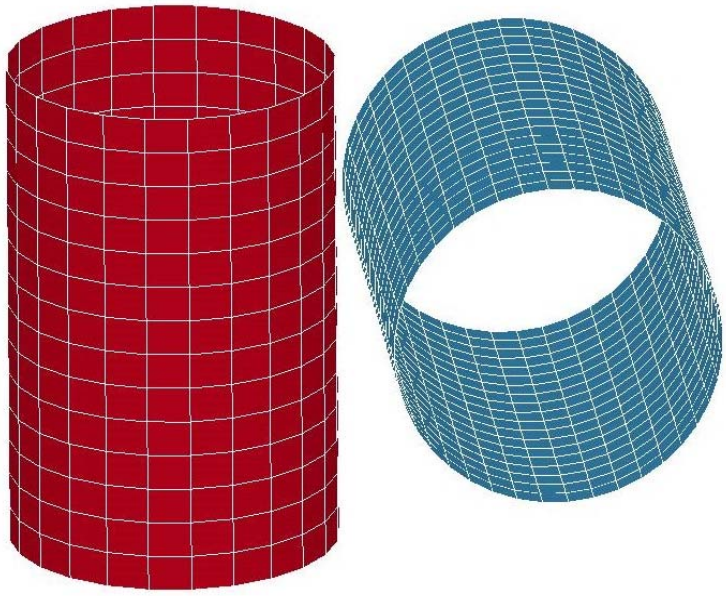
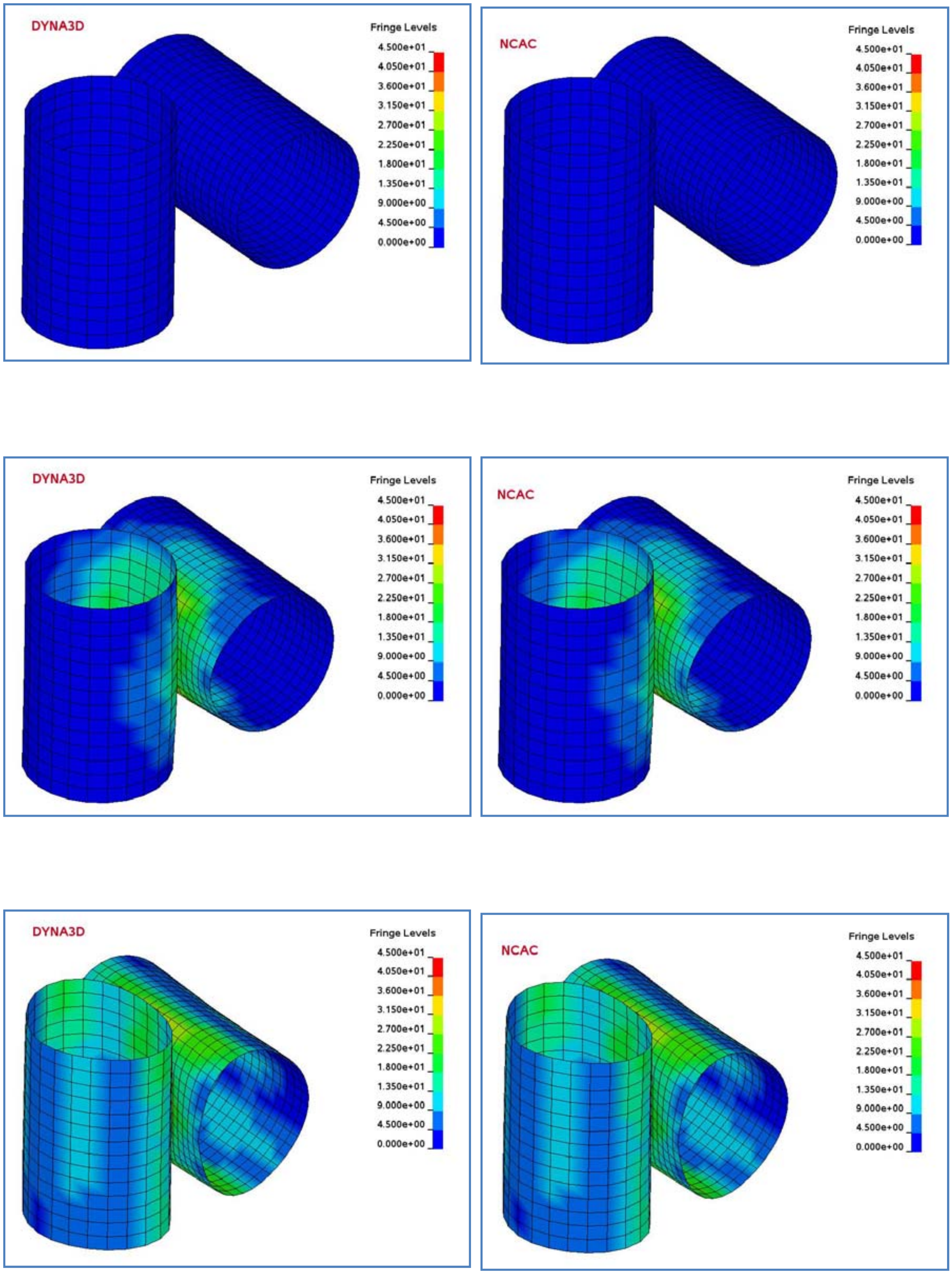


Figure 5-18 Contact-impact between two tubes -- Undeformed FE model

www.manaraa.com



www.manaraa.com

Continued…

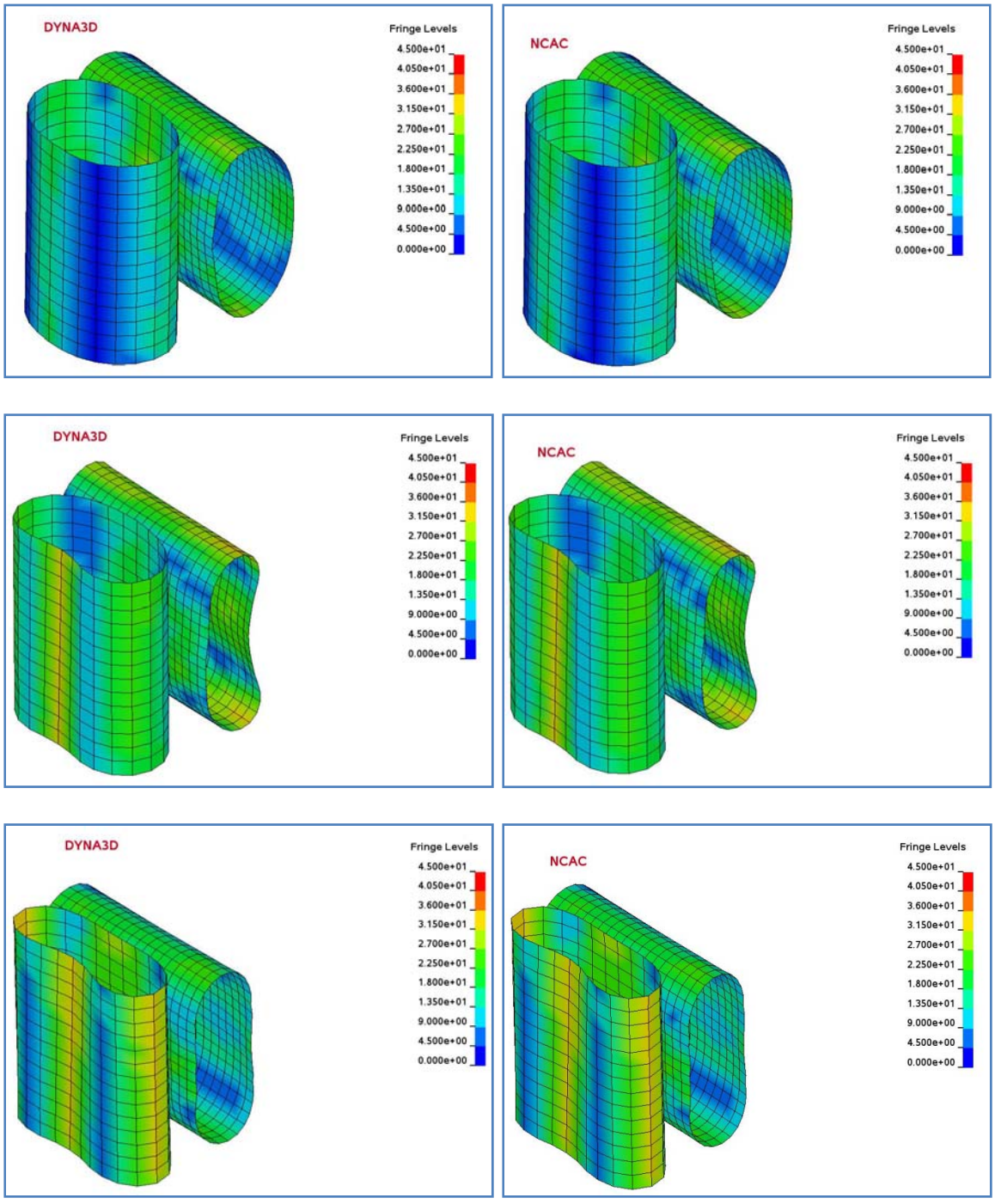


Figure 5-19 Contact-impact between two tubes -- Stress configuration

www.manaraa.com

5.2.2 Example 7 (Crushing symmetric tube between rigid walls)

In this example, the ability of the new contact algorithm to search penetration in a self-contact scenario is tested. A square tube with grooves and notches on its length is fixed at the bottom and crushed from the top using a rigid wall. Elements were assigned elastic-plastic with failure material model and 3 mm thickness. Figure 5-20 shows the initial configuration of the tube. Since the tube and loading conditions are axisymmetric, a 'quarter' model would give the same results as that of complete model.

Figure 5-21 shows comparison of stress states between the original single-surface contact algorithm and using the new contact algorithm at different stages of simulation. It can be noted from these results that the new contact algorithm is capable of searching self-contacts and provides accurate results.

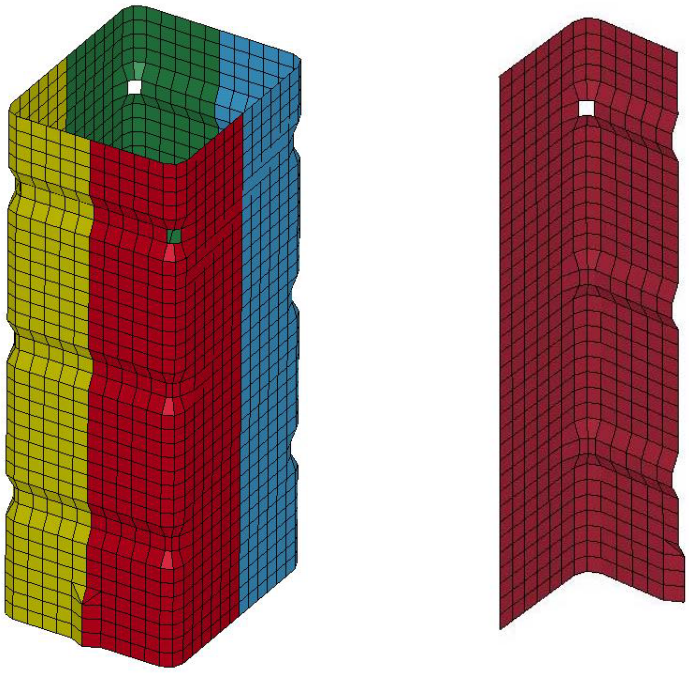
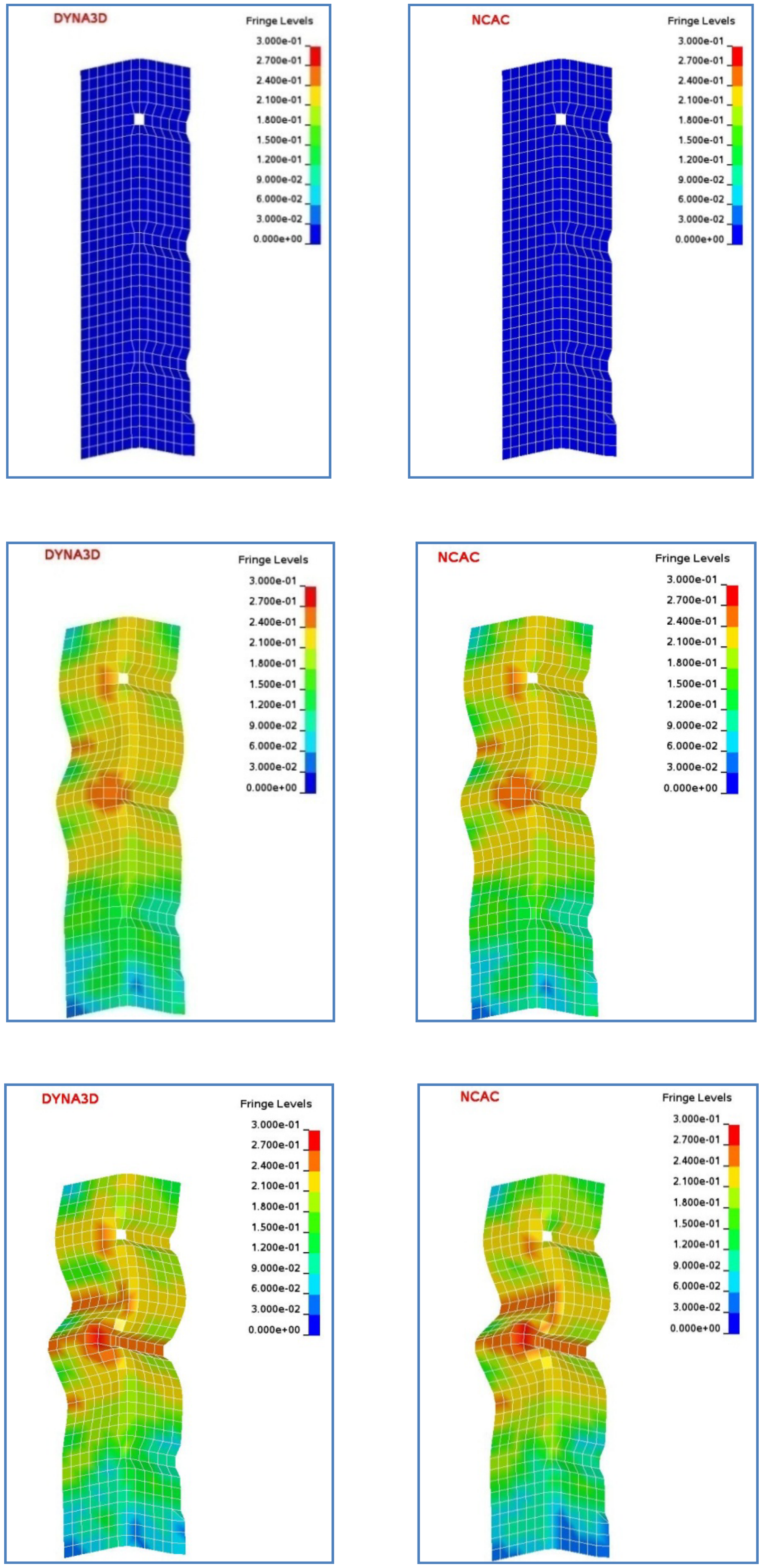


Figure 5-20 Symmetric tube crush -- Initial configuration of full and quarter model

www.manaraa.com



www.manaraa.com

Continued…

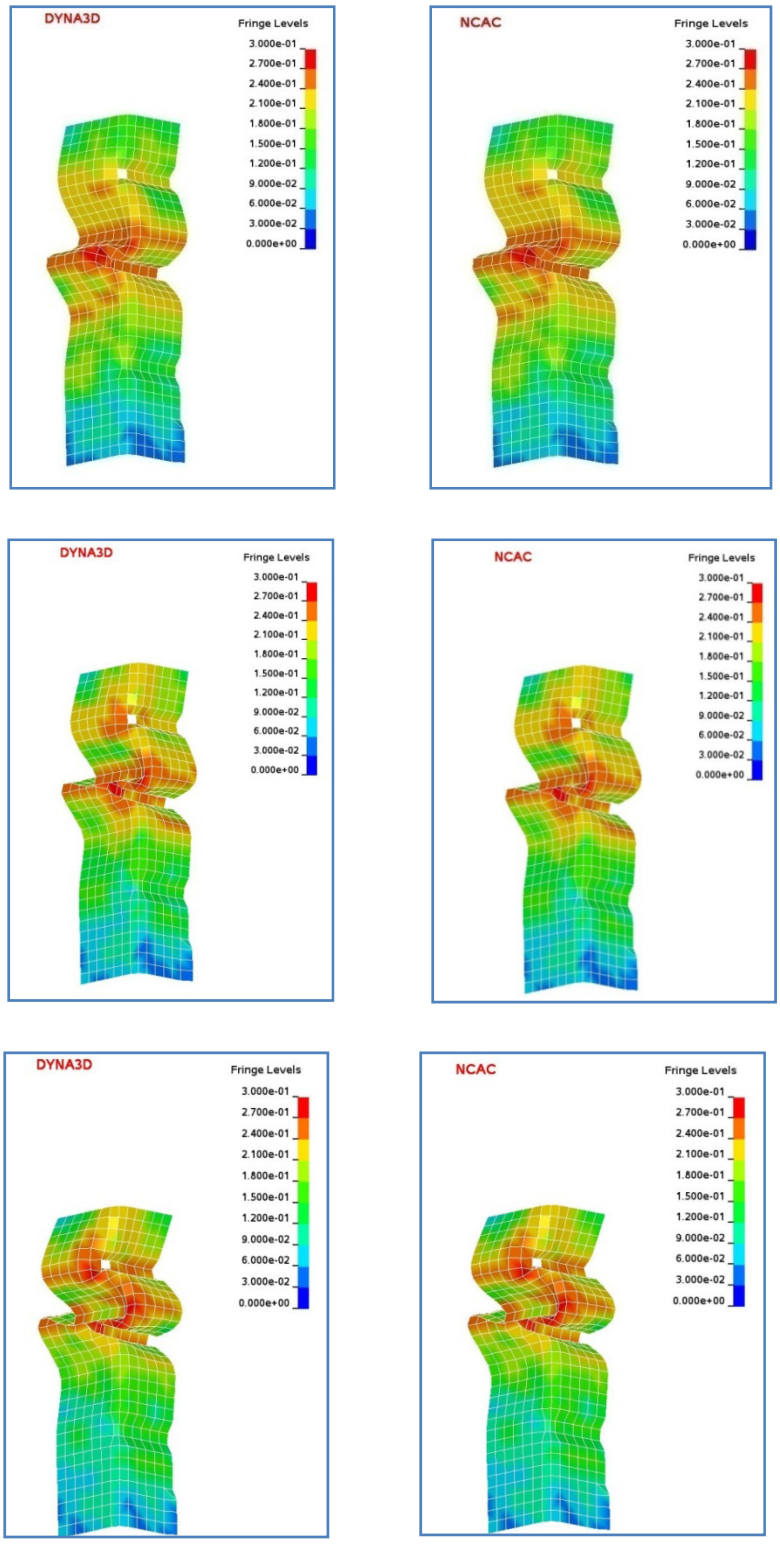


Figure 5-21 Symmetric tube crush -- Stress configuration

www.manaraa.com

5.2.3 Example 8 (Hertz contact problem)

The theory behind Hertz contact problem is explained in chapter 2. To simulate Hertz contact problem and validate the stresses computed by the code, a finite element model of a half-cylinder pressed against a flat base was created as shown in Figure 5-22. It is modeled as plane-stress problem with a thickness of 1 mm. Both parts, half-cylinder and flat base, were assigned elastic material properties. The half-cylinder was assigned an elastic modulus of the 3E5 MPa and a Poisson's ratio of 0.3. The flat base was assigned an elastic modulus of 1E5MPa and a Poisson's ratio of 0.33. A total load of 10 kN was distributed equally among the nodes on the top edge of cylinder. Nodes on the bottom edge of the flat base were constrained in all directions. Figure 5-22 shows the model setup and finite element mesh at beginning of the simulation.

The model was simulated using the current contact algorithms in DYNA3D and using the new contact algorithm. The results from the simulations were compared with theoretical values and are presented in the following section.

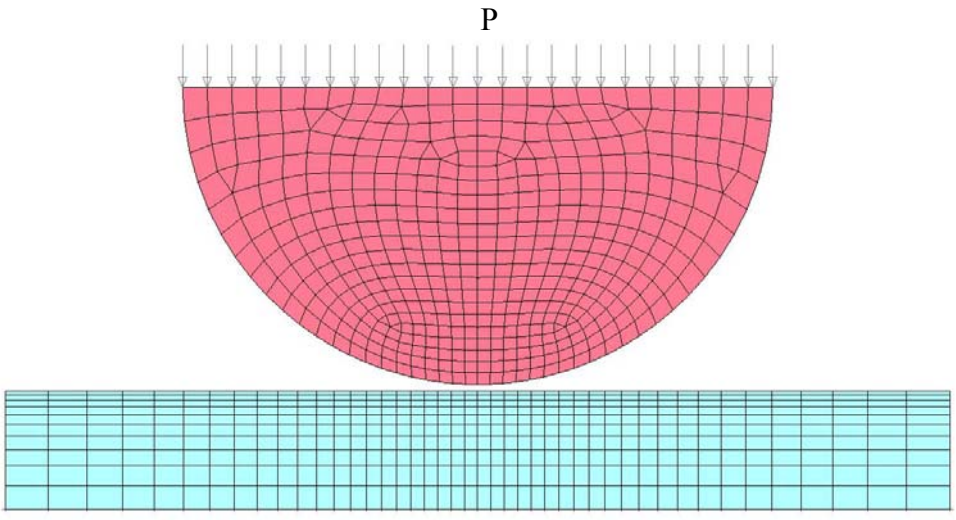


Figure 5-22 Finite element configuration of the Hertz contact problem

www.manaraa.com

Theoretical maximum compressive stress is given by

$$Max\ \sigma_c = 0.798\sqrt{\frac{P}{DE^*}} \tag{5.1}$$

where P is the applied load per unit lengths, D is diameter of the cylinder and E^* is the equivalent elastic modulus which is given by

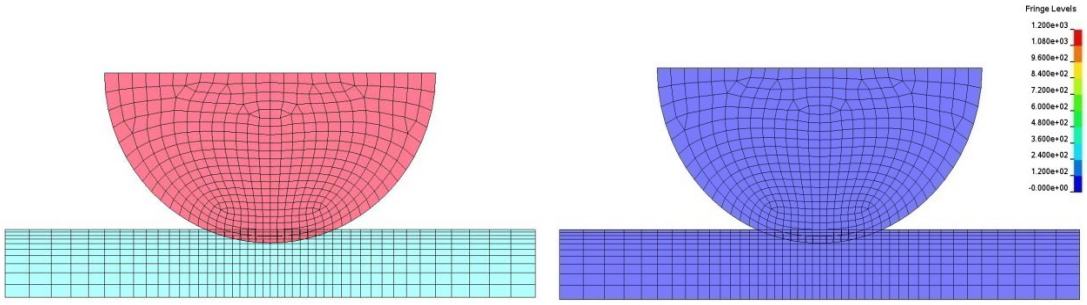
$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \tag{5.2}$$

Using the equations 5.1 and 5.2 and the assigned values, theoretical maximum compressive stress obtained is 10367.3 N/mm2. This theoretical value is compared to the simulation results check the validity of the contact algorithm.

5.2.3.1 Current Interfaces from DYNA3D

Hertz contact problem was simulated using DYNA3D's currently available contact algorithms. DYNA3D's 'single-surface' and 'nodes-to-surface' contacts failed to detect the penetration between the two parts. As no node is penetrating any surface, the contact interfaces fail. They also fail to check edge-to-edge penetrations since the elements are in the same plane. Figure 5-23 shows configuration and pressure distribution after few cycles of the simulation. Element stress is zero throughout the model indicating no contact force has been applied on any of the element.

www.manaraa.com



(a) FE configuration (b) Stress distribution

Figure 5-23 Using current contact interfaces

5.2.3.2 New contact algorithm using DYNA3D

Using the newly developed contact algorithm, the Hertz contact problem was simulated with three different mesh configurations. The element size and number of elements were varied in these configurations. In each case, maximum compressive stress is measured at the contact point and compared with the theoretical value. Figure 5-24 to 5-26 show the stress distribution in the three configurations after simulation reaches steady state. Table 5-1 shows the comparison and percentage error in calculating maximum compressive stress.

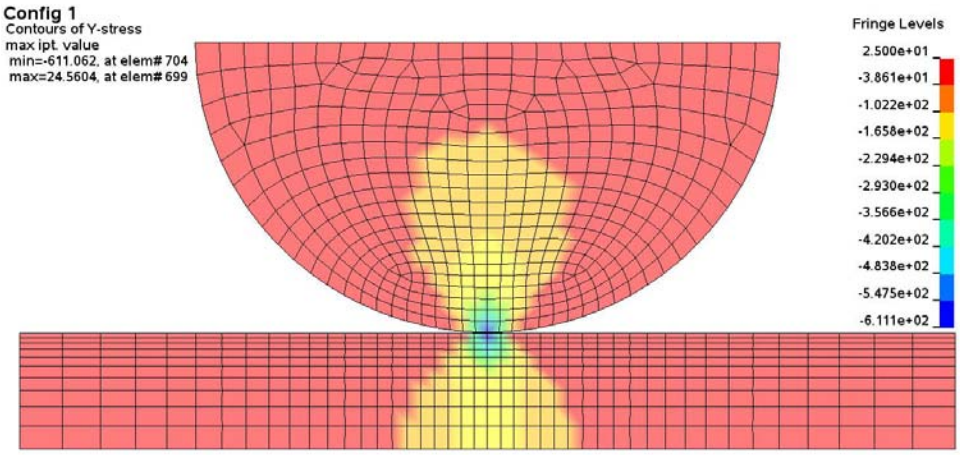


Figure 5-24 Stress distribution in Hertz contact problem, configuration-1

www.manaraa.com

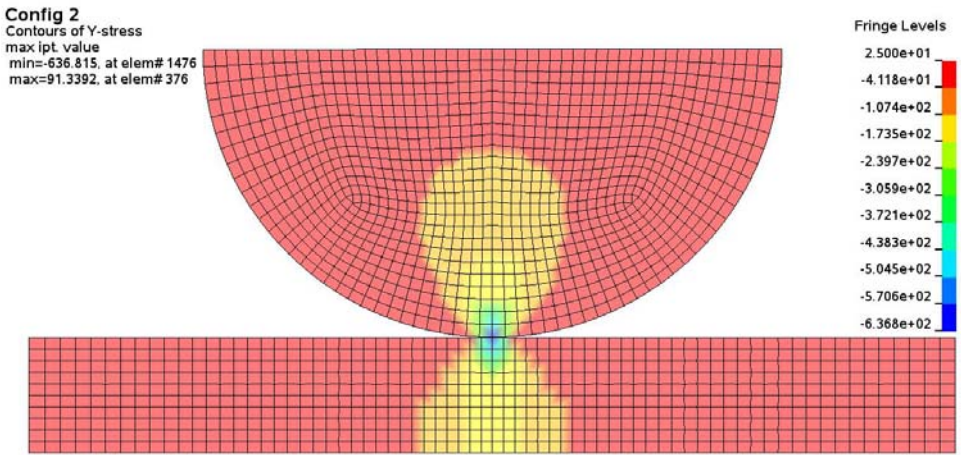


Figure 5-25 Stress distribution in Hertz contact problem, configuration-2

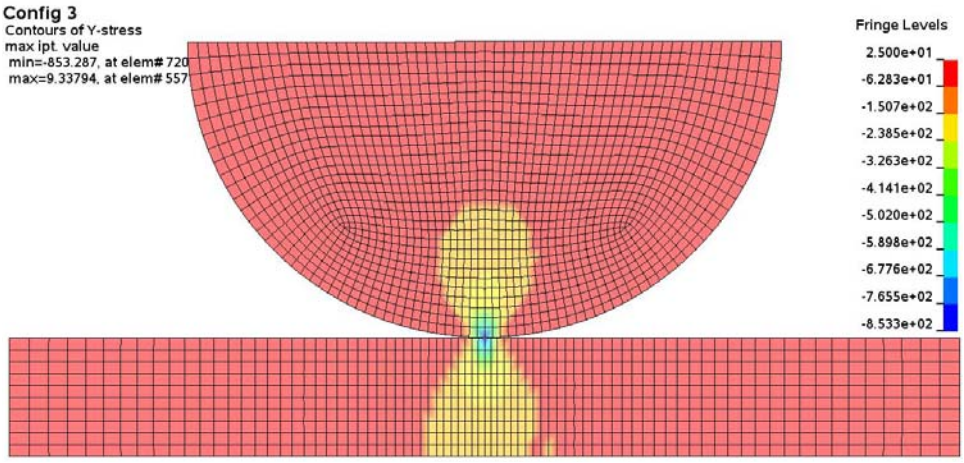


Figure 5-26 Stress distribution in Hertz contact problem, configuration-3

From Table 5-1, it can be seen that the difference between maximum compressive stress and the theoretical value decreases as the element size gets smaller. The difference could be for the reason that the stresses are computed at the integration points which are at the center of the elements, but the maximum stress will be at contact points. It can be shown that solution converges to the exact or theoretical value by making the element size infinitesimal. But due to the limitation on computation time, here in this research, only the trend towards convergence is shown.

www.manaraa.com

Table 5-1 Stress comparison in different configurations of Hertz contact problem

Configuration	Maximum compressive stress (MPa)	Difference from theoretical value
Theoretical	1036.73	--
Configuration-1	611.06	41.05 %
Configuration-2	636.82	38.56 %
Configuration-3	845.28	18.46 %

5.2.4 Example 9 (Contact patch test)

The contact patch test is used to assess stability and consistency of the contact algorithm by simulating a simple problem. In this problem, two rectangular plates that are on same plane are made to contact each other on their edges. Two edges of the bottom plate and one edge of the top plate are constrained as shown in Figure 5-27. A uniformly distributed varying load was applied on the top edge of the top plate. Keeping the total applied load constant and varying the mesh densities, three different configurations of the problem were simulated. Maximum stress and stress variation at the contact surface were monitored to see whether the contact will induce fictitious localized stresses.

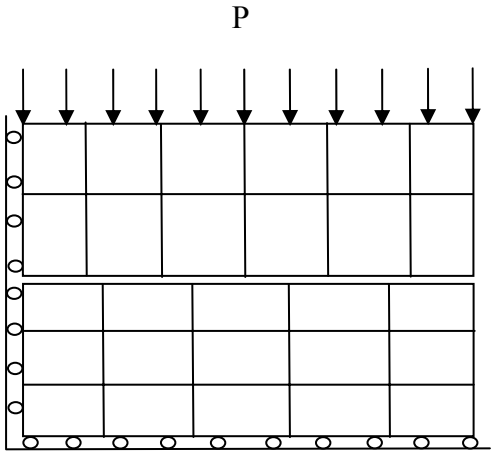
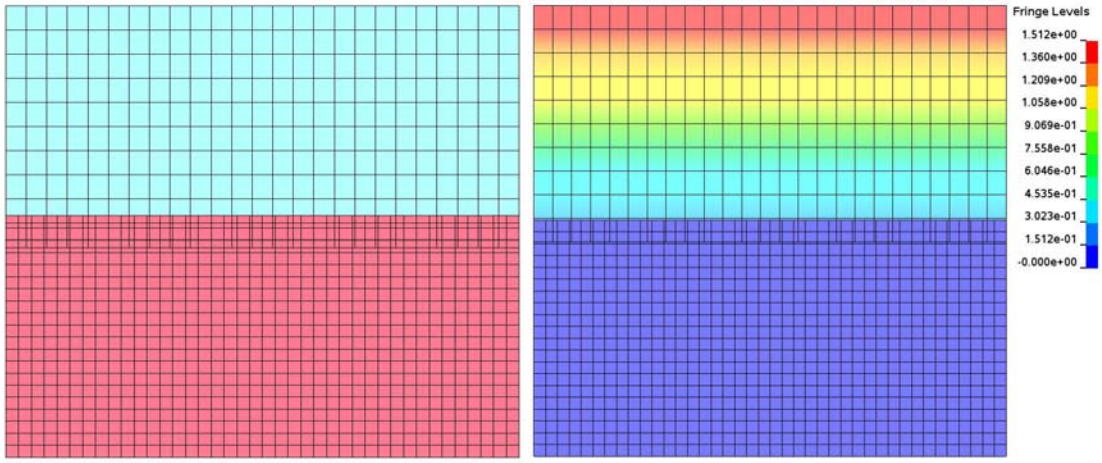


Figure 5-27 Simple contact patch test problem

www.manaraa.com

5.2.4.1 Current Interfaces from DYNA3D

Contact patch problem was simulated using DYNA3D's currently available contact interfaces. Similar to the case of Hertz contact simulation, in the contact patch simulation, DYNA3D's 'single-surface' and 'nodes-to-surface' contacts failed to detect penetration. The elements traverse each other without contact force being applied on any of the elements. Figure 5-28 shows the mesh configuration and pressure distribution after few cycles. It can be seen from the stress distribution plot that the elements from bottom plate do not experience any stress. Variation in stress in the top plate is due to varying point load applied on the top edge.



(a) Mesh configuration (b) Stress distribution

Figure 5-28 Using current contact interfaces in DYNA3D

5.2.4.2 New contact algorithm using DYNA3D

Three different mesh configurations of the contact patch problem were simulated using the new contact algorithm. For these configurations, the stress distribution along the contact surface and displacements of nodes along the contact surface are measured.

www.manaraa.com

Figure 5-29 through 5-11 show stress distribution and displacement of nodes along the contact surface.

From the figures it can be seen that a minimal variation of stresses along the contact surface is observed in all configurations. The stress level at locations where the nodes from the two plates vertically coincide are slightly but not significantly higher compared to other locations. From the figures, the displacement of the nodes along the contact surface is uniform. A uniform distance between the plates is maintained throughout the simulation once the plates come in contact. Contact forces applied on the nodes are accurate to maintain the correct distance between the two plates. Figure 5-35 shows the distance between the two plates at various locations for configuration-1, the nodes being chosen randomly along the length.

Maximum stress seen in the elements varied with the mesh configuration. Table 5-2 shows maximum Von-Mises stress seen in different configurations of the model.

Table 5-2 Maximum Von-Mises Stress from three different configurations

	Configuration-1 N/mm^2	Configuration-2 N/mm^2	Configuration-3 N/mm^2
Maximum V-M stress	103.74	121.36	231.61

www.manaraa.com

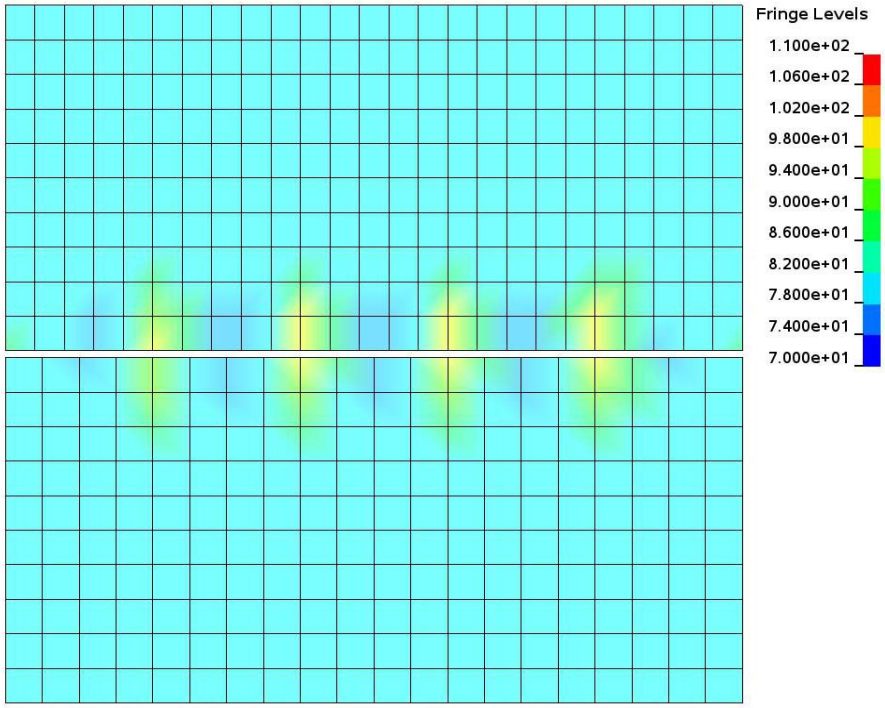


Figure 5-29 Stress (V-M) distribution in configuration-1 using new contact algorithm

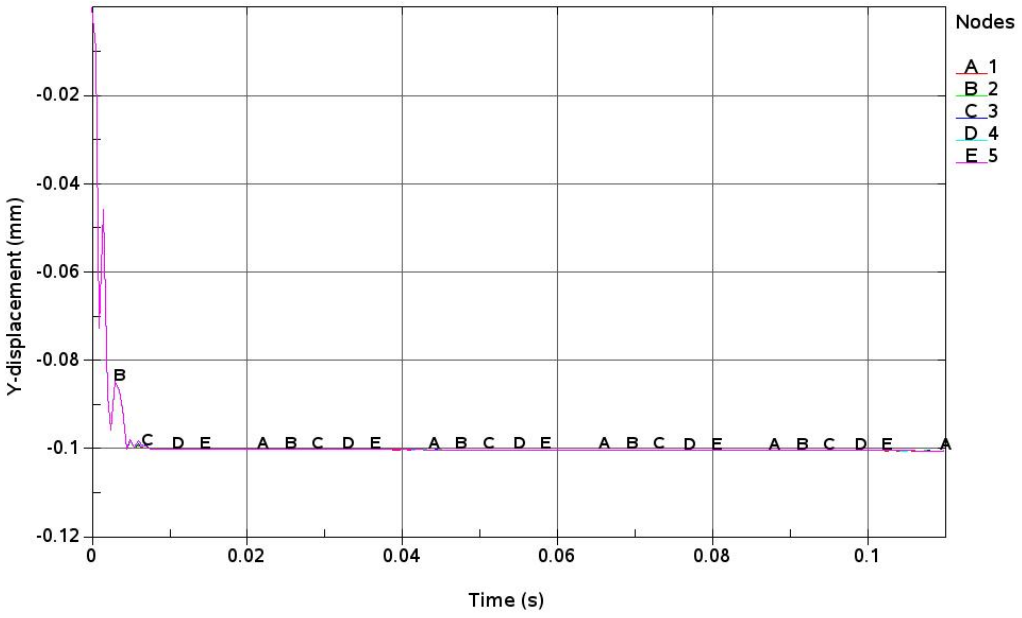


Figure 5-30 Nodal displacements along contact surface, new contact algorithm, cofiguration-1

www.manaraa.com

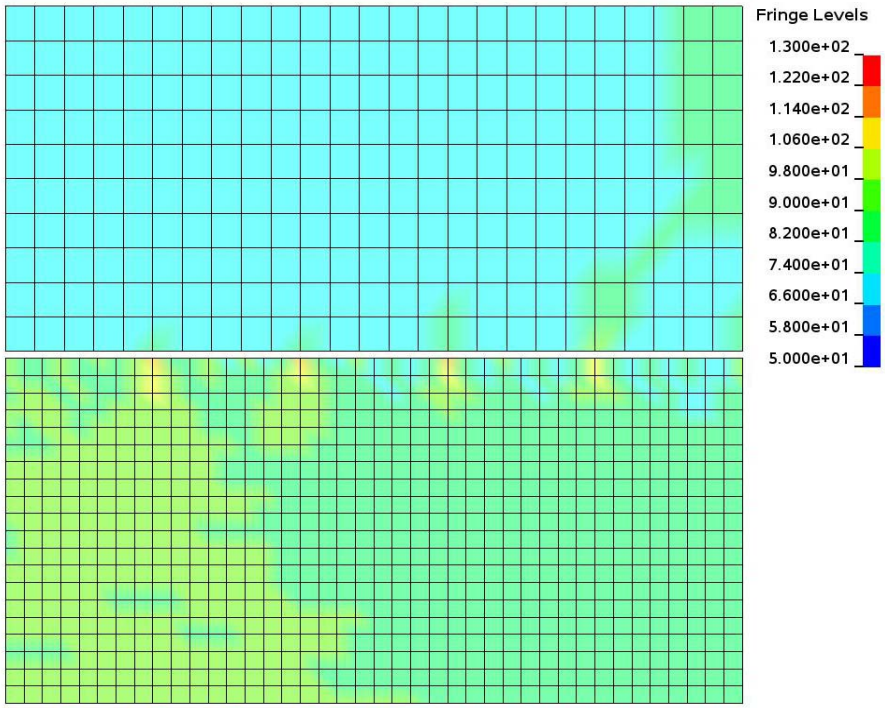


Figure 5-31 Stress (V-M) distribution in configuration-2 using new contact algorithm

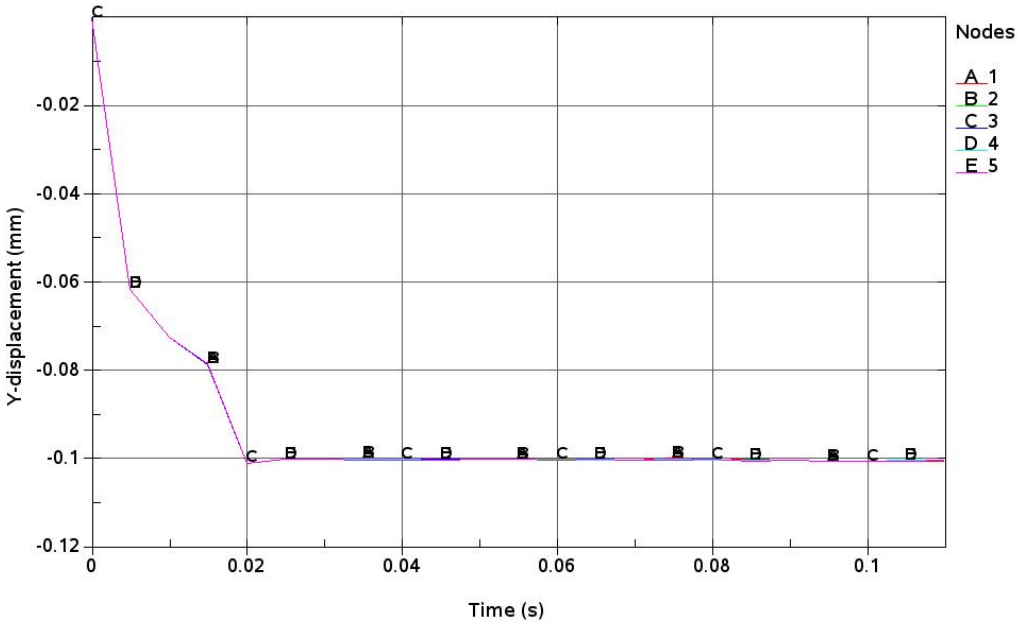


Figure 5-32 Nodal displacements along contact surface, new contact algorithm, configuration-2

www.manaraa.com

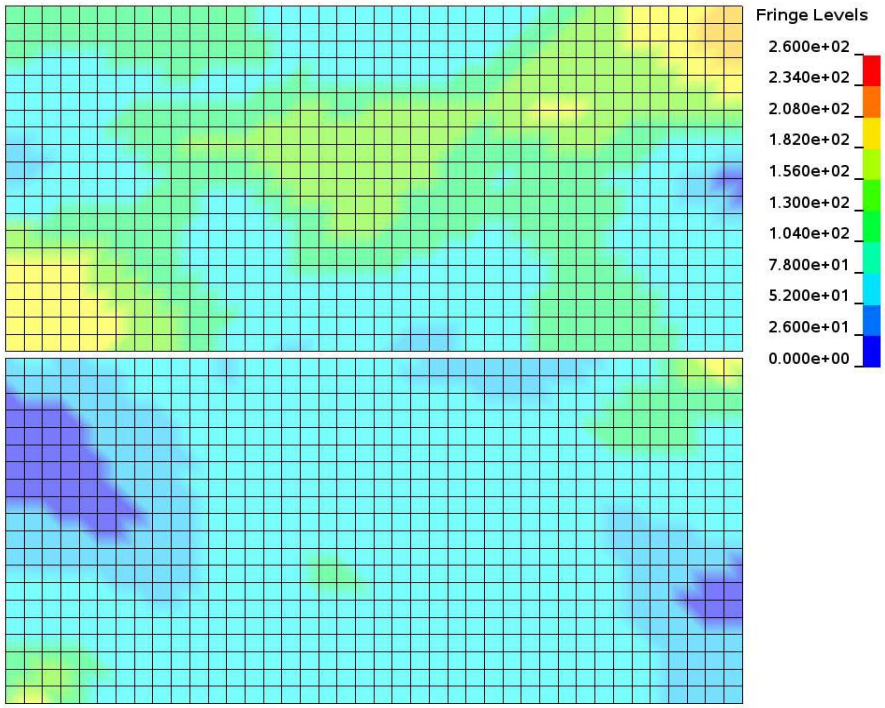


Figure 5-33 Stress (V-M) distribution in configuration-3 using new contact algorithm

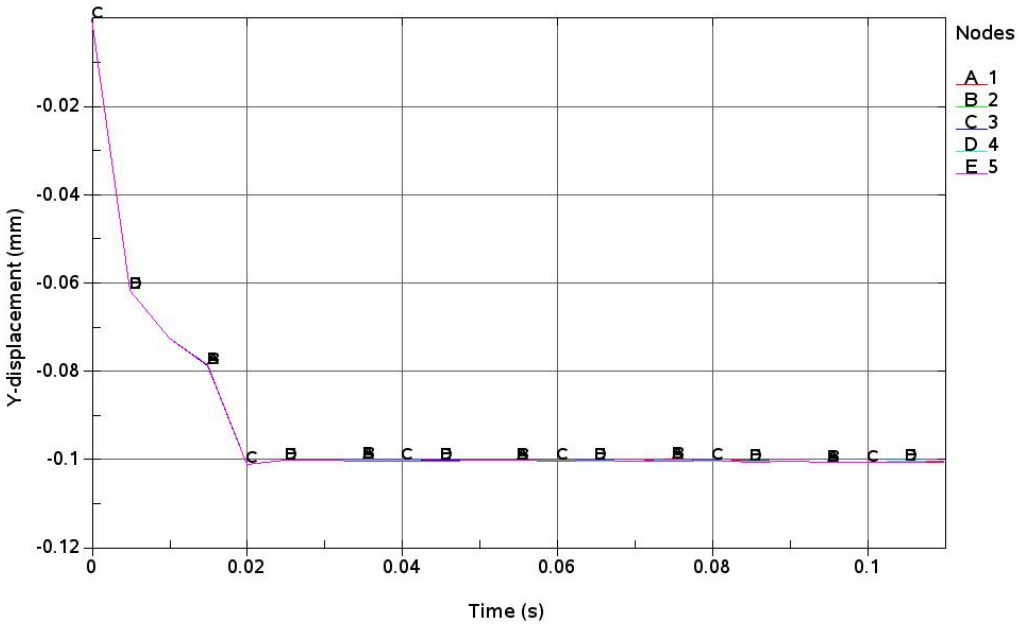


Figure 5-34 Nodal displacements along contact surface, new contact algorithm, configuration-3

www.manaraa.com

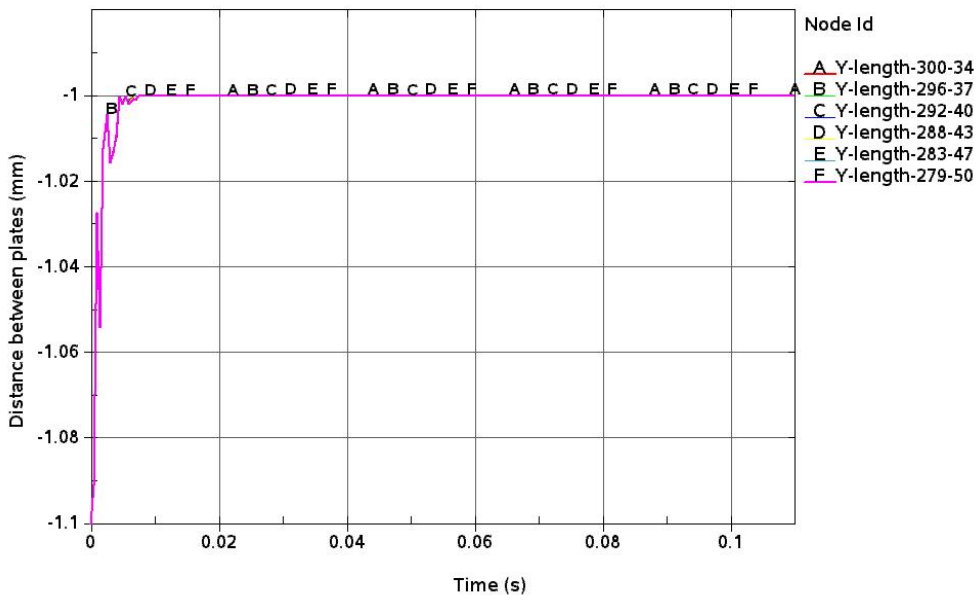


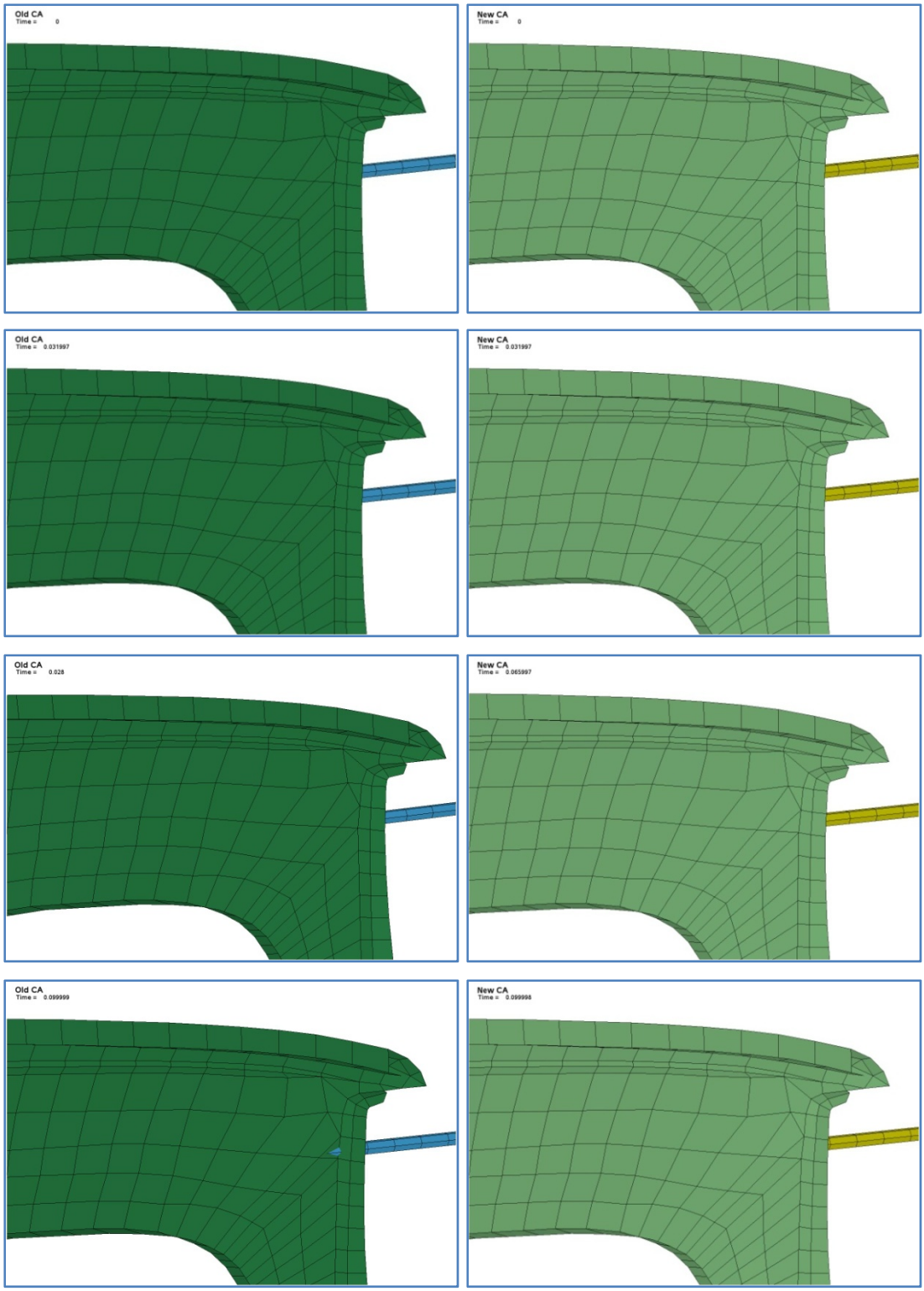
Figure 5-35 Distance between two plates along the contact surface, configuration-1

5.3 Application problems

To show the improvements in the new contact algorithm over existing contact algorithms in DYNA3D while solving real world problems, two examples are simulated and the results are shown in this section. In the first example, an impact between a cable guardrail and front fender of a C2500 pickup truck is simulated. Snap shots from this simulation at various times are presented in Figure 5-36. In the second example, an impact between a portable concrete barrier and bumper of a C2500 pickup truck is simulated. Figure 5-37 shows the snap shots from the simulation of bumper and barrier impact at various times. In Figures 5-36 and 5-37, results from the simulation using current algorithm are shown in left column and results from the new contact algorithm are presented in right column. In both the examples, it can be seen that the new contact algorithm prevents the penetration and provides more accurate results than the current algorithm.

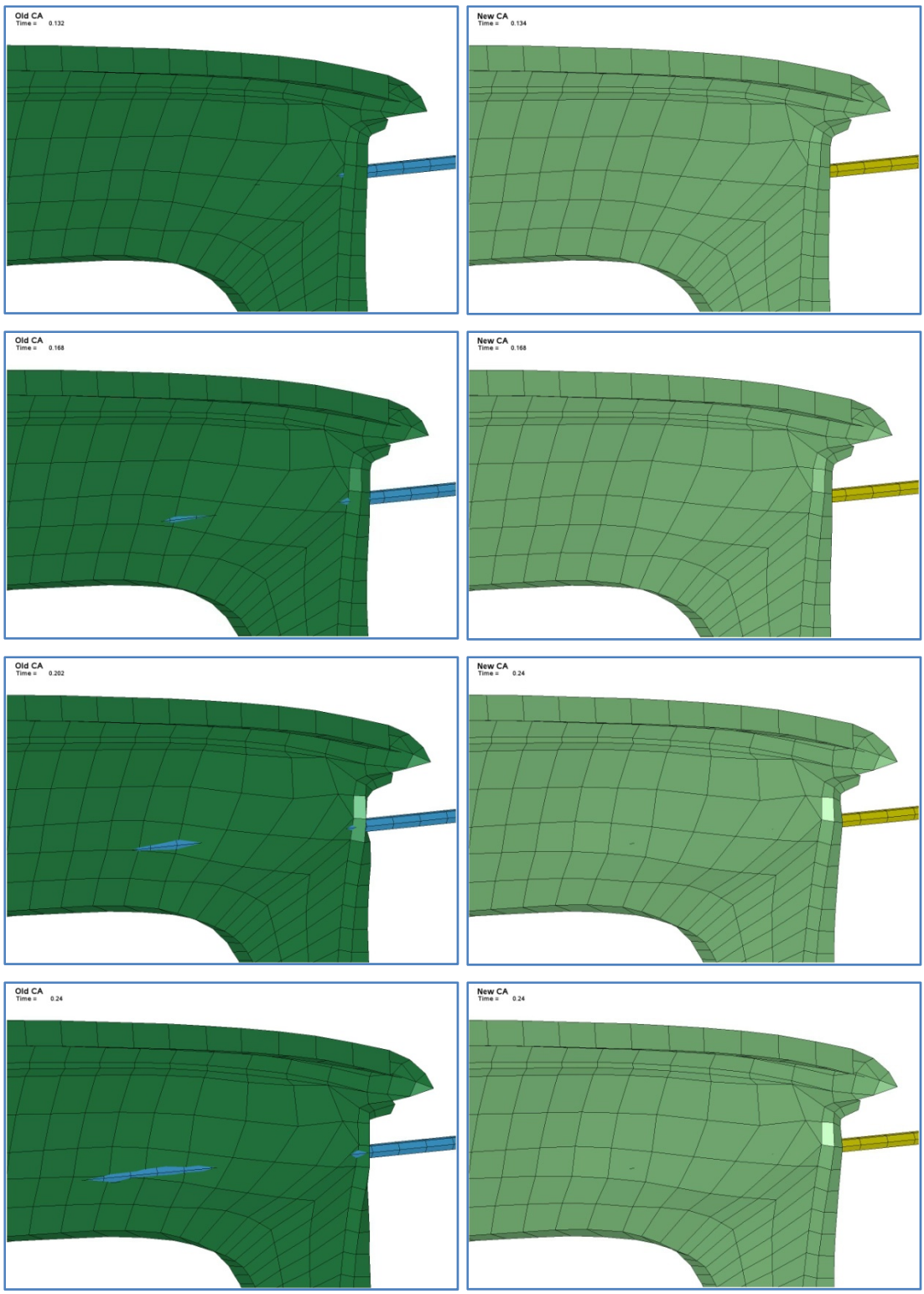
www.manaraa.com

5.3.1 Example 1 (Impact between fender and cable guardrail)



Continued…

www.manaraa.com

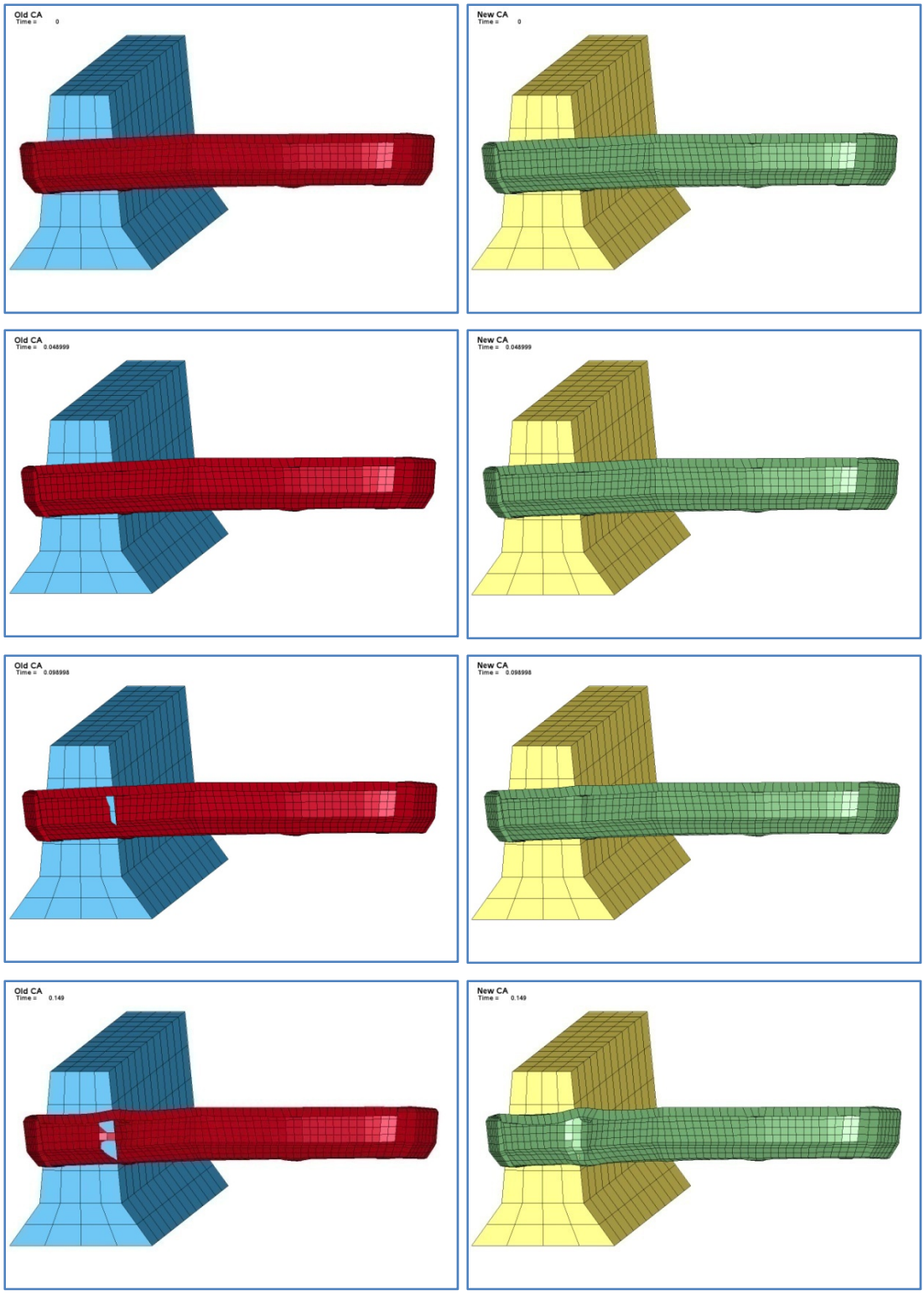


Current contact algorithm New contact algorithm

Figure 5-36 Contact between front fender of C2500 and cable guardrail

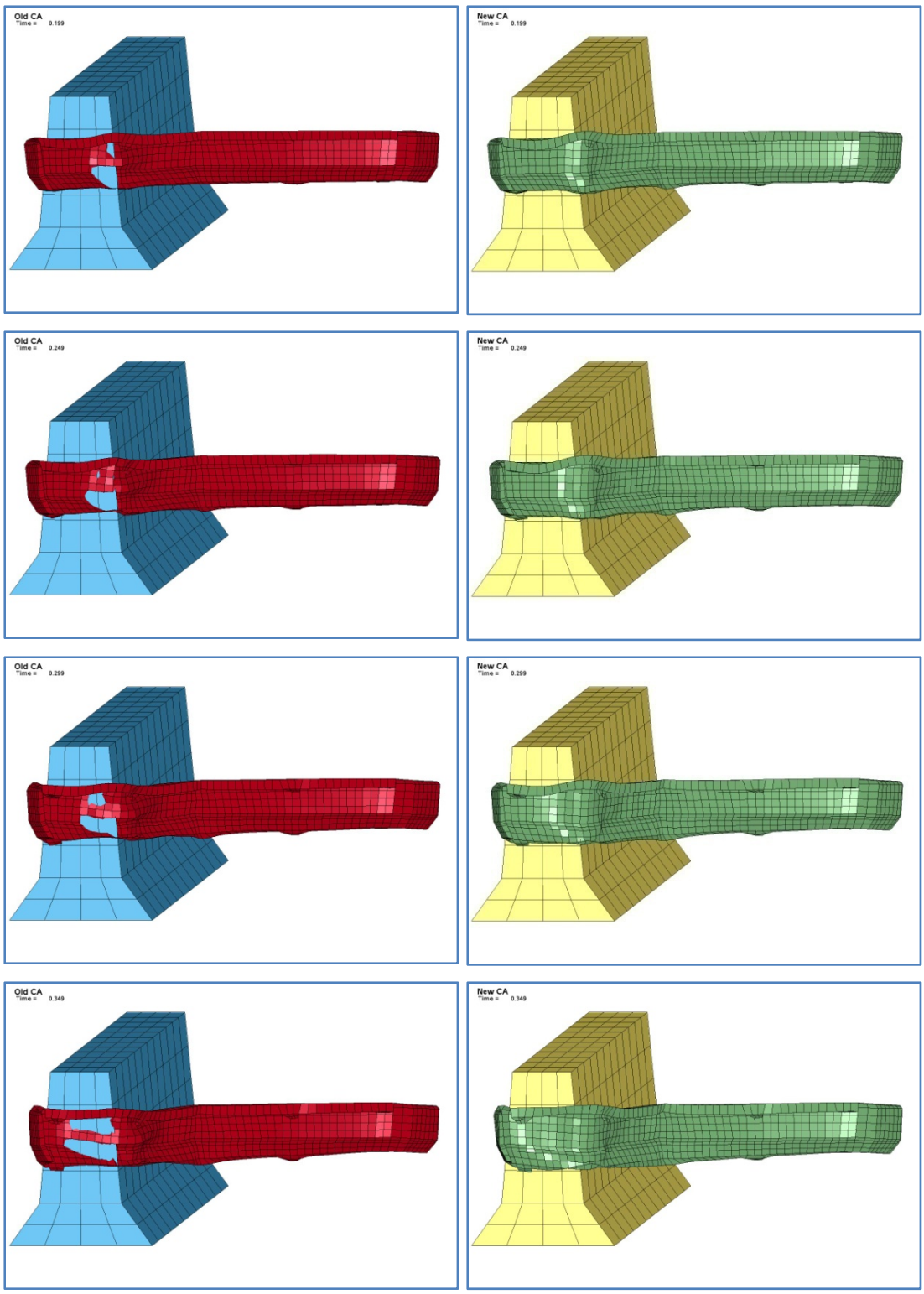
www.manaraa.com

5.3.2 Example 2 (Impact between bumper and concrete barrier)



Continued…

www.manaraa.com



Current contact algorithm New contact algorithm

Figure 5-37 Contact between bumper of C2500 and portable concrete barrier

www.manaraa.com

6. CONCLUSIONS AND RECOMMENDATIONS

Approximately half of all the numerical problems that are encountered during a finite element simulation are from contact interfaces. Depending on the complexity and number of parts and elements in the model, about half of the computation time is spent on contact interfaces. Hence, any error that surfaces due to contact interface algorithms prolongs total analysis time.

Current contact algorithms that are used in explicit finite element codes have come a long way in improving their accuracy and efficiency. The majority of the contact search algorithms that are used in popular FEM crash codes use the concept of 'preventing slave-nodes from penetrating master-surface'. Using this concept contact algorithms detect and remove most of the possible penetrations that occur, however there is still room for improvement when it comes to edge-to-edge penetration check and contact force computation. With the use of nodal base projection to offset the element thickness, some of the edge-to-edge penetration problems have been overcome. To make the contact algorithms more accurate in detecting penetrations than what they are today, there is a need for new contact search algorithm. The new contact algorithm should be able to detect penetrations at all times and apply just enough force to remove those penetrations.

In this research, a new contact algorithm that includes a new global search method and a new local search method has been proposed. Emphasis is given to accuracy in detecting penetrations at all times and applying the right magnitude and direction of force

www.manaraa.com

at correct locations. A new global search method which uses the concept of enclosing spheres around nodes combined with bucket sorting has been proposed. Using this method all possible combinations of 'slave-master' element pairs which are in contact or might come in contact over the next few cycles are identified. The sphere enclosing each node has a radius large enough to enclose all the elements that are connected to the node. The bucket sorting used in the new global search checks for intersections of spheres with the buckets rather than presence of nodes in the bucket that is used in current algorithms. Unlike current algorithms which use slave-node—master-segment pair and allocates, in a cycle, only one master segment for a slave node, the new algorithm use slave-master element pair. From this treatment, if a node is penetrating two or more elements simultaneously, every penetration is identified.

The proposed new local search method assumes uniform thickness across each element. The geometric surface of the beam elements was considered to be combination of a cylinder and two spheres whose radii are equal to the radius of the beam. The geometric surface of shell elements has half-cylinders at the edges and spheres at the corners whose diameters are equal to the thickness of the element. With this consideration, the geometry of the contact surface is interpreted accurately and problems associated in finding penetration in a skewed mesh are eliminated.

Constant stiffness that is used in computing contact force in current contact algorithms is replaced by exponentially varying stiffness in the new contact algorithm. When compared to the constant stiffness, the varying stiffness applies significantly higher forces when the penetration becomes large. With this approach the nodes are prevented from passing through the element in cases where the inertia or loads are high. By varying

www.manaraa.com

the stiffness, penetrations are strictly removed and accurate distance between contact surfaces is maintained.

The new contact algorithm has been implemented in DYNA3D and validated. To validate the algorithm, Hertz contact problem and contact patch test were used. In the Hertz contact problem, three different mesh size configurations were simulated and the results were compared to the theoretical value. It is shown that with the new contact algorithm, the maximum stress in Hertz contact problem tends to converge towards exact solution or theoretical value when the element size becomes smaller. In contact patch test, a simple problem is simulated with three different mesh configurations and results are compared with the results from DYNA3D. It is shown that the new contact algorithm accurately computes and maintains the distance between contact surfaces with an acceptable variation in the stress along the contact surface.

Shortcomings and Recommendations for Future Research

The validation tests and example problems show that the new contact algorithm detects penetration at all times and applies the right magnitude of contact forces on the nodes to remove the penetration. However, it has few exclusions and shortcomings that need to be addressed before it can be used to solve general non-linear dynamic problems.

Solid elements and surfaces are not considered in the contact search. Even though it is a simple task of including the free surfaces of solid elements, the algorithm should be modified to consider zero thickness of the solid segments and also to treat their edges and corners differently.

www.manaraa.com

The new algorithm does not delete failed elements that are in the contact. Provisions should be made in the algorithm such that the users have an option of deleting the elements from contact once they fail due to certain failure criteria.

Initial penetrations are not taken into consideration in the new contact algorithm. If initial penetrations are present in the model at the beginning of the simulation, the simulation will crash due to high contact forces. Care should be taken to remove all initial penetrations before using the new contact algorithm.

Every quadrilateral element is treated as two triangular elements while checking for penetrations. If an element is severely warped, as shown in Figure 6-1, the geometric surface that is seen by the contact algorithm varies depending on what order the element is split into triangles. A different three-node combination results in a different contact surface which in turn results in different end results. Care should be taken to avoid defining warped elements.

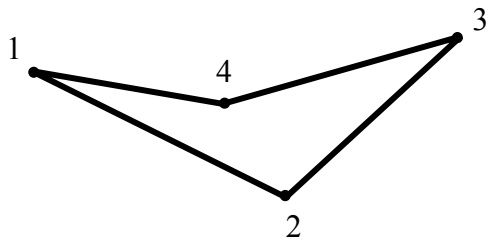


Figure 6-1 Severely warped element

The new contact algorithm considers only uniform circular cross-section for the beam elements. Other cross-sections such as rectangular and varying cross-section should be included to make the new contact algorithm more versatile.

www.manaraa.com

Lastly, in this research, little focus was on the make the algorithm more efficient. As a result, the computation time required to simulate problems using new contact algorithm is significantly higher than the time required using current algorithms. The efficiency of the contact algorithm can be improved by optimizing the code. Future work should include optimizing the new algorithm to make it work seamlessly with rest of the DYNA3D code.

www.manaraa.com

REFERENCES

1. **Whirley, Robert G and Egelmann, Bruce E.** *DYNA3D A Nonlinear, Explicit, Three-Dimensional Finite Element Code For Solid and Structural Mechanics-User Manual.* s.l. : Lawrence Livermore National Laboratory, 1993. UCRL-MA-107254 Rev. 1.

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

3. *Three-dimensional penetration computation.* **Belytschko, T, Kennedy, J M and Lin, J I.** Lausanne, Switzerland : s.n., August 17-21, 1987, Transactions of the 9th International Conference on Structural Mechanics in Reactor Technology, pp. 83-88.

4. *A single surface contact algorithm for the postbuckling analysis of shell structures.* **Benson, D J and Hallquist, J O.** 2, 1990, Computer Methods in Applied Mechanics and Engineering, Vol. 78, pp. 141-163.

5. **Hallquist, J O.** *LS-DYNA Theoritical Manual.* s.l. : Livermore Software Technology Corporation, 2005.

6. *A simple algorithm for three-dimensional finite element analysis of contact problems.* **Papadopoulos, P and Taylor, R L.** 1993, Computers and Structures, Vol. 46, pp. 1107-1118.

www.manaraa.com

7. *A unified contact algorithm based on the territory concept.* **Zhong, Z H and Nilsson, L.** 1996, Computer Methods in Applied Mechanics and Engineering, Vol. 130, pp. 1-16.

8. *The position code algorithm for contact searching.* **Oldenburg, M and Nilsson, L.** 1994, International Journal for Numerical Methods in Engineering, Vol. 37, pp. 359-386.

9. *The vectorized pinball contact impact routine.* **Belytschko, T and Neal, M O.** Anaheim, CA : s.n., 1989, Transactions of the 10th International Conference on Structural Mechanics in Reactor Technology, Vol. B, pp. 161-166.

10. *Sliding interfaces with contact-impact in large-scale Lagrangian computations.* **Hallquist, J O, Goudreau, G L and Benson, D J.** 1985, Computer Methods in Applied Mechanics and Engineering, Vol. 51, pp. 107-137.

11. *Contact-impact by the pinball algorithm with penalty and Lagrangian methods.* **Belytschko, T and Neal, M O.** 1991, International Journal for Numerical Methods in Engineering, Vol. 31, pp. 547-572.

12. *Inside-Outside contact search algorithm for finite element analysis.* **Wang, S P and Nakamachi, E.** 1997, International Journal for Numerical Methods in Engineering, Vol. 40, pp. 3665-3685.

13. *FFS contact searching algorithm for dynamic finite element analysis.* **Wang, F, Cheng, J and Yao, Z.** 2001, International Journal for Numerical Methods in Engineering, Vol. 40, pp. 655-672.

www.manaraa.com

14. *NBS contact detection algorithm for bodies of similar size.* **Munjiza, A and Andrews, K R F.** 1998, International Journal for Numerical Methods in Engineering, Vol. 43, pp. 131-149.

15. *Using space filling curves for efficient contact searching.* **Diekmann, R, et al.** 16th IMACS World Congress.

16. *Efficient contact search for finite element analysis.* **Diekmann, R, et al.** 2000, European Congress on Computational Methods in Appied Sciences and Engineering.

17. *A Pinball method by direct localization of the impact area.* **Petkevicius, K, Kulak, R and Marchertas, A.** 2003, Transactions of the 17th International Conference on Structural Mechanics in Reactor Technology, Vols. J04-1.

18. *The splitting pinball method for contact-impact problems.* **Belytschko, T and Yeh, L S.** 3, 1993, Computer Methods in Applied Mechanics and Engineering, Vol. 105, pp. 375-393.

19. *A parallel finite element contact/impact algorithm for non-linear explicit transient analysis: Part I -- The search algorithm and contact mechanics.* **Malone, J and Johnson, N.** 1994, International Journal for Numerical Methods in Engineering, Vol. 37, pp. 559-590.

20. *A Perturbed Lagrangian formulation for the finite element solution of contact problems.* **Simo, J C, Wriggers, P and Taylor, R L.** 1985, Computer Methods in Applied Mechanics and Engineering, Vol. 50, pp. 163-180.

www.manaraa.com

21. *Contact Modeling -- Forces.* **Adams, G G and Nosonovsky, M.** 2000, Tribology International, Vol. 33, pp. 431-442.

22. **Timoshenko, S P and Goodier, J N.** *Theory of Elasticity.* s.l. : McGraw-Hill Inc, 1970.

23. *Uber die beruhurng fester elastischer korper (On the contact of elastic solids).* **Hertz, H.** 1882, J reine und angewandte Mathemacik, pp. 94-156.

24. *Stability and patch test performance of contact discretization and a new solution algorithm.* **El-Abbasi, N and Bathe, K-J.** 2001, Computers and Structures, Vol. 79, pp. 1473-1486.

25. *Mesh matching and contact patch test.* **Tan, D.** s.l. : Springer-Verlag, 2001, Computational Mechanics, pp. 135-152.

26. *On a patch test for contact problems in two dimensions.* **Taylor, R L and Papodopoulos, P.** [ed.] P Wriggers and W Wanger. Berlin : Springer-Verlag, 1991, Computational Methods in Nonlinear Mechanics, pp. 690-702.

27. *The patch test as a validation of a new finite element for the solution of convection-diffusion equations.* **Sacco, R, Gatti, E and Gotusso, L.** s.l. : Computer Methods in Applied Mechanics and Engineering, 1995, Vol. 124, pp. 113-124.

28. **Fortin, M and Glowinsky, R.** *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems.* Amsterdam-New York : North-Holland Publ. Co., 1983.

www.manaraa.com

29. **Bathe, K-J.** *Finite Element Procedures.* NJ : Prentice-Hall: Englewood Cliffs, 1996.

30. **Belytschko, T, Liu, W K and Moran, B.** *Nonlinear Finite Elements for Continua and Strucutres.* s.l. : John Wiley & Sons, 2000.

www.manaraa.com