SAND96-2951
Internal Distribution Only
Printed February 1997

Modern Industrial Simulation Tools

Michael K. W. Wong and James S. Peery
Computational Physics Research and Development Department
Sandia National Laboratories
Albuquerque, NM 87185-0819

Abstract

This report documents the work performed under the project, Modern Industrial Simulation Tools: Kernel-Level Integration of High Performance Parallel Processing, Object-Oriented Numerics, and Adaptive Finite Element Analysis [1]. This project was performed under the auspices of the Department of Energy (DOE), High Performance Parallel Processing (HPPP) program. This work spans the period from April 1, 1994 to September 30, 1996.

The objective of this work was to develop a scalable parallel, object-oriented, adaptive finite element kernel. Subsequent work would meet objectives associated with developing and demonstrating software tools built on this kernel to perform advanced simulation. Due to funding and programmatic limitations, this project was terminated early. However, a basic, functioning H-P adaptive kernel was developed for two-dimensions and a prototype application code was developed on top of this kernel.

This report will briefly summarize the important technologies making up the kernel, and describe an H-Adaptive finite element prototype code developed by combining the H-P adaptive kernel and solid dynamics finite element algorithms used at Sandia.

Intentionally Left Blank

Contents

Introduction 1
Background 1
H-P Adaptivity 1
Object-Oriented Programming (OOP) 4
DARWIN H-Adaptive Prototype 4
Conclusion 8
References 9

Figures

1 Elements on either side of an edge. (a) 1:1 correspondence, (b) 2:1 correspondence. 3
2 Enforcement of 2:1 rule, (a) pending (shaded) and enforced (hatched) element refinements, (b) mesh after refinement. 3
3 Class diagram for SnlHydroEntity. 5
4 Boundary condition application (a) before adaptation, (b) refinement after boundary condition application (), (c) refinement before boundary condition application (). 6
5 Test problem for DARWIN 7
6 Test problem showing refinement front propagating from wall. 7
7 Density time history 0.2 cm from wall (dashed line: no refinement, solid line: 3 levels of refinement 8

Modern Industrial Simulation Tools

Introduction

This report documents the work performed under the project, Modern Industrial Simulation Tools: Kernel-Level Integration of High Performance Parallel Processing, Object-Oriented Numerics, and Adaptive Finite Element Analysis [1]. This project was performed under the auspices of the Department of Energy (DOE), High Performance Parallel Processing (HPPP) program. This work spans the period from April 1, 1994 to September 30, 1996.

The objective of this work was to develop a scalable parallel, object-oriented, adaptive finite element kernel. Subsequent work would meet objectives associated with developing and demonstrating software tools built on this kernel to perform advanced simulation. Due to funding and programmatic limitations, this project was terminated early. However, a basic, functioning H-P adaptive kernel was developed for two-dimensions and a prototype application code was developed on top of this kernel.

This report will briefly summarize the important technologies making up the kernel, specifically, H-P adaptive finite element technology and object-oriented programming techniques. Parallel processing technology was intended to be a part of this work. However, due to the early termination of this project, this topic was not thoroughly developed nor implemented.

Background

H-P Adaptivity

The goal of adaptive finite element methods is to optimize the solution, providing improved accuracy of the solution while maintaining efficiency of computational effort. The adaptivity method used in this work is called h-p adaptivity [2-4]. This type of adaptivity is the combination of the h- and p-methods of adaptivity.

The h-method of adaptivity controls the characteristic length, *h*, of elements of the finite element mesh. This is accomplished by refining or subdividing the elements. This refinement increases the density of elements and nodes in areas of high error. Such errors are inherent in the finite element approximation due to spatial discretization into finite elements.Thus, the smaller the elements, the higher the resolution of the spatial discretization, resulting in decreased error and increased accuracy.

The p-method of adaptivity controls the order of the approximating polynomials used within the finite elements. Generally, higher order features of calculated response may be

obtained with higher order approximating polynomials for a given element size and shape. Thus, accuracy of the solution may be increased with higher order approximations.

Often, the h- or the p-method is more appropriate for the solution in a particular region of the calculation, at a particular period of time of the calculation. Thus, the hp-method of adaptivity seeks to combine both the h- and p-methods in an optimal mix by controlling both the size of the elements and the order of the approximations, applying the method providing the most accuracy at the least cost. Depending on the situation, cost may be defined in terms of computational load, computation time, storage, or data communication loads.

For example, in a shock propagation problem, elements ahead of the shock are in a quiescent state. At the shock, however, a singularity exists. Artificial viscosity methods can smear the shock over several elements, at the expense of decreasing the local resolution of the shock front. By reducing the element size and increasing the number of nodes and elements in the region of the shock front, the artificial viscosity smearing is reduced and the discontinuity more highly resolved. Thus, h-adaptivity is preferable here. Behind the shock front, relatively constant shocked states may exist with little spatial or temporal variation of variables. Thus, the computational load may be reduced by reducing the level of refinement in this region. Accuracy may be maintained by using fewer elements of higher-order to compute the response. So, p-adaptivity may be appropriate in this region.

H-Adaptivity Concepts

As discussed above, h-adaptivity involves subdividing or refining an element to increase accuracy by increasing the density of elements and nodes in a particular region of interest. Basically, the method involves replacing an element with two or more elements within the same area (2D) or volume (3D). Variable information is transmitted from the original large element (parent) to the smaller elements (children).

There are two approaches to this type of refinement: isotropic and anisotropic. Isotropic refinement implies that child elements are created by dividing all the edges of the element in half. This creates four child elements in 2D and eight in 3D. Anisotropic refinement creates a minimum of two child elements in a selected direction. Anisotropic refinement sometimes produces similar accuracies to isotropic refinement but with fewer elements, particularly for problems in which response is strongly oriented along a particular mesh direction.

The concept of refining elements locally leads to a condition in which for a given edge, there is not a 1:1 correspondence of elements. Rather, there may be a 2:1 correspondence of elements on either side of the edge as shown in Figure 1. There are a number of methods to address the additional node. These include treating the extra node as a constrained node in which the response of the node is constrained by the nodes on either side. Alternatively, the extra node can be handled on the larger element by providing an element description allowing for five, rather than four, noded connectivity. In this work, the former method was used where the constraints imposed by the adjacent nodes are expressed as algebraic constraint equations.

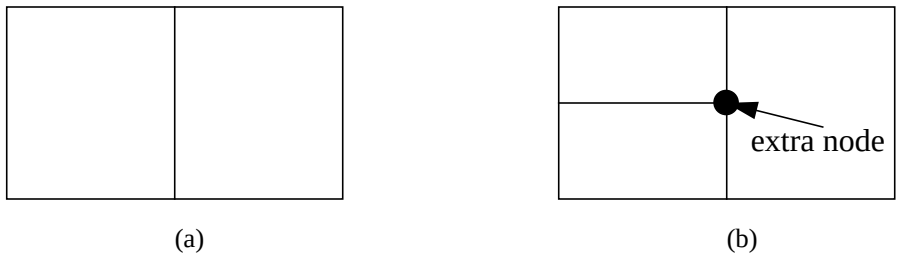


Figure 1. Elements on either side of an edge. (a) 1:1 correspondence, (b) 2:1 correspondence.

To keep the constraint equations manageable, a 2:1 rule is imposed. This means that in Figure 1(b), the elements on the left cannot be split horizontally without first splitting the element on the right. Essentially, the left edge of the right element can have at most two neighboring elements. To enforce this requirement, implies a potential cascade of refinement as shown in Figure 2.In this case, if isotropic refinement is performed on the two shaded elements, the two hatched elements will have to be refined first, in order to enforce the 2:1 relationship across an edge.

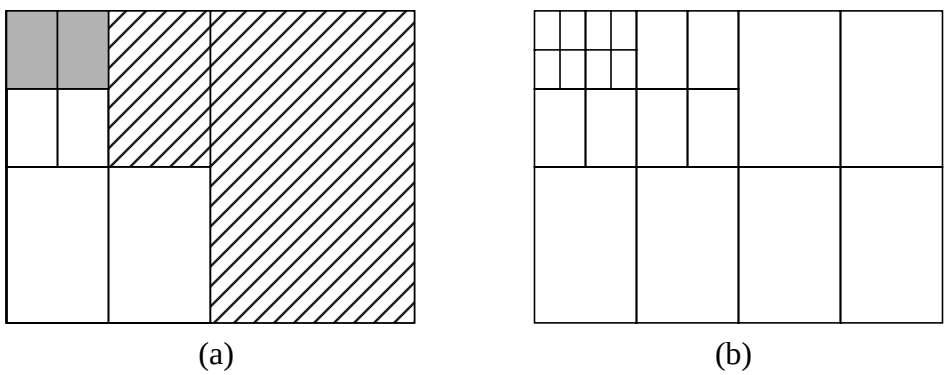


Figure 2. Enforcement of 2:1 rule, (a) pending (shaded) and enforced (hatched) element refinements, (b) mesh after refinement.

Unrefinement of elements presents additional issues regarding enforcement of the 2:1 relationship. For an element to be unrefined, its siblings must also be marked for unrefinement in the isotropic case or at least a neighboring sibling in anisotropic case. Furthermore, if the unrefinement would violate the 2:1 relationship with neighboring elements, the unrefinement must be postponed.

In general, a combination of refinements and unrefinements may occur at any step in a calculation. The determination to refine or unrefine is based on the results of an error estimation process. In areas of high error, elements are marked for refinement. In areas where the error estimated for a parent element is low, the child elements are marked for unrefinement. All pending refinement and unrefinement marks must be validated with respect to the entire mesh. Thus, after an element-by-element evaluation of error, all

pending refinement and unrefinement marks must be adjusted if necessary to produce a consistent set of refinements and unrefinements, primarily to enforce the 2:1 relationship.

Generally, refinement requirements override unrefinement requirements. For example, a refinement cascade such as in Figure 2, may cause refinement in elements adjacent to elements that were initially marked for unrefinement. Simultaneous refinement and unrefinement in these respective areas could violate the 2:1 relationship. If such a violation would occur, the unrefinement would have to be postponed.

Object-Oriented Programming (OOP)

Much of the difficulty associated with implementing adaptivity in existing finite element codes has been the difficulty in managing the dynamic nature of the topology and its effect on the data structures used in the code [3]. One of the most promising methods of managing this complexity is to use object-oriented programming techniques [5]. For this work, the C++ language was selected as it is arguably the most widely available and supported of object-oriented languages [5-7].

The most significant departure from previous h-adaptive finite element codes (particularly those written in Fortran 77 for vector processors) has been the shift of data from separate long arrays of variables, to locating data within elements, edges, or nodes. This move alleviated the traditional difficulty with reorganizing arrays following each application of adaptivity. Thus, variables could be encapsulated within the *entity* (node, element, etc.) and manipulated without worrying about indexing across a set of variable arrays.

The dynamic memory allocation features of the language permitted straightforward capabilities to create and delete entities and the associated data as required by the adaptivity process.

The major product using the OOP style was the PHLEX2000 hp-adaptive finite element kernel. This was developed from preceding work in hp-adaptive technology [8]. The software reuse concepts within the C++ language allowed this kernel to be linked with a simple Lagrangian physics kernel written at Sandia and based on finite element solution techniques used at Sandia [9]. The result was a basic prototype for an object-oriented hp-adaptive finite element program. The ultimate goal was to develop this prototype to actually utilize p-adaptivity and be capable of parallel computation. Unfortunately, the constraints of the project limited the prototype development to a basic serial h-adaptive capability.

DARWIN H-Adaptive Prototype

The prototype code utilizing the PHLEX2000 kernel with a Lagrangian physics driver was called DARWIN. It was developed at Sandia to test the capability of interfacing with the PHLEX kernel and to provide a test bed for performing h-adaptivity on shock wave simulations.

The structure of the DARWIN code was built upon three major libraries: the PHLEX2000 library, a tools library, and a graphical user interface (gui) library. The PHLEX2000 library was created at Sandia by packaging the PHLEX2000 kernel into an archive library. The tools library consisted of standard C++ utilities used at Sandia for interfacing with the EXODUS format for Sandia finite element databases. The gui library was developed by COMCO and used to facilitate the development and debugging process.

The fundamental class of DARWIN was the SnlHydroEntity, shown in Figure 3. This was derived from the SnlPhysicsEntity. The purpose of this class was to hold the data associated with the physics being performed such as density, internal energy, temperature, and deformation rate. It also contained the functions to update the data in the course of a time step such as a method for calculating deformation rate from the velocity gradient. Each of the physics entities contained a pointer to the corresponding Phlex_TopoEntity which maintained the topological and adaptivity data.

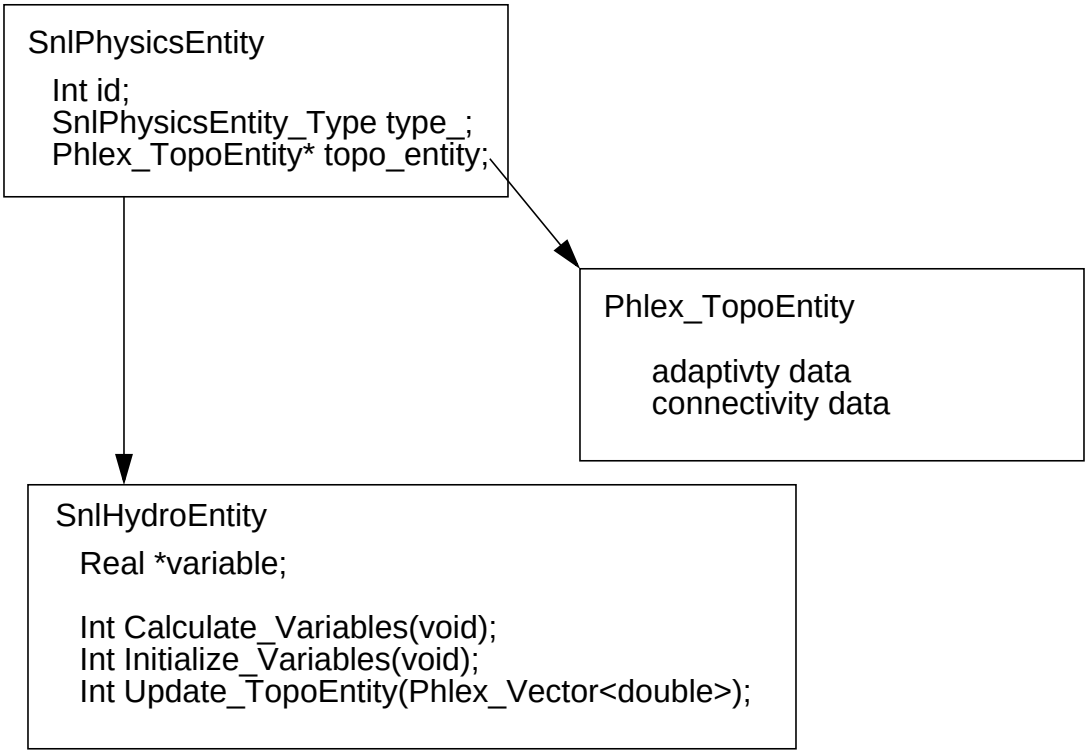


Figure 3. Class diagram for SnlHydroEntity.

Nodal data such as coordinates and velocity were maintained by the Phlex_Vertex class, a zero-order topological entity.

The solution procedure involved computation of nodal forces from the pressure stored in each SnlHydroEntity. Then, nodal forces and masses were used to compute accelerations. Boundary conditions were applied as necessary to modify the acceleration or subsequent velocities and position. With a consistent kinematic state, the deformation rate was

computed from the velocity gradient. The deformation rate was used to compute the change in element volume and thus, the change in density. This information was used to update the equation of state data, computing new updated pressure for the next cycle.

After some experimentation, it was determined that adaptivity was best performed after the unconstrained accelerations were computed from the force balance, but before boundary conditions were applied. If the adaptivity was performed after the boundary conditions were applied, then the effect of the boundary conditions would be present two nodes from the boundary as opposed to nodes immediately adjacent to the boundary as shown in. Figure 4

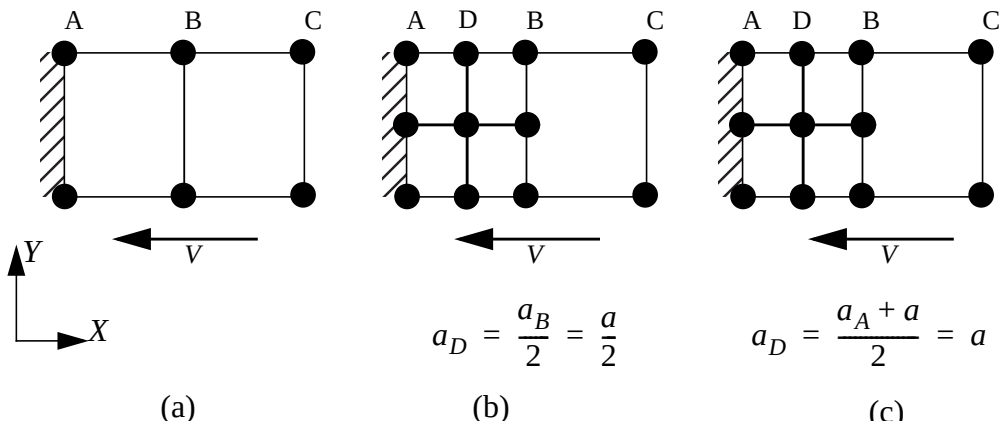


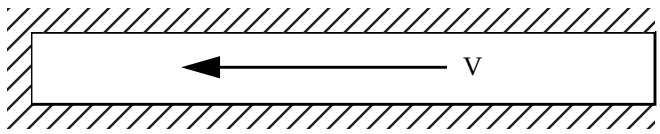
Figure 4. Boundary condition application (a) before adaptation, (b) refinement after boundary condition application ($a_A = 0$), (c) refinement before boundary condition application ($a_A \neq 0$).

In this example, assume all nodes are fixed in the y-direction. Furthermore, nodes aligned at A are fixed in the x-direction with the mesh moving toward the boundary at A with a velocity, *V*. Initially, the unconstrained momentum balance produces acceleration at all nodes, *a*. If the refinement is performed after applying boundary conditions, as shown in Figure 4(b), then the new nodes, at D, will have half of the acceleration which is produced if refinement occurs before applying boundary conditions, as in Figure 4(c).

Thus, the traditional Lagrangian momentum balance sequence was interrupted by a function referred to as the Adaptivity_Controller. The purpose of the adaptivity controller was to provide overall control of the refinement process. The first step evaluated each element for the value of an error indicator. Based on the value of the error indicator, elements and edges were marked for refinement. The second step evaluated all of the edge and element marks for consistency across the entire mesh. Marks were adjusted if necessary to produce consistent marking. Third, the refinement functionality of the Phlex kernel was called, to refine or unrefine the marked edges and elements. Upon returning from the Phlex kernel, the array of SnlHydroEntities was updated: new items were added, other items deleted, as indicated by the new array of topological element entities from the Phlex kernel. The physics variable data held by the SnlHydroEntities was updated to reflect the newly adapted state.

The error indicator was a simple method based upon the density gradient. The density contained by the SnlHydroEntities was lumped at the element nodes and assembled. The gradient operator for the elements was used to obtain the density gradient, a vector quantity. The magnitude of the gradient was used as an error indicator while the directionality of the gradient indicated refinement direction.The gradient magnitude alone was used to guide unrefinement. Additionally, the value of the error indicator had to be normalized across the mesh and subjected to thresholds to prevent local mesh refinement or computational noise.

This approach for error analysis is crude in comparison to more rigorous approaches presented by others. However, the selection of the error indicator depends, to a certain extent, on the nature of the problem being analyzed. The selected test problem for this prototype code, shown in Figure 5, was a contained column of ideal gas moving with a



Density = 1.0 g/cc
Temperature = 298 K
Velocity = 10000 cm/s

Figure 5. Test problem for DARWIN

relatively high initial velocity toward a fixed wall. Thus, the physical feature driving the refinement was a planar shock wave propagating through the gas, away from the wall.

The more sophisticated error estimators and indicators are quite applicable for parabolic or elliptic problems. However, these methods were not well suited for hyperbolic problems. In particular, these methods were not robust enough to respond to the rapidly propagating shock front. Thus, the density gradient provided a simple, easily understood alternative which worked adequately for this prototyping effort.

Figure 6 shows this test problem as the mesh refines at the stagnation jump. This refinement front propagates to the right, away from the wall. Elements in the compressed region between the wall and refinement front get unrefined. Thus, DARWIN was able to

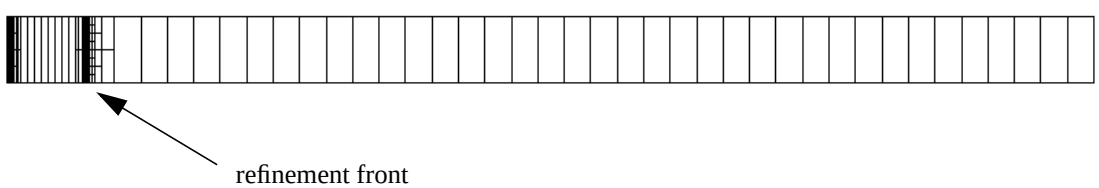


Figure 6. Test problem showing refinement front propagating from wall.

concentrate the high resolution mesh at the shock front where the computational effort needed to be focused. Figure 7 shows the increase in precision of the solution produced by three levels of refinement (an original element may refine to eventually produce great-grandchildren).

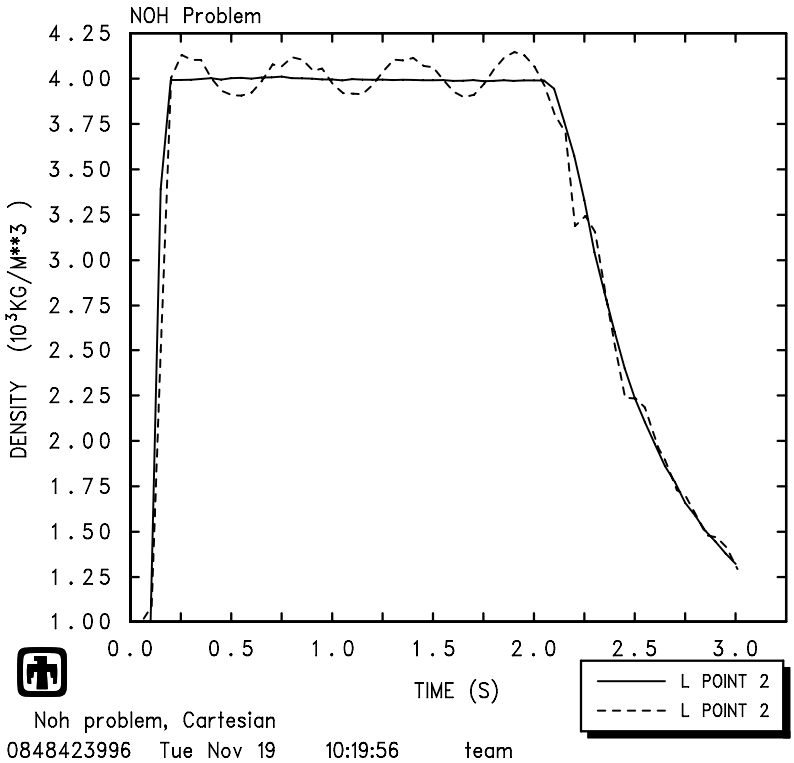


Figure 7. Density time history 0.2 cm from wall (dashed line: no refinement, solid line: 3 levels of refinement.

The fixed, single-point integration scheme and the element-centered data storage essentially fixed the order of the elements as bilinear, 4-noded quadrilaterals. Thus, this prototype was not capable of exercising the P-adaptivity to enrich the order of the elements.

Conclusion

This work successfully developed and prototyped the concepts of object-oriented programming of H-P adaptive finite element techniques. The adaptive kernel, Phlex, has potential commercial value for its parent company. The use of the Phlex adaptive kernel in the DARWIN prototype demonstrated key capabilities necessary to develop production H-adaptivity capabilities in Sandia codes in the future. Specifically, this prototype demonstrated that under-integrated, 4-noded quadrilateral finite elements, with element-centered quantities can be successfully refined and unrefined for high-strain rate hyperbolic problems. The element formulation and solution algorithms are compatible with element refinement. Furthermore, the prototype demonstrated that use of H-adaptivity with this

element formulation in an explicit time integration scheme is practical, provided proper care is exercised as to when the refinement actually occurs.

Currently, there are no plans to continue the development of DARWIN. Rather the lessons learned through this work will be applied to future work in other efforts. Meanwhile, it is anticipated that COMCO will continue development of the Phlex kernel into a full-featured commercial product.

References

[1]"Modern Industrial Simulation Tools: Kernel-Level Integration of High Performance Parallel Processing, Object-Oriented Numerics, and Adaptive Finite Element Analysis," U.S. Department of Energy, Solicitation DE-SA-03-93DP400010, 1993.

[2]T. J. Liszka, "An Introduction to H-P-Adaptive Finite Element Method," personal correspondence, September 18, 1994.

[3]L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy, "Toward a Universal h-p Adaptive Finite Element Strategy: Part 1. Constrained Approximation and Data Structure," Computational Methods in Applied Mechanics and Engineering, **77** (1989), pp 79-112.

[4]J. T. Oden, L. Demkowicz, T. Westermann, and W. Rachowicz, "Toward a Universal h-p Adaptive Finite Element Strategy: Part 2. A posteriori Error Estimates," Computational Methods in Applied Mechanics and Engineering, **77** (1989), pp 113-180.

[5]G. Booch, "Object-Oriented Design with Applications," Benjamin/Cummings Publishing, 1991.

[6]S. B. Lippman, "C++ Primer," 2nd ed., Addison Wesley, 1991.

[7]J. O. Coplien, "Advanced C++ Programming Styles and Idioms," Addison Wesley, 1992.

[8]"PHLEXsolid User Manual," Computational Mechanics Company, Inc., 1994.

[9]L.M. Taylor and D. P. Flanagan, "PRONTO3D: A Three-Dimensional Transient Solid Dynamics Program," SAND87-1912, 1989.