INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **31**: 113–120 (1999)

THE NUMERICAL SIMULATION OF STRONGLY UNSTEADY FLOW WITH HUNDREDS OF MOVING BODIES

RAINALD LÖHNER[a,*], CHI YANG[a], JOSEPH D. BAUM[b], HONG LUO[b], DANIELE PELESSONE[c] AND CHARLES M. CHARMAN[c]

[a] *GMU/CSI, George Mason University, Fairfax, VA 22030-4444, USA*
[b] *Science Applications International Corporation, McLean, VA 22102, USA*
[c] *General Atomics, San Diego, CA, USA*

SUMMARY

A methodology for the simulation of strongly unsteady flows with hundreds of moving bodies has been developed. An unstructured grid, high-order, monotonicity preserving, ALE solver with automatic refinement and remeshing capabilities was enhanced by adding equations of state for high explosives, deactivation techniques and optimal data structures to minimize CPU overheads, automatic recovery of CAD data from discrete data, two new remeshing options, and a number of visualization tools for the preprocessing phase of large runs. The combination of these improvements has enabled the simulation of strongly unsteady flows with hundreds of moving bodies. Several examples demonstrate the effectiveness of the proposed methodology. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: moving bodies; numerical simulation; unsteady flows

1. INTRODUCTION

A number of engineering applications require the prediction of strongly unsteady flow fields interacting with many (possibly thousands) of moving bodies. Examples include flare and submunition deployment, fragmentation, debris impact, wall breach and spalation. In order to carry out simulations of this kind, the following requirements must be placed on the methodology used:

- Accurate simulation of strongly unsteady flows with discontinuities;
- Ability to deal with extreme discontinuites in density ($> 1{:}1200$) and pressure;
- Ability to handle several complex equations of state simultaneously;
- Tracking and update of many independently moving bodies;
- Correct treatment of body–body interactions, such as contact, spalation, etc.;
- Ability to treat topological changes via automatic recovery of discrete data and patching to analytical data;
- Rapid and error-free problem set-up and grid generation; and
- Meaningful data reduction and visualization.

* Correspondence to: GMU/CSI, George Mason University, Fairfax, VA 22030-4444, USA.

CCC 0271–2091/99/170113–11$17.50

Over the years, we have developed and applied (see, e.g. [1–7]) a methodology to meet most of these requirements. The numerical techniques used for the computational fluid dynamics (CFD) aspects are based on unstructured finite element techniques, using tetrahedral meshes, to treat complex geometries and/or physics. Extensive use is made of FEM–FCT [8] or other monotonicity preserving schemes [9] to handle transient discontinuities. An arbitrary Lagrangean–Eulerian (ALE) frame of reference is employed for all equations, enabling the use of moving grids. Adaptive mesh refinement [10] is used extensively to track shocks and other discontinuities. Regions of elements distorted due to mesh motion are regridded automatically [11] using an advancing front technique [12–15]. For the computational structural dynamics (CSD) aspects, rigid bodies are treated through either six degrees of freedom (6DOF) integrators, or a loose coupling [5,6,16] to impact codes with optimized contact algorithms [17–23].

The current thrust is directed towards complex equations of state, better data structures for many-body applications, speed via deactivation of edges, automatic handling of topology changes, better remeshing strategies for moving bodies, improved preprocessing and visualization tools, and validation through comparison with experimental data. This paper reports on progress made in each one of these areas, which has, synergistically, *resulted in the ability to simulate, on a fairly routine basis, flows that interact with hundreds of moving bodies.*

2. EQUATIONS OF STATE

Most high explosives are well-modeled by the Jones–Wilkins–Lee (JWL) equation of state, given by

$$p = A\left(1 - \frac{\omega}{R_1 v}\right)e^{-R_1 v} + B\left(1 - \frac{\omega}{R_2 v}\right)e^{-R_2 v} + \omega\rho e, \tag{1}$$

where v denotes the relative volume of the gas

$$v = \frac{V}{V_0} = \frac{\rho_0}{\rho}. \tag{2}$$

Afterburning is modeled by adding energy via a burn coefficient λ, which is obtained from

$$\lambda_{,t} = ap^{1/6}\sqrt{1-\lambda}, \tag{3}$$

where $\lambda = 0$ for the unburned state and $\lambda = 1$ for the fully burned material. After updating λ, the energy released is added as follows:

$$(\rho e)\big|^{n+1} = (\rho e)\big|^{n} + \rho Q(\lambda^{n+1} - \lambda^{n}), \tag{4}$$

where Q is the afterburn energy. Compared with the five unknowns required for the Euler equations with an ideal air equation of state, we require an additional two: the burn fraction b to determine which part of the material has ignited, and the afterburn coefficient λ. Observe that in the expanded state ($v \to \infty$), the JWL equation of state reduces to

$$p = \omega\rho e = (\gamma - 1)\rho e, \tag{5}$$

where the correlation of ω and γ becomes apparent. The transition to air is made by comparing the density of air with the density of the high explosive. Given that $A \gg B$, the decay of the first term in Equation (1) with increasing v is much faster. This implies that as v increases, we have

$$p \to Be^{-R_2 v} + \omega \rho e. \tag{6}$$

The mixture of high explosive and air is considered as air when the effect of the B term may be neglected, i.e.

$$\frac{p}{p_{\mathrm{cj}}} = \epsilon = Be^{-R_2 v}, \tag{7}$$

where p_{cj} denotes the Chapman–Jouget pressure and $\epsilon = O(10^{-3})$.

3. DEACTIVATION ZONES

Consider a typical explosion simulation. The major portion of CPU time is required to simulate the burning material. This is because the pressures are very high, and so are the velocities of the fluid particles. Once the material has burned out, one observes a drastic reduction of pressures and velocities, which implies a dramatic increase in the allowable time step. Even though shocks travel much larger distances, this postburn diffraction phase takes less CPU time than the burn phase. In order to speed up the simulation, the portions of the grid outside the detonation region are deactivated. The detonation velocity provides a natural speed beyond which no information can travel. Given that the major loops in an unstructured grid flow solver are processed in groups (elements, faces, edges, etc.) for vectorization, it seemed natural to deactivate not the individual edge, but the edge group. In this way, all inner loops can be left untouched, and the test for deactivation is carried out at the group level. The number of elements in each edge, face or element group is kept reasonably small ($O(128)$) in order to obtain the highest percentage of deactivated edges without compromising performance. The points are renumbered according to their ignition time assuming a constant detonation velocity. In this way, the work required for point-loops is minimized as much as possible. The edges and points are checked every five to ten time steps and activated accordingly. This deactivation technique leads to considerable savings in CPU at the beginning of a run, where the time step is very small and the zone affected by the explosion only comprises a small percentage of the mesh.

4. DYNAMICS AND INTERACTION OF MANY BODIES

The dynamics and interaction of many bodies pose a number of challenging problems. Two areas may be singled out as particularly critical:

- Contact algorithms, and
- Optimal data structures for the handling of body motion.

4.1. Contact algorithms

For simple store separation problems (see [24] for a cross-section of current capabilities), contact rarely appears as an issue. However, for applications with many moving bodies, contact between the bodies is highly likely, and must, therefore, be accounted for. In order to treat contact, we employ a loose coupling algorithm [16], linking the CFD code to an

impact CSD code. CSD codes for impact have a long tradition of efficient contact algorithms, and it seemed, therefore, prudent to follow this path. The bodies in the flow field are discretized using unstructured 8-noded brick finite elements, and are treated as either rigid, elastic or elasto-plastic materials. The contact algorithm intrinsic in these codes then accounts for body–body interactions. Any contact algorithm is composed of two phases:

(a) detection of proximity/penetration, and
(b) enforcement of non-penetration.

The proximity/penetration tests are typically handled using a bin data structure. This works well for typical CSD meshes that are characterized by being of fairly uniform size. To date, the enforcement of penetration was handled by a repulsive spring system or Lagrange multipliers [17–23]. Both approaches did not guarantee non-penetration. While this is of no consequence for CSD applications, it complicates coupled CFD/CSD. Ideally, a gap should be left between contacting elements so that a fluid volume can still be inserted there. As the CFD surface follows the CSD surface, any penetration of CSD faces will lead to negative volumes in the CFD mesh. A common attempt to circumvent this difficulty is by activating the repulsive forces between close, potentially contacting faces before the actual penetration takes place. This is done by specifying a 'minimum safe distance' for contact. While this approach works in most cases, we have found cases where the 'minimum safe distance' had to be set to physically meaningless values in order to avoid penetration. We believe this is an area of research that deserves further study.

4.2. Optimal data structures

Traditional applications of fluids interacting with moving bodies only considered a limited number of moving bodies. For example, store separation applications seldomly include more than one to five moving bodies [24]. Computing with such a low number of moving bodies implies that, when doing force evaluations, point movement, etc., the CPU penalty incurred by careless coding is barely noticeable. The situation reverses when one faces problems with hundreds of moving bodies. As an example, consider the evaluation of body forces and moments from surface pressures. A simplistic way to evaluate these would be to loop over all the faces, filter the ones belonging to a given body, and then sum up forces and moments for this reduced list of faces. This implies `nbody`-loops over the faces, where `nbody` is the number of bodies in the flow field. It is clear that for a large number of bodies, the CPU penalty incurred by such a procedure is severe. A sample run with more than 200 bodies and several million elements indicated that body motion required 40% of the overall CPU time. For this reason, a number of data structures were implemented to arrive at optimal speeds for the handling of body motion. Among these, the following are of particular relevance:

– linked lists to gain rapid access to the faces comprising a body (force evaluation);
– linked lists to gain rapid access to the points comprising a body (movement of points on a body);
– colored list of edges in the moving mesh portion (mesh movement).

With these data structures, the overhead for force, moment and rigid body point movement calculations could be reduced to less that 5% of the overall CPU time.

5. TOPOLOGY CHANGES AND AUTO-CADDING

For simulations of fragmenting materials or wall breach, the topology of the computational domain will change during the course of a simulation. It is, therefore, necessary to devise algorithms to recover, from the updated 'wetted surface' of the structural code, the new surface definition of the flow field. This recovery is then followed by global remeshing and interpolation before the coupled CFD/CSD run can proceed. One CAD database change consists of the following operations (see Figure 1):

- Remove from the CAD database any data associated with the 'wetted surfaces'; this yields a reduced CAD database, denoted by Set 1;
- Recover discrete surface patches, lines and end-points [14,15] from the updated 'wetted surface';
- Impose the desired boundary conditions and mesh parameters for the updated 'wetted surface' patches; this yields a 'wetted surface' CAD database, denoted by Set 2;
- Merge Set 1 and Set 2.

6. REMESHING STRATEGIES

Any field simulation with boundaries that undergo severe movement will need some form of mesh adjustment to cope with the change of spatial resolution dictated by the geometry and the physics. Non-conforming grids (e.g. Cartesian grids) do this by refining and coarsening the grid with a subsequent adjustment of boundary conditions at the surface. Overlapping grids change the interpolation information in the overlap zones. Unstructured, conforming grids typically remove deformed elements and remesh the voids thus created. To date, we have used a combination of local and global remeshing to solve this class of problems. Over the last year, it has become apparent that two other ways of remeshing are also very useful:

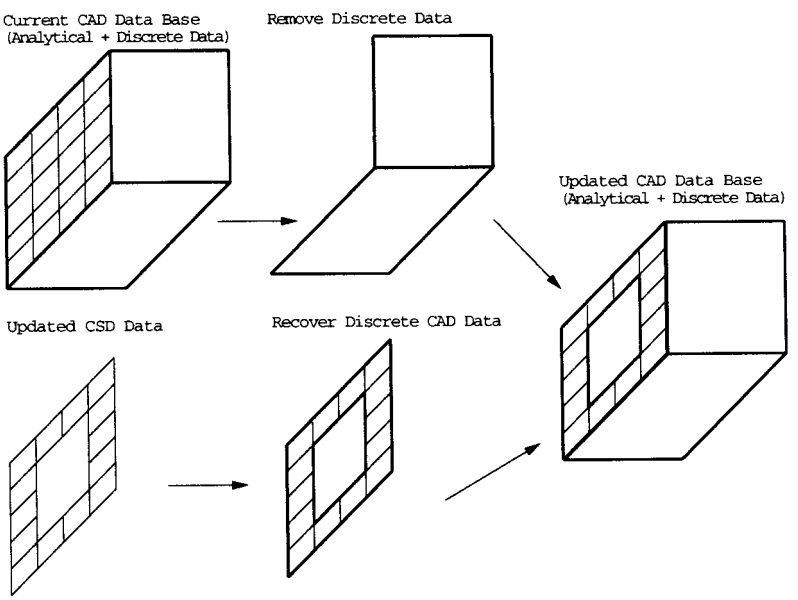


Figure 1. Automatic update of CAD database.

(a) remeshing only the ALE region, and
(b) excluding the highly distorted Navier–Stokes region grids from remeshing.

The first option is particularly attractive if the number of deforming (moving) mesh layers surrounding bodies in motion comprises only a fraction of the total volume. Global remeshing is comparatively expensive in this case, with no additional advantage. The second option is essential for RANS simulations. For this class of problems, the regions of highly stretched elements close to the bodies in motion are moved in a rigid fashion, just as the surface points. These two remeshing options have improved dramatically the ability to simulate problems with many moving bodies.

7. PREPROCESSING TOOLS

The definition of boundary conditions, surface data and desired elements size and shape in space is tedious enough for problems with few moving bodies. It is onerous for hundreds of moving bodies. Errors in the input data become impossible to discern without a graphical, intuitive preprocessing tool. We have, therefore, implemented into our FECAD preprocessing tool a number of features to define and check such items as surface geometry, boundary conditions, body number, background sources, size attached to CAD data, etc. It is hard to underestimate the benefit of this graphical preprocessing toolkit. Suffice it to say that without it, it would have been impossible to conduct the calculations performed.

8. NUMERICAL EXAMPLES

The proposed algorithms were used to conduct a series of fragmentation studies. In all cases, the high explosive was modeled using a JWL equation of state and the flow solver option employed was edge-based FEM–FCT [8,9].

8.1. Weapon fragmentation

As a first example, we consider a weapon fragmentation experiment conducted recently. The weapon used consisted of a thick wall cylinder, serrated into 32 rows of fragments, 16 fragments per row, for a total of 512 'small' fragments, and thick nose and tail plates. While the cutting pattern was identical for all rows, the pattern was rotated between rows by angles that produced no symmetric pattern. Thus, the whole weapon had to be modeled. Each fragment weighed about 380 g. The serrated weapon had tabs that kept the minimum distance between the fragments to 0.5 mm, while the average size of each fragment was a 1–4 cm per side. In contrast, the room size was several meters. This huge disparity in dimensions required the use of sources attached to each flying body in order to ensure a uniform, high-resolution mesh about each fragment. Plate 1(a) shows the computational domain as well as snapshots of the surface grid employed. Each of the fragments is treated as a separate, freely flying rigid body whose velocity and trajectory were determined by integrating the six ordinary differential equations describing the balance of forces and moments. Plate 1(b)–(i) show a sequence of snapshots of pressure, detonation products velocity, and fragment and mesh velocity, at several times. The pressure results are shown for a planar cut. Note that this is not a plane of symmetry. The figures show the detonation wave, the accelerating fragments, and the sharp

capturing of the shock escaping through the opening gaps. The results indicate that acceleration to the final velocity takes about 120 μs. The predicted velocities fall within the 10% error band of the experimental measurements. Given the complexity of the physical phenomena being modeled, such a correlation with experimental data is surprisingly good. Typical meshes for this simulation were of the order of 3–4 Mtets. The simulation was carried out on an eight processor SGI Origin 2000, and took approximately 2 weeks. For more details, see [7].

8.2. Generic weapon fragmentation

This second example shows a fully coupled CFD/CSD run. The structural response was calculated using GA-DYNA [21–23]. The structural elements were assumed to fail once the average strain in an element exceeded 60%. At the beginning. the CFD domain consisted of two separate regions. These regions connected as soon as fragmentation started. In order to handle narrow gaps during the break-up process, the failed CSD elements were shrunk by a fraction of their size. This alleviated the time step constraints imposed by small elements without affecting the overall accuracy. The final break-up led to approximately 1200 objects in the flow field. Plate 2 shows the fluid pressure and CSD surface velocity at three different times during the simulation. Typical meshes for this simulation were of the order of 8 Mtets and the simulations required of the order of 50 h on the SGI Origin 2000 running on 32 processors.

9. CONCLUSIONS AND OUTLOOK

A methodology for the simulation of strongly unsteady flows with hundreds of moving bodies has been developed. The numerical techniques used for the computational fluid dynamics (CFD) aspects are based on unstructured finite element techniques, using tetrahedral meshes, to treat complex geometries and/or physics. Extensive use is made of FEM–FCT or other monotonicity preserving schemes to handle transient discontinuities. An arbitrary Lagrangean–Eulerian (ALE) frame of reference is employed for all equations, enabling the use of moving grids. Adaptive mesh refinement is used extensively to track shocks and other discontinuities, and regions of elements distorted due to mesh motion are regridded automatically. High explosives are modeled using the Jone–Wilkins–Lee (JWL) equation of state. A deactivation technique has been implemented to minimize the CPU requirements during the burning phase of the explosive. Optimal data structures are included to minimize CPU overheads for problems with many bodies. For the computational structural dynamics (CSD) aspects, rigid bodies are treated through either six degrees of freedom (6DOF) integrators, or a loose coupling to impact codes. An automatic CAD data update technique is used to treat situations with changing topologies. Two new remeshing options for the class of problems considered here have significantly decreased CPU requirements. In addition, several preprocessing tools have been developed to define and check input data. This, as it turns out, is one of the keys to successfully conducting simulations with hundreds of moving bodies.

The combination of these different improvements has resulted in the ability to simulate flows that interact with hundreds of moving bodies.

Future developments will center on more sophisticated equations of state, improved diagnostics and simplified data reduction.

ACKNOWLEDGMENTS

This work was partially supported by DSWA, with Drs Michael Giltrud and Darren Rice as the technical monitors, as well as AFOSR, with Dr Leonidas Sakell as the technical monitor.

REFERENCES

1. J.D. Baum and R. Löhner, 'Numerical simulation of shock interaction with a modern main battlefield tank', *AIAA-91-1666*, 1991.
2. J.D. Baum, H. Luo and R. Löhner, 'Numerical simulation of a blast inside a Boeing 747', *AIAA-93-3091*, 1993.
3. J.D. Baum, H. Luo and R. Löhner, 'Numerical simulation of blast in the World Trade Center', *AIAA-95-0085*, 1995.
4. J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman, 'A coupled fluid/structure modeling of shock interaction with a truck', *AIAA-96-0795*, 1996.
5. J.R. Cebral and R. Löhner, 'Fluid–structure coupling: extensions and improvements', *AIAA-97-0858*, 1997.
6. J.R. Cebral and R. Löhner, 'Conservative load projection and tracking for fluid–structure problems', *AIAA J.*, **35**, 687–692 (1997).
7. J.D. Baum, H. Luo and R. Löhner, 'The numerical simulation of strongly unsteady flows with hundreds of moving bodies', *AIAA-98-0788*, 1998.
8. R. Löhner, K. Morgan, J. Peraire and M. Vahdati, 'Finite element flux-corrected transport (FEM-FCT) for the Euler and Navier–Stokes equations', *Int. J. Numer. Methods Fluids*, **7**, 1093–1109 (1987).
9. H. Luo, J.D. Baum and R. Löhner, 'Edge-based finite element scheme for the Euler equations', *AIAA J.*, **32**, 1183–1190 (1994).
10. R. Löhner and J.D. Baum, 'Adaptive *H*-refinement on 3D unstructured grids for transient problems', *Int. J. Numer. Methods Fluids*, **14**, 1407–1419 (1992).
11. R. Löhner, 'Three-dimensional fluid–structure interaction using a finite element solver and adaptive remeshing', *Comput. Syst. Eng.*, **1**, 257–272 (1990).
12. R. Löhner and P. Parikh, 'Three-dimensional grid generation by the advancing front method', *Int. J. Numer. Methods Fluids*, **8**, 1135–1149 (1988).
13. A. Shostko and R. Löhner, 'Three-dimensional parallel unstructured grid generation', *Int. J. Numer. Methods Eng.*, **38**, 905–995 (1995).
14. R. Löhner, 'Regridding surface triangulations', *J. Comput. Phys.*, **126**, 1–10 (1996).
15. R. Löhner, 'Progress in grid generation via the advancing front technique', *Eng. Comput.*, **12**, 186–210 (1996).
16. R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman, 'Fluid–structure interaction using a loose coupling algorithm and adaptive unstructured grids', *AIAA-95-2259*, 1995.
17. D.J. Benson and J.O. Hallquist, 'A single surface contact algorithm for the post buckling analysis of shell structures', *Comput. Methods Appl. Mech. Eng.*, **78**, 141 (1990).
18. T. Belytschko and M. O'Neal, 'Contact–impact by the pinball algorithm with penalty and Lagrangian methods', *Int. J. Numer. Methods Eng.*, **31**, 547 (1991).
19. M.W. Heinstein, S.W. Attaway, J.W. Swegle and F.J. Mello, 'A general purpose contact detection algorithm for non-linear structural analysis codes', *SAND92-2141*, Sandia National Laboratories, May 1993.
20. J.G. Malone and N.L. Johnson, 'A parallel finite element contact/impact algorithm for non-linear explicit transient analysis. Part I. The search algorithm and contact mechanics', *Int. J. Numer. Methods Eng.*, **37**, 559 (1994).
21. D. Pelessone and C.M. Charman, 'Adaptive finite element procedure for non-linear structural analysis', *ASME/JSME Pressure Vessels and Piping Conference*, Honolulu, HI, July 1990.
22. D. Pelessone and C.M. Charman, 'An adaptive finite element procedure for structural analysis of solids', *ASME Pressure Vessels and Piping Conference*, Orlando, FL, July 1997.
23. D. Pelessone and C.M. Charman, 'A general formulation of a contact algorithm with node/face and edge/edge contacts', *ASME Pressure Vessels and Piping Conference*, San Diego, CA, July 1998.
24. L. Sakell and D.D. Knight (eds.), *Proceedings of the 1st AFOSR Conference on Dynamic Motion CFD*, Rutgers University, New Brunswick, NJ, June 1996.

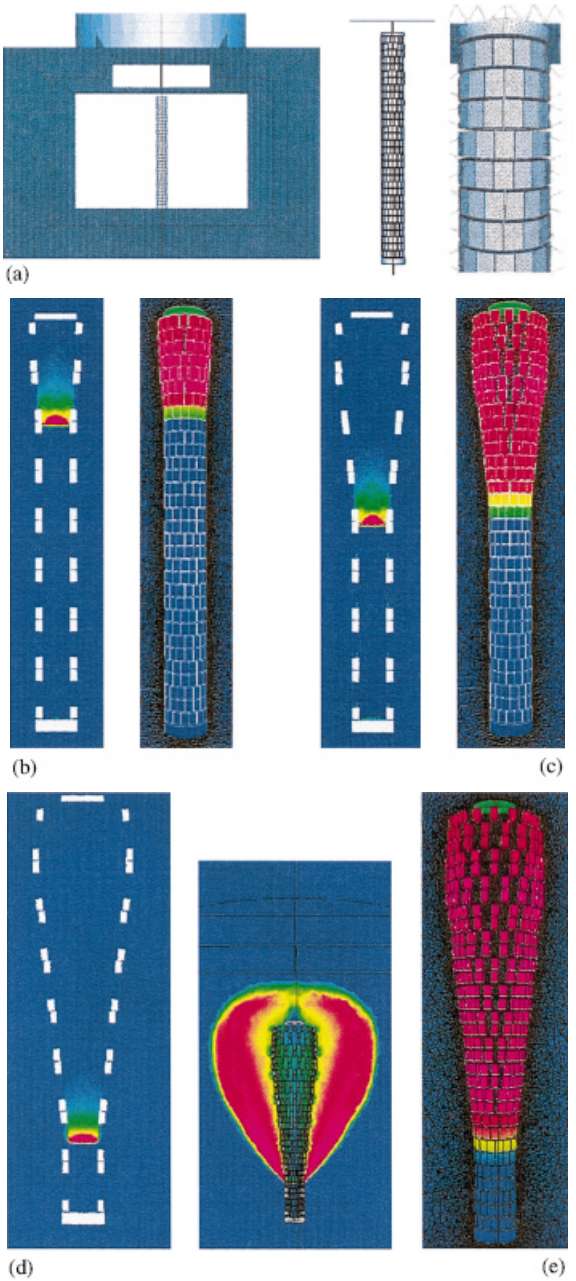


Plate 1. (a) Computational domain; (b) and (c) pressure contours and fragment velocities at $t = 72$ and $t = 139.7$ μs; (d) and (e) pressure contours, detonation products and fragment velocity at $t = 225.7$ μs; (f) and (g) pressure contours, detonation products and fragment velocity at $t = 297.3$ μs; (h) and (i) pressure contours and fragment velocities at $t = 501$ and $t = 900$ μs.

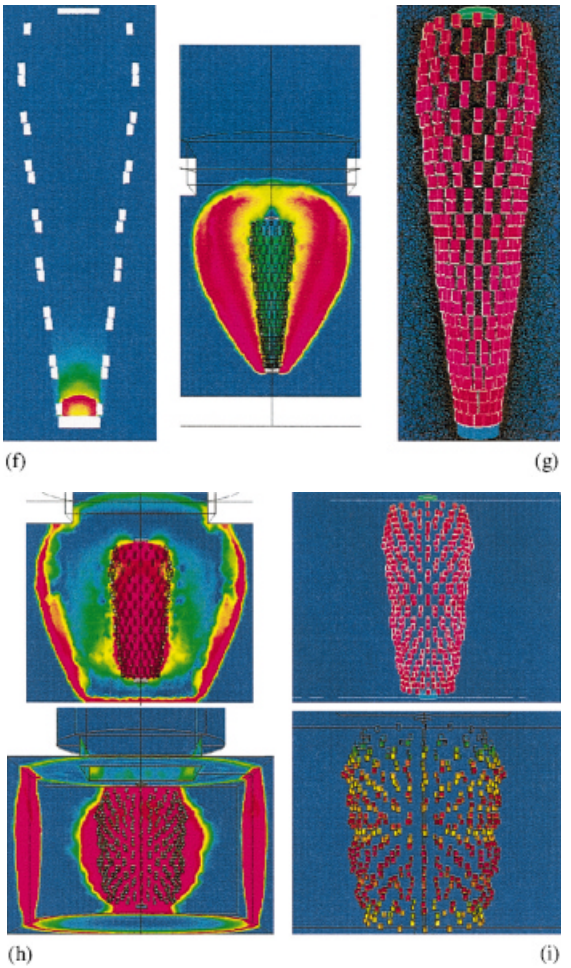
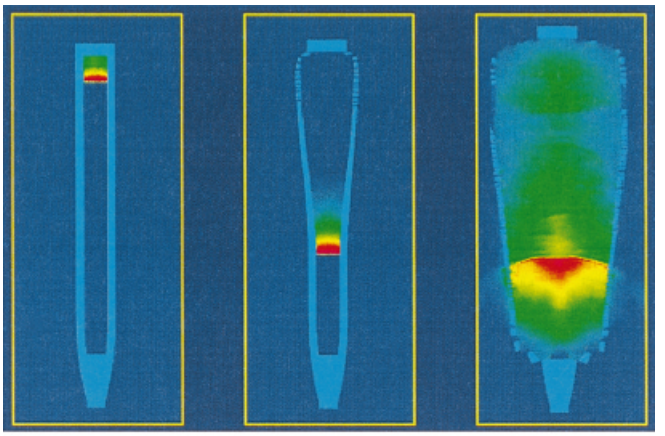
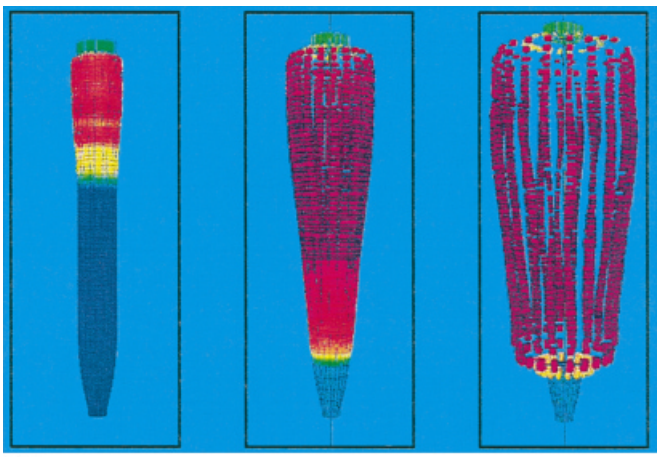


Plate 1 (*Continued*)



(a)



(b)

Plate 2. (a) Pressure at different times; (b) surface velocity at different times.