SAE TECHNICAL PAPER SERIES

1999-01-0104

The DyMesh Method for Three-Dimensional Multi-Vehicle Collision Simulation

Allen R. York
A.R. York Engineering, Inc

Terry D. Day
Engineering Dynamics Corp.

Reprinted From: Accident Reconstruction: Technology and Animation IX (SP-1407)

SAE The Engineering Society For Advancing Mobility Land Sea Air and Space® INTERNATIONAL

International Congress and Exposition
Detroit, Michigan
March 1-4, 1999

ISSN 0148-7191

Printed in USA

1999-01-0104

The DyMesh Method for Three-Dimensional Multi-Vehicle Collision Simulation

Allen R. York
A.R. York Engineering, Inc

Terry D. Day
Engineering Dynamics Corp.

ABSTRACT

Two-dimensional collision simulation has been used successfully for two decades. Two- and three-dimensional momentum methods are also well known. Three-dimensional collision simulation can be accomplished using finite element methods, but this is not practical for interactive collision simulation due to long mesh generation times and run times which may take several days. This paper presents an approach to collision simulation using a new algorithm to track interacting vehicle surface meshes. Three-dimensional forces due to vehicle crush are taken into account during the solution and the damage profile is visualized at run time. The new collision algorithm is portable in that it takes as input vehicle material properties and surface geometries and calculates from their interaction three-dimensional forces and moments at the vehicle center of gravity. Intervehicle mesh forces may be calculated from a user-defined force-deflection relationship. The derivation is discussed. The paper includes examples using arbitrarily shaped three-dimensional bodies as a proof of concept. Samples are also included using three-dimensional vehicle meshes. The simulations are shown to agree favorably with theory, test, and finite element results.

INTRODUCTION

MOTOR VEHICLE CRASHES are the leading cause of death for persons below the age of 40. In the U.S., 370,000 people have died on the nations highways since 1990. Another 30 million have been injured.[1][1] The cost, in both real dollars and human suffering, is enormous. Clearly, because of its impact on our society, it is important that we focus on improving our understanding of the cause of highway crashes.

Engineers and physicists have understood the mechanics of vehicle collisions for several decades. Until the mid-seventies, collisions were analyzed by impulse-momentum methods using a slide rule or hand-held calculator. The 2- and 3-dimensional momentum equations can be arranged to calculate impact velocities for two or more vehicles given a set of exit velocities. Research by Emori [2] suggested energy-based methods that were ultimately turned into an algorithm by Campbell, first in his Ph.D. thesis [3] and later in his important SAE paper [4] published while he was at General Motors. McHenry developed the CRASH computer program [5] around these two (momentum and energy) methods. The CRASH model has been extended, first by NHTSA [6] and then by others [7,8,9]; still it is based on the momentum and energy methods in use for several decades. Additional researchers [10,11, 12] have programmed the momentum equations to calculate exit velocities from assumed impact velocities.

In the mid-seventies, McHenry also developed the first collision simulation model, SMAC [13]. Collision simulation was a fundamentally new approach to collision analysis made possible by the digital computer. Collision simulation methods calculate the collision forces and moments acting on the vehicle, then solve the equations of motion at small, user-defined time intervals. The SMAC algorithm has been extended by others [14,15,16], and is still the most popular simulation method in use today.

Finite element technology has existed for decades. It has been during the last decade or so that large-scale dynamic simulations with the finite element method have become possible. The processor speeds and memory with modern workstations and desktop personal computers allow dynamic finite element simulations with reasonable fidelity. Nonlinear material properties, large deformations, and complex contact conditions are typical of these analyses. These types of simulations are done by vehicle designers and manufacturers.

1. Numbers in brackets designate references found at the end of the paper.

Safety engineers have not used the finite element method outside of the realm of manufacturing and design. A single collision simulation typically takes several hours or even days to process on high-speed super computers.[29] In addition, developing a structural model for a single vehicle may take several weeks or more. Thus, the finite element method is not practical for use by those researchers who reconstruct collisions that occur on public roads.

Therefore, reconstructions are still performed using simple momentum and energy methods, or extended versions of the SMAC algorithm.

There exists room to improve the current methods for reconstructing collisions. However, the quantam leap from current methods to using a complex finite element code is not practical. A compromise approach has been developed employing techniques from finite element technology, and from the current methods, while still making a large advance in the state-of-the-art. This approach is called DyMesh (**D**ynamic **M**echanical **s**hell).

The purpose of this paper is to provide a detailed description of the DyMesh model. The process of integrating DyMesh into a typical simulation model is also provided. Finally, examples of results using DyMesh are presented, first on simple 3-dimensional shapes, and then on vehicle vs barrier and vehicle vs vehicle crashes.

GENERAL DESCRIPTION

This section gives an overview of the two main features of the DyMesh method - collision detection and collision forces.

COLLISION DETECTION – The DyMesh method is based on a collision algorithm which uses a triangular mesh defined by discretizing the exterior surfaces of a vehicle(s). The algorithm detects the interaction of vehicles by monitoring the positions of nodes, or vertices, and surfaces representing the exterior of the vehicles. The concept of a master surface and slave node is used. This mirrors proven technology used with contact algorithms for many years.

Contact algorithms are necessary in finite element analysis to simulate the contact and impact between bodies. These algorithms are widely used in explicit Lagrangian finite element codes such as PRONTO3D [17,18], EPIC-3 [19], and DYNA3D [20] and have been used successfully to simulate problems such as crush, impact, and penetration [21,22,23]. Contact algorithms are also used in static or quasi-static codes like JAC [24] and NIKE [25].

The basic operations performed in a contact algorithm are location and restoration. Location involves searching for surfaces in contact with one another and defining the contact geometry such as depth of penetration. Restoration involves finding the penetration depth and moving the slave node (vertex) back to the slave surface. In this context, we are using the classic definition of restoration as it is used by finite element contact algorithms. It is not at all related to restitution. Restoration is what causes the surfaces to remain in contact. The terminology used in the literature usually addresses one of the surfaces as the master and the other the slave. In three-dimensions, surfaces may be defined by four-noded quadrilateral elements or three-noded triangular elements. In the finite element method, quadrilateral surface elements are one side of a solid brick or hex element, and triangular elements are one side of a solid tetrahedral element.

Various algorithms treat the surfaces and forces using different methods [23,25]. The general idea is to move, or cause to move, one or both of the interpenetrating surfaces represented by discrete nodes so that the surfaces are in line-to-line contact. In Taylor's method [17], forces proportional to penetration depth are applied to slave nodes, which accelerate the node out of the penetrated surface. The slave node force is also apportioned to the nodes of the master surface. The enforcement of this no-penetration condition may occur over one or several timesteps. Belytschko [19] moves slave nodes onto the master surface and modifies the velocity of the node by $\Delta\mathbf{v}$ where $\Delta\mathbf{v} = \Delta\mathbf{x}/\Delta t$. The displacement of the node defines the displacement vector $\Delta\mathbf{x}$ and the current time step is the value for Δt. Then the momentum associated with modifying the slave node velocity is apportioned to the master nodes.

Contact enforcement in finite element methods is achieved on a local level. That is, depending on the particular method, forces, velocities, and/or momentum are prescribed to nodes and elements on the surface of the discretized body.

There have been many algorithms proposed to search for contact as it can be the most expensive component of the algorithm. An automatic global contact algorithm is desired to minimize the user definition of pairs of contact surfaces. Efficiency is important as the potential for much unnecessary and time-consuming searching exists in a global approach to contact.

There are many cases of contact, such as with corners and self-contact, that may cause problems for contact algorithms. An algorithm that monitors the velocity vectors of nodes penetrating a surface reduces the uncertainty involved in determining contact for many cases [26].

Figure 1 shows two bodies discretized on their exterior with nodes and elements. Consider the left body to be the master and the right to be the slave. At time t^k the bodies are moving towards one another with velocities $\boldsymbol{v}_m$ and $\boldsymbol{v}_s$, but there is no contact. At $t = \tilde{t}^{k+1}$ several slave nodes have penetrated the master surfaces. Slave node *s* has penetrated a distance *p*. The tilde on $\tilde{t}^{k+1}$ indicates that this is not the configuration at the end of the time step. The penetrating slave nodes have yet to be moved or restored to the master surface.

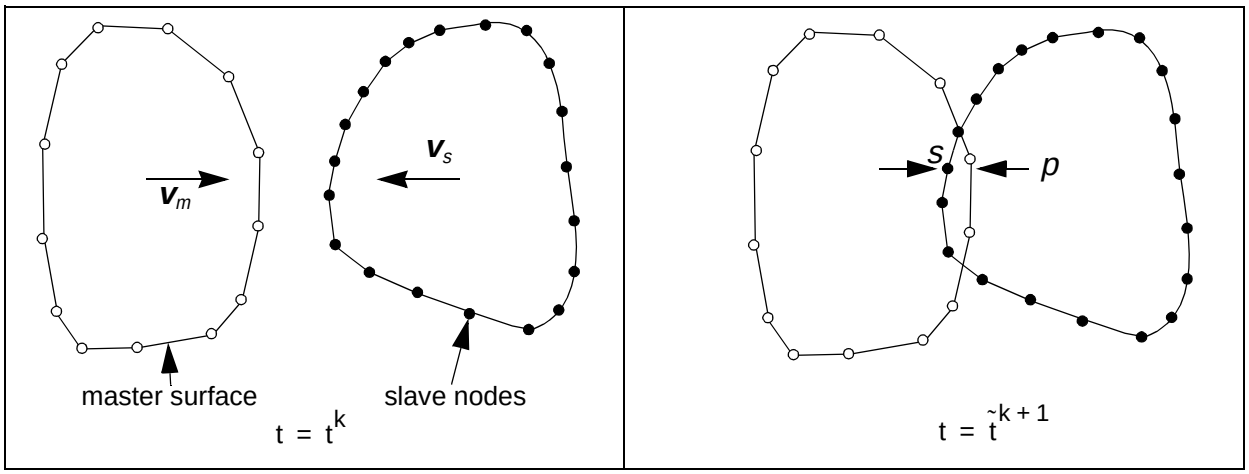


Figure 1. Master Surface-Slave Node Concept

Once penetration [Figure 2(a)] is detected, the pushback direction is determined, and slave node s is restored to be consistent with the kinematic constraints between the two surfaces. In most explicit, finite element, dynamics codes a force is applied to the slave node in a direction normal to the master surface $\boldsymbol{n}_m$ [Figure 2(b)], which is usually used as the pushback direction. This force accelerates the node to the outside of the master surface. The vector $\boldsymbol{n}_m$ passes through the point on the master surface to which the slave node projects. Another method used in finite element codes applies the slave node force in a direction of its normal $\boldsymbol{n}_s$. In this work since the impact forces are accelerating the CG and not individual nodes separately, nodes are restored by adjusting their three-dimensional coordinates along the pushback direction, and then calculating a force. It has been found that for this application an effective pushback direction is in the opposite direction of the node's velocity. In this case, the intersection of the line defined by the velocity vector with the master surface is calculated, and the slave node is moved to this point of intersection. This point of intersection is called the contact point.

COLLISION FORCES – After the slave nodes are restored [Figure 2(c)], the vector $\boldsymbol{\delta}^{k+1}$ is calculated. The force on the slave node is a function of the vehicle stiffness parameters, the volume of material associated with the crush of slave node *s*, and $\Delta\boldsymbol{\delta}$ where $\Delta\boldsymbol{\delta} = \left|\boldsymbol{\delta}^{o} - \boldsymbol{\delta}^{k+1}\right|$.

A difference between explicit, dynamic, finite element codes and this vehicle simulation algorithm is that the restoring force magnitude depends on the cumulative $\Delta\boldsymbol{\delta}$ as opposed to depending only on the instantaneous local surface penetration *p* (Figure 1).

During loading, the collision force at each slave node is based on a general purpose, 3rd-order polynomial force-deflection relationship,

$$F_{coll} = k_0 + k_1\Delta\delta + k_2\Delta\delta^2 + k_3\Delta\delta^3 + \kappa(\dot{\Delta\delta}) \qquad (1)$$

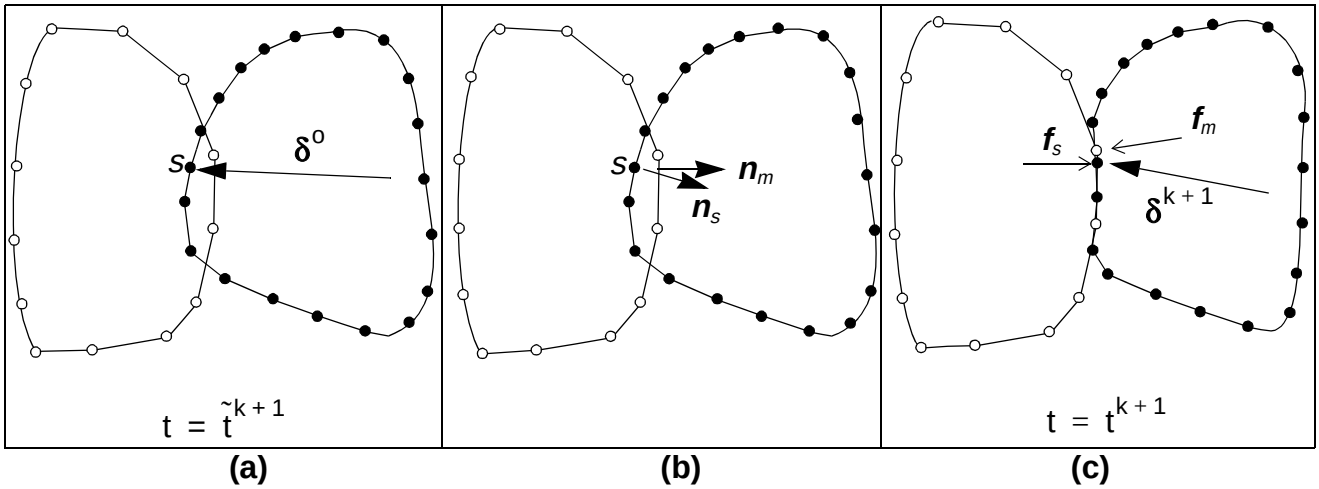


Figure 2. Restoration Phase

where k_0, k_1, k_2 and k_3 are the polynomial coefficients (note that if k_2=k_3=0, the model reduces to that described by Campbell [4]), and κ is a damping coefficient. The force-deflection relationship may also include a saturation force, F_{max}, a saturation deflection, d_{max}, and velocity- dependent node damping. A null band, d_0, is used for small node deflections (typically 0.5 inches or less). A force-deflection model for loading and unloading is shown in Figure 3.

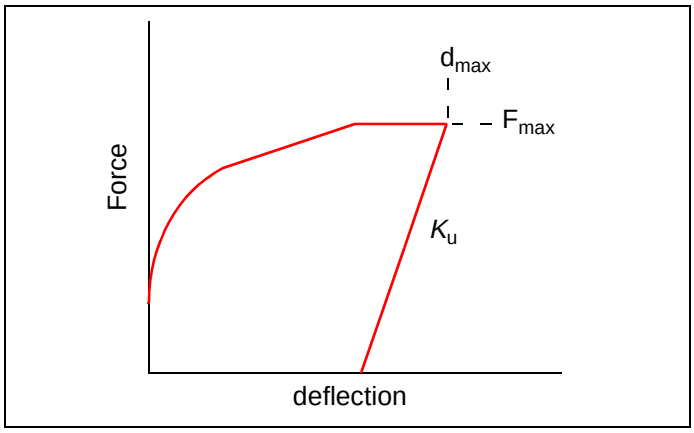


Figure 3. Force-deflection Relationship Using a 3rd Order Polynomial with Saturation Force and Deflection and Linear Unloading.

During unloading, the collision force is determined by an unloading slope, K_u. Specification of the unloading slope allows the model to account for restitution (a large unloading slope causes minimal restitution, while a smaller unloading slope would cause higher restitution). Note that the unloading slope must cause the residual node deformation to be zero or positive. The presence of unloading is determined by monitoring the deformation rate.

Tangential forces may also exist between interacting meshes. The tangential force is calculated as the product of the normal component of the collision force, F_{coll} (equation 1), and the friction coefficients for interacting nodes. The direction of the tangential force is determined from the direction of the relative node velocities.

Polynomial coefficients can be estimated from traditional crash test data and by estimating a crush height. For barrier crashes, this height is normally set to 30 inches (see Figure 4). Thus, by dividing the traditional A and B coefficients by 30 inches and setting k_2 and k_3 equal to zero, reasonable estimates for k_0 and k_1 are obtained. Note that because the collision forces are impulsive (i.e., occur over a short time interval), a reasonable range of estimates does not significantly affect the total computed speed change, but does affect the peak acceleration and duration of the collision (see comparisons with barrier and pole collisions later in this paper).

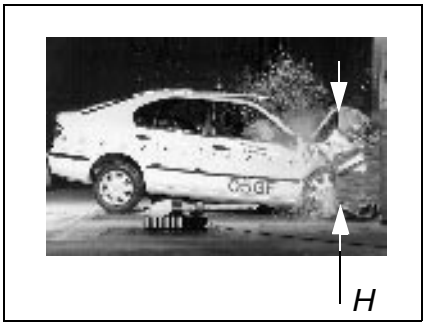


Figure 4. Height of Crush Region

NUMERICAL IMPLEMENTATION

This section discusses the numerical procedures used to implement the previously outlined methodology. The topics include initialization, collision detection, and the calculation of collision and friction forces.

INITIALIZATION – Initialization calculations are only done once. To initialize the collision algorithm the vectors from the vehicle CG to each node are calculated. These vectors are denoted by $\delta^{o}_{i,j}$, where *i* is the node number, and *j* is the vehicle index. The superscript indicates the time level is zero. The δ vectors are used to calculate the crush forces, and will be described in more detail later.

Next, the connected surface neighbors to each node are determined for each vehicle. Figure 5 illustrates the concept for node *i*. There are six Level 1 neighbors (in this example), denoted $N1_{n,j}$ where *1* indicates a Level 1 or connected neighbor, *n* ranges from one to the number of neighbors, and *j* is the vehicle index. Level 2 neighbors are also stored in memory. These are surfaces that share a node with a connected neighbor. The vehicle mesh connectivity that defines which nodes form a triangular surface does not change during a simulation. Thus, neighbors remain constant for a given mesh. These neighbor data are used to associate an area with a penetrating node and sometimes are used in the algorithm that searches to determine through which surface a node has penetrated. In addition, at the beginning of each timestep the area and outward normal for each surface is calculated.

COLLISION DETECTION – A gross collision detection is used during the simulation to initiate the detailed collision algorithm. A bounding box is placed around each vehicle. When these boxes overlap, the detailed collision algorithm is used.

When the collision algorithm is invoked to search for collisions, a neighborhood around each node is searched for potential surfaces through which it may have penetrated. This neighborhood is a rectangular bounding box whose

dimensions can be varied depending on the problem. For each slave node the potential contact surfaces are tested to determine which one, if any, it has penetrated. The sequence of tests is described below.

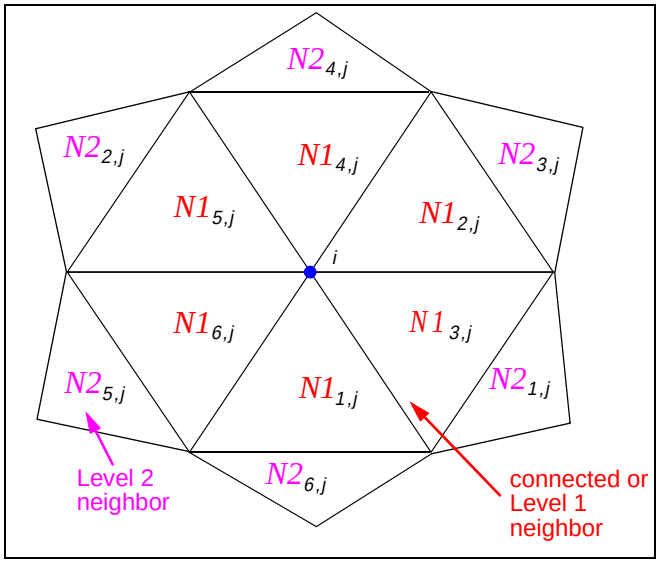


Figure 5. Neighbor Surfaces of Node *i*

First, the intersection of the line formed by the velocity vector of a slave node with the surface in question is calculated. This intersection or contact point, $\mathbf{x}_{cp}$, is defined as

$$\mathbf{x}_{cp} = \mathbf{x}_s + \mathbf{v}\alpha \qquad (2)$$

where $\mathbf{x}_s$ is the vector from the origin to the slave node, $\boldsymbol{v}$ is the velocity vector, and α is a parameter given by

$$\alpha = \frac{(\mathbf{x}_i - \mathbf{x}_s) \cdot \boldsymbol{n}}{\mathbf{v} \cdot \boldsymbol{n}} \qquad (3)$$

where $\boldsymbol{x}_i - \boldsymbol{x}_s$ is a vector originating at any one of the three triangular surface nodes (i=1,2, or 3), and $\boldsymbol{n}$ is the outward normal vector to the surface. The possibility exists for the denominator in equation 3 to be zero. In this case the velocity is parallel to the surface and there is no intersection between the two.

Several tests are made to determine if this contact point, $\mathbf{x}_{cp}$, is valid. First, the vector from the contact point to the slave node must be in the same direction to within 180 degrees of the velocity vector. Otherwise, the slave node could not have possibly penetrated the surface in question. This check amounts to calculating the dot product of the two vectors and ignoring those surfaces for which the dot product is negative. Another simple check requires that the slave node be inside the potential surface. This is done by taking the dot product of the vector normal to the surface with the vector from the contact point to the slave node. In this case the dot product must be less than zero.

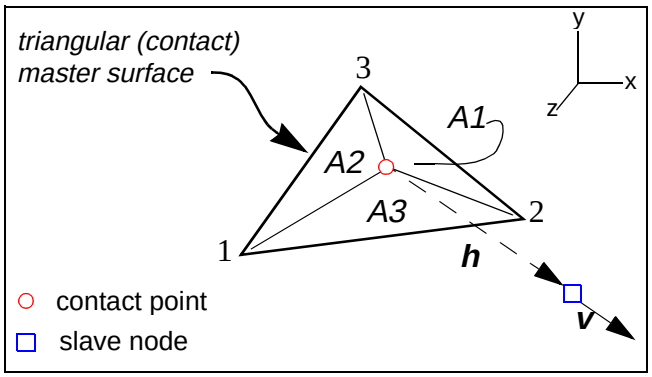


Figure 6. Contact Point Inside Surface

Next, the area formed by the three triangles and the contact point is calculated and compared to the area of the triangular surface in question (Figure 6). If $A1 + A2 + A3 = A_{123}$ the surface is still considered a candidate for contacting the slave node.

Another check determines if the distance from the slave node to the contact point is too far to be realistic This distance is denoted by the vector $\boldsymbol{h}$, shown as a dashed line in Figure 6. The average velocity of the master surface and the velocity of the slave node are used to determine a maximum distance that could possibly exist between the contact point and the slave node. If this distance is too great, the surface is no longer considered for contact with the slave node. If multiple surfaces meet all the above criteria, the one that minimizes the distance between the contact point and slave node is chosen.

COLLISION FORCES – There are no collision forces until there is deformation. Deformation occurs when a slave node is restored to its non-penetrating position which is defined to be on the contact or master surface. Figure 7(a) illustrates slave node s penetrating a surface in two-dimensions. In Figure 7(b) the penetrating nodes are restored to the contact points and the vector $\Delta\boldsymbol{\delta}$ is shown for node s. The change in the $\boldsymbol{\delta}$ vector at each node is used to calculate collision forces.

In this work an omni-directional stiffness is used although the use of orthotropic or other spatially varying stiffnesses is not precluded. The slave node collision force, $\mathbf{F}_{s,\,coll}$, due to the deformation $\Delta\boldsymbol{\delta}$ follows a tri-linear curve that includes unloading. A $\boldsymbol{\delta}$ vector component is in a state of loading when the deformation rate, $\Delta\dot{\boldsymbol{\delta}}_i$, is greater than a critical deformation rate $\Delta\dot{\boldsymbol{\delta}}_c$. For this loading regime the collision force is given by

$$\mathbf{F}_{s,\,coll} = (\tilde{A} + \tilde{B}|\Delta\boldsymbol{\delta}|)A_s\,\mathrm{sign}(\Delta\boldsymbol{\delta})\mathbf{1} \qquad (4)$$

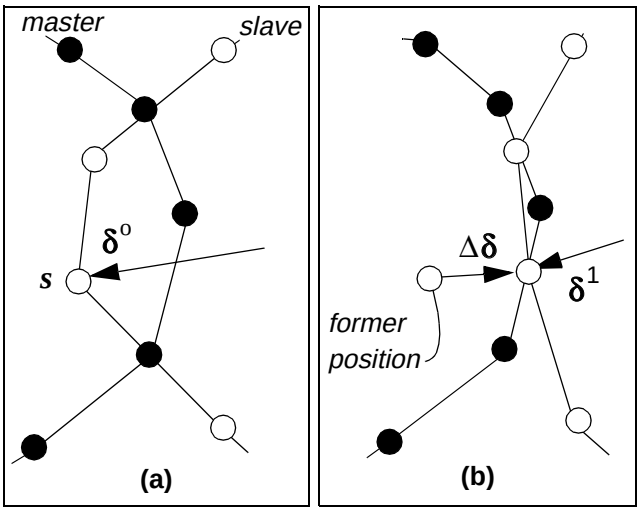


Figure 7. (a) Penetrating Slave Node and (b) Restored Slave Node in Two-Dimensions

where **1** is a unit vector in the direction of $\Delta\boldsymbol{\delta}$, A_s is the area associated with the penetrating slave node, $|\Delta\boldsymbol{\delta}|$ is the absolute value of $\Delta\delta$, $\tilde{A} = A/H$ and $\tilde{B} = B/H$ are the modified stiffness coefficients, and $\mathrm{sign}(\Delta\boldsymbol{\delta})$ applies the sign of each $\Delta\boldsymbol{\delta}$ component to each force component. The area associated with each penetrating slave node is approximated by

$$A_s = \frac{1}{3}\sum_{i=1}^{N_n} A_i \qquad (5)$$

where N_n is the number of connected neighbors, and Ai is the area of the i^{th} neighbor surface. In the loading regime individual $\Delta\delta$ vector components contribute to forces in their respective directions (omni-directional stiffness).

To unload in exactly the same direction as the slave node was loaded would require knowledge of the entire deformation history of the node. An efficient compromise approach used by DyMesh is to define the unloading direction to be the vector originating at the current slave node location and ending at the original slave node location, in the local vehicle coordinate system. When the slave node deformation rate falls below the critical deformation rate, $|\Delta\dot{\boldsymbol{\delta}}| < \Delta\dot{\delta}_c$, the δ vectors grow a distance defined by the unloading slope K_u in Figure 3. The growth of the vector is the maximum magnitude of the collision force divided by the unloading slope, $\delta_u = (\max\|F_{s,coll}\|)/K_u$. When the δ vector unloads or grows, in most cases, it will cause a contact condition to be enforced with the surface of the other vehicle. This contact condition reduces the growth of the δ vector. The corresponding force is calculated based on the new total length of the δ vector and the unloading slope K_u. Each time step in the unloading regime the δ vector will grow until the total unloading distance is δ_u.

Once the collision force is determined for a slave node, the force is distributed to the three nodes of the corresponding master surface. This distribution is accomplished using standard finite element shape functions which apportion the force to the nodes. The force magnitude on master nodes is

$$F_i = F_{s,coll}\, L_i \qquad (6)$$

where $F_{s,coll}$ is the magnitude of the slave node collision force and L_i is the shape function associated with master node *i*. The shape functions are given by

$$L_i = A_i / \sum_{j=1}^{3} A_j \qquad (7)$$

where A_i is the sub-area as shown in Figure 6. The forces F_i are applied in a direction normal to the master surface.

An elegant feature of the loading and unloading is that it is handled naturally through the contact algorithm. Collision forces are only present when there is contact between the vehicles. The unloading methodology is realistic in that it results in a body pushing itself away from the other body in a reasonable and controllable manner.

FRICTION FORCES – Friction forces are only present when there is contact between a slave node and master surface. The relative tangential velocity between the master surface and slave node at the contact point is determined. If the relative velocity is zero there is no friction. If there are nonzero friction forces, F_f,

$$F_f = (\mathbf{F}_{s,coll} \cdot \mathbf{n})\mu \qquad (8)$$

are applied in equal and opposite directions along the relative tangential velocity vector where ***n*** is the master surface normal, and μ is the friction coefficient.

DYMESH FLOW CHART – Figure 8 shows the main elements of the DyMesh algorithm which are enclosed in the dashed box. DyMesh is only invoked if the gross collision check indicates a collision is occurring or just about to occur. If there is no collision or when DyMesh has completed calculating the collision and friction forces the simulation program continues by calculating other external forces such as aerodynamic drag and tire forces (see section entitled Typical System Model).

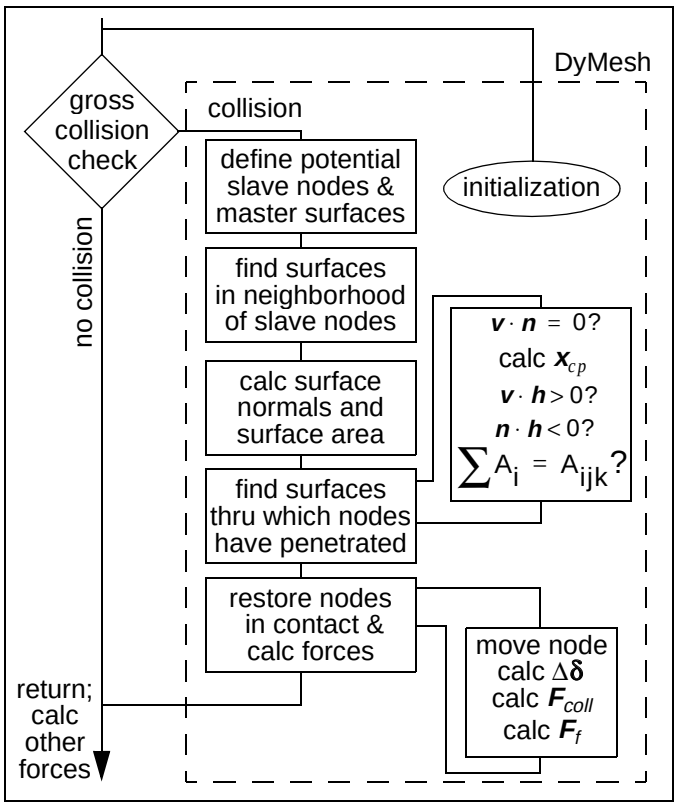


Figure 8. DyMesh Flow Chart

The DyMesh algorithm can be called in a variety of ways. For example, DyMesh can be called twice each timestep (swapping the master and slave objects), or it can be called once each timestep, alternating the master and slave objects. The latter approach reduces computation time, and was used in this work.

DYMESH OUTPUT – The output from DyMesh is a 3-dimensional force vector acting at each of the damaged nodes or vertices. Since the current vertex coordinates are known, it is a simple manner to calculate the forces and moments acting at the vehicle CG:

$$F_{x,\,cg} = \sum_{i=1}^{N_d} F_{i_{x,\,coll}}$$

$$F_{y,\,cg} = \sum_{i=1}^{N_d} F_{i_{y,\,coll}} \qquad (9)$$

$$F_{z,\,cg} = \sum_{i=1}^{N_d} F_{i_{z,\,coll}}$$

$$M_{x,\,cg} = \sum_{i=1}^{N_d} \left(-F_{i_{y,\,coll}} z_i + F_{i_{z,\,coll}} y_i\right)$$

$$M_{y,\,cg} = \sum_{i=1}^{N_d} \left(-F_{i_{z,\,coll}} x_i + F_{i_{x,\,coll}} z_i\right) \qquad (10)$$

$$M_{z,\,cg} = \sum_{i=1}^{N_d} \left(-F_{i_{x,\,coll}} y_i + F_{i_{y,\,coll}} x_i\right)$$

Here, i is a damaged vertex, F_i is the force on the damaged vertex, N_d is the number of damaged vertices, and x_i, y_i, and z_i are the vertex vehicle-fixed coordinates.

TYPICAL SYSTEM MODEL

This section discusses how the output from DyMesh can be easily coupled into a vehicle simulation program. Simulation models normally include four common components:

- main control routine,
- numerical integration routine,
- free-body analysis of the objects,
- acceleration calculation.

A flow chart for a typical simulation model is shown in Figure 9. The DyMesh algorithm contributes to the force calculations. It should be noted that these components are required by all simulations, whether they are used to model humans, vehicles, even the weather. The primary difference is in the free-body analysis of the objects. In our case, the objects are vehicles and our analysis includes six degrees of freedom for the collision forces and moments acting on each sprung mass.

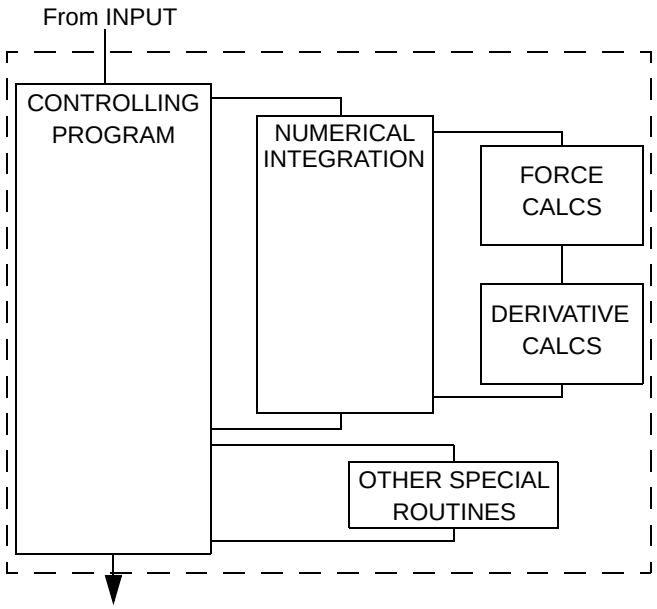


Figure 9. Simulation Flow Chart

These four basic system components are described below.

MAIN CONTROL ROUTINE – The main control routine does just what its name suggests: it controls the sequence of calculations. The following tasks are performed by the typical control routine:

- initializes all variables
- calls the numerical integration routine
- calls the output routine
- performs logical checks that affect execution.

DyMesh requires that several parameters be initialized, including the initial (undeformed) mesh coordinates, vertex velocities, and forces and deflections material parameters (force vs deflection, friction, damping) for each vertex. Calling the numerical integration starts the process that updates the velocities and positions for each timestep (see below). The output routine updates the current simulation results for each timestep. This step includes updating the current vertex coordinates so they may be visualized as vehicle damage. Logical checks include termination conditions and setting various flags to control execution.

NUMERICAL INTEGRATION – The numerical integration routine causes the free-body analysis to be executed for each timestep. Typically a fourth order routine is used, meaning that the free body analysis is performed four times per timestep. After the accelerations have been computed (see below), they are returned to the numerical integration routine to be integrated. This integration process updates the velocity and position for the next timestep. Any valid numerical integration method may be used. Common methods include Runge-Kutta, Adams-Moulton and Hamming's Modified Predictor-Corrector.

FREE BODY ANALYSIS – DyMesh calculates the collision forces and moments acting on each vehicle. This process was described above in detail.

Other forces acting on the vehicle may include tire forces, aerodynamic forces and suspension forces. After each force producer is executed, the results are summed to produce the total vehicle-fixed forces and moments acting on the vehicle.

ACCELERATION VECTOR – Each force vector has an associated acceleration vector. According to Newton's 2nd law, the acceleration in each direction is equal to the product of the force and mass (linear acceleration) or the product of the moment and rotational inertia (rotation). Mathematically these equations of motion are expressed as:

$$\begin{aligned} \sum F_x &= m(\dot{u} - vr + wq) \\ \sum F_y &= m(\dot{v} + ur - wp) \\ \sum F_z &= m(\dot{w} + uq - vp) \end{aligned} \tag{11}$$

$$\begin{aligned} \sum M_x &= I_{xx}\dot{p} + qr(I_{zz} - I_{yy}) \\ \sum M_y &= I_{yy}\dot{q} + pr(I_{xx} - I_{zz}) \\ \sum M_z &= I_{zz}\dot{r} + pq(I_{yy} - I_{xx}) \end{aligned} \tag{12}$$

where m is the sprung mass, u, v and w are forward, lateral, and vertical velocity, p is roll (about x), q is pitch (about y), r is yaw (about z), and I_{jj} is inertia about axis j, all in vehicle-fixed coordinates.

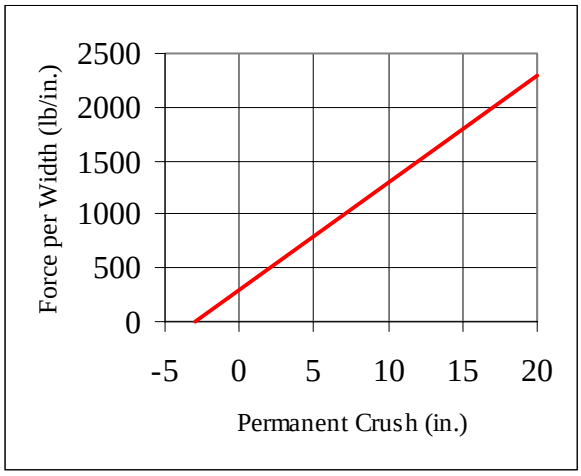
To solve the equations, the mass matrix is inverted and the accelerations are computed using the SimSol (simultaneous solution) function. [28]

OUTPUT – The primary output from DyMesh is the current vertex coordinates and level of force acting at each vertex on the mesh in the vehicle-fixed coordinate system. The vertex coordinates allow the damaged mesh to be visualized at each timestep. Intermediate values available from DyMesh include vertex deflection and deflection rates.

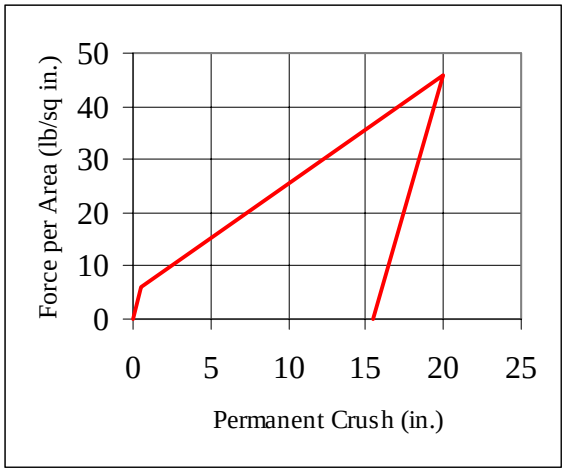
EXAMPLES

The first two examples use simple geometries to demonstrate the algorithm. The first is a cube impacting a rigid wall. Next, two colliding spheres are simulated. Then three examples are presented with vehicle meshes. The first of these is a simulation of a 35 mph barrier impact test on a Ford Escort. Then a collision between a Festiva and a telephone pole is simulated. Finally, a two-vehicle simulation is presented.

In these examples piecewise-linear force-deflection relationships are used. For vehicles, the standard two-dimensional force-per-width versus deformation relationship [Figure 10(a)] is extended to three-dimensions using equation 1 and assuming $k_2 = k_3 = \kappa = 0$. The three-dimensional model takes the form shown in Figure 10(b). The non-permanent deformation represented by G ($G = A^2/2B$) is accounted for in the unloading, and the forces are per unit area instead of width. Conversion to the three-dimensional form is achieved by dividing A and B by the height H (Figure 4) of the vehicle crushed when the stiffness parameters were generated. The portion of the curve with a slope K_u is the unloading path.



(a)



(b)

Figure 10. (a) Standard 2D and (b) New 3D Force Deflection Curve

The examples shown use an omni-directional stiffness. That is, if a node is deformed equally in all three directions the force in each direction will be equal. The capability is for each node to have unique stiffnesses, and the stiffnesses for each node may be a function of the coordinate direction. Also, as stated earlier, the force model is not limited to piecewise linear segments. More complex quadratic or cubic functions may be defined by using a nonzero k_2 or k_3 in equation 1.

CUBE-WALL COLLISION – This simulation is of a unit cube impacting a rigid wall at a velocity of 2 (consistent units). A cross-section view of the problem set-up is shown in Figure 11. The cube is made up of 8 vertices and 12 elements. This simple problem provides an opportunity to compare the DyMesh algorithm results to theoretical results.

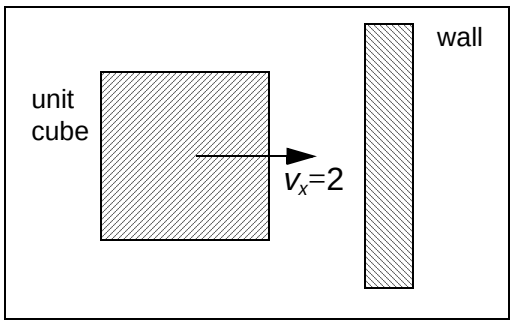


Figure 11. Cube Collision with a Rigid Wall

All stiffness parameters are zero except for $\tilde{B} = 25$. The cube has a mass of 0.3. The initial kinetic energy is

$$KE = \frac{1}{2}(0.3)4 = 0.6 . \quad (13)$$

The unloading slope is set to twice the loading slope. Therefore, the cube should rebound from the wall with one-half of the original kinetic energy. Also, the maximum deformation can be calculated based on conservation of energy. Each of the four vertices will absorb one-quarter of the energy. Equating the energy used in deforming each vertex to kinetic energy gives

$$\frac{1}{2}F_{coll}\Delta\delta = \frac{0.6}{4} \quad (14)$$

where the factor of one-half comes from knowing one-half of the energy is used to induce permanent deformation. Since the loading slope, $\tilde{B}$, is 25, the maximum theoretical deformation can be solved for in equation 14 by substituting $F_{coll} = 25\Delta\delta$ which gives

$$\frac{1}{2}25(\Delta\delta_{max})^2 = \frac{0.6}{4}$$
$$\text{or} \quad (15)$$
$$\Delta\delta_{max} = 0.1095 .$$

Since only one-half of the damage will be left following restitution, the final deformation is given as

$$\Delta\delta_{final} = \frac{\Delta\delta_{max}}{2} = 0.05475 . \quad (16)$$

The energy and deformation history for a vertex in the DyMesh simulation are shown in Figures 12 and 13. It can be seen that the final kinetic energy is 0.3, or one-half of the original value of 0.6, which agrees with theory. In the lower plot, the maximum deformation is 0.1095 which agrees with theory. The final deformation, $\Delta\delta_{final}$, is 0.0574 which differs by 4.8% from the theoretical value of 0.05475.

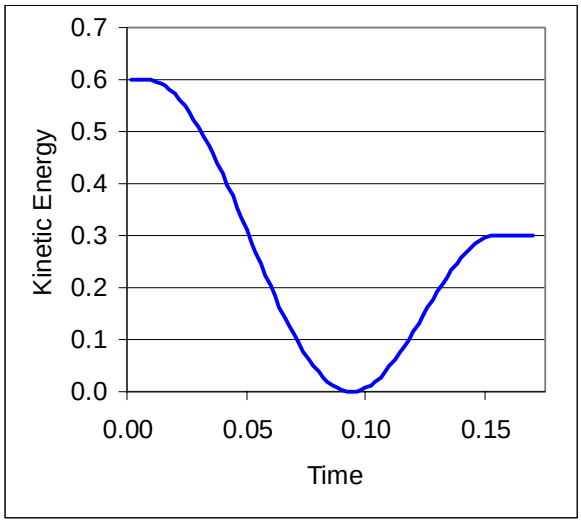


Figure 12. Cube-Wall Results - Kinetic Energy

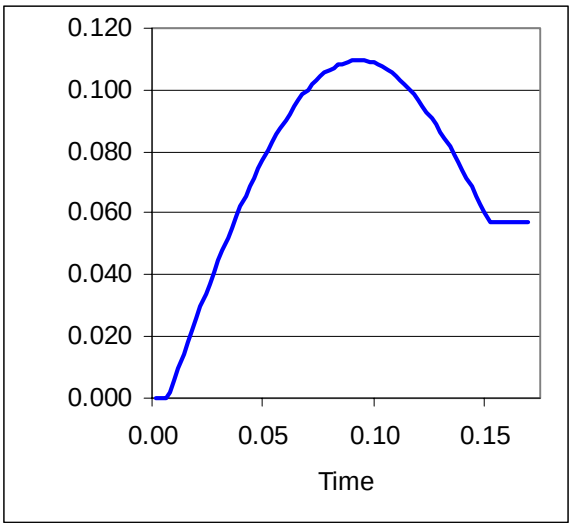
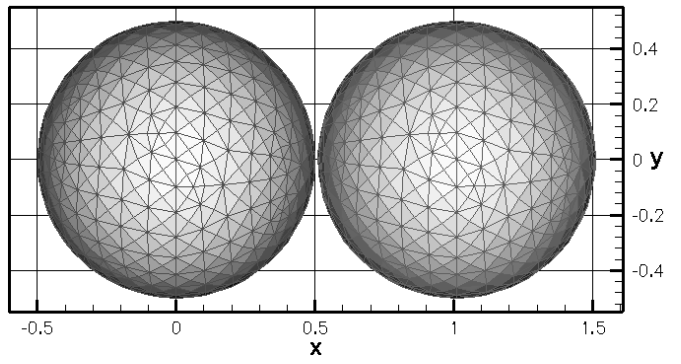


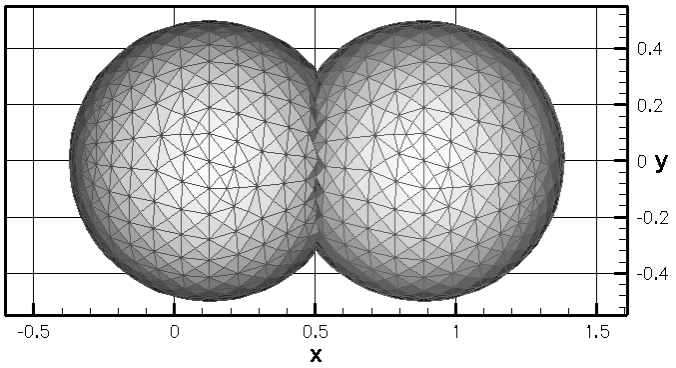
Figure 13. Cube-Wall Results - Deformation

SPHERE-TO-SPHERE COLLISION – This example illustrates the DyMesh algorithm on two spheres discretized by 1,452 triangular surfaces. The problem set-up is shown in Figure 14(a). Both left and right spheres have a mass of 0.1, a radius of 0.5, a stiffness $\tilde{B} = 500$, and a velocity of 3 (relative velocity of 6) in the *x* direction. A time step of two milliseconds is used.

Two simulations are presented - without and with restitution. Ideally, the interface between the colliding spheres should be flat. Figure 14 shows the two spheres at t=0 and t=0.7, and Figure 15 shows two views of the left sphere at t=0.7, without restitution. The surface area which contacted the right sphere is nearly flat.



(a)



(b)

Figure 14. (a) Initial Positions and (b) Positions at t=0.7

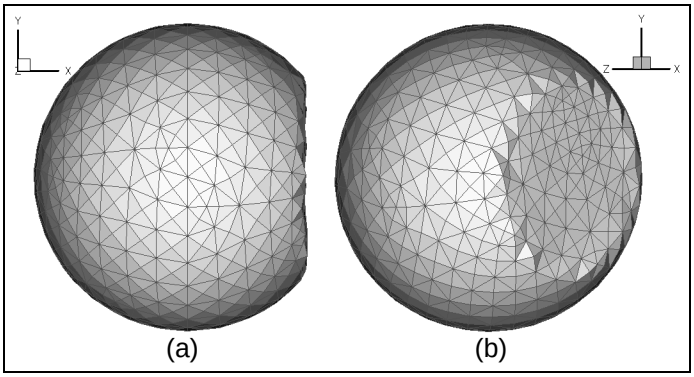


Figure 15. Sphere Geometry After the Collision: (a) Side View and (b) rotated 30 Degrees about the Y Axis

History plots of kinetic energy, velocity, and acceleration are shown in Figure 17. It can be seen that without restitution the impact is plastic. Both spheres come to rest following the impact, and the acceleration quickly goes to zero.

With restitution both spheres rebound and the acceleration more gradually approaches zero as the spheres separate as seen in the plots in Figure 17. The geometry of the left sphere during restitution is shown in Figure 16. The discrete time step and alternate swapping of master and slave surfaces causes slight irregularity in the surface as the δ vectors are lengthened.

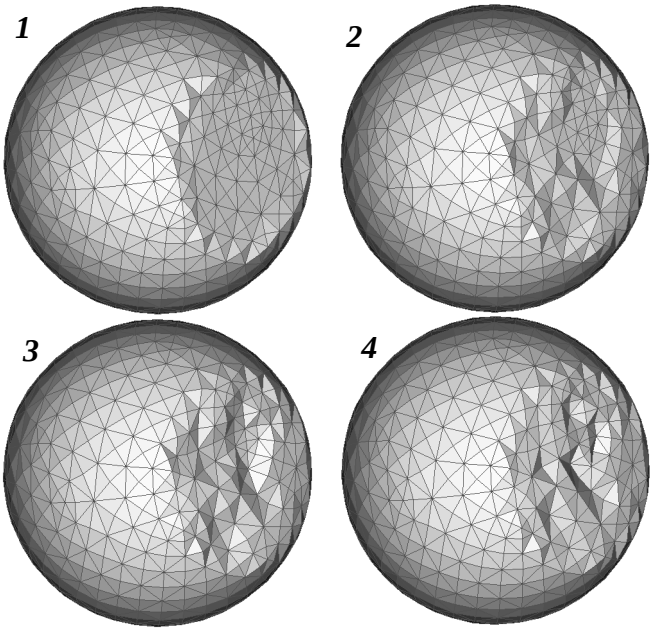


Figure 16. Sphere Geometry During Restitution

VEHICLE-TO-BARRIER COLLISION – In this simulation the DyMesh method is used to simulate a 1997 Ford Escort barrier crash test. A government report fully documents the test.[27] The vehicle weighed 2,963 lb and impacted the barrier at 35.1 mph (617.8 in/s). The damaged region length L was measured to be 53.1 in. The damage profile is shown in Figure 18.

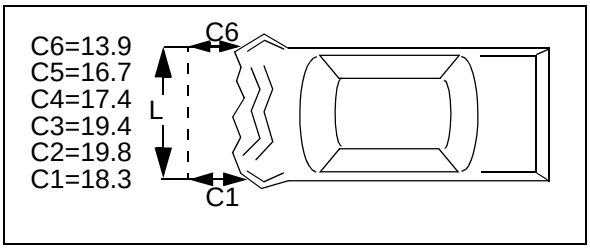


Figure 18. Escort Damage Profile

Assuming that the vehicle can withstand a 7 mph collision without damage, the standard stiffness coefficients are A=501 lb/in and B=115 lb/in^2. Assume that the damaged vehicle height in the test is 35 in., and the modified stiffness coefficients are $\tilde{A}$ = 14.3 lb/in^2 and $\tilde{B}$ = 3.3 lb/in^3. These constants will be used in the DyMesh simulation.

Figure 19 shows the mesh of the escort and the barrier at the beginning of the simulation. The Escort mesh has 2,075 nodes and 3,827 triangular elements. The barrier is made of eight nodes and six large triangular elements. To emphasize the fact that only collision forces are being considered, tires are not shown and are not part of the calculation. The time step used in the simulation is one millisecond, and the entire simulation takes 40 seconds to run using a 233 MHz Pentium II processor.

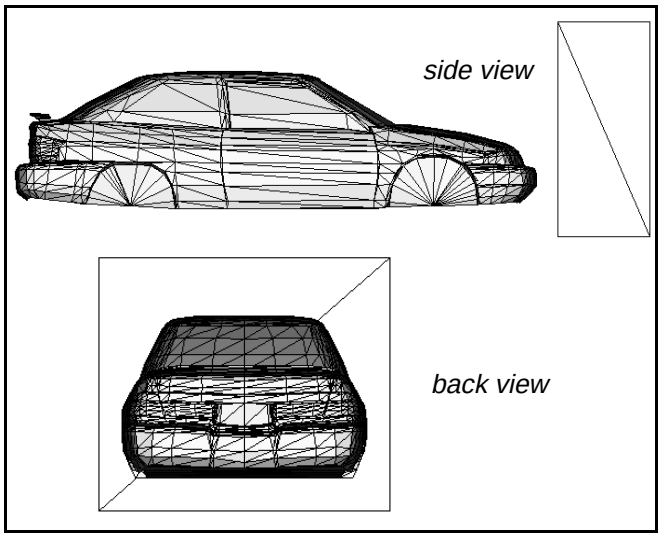
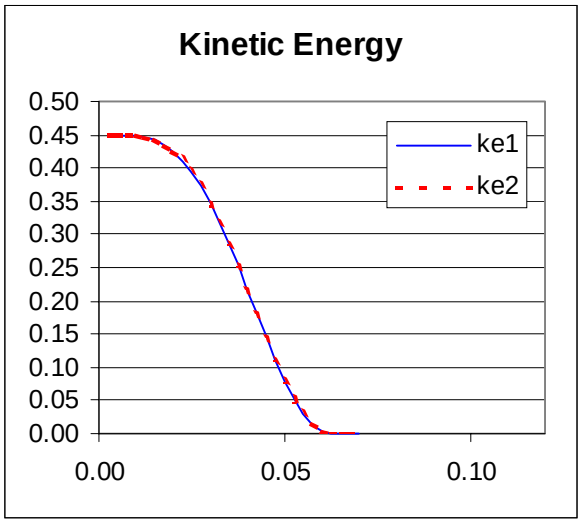
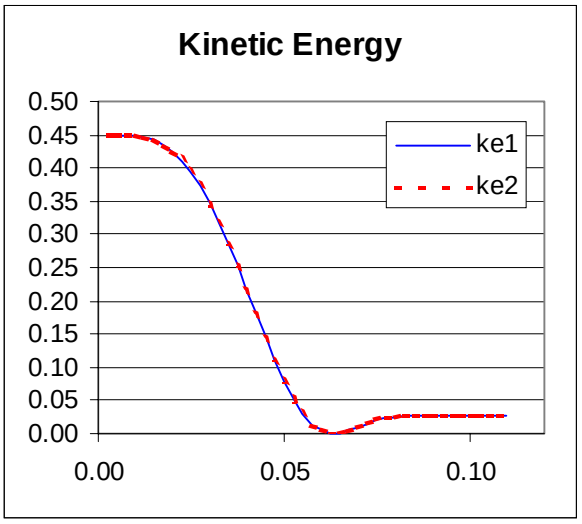


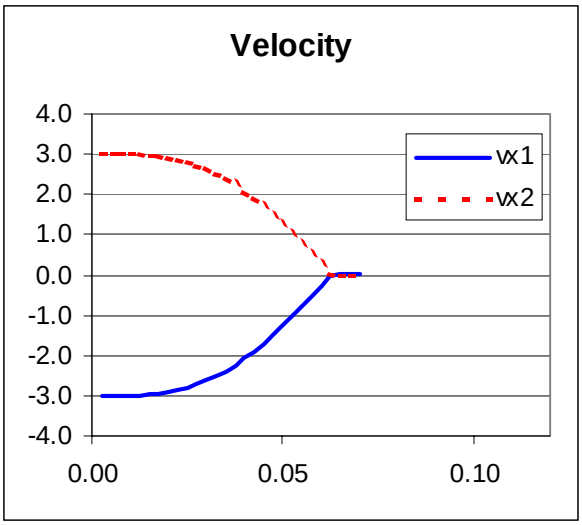
Figure 19. Escort and Barrier Mesh

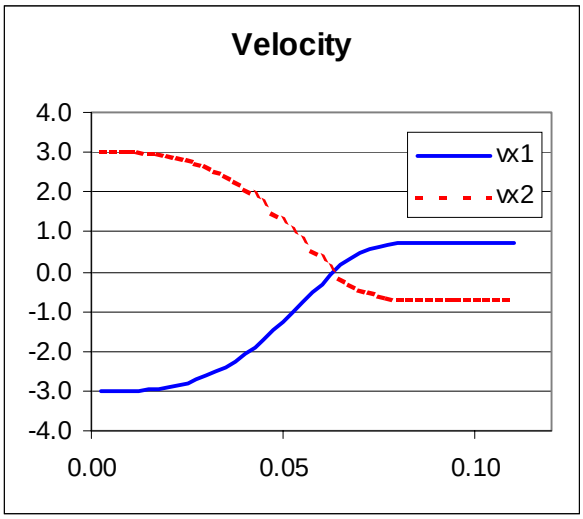
Figure 20 shows the velocity and acceleration history of the DyMesh simulation compared with the test. The change in velocity in the test was 705 in/s compared with 742 in the DyMesh simulation - a difference of 5.2 percent. The peak acceleration in the test was -40 G compared with -44 G in the DyMesh simulation - a difference of ten percent. These results suggest the selected $\tilde{A}$ and $\tilde{B}$ stiffness coefficients were too stiff. Reducing these values would reduce the peak acceleration and increase the duration of the collision. Note however, the selection of the stiffness coefficients have little effect on the total velocity change. As a demonstration of this, two additional barrier simulations were run with stiffness coefficients ten percent higher and lower. The peak acceleration differed by as much as 16.8 percent from the nominal case, but the velocity change only differed by 0.5 percent at most.

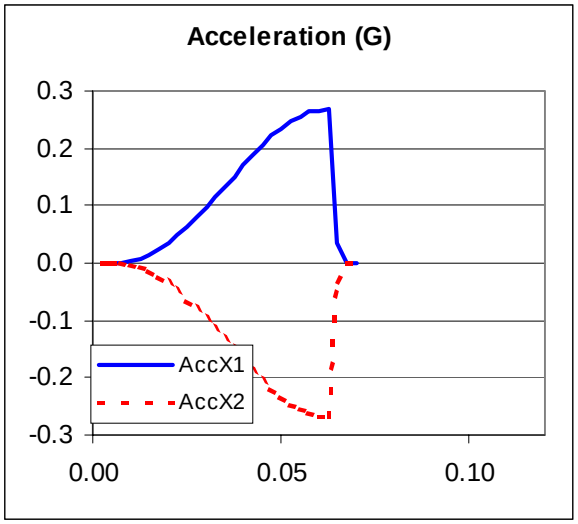
Figure 21 shows the deformed mesh geometry with contours of deformation overlaid on the mesh. The maximum deformation occurs in the center of the bumper, that part of vehicle which protrudes the most. It is interesting to note the contour changes between 75 ms and 100 ms. At 100 ms it is evident from the recession of the darkest contour that there has been some restitution (i.e., there is less area covered by the darkest contour at 100 ms compared with 75 ms).

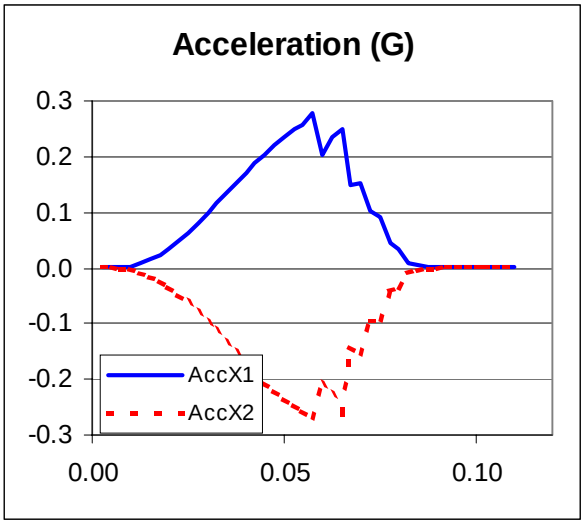








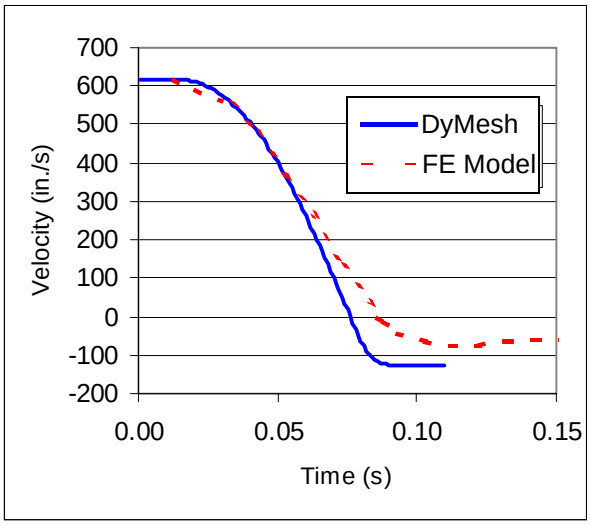




without restitution

with restitution

Figure 17. History Plots for Sphere Collision Simulation



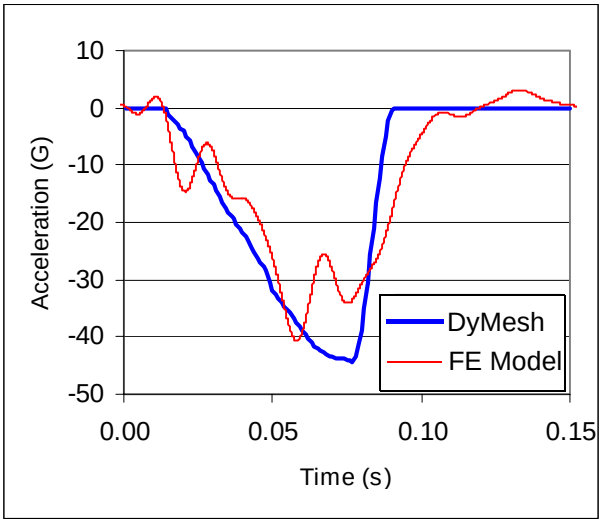


Figure 20. Comparison of Velocity and Acceleration History

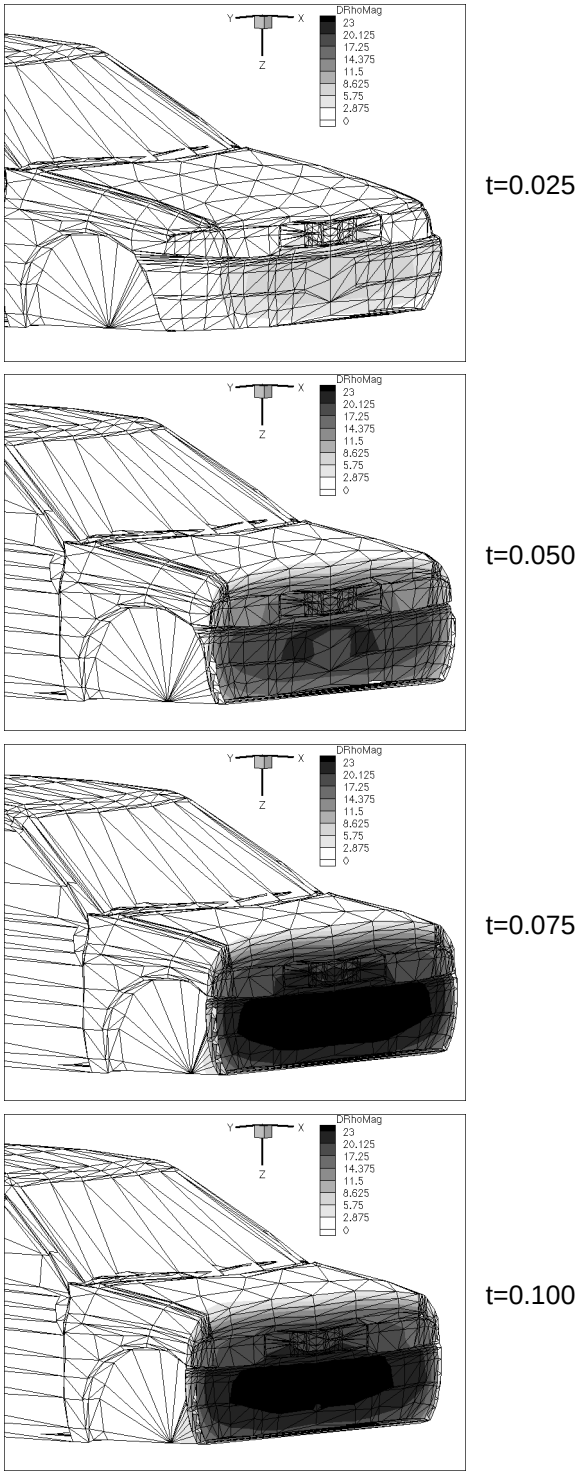


Figure 21. Escort Mesh Geometry and Deformation Contours

VEHICLE-TO-POLE COLLISION – This simulation is of a Ford Festiva collision with a rigid telephone pole. A similar simulation of a Ford Festiva is presented by Grau [29] using the finite element model available from the George Washington University. The DyMesh Festiva model is composed of 5020 nodes and 9403 triangular elements.

The zone of impact has been refined to accurately resolve the contact with the simulated telephone pole. The rigid telephone pole is constructed from 1,053 nodes and 2,080 triangular elements and is 8.7 inches in diameter. The Festiva is traveling 20 mph when it impacts the telephone pole as shown in Figure 22.

The mass associated with the Grau finite element simulation was suspect; however, for purposes of comparison, the same was used in the DyMesh simulation.[30] The modified stiffness coefficients used are $\tilde{A}$=9.12 lb/in^2 and $\tilde{B}$=1.80 lb/in^3. Since the finite element model results showed no rebound velocity from the pole, the unloading stiffness was set to infinity so that there is no restitution.

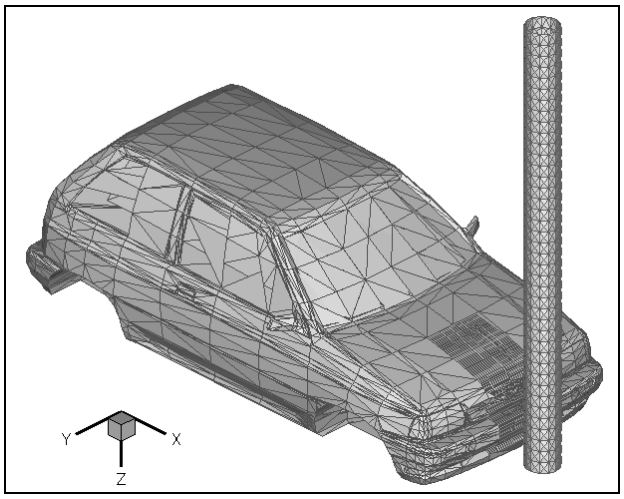


Figure 22. Initial Positions for Festiva-Pole Collision

Figure 23 shows the deformed geometry at 20, 80, and 100 milliseconds. The rigid pole has been removed from this figure for clarity. It can be seen that the vehicle takes on the circular shape of the pole. At 80 ms a few element edges are seen stretched through the pole. Only nodes or vertices, not edges, have contact prescribed in DyMesh. (Please note the occasional irregularity in the windshield and window mesh is due to rendering and is not a result of DyMesh.)

Figure 24 shows the acceleration history plot of Grau's finite element simulation and the DyMesh simulation. The rise time for the DyMesh simulation is slightly longer, but the agreement in peak acceleration is good with the difference being less than ten percent. In this example the selected stiffness coefficients were too soft, resulting in the underestimation of peak acceleration and the overestimation of crush depth. Again, it is clear that the change in velocity agrees well with the finite element model (Figure 25).

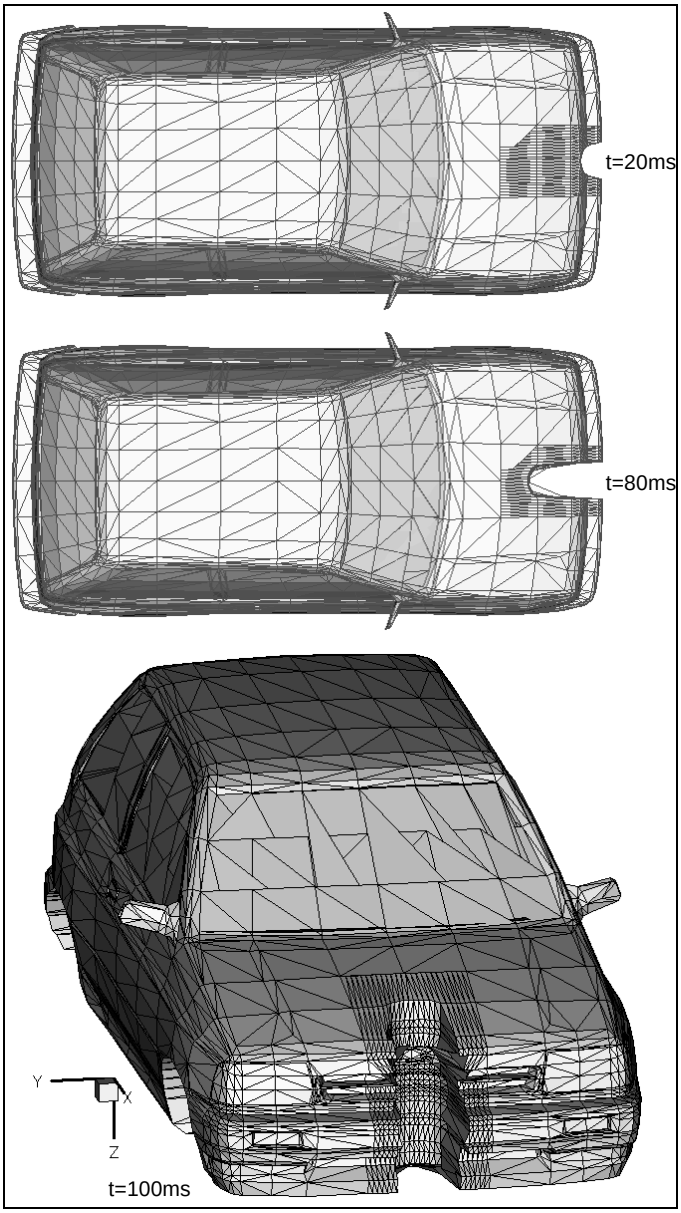


Figure 23. Deformed Geometry (pole not shown)

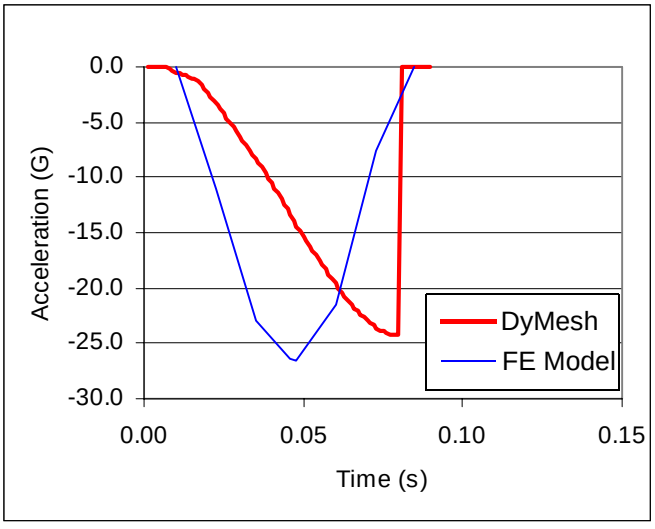


Figure 24. Acceleration Comparison

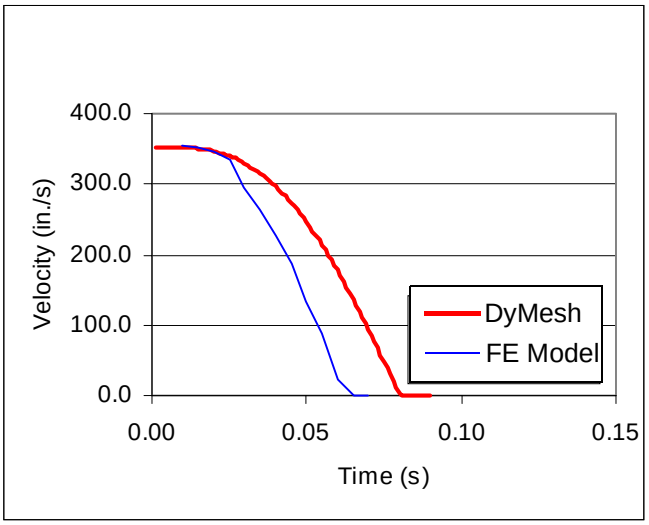


Figure 25. Velocity Comparison

VEHICLE-TO-VEHICLE COLLISION – This offset frontal impact simulation between two Escorts is a severe test of the DyMesh algorithm. The irregular shapes of the bumpers and varying size of elements challenge the algorithm to maintain a reasonable mesh shape during deformation.

Figure 26 shows the initial positions of the two vehicles. Both vehicles are assumed to weigh 2975 lb and have yaw inertia of 20,000 lb-s^2-in. The vehicle's x-axes are initially colinear. Both the left and right vehicles have an initial velocity of 15 mph. Vehicle stiffnesses are $\tilde{A}$ = 14.0 lb/in^2 and $\tilde{B}$ = 3.3 lb/in^3.

A time step of one millisecond is used in the simulation. Twenty milliseconds of simulation time are achieved for every one minute of elapsed (wall-clock) time when running on a Pentium II, 233 MHz processor. The simulation is run until the X velocity of both vehicles reaches zero.

Figure 27 shows both vehicles at various times during the simulation. The vehicles begin to rotate due to the offset forces. At t=88 ms the vehicles are shown contacting each other as well as with the left vehicle removed to expose the current damage profile produced by DyMesh.

Correct restoration in the contact algorithm is seen in the damage profile. Incorrect restoration would result in the vehicle meshes moving through one another without damage. The robustness of the contact algorithm in DyMesh is evidenced by the smooth and continuous appearance of the damage (Figure 28) even though the vehicle surface mesh uses elements of different sizes and aspect ratios.

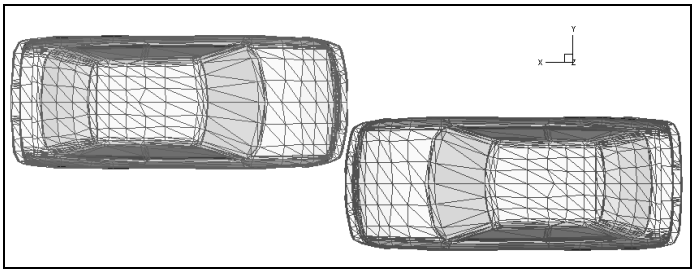


Figure 26. Initial Positions for the Offset Frontal Impact

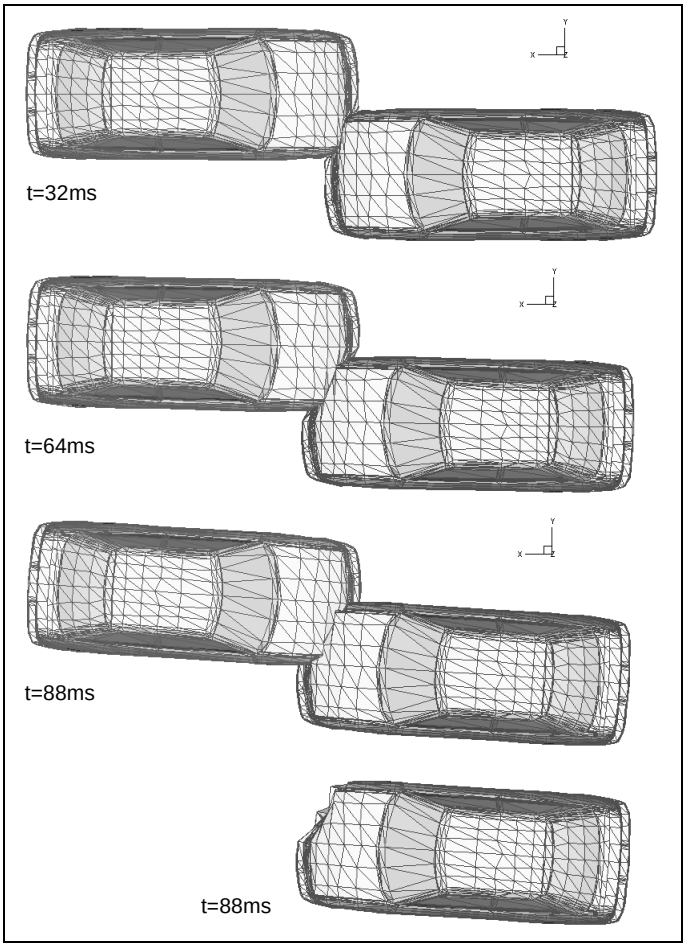


Figure 27. Vehicles at Various Times

Figure 29(top) shows the X velocity of each vehicle which illustrates the symmetry of the collision simulation. Figure 29(bottom) shows the crush area calculated for each vehicle. It is important to note that the crush areas are calculated independently for each vehicle. For a symmetric collision such as this one, the crush areas would ideally be identical. The plot indicates the areas are the same, except for a small time around 70 milliseconds.

DISCUSSION

The DyMesh method has the potential to allow more realistic collision simulations than previously possible in an interactive simulation environment. Three-dimensional forces and moments, as well as damage profiles, can be visualized. Vehicle damage is not as realistic as with the finite element method (i.e., DyMesh will not simulate hood buckling), but the results for change in velocity are good. The simulation run times are on the order of seconds or minutes, not several hours which is typical of finite element methods.

The overall resolution of the simulation is dependent upon the mesh resolution, as was demonstrated in the telephone-pole example. Penetration of bodies smaller than the mesh cannot be resolved by DyMesh.

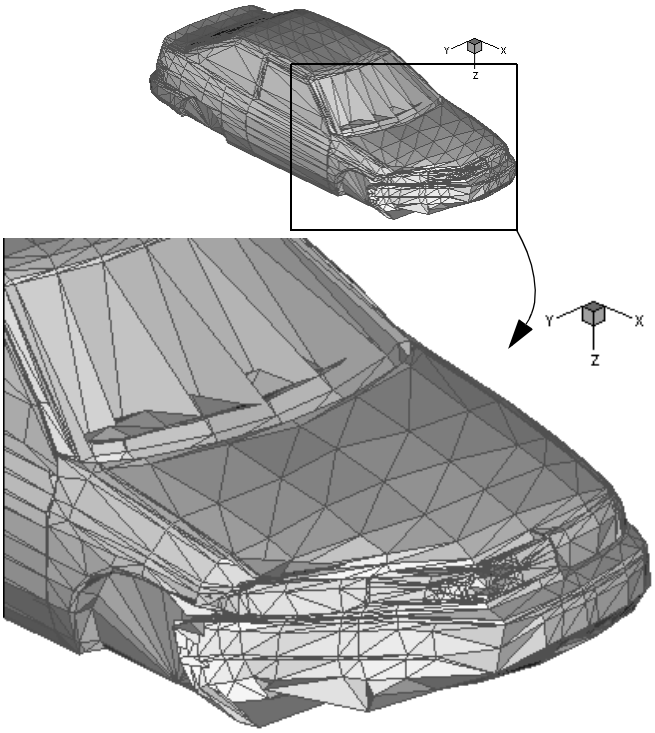


Figure 28. Close-up Of Damage Profile

The method is not dependent upon the vehicle shape being described by a simplified geometric body, such as a parallelepiped. The method holds promise for car-to-barrier, car-to-car, heavy truck and articulated vehicle collisions. Underride can be simulated in this three-dimensional method.

The DyMesh method requires a vehicle mesh and 3-dimensional stiffness coefficients, $\tilde{A}$ and $\tilde{B}$. Meshes of reasonable resolution are available from a variety of sources. [e.g., 31,32]. Initial tests presented in this paper suggest $\tilde{A}$ and $\tilde{B}$ calculated from currently available A and B coefficients are sufficient for speed change calculations required in crash reconstruction. This finding is supported by previous experience with collision simulation programs (e.g., [14,16]). Experience has shown these estimates are also sufficient for use as a starting point for rollover simulation [33].

Collision pulses from DyMesh may also be used for other purposes, including occupant simulations. For these uses, the proper selection of $\tilde{A}$ and $\tilde{B}$ is more important. For these applications, the user must confirm the simulated and measured crush depths match. At this time it is not felt that additional testing is required to use DyMesh.

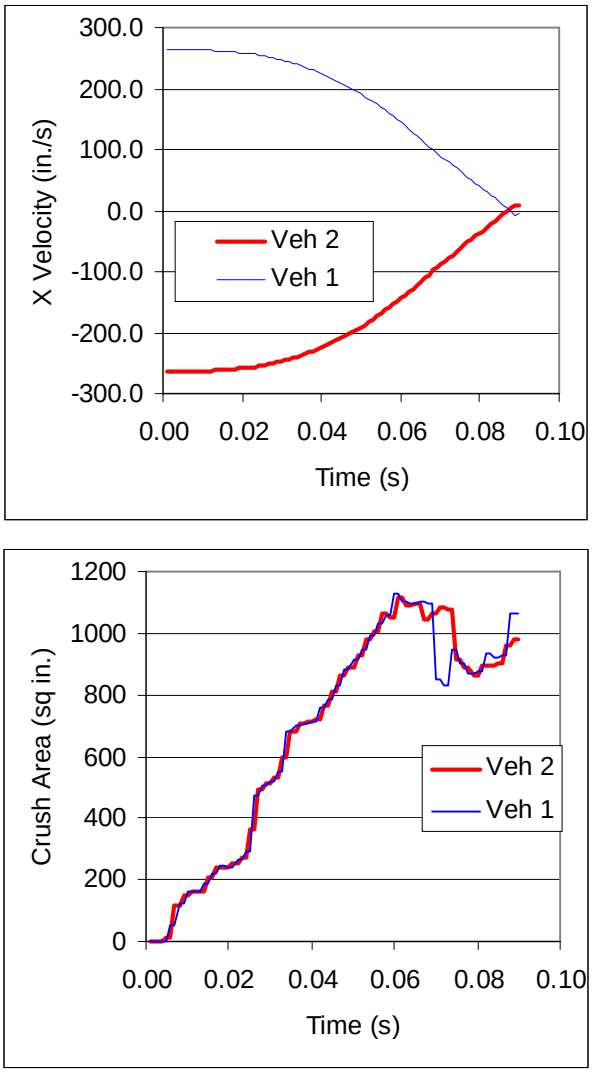


Figure 29. Velocity and Crush Area Histories

Further work needs to be performed to evaluate the use of DyMesh in collisions wherein the mechanism of contact is primarily shear, for example, the case of a passenger car sliding under the side of a semi-trailer.

CONCLUSIONS

The DyMesh method is described for calculating the three-dimensional interactions of colliding vehicles or structures. This method makes possible a more realistic numerical calculation and enhances the visual effectiveness of the simulation.

A novel method for computing the vehicle deformations based on a contact algorithm has been implemented. The force-crush model is derived which transforms the usual two-dimensional relationship to three-dimensions. The equations of motion are presented which depend on the three-dimensional collision forces output from DyMesh.

Examples have shown the DyMesh method to be robust and effective. This work lays the foundation for a more thorough validation effort to be undertaken in the near future.

REFERENCES

1. U.S. Dept. of Transportation, National Highway Traffic Safety Administration, Washington, DC, 1998 (1997 and 1998 figures are estimated).
2. Emori, R.I., “Analytical Approach to Automobile Collisions,” SAE Paper No. 68016, Society of Automotive Engineers, Warrendale, 1968.
3. Campbell, K.L., “Energy as a Basis for Accident Severity - A Preliminary Study,” Doctoral Thesis, University of Wisconsin, Dept. of Mechanical Engineering, Madison, 1972.
4. Campbell, K.L., “An Energy Basis for Collision Severity,” SAE Paper No. 74565, Society of Automotive Engineers, Warrendale, 1974.
5. McHenry, R.R., “Extensions and Refinements of the CRASH Computer Program Part I, Analytical Reconstruction of Highway Accidents,” PB76-252114, 1976.
6. Noga, T., Oppenheim, T., “CRASH3 User's Guide and Technical Manual,” NHTSA, U.S. Dept. of Transportation, Washington, DC, 1981.
7. Day, T.D., Hargens, R.L., “An Overview of the Way EDCRASH Computes Delta-V, “SAE Paper No. 870045, Society of Automotive Engineers, Warrendale, 1987.
8. Bigg, G., Moebes, T., WinCrash User's Manual, AR Software, Redmond, 1996.
9. Fonda, A.G., “CRASH Extended for Desk and Handheld Computers,” SAE Paper No. 870044, Society of Automotive Engineers, Warrendale, 1987.
10. Woolley, R., “The IMPAC Computer Program for Accident Reconstruction,” SAE Paper No. 850254, Society of Automotive Engineers, Warrendale, 1985.
11. Smith, G., “Conservation of Momentum Analysis of Two-Dimensional Colliding Bodies, With or Without Trailers, “SAE Paper No. 940566, Society of Automotive Engineers, Warrendale, 1994.
12. Stephan, H., Moser, A., “The Collision and Trajectory Models of PC-CRASH,” SAE Paper No. 960886, Society of Automotive Engineers, Warrendale, 1996.
13. McHenry, R.R., Jones, I.S., Lynch, J.P., “Mathematical Reconstruction of Highway Accidents - Scene Measurement and Data Processing System,” Calspan Report No. ZQ-5341-V-2, DOT HS-801 405, February, 1975.
14. Day, T.D., Hargens, R.L., “An Overview of the Way EDSMAC Computes Delta-V,” SAE Paper No. 880069, Society of Automotive Engineers, Warrendale, 1988.
15. Bigg, G., Moebes, T., WinSmac User's Manual, AR Software, Redmond, 1998.
16. Day, T.D., “An Overview of the EDSMAC4 Collision Simulation Model,” SAE Paper No. 1999-01-0102, Society of Automotive Engineers, Warrendale, 1999.
17. Taylor, L.M., and Flanagan, D.P., “PRONTO 3D A Three-Dimensional Transient Solid Dynamics Program, SAND87-1912, Sandia National Laboratories, March 1989.
18. S.W. Attaway, “Update of PRONTO 2D and PRONTO 3D Transient Solid Dynamics Program,” Sandia National Laboratories, SAND90-0102, November 1994.
19. T. Belytschko and J.I. Lin, “A New Interaction Algorithm with Erosion for EPIC-3,” U.S. Army Ballistic Research Laboratory Report BRL-CR-540, February 1985.
20. J.O. Hallquist and D.J. Benson, “DYNA3D user’s manual (nonlinear dynamic analysis of structures in three dimensions,” Lawrence Livermore National Laboratory, UCID-19592, Rev. 3, 1987.
21. J.D. Gruda and A.R. York, “Crashworthiness of the AT-400A Shipping Container,” Development, Validation, and Application of Inelastic Methods for Structural Analysis and Design, PVP-Vol 343, 1996 International Mechanical Engineering Congress and Exposition, pp. 115-121.
22. A.R. York and A.M Slavin, “Design and Analysis of a High-Performance Shipping Container for Large Payloads,” The 11th International Conference of the Packaging and Transportation of Radioactive Material (PATRAM ‘95), Dec 3-8, 1995, Las Vegas, Nevada, Proceedings Vol IV, pp. 1736-1743.
23. D.J. Benson and J.O. Hallquist, “A single surface contact algorithm for the post-buckling analysis of shell structures,” Computer Methods in Applied Mechanics and Engineering, 78, pp. 141-163 (1990).
24. J.H. Biffle, “JAC - A Three-Dimensional Finite Element Computer Program for the Nonlinear Quasi-Static Response of Solids with the Conjugate Gradient Method, Sandia National Laboratories, SAND87-1305.
25. J.O. Hallquist, “NIKE3D: An implicit, finite deformation, finite element code for analyzing the static and dynamic response of three-dimensional solids,” Lawrence Livermore National Laboratory, UCID-18822, Rev. 1., 1984.
26. M.W. Heinstein, S.W. Attaway, J.W. Swegle, and F.J. Mello, “A General-Purpose Contact Detection Algorithm for Nonlinear Structural Analysis Codes,” Sandia National Laboratories, SAND92-2141, April 1995.
27. U.S. Department of Transportation, National Highway Traffic Safety Administration, “New Car Assessment Program (NCAP) Frontal Barrier Impact Test,” Report No. MGA-96-N018, September 23, 1996.

28. HVE Developer's Toolkit, Version 2, Engineering Dynamics Corporation, Beaverton, OR, October 1998.
29. Grau, C.A. and Huston, R.L., "Validation of an Analytical Model of a Right-Angle Collision between a Vehicle and a Fixed, Rigid Object," International Journal of Crashworthiness, V 3 No. 3, pp 249-264.
30. Personal communication with Cesar Grau, December 7, 1998.
31. EDVDB Vehicle Database, Version 2, Engineering Dynamics Corporation, Beaverton, OR 1998.
32. Viewpoint Datalabs, Orem UT, 1998.
33. HVE User's Manual, Version 2, Engineering Dynamics Corporation, Beaverton, OR, 1998.

ABOUT THE AUTHORS

Allen R. York has worked in the area of computational mechanics for ten years. Before forming A.R. York Engineering, Inc., he worked nine years at Sandia National Laboratories. He can be reached by email at: aryork@flash.net.

Terry D. Day has worked in the area of motor vehicle handling and collision simulation for 18 years. He can be reached at Engineering Dynamics Corporation by email: day@edccorp.com.

TRADEMARKS