RHALE: A MMALE Shock Physics Code for Arbitrary Meshes*

James S. Peery and Kent G. Budge
Sandia National Laboratories
Albuquerque, NM 87185, USA
email: jspeery@cs.sandia.gov and kgbudge@cs.sandia.gov

ABSTRACT

This paper describes RHALE, a two and three-dimensional, multi-material, arbitrary Lagrangian-Eulerian (MMALE) shock physics code written in C++. RHALE is the successor to CTH, Sandia's 3-D Eulerian shock physics code, and will be capable of solving problems that CTH cannot adequately address. RHALE employs a three step solution algorithm for the equations of motion: (1) a Lagrangian step capable of solving for either structural or hydrodynamic responses, (2) a remesh step to prevent mesh tangling or to facilitate Eulerian hydrodynamic algorithms, and (3) an Eulerian remap step.

The Lagrangian step is solved using finite elements on an unstructured grid. RHALE incorporates new Lagrangian capabilities, which include arbitrary mesh connectivity, superior artificial viscosity and spurious vorticity control, and improved equations of state

The arbitrary Lagrangian-Eulerian (ALE) algorithm comprises both the remesh and remap steps. The multi-material extension to ALE is termed MMALE and has been generalized for arbitrary grids in both two and three-dimensions in RHALE. The MMALE feature in RHALE provides the accuracy of a Lagrangian code while allowing a calculation to proceed under very large mesh distortions. Coupling an arbitrary connected grid to the MMALE algorithm facilitates modeling of complex shapes with a minimum number of computational cells.

RHALE allows regions of a problem to be modeled with Lagrangian, Eulerian or ALE meshes. In addition, regions can switch from Lagrangian to ALE to Eulerian based on user input or mesh distortion. For arbitrary meshes, new node locations are determined with equipotential schemes. Element quantities are advected with donor, van Leer, or Super-B algorithms. Nodal quantities are advected with second order schemes developed by Benson and Margolin which uses the element advection algorithms. Currently, material interfaces within an element are determined with a SLIC algorithm; however, more accurate two and three dimensional versions of interface trackers are being investigated.

1. Introduction

Theoretical studies of hypervelocity impact phenomena frequently make heavy use of computer simulations. The computational kernel of these simulation codes (frequently called "shock codes" or "wave codes") must be highly efficient and capable of handling strong shocks and large deformations.

The strong shock code CTH [1] has been used extensively in the simulation of hypervelocity impacts. This code uses an Eulerian finite-volume formulation of the equations of motion, includes sophisticated equations of state, and has simple strength and fracture models. Its results are quite accurate in comparison to available experimental data for many classes of problems. It is heavily supported, widely used throughout the DOE and DOD communities, and continues to be enhanced; for example, a version suitable for massively parallel computers (PCTH) is currently in development [2].

However, CTH suffers from the limitations inherent in any shock code based on a purely Eulerian formulation. Since material is permitted to flow through a fixed mesh, advection algorithms are required which introduce numerical dissipation. Contact between surfaces is difficult to model. Since Eulerian formulations generally use a regular mesh, the size of mesh elements can be varied only in rather limited ways which means that element dimensions tend towards the smallest length scale of interest in the calculation. Large regions of empty space must also be included in many grids. The latter two limitations make many interesting calculations prohibitively expensive.

Shock codes based on Lagrangian formulations avoid these difficulties. Since the mesh moves with the material, no advection takes place and the associated numerical dissipation is avoided. Contact surfaces are handled much more effectively than in an Eulerian code. Element connectivities and volumes may generally be quite arbitrary, reflecting very different length scales in different regions of the problem domain. Thus, Lagrangian codes are preferred for problems requiring high numerical accuracy

or where differing length scales are important, or in which large deformations do not take place. However, they fail when large deformations are present, because highly distorted elements lose accuracy or may invert (thus halting the calculation).

The RHALE code uses an arbitrary Lagrangian-Eulerian (ALE) formulation in an effort to get the best of both worlds. Although users can specify a purely Lagrangian or purely Eulerian calculation for portions of the problem domain, the normal mode of operation is for a calculation to proceed in Lagrangian fashion until elements become highly distorted (as measured by various criteria specifiable by the user). At this point, material is permitted to flow between elements in the most deformed portion of the mesh so as to reduce the distortion to acceptable levels. This formulation permits accurate treatment of contact surfaces and has less numerical dissipation than a purely Eulerian calculation (because less advection takes place). Thus, quite large deformations occur without the calculation failing, unlike a purely Lagrangian calculation.

Another innovative feature of the RHALE development project is that the code is being written in C++. We feel that the object-oriented programming paradigm, which C++ supports, is the best approach for the development of such a large code. We are addressing some of the known efficiency problems with C++ through a variety of programming techniques, including reference counting, deferred expression evaluation, and hidden calls to assembly language routines.

RHALE is being developed in parallel with PCTH and the two projects are expected to share much of their coding. However, RHALE represents the next generation of strong shock codes and may eventually replace CTH. It should be capable of performing any calculation that CTH can execute (though perhaps somewhat less efficiently) and will handle additional calculations that CTH cannot currently perform.

2. Lagrangian Algorithms

RHALE supports three types of meshes: pure Lagrangian, pure Eulerian, and ALE. All meshes may have arbitrary connectivity between elements. The user may specify different mesh types for different regions of a problem. For example, the projectile and target region in an impact calculation may be calculated in ALE mode while the far-field target response calculation may take place in pure Lagrangian mode.

We have experimented extensively with a variety of artificial viscosity formulations. None has proven consistently superior to the scalar bulk viscosity formulation that has been used for over thirty years. However, we are continuing to experiment with the spurious vorticity correction methods discussed by Dukowicz and Meltz [3] and may eventually incorporate a spurious vorticity control method in RHALE.

The finite element technology of RHALE consists of uniform-strain quadrilaterals (in 2-D) or hexahedrons (in 3-D). Frame invariance for the constitutive models is achieved by using a corotating frame formulation similar to that of the code PRONTO [4]. The row-summed lumped mass is used to diagonalize the mass matrix (avoiding large matrix inversion) and the time integration is carried out using an explicit central-difference method [5]. Thus, individual time steps are computationally efficient but the maximum time increment is limited by a Courant stability condition.

Since RHALE uses a uniform-strain quadrilateral, spurious zero-energy modes (hourglass modes) exist and must be damped. We provide both the control method used in PRONTO [4], which is applicable to materials with strength, and a version of the Margolin-Pyun method [6], which is applicable to fluids.

RHALE uses the equation of state library being developed for PCTH. This library includes several models: Mie-Gruneisen, ideal gas, JWL for explosives, and SESAME tables. It also has sophisticated strength models for the deviatoric response, including the Johnson-Cook viscoplasticity model and the Johnson-Holmquist brittle material model. Additional models planned for implementation include high explosive programmed burn and reactive burn models, and a fracture model.

RHALE can model 2-D Cartesian, 2-D axisymmetric, and 3-D Cartesian geometries. After much discussion and experimentation, we chose a volume-weighted rather than an area-weighted axisymmetric element [4]. The former has the advantages that the nodal lumped masses are only slightly time dependent and that the element passes a restricted patch test. The latter formulation has the advantages of a closer correspondence between the 2-D Cartesian and axisymmetric formulations and of being better conditioned for implicit methods. Because RHALE currently uses an explicit time integration method, the latter consideration is not important.

3. MMALE Algorithms

The MMALE addition to RHALE involves remeshing to relieve element distortion, and remapping velocities and state variables to the new mesh while conserving global quantities. These steps and their substeps are described below.

3.1 Remesh

The remeshing phase of the MMALE method determines new node locations that will partially alleviate an elements' distortion. In RHALE, a node can be of type Lagrangian, Arbitrary, or Eulerian.

Remeshing is a three step process. First, nodes that meet the criteria for repositioning are tagged. Eulerian nodes are returned to their original previous time cycle location, Lagrangian nodes remain fixed in the remesh/remap phase, and Arbitrary nodes are moved based on distortion criteria. Next, new positions are calculated for tagged nodes. Finally, the actually placement of the tagged nodes is limited to some fraction of their calculated movement.

One set of criteria for moving an Arbitrary node is given by Barton in the HEMP [8] code. Barton's criteria consists of an angle test and a volume test. For Arbitrary nodes, the angles formed by the element edge vectors that originate at the node are calculated. The ideal angle is 90 degrees. Additionally, the volume/area of each element that is connected to a node is determined. These calculations must be performed for each element that surrounds a tagged node. To determine if an Arbitrary node should

move, a user defined minimum angle and minimum to maximum area ratio criteria are used.

In order to perform the remeshing phase of MMALE, one must determine where an Arbitrary node is to be moved. There are many techniques for deciding where to move a node, but one of the most successful and the one used in this MMALE algorithm for two dimensional problems is the method based on Winslow's [9] work with equipotential smoothing. Winslow's method is based on inverting Laplace's equation, in the following form:

$$\alpha x_{\phi\phi} - 2\beta x_{\phi\varphi} + \gamma x_{\varphi\varphi} = 0 \tag{1}$$

$$\alpha y_{\phi\phi} - 2\beta y_{\phi\varphi} + \gamma y_{\varphi\varphi} = 0 \tag{2}$$

where

$$\alpha = x_{\varphi}^2 + y_{\varphi}^2, \beta = x_{\phi}x_{\varphi} + y_{\phi}y_{\varphi}, \text{ and } \gamma = x_{\phi}^2 + y_{\phi}^2. \tag{3}$$

These equations can be approximated with second order central differencing techniques, yielding the new position *x'* as

$$x' = \frac{1}{2(\alpha+\gamma)}\left(\alpha(x_4 + x_8) + \gamma(x_2 + x_6) - \frac{1}{2}\beta(x_3 - x_5 + x_7 - x_9)\right) \tag{4}$$

where

$$x_{\phi} = \frac{1}{2}(x_4 - x_8)\ ,\ x_{\varphi} = \frac{1}{2}(x_2 - x_6)\ , \text{ and } x_{\phi\varphi} = \frac{1}{4}(x_3 - x_5 + x_7 - x_9) \tag{5}$$

and equations for the y coordinate are similarly developed. These equations can be applied for nodes connected to four elements. For an arbitrary number of connections, a method developed by Budge [10] is used.

An iterative procedure is used to find the *x* and *y* coordinates for nodes that satisfy Laplace's equation or the method developed by Budge. Jacoby [11]iteration can be used to solve these equations. However, MMALE algorithms are interested in making small changes to the mesh and therefore these equations are generally solved once per remap step.

3.2 Remap

The explicit relocation of nodes creates an advection problem. Since time is not involved, the advection problem simplifies to a remapping problem. The remap phase consists of determining volumes fluxes, determining material fluxes from volume fluxes through the use of an interface tracker, and advecting material variables and velocities. The various aspects of the remap phase are discussed in the following sections.

3.2.1 Determining Volume Fluxes

The first step in any advection scheme is to determine the volume fluxes created by nodal movement. When a node is moved, volume fluxes are generated through the faces/sides of the elements of which the node is connected. The volume flux through an element face is given by the change in volume produced by the movement of nodes on that face. With this definition of volume flux, the new volume of an element is given by

$$V_{new} = V_{old} + \sum_{i=1}^{ns} \Delta V_i \tag{6}$$

where *i* refers to an element's face and *ns* refers number of element faces (four in two dimensions and six in three dimensions).

3.2.2 Material Fluxes

In MMALE, a simulation generally begins with single material elements. If it remains Lagrangian, the elements remain single material; however, for problems of interest, elements quickly become distorted and remeshing/remapping is used to relieve distortion. The remeshing/remapping phase creates multi-material elements and thus, the volume of each material within the volume fluxes must be determined.

Currently, RHALE determines material volume fluxes with the SLIC [12] interface tracking algorithm. This algorithm places materials within a cell in an order from left to right by determining the materials contained in neighboring cells and assuming a planar interface between materials. The material order determines the precedence of materials to be used in the material makeup of the volume flux. In the future, Youngs' [13] interface tracker will be added to RHALE.

3.2.3 Element Centered Advection

Once the nodes have been moved, the element centered variables are advected to their new locations. Isotropic advection assumes that the material is advected through all faces of the element simultaneously. While this discussion assumes that advection occurs isotropically, RHALE actually makes one-dimensional remap sweeps through the mesh to aid in corner coupling.

Advection algorithms are expressed in volume and mass coordinates. Using the above definition of volume flux, a newly advected element centered variable is given by

$$f_j^n = \frac{f_j^o V_j^o + \sum_{i=1}^{ns} \tilde{f}_i \Delta V_j}{V_j^n} \tag{7}$$

where the $\tilde{f}_i$'s are determined by the type and order of the advection algorithm and V represents either volume or mass. Most intensive quantities, for example, energy density, are fluxed with mass.

The simplest advection scheme is a first order method because it does not involve evaluating derivatives. However, it is also the most diffusive. The advecting material is assumed to carry the average value of an element centered variable from which the material originated. This is analogous to first order upwinding in finite difference methods. For a first order method, the f_i's are given by

$$f_i = \begin{cases} f_n \ if \ \Delta V_{ji} < 0 \\ f_i \ if \ \Delta V_{ji} > 0 \end{cases} \tag{8}$$

where n is the neighbor's value.

Two second order advection methods are available in RHALE, van Leer [14] and Super-B [15]. Both are based on slopes among the donor, acceptor and behind elements. For the van Leer method, this relationship is shown in the figure below where the x-axis represents volume or mass and the y-axis represents a quantity to be fluxed. The three slopes are given by

$$s1 = (2(f_{i+1} - f_i))/X_i \tag{9}$$

$$s2 = (2(f_i - f_{i-1}))/X_i \tag{10}$$

$$s3 = (2(f_{i+1} - f\ \ ))/(X_{i+1} + X_i) + (2(f_i - f_{i-1}))/(X_i + X_{i-1}) \tag{11}$$

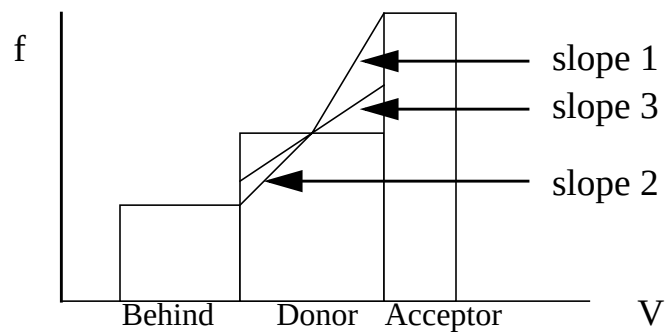

Figure 1: Slopes Used in van Leer Advection Algorithm

For the van Leer scheme, the slope used is given by

$$slope' = \begin{cases} 0 \ if \ \left| \frac{sign(s1) + sign(s2) + sign(s3)}{3} \right| \neq 1 \\ min(s1, min(s2, s3)) \end{cases} \tag{12}$$

and the value used for $\tilde{f}$ is given as

$$\tilde{f}_i = \begin{cases} f_n + \frac{slope'}{2}((V_i + V_n) - \Delta V_n) \ if \ \Delta V_i = 0 \\ f_i + \frac{slope'}{2}((V_i + V_n) - \Delta V_i) \ if \ \Delta V_i > 0 \end{cases} \tag{13}$$

3.2.4 Vertex Centered Advection

Vertex centered advection is very similar in concept to element centered advection and is required for advecting nodal quantities such as momentum. For vertex centered variables, however, a staggered grid is used so that vertices lie at the center. For a logically connected grid, donor, acceptor and behind nodes are known for each direction and staggered fluxes and advection quantities can be determined. Amsden [16] developed the YAQUI algorithm for this type of mesh. However, determining staggered mesh quantities can be very expensive. In addition, for an arbitrary mesh, the staggered mesh can have very odd shapes and advection quantities are difficult if not impossible to calculate. Attempts to eliminating these problems have resulted in sev-

eral element centered methods for advecting vertex centered variables.

In RHALE there are three element centered options for nodal variables (vertex centered), SALE [17], SHALE [18], and HIS [19]. All of these methods project nodal variables to element centers, advect the new element centered variables with the element centered methods described above, and project these values back to the nodes. SALE simply averages nodal variables to calculate element centered quantities and thus is first order. SHALE averages both the nodal variable and its derivatives and thus is second order. Both SALE and SHALE are monotonic with respect to the element centered advection but not with respect to the nodal variables. This can lead to new maximums or minimums for nodal quantities. Benson's HIS algorithm is second order and monotonic. The HIS algorithm places nodal variables at element centers, advects the element quantities and assembles the values back at the nodes. The HIS method has been extended to arbitrary meshes in RHALE.

4. C++

To facilitate code development and maintenance, we have chosen to write RHALE in C++ [20]. We feel that the object-oriented programming paradigm provides superior reliability, reusability, and portability. Much of our coding is being developed in common with PCTH, which is also being written in C++.

Our basic classes represent scalar, vector, and tensor fields. Various operations between objects of these classes are represented by the usual C operators, so that, for example, the equation

$$\vec{f} = \vec{\nabla}_{\vec{x}} \bullet \mathbf{T} + \vec{b} \tag{14}$$

can be coded as

```
f = Div(T,X) + b;
```

where `X`, `f` and `b` are objects of a vector field class and `T` is an object of a tensor field class. These objects contain both the data representing the field and a pointer to a data structure describing the mesh topology. Thus, indexes and loops are hidden in the class definition.

We find that considerable care is required in developing the basic classes to avoid inefficiencies. For example, many unnecessary copy operations take place unless reference counting techniques are used. In memory-critical applications, unacceptably large amounts of memory may be allocated for intermediate results of expression evaluation unless deferred expression evaluation is also implemented. Finally, the large number of memory allocation/deallocation operations that are necessary for field operations can be very expensive on certain architectures or operating systems; it is then necessary to take control of heap management, using the C++ language features provided for that purpose.

References

1. J.M. McGlaun, S.L. Thompson, and M.G. Elrick, "CTH: A three Dimensional Shock Wave Physics Code," Int. J. Impact Engng, Vol. 10, 1990, pp. 351-360.
2. A. Robinson et al., "Massively Parallel Computing, C++, and Hydrocode Algorithms," presented at the ASCE 8th Conference on Computing, Dallas, TX, June 10-12, 1992.
3. J.K. Dukowicz and B.J.A. Meltz, "Vorticity Errors in Multidimensional Lagrangian Codes." Journal of Computational Physics **99**, 1992, pp. 115-134.
4. L.M. Taylor and D.P. Flanagan, "PRONTO-2D: A Two-Dimensional Transient Solid Dynamics Program," Sandia National Laboratories, SAND86-0594.
5. T.J. Hughes, The Finite Element Method, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.
6. L.G. Margolin and J.J. Pyun, "A Method for Treating Hourglass Patterns," in Proceedings of the Fifth International Conference on Numerical Methods in Laminar and Turbulent Flow, Montreal, Canada, 1987.
7. D.J. Benson, "An Efficient, Accurate, Simple ALE Method for Nonlinear Finite Element Programs." Computer Methods in Applied Mechanics and Engineering **72**, 1989, pp. 309-350.
8. R.W. Sharp and R.T. Barton, "HEMP Advection Model," Lawrence Livermore National Laboratories, UCID-17609.
9. A.M. Winslow and R.T. Barton, "Rescaling of Equipotential Smoothing," Lawrence Livermore National Laboratories, UCID-19846.
10. K.G. Budge, "ALE Shock Calculations Using a Stabilized Serendipity Rezoning Scheme." presented at 1991 APS Topical Conference on Shock Compression of Condensed Matter, 1991.
11. W.H. Press, B.F. Flannery, S.A. Teukolsky, and W.T. Vetterling, Numerical Recepies, Cambridge University Press, Cambridge, pp. 653-655, 1986.
12. W.F. Noh and P. Woodward, 1976. "SLIC (Simple Line Interface Calculation),"in Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics, Twente University, Enschede, June 28 - July 2, 1976.
13. D. L. Youngs, "An Interface Tracking Method for a 3D Eulerian Hydrodynamics Code," AWRE/44/92/35.
14. B. Van Leer, Computing Methods in Applied Sciences and Engineering, VI, Elsevier Science, New York, 1984.
15. R. Christensen, Lawrence Livermore Laboratory, Livermore, CA., private communication,1991.
16. A.A. Amsden and C.W. Hirt, Los alamos Scientific Report LA-5100 (unpublished).
17. A.A. Amsden, H.M. Ruppel, and C.W. Hirt, Los alamos Scientific Report LA-8095 (unpublished).
18. L.G. Margolin and C.W. Beason, "Remapping on the Staggered Mesh," presented at the Fifth NECDC, Boulder, CO., October 11-14, 1988.
19. D.J. Benson, "Momentum Advection on a Staggered Mesh," Journal of Computational Physics, 100, pp. 143-162.
20. M.A.Ellis and B.Stroustrup, The Annotated C++ Reference Manual, Addison-Wesley Publishing Company, Reading, MA, 1990.

*This work performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract number DH-AC04-76DP00789.