

RECENT PROGRESS IN ALEGRA DEVELOPMENT AND APPLICATION TO BALLISTIC IMPACTS¹

RANDALL M. SUMMERS, JAMES S. PEERY, MICHAEL W. WONG,
EUGENE S. HERTEL, JR., TIMOTHY G. TRUCANO, and LALIT C. CHHABILDAS

Sandia National Laboratories, Computational Physics Research and Development Department
Albuquerque, NM 87185-0819 USA

Summary—ALEGRA is a multi-material, arbitrary-Lagrangian-Eulerian (ALE) code for solid dynamics being developed by the Computational Physics Research and Development Department at Sandia National Laboratories. It combines the features of modern Eulerian shock codes, such as CTH, with modern Lagrangian structural analysis codes. With the ALE algorithm, the mesh can be stationary (Eulerian) with the material flowing through the mesh, the mesh can move with the material (Lagrangian) so there is no flow between elements, or the mesh motion can be entirely independent of the material motion (Arbitrary). All three mesh types can coexist in the same problem, and any mesh may change its type during the calculation. In this paper we summarize several key capabilities that have recently been added to the code or are currently being implemented. As a demonstration of the capabilities of ALEGRA, we have applied it to the experimental data taken by Silsby.

INTRODUCTION

Studies of hypervelocity impact phenomena make heavy use of computer simulations. These simulations are very useful in the analysis of experiments and are indeed necessary to extrapolate beyond the test data base into areas that are prohibitively expensive or otherwise too impractical to explore experimentally. Consequently, considerable effort has been devoted to developing the models and computer codes to treat the strong shocks and large deformations that occur during hypervelocity impact.

In 1990, an effort was launched at Sandia National Laboratories to develop a state-of-the-art code that combined the modeling features of modern Eulerian shock codes, such as CTH [1], with the improved numerical accuracy of modern Lagrangian structural analysis codes. This code, initially called RHALE [2] and now called ALEGRA, uses an arbitrary Lagrangian-Eulerian (ALE) formulation on an unstructured mesh. This formulation allows the user to designate whether material should flow through a stationary mesh (pure Eulerian), whether the mesh should move with the material (pure Lagrangian), or whether the mesh should move independently from the material motion (arbitrary). This latter capability permits a calculation to proceed in Lagrangian fashion until the mesh becomes too highly distorted. At that time, mesh points in the most deformed portion of the mesh are moved to reduce the distortion to acceptable levels. The advantage is that numerical dissipation is avoided until large deformations occur and then is limited to only those regions where there are severe mesh distortions and the mesh must be moved.

1. This work performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract number DE-AC04-94AL85000.

ALEGRA is written predominantly in the C++ programming language to take advantage of object-oriented programming techniques in managing the inherent complexity of the physics being modeled, though we have limited our use of some features of C++ to avoid efficiency problems. However, we have also recognized the utility of incorporating various Fortran-based models and libraries if they best serve our modeling needs and if they are sufficiently mature and robust. In many cases there is little advantage to rewriting such software.

Much progress has been made over the past couple years in ALEGRA development. Its use by the shock wave physics community has begun to accelerate, and several new features have recently been added or will be soon. We have developed a new generic model interface to handle models compliant with the Model Interface Guidelines (MIG) specifications being developed at Sandia for code-independent model implementation. Work is currently in progress to provide hp-adaptivity to the general finite element solution techniques available in ALEGRA. We have implemented dynamic relaxation in ALEGRA as a technique to solve quasi-static problems in which the time evolution is so slow that an equilibrium solution is an excellent approximation at any given time. As part of a demonstration project to investigate novel approaches to coupling large physics codes, we used a new CORBA-compliant programming environment to couple ALEGRA with COYOTE, a large thermal analysis code. We have enhanced ALEGRA to run on several distributed memory parallel computers. We have revised the code data storage architecture to use element and vertex based data structures that enable data locality, allow efficient cache utilization, permit flexible data migration and efficient advection algorithms, and remove the need for use of overloaded operators.

As a demonstration of the capabilities of ALEGRA, we have applied it to the experimental data taken by Silsby [3]. These experiments consisted of $L/D=23$ tungsten rods (nominally 50 and 100 gm) normally impacting a semi-infinite block of steel at velocities ranging from 1.29 to 4.45 km/s. The experimental data consists of cavity depths and diameters. This data set has been used previously [4, 5] for the validation of shock physics analysis packages (hydrocodes). ALEGRA offers superior agreement with experimental data at the lowest impact velocity as compared to CTH and is similar to CTH at higher velocities. We believe that ALEGRA's superior treatment of the material mechanics is responsible for the improvements at low velocities. This improved treatment may lead to a better understanding of defeat mechanisms for low velocity projectiles.

MATERIAL MODELS

Material model capabilities in ALEGRA include common equation of state and constitutive models. Models are available to handle material mechanical properties, fracture, HE burn, and thermal conductivity. A new generic model interface has been developed to handle models compliant with the Model Interface Guidelines (MIG) specifications [6] being developed at Sandia for code-independent model implementation.

The objective of MIG is to facilitate code re-use by establishing certain requirements for numerical material models and host computer codes that will allow the models to be rapidly installed in the host code without changing the model subroutines. Since the basic model is intact, it should be able to run consistently on different host codes provided that the input and output variables are used consistently. To ensure this consistency, MIG defines a dictionary of technical terms so that model and code developers can share a common vocabulary when specifying model variables. The basic idea is to define a standard protocol for the use of material models that captures in a generic way their common interface needs. MIG compliance in ALEGRA has led to direct implementation of several sophisticated models used in other codes or created by independent model developers.

To simplify the process of incorporating new models into ALEGRA, the database structure has been revised from previous versions of the code. A *material* in ALEGRA now essentially means

a collection of models (compliant with MIG or not) for various state and transport variables and the parameters and algorithms that describe those models. On the other hand, *material data* means the values of those property variables describing the state of material in a particular mesh element. The material data are stored in *data cells*, each of which contains all the property variables for a specific material in a specific element. This separation of material models and the data generated by these models leads to more modularity for the models and greater flexibility in installing and using such models.

HP-ADAPTIVITY

Work is currently in progress to provide hp-adaptivity to the general finite element solution techniques available in ALEGRA. H-adaptivity is the *refinement* of the finite element mesh by increasing the number of elements, that is, by changing the characteristic element size, h . Conversely, p-adaptivity is the *enrichment* of finite elements by varying the order of the polynomial degree, p , of the element shape functions. When combined, hp-adaptivity allows both h and p to be manipulated in response to specific problem characteristics, resulting in more efficient spatial discretizations. The combination of these two adaptivity methods improves the accuracy of the solution and increases the resolution of a given initial mesh by refining and/or enriching elements in regions of high numerical error.

Our primary effort in this area is to implement hp-adaptivity for Lagrangian solid dynamics in ALEGRA using object-oriented programming techniques that can be applied in a massively parallel environment. Currently, h-adaptive techniques are being implemented in 2D. Figure 1 shows an example of mesh refinement calculated by ALEGRA, illustrating how a shock propagating

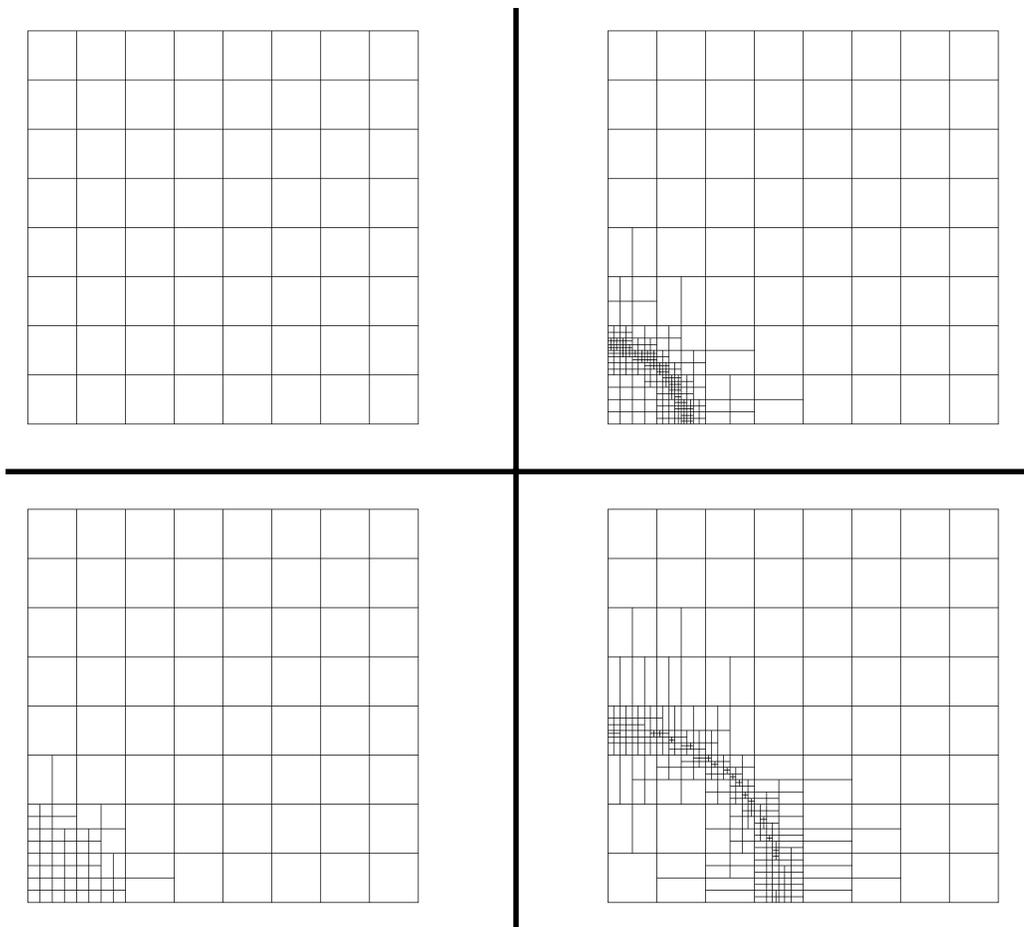


Fig. 1. Mesh refinement example.

outward from the lower left-hand corner would be treated with h-adaptivity. The elements ahead of the shock would refine for increased resolution of the pressure gradient at the front, while elements behind the shock would unrefine as gradients decreased. Note that the number of times an element may be refined must be limited; otherwise refined element sizes would be too small to be practical on the shock front. In the future, h-adaptivity will be extended to 3D and coupled with p-adaptivity techniques. Eventually, we will extend this capability by coupling hp-adaptivity with r-adaptivity (ALE) and applying hp-adaptivity to other physics treated by ALEGRA.

DYNAMIC RELAXATION

Many problems of interest involve highly dynamic transients on very small (microsecond) time scales as well as relatively long periods of time during which the problem is near dynamic equilibrium. A solid dynamics code with the additional capability to solve quasi-static problems efficiently is an ideal tool for these types of problems. Dynamic relaxation is a technique for solving equilibrium or quasi-static problems in continuum and structural mechanics [7, 8]. When dynamic relaxation is used, an artificial damping term is introduced to the equation of motion and small dynamic time steps are used as iterations to solve for the quasi-steady (zero-velocity) state under a constant load corresponding to some point relatively far out in time. Since dynamic relaxation is independent of the spatial discretization method, and since it uses explicit central differencing in time, it lends itself to the modification of typical dynamics codes to solve these types of problems.

To implement dynamic relaxation, an artificial damping term is added to the discretized equation of motion:

$$M_i \frac{v_i^{n+1/2} - v_i^{n-1/2}}{\Delta t^n} + \eta M_i v_i^{n+1/2} = f_i^n + b_i^n \quad (1)$$

where, M is mass, v is velocity, Δt is the time step size, b is the external force (from fields and boundary loads), f is the resultant of all internal forces, subscript i identifies a degree of freedom, and superscript n identifies a time step. The damping coefficient η determines the rate of kinetic energy removal from the system. At convergence, the velocities (and hence the damping term) go to zero, and an equilibrium solution to the problem has been obtained. A quasi-static simulation, is simply a succession of equilibrium solutions for relatively slowly varying external forces.

Dynamic relaxation has been recently implemented in ALEGRA and demonstrated on several simple test problems. As presently implemented, η is a global constant set by the user and applied to all nodes at all times, but extension to values of η as functions of time or space is straightforward. The method was applied in an ALEGRA analysis of the ability of a missile component to withstand an internal hydrostatic pressure of 25,000 psi (172 MPa) during fabrication and testing. Design requirements for the component mandated that it exhibit no plastic deformation for these conditions. However, the calculated equivalent plastic strain through the component at equilibrium indicated that significant deformation would take place under the prescribed loading conditions.

COUPLING WITH OTHER FINITE ELEMENT CODES

We carried out an exploratory project last year to demonstrate the feasibility of using tools based on CORBA (Common Object Request Broker Architecture) [9] to couple the physics in various finite element codes running on a heterogeneous network. CORBA is an emerging standard for distributed object computing sponsored by the Object Management Group (OMG), a

consortium of several hundred software vendors, developers, and users. We initiated our effort as a proof-of-concept to determine whether independently developed codes could be successfully coupled in this manner and applied to problems of interest to the Department of Energy's Defense Programs.

Our motivation was to show that existing applications, though written in different languages and running on separate, perhaps dissimilar platforms, could be easily and efficiently coupled with minimal modifications. The overall goal of this coupling is to obtain an integrated, multi-physics capability and thus capitalize on the significant investment made in these codes over many years.

The Decision Support Systems Department in Sandia's Information Systems Engineering Center is developing a software system called IMPRESARIO (Integrated Multiple Platform for REmote-sensing Simulation And Real-time Interactive Operation), which is based on commercial CORBA-compliant distributed software. IMPRESARIO provides an integrated modeling, simulation, and data visualization framework to facilitate the coupling of independently created models of different physical processes into a unified system. It orchestrates the interactions of these models by providing standard interfaces for data sharing and an enhanced input/output interface.

As an exercise, we coupled ALEGRA with COYOTE [10], a thermal analysis code written in FORTRAN and used to perform conduction and radiation heat transfer calculations, under the IMPRESARIO framework. In a coupled thermomechanical problem, ALEGRA calculates coordinate displacements and mechanical work in response to applied forces, while COYOTE calculates temperature changes in response to heat conduction and enclosure radiation.

Only minor top-level restructuring was required to interface the codes with IMPRESARIO. The physics coupling was accomplished by developing three abstract data types, representing mesh coordinates, temperatures, and mechanical work, and by exchanging objects of these types between the codes at the beginning of each computational step through the IMPRESARIO interface. The codes start with identical representations of the finite element mesh and perform their respective calculations in parallel with each other. After the codes finish a step, ALEGRA sends new mesh coordinates and mechanical work to COYOTE, which updates its mesh and volumetric heating data before starting its next step. Likewise, COYOTE sends new temperatures to ALEGRA, which updates its material states before starting its next step. In this manner, the codes maintain consistency with each other while running in lock step.

We concluded this exploratory project by running the codes on a prototype two-dimensional impact/heat transfer problem. This problem represented an axisymmetric canister undergoing an impact on its center axis with an imposed heat flux on its bottom surface. For the purpose of this demonstration, material properties were modified to artificially match the time scales for the heat transfer with those of the impact.

This project demonstrated that CORBA-based frameworks such as IMPRESARIO hold much promise in successfully coupling a variety of different physics embodied in a diverse collection of existing stand-alone applications within a distributed computing environment. Examples include heat transfer, shock wave physics, quasi-static mechanics, chemical kinetics, and electromagnetics applications running on a heterogeneous network of PCs, workstations, and massively parallel supercomputers. The challenge for the future will be to collect and integrate finite element codes treating these and other physical phenomena into a comprehensive system easily applied by analysts to very complex problems.

PARALLELIZATION

ALEGRA has been enhanced to run on several distributed-memory parallel computers. This was done because we needed the enormous memories and processor speed of massively parallel processor (MPP) computers to analyze large, three-dimensional problems. The memory requirements for our codes scale inversely with the cube of the zone size. For example, if we halve the

mesh size in each direction, then the memory requirement increases by a factor of eight. These codes use explicit, time-stepping integration schemes so the time step scales inversely with the mesh size. For example, if we halve the mesh size, then the code cuts the time step in half. Therefore, the Floating Point Operations (FLOPs) scale as the fourth power of the mesh size. Since the FLOP requirements increase faster than the memory requirements, simply increasing the memory on existing supercomputers is not a good solution to running larger problems because the run time quickly becomes excessive.

ALEGRA runs on distributed-memory parallel computers that are constructed from multiple compute nodes (see Fig. 2). Each compute node has a central processing unit (CPU), memory, and may have special hardware such as a vector processor or input/output processor. A high speed communication network connects the compute nodes. This computer model fits many parallel computers, including Sandia's 1840 compute node Intel Paragon, 1024 compute node nCUBE2, networks of workstations, and even shared-memory parallel computers like the Cray YMP.

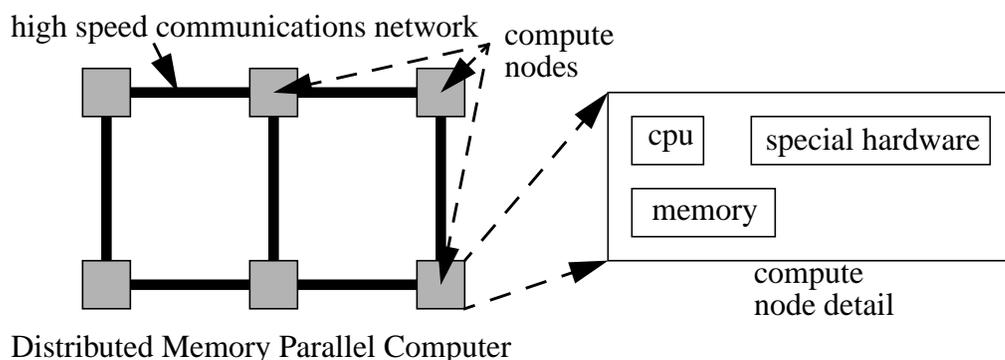


Fig. 2. Distributed memory parallel computer.

We use the lightweight, high-performance Sandia/University of New Mexico Operating System (SUNMOS) [11] on our Intel PARAGON. We designed our software to easily switch the inter-node communication software. We have run with PVM3, MPI, and vendor-specific message passing interfaces.

The data base of a large, three-dimensional problem is too large to fit on any single compute node. Therefore, ALEGRA was designed with the Single Program Multiple Data (SPMD) paradigm, in which the mesh is decomposed into sub-meshes so that each processor gets a single sub-mesh with approximately the same number of elements (see Fig. 3). Good mesh decomposition is

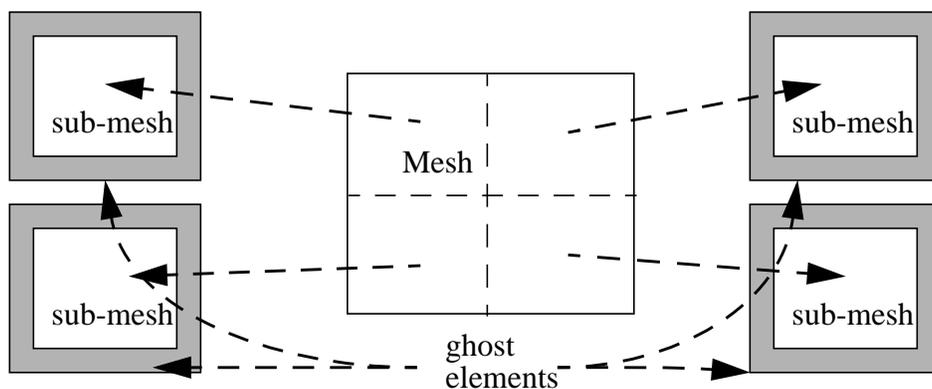


Fig. 3. Mesh decomposition with ghost elements.

important to minimize the memory requirements, balance the work on the compute nodes and minimize the data passed between compute nodes. Whereas rectangular meshes are relatively easy to decompose, subdividing the arbitrary connectivity meshes used by ALEGRA is much more difficult. We use Sandia's Chaco [12] routines to decompose these meshes.

ALEGRA uses one layer of ghost elements around the sub-mesh perimeter for boundary conditions. The database for the ghost elements must be updated once each computational cycle by interprocessor communication. These additional ghost elements represent a parallel processing cost that can be quite large for compute nodes with a small number of elements. For example, for a 1,000-element mesh, approximately half the elements are boundary elements. For large, roughly cubic meshes, the fraction of boundary elements goes to approximately $6/N^{1/3}$, where N is the number of elements, so a million-element mesh will include only about 6% boundary elements. Therefore, we use sub-meshes that fill each compute node's memory to minimize the number of ghost elements. Minimizing the number of boundary elements also minimizes the amount of data passed between compute nodes.

EFFICIENCY

The ALEGRA data storage architecture recently underwent a massive revision. The original version of ALEGRA used "smart arrays" for all of its data storage, each array containing the values of a particular physics or material variable for all elements or vertexes in the problem. C++ overloaded operators were used to perform most of the calculations on these arrays. However, this computational approach resulted in four memory accesses for every floating point operation. On vector processors this approach resulted in relatively good efficiencies, but on more modern cache-based workstation and MPP architectures, this method results in very poor performance due to the mismatch between processor speed and memory bandwidth. Consequently, ALEGRA exhibited very slow run times when compared with other hydrodynamics and solid dynamics codes. Furthermore, this data storage scheme is poorly suited for dynamic load balancing, mesh adaptivity, and data reduction algorithms.

We determined that simply removing all overloaded operators would not result in sufficiently good efficiencies on cache-based processors. It is apparent that these processors require data locality and cache utilization in order to break the memory bandwidth deadlock. In effect, the array data structures represent individual rows in a matrix and the columns are element or node indexes. To achieve good efficiencies one needs to transpose this matrix so that the rows represent an element or node index and the columns represent the data for that element or node.

We have developed, within the C++ language, element- and vertex-based data structures that achieve this transposition. This type of data structure enables data locality and removes the reliance on overloaded operators. In addition, these data structures allow for flexible data migration and efficient advection algorithms. For example, with an element-based data structure, one can advect all the required variables for a single element before moving to the next element, significantly reducing the amount of communication required. Data migration and adaptivity algorithms become much easier to implement when an element or vertex knows everything that must be moved or processed. In effect, these data structures are containers that can be moved with one simple function.

Using these new data structures, we have demonstrated over an order of magnitude increase in speed on several cache-based processors. Table 1 compares the grind times (in $\mu\text{s}/\text{zone}/\text{cycle}$) for ALEGRA version 3 (the current version) with those for the previous version for two test problems on various processors. The head impact problem, simulating a human head impacting a rigid wall, was run for both a 2D Lagrangian mesh and a 3D translation of the 2D mesh. The balls and jacks problem, which tests the ALEGRA interface tracking algorithms for multi-material advection, was run on an Eulerian mesh.

Table 1. Grind times ($\mu\text{s}/\text{zone}/\text{cycle}$)

<i>Code Version</i>	<i>Head Impact Sparc10 (125 Mz) 2D / 3D</i>	<i>Head Impact SP2</i>	<i>Head Impact i860</i>	<i>Balls & Jacks Sparc10 (125 Mz)</i>
Previous (V2)	2420 / 9691	1290	~9000	~2600
Current (V3)	156 / 422	89	621	153

APPLICATION TO LONG ROD IMPACTS

The Silsby experiments consisted of tungsten alloy long rods normally impacting a large (semi-infinite) block of steel. Semi-infinite here means that the block was large enough so that any release waves from the edge of the block would not reach the penetration region until the penetration was complete. The actual rods were threaded over their length to simplify the sabot design for launch. We selected six of the Silsby experiments for simulation by ALEGRA. The selection criteria were based on the completeness of the experimental record and the normality of the impact conditions. The initial conditions for the experiments selected are displayed in Table 2.

Table 2. Silsby experimental data

<i>Experiment</i>	<i>Impact Velocity (km/s)</i>	<i>Rod Length (cm)</i>	<i>Rod Diameter (cm)</i>	<i>Mass (g)</i>
1929	1.29	15.575	0.3433	100.0
5835	2.65	15.575	0.3380	96.96
5839	3.45	15.575	0.3376	96.73
5841	2.37	12.18	0.2638	46.20
5842	3.58	12.18	0.2633	46.02
5844	4.46	12.18	0.2615	45.41

All ALEGRA simulations were completed using an Eulerian mesh for the entire problem domain. This mesh and solution technique may not be the optimal technique for ALEGRA, but we chose to use an Eulerian mesh to allow for a direct comparison with other simulation techniques, such as CTH. The mesh was two-dimensional axisymmetric and consisted of a region of square zones (scaled for 5 zones across the initial rod radius) that encompassed the final cavity shape. The square-zoned mesh was transitioned to a region of arbitrary connectivity to contain adequate material to assure no reflected waves interacted with the cavity development process. A total of ~58000 zones were used for all six simulations. The meshes for the three 50 g projectile simulations were identical, as were the meshes for the three 100 g projectile simulations.

All simulations used identical models for the deviatoric and volumetric response of the materials. A simple perfectly plastic, linearly elastic yield surface was used for both the tungsten and

steel. A linear Us-up (Mie-Gruneisen) model was used for both materials. Parameters for both were identical to those used in the previous study by Hertel [5]. The reference for Kmetyk and Yarrington [4] lists a density of the tungsten that we believe is in error (19 rather than 17 gm/cc). We believe the error is a typo since CTH with the correct value gives results consistent with the published results of Hertel [5].

For comparison purposes, we completed a series of CTH simulations using zoning identical to ALEGRA in the cavity formation region. The CTH simulations were completed using values for the deviatoric and volumetric models identical to those used in ALEGRA.

Table 3 lists the results of the ALEGRA and CTH simulations and comparisons to the experimental data.

Table 3. ALEGRA results for the Silsby experiments

<i>Experiment</i>	<i>Exp. Penetration Depth (cm)</i>	<i>Exp. Cavity Diameter (cm)</i>	<i>ALEGRA Penetration Depth (cm)</i>	<i>ALEGRA Cavity Diameter (cm)</i>	<i>CTH Penetration Depth (cm)</i>	<i>CTH Cavity Diameter (cm)</i>
1929	8.0	1.25	8.43	1.32	10.70	1.2
5835	22.85	1.92	23.72	2.14	24.08	1.8
5839	24.15	2.60	24.70	2.54	24.58	2.6
5841	16.52	1.30	17.93	1.44	18.11	1.3
5842	18.86	2.01	18.16	2.26	19.11	2.0
5844	19.37	2.72	19.52	2.64	19.22	2.6

From an inspection of the data, one can see that ALEGRA is significantly more accurate than CTH at low impact velocities. Both codes do an excellent job of matching the experimental data at higher velocities. As a default, ALEGRA uses a polar stress rate rather than the more typical (for hydrocodes) Jaumann rate and carries material-specific stress deviators. CTH uses the Jaumann rate and stress deviators for the cell only. At the higher impact velocities, the impact pressures are substantially higher than the yield strength of both materials and the interactions are predominately hydrodynamic. Under these conditions, the importance of the stress rate and stress deviator formalism is minimal. At the lowest impact velocity, the impact pressures are similar to the yield strengths of the material and details of the stress treatments become important. We believe that the improved treatment of the stress is responsible for the better agreement with the experimental data that is seen in the ALEGRA simulations at the lowest impact velocity.

CONCLUSIONS

ALEGRA was applied to data taken by Silsby where tungsten alloy long rods ($L/D=23$) impacted semi-infinite steel blocks. Depth of penetration and cavity diameters were recorded experimentally for a range of impact velocities. No time resolved data were taken. The ALEGRA simulations compared very favorably to the experimental data. ALEGRA was more accurate than CTH at predicting the depth of penetration and cavity diameter for the lowest impact velocity. Both codes showed excellent agreement with experimental data at the higher velocities. We believe that the improved treatment of the stress evolution is responsible for this improved agreement with experimental data.

ALEGRA has significantly more capabilities than conventional (Eulerian) hydrocodes and offers a more complete treatment of the material mechanics. As such, it has the potential for allowing substantial advances in our understanding of ballistic impacts and multi-mode ballistic and structural interactions. Continuing enhancements to ALEGRA capabilities, such as hp-adaptivity and parallelization, will further improve our ability to analyze and understand these phenomena by allowing finer resolution in regions of interest and larger calculational meshes.

REFERENCES

1. J. M. McGlaun, S. L. Thompson, and M. G. Elrick, A brief description of the three-dimensional shock wave physics code CTH. SAND89-0607, Sandia National Laboratories, Albuquerque, NM (1989).
2. K. G. Budge and J. S. Peery, RHALE: a MMALE shock physics code written in C++. *Proc. 1992 Hypervelocity Impact Symp.*, Austin, Texas, Nov. 17-20 (1992).
3. G. F. Silsby, "Penetration of semi-infinite steel targets by tungsten long rods at 1.3 to 4.5 km/s. *Proc. 8th Int. Symp. Ballistics*, Orlando, FL (1984).
4. L. N. Kmetyk and P. Yarrington, Cavity dimensions for high velocity penetration events - a comparison of calculational results with data. SAND88-2693, Sandia National Laboratories, Albuquerque, NM (1989).
5. E. S. Hertel, A comparison of the CTH hydrodynamics code with experimental data. SAND92-1879. Sandia National Laboratories, Albuquerque, NM (1992).
6. R. M. Brannon and M. K. Wong, MIG version 0.0 model interface guidelines: rules to accelerate installation of numerical models into any compliant parent code. SAND96-2000, Sandia National Laboratories, Albuquerque, NM (1996).
7. P. Underwood, Dynamic relaxation. In *Computational Methods for Transient Analysis*. T. Belytschko and T. J. R. Hughes, eds., Amsterdam, Elsevier Science Publishers pp. 245-265 (1983).
8. S. Silling, Addition of dynamic relaxation to a hydrocode. Draft report distributed with cover memo dated October 13 (1995).
9. Object Management Group (OMG), *The Common Object Request Broker: Architecture and Specification, Revision 2.0*, OMG Technical Document PTC/96-03-04 (1995).
10. D. K. Gartling and R. E. Hogan, COYOTE - a finite element computer program for nonlinear heat conduction problems; part I - theoretical background. SAND94-1173, Sandia National Laboratories, Albuquerque, NM (1994).
11. A. B. Maccabe, K. S. McCurley, R. Riesen, and S. R. Wheat, SUNMOS for the Intel Paragon: a brief user's guide, *Proc. Intel Supercomputer User's Group. 1994 Annual North America Users' Conf.* (1994).
12. B. Hendrickson and R. Leland, The Chaco user's guide. SAND93-2339, Sandia National Laboratories, Albuquerque, NM (1993).