TOWARD PARALLEL, ADAPTIVE MESH REFINEMENT FOR CHEMICALLY REACTING FLOW SIMULATIONS[1]

K.D. Devine[2], J.N. Shadid[2], A.G. Salinger[2], S.A. Hutchinson[2], and G.L. Hennigan[3]

ABSTRACT: Adaptive numerical methods offer greater efficiency than traditional numerical methods by concentrating computational effort in regions of the problem domain where the solution is difficult to obtain. In this paper, we describe progress toward adding mesh refinement to **MPSalsa**, a computer program developed at Sandia National Laboratories to solve coupled three-dimensional fluid flow and detailed reaction chemistry systems for modeling chemically reacting flow on large-scale parallel computers. Data structures that support refinement and dynamic load-balancing are discussed. Results using uniform refinement with mesh sequencing to improve convergence to steady-state solutions are also presented.

1. INTRODUCTION

Adaptive mesh refinement has been used with great success for a variety of applications [1, 5]. With adaptive refinement, elements are subdivided into smaller elements in regions of the problem domain where greater resolution is needed. In this way, computational effort is concentrated in regions where it is most needed, without wasting high-resolution computation in other regions. While adaptive mesh refinement has been widely used on serial computers, its use on parallel computers is complicated by the need for distributed data structures, dynamic load balancing, data migration, and maintenance of a distributed element database. In this paper, we describe our efforts toward implementing parallel mesh refinement in **MPSalsa**, an unstructured finite element computer program developed at Sandia National Laboratories to solve coupled three-dimensional fluid flow and detailed reaction chemistry systems for modeling chemically reacting flow on large-scale parallel computers [12, 13]. In particular, we describe the data structure design to support refinement and dynamic load balancing, the octree data structure used to store refined meshes, and issues that arise due to the implementation of refinement in parallel. We have implemented uniform refinement of 2D and 3D meshes as a step toward adaptive refinement. This step is a useful one. Through mesh sequencing, uniform refinement can be used to accelerate convergence of the nonlinear solver and enable steady-state solution of problems too difficult to solve from trivial initial guesses. We present three examples of the benefits of mesh sequencing.

2. OVERVIEW OF MPSALSA

MPSalsa computes the solution of the conservation equations for momentum, total mass, thermal energy, and individual gas and surface phase chemical species for low Mach number flows. These equations, shown in Table 1, form a complex set of coupled, nonlinear PDEs. The continuous problem is spatially approximated by a Petrov-Galerkin finite element method. For transient problems, this spatial approximation is coupled with first- and second-order dynamically controlled time-stepping methods. Necessary transport properties, diffusion coefficients, kinetic rate constants and diffusion velocities are obtained from the CHEMKIN [10] library. The resulting nonlinear system is solved by an inexact Newton method with back-tracking [14]. In the inexact Newton method, nonlinear residual information is used to determine the accuracy to which the linear subproblems are solved. Back-tracking is a technique for further improving the robustness of the nonlinear solver. It shortens a Newton step as needed to ensure that the nonlinear residual has been reduced adequately before the step is accepted.

The **Aztec** library [9] of parallel preconditioned Krylov techniques is used to solve the resulting linear equations. The parallel Krylov algorithms implemented in **Aztec** include conjugate gradient, conjugate gradient squared, generalized minimal residual (GMRES) and transpose-free quasi-minimal residual methods. The available preconditioners are row sum and block Jacobi scaling, block Jacobi preconditioning, Neumann series and least-squares polynomial methods, and many additive Schwarz domain decomposition preconditioners using various incomplete LU factorizations with variable overlap.

1. This work was partially funded by the U.S. Department of Energy's Mathematical, Information and Computational Sciences Division, and was carried out at Sandia National Laboratories operated for the U.S. Department of Energy under contract no. DE-ACO4-94AL85000.

2. Sandia National Laboratories, Albuquerque, NM, 87185-1111, kddevin@cs.sandia.gov.

3. New Mexico State University, Las Cruces, NM.

Momentum	$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla\bullet(\rho \mathbf{u}\mathbf{u}) - \nabla\bullet\mathbf{T} - \rho\mathbf{g} = 0$
Total Mass	$\frac{\partial \rho}{\partial t} + \nabla\bullet(\rho \mathbf{u}) = 0$
Thermal Energy	$\hat{C}_p\left[\frac{\partial(\rho T)}{\partial t} + \nabla\bullet(\rho \mathbf{u} T)\right] = -\nabla\bullet\mathbf{q} + \Phi + \dot{Q} + \sum_{k=1}^{N_s} h_k \nabla\bullet\mathbf{j}_k - \sum_{k=1}^{N_s} h_k W_k \dot{\omega}_k$
Species Mass Fraction for Species k	$\frac{\partial(\rho Y_k)}{\partial t} + \nabla\bullet\rho\mathbf{u}Y_k = -\nabla\bullet\mathbf{j}_k + W_k\dot{\omega}_k,\ k = 1, 2, \ldots, N_s - 1$, and $Y_{N_s} = 1 - \sum_{k=1}^{N_s - 1} Y_k$

Table 1. Governing Conservation Equations

We use **Chaco** [8], a general graph partitioning tool, to divide the initial finite element mesh into subdomains that are assigned to processors. **Chaco** constructs subdomain mappings that have low communication volume, good load balance, few message start-ups and only small amounts of congestion. It supports a variety of new and established graph partitioning heuristics, such as spectral techniques, geometric methods, multilevel algorithms and the Kernighan-Lin method. For the results in this paper, multi-level methods with Kernighan-Lin improvement were used.

3. Implementation of Parallel Mesh Refinement

MPSalsa was originally written with array-based data structures. For example, data such as nodal coordinates, global node numbers and global element numbers were stored in individual arrays. With mesh refinement, however, new node and element information needs to be created and added to the mesh data structures. Moreover, with dynamic load balancing, movement of data between processors requires entries to be added to and deleted from a processor's local arrays. To simplify these processes, we have redesigned **MPSalsa** to used dynamic tree-based data structures instead of arrays. These data structures allow easy insertion and deletion of entities (such as nodes and elements) during refinement, coarsening and data migration. They also enable efficient searches of the entities by global number or position.

We have also made **MPSalsa**'s data structures more object-oriented. Instead of storing entity information in many identically indexed arrays, the new data structures collect all information for a mesh entity into one contiguous structure. For example, a node's structure consists of its local and global node numbers, its coordinates in the domain, information about its boundary conditions and unknowns, and a pointer to the solution vector for its unknowns. The object-oriented data structures simplify data migration during dynamic load balancing, since all of an entity's data can be copied from one location rather than gathered from several different arrays. Moreover, with the object-oriented data structures, cache usage during the matrix-fill portion of the finite element code improved, since data for nodes and elements were stored contiguously in memory.

We chose the octree (quadtree) data structure to implement mesh refinement for 3D (2D) meshes. The octree data structure has been used in many serial adaptive mesh refinement codes [5] and mesh generators [2]. Work has begun to implement parallel versions of the octree for mesh generation [15]. In the octree data structure, coarse-mesh elements are divided into some number of finer elements. These fine elements are stored as "children" of the coarse-mesh "parent" element in a hierarchical tree data structure (see Figure 1). While this data structure has higher storage and bookkeeping overhead than other refinement methods [3], it has several advantages that make it attractive. Coarsening a refined mesh can be easily done by removing child elements from the mesh and the octree data structure. Since information is available on several mesh levels, meshes of differing resolutions may be used to solve different aspects of a problem. And since the octree maintains connectivity information between generations, multi-level preconditioners and solution methods can be implemented in a straight-forward manner.

MPSalsa and **Aztec** use a nodal decomposition of the problem domain; each node is "owned" by a unique processor. All contributions to the equations associated with a node's unknowns are computed by its owning processor. Along processor subdomain boundaries, external nodes' values are communicated from their owning processors and used to contribute to owned nodes' equations. As a result of this nodal decomposition, newly created nodes must be assigned uniquely to processors. A heuristic assignment

DISCLAIMER

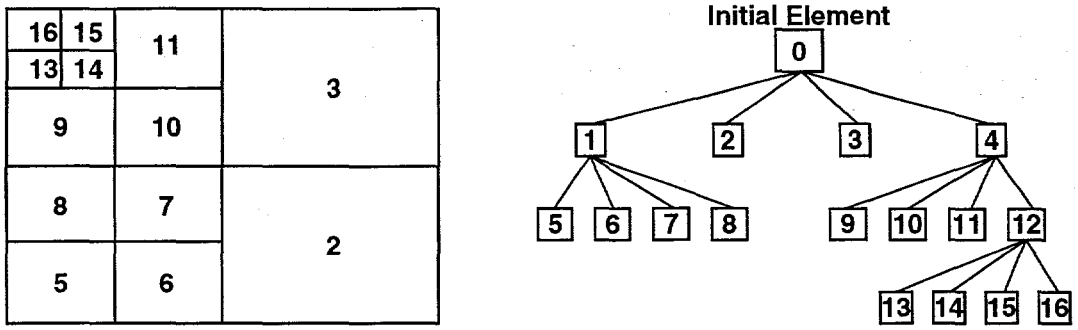

Figure 1. Adaptive refinement of an initial element (element 0) into three levels (left) and the corresponding quadtree data structure (right).

strategy is currently used. Each new node along an edge is assigned to the lowest-numbered processor owning a node of the parent element along the edge. Similarly, each new node on a surface is assigned to the lowest-numbered processor owning a node on the parent's surface. Center nodes are assigned to the processor owning node 0 of the parent element. Using this heuristic, unique assignment of new nodes to processors can be accomplished without communication between processors. This scheme also maintains the communication patterns of the parent mesh; i.e., a processor communicates with the same neighboring processors for both the coarse and the resulting fine meshes. However, load balance is not taken into account with this strategy. Some processors may receive significantly more new nodes than other processors. The effects of this load imbalance can be seen in some of the following experiments with uniform refinement.

4. Mesh Sequencing Experiments

Mesh sequencing is a technique for accelerating the convergence of the nonlinear solver for steady-state problems. In mesh sequencing, a solution is found on a coarse mesh. The coarse mesh is then refined uniformly (e.g., see Figure 2). The solution on the coarse mesh is interpolated (using the finite element basis functions) to the fine mesh. The interpolated solution is used by the nonlinear solver as an initial guess on the fine mesh. In this way, a fairly good initial guess to the desired fine-mesh solution can be obtained inexpensively on the coarse mesh, thus decreasing the amount of work needed to obtain the fine-mesh solution. Moreover, for some highly nonlinear problems, trivial initial guesses do not produce converged solutions on fine meshes, but the improved initial guesses obtained from coarse-mesh solutions enable convergence on the fine meshes.

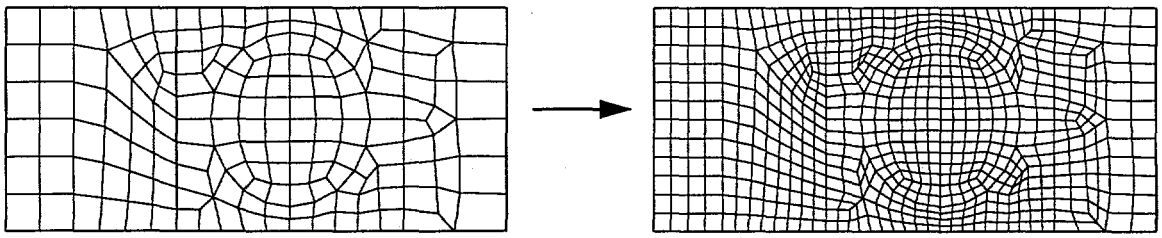

Figure 2. Uniform refinement of an unstructured quadrilateral mesh for mesh sequencing.

In the following examples, a GMRES linear solution algorithm with a Krylov sub-space size of 200 was used. Incomplete LU preconditioning with domain overlap and node reordering was applied to the linear systems. Backtracking and the inexact Newton method were used in the first two examples; in Example 3, we used an exact Newton method without backtracking. A trivial initial guess $\mathbf{x} = 0$ was used for all coarse-mesh solutions and non-sequenced runs. In each case, we compare the number of Newton steps, the number of iterations of the linear solver, the number of backtracking steps (where appropriate), the load balance θ given by

$$\theta = \frac{\text{average number of nodes per processor}}{\text{maximum number of nodes per processor}},$$

and the total execution time for sequenced and non-sequenced runs. Sequenced mesh runs are specified by the sizes of the meshes separated by an arrow; e.g., $50 \times 50 \rightarrow 100 \times 100$ indicates a mesh sequence from a

50×50 -element mesh to the 100×100 -element mesh obtained from one level of refinement. Statistics for a sequenced run are given as sums, where each summand applies respectively to a mesh of the sequence.

Example 1. Lid Driven Cavity

In this standard benchmark problem [7, 11], the momentum transport and total mass conservation equations defined in Table 1 are solved on a unit square to simulate confined flow driven by a moving boundary ($u_1 = 4,000$) on the upper wall. No-slip boundary conditions are applied on all other walls. Other parameters are chosen so that the Reynolds number Re = 4,000. A solution computed on a 200×200 -element mesh is shown in Figure 3.

We solved this problem using mesh sequencing on 50×50 -, 100×100 -, and 200×200 -element meshes. In Table 2, we show the resulting solver statistics and solution times (on 20 processors of the Intel Paragon) for each mesh. Because the problem is highly nonlinear and the load balancing for the fine meshes was good ($\theta \geq 0.92$), mesh sequencing was highly effective. The mesh sequence $50 \times 50 \rightarrow 100 \times 100 \rightarrow 200 \times 200$ reduced the execution time 48% relative to solving the problem from a trivial initial guess on a 200×200 -element mesh.

Mesh or Mesh Sequence	Number of Newton Steps	Number of Linear Iterations	Number of Backtracking Steps	Load Balance θ	Total Solution Time (secs.)
50x50	42	1368	20	0.99	108.57
100x100	40	2192	19	1.00	500.87
200x200	38	4719	17	1.00	3070.37
50x50 → 100x100	42 + 22	1368 + 1566	20 + 4	0.99 → 0.95	432.55
50x50 → 200x200	42 + 24	1368 + 2948	20 + 6	0.99 → 0.92	1950.87
100x100 → 200x200	40 + 16	2192 + 2924	19 + 2	1.00 → 0.97	2057.37
50x50 → 100x100 → 200x200	42 + 22 + 15	1368 + 1566 + 1829	20 + 4 + 1	0.99 → 0.95 → 0.92	1578.98

Table 2. Mesh Sequencing Experiments for Example 1.

Example 2. Thermal Convection

In this problem, we model thermal convection (or buoyancy-driven) flow in a differentially heated square box in the presence of gravity. The momentum transport, energy transport and total mass conservation equations defined in Table 1 are solved on a unit square. No-slip boundary conditions are applied on all walls. The temperature on the heated wall and other parameters are chosen so that the Rayleigh number Ra = 1,000,000. In Figure 4, we show the solution of this problem on a 128×128 -element mesh. Thermal boundary layers form along the hot and cold walls of the domain.

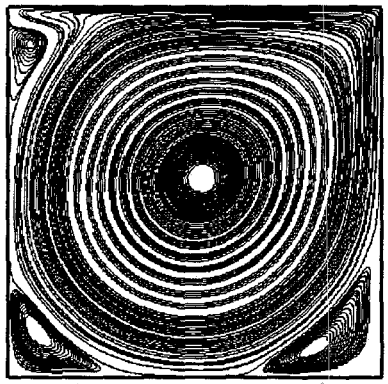

Figure 3. Contour plot of the stream function for Example 1 at Re = 4,000.

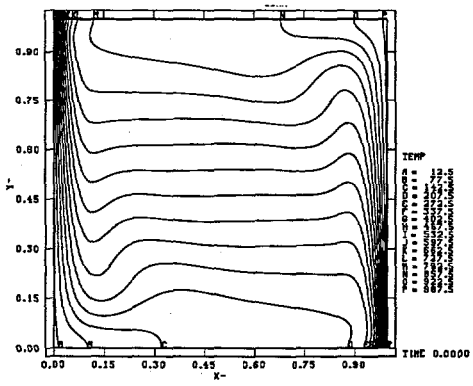

Figure 4. Contour plot of temperature for Example 2 at Ra =1,000,000.

Mesh sequencing was used to find solutions to this problem on 64×64, 128×128, and 256×256 -element meshes. In Table 3, we show the resulting solver statistics and solution times (on 64 processors of the Intel Paragon) for each mesh. Using mesh sequencing from a 128×128 -element mesh to the 256×256 -element mesh improved the solution time by 44.9% compared to solving the problem directly on

a 256×256-element mesh. Mesh sequencing from a 64×64-element mesh to the 256×256-element mesh also showed over 40% reduction of execution time relative to solving the fine-mesh problem from a trivial initial guess. However, the superior load balance and initial guess of the $128 \times 128 \rightarrow 256 \times 256$ sequence made that sequence more effective.

Mesh or Mesh Sequence	Number of Newton Steps	Number of Linear Iterations	Number of Backtracking Steps	Load Balance θ	Total Solution Time (secs.)
64x64	27	1761	8	0.97	53.8
128x128	30	2047	13	1.00	228.09
256x256	35	7773	11	1.00	1618.05
64x64 → 128x128	27 + 5	1761 + 1058	8 + 0	0.97 → 0.85	128.47
64x64 → 256x256	27 + 11	1761 + 3647	8 + 0	0.97 → 0.78	959.19
128x128 → 256x256	30 + 10	3047 + 3311	13 + 0	1.00 → 0.94	906.94

Table 3. Mesh sequencing experiments for Example 2.

Example 3. Tilted Chemical Vapor Deposition (CVD) Reactor

We demonstrate the effectiveness of mesh sequencing for 3D unstructured meshes using the Tilted CVD Reactor. In Figure 5, we show the reactor configuration and an example solution in the reactor. The reactants enter the reactor through the rectangular region on the left. The rectangular susceptor has a disk in its middle that spins to produce more uniform deposition. The base of the reactor slopes up at a nine-degree angle to accelerate the flow and shrink the boundary layer over the disk; these effects offset the decline in the deposition rate down the length of the reactor due to reactant consumption. A reaction mechanism with four chemical species was used, resulting in nine unknowns per node.

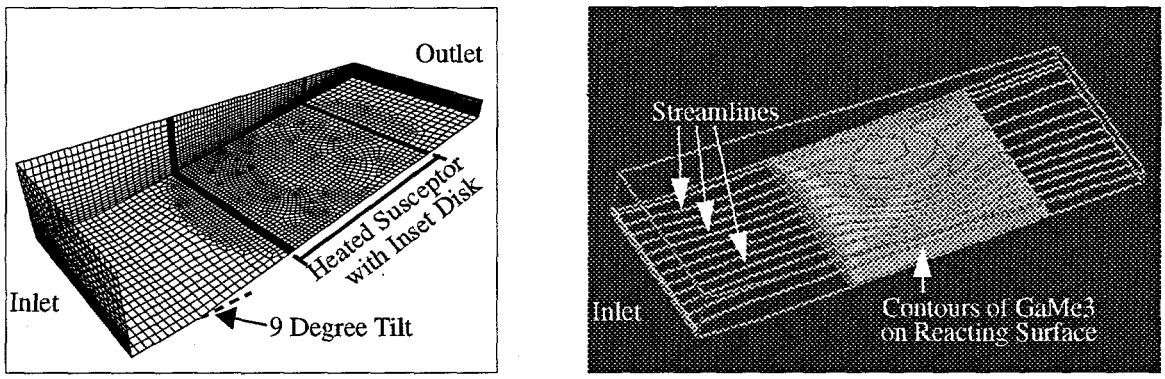

Figure 5. Reactor configuration and example solution for the Tilted CVD Reactor used in Example 3.

We solved this problem with an SGI workstation using a 336-element coarse mesh. We uniformly refined the coarse mesh to produce a 2688-element fine mesh. We computed steady-state solutions both from a trivial initial guess on the fine mesh and via mesh sequencing from the coarse mesh to the fine mesh (see Table 4). By using mesh sequencing, we reduced the execution time needed to obtain a fine-mesh solution by 39.5% relative to the use of the trivial initial guess.

Mesh or Mesh Sequence	Number of Unknowns	Number of Newton Steps	Number of Linear Iterations	Total Solution Time (secs.)
336 elements	5211	8	23	155.88
2688 elements	32,445	9	97	3645.48
336 → 2688 elements	5211 → 32,445	8 + 5	23 + 49	2205.85

Table 4. Mesh Sequencing Experiments for Example 3.

The need for dynamic load balancing or a more sophisticated assignment of nodes to processors is more evident for this 3D problem than for the 2D problems presented earlier. In experiments with this

example run on 16 processors of the Intel Paragon, load imbalance was costly. For the trivial initial guess solution, the fine mesh was perfectly balanced among the processors by **Chaco**; i.e., $\theta = 1.0$. The resulting execution time was 827 seconds. With mesh sequencing, the coarse mesh was also perfectly balanced among the processors; the coarse mesh solution required 70 seconds. However, when the coarse mesh was refined, the assignment of newly created nodes to processors resulted in load imbalance: $\theta = 0.74$. As a result, the total execution time using mesh sequencing was 901 seconds.

5. Conclusions and Future Work

We have described our progress toward the implementation of parallel mesh refinement in **MPSalsa**. To date, parallel uniform refinement has been implemented. We have used uniform refinement for mesh sequencing, a technique that accelerates convergence to steady-state solutions. Experiments with mesh sequencing have shown reductions in execution time of approximately 40% on 2D and 3D problems.

In our future work, we will continue toward parallel adaptive mesh refinement by designing refinement strategies and implementing modifications to the finite element assembly needed along interfaces between coarse and fine elements. We will investigate effective error estimation techniques for reacting flows and use them to guide the adaptive refinement. And we will implement dynamic load balancing and data migration strategies along the lines of [4, 6] to redistribute load imbalance created by adaptive refinement.

6. References

1. I. Babuska, O.C. Zienkiewicz, J. Gago, and E.R. de A. Oliveira, editors. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations.* John Wiley & Sons, Ltd. (1986).
2. P.L. Baehmann, S.L. Wittchen, M.S. Shephard, K.R. Grice, and M.A. Yerry. "Robust geometrically based automatic two-dimensional mesh generation." *Int. J. Numer. Meths. Engrg.*, **24** (1987) 1043-1078.
3. M.J. Berger and J. Oliger. "Adaptive mesh refinement for hyperbolic partial differential equations." *J. Comput. Phys.*, **53** (1984) 484-512.
4. R. Biswas and L. Oliker. "Load balancing unstructured adaptive grids for CFD problems." Proceedings of the 8th SIAM Conf. on Parallel Processing for Scientific Computing, Minneapolis, MN, March, 1997.
5. K. Clark, J.E. Flaherty and M.S. Shephard, editors. Special Issue on Adaptive Methods for Partial Differential Equations. *Appl. Numer. Math.*, **14** (1994).
6. H.L. deCougny, K. Devine, J.E. Flaherty, R. Loy, C. Ozturan, and M.S. Shephard. "Load balancing for the parallel adaptive solution of partial differential equations." *Appl. Numer. Math.*, **16** (1994) 157-182.
7. U. Ghia, K.N. Ghia, and C.T. Shin. "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method." *J. Comput. Phys.*, **48** (1982) 387-411.
8. B. Hendrickson and R. Leland. "A user's guide to Chaco, Version 1.0." Sandia National Laboratories Technical Report, SAND93-2339, Albuquerque, NM, (1993).
9. S. A. Hutchinson, J. N. Shadid, and R. S. Tuminaro. "Aztec User's Guide: Version 1.0." Sandia National Laboratories Technical Report, SAND95-1559, Albuquerque, NM, (1995).
10. R.J. Kee, F.M. Rupley, and J.A. Miller. "Chemkin II: A Fortran chemical kinetics package for the analysis of gas-phase chemical kinetics." Sandia National Laboratories Technical Report, SAND89-8009, Albuquerque, NM, (1989).
11. R. Schreiber and H.B. Keller. "Driven cavity flows by efficient numerical techniques." *J. Comput. Phys.*, **49** (1983) 310-333.
12. J. N. Shadid, H. K. Moffat, S. A. Hutchinson, G. L. Hennigan, K. D. Devine, and A. G. Salinger. "MPSalsa: A Finite Element Computer Program for Reacting Flow Problems: Part 1 - Theory Document." Sandia National Laboratories Technical Report, SAND95–2752, Albuquerque, NM, (1996).
13. A.G. Salinger, K.D. Devine, G.L. Hennigan, H.K. Moffat, S.A. Hutchinson, and J.N. Shadid. "MPSalsa: A Finite Element Computer Program for Reacting Flow Problems: Part 2 -- User's Guide and Examples." Sandia National Laboratories Technical Report, SAND96-2331, Albuquerque, NM, (1996).
14. J.N. Shadid, R.S. Tuminaro and H.F. Walker. "An Inexact Newton Method for Fully-Coupled Solution of the Navier-Stokes Equations with Heat and Mass Transport." Sandia National Laboratories Technical Report, SAND97-0132, Albuquerque, NM, (1997).
15. M.L. Simone, H.L. deCougny, and M.S. Shephard. "Tools and techniques for parallel grid generation." Proc. of the 5th Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Starkville, MS, April, 1996.