UCRL--94028

DE86 000691

THE APPLICATION OF DYNA3D TO LARGE SCALE CRASHWORTHINESS CALCULATIONS

David J. Benson and John O. Hallquist
University of California,
Lawrence Livermore National Laboratory
Livermore, California

M. Igarashi, K. Shimomaki, and M. Mizuno
Suzuki Motor Co., Ltd
Japan

ABSTRACT

This paper presents an example of an automobile crashworthiness calculation. Based on our experiences with the example calculation, we make recommendations to those interested in performing crashworthiness calculations. The example presented in this paper was supplied by Suzuki Motor Co., Ltd, and provided a significant shakedown for the new large deformation shell capability of the DYNA3D code.

INTRODUCTION

Crashworthiness engineering is a high priority at Lawrence Livermore National Laboratory because of its role in the safe transport of radioactive material for the nuclear industry and the military. The greatest difficulty in designing for crashworthiness is evaluating a proposed design. Experiments are very expensive and analytical solutions do not exist. Numerical methods must be used.

Finite difference codes were developed two decades ago to analyze two dimensional problems involving large plastic deformations, and many of them are still used. Some of the algorithms used in these codes, such as radial return plasticity, are used in modern finite element codes. Finite difference codes rely on a logically regular mesh, and using them on a problem with a complicated topology is often a challenge. Defining contact surfaces with corners is also a problem with many finite difference codes.

The finite element method was developed to overcome many of the limitations of the finite difference method. For structural calculations, the finite element method has largely replaced the finite difference method, but the finite difference method is still preferred by many for fluid dynamics problems. It is also still used in many hydrodynamics calculations, such as shaped charge design, where the pressure is high enough for metal to behave like a fluid.

Computer hardware has advanced substantially over the past twenty years. Modern supercomputers, such as the Cray-1, can perform over 100 MFLOPS (Millions of floating point operations per second) provided the software is written to take advantage of their hardware. Many people were initially disappointed with supercomputers because they discovered that most old programs only ran three or four times faster on the Cray-1 than on the CDC-7600.

One of the first finite element programs to take full advantage of supercomputers is DYNA3D [1,2,3], a completely vectorized, explicit finite element program for solving three-dimensional, inelastic, large deformation structural dynamics problems. Originally developed by Hallquist, and now co-developed by Hallquist and Benson, it was first released in 1976, and has undergone continual development ever since. Its average execution rate ranges from 35 to 72 MFLOPS on the Cray/XMP, depending on the element and material types. When multi-tasking becomes available on a production basis for the Cray/XMP-48, we anticipate that DYNA3D will execute at a rate of up to 260 MFLOPS.

Most of the original applications of DYNA3D involved thick structural members, and the lack of shell and beam elements was not important. When bending effects were important, they were included by using several brick elements through the thickness of a structure. The maximum timestep in an explicit calculation is determined by the thinnest dimension of a brick element. Thin shell structures therefore increase the cost of an analysis by several orders of magnitude if they are modeled with brick elements.

Even with our access to supercomputers, we are computer bound by the size of our applications. Our typical applications involve 10,000 to 50,000 elements. Fine meshes are required to adequately resolve stress and strain gradients, and we are often unable to obtain the desired resolution with current problem sizes. Several applications of interest would require more than 1,000,000 elements to obtain the resolution common in two dimensional calculations. We have been extremely motivated to obtain optimum speed.

Recent applications of DYNA3D now frequently involve thin structural elements, and an accurate, efficient shell element is a necessity. We based our element on the formulation by Hughes and Liu [4,5]. Our original implementation was not vectorized for the Cray, and it was proved to be extremely inefficient. Our current implementation [10] is over one hundred times faster and it has been used extensively for large scale calculations. Without the shell element, DYNA3D is clearly inappropriate for automobile crashworthiness calculations, but with it, however, we believe that it is an effective tool.

EXPLICIT VERSUS IMPLICIT TIME INTEGRATION FOR CRASHWORTHINESS ANALYSIS

Explicit finite element codes are used for analyzing problems that have a significant high frequency content in their transient response. Typical problems that fit that description are automobile crashes and shaped-charge design. The spatial resolution of explicit calculations is usually higher than is possible with implicit calculations because there is no need to solve any simultaneous equations. Recent calculations that were run on DYNA3D include a shell model with over 20,000 elements and 120,000 degrees-of-freedom. Only two hours of CPU on a Cray/XMP were required. In contrast, a verification calculation with NIKE3D, an implicit code, which used only 2000 elements, required over eight hours of CPU. The high spatial resolution is often very important in dynamic buckling calculations because the imperfections of the parts must be included for an accurate solution. The principal limitation with explicit calculations is the limit on the integration time step size. Sound travels in most metals at a rate of roughly .5cm/microsecond, and the Courant stability criterion states that the time step must be small enough that a sound wave cannot travel across the smallest element during one integration step. Typical time steps are below one microsecond, and therefore, explicit calculations are impractical for events occurring over a duration of seconds.

Implicit codes are necessary for static solutions and for problems where the duration is long enough that the loading can be regarded as quasi-static. Dynamic problems involving space platforms, where the frequencies of the lower modes are often below one Hertz, are one area where implicit codes are far more productive than explicit codes. Most of the problems analyzed with NIKE3D are quasi-static problems, such as metal forming. We do not expect the situation to change until iterative equation solvers, such as the element-by-element method [11], are perfected.

The central trade-offs of explicit versus implicit is between the time step size and computer resources. Explicit calculations use very little memory, but take very small timesteps. Implicit calculations require the entire memory, if possible, and considerable disk space for the storage of a large stiffness matrix, but time steps thousands of times larger than in an explicit calculation are possible.

To make explicit calculations competitive with implicit calculations, each time step must be as cheap as possible. The implication is, therefore, that the elements should be as simple as possible. The simplest elements use linear displacement fields with uniformly reduced integration and hourglass control. Quadratic elements, for the same spacing between nodes, require smaller time steps than linear elements. They also require many more operations per node than linear elements. Virtually all explicit codes are therefore restricted to linear elements. To further reduce the computational cost of an element, various simplifications are introduced to reduce the operation count, and they usually depend on the fact that the geometry of an element does not change very much over a single small time step.

Based on the need for high spatial resolution necessary for accurately modeling the geometry and buckling that occurs during a car crash, and the short duration of a crash, we believe that explicit finite element methods are more cost effective than implicit methods.

A REVIEW OF DYNA3D

A complete development of the theoretical and computational methods of DYNA3D is too long for this paper; the interested reader can get the basics from the now dated Theoretical Manual [1] and the papers cited in our references. Our emphasis, after a few paragraphs to establish notation, is on the features in the program that are of particular interest in crashworthiness design.

Our viewpoint is strictly Lagrangian. Material points are identified by their initial location, $\underline{X}$, in the undeformed body. The current position of a point, $\underline{x}(\underline{X},t)$, is a function of time and its initial location.

The principal of virtual work is the foundation of the displacement finite element method. It is the weak form of Cauchy's first law of motion and is related by applying the divergence theorem.

$$\delta\pi = \int_\Omega \rho\ddot{x}_i \delta x_i d\Omega + \int_\Omega \sigma_{ij}\delta x_{i,j} d\Omega - \int_\Omega \rho f_i \delta x_i d\Omega - \int_{\Gamma_t} t_i \delta x_i d\Gamma = 0 \quad (1)$$

where

- ρ is the density.
- $\underline{x}$ is the displacement.
- σ is the Cauchy stress.
- $\underline{f}$ is the body force.
- $\underline{t}$ is the surface traction.
- Ω is the domain of the body.
- Γ_t is the section of the boundary of Ω that has tractions applied.
- $\delta\pi$ is the virtual work.

The finite element method interpolates $\underline{x}$ throughout the body from its nodal values.

$$x_i = \phi(\xi,\eta,\zeta)_\alpha x_{\alpha i} \quad (2)$$

where

- ϕ_α is the isoparametric shape function at node α.
- $\underline{x}_\alpha$ is the displacement at node α.

ξ, η, ζ are the isoparametric coordinates of a material point $\underline{X}$.

DYNA3D has isoparametric eight node brick elements, four node shell elements and two node beam elements. The interpolation methods for the shell and beam elements are more complicated than equation (2) because of their rotational degrees of freedom. All of the elements are formulated to handle large, nonlinear deformations.

Substitution of equation (2) into equation (1) gives a set of simultaneous equations for the accelerations. The superscript n refers to the n-th integration time step.

$$M_{\alpha i \beta j} \ddot{x}^n_{\beta j} = F^n_{\alpha i} \quad (3)$$

where

$M_{\alpha i \beta j}$ is the mass matrix.

$F_{\alpha i}$ is the sum of the internal and external forces.

We use a diagonal, lumped mass matrix in DYNA3D, which makes solving for the accelerations in equation (3) trivial. A lumped mass matrix also leads to more accurate answers in many situations, especially those involving shock waves.

The velocities and displacements are calculated using central difference integration, an explicit method which is second order accurate.

$$\dot{x}^{n+1/2}_{\alpha i} = \dot{x}^{n-1/2}_{\alpha i} + h^n \ddot{x}^n_{\alpha i} \quad (4)$$

$$x^{n+1}_{\alpha i} = x^n_{\alpha i} + h^{n+1/2} \dot{x}^{n+1/2}_{\alpha i} \quad (5)$$

$$h^{n+1/2} = \frac{1}{2}(h^{n+1} + h^n) \quad (6)$$

where

h is the integration stepsize.

In addition to the usual features found in completely nonlinear finite element programs, our experience indicates that there are three additional capabilities that are critical to crashworthiness analysis: 1) a broad range of efficiently implemented material models, 2) sophisticated contact algorithms for the impact interactions, and 3) a rigid body capability to represent the bodies away from the impact zones at a greatly reduced cost without sacrificing any accuracy in the momentum calculations.

Crashes, by definition, involve contact between surfaces; any program used for crashworthiness analysis must have a variety of sophisticated contact/impact algorithms. The first involves two arbitrary surfaces, such as a bumper hitting a barrier, where the surfaces deform and undergo large relative displacements. The second type is single surface contact which occurs when a surface folds over on to itself, e.g., the controlled crush structural members of a car. In DYNA3D, both types of contact are easily modeled, including, friction and automatic closure and separation, by using the penalty method [12]. When a node penetrates a surface, a force proportional to the penetration is applied normal to surface on the penetrating node and the reaction force is distributed to the four nodes defining the appropriate surface segment. A friction force based on the normal force is also applied if it is requested.

The most expensive part of contact algorithms is the search algorithm for determining the contact points, and not, as might first be expected, the actual calculation of the contact force. Efficient and reliable search algorithms [12] have been developed for DYNA3D over the years; virtually every production analysis performed with DYNA3D uses the contact/impact algorithms, therefore any flaws in new versions of the algorithms are quickly exposed and corrected.

Momentum is usually the driving force in crashes. Any moving structure must be modeled in its entirety, no matter how small the impact area, in order to assure the accuracy of the momentum calculations. Away from the impact zones, we desire the cheapest possible representation of the body that accurately models the body's inertial properties so that the cost of the analysis is minimized. To that end, David Benson and John Hallquist implemented a rigid body material type in DYNA3D [13] based on the earlier theoretical work by Benson for his thesis [14]. Each finite element is assigned a material number in the data. All of the elements that share a common material number, where that material is specified to be rigid, define a rigid body. Separate rigid bodies with different material numbers can also be merged to form a single rigid body. In addition to all of the standard boundary conditions, contact surfaces, and body forces acting on the bodies, joint constraints, such as universal joints, are used to tie bodies together.

SHELL ELEMENT SELECTION

The Hughes-Liu element, which we chose for DYNA3D, is a degenerated shell element, an approach originally presented by Ahmad, et al. [15] for small strain, small deformation problems and which is also used by many others. Hughes and Liu studied three large strain, large deformation elements in their paper. Using their notation, we implemented U1, a 4-node shell with uniform reduced integration, in DYNA3D, and S1, a 4-node shell with selective reduced integration, in NIKE3D.

There is a substantial body of literature devoted to shell elements. We did not make a comprehensive review of the literature in making our choice, rather the Hughes-Liu formulation has several qualities that we find desirable and we chose it based on those qualities.

First and foremost, it is a linear element. Our brick elements are linear and we saw no point in implementing a higher order shell element.

The second reason is the simplicity of the element. Simplicity, in our experience, usually translates into robustness and computational efficiency.

Thirdly, it is based on degenerating a brick element into a shell element. A substantial amount of time was invested in optimizing the brick elements in our codes and many of the techniques developed for the bricks were transferred to the shell elements.

Fourth, it accommodated finite transverse shear strains. While in most cases the zero transverse shear strain assumption, as in corotational formulations, is a good approximation, we prefer not to make it. The computational overhead with retaining the additional shear strain appears to be insignificant.

The fifth reason, in our experience, has not turned out to be valid. Hughes and Carnoy [8] presented an algorithm for including the thinning effects of large membrane strains in the element as an extension to the original Hughes-Liu formulation. In our implementation of this modification in NIKE3D, we found that more corrector iterations were needed at each step and the results were too flexible in some applications. We have therefore not included the extension in our current implementation.

SHELL ELEMENT IMPLEMENTATION

In our first implementation of the Hughes-Liu shell, we used reduced/selective integration. This approach requires an assembly of the strain displacement matrix, B, by adding a partial strain displacement matrix for the transverse shear strains, computed at the centroid, to its complement at the Gauss integration points. Therefore, with reduced/selective integration we face the added cost of the centroidal calculations for B. The cost of B plus the cost of the constitutive model dominate the operation counts as can be seen in Table 1. Our choice of one point integration has reduced cost by nearly a factor of four over the reduced/selective option. Our operation counts take advantage of the simplifications that are possible at the element centroid to reduce the operation count, i.e. an integration point computed away from the centroid requires additional operations. The same simplification for one point integration of the brick has already been discussed [1,3].

EXAMPLE: Suzuki Motor Body Impact Test

All of the experimental work and mesh generation was performed by the last three authors at Suzuki Motor.

The car chassis, including the suspension, weighs 255.3 kgm. The mesh, shown in Figure 1, has 3439 shell elements and 3109 nodes. Although a whole car is shown in Figure 1, only half the car is modeled. To approximate the effects of the suspension inertia the density of the material behind the firewall was scaled to give the mesh the correct overall mass.

Accelerometers on the chassis and high speed photography provided us with the experimental data for comparison to our calculations. The initial velocity is 46.6 km/h and the response time is 55ms. Figure 2 shows a sequence of deformed shapes calculated by DYNA3D and Figure 3 shows the experimental and calculated velocity-time curves at one point in the chassis.

Our calculation agrees quite well with the experiment out to twenty or thirty milleseconds. The deformed states show buckles forming in the proper locations and in the proper sequence. At later times, two factors cause the large disparity between the calculated and experimental results.

The main factor is the omission of the suspension system. During the crash, it is clear from the high-speed photography that the front tires are crushed in the fender wells and therefore they are carrying a substantial part of the impact load. The tires also act to limit the overall length of the crush. The front stabilizer bar was severely deformed, which indicates that it also carried a substantial load. Both the tires and the front stabilizer need to be included in the mesh for an accurate analysis.

Rotate fiber vectors and compute miscellaneous quantities including Hughes-Winget [10] rotation matrix			238 [127]
Hourglass control			366 [77]
Transform force components and update global force vector			60 [24]
For each integration point through thickness	B matrix leading to strains in lamina system	351 [207]	
	Apply Jaumann rate and rotate stresses into lamina system	108 [60]	
	Elastic stress evaluation	22 [9]	
	Global stresses	79 [42]	
	Nodal forces	122 [61]	
TOTAL		682 [379] *(Integ. pnts.)	654 [228] +

Table 1. Operation count for the elastic Hughes-Liu shell in DYNA3D. This operation count includes adds, subtracts, multiplies, and divides and is independent of vectorization. Quantities in square brackets are a subtotal that includes only multiplication operations. The cost is dominated by the cost of the element integration which makes one point integration with multiple thickness points very attractive relative to 2 × 2 reduced selective integration.

A second factor in under predicting the deceleration was interpenetration of the buckles at the front of car in the analysis. Contact between the buckles stiffen a structure, and therefore increases the deceleration. Much more buckling occurred in the analysis than in the experiment for reasons discussed in the previous paragraph. Proportionately more contact (penetration) between the buckles therefore occurs in the analysis than in the experiment, making it difficult for us to assess the significance of the contact between buckles in the experiment. It is probably not significant in comparison to the effect of the tires and front stabilizer, but it is easy to include with the single surface contact algorithm.

Approximately twenty hours of CPU on a Cray/XMP were required for this calculation, but it could be reduced to less than an hour with the proper mesh. As stated earlier, the time step size is controlled by the minimum mesh dimension in explicit finite element program like DYNA3D. Although the car is roughly 3000mm long, some of the elements are less than 1mm long. The geometry of the car is very complicated, and

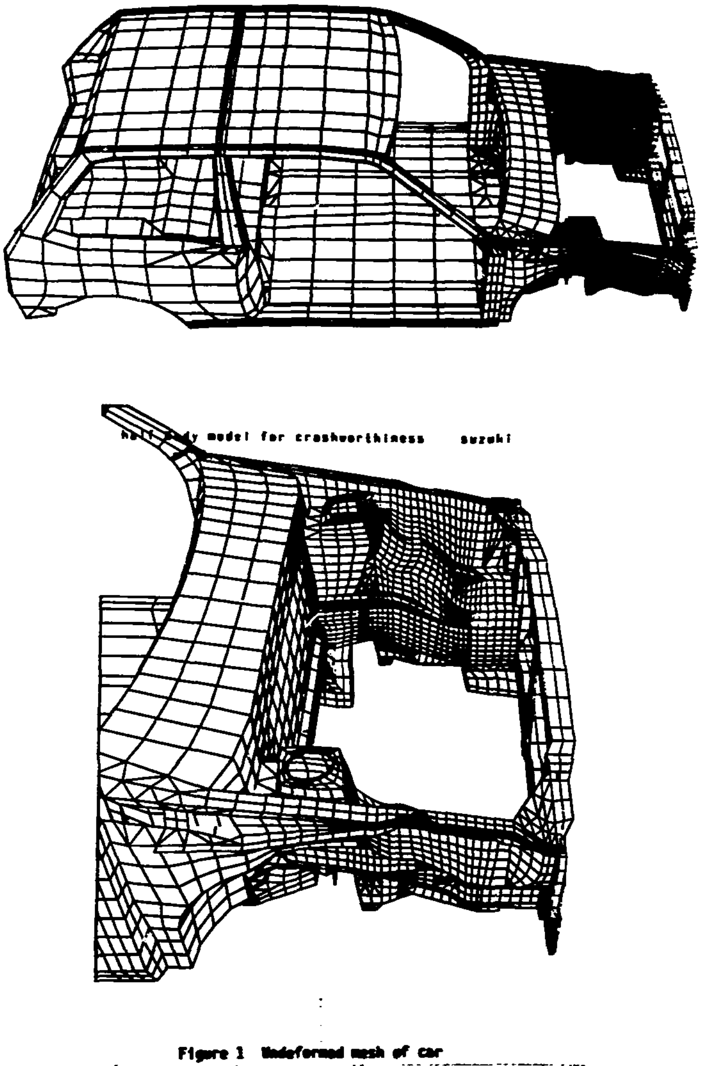

Figure 1 Undeformed mesh of car

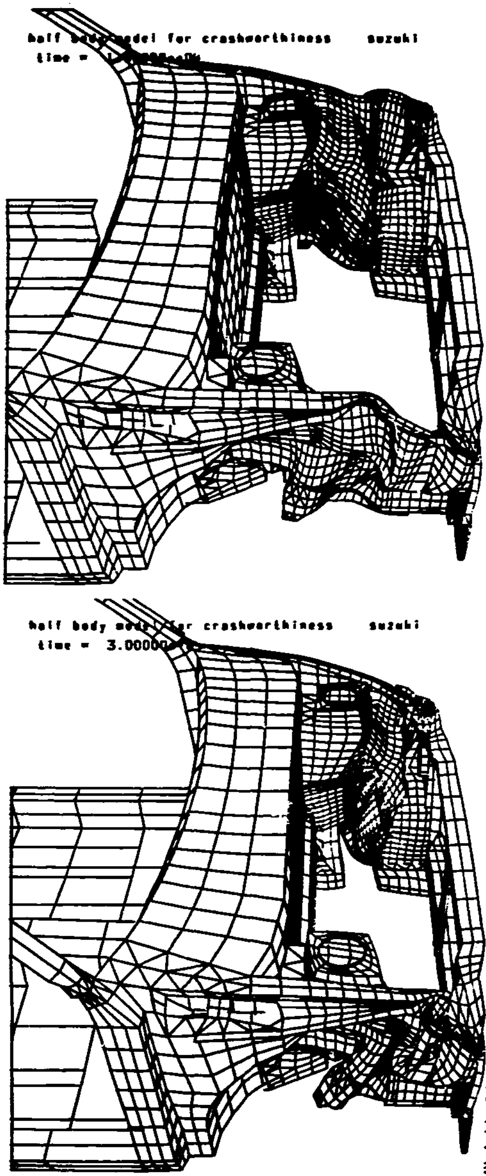

Figure 2. Sequence from a car crash at 10, 20, 30, and 40 milliseconds.

therefore it was convenient to use some triangular elements that are almost slivers to complete the mesh. Given the size of the car, restricting the minimum element size to be at least 20mm would not adversely affect the accuracy of the calculation. A minimum mesh dimension of 20mm would allow a time step twenty times larger than in our analysis, reducing the calculation cost to one hour.

CONCLUSIONS

Given a full vehicle model defined by four thousand shell elements of reasonable dimensions, it can be analyzed on a Cray in less than 1 CPU hour. Furthermore, for a little additional cost, the effects of the suspension parts and car engine can be modeled, many of them as rigid bodies. Even the dummies could be modeled. On a Cray/XMP-48, these calculations could take as little as 15 or 20 minutes with the multi-tasking of all four processors.

We are not suggesting that finite element analysis will entirely replace experiments, but we do believe that it can reduce the number of experiments and improve their design.

ACKNOWLEDGMENT

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

REFERENCES

1. Hallquist, J.O., "Theoretical Manual for DYNA3D," University of California, Lawrence Livermore National Laboratory, Rept. UCID-19401 1982.

2. Hallquist, J.O., Benson, D.J., "DYNA3D User's Manual (Nonlinear Dynamic Analysis of Solids in Three Dimensions)," University of California, Lawrence Livermore National Laboratory, Rept. UCID-19592 Rev. 2, 1986.

3. Goudreau, G.L. and Hallquist, J.O., "Recent Developments in Large Scale Finite Element Lagrangian Hydrocode Technology," Journal of Computer Methods in Applied Mechanics and Engineering 30, 1982.

4. Hughes, T.J.R. and Liu, W.K., "Nonlinear Finite Element Analysis of Shells: Part I. Three-Dimensional Shells," Computer Methods Applied in Mechanics 27, 1981, pp. 331-362.

5. Hughes, T.J.R. and Liu, W.K., "Nonlinear Finite Element Analysis of Shells: Part II. Two-Dimensional Shells," Computer Methods Applied in Mechanics 27, 1981, pp. 167-181.

6. Hughes, T.J.R., Cohen, M. and Haroun, "Reduced and Selective Integration Techniques in the Finite Element Analysis of Plates," Nuclear Engineering Design 46, 1978, pp. 203-222.

7. Hughes T.J.R., Liu, W.K., and Levit, I., "Nonlinear Dynamics Finite Element Analysis of Shells," Nonlinear Finite Element Analysis in Structural Mechanics, Eds. W. Wunderlich, E. Stein, and K.J. Bathe, Springer-Verlag, Berlin, 1981, pp. 151-168.

8. Hughes, T.J.R. and Carnoy, E., "Nonlinear Finite Element Shell Formulation Accounting for Large Membrane Strains," Nonlinear Finite Element Analysis of Plates and Shells, ASME AMD-48, 1981.

9. Hallquist, J.O., "NIKE3D: An Implicit Finite-Deformation, Finite Element Code for Analyzing the Static and Dynamic Response of Three-Dimensional Solids," University of California, Lawrence Livermore National Laboratory, Rept. UCID-18822, Rev. 1, 1984.

10. Hallquist, J.O., Benson, D.J. and Goudreau, G.L., "Implementation of a Modified Hughes-Liu Shell into a Fully Vectorized Explicit Finite Element Code," Proceedings of the International Symposium on Finite Element Methods for Nonlinear Problems, Trondheim, Norway, Aug. 12-16, 1985.

11. Hughes, T.J.R., Ferencz, R.M. and Hallquist, J.O., "Experience With an Element-by-Element Iterative Strategy for Solid Mechanics on a Cray/XMP-48," Proceedings of the Workshop on Scientific Applications and Algorithm Design for High Speed Computing, 1986.

12. Hallquist, J.O., Goudreau, G.L. and Benson, D.J., "Sliding Interfaces With Contact-Impact in Large-Scale Lagrangian Computations," Computational Methods in Applied Mechanics and Engineering, Vol. 51, No. 1/3, 1985.

13. Benson, D.J., Hallquist, J.O., "A Simple Rigid Body Algorithm for Structural Dynamics Programs," International Journal for Numerical Methods in Engineering to appear 1986.

14. Benson, D.J., "The Simulation of Deformable Mechanical Systems Using Vector Processors," Dissertation, The University of Michigan, 1983.

15. Ahmad, S., Irons, B.M. and Zienkiewicz, O.C., "Analysis of Thick and Thin Shell Structures by Curved Finite Elements," International Journal for Numerical Methods in Engineering 2, 1970.

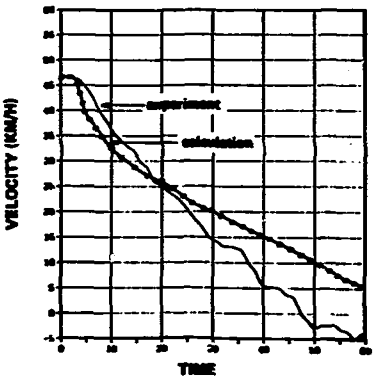

Figure 3. Experimental and calculated velocity curves.