



LAWRENCE  
LIVERMORE  
NATIONAL  
LABORATORY

# Parallel Adaptive Multi-Mechanics Simulations using Diablo

D. Parsons, J. Solberg

December 6, 2004

NECDC 2004

Livermore, CA, United States

October 4, 2004 through October 7, 2004

## **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 product, 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.

## **Parallel Adaptive Multi-Mechanics Simulations using Diablo (U)**

**Dennis Parsons\* and Jerome Solberg\***

\*Methods Development Group, Defense Technologies Engineering Division,  
Lawrence Livermore National Laboratory, Livermore, CA 94551-0808

*Coupled multi-mechanics simulations (such as thermal-stress and fluid-structure interaction problems) are of substantial interest to engineering analysts. In addition, adaptive mesh refinement techniques present an attractive alternative to current mesh generation procedures and provide quantitative error bounds that can be used for model verification. This paper discusses spatially adaptive multi-mechanics implicit simulations using the Diablo computer code. (U)*

### **Diablo and Multi-Mechanics Simulations**

Diablo is an ASC-funded code developed to address both the multi-mechanics simulation needs of engineering analysts and the potential of MPP computing platforms. The initial focus of the effort is on implicit structural mechanics (e.g., low frequency dynamics, quasistatics and free vibration) coupled with heat conduction. Subsequent extensions to other mechanics domains are envisioned in the future. A flexible software architecture has evolved that combines the versatility of object-oriented design with the performance of Fortran.

Diablo defines the model as an assembly consisting of multiple subassemblies that correspond to distinct components of the artifact under consideration. The precise specification of each subassembly is the responsibility of the analyst, but should be guided by factors such as model generation considerations and the need to simulate multi-step assembly procedures. This view of the user model is intended to allow the analyst to perform complete simulations of the as-built artifact in its design environment.

Given the model defined as an assembly of subassemblies, Diablo then conducts a multi-mechanics simulation by evolving a set of fields (e.g., displacements, temperatures, specie concentrations) according to the conservation laws that are appropriate to the selected mechanics. Currently, all of the mechanics utilize a common mesh, a reasonable restriction for the problems of current interest that eliminates the need to map fields between overlaid meshes. However, future work may employ mechanics-specific meshes within a nested AMR strategy. In addition, the core Diablo capabilities use a common domain decomposition of the various subassemblies.

*Proceedings from the NECDC 2004*

Consider a two-field simulation (e.g., thermal-stress analysis) for which the conservation laws have been written in weak form and discretized using the finite element method. The resulting system of nonlinear equations can be written as

$$\mathbf{R}(f_1, f_2) = \mathbf{0}, \quad (1)$$

where the multi-field residual  $\mathbf{R}(f_1, f_2)$  is a nonlinear function of the two fields  $f_1$  and  $f_2$ . In the case of thermal-stress analysis,  $f_1$  and  $f_2$  would correspond to nodal displacements and temperatures. Diablo uses Newton iteration to drive this residual toward zero, thereby requiring the solution of a linearized system of equations of the form

$$\begin{bmatrix} \frac{\partial R_1}{\partial f_1} & \frac{\partial R_1}{\partial f_2} \\ \frac{\partial R_2}{\partial f_1} & \frac{\partial R_2}{\partial f_2} \end{bmatrix} \begin{Bmatrix} \delta f_1 \\ \delta f_2 \end{Bmatrix} = \begin{Bmatrix} R_1(f_1, f_2) \\ R_2(f_1, f_2) \end{Bmatrix} \quad (2)$$

for the field increments  $\delta f_1$  and  $\delta f_2$ . Two approaches may be adopted: partitioned coupling and monolithic coupling [1,2].

Partitioned coupling is the typical strategy employed for multi-mechanics simulations, and eliminates the cross-coupling terms from Equation (2) while keeping the residual as a function of all fields. Thus the linearized system decouples as

$$\begin{bmatrix} \frac{\partial R_1}{\partial f_1} & \mathbf{0} \\ \mathbf{0} & \frac{\partial R_2}{\partial f_2} \end{bmatrix} \begin{Bmatrix} \delta f_1 \\ \delta f_2 \end{Bmatrix} = \begin{Bmatrix} R_1(f_1, f_2) \\ R_2(f_1, f_2) \end{Bmatrix}. \quad (3)$$

Advancing the state over a single global time step then requires iterations over each active mechanics, in which the relevant fields are updated using a sequence of Newton iterations. The decoupled system enables separate coding of the individual mechanics, thereby allowing multiple domain experts to participate independently in the development of Diablo.

Monolithic coupling utilizes a multi-field left-hand side operator, thereby preserving the cross-coupling terms in Equation (2) that must then be solved for the field increments. Within the Diablo framework, this approach is viewed as an additional type of mechanics (e.g., thermostress) that must be implemented at the element level.

## **Parallel Code Architecture**

The bulk of Diablo is written in object-oriented Fortran 95, with some additional routines in C. Although Fortran is not widely recognized as an object-oriented language, the relatively recent availability of modern elements such as modules, user-defined data types, pointers and dynamic memory management enable the scientific programmer to take advantage of at least some of the tenets of object-oriented programming such as abstraction, encapsulation and modularity. Although Fortran does not yet allow class hierarchies to be expressed using inheritance, aggregation is permitted (which may in fact be an appropriate expression of the hierarchies inherent to a finite element data model).

Diablo's architecture is planned and monitored graphically using the Unified Modeling Language (UML), which is proving to be a useful tool that helps avoid the inscrutable coding styles that have plagued many scientific and engineering codes written in Fortran in the past. The resulting modular design readily allows multiple programmers to introduce extensions to the core capabilities such as mechanics formulations, element kinematics and material models.

The subassemblies are partitioned into domains for parallel computations in an element-based manner using standard packages such as Metis. The resulting inter-domain communications are performed using calls to the MPI library for both the collective and point-to-point communications. Static load balancing is currently attempted during the pre-analysis domain decomposition using Metis, although dynamic load balancing strategies are being considered as part of the parallel AMR implementation discussed later in this paper.

### **Example 1: Parallel Contact for Thermal-Stress Analysis**

To briefly demonstrate Diablo's capabilities, consider the results of the coupled thermal-stress analysis shown in Figure 1. The model geometry is derived from AWE's Modal Analysis Correlation Experiment, and includes a combined thermal-stress contact surface between the sphere and the pad on the right that is distributed across multiple processors. Figures 1a and 1b show the distribution of temperatures when the model is subjected to a series of nodal temperatures around the equator of the hollow sphere for different values of the contact resistance employed in the thermal contact surface. The decrease in temperatures for the pad on the right in Figure 1b is a result of increased contact resistance that reduces the energy flow to the pad for that case.

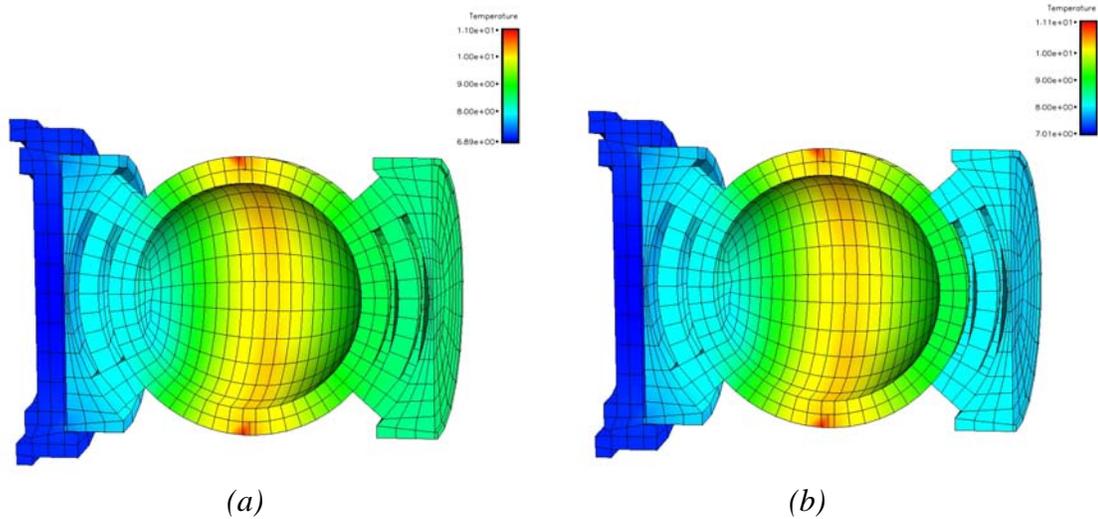


Figure 1: Computed temperature distributions for the AWE test problem.

## AMR for Multi-Mechanics Simulations

AMR has obvious advantages for accurate engineering simulations, such as automatically providing optimal meshes for a specified error tolerance and demonstrating solution verification. Diablo provides the analyst a capability for  $h$ -refinement of hexahedral elements. This approach (as opposed to  $p$ -refinement or refinement of meshes based on tetrahedral discretizations) was chosen based on the historical success of low order brick elements in large deformation solid mechanics simulations.

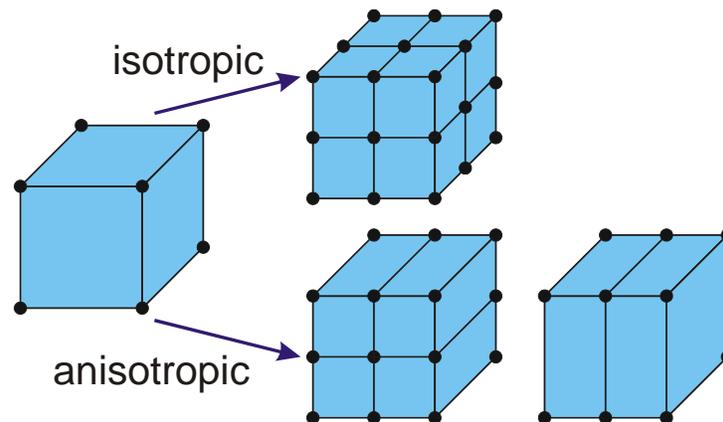
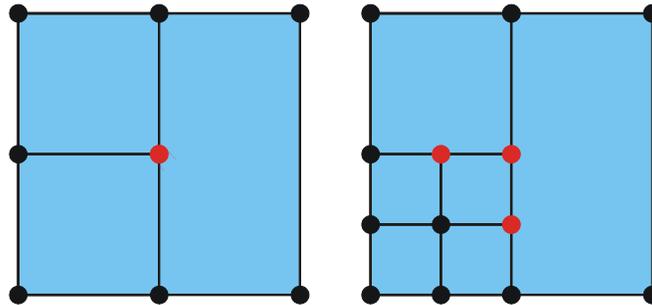


Figure 2: Isotropic and anisotropic hexahedral mesh refinement.

The AMR implementation is able to perform both isotropic and anisotropic refinement as well as derefinement of a mesh, as demonstrated in Figure 2. Isotropic refinement is directionally invariant, causing a single hexahedron to be refined into eight new hexahedrons. Anisotropic refinement requires that the error estimator identify local element directions for refinement, causing a hexahedral element to be split into two or four new elements. Anisotropic refinement is useful for problems that contain strong

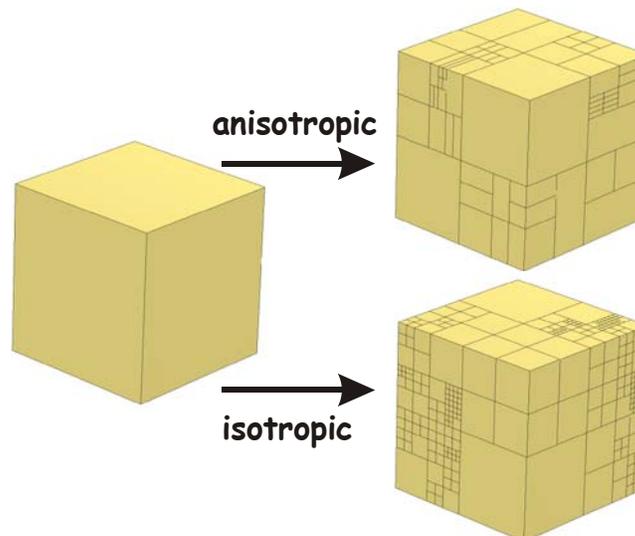
*Proceedings from the NECDC 2004*

directional dependencies, for meshes that possess poor aspect ratios, and for general implicit analysis, where the cost of solving equations can substantially increase with problem size. In order to incorporate anisotropic refinement, and to drive refinement solely with a specified error estimator, arbitrary node irregularity is allowed between elements as shown in Figure 3.



**Figure 3: Arbitrary node irregularity between refined elements.**

Figure 4 shows randomly generated isotropic and anisotropic  $h$ -refinements of a single coarse element. The sequence of meshes used to produce the fine meshes shown in the figure were subjected to the boundary conditions employed in the well-known patch test to verify that all of the various software modules were coded correctly. In particular, these tests ensured that the constraints required to enforce inter-element continuity of the nodal quantities were enforced.



**Figure 4: Isotropic and anisotropic refinement of a single hexahedral element.**

Two categories of error estimators are applied to large-deformation solid mechanics problems with contact: patch-based recovery [3,4] and residual-based [5]. Patch-based recovery procedures smooth computed solution gradients to produce an improved estimate of the true solution; the estimated error is then the difference between the

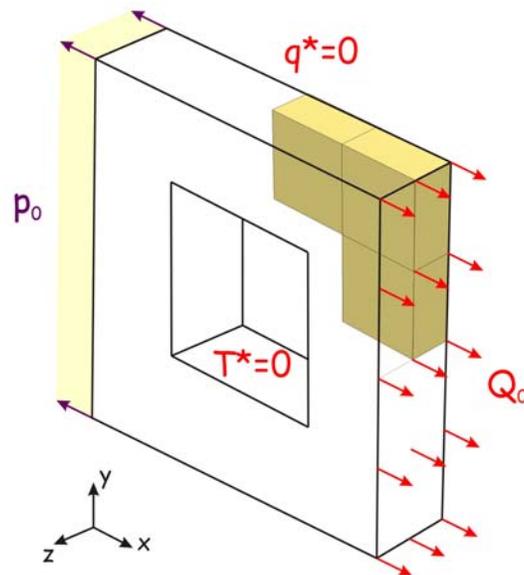
*Proceedings from the NECDC 2004*

original and smoothed solutions. Residual-based procedures reflect the degree to which the computed solution satisfies the governing equations. The two techniques tend to be complementary. Recovery procedures are inexpensive and, in general, robust even at the coarse mesh level; however, they cannot provide guaranteed bounds, since they do not use any information from the underlying equation to be solved. Residual-based procedures are generally more expensive and less robust at the coarse mesh level; however, at the fine mesh level they provide a more accurate picture of the error and, for some sets of constitutive laws and boundary conditions, can even provide guaranteed bounds. It is anticipated that a combination of the two techniques will prove to be most effective.

**Parallelization of AMR in a Message-Passing Environment**

The basic data object that is employed in the mesh refinement procedures is the AMR block. The AMR block can be viewed as a single hexahedral element on the initial coarse mesh presented for adaptive refinement. Thus, isotropic or anisotropic refinement of a mesh is implemented as refinement of the individual AMR blocks, which in turn produces new elements and nodes in the AMR database that are eventually transferred to the global mesh database. During the refinement procedures, the elements and nodes in an AMR block are referenced using two sets of octrees. Manipulation of selected elements and nodes is then readily achieved using suitable recursive procedures. After the elements within an AMR block have been refined, hanging nodes are identified by searching the nodal octree for all nodes present within the index space defined by each element in the AMR block. Non-planar geometries are resolved using Gregory patches, thereby enabling new nodes to be correctly projected to geometric boundaries and material interfaces.

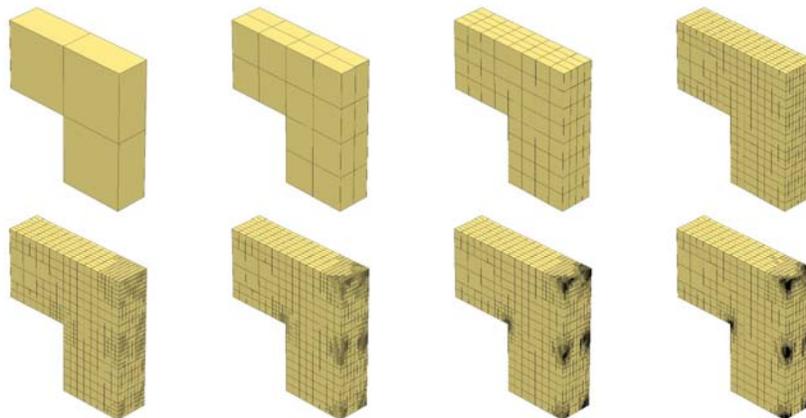
The algorithms and data structures used to manipulate multiple AMR blocks in serial mode also work in a message-passing environment, thereby efficiently enforcing inter-block constraints when the individual block data are distributed across a massively-parallel computer. This has required the development of message-passing routines that pack block data on one processor, transfer the data to the appropriate neighbor, and then unpack the data on the adjacent block face. Initial attempts at native load balancing algorithms are ongoing.



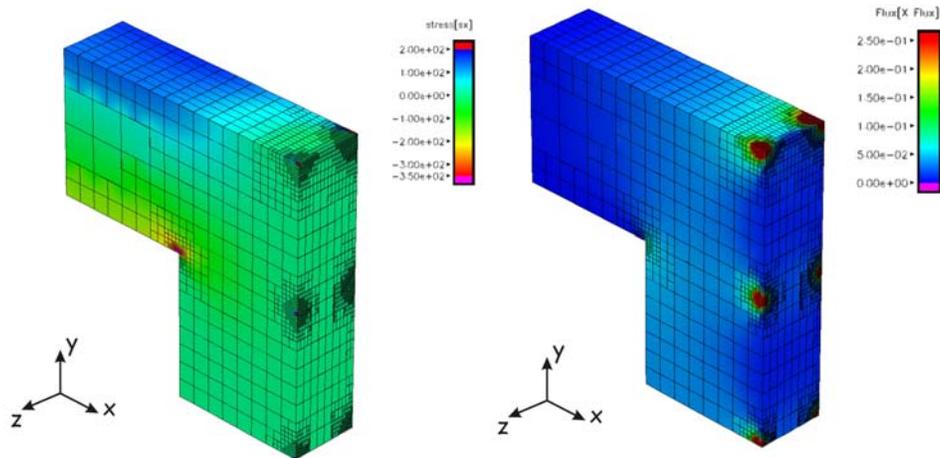
**Figure 5: Multi-mechanics AMR demonstration problem.**

### **Example 2: AMR for Thermal-Stress Analysis**

To demonstrate the combination of adaptive mesh refinement and multi-mechanics simulation in Diablo, consider the problem geometry shown in Figure 5. A square plate with a central square hole is subjected to both mechanical (uniform horizontal tractions on the outside vertical faces) and thermal (point heat fluxes located at various locations on the outside vertical faces, zero heat flux elsewhere on the outside faces and a fixed temperature on the inside surfaces) loads. Figure 6 shows the sequence of adapted meshes generated when using Diablo to solve this problem; Figure 7 shows the computed distributions of mechanical stress and thermal gradients on the finest of these discretizations.



**Figure 6: Sequence of meshes generated using Diablo AMR capability.**



**Figure 7: Stress and heat flux distribution.**

## Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

## References

- [1] Farhat, C., Lesoinne, M. and Maman, N., 1995. Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Fluids*, 21, 807-835.
- [2] Lohner, R. et al., 1995. Fluid-structure interaction using a loose coupling algorithm and adaptive unstructured grid. *Computational Fluid Dynamics Review 1995*, John Wiley, New York, 755-776.
- [3] O. C. Zienkiewicz and J. Z. Zhu, 1987. A simple error estimator and adaptive procedure for practical engineering analysis. *International Journal for Numerical Methods in Engineering*, 24, 337-357.
- [4] J. Z. Zhu and O. C. Zienkiewicz, 1990. Superconvergence recovery technique and a posteriori error estimators. *International Journal for Numerical Methods in Engineering*, 30, 1321-1339.
- [5] M. Ainsworth and J. T. Oden, 2000. *A posteriori error estimation in finite element analysis*. Wiley Interscience.