

Conf-9410254-8

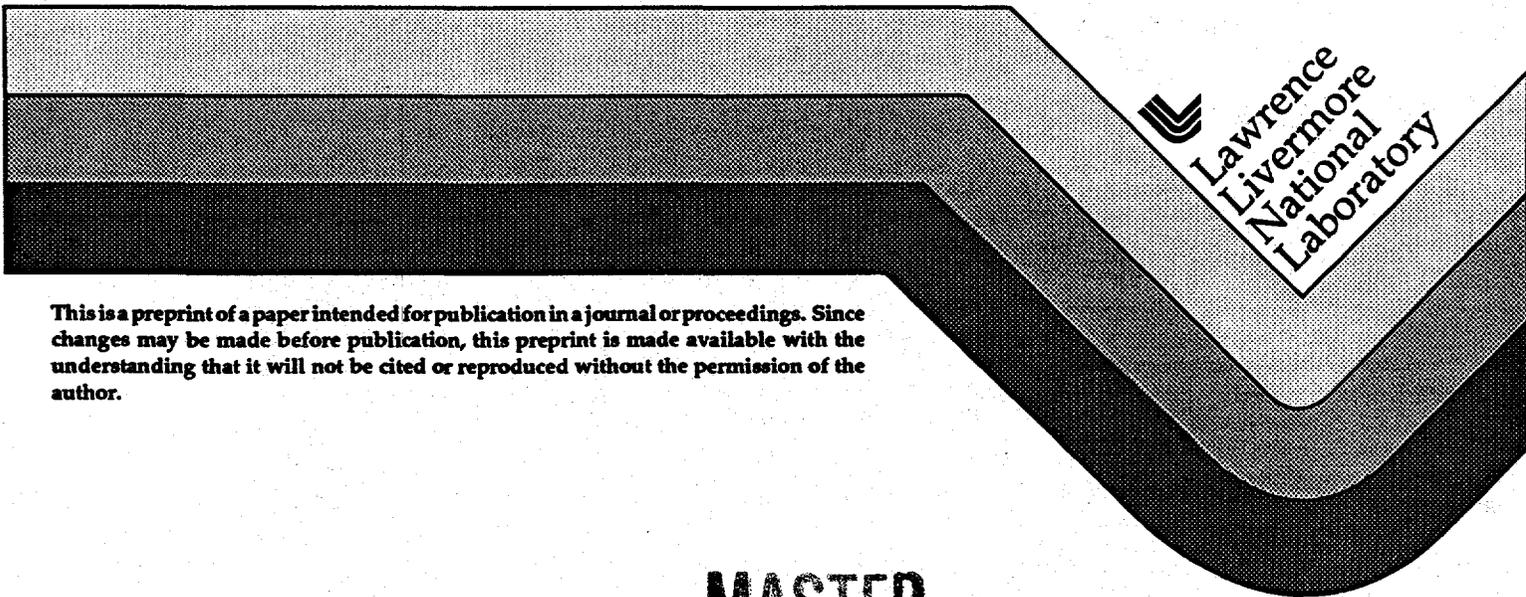
UCRL-JC- 119044
PREPRINT

PMESH: A Parallel Mesh Generator

David D. Hardin

This paper was prepared for submittal to the Eighth Nuclear Explosives
Code Developers' Conference (NECDC)
Las Vegas, Nevada
October 25-28, 1994

October 21, 1994



This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED 35

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

PMESH: A Parallel Mesh Generator

David D. Hardin
Lawrence Livermore National Laboratory

The Parallel Mesh Generation (PMESH) Project is a joint LDRD effort by A Division and Engineering to develop a unique mesh generation system that can construct large calculational meshes (of up to 10^9 elements) on massively parallel computers. Such a capability will remove a critical roadblock to unleashing the power of massively parallel processors (MPPs) for physical analysis. PMESH will support a variety of LLNL 3-D physics codes in the areas of electromagnetics, structural mechanics, thermal analysis, and hydrodynamics. (U)

Introduction

The increased computational power provided by MPPs will significantly enhance the physical modeling of complex 3-D objects and systems. MPPs now support numerical simulations with over 10^7 mesh elements (or zones), with 10^8 to 10^9 elements becoming possible in the foreseeable future. Work is planned or presently underway to port many of LLNL's physical modeling codes to MPPs. Unfortunately, the software to generate and manipulate the large meshes needed for these simulations does not exist, since state-of-the-art mesh generation with workstation-based programs is on the order of 10^5 to 10^6 elements. In many cases today's workstation would require several months to produce 10^8 elements, if one assumes it had sufficient memory and disk storage, which it does not. Obtaining the full benefit of MPPs will require increasing our mesh generation capability by two or three orders of magnitude; achieving this will require moving the mesh generation process itself to the MPP.

The Parallel Mesh Generation (PMESH) Project is a joint LDRD effort by A Division and Engineering's Computational Thrust Areas (Mechanics and Electronics/Electromagnetics) to produce a mesh-generation system which will satisfy their current and future 3-D modeling needs. PMESH has been designed from its inception to generate large meshes on MPPs and to support a variety of LLNL 3-D physics codes, including those in the areas of electromagnetics (DSI3D, TSAR), structural mechanics (DYNA3D, NIKE3D), hydrodynamics (FLAG, ALE3D, CALE3D), and thermal analysis (TOPAZ). PMESH's capabilities are needed *now*, as versions of DSI3D and DYNA3D already running on the Meiko. Indeed, DSI3D has already run a 20 million zone problem with 128 Meiko processors. Moreover, problem sizes for all our 3-D codes will grow as MPPs increase in number of processors, processor speed, and memory per processor.

Certain aspects of most current mesh generators are worth noting. First, entering the geometry for a complex problem is typically very tedious. Even if one has a

complete 3-D solid geometry description available in a modern CAD system, typically much of that information is lost during transfer to the mesh generator because it accepts only the surfaces bounding the various parts of the geometry. As a result, in the generator the user must respecify the orientations and interconnections of the surfaces to form 3-D parts. For complex problems, this may involve hundreds or even thousands of surfaces. Finding the surfaces in this morass that define a particular part can be a tedious and intimidating task.

A second shortcoming of many current generators is their inability to handle the other kinds of information needed as input by analysis codes: initial and boundary conditions, material compositions, energy sources, and various physics sentinels needed to select and define the model to be run. It is impractical for the user to edit a file defining 10^8 elements to add this information. In brief, the problem is that the mesh generator is just a mesh generator, not a problem generator.

Ideally the user could annotate the geometry's edges, surfaces, and volumes with this information, with all vertices, edges, faces, and volumes of the generated elements automatically tagged by means of information inherited from the geometry. In addition to specifying this information at a higher level, the user would benefit from doing this task just once, with every subsequently generated mesh inheriting the necessary data, regardless of whether it was a coarser or finer mesh, a differently structured mesh, or a mesh for a different analysis code. This capability would be particularly useful in areas where different kinds of analyses are routinely run on the same geometry, such as aerodynamic and radar cross-section calculations of a proposed aircraft design.

PMESH does a number of things differently in order to circumvent these difficulties. First, it distributes the generation process appropriately between the user's workstation and the MPP. Interactive tasks such as defining the geometry and annotating it with boundary conditions and materials are performed on the workstation. Computationally intensive tasks such as

the actual mesh generation are done on the MPP. The core of PMESH is a full 3-D solid modeling kernel which runs on both the workstation and on every node of the MPP; this kernel allows the geometry to be specified and manipulated independently of any mesh definition. Also, there is no loss of information during the transition from the geometry-definition phase to the mesh-generation phase. As a result, multiple meshes differing in both structure and refinement can be generated for various codes without the need for the user to re-specify the geometry or the mathematical model.

PMESH is considered to be of critical importance to the above-mentioned participating groups because mesh generation is presently a major limiting factor in simulating larger and more complex 3-D geometries. When completed, PMESH will significantly enhance LLNL's capabilities in physical simulation.

Previous Work

Commercial mesh generation codes typically run on UNIX workstations and generate a maximum of about 10^5 - 10^6 elements. Examples are Ansys, Aries, AutoCAD, EMAS, GridPro, IDEA-S, PATRAN, Pro/ENGINEER, and TrueGrid. Generating this many elements can take hours or days even on relatively powerful workstations. Cray Research (CRI) is developing a meshing package based on a Lockheed CAD package (ACAD) which has been used to generate meshes with up to about 3×10^7 elements. However, Cray's tool is limited to creating completely regular, orthogonal, structured meshes for a specific electro-magnetics algorithm. Problems with approximately 10^7 elements represent the upper bound on the size of electrical problems one can solve on present-day Cray supercomputers, and the maximum mechanics problem size is probably less.

There are many non-commercial mesh generation projects. The National Grid Project (NGP) at Mississippi State University is a project funded by a consortium of industry and government organizations which is developing a general-purpose meshing tool which will meet the needs of the consortium members (Thompson, 1992). LLNL is a member of the consortium and will receive source code which may be useful in this LDRD project. Sandia National Laboratories has formed a CRADA with a number of automobile and aerospace firms to develop mesh generation capabilities for industry (Blacker et al., 1991). The goal of the SNL CRADA is to develop a tool (CUBIT) which will simplify the meshing process for structural simulation, and the team will focus on the use of paving and plastering techniques.

Smaller efforts are also underway. Shephard et al. at RPI is doing a variety of work in the unstructured mesh development arena for industrial applications, including generation by the finite octree method (Shephard and Georges, 1991).

Each of these projects is focusing on different aspects of the mesh generation problem, but none is considering the problem addressed here, namely, generating the very large meshes that will be required to utilize the next-generation MPPs. In particular, none of the above was a parallel mesh generation effort when PMESH began a year ago.

However, some work is underway in massively parallel mesh generation. Researchers at JPL have developed a technique by which a coarse mesh generated on a scalar machine can be subdivided uniformly on a MPP to produce a large mesh. Although this technique has merit for some applications, it provides no support for local mesh refinement, and geometry information that cannot be represented on the coarse mesh is lost.

Lohner et al. (1991) at George Washington University used a mesh growing technique called the advancing front method to generate 2-D unstructured triangular meshes on a MPP. He has previously used the method to sequentially generate 3-D unstructured tetrahedral meshes and has expressed the intent to extend his MPP work to 3-D.

Weatherill (1992) at Swansea has done unstructured 3-D tetrahedral meshes based on Delaunay triangulations, and some of his software is available to us as part of NGP. He has generated three million element 2D triangular meshes and has begun parallelizing his algorithms (Weatherill, 1993).

Within the last year both Shephard and CRI have begun parallelizing their methods. A difficulty with all the unstructured approaches is that most LLNL 3-D analysis codes require or prefer non-tetrahedral meshes. Meanwhile, CRI's method remains highly specialized.

Recently Chrisochoides (1994) at Syracuse has discussed a parallelized method of generating block-structured hexagonal meshes. It has only been demonstrated for simple geometries subdivided into a few sub-blocks and was motivated by the desire to avoid the expense of computing an optimal mapping of a sequentially generated mesh onto the distributed memory on an MPP.

This last point is worth emphasizing to avoid possible confusion. Much work has been done on the problem of mapping a sequentially generated mesh onto the distributed memory of an MPP for the solution of PDEs. One well-known example is recursive spectral bisection method (Simon, 1994). The central point of the present work is that, for really large meshes, the mesh itself must be generated in parallel, not mapped later.

In summary, a large number of efforts are presently underway in mesh generation. Many of them are narrowly focused on particular types of meshes or do not address large meshes. Also, it again should be emphasized that the problem is not simply one of mesh generation. For such large meshes, it is also necessary to solve the problems of annotating the mesh with analysis code information and of visualizing the mesh.

Approach

The PMESH architecture includes a graphical user interface (GUI), a powerful 3-D solid modeler, a software "backplane" that integrates a number of different meshing algorithms and provides communication between the user's workstation and the MPP, and a modeling database that will be used to guide the production of meshes for various physics codes. We plan both to develop some new meshing algorithms (e.g., mixed meshes of tetrahedra, triangular prisms, pyramids, and hexahedra) and to integrate "standard" meshing algorithms developed elsewhere, e.g., the National Grid Project (NGP). As this is a fast developing field, we shall monitor outside efforts that may offer opportunities for collaboration.

A central feature of PMESH is that it distributes tasks appropriately between the workstation and the MPP (see Fig. 1). Interactive work, such as defining geometry, materials, boundary conditions, and meshing constraints, is performed on the workstation. Floating point and memory intensive work, such as generating and verifying the mesh, is done on the MPP.

Another key aspect of PMESH's design is that it is intended to serve many different physics codes. At least the following analysis codes will be supported: DSI3D (time-domain electromagnetics); DYNA3D/NIKE3D (structural analysis); TOPAZ (thermal analysis); and FLAG and ALE3D. It will accomplish this goal in part by allowing and encouraging the user to specify the problem at a higher and more abstract level than is now generally the case.

All meshes will be derived from a geometry model created on any of several commercial packages that will be supported. We plan to support Pro/ENGINEER, ACIS-based products such as AutoCAD, and the data interchange standards IGES and PDES/STEP. The user completes the physics model by annotating the geometry, not the mesh, with data such as boundary conditions and materials. The annotated geometry contains all information about the problem which is necessary for whatever modeling the user is interested in performing. (We assume that the annotated geometry is compact enough to manipulate on a workstation.)

To produce a block-structured mesh, the user must subdivide the geometry into logically hexahedral subregions. If necessary or desired, these regions may be degenerate hexahedra. For instance, collapsing one of the quad faces to a single point produces a pyramid. The block structure is incorporated into the annotated geometry abstraction.

Before sending the geometry to the MPP, we must decide how to distribute among the processors the work of generating the mesh from the geometry. In the case of block-structured hexahedral meshes, the number of nodes on each edge of a block is specified by the user. Hence, the size of a block-structured mesh is known before beginning to generate the mesh. Initially we will

simply assign an entire block to a processor. At a face or edge shared by two or more blocks, all processors involved know the common geometry. Thus, in this approach, interprocessor communication is required only to ensure that all processors use the same global node numbers for shared nodes. We solve this by a simple algorithm that assigns "ownership" of a shared node to a processor, which means it has the responsibility of determining the node's unique global node number. The drawback to this simple scheme of giving an entire block to a processor is that the processors' work load may be badly out of balance if there are too few blocks or if the number of zones in the blocks varies significantly.

We may postpone improving this work distribution algorithm until we have gained experience with generating unstructured meshes. We expect these to be much harder to statically load balance due to the difficulty of a priori estimating the number of zones that will be generated in the various parts of the geometry.

The annotated geometry, including processor work assignment or load-balancing information, is sent to every processor of the MPP to generate the mesh. The zonal nodes, edges, faces, and volumes of the generated mesh inherit the physics quantities from the underlying geometry. Upon specifying a specific analysis code, this information is translated to that code's input format, including the correct names and numerical encodings of physics flags.

We anticipate major changes in how users verify the acceptability of a mesh. Inspecting 3-D meshes is much more difficult than inspecting 2-D meshes. Hence we believe it will be necessary for the user to mathematically specify acceptance criteria that can be checked by the program itself. The quality of the mesh (maximum aspect ratio, minimum and maximum interior angles, etc.) and the fidelity of its representation of the underlying geometry will be ascertained using mesh verification tools which will also exist on the MPP. Automatic verification will be very important for these large meshes as it is totally impractical for a human to manually examine 10^9 3-D zones.

We are just starting to study how to visualize these large meshes. Any operation applied to the entire mesh can only be done to the distributed mesh on the MPP. It will be necessary to limit plots to a subset of the mesh in order to have reasonable data transfer and plot times. There is a spectrum of choices as to how to divide the visualization work between the workstation and the MPP. One extreme is to do everything on the MPP, sending only a raster image to the workstation. The other extreme is to extract a subset of the data itself and send it to the workstation for rendering. Of course, the optimal choice depends heavily on the type of plot, the speed of the workstation's graphics hardware, and the speed of communication between the workstation and the MPP. Even today, there are orders of magnitude differences in these last two quantities between various users' equipment.

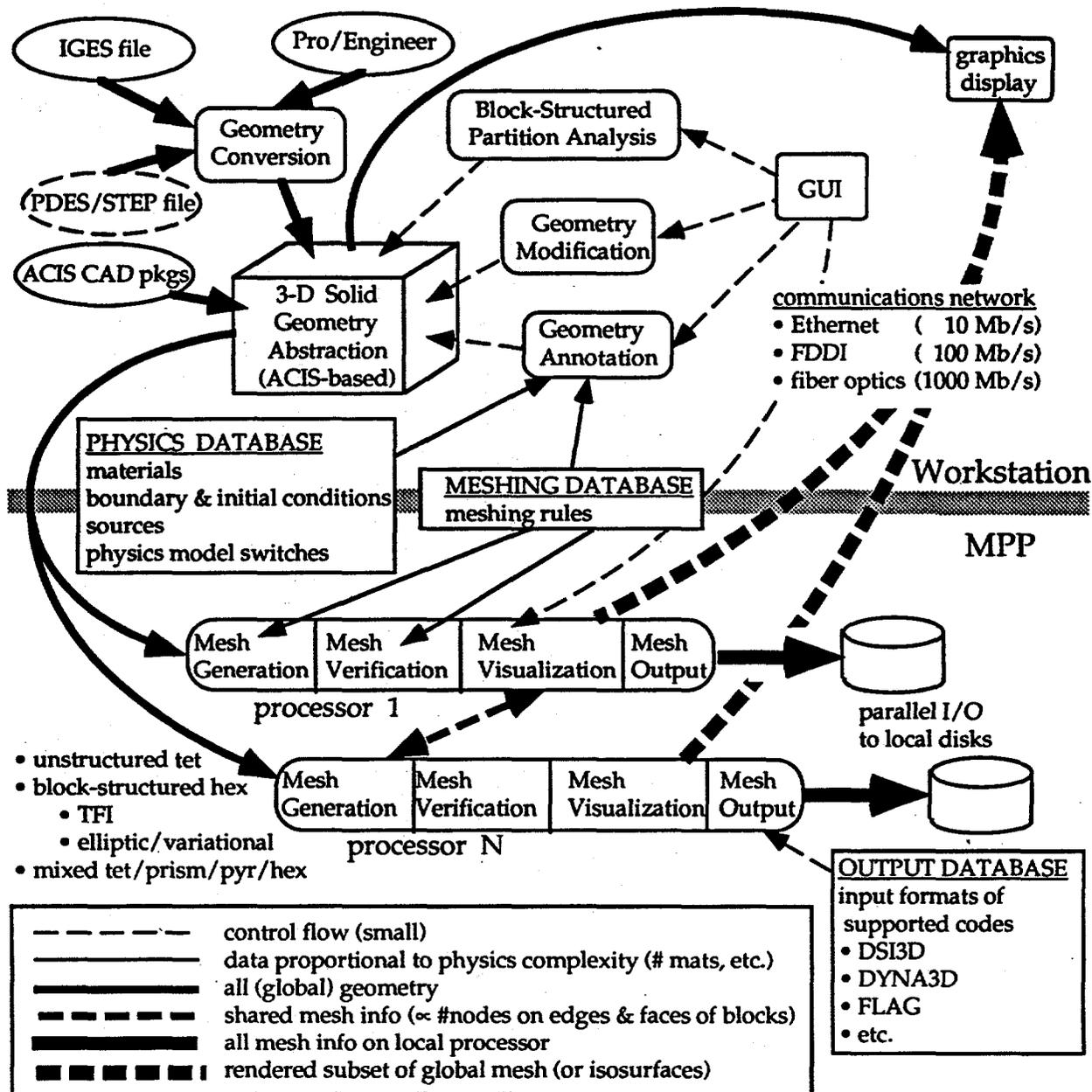


Figure 1. Data flow in PMESH. An abstraction of the geometry is constructed on the workstation, with all mesh generation done on the MPP.

Progress

We have designed the overall system architecture and its major components. We decided to base PMESH on Spatial Technology's ACIS® Geometric Modeler, a 3-D solid-geometry kernel system that is used in several CAD products. At a cost of only \$25,000 this system gives PMESH a very powerful solids modeler that would probably require 50 to 100 man-years of effort to

reproduce. Building PMESH around a solids modeler gives it significant advantages over systems such as the National Grid Project which have fewer geometric capabilities. Given that we wished to use C++ as the primary PMESH implementation language, an added advantage of ACIS is that it is written in C++ and has a C++ interface.

Initially we are generating only block-structured hexahedral meshes. In this common approach (used in INGRID and many commercial generators), the

geometry is decomposed into a collection of volumes which are topologically equivalent to rectangular blocks (hexahedra). Each block is then given a logically rectangular mesh of J nodes by J nodes by K nodes. Currently this is done with trilinear transfinite interpolation (TFI), an algebraic method. PMESH supports both uniform and non-uniform distributions of mesh nodes along block edges. This is all the information that a user needs to specify in order for the TFI algorithm to mesh the surfaces and the interiors of the blocks. (With TFI we have already meshed DYNA3D and DSI3D test problems.)

The PMESH graphical user interface (GUI) presently can display the geometry and any block-structure created so far, it and permits the user to click on an edge or surface in the geometry and tag that entity with any of several attributes selected from a menu. The GUI menu now supports specifying the number of nodes to generate on an edge, the node distribution, and several other quantities.

In addition, PMESH already contains algorithms to simplify the user's tasks and to avoid potential errors. For example, the mesh is required to be consistent where multiple blocks share edges or faces. To enforce this constraint, PMESH walks through the block structure and partitions the edges into sets of edges which are logically constrained to have the same number of nodes. Thus the user has merely to designate the desired number of mesh nodes for one edge within each set of equivalent edges.

At present, the user manually transfers the annotated geometry to the MPP (via ftp) and invokes the "back-end" of the system. The back-end generates the mesh from the annotated block-structured geometry and runs on workstations as well as on the Meiko MPP. Thus users who need only small meshes can generate them on their own workstation without being forced onto the MPP. In addition, users with large problems can generate small trial meshes locally before moving to the MPP.

In June 1994 PMESH generated a 100 million (10^8) zone problem in 8.5 minutes with 64 Meiko processors. This problem had 63 blocks, each with a mesh of $117 \times 117 \times 117$ nodes, resulting in 1.6 million zones being generated on each processor. The calculation of the nodal coordinates was not vectorized. Also, there are some significant other inefficiencies that should be removed. We estimate that fixing these problems may allow this calculation to run in 4 minutes or less. Also, at the time of this result, the Meiko system software limited a user job to half the available memory. Thus, it may be possible to generate nearly 800 million zones on the entire 256 processor machine.

Future Work

Because algebraic methods of mesh generation may produce invalid meshes for some complex blocks, the TFI method will be augmented by more robust but

slower block-structured methods based on solving elliptic equations or variational principles. These methods can be orders of magnitude slower than TFI, and parallelizing them will have a much larger payoff than in the case of TFI.

In addition, we will extend the system to generate unstructured meshes, starting with tetrahedral meshes. Unstructured mesh methods are expected to be more difficult to parallelize; in particular, it may be nearly impossible to choose an a priori distribution of the work that balances the load on the processors. If so, dynamic load balancing may ultimately be required. For codes such as DSI3D and FLAG which support five-sided elements, we will develop strategies for generating mixed meshes containing tetrahedra, pyramids, triangular prisms, and hexahedra as a means of better coping with complicated geometries.

How to do graphics on large data sets distributed across an MPP, including deciding which tasks should be done on the MPP and which on the workstation, is a significant research problem in itself. Our initial approach to mesh visualization will be to extract a subset of the mesh data from the MPP and send it to the workstation for rendering. This has the advantage that many operations (zoom, pan, rotate, etc.) can be done locally without returning to the MPP. Given the continuing increase in workstation and communications speeds, we feel this is a solution that is rapidly becoming more attractive.

Another activity that we are just beginning is to establish workstation-MPP communications that are "transparent" to the user. Initially, we expect the workstation to "reexec" the back-end of the system on the Meiko and establish TCP sockets over Ethernet between the two machines for exchanging control data from the workstation and visualization data from the MPP.

References

- Thompson, J. F., "National Grid Project," *Computing Systems in Engineering*, 3, 393-399 (1992).
- Blacker, T.D., Stephenson, M.B., and Canaan, S., "Analysis automation with paving: A new quadrilateral meshing technique," *Adv. Eng. Software*, 13(5/6) (1991).
- Shephard, M.S. and Georges, M.K., "Automatic three-dimensional mesh generation by the finite octree technique," *International Journal for Numerical Methods in Engineering*, 32, 709-749 (1991).
- Lohner, R., Camberos, J., and Merriam, M., "Parallel unstructured grid generation," in *Unstructured Scientific Computation on Scalable Multiprocessors*, ed. by P. Mehrotra, et al. (MIT Press, 1991).

Weatherill, N., "Unstructured Triangular Grid Generation by the Delaunay Triangulation," available as part of the National Grid Project documentation, 1992.

Weatherill, N., private communication (1993).

Chrisochoides, N., "An Alternative to Data Mapping for Parallel PDE Solvers: Parallel Grid Generation," *Proceedings of the Scalable Parallel Libraries Conference* (IEEE Computer Press, 1994).

Simon, H. D., "Partitioning of Unstructured Problems for Parallel Processing," *Computing Systems in Engineering*, 2,(2/3), 135-148 (1991).

Acknowledgements

The other members of the PMESH Project team are Freddie L. Barnes, John C. Compton, Douglas L. Dickson, Donald J. Dovey, Gary W. Laguna, and A. Ellen Tarwater.

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