Application of visualization tools in solid mechanics

by S. Moini

With increasingly complex digital simulations and computations, large volumes of numerical output are generated, and users must select more effective techniques for handling and displaying such output in order to extract relevant information. In this study, visualization techniques such as animation, tracking, and 2D/3D color displays are imbedded in implicit and explicit finite element codes for solving complex solid-mechanics problems. With these techniques, the investigator can more fully utilize computer time and better understand the results of long and costly computations. This investigation demonstrates the effectiveness of different visualization techniques and distributed computing on an IBM platform.

Introduction

Digital simulation is an essential tool for design and performance evaluation. The quantity of output data is often very large, and extensive postsimulation operations are required in order to produce visual images. The idea of wanting to create visual output from a scientific simulation is not new, but only recently have people begun to investigate the process of visualization and think about developing software and hardware with scientific visualization requirements in mind. In a very simple form, one can define visualization as the process of aiding the scientist (investigator) in data analysis and comprehension by transforming numbers (symbolic) into images (geometric). For proper and successful implementation of any visualization tool, one must satisfy certain requirements. The ultimate visualization requirements include 1) accurate representation, 2) unambiguous interpretation, 3) multivariable interpretation, 4) volumetric display capability for three-dimensional representation, 5) animation, and 6) interactive capability. There are several important factors involved in data visualization. These include format conversion, storage, retrieval, user interface, visualization methods, and final result representation (i.e., display and presentation format).

Visualization has always played an important role in the study of structures and solid mechanics. Traditionally this has been accomplished experimentally, with visualization being part of the measurement process. For example, extensive studies of crash analysis have been carried out by means of very expensive physical impact testing. Experimental visualization yields rich data sets of very high resolution, but they are essentially images and are well suited to conventional two-dimensional image processing and analysis. More recently, both experiments

and numerical simulations have begun to yield high-resolution multivariate data. These rich and complex data sets can consist of up to six or more quantities defined at each of thousands, sometimes millions, of node points in a two-, three-, or perhaps four-dimensional domain. Often the quantities of interest are not directly in the data set, but must be derived from it using information about the underlying physics and mathematics. Given the large size of the data sets, this requires a combination of high-performance simulation with visualization, effectively utilizing both the large bandwidth of the human visual system and the capacity of supercomputers for processing large amounts of data. Consequently, it is an extremely effective way to facilitate the understanding of large and complex data sets.

One of the major problems that confronts anyone attempting scientific visualization is that workstations designed for interacting with users and producing visual information do not typically have the processing power to perform large computational simulations. On the other hand, supercomputers designed for performing very large and complex computational simulations do not typically have methods of easily bringing the visual results of the simulation to the desk of the investigator.

The traditional approach to addressing this problem has been the use of preprocessors and postprocessors. In that case, the computation-intensive part of the simulation is handled by a supercomputer and binary output transferred to a workstation for postprocessing and visualization. Most often, when the problem size is very large for a workstation to handle, or the supercomputer CPU cycles are not fully utilized and can be used freely, the postprocessing also is performed on the host computer.

Another approach is to use distributed computing, having different computers on the network perform pieces of a large problem. During execution, the data flow between program modules, any of which may reside on a remote machine. The process of transforming the results of a simulation into an image or a series of animations can be thought of as a pipeline. The information flows through a set of programs (gates) which create the final image. **Figure 1** shows a simplified visualization pipeline model. The problem with this simplified model is that all information (data) flows between processes move only in one direction. This will ultimately limit the flexibility and interactivity of the system. The steps shown in Figure 1 can be distributed among different CPUs connected by a network. The local scientific workstation is used to prepare the input data (preprocessing), set up the simulation, monitor progress of the simulation, and prepare the final output for viewing (postprocessing). The supercomputer is used primarily for the computation-intensive calculations of

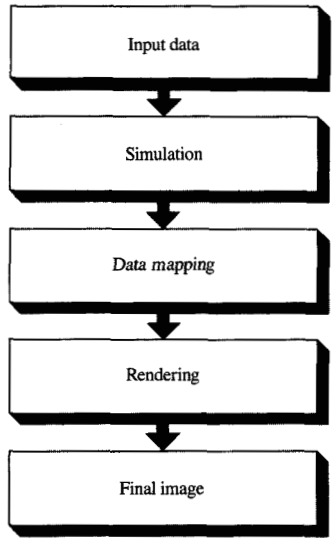

Figure 1

Simplified visualization pipeline.

the simulation model, and in some cases for mapping from simulation phase to image phase. Ideally the intercomputer connection should be fairly transparent to the scientist, and the overhead to set up such a task should be minimal.

Parallel to the idea of distributed visualization and the ultimate goal of scientific visualization is the concept of "pseudo" real-time visualization (tracking and steering). Since the real-time events happen very rapidly (anywhere from picoseconds to microseconds), real-time visualization of these events is not practical. On the other hand, simulation of these events takes minutes or perhaps hours. Therefore, the terminology *"pseudo" real time* is used to reflect the simulation time rather than actual event time. A successful computing and visualization environment should allow the investigator to display the output of each time step of the simulation as soon as it is available, perhaps during the following time step, while the simulation is in process. This technique lends itself to a fully interactive simulation and visualization. If there is a problem with simulation, the user can realize that there is an error and correct it immediately. This method has drastically lower debugging and postprocessing time requirements and associated costs than other methods. It can be implemented in two ways. The easier method is tracking; visualization is done during simulation but it is not

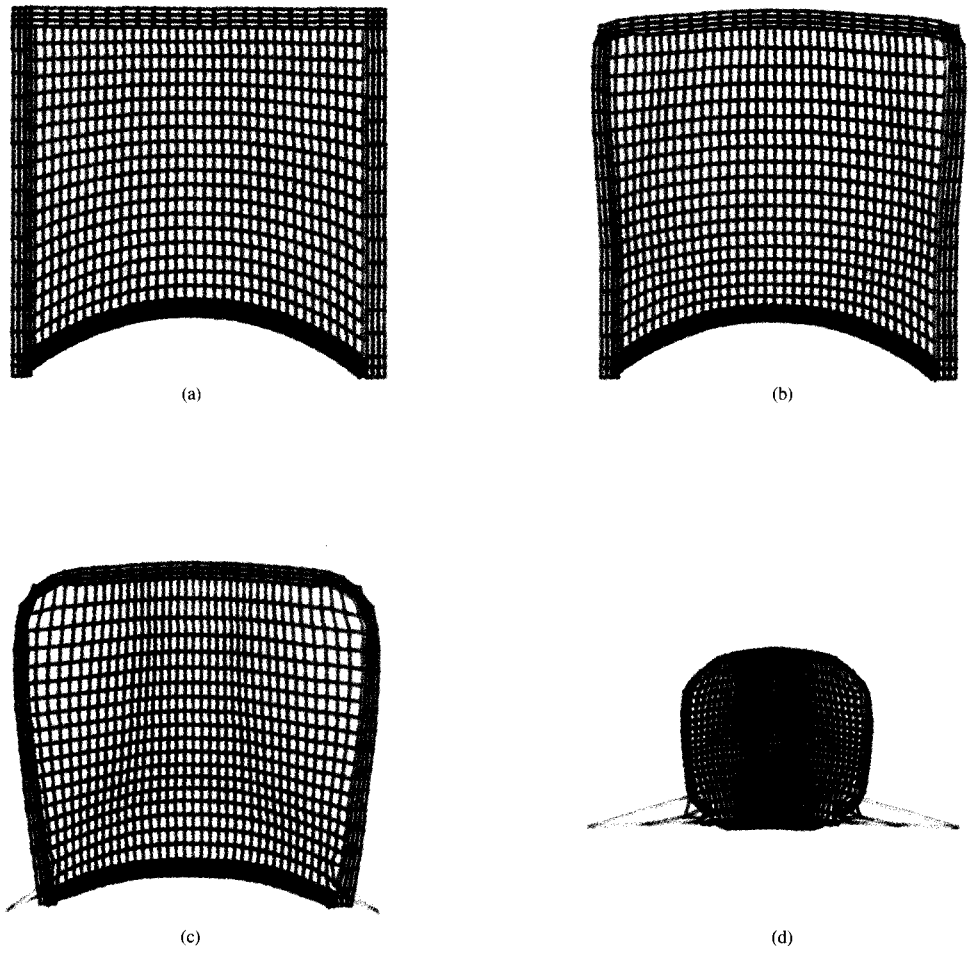

Figure 2

Shaped charge, error in input file. Deformed shapes corresponding to simulation times of (a) 0.0, (b) 0.79×10^{-5}, (c) 0.12×10^{-4}, and (d) 0.18×10^{-4} s.

interactive. The better method, though it is extremely complex, is steering [1, 2]. In this case visualization is accomplished during simulation, but analysis involves modifying the simulation interactively during the simulation.

Advances in hardware and software technology permit a wide range of new techniques for examining the global characteristics of each output variable and its interrelationship with other variables. With effective tools and appropriate methods for examining the entire output, there is a possibility of improved problem understanding and the discovery of physical results that may not be observed in a limited review of selected output variables. The investigator can also confirm that the model is physically correct, that the numerical computation is free of error, and that the iterative computation converges rapidly to reasonable results.

As computational resources become more powerful, simulations become more complex, resulting in large volumes of numerical data. Most commonly used graphical methods limit the investigator to looking at a

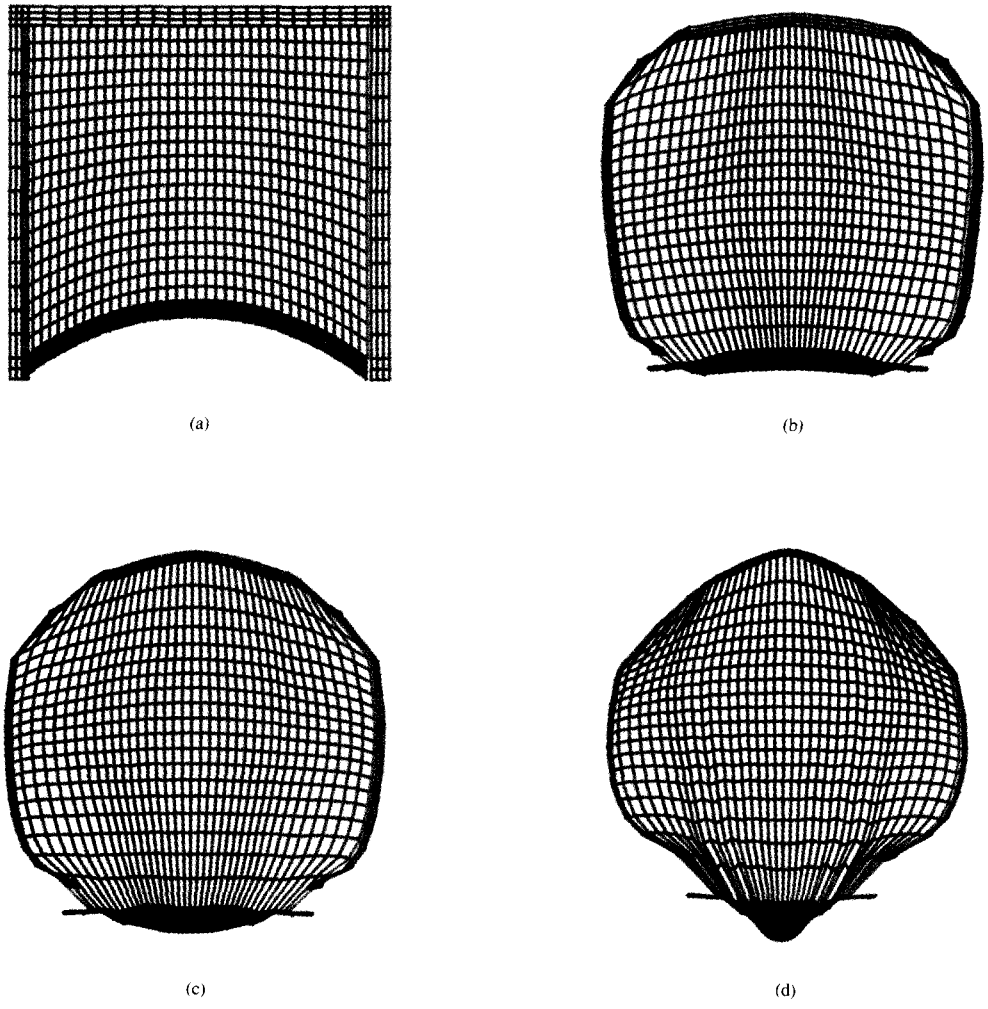

Figure 3

Shaped charge, correct input file. Deformed shapes corresponding to simulation times of (a) 0.0, (b) 0.19×10^{-5}, (c) 0.25×10^{-4}, and (d) 0.37×10^{-4} s.

few critical output variables rather than examining the complete set of output data. Graphical methods range from simple x–y plots to three-dimensional displays. These methods can also be used to obtain quantitative information such as, e.g., position, stress, strain, pressure, and temperature values at selected nodal points. In a research and development laboratory, both simulation and visualization can be at a premium. For this reason, an efficient method of utilizing both computational and graphical networks is required.

Applications in solid mechanics

The application of finite element spatial discretization to the equations of solid and structural mechanics leads to a system of semi-discrete and, in general, nonlinear equations. The methodologies for time integrating such systems may be divided into two classes, implicit and explicit, the essential difference being that the former requires the solution for a nontrivial system of equations. Each of these methods can be used in Lagrangian finite element programs for large-scale solid and structural dynamic calculations. The explicit codes, while restricted

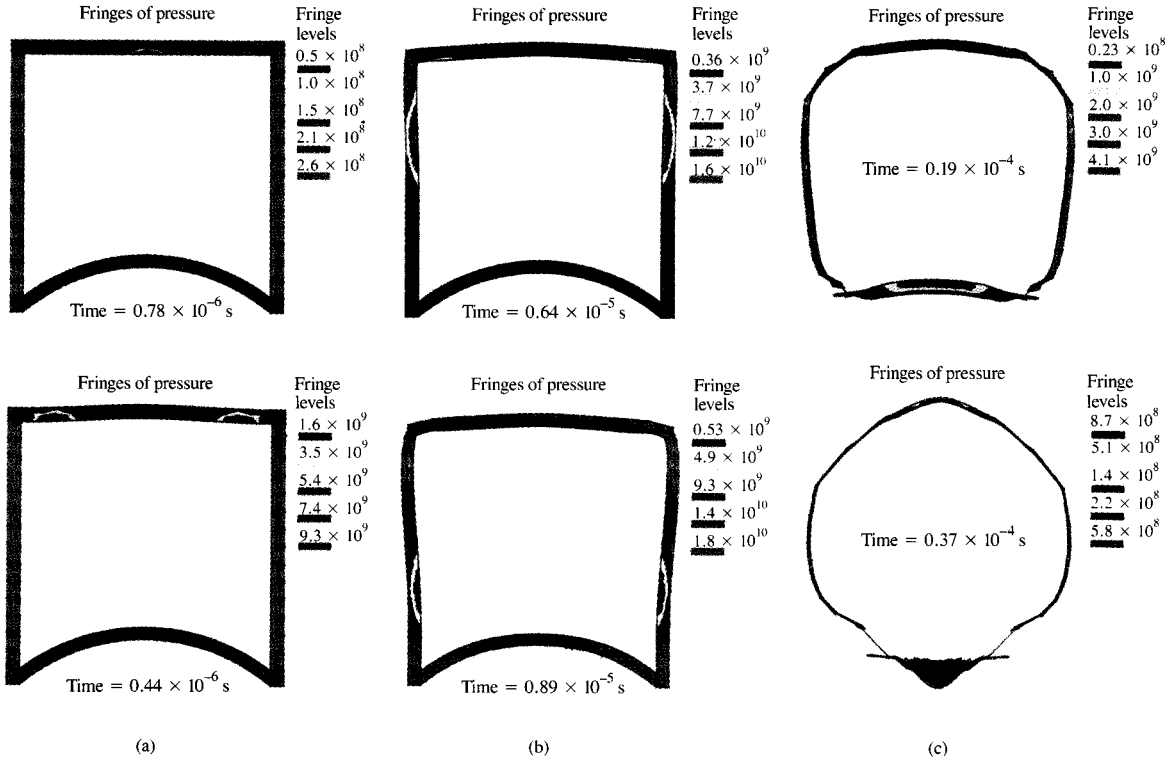

Figure 4

Shaped charge, fringes of pressure (Pa) corresponding to simulation times of (a) 0.78 to 4.4, (b) 6.4 to 8.9, and (c) 19 to 37 μs.

by stability conditions, can attain efficient solutions to a wide class of large-deformation, contact-impact problems. The implicit codes, using an unconditionally stable time-integration scheme, can be more efficient when the physically relevant time-step size is several orders of magnitude larger than that permitted by explicit codes.

In two-dimensional problems, the computational cost of both implicit and explicit schemes is dominated by the evaluation of nonlinear constitutive relationships. This leads to a natural breakdown in the way they are used. Explicit schemes are used for wave propagation, hydrodynamics, and high-frequency dynamics, and implicit schemes for statics, quasi-statics, and low-frequency dynamics. Traditionally, the vast majority of three-dimensional problems have been solved with explicit codes, even for low-frequency response, because the computational burden in both storage and CPU entailed in implicit codes by repeatedly factorizing a large system of equations overwhelms available resources.

Visualization techniques are applied to implicit and explicit finite element codes for solving complex solid-mechanics problems. In these codes, automated step-size control is used to eliminate termination due to divergence. If convergence fails, it automatically backs up to the last converged state, restarts with a new step size, and continues to iterate. This procedure continues until convergence is achieved. During the execution, the user can monitor (track) the status of the program. The solution-tracking capability imbedded in these codes allows the investigator to see the results of each simulation time step as it is advancing. The user may halt execution at any time step by terminal sense-switch controls and enter into an interactive graphics and rezoning phase. Rezoning implementation consists of three parts. First, nodal values are generated on the old mesh for all variables to be remapped. Then, the user can rezone one or more materials either interactively or automatically with a command file. In this phase, current results can be interactively displayed. Finally, the re-

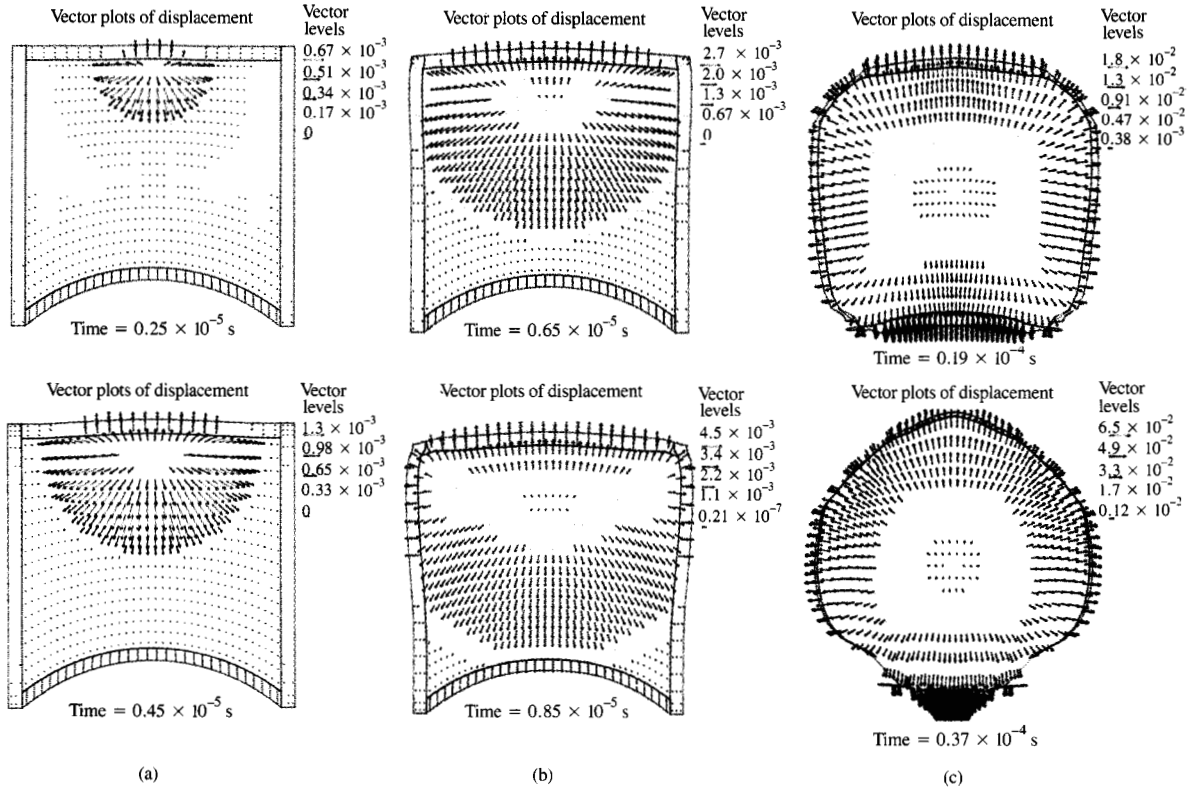

Figure 5

Shaped charge, vector plots of displacements (m) corresponding to simulation times of (a) 2.5 to 4.5, (b) 6.5 to 8.5, and (c) 19 to 37 μs.

meshed regions are initialized by interpolating from nodal point values of the old mesh, and execution continues. In an interactive display phase, binary files generated at the time execution halted are read, and contours, time histories, and deformed shapes are plotted. If desired, they can be colored for better two- and three-dimensional visualization. Contours and fringes of nearly 100 different quantities may be plotted. A variety of strain, stress, pressure, displacement, and velocity measures can be computed and displayed. Momentum can be computed from material and interface pressure, and shear profiles can be plotted along slidelines. (Slidelines are interface lines between two meshes, on which sliding, voids, and friction are permitted.) The tracking part is menu-driven, and the investigator can select any of the nearly 100 different quantities to be displayed. The user can switch from tracking one quantity to another at any time step by selecting that item on the menu. These features reduce debugging time, increase users' knowledge of the simulation process, reduce human error, and increase productivity [3].

Data from solid-mechanics simulations

The visualization tools were imbedded in two- and three-dimensional implicit and explicit finite element codes developed at the Lawrence Livermore National Laboratory [4–6] and used in the following experiments.

The two-dimensional explicit finite element program DYNA2D is used to simulate the behavior of a type of shaped charge called a self-forging fragment. Calculations were carried out to the formation of the fragment. A finite element mesh consisting of 1112 elements is used in the shaped-charge simulation model. Slidelines are initially placed between the plate–high-explosive (HE), cylinder–HE, and plate–cylinder interfaces. By 10 ms the HE has burned, and by 30 ms the pressure in the HE has dropped to negligible levels. At 30 ms the cylinder and all the slidelines are removed, and only the plate calculations are continued to the termination time of 90 ms. The first attempt at this experiment produced the error "element number 548 of material number 1 has a negative area," a restart file was written, and execution stopped after approximately 21 ms of simulation. (Material number 1

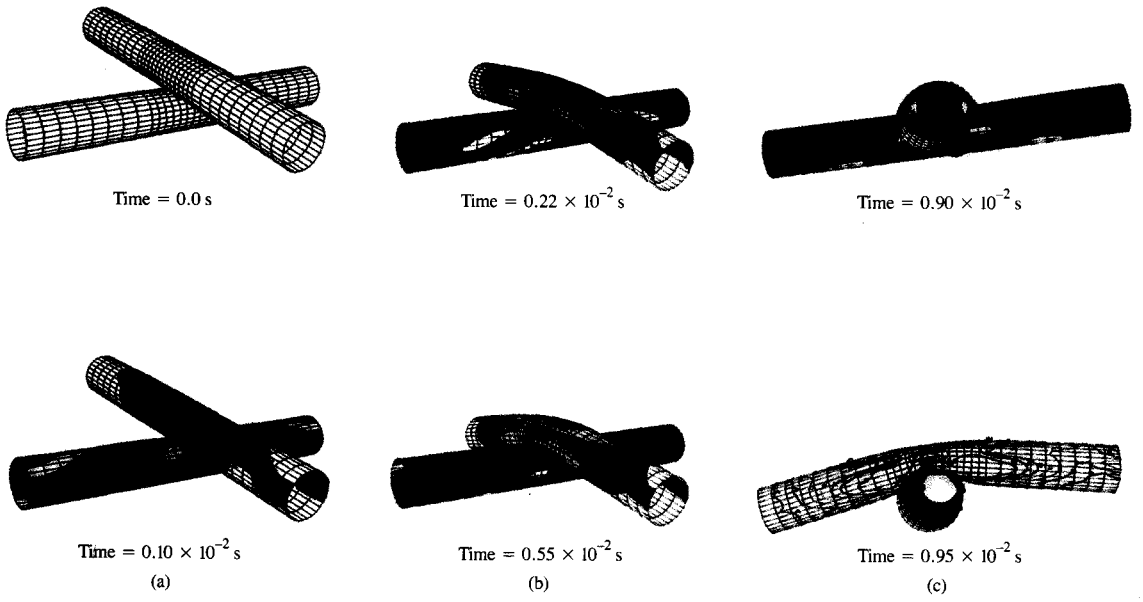

Figure 6

Pipe whip, contours of effective stress (GPa) corresponding to simulation times of (a) 0 to 1.0, (b) 2.2 to 5.5, and (c) 9.0 to 9.5 ms.

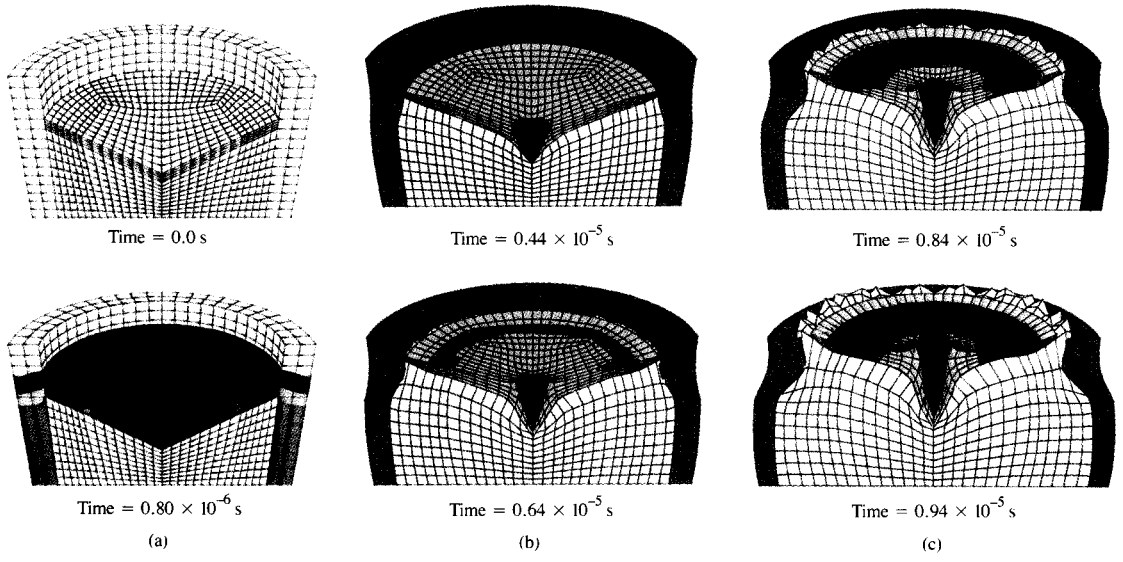

Figure 7

Shaped charge, fringes of pressure (Pa) corresponding to simulation times of (a) 0 to 0.8, (b) 4.4 to 6.4, and (c) 8.4 to 9.4 μs.

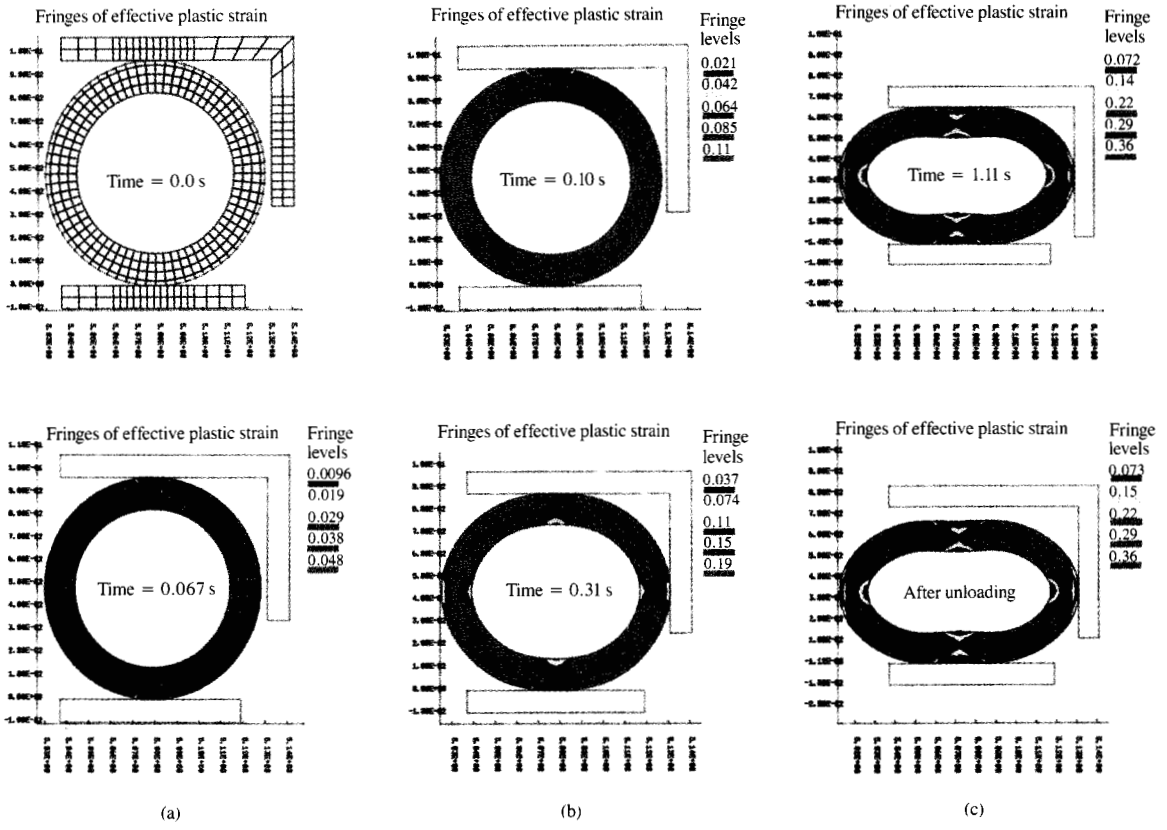

Figure 8

O-ring test, contours of plastic strain (m/m). (a) Initial condition, 0 to 0.067 s; (b) onset of plastic strain, 0.10 to 0.31 s; (c) maximum strain, 1.11 s, and residual strain (load removed).

is the HE.) Reviewing the time history plots revealed that the first few nodes of the HE were being pushed right through the cylinder. The program detected the error when the element boundaries crossed and produced a negative net element area. **Figure 2** shows a few frames of this error condition. The error was caused by the very simple mistake of not defining the last interface node between cylinder and HE as a slideline node. As soon as this simple input error was corrected, the program ran to completion without any error. **Figure 3** displays a few snapshots of error-free simulation results. Without the pseudo-real-time visualization method used in this experiment, finding and correcting the error would have been a difficult task. **Figures 4(a), 4(b),** and **4(c)** illustrate an animation of contours of pressure for cylinder and plate, and **Figures 5(a), 5(b),** and **5(c)** display an animation of displacement vectors. The motion of the wavefront is clearly visible throughout Figures 4 and 5.

In another experiment, the three-dimensional explicit finite element program DYNA3D is used to simulate the behavior of two colliding pipes. It is postulated that in the event of a pipe break, the highly pressurized fluid in the pipe could cause the pipe to swing and strike other pipes. From 1680 shell elements, a three-dimensional model is developed which consists of two steel pipes, both having a length of 250 cm, O.D. of 8.5 cm, and thickness of 1.0 cm. One pipe is oriented horizontally and fixed at both ends; the other pipe swings about one end in a plane normal to the horizontal pipe. The angular velocity of the swinging pipe at the time of impact is assumed to be 50 radians per second. Two hundred steps were used, with a termination time of 10 ms. The results of animated pressure contours are

shown in **Figures 6(a)**, **6(b)**, and **6(c)**. The pipe begins to rebound at approximately 7 ms, which is in good agreement with experimental results [4].

In the third experiment, DYNA3D is used to simulate the behavior of a self-forging fragment similar to that of the two-dimensional example. Calculations were performed on the formation of the fragment produced by the plate. A finite element mesh consisting of 5200 elements and 6816 nodes is used in the shaped-charge simulation model. Slidelines are placed between the plate–HE, cylinder–HE, and plate–cylinder interfaces. Fifty time steps were used, with a termination time of 10 ms. The animated sequence of contours of pressure produced by the three-dimensional shaped-charge model is shown in **Figures 7(a)**, **7(b)**, and **7(c)**.

In the fourth experiment, the two-dimensional implicit finite element program NIKE2D is used to simulate the behavior of an O-ring subjected to static forces. An axisymmetric finite element model of a metal O-ring seal in a steel flange, shown in Figure 8(a), is used to investigate the interface pressure between the seals and flanges. The mesh consists of 1126 nodes and three material types, steel flanges, an Inconel X-750 inner ring, and a silver O-ring. Slidelines are placed between the O-ring and flange interfaces. Loads are imposed by specifying the displacement of the nodal points along the top of the flange such that the final displaced state of 0.83 mm is reached after 14 increments. Eight additional increments are used to unload from the peak deformation. A total of 22 solution steps are used, with the termination time of 3.0 seconds. The animated sequence of contours of plastic strain in Inconel and silver is shown in **Figures 8(a)**, **8(b)**, and **8(c)**. The last picture in the sequence shows the residual deformation after unloading.

In the last experiment, NIKE2D is used to simulate the behavior of a contact interface in a metal-forming test. A finite element model of a stainless steel sleeve and elastic die, shown in Figure 9(a), is used to investigate the pressures required to collapse the sleeve into the die with a large amount of sliding along the interface. The mesh consists of 583 nodes and two material types. A pressure loading is applied in 50 equal increments to a peak value of 0.5 GPa and is removed in two increments, for a total of 52 solution steps. When the pressure reaches 0.48 GPa, the sleeve is completely collapsed. An animated sequence of deformed shapes with contours of plastic strains and displacement vectors superimposed on them is shown in **Figures 9(a)–9(e)**, respectively. The simulation results are in good agreement with experiment [6].

Discussion

At present, most investigators interact with data either in a very quantitative way, using a single contour or a line mesh plot, or in a very global way, by studying the deformation and displacement behavior of the model. Distributed computing methods combined with tracking and limited steering capability provide two- and three-dimensional display, animation, and multiple-window imaging capabilities as options for the investigator. In the course of this study, the tracking capabilities were used with distributed computing methods. A combination of an IBM 3090[1], an RT[1] system, and a PS/2[1] connected by a local area network were used. The main programs resided on the 3090 or RT, and results were displayed on an IBM 5080, 3279 display, or RT display in a native mode, or on an RT display, PS/2 display, or other equipment manufacturers' machines by means of the X-Window[2] interface. The interface among the IBM 3090, RT, and PS/2 is transparent to the user.

The techniques discussed here for viewing multivariate complex data in three-dimensional space are very important tools for researchers. Representation of 3D data as 3D images instead of a set of 2D slices is invaluable for the interpretation and understanding of such data. Figures 6 and 7 demonstrate the ease with which such complex physical processes modeled in three dimensions can be studied. Although the simulation itself is a well-known and well-established process, the tracking, partial steering, and visualization techniques described are not yet used by most researchers and are shown here to be of paramount importance in debugging, understanding, and interpretation of such complex 3D simulation processes and data.

The tracking, animation, 2D–3D display and multiple-window imaging techniques described in this paper are imbedded in three specific solid mechanics simulation codes. However, these methods can be used in a wide range of applications. In complex simulation models, the entire output can be examined for all parameter values, not just a few well-known output variables. With detailed results displayed for evaluation, the simulation process can be better understood, and new design criteria may be formulated. Without these tools, a scientist or a design engineer may spend hundreds of hours of CPU time and take weeks to find and correct a design error. The inclusion of visualization tools in large and complex programs helps users to monitor every state of the execution and obtain and correct results in the shortest time with minimum expense.

Acknowledgment

I take this opportunity to express my sincere thanks to B. H. Armstrong for carefully reading the first draft of this manuscript and making very helpful comments.

[1] 3090 is a trademark, and RT and PS/2 are registered trademarks, of International Business Machines Corporation.

[2] The X-Window System is a trademark of MIT.

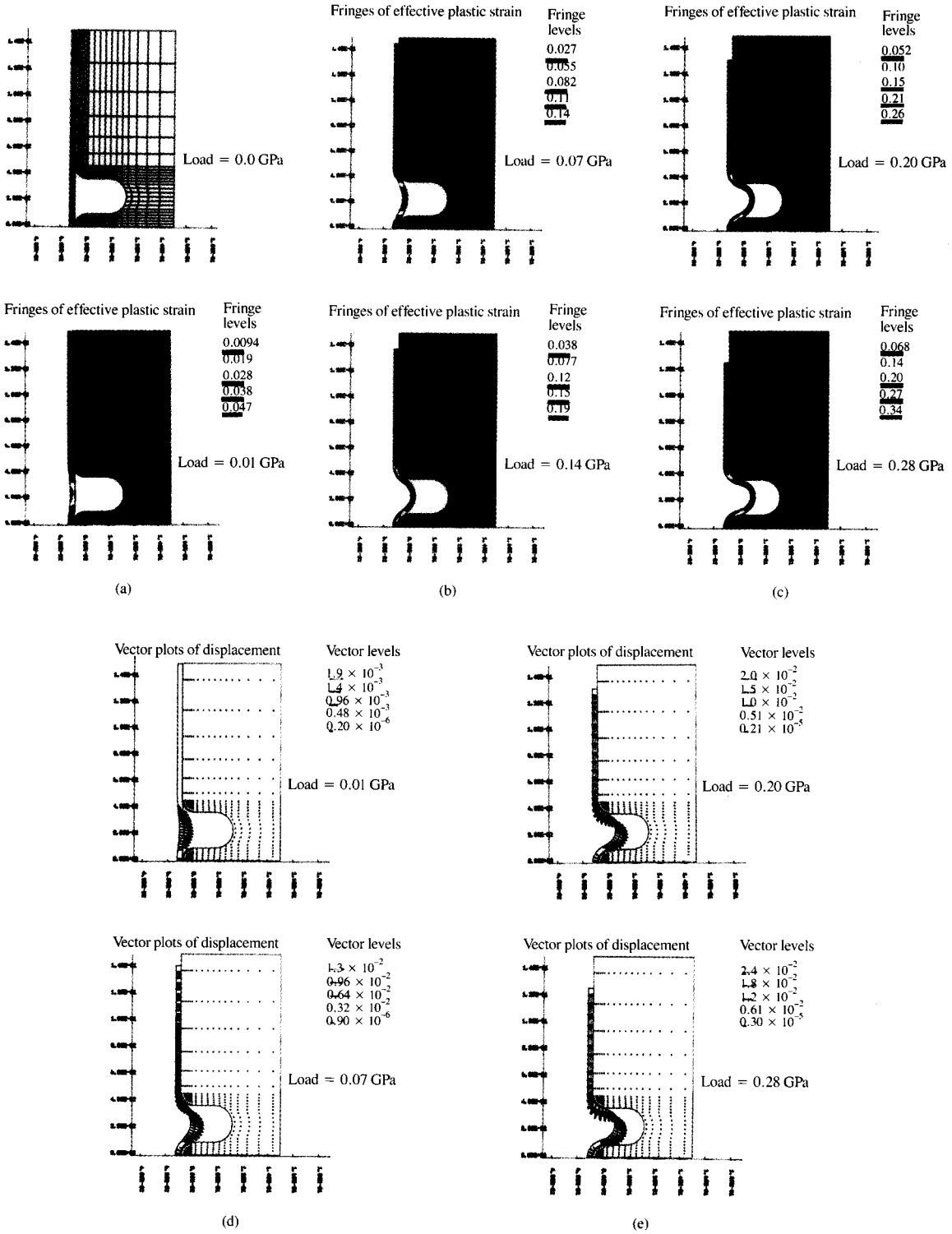

Figure 9

Metal forming: (a–c) contours of plastic strain (m/m); (d, e) displacement vectors.

References

1. W. E. Robbins and S. S. Fisher, Eds., "Three-Dimensional Visualization and Display Technologies," *SPIE Proc.* **1083**, 144–180 (January 1989).
2. J. L. Martin and S. F. Lundstrom, Eds., "Science and Applications," *Proceedings of Supercomputing 88*, IEEE Computer Society Press, Piscataway, NJ, 1988, pp. 14–230.
3. Edward J. Farrell, Steven E. Laux, Phillip L. Corson, and Edward M. Buturla, "Animation and 3D Color Display of Multiple-Variable Data: Application to Semiconductor Design," *IBM J. Res. Develop.* **29**, No. 3, 302–315 (1985).
4. J. O. Hallquist and D. J. Benson, *DYNA3D Users' Manual*, Lawrence Livermore National Laboratory, Livermore, CA, 1987.
5. J. O. Hallquist, *DYNA2D Users' Manual*, Lawrence Livermore National Laboratory, Livermore, CA, 1988.
6. J. O. Hallquist, *NIKE2D Users' Manual*, Lawrence Livermore National Laboratory, Livermore, CA, 1986.

Received November 10, 1989; accepted for publication January 3, 1991

Samad Moini *IBM Scientific Center, 1530 Page Mill Road, Palo Alto, California 94304.*

Dr. Moini is a scientific staff member in the Numerically Intensive Computing Applications group. His interests include semiconductor process, device, and circuit simulation, stress analysis by finite element methods, and scientific visualization. Before joining IBM, he was the lead engineer of a real-time hardware design team at the Advanced Product Division of Link Flight Simulation. Dr. Moini received his M.S. and Ph.D. degrees from the University of California at Davis. He has taught undergraduate engineering at UC Davis and at the California State University at Chico, and VLSI design at Santa Clara University. Dr. Moini is a member of the Institute of Electrical and Electronics Engineers and Sigma Xi.