A Parallel Cartesian Approach for External Aerodynamics of Vehicles with Complex Geometry

M. J. Aftosmis
Mail Stop T27B
NASA Ames Research Center
Moffett Field, CA 94404
aftosmis@nas.nasa.gov

M. J. Berger and G. Adomavicius
Courant Institute
12 Mercer St.
New York, NY 10012
berger@cims.nyu.edu

ABSTRACT

This workshop paper presents the current status in the development of a new approach for the solution of the Euler equations on Cartesian meshes with embedded boundaries in three dimensions on distributed and shared memory architectures. The approach uses adaptively refined Cartesian hexahedra to fill the computational domain. Where these cells intersect the geometry, they are cut by the boundary into arbitrarily shaped polyhedra which receive special treatment by the solver. The presentation documents a newly developed multilevel upwind solver based on a flexible domain-decomposition strategy. One novel aspect of the work is its use of space-filling curves (SFC) for memory efficient on-the-fly parallelization, dynamic re-partitioning and automatic coarse mesh generation. Within each subdomain the approach employs a variety reordering techniques so that relevant data are on the same page in memory permitting high-performance on cache-based processors. Details of the on-the-fly SFC based partitioning are presented as are construction rules for the automatic coarse mesh generation. After describing the approach, the paper uses model problems and 3-D configurations to both verify and validate the solver. The model problems demonstrate that second-order accuracy is maintained despite the presence of the irregular cut-cells in the mesh. In addition, it examines both parallel efficiency and convergence behavior. These investigations demonstrate a parallel speed-up in excess of 28 on 32 processors of an SGI Origin 2000 system and confirm that mesh partitioning has no effect on convergence behavior.

INTRODUCTION

Recent years have witnessed the rapid maturation of embedded-boundary Cartesian approaches. The work in references [1]-[7] (among many others) demonstrate that the approach can be used to robustly compute flows around vehicles with a high degree of geometric complexity. This strength is largely due to the underlying observation that cells in these meshes are purely Cartesian (away from geometry) or arbitrarily shaped polyhedra (where initially Cartesian hexahedra are clipped against the body's surface). Figure 1 illustrates the types of cells found in these meshes. Note that *cut-cells* as shown in fig. 1b may be split into any number of unconnected regions by the geometry, such *split-cells* imply that the index space of the Cartesian hexahedra will not, in general, be the same as that of the control volumes integrated by the solver.

The observation that cells in a Cartesian mesh are either cut or un-cut has important implications for both mesh generation and solver efficiency. Since cut-cells are assumed to be arbitrarily shaped, the geometric complexity of a particular configuration does not impact the mesh generation process, and thus mesh gener-

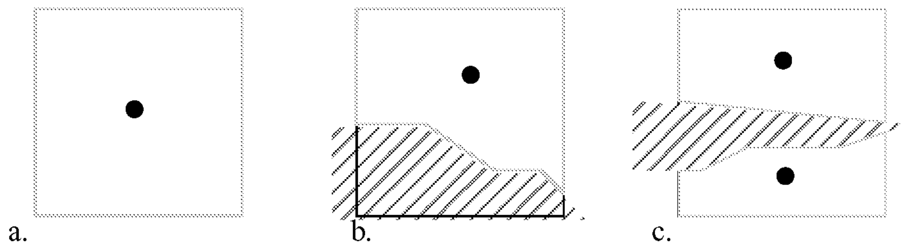


Figure 1: Types of cells in Cartesian meshes with embedded boundaries: a) a *volume cell*, b) a *cut-cell*, c) a *split-cell* cut into two polyhedra.

ation systems – like those in [2], [4], and [6] can be fully automated. Moreover since the vast majority of the domain is discretized with simple hexahedra, the process can be extremely fast. As an example, the mesh generator in ref. [2] produced approximately 1 x 10^6 cells/minute on moderately powered desktop workstations in 1997[2].

Advocates of Cartesian approaches often note that solvers which take advantage of these meshes may use simplified discretization formulae in the pure Cartesian cells off-body and yet still take extra care to accurately integrate the cut-cells which have embedded geometry. Such arguments note that un-cut cells fill the volume of space around the geometry. Thus, while a typical mesh may contain $O(N^3)$ off-body cells, only $O(N^2)$ cut-cells actually intersect the body itself. Following this reasoning, one sees that since a simplified form of the spatial discretization operator is applied to the vast majority of the cells in the domain. The net savings in operation count can be dramatic. In addition, throughout much of the domain, the solver operates on pure Cartesian meshes. Without mesh skewing or stretching to hinder performance or stability, the solver therefore may achieve its full order of accuracy in cells with purely Cartesian stencils.

While Cartesian mesh generators have largely overcome an important obstacle in the CFD process, solvers which take full advantage of the approach have been less convincingly documented. Moreover, removal of the mesh generation bottleneck from the analysis cycle places a renewed emphasis on flow solver efficiency. The current research explores the issues of accuracy and efficiency. The approach uses domain-decomposition to target the current crop of shared and distributed memory computing platforms, and multilevel smoothing to enhance convergence. Wherever possible, the solver uses an appropriately simplified operator for the spatial discretization of the pure Cartesian cells. In this workshop paper, we present a brief outline of the finite-volume discretization and multigrid scheme before shifting focus to the domain decomposition and coarse mesh generation. Results are presented for a variety of model problems and 3-D configurations, and these provide a basis for a preliminary assessment of the accuracy and efficiency of the solver.

SPATIAL AND TEMPORAL DISCRETIZATION

Embedded boundary Cartesian approaches discretize the computational domain with either "volume cells" which are the adaptively Cartesian hexahedra filling the space away from boundaries, and "cut-cells" which are formed by the Cartesian cells which actually intersect the surface. As shown by Figure 1, volume cells always have six coordinate aligned faces, while cut-cells are considered to be arbitrarily shaped polyhedra. "Split-cells" refers to a subset of cut-cells which are actually split into multiple, non-communicating, flow polyhedra by the geometry. The solver uses a cell-centered finite-volume scheme for the spatial discretization with the state vector stored at the cell-center of each of the Cartesian hexahedra. In boundary cut-cells, these quantities are stored at the centroid of the actual polyhedron formed by the intersection of the Cartesian cell with the body. The fact that some cut-cells may indeed be split-cells indicates that the index space of the control volumes is not necessarily the same as that of the set of Cartesian hexahedra from which the mesh was constructed.

Within each control volume, the spatial integration scheme proceeds by traversing a face-based data structure to reconstruct a piecewise linear polynomial distribution of each state variable within the cell as in the linear-reconstruction approach of ref. [8]. A least–squares procedure is used to provide gradient estimates within each cell based on solution of the normal equations of the local mass matrix. State vectors are reconstructed from the cell centroids to the face centroids, and the flux quadrature uses a midpoint integration coupled with either a van Leer flux-vector splitting, or the approximate Riemann solver of Colella[9].

Evolution is performed using a modified Runge-Kutta scheme to drive a recursively implemented FAS (Full Approximation Storage) multigrid scheme[10]. This scheme may be used in conjunction with a local block Jacobi preconditioner which requires the inversion of 5 x 5 matrix for each control volume in the computational domain[11]. When coupled with the upwind spatial discretization, this preconditioner has been shown to efficiently cluster the residual eigenvalues for rapid annihilation by the multigrid scheme[13]. Implementation of such a preconditioner is planned in the near future.

Further details of the spatial and temporal operators and aspects of its implementation which impact the overall efficiency of the approach will be presented in an upcoming paper[12].

DOMAIN DECOMPOSITION

One novel aspect of this work lies in its approach toward domain decomposition. The option exists to apply a commercial grade uni-processor partitioner like the multi-level nested dissection tool in reference [14] or its multi-processor variant[15]. However, an attractive alternative stems from exploiting the nature of Cartesian meshes. We have built-in a partitioner based upon the use of space-filling curves, constructed using either the Morton or Peano-Hilbert orderings[16]. Both of these orderings have been used for the parallel solution of *N*-body problems in computational physics[17], and the later scheme has been proposed for application to algebraic multigrid[18] in the solution of elliptic PDEs and dynamic repartitioning of adaptive methods[19]. Figure 2 shows both Peano-Hilbert and Morton space-filling curves constructed on Cartesian meshes at three levels of refinement. In two dimensions, the basic building block of the Hilbert curves is a “U” shaped line which visits each of 4 cells in a 2 x 2 block. Each subsequent level divides the previous level’s cells by nested dissection, creating subquadrants which are, themselves, visited by U shaped curves as well. This “U-ordering” has locality properties which make it attractive as a partitioner[19]. Similar properties exist for the Morton ordering which uses an “N” shaped curve as its basic building block. Properties and construction

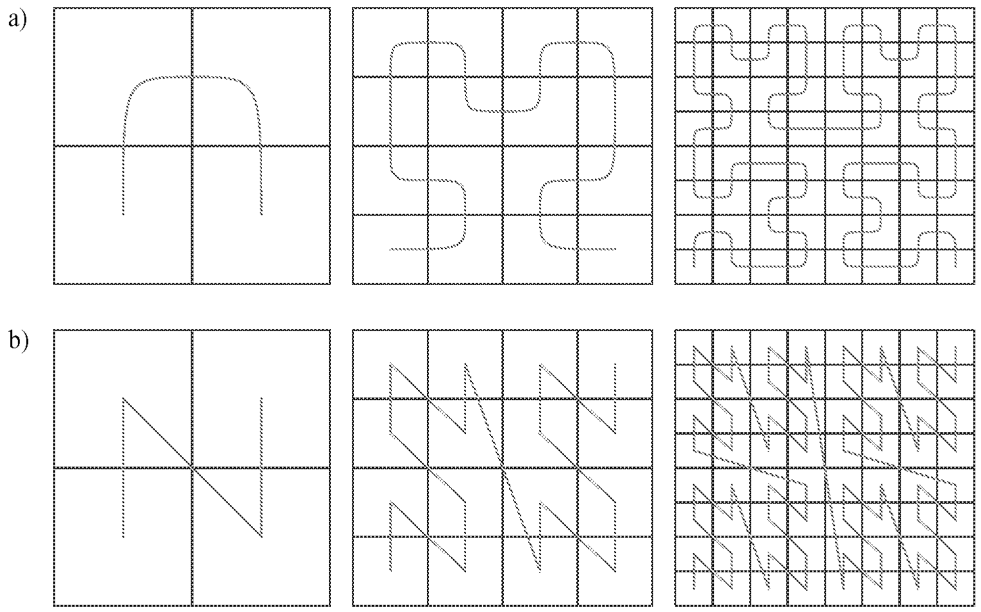


Figure 2: Space-filling curves used to order three Cartesian meshes in two spatial dimensions: a) Peano-Hilbert or “U-ordering”, b) Morton or “N-ordering”.

rules for these space-filling curves are discussed in refs. [20] and [21]. For the present, we note only that such orderings have 3 important properties.

1. **Mapping** $\Re^d \to \boldsymbol{U}$: The U and N orderings provide a unique mappings from the d-dimensional physical space of the problem domain $\Re^d$ to a one-dimensional hyperspace, ***U,*** which one traverses following the curve. In the U-order, two cells adjacent on the curve remain neighbors in this one-dimensional hyperspace.

2. **Locality**: In the U-order, each cell visited by the curve is directly connected to two face-neighboring cells which remain face-neighbors in the one dimensional hyperspace spanned by the curve. Locality in N-ordered domains is almost as good[16].

3. **Compactness**: Encoding and decoding the Hilbert or Morton order requires only local information. Following the integer indexing for Cartesian meshes outlined in ref. [2], a cell's 1-D index in ***U*** may be constructed using only that cell's integer coordinates in $\Re^d$ and the maximum number of refinements that exist in the mesh. This aspect is in marked contrast to other partitioing schemes based on recursive spectral bisection or other multilevel decomposition approaches which require the entire connectivity matrix of the mesh in order to perform the partitioning.

To illustrate the property of *compactness*, consider the position of a cell i in the N-order. One way to construct this mapping would be from a global operation such as a recursive lexicographic ordering of all cells in the domain. Such a construction would not satisfy the property of *compactness.* Instead, the position of i in the N-order may be deduced solely by inspection of cell i's integer coordinates (x_i, y_i, z_i).

Assume $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)$ is the bitwise representation of the integer coordinates (x_i, y_i, z_i) using m-bit integers. The bit sequence $\{\tilde{x}_i^1\tilde{y}_i^1\tilde{z}_i^1\}$ denotes a 3-bit integer constructed by interleaving the first bit of x_i, y_i and z_i. One can then immediately compute cell i's position in ***U*** as the $3m$-bit integer $\{\tilde{x}_i^1\tilde{y}_i^1\tilde{z}_i^1\tilde{x}_i^2\tilde{y}_i^2\tilde{z}_i^2\ldots\tilde{x}_i^m\tilde{y}_i^m\tilde{z}_i^m\}$. Thus, simply by inspection of a cell's integer coordinates, we are able to directly calculate its position in the one-dimensional space ***U*** without any additional information. Similarly compact construction rules exist for the U-order[21].

Figure 3 illustrates these mapping and locality properties for an adapted two-dimensional Cartesian mesh, partitioned into three subdomains. The figure demonstrates the fact that for adapted Cartesian meshes, the hyperspace ***U*** may not be fully populated by cells in the mesh. However, since cell indices in ***U*** may be explicitly formed, this poses no shortcoming.

The quality of the partitioing resulting from U-ordered meshes have been examined in Ref.[19]. and were found to be competitive with respect to other popular partitioners. Weights can be assigned on a cell-by-cell basis. One advantage of using this partitioning strategy stems from the observation that mesh refinement or

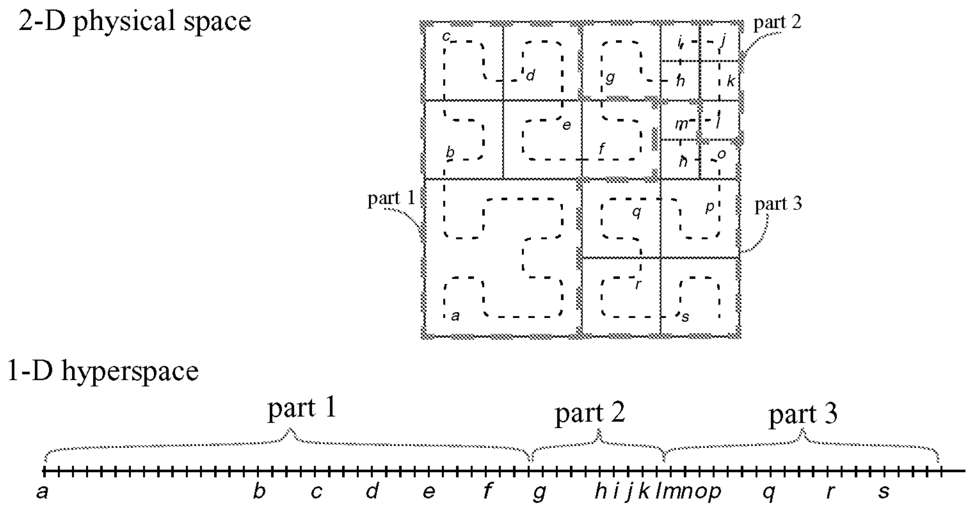


Figure 3: An adapted Cartesian mesh and associated space-filling curve based on the U-ordering of $\Re^2 \to U$ with the U-ordering illustrating locality and mesh partitioing in two spatial dimensions. Partitions are indicated by the heavy dashed lines in the sketch

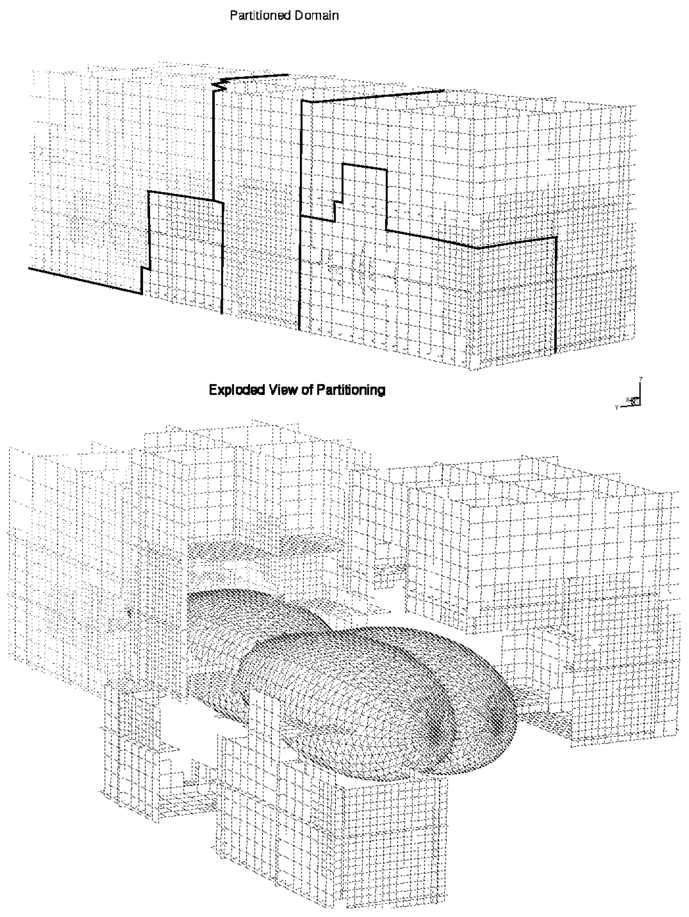


Figure 4: Partitioning of 6 level adapted mesh around a triple teardrop geometry with 240000 cells into 4 subdomains using space-filling curves. The mesh is shown by a collection of cutting planes through each partition.

coarsening simply increases or decreases the population of $\boldsymbol{U}$ while leaving the relative order of elements away from the adaptation unchanged. Re-mapping the new mesh into new subdomains therefore only moves data at partition boundaries and avoids global remappings when cells adaptively refine during mesh adaptation. Recent experience with a variety of global repartitioners suggest that the communication required to conduct this remapping can be an order of magnitude more expensive than the repartitioning itself[22]. Additionally, since the partitioning is basically just a re-ordering of the mesh cells into the U-order, the entire mesh may be stored as a single domain, which may then be partitioned into any number of subdomains on-the-fly as it is read into the flow solver from mass storage. This approach permits the mesh to be stored as a single unpartitioned file. In a heterogeneous computing environment where the number of available processors may not be known at the time of job submission, the value of such flexibility is self-evident.

Figure 4 shows an example of a three dimensional Cartesian mesh around a triple teardrop configuration partitioned using the U-order. The mesh in this figure contains 240000 cells and is indicated by several cutting planes which have been passed through the mesh, with cells colored by partition number. The upper frame shows the mesh and partition boundaries, while the lower frame offers further detail through an exploded view of the same mesh. In determining partition boundaries in this example, cut-cells were weighted 10x as compared to un-cut Cartesian hexahedra.

SUBDOMAIN CONNECTIVITY MATRIX LOCALITY

Figure 5 illustrates the cell adjacency matrix within a typical subdomain after applying the U-ordering. Each cell face in the subdomain results in a point on this graph whose coordinates are the indices of the neighboring cells. As indicated on the figure, this matrix is block structured, and the regions stemming from the various cell types in the subdomains are labeled. Computation of the residual occurs in the two diagonal blocks labeled "volume cells" and "cut-cells", while the overlap regions are updated via data-exchange with neighboring subdomains. Examination of the structure of these diagonal blocks indicates high total bandwidth requirements. The face list within each subdomain is sorted by the lowest cell index which the face connects to, and thus a loop over the face lists of either the volume or cut-cells accesses data in these two blocks row-by-row, from the top down. However, since the cells are clustered into "arms" off the main diagonal, memory pages loaded to access one cell will be subsequently hit many times as other nearby cells are requested by subsequent faces. Despite this, further bandwidth reduction and diagonal dominance may still be advantageous on some cache-based computing architectures or for use with some matrix inverters or preconditioners. The standard technique for alleviating this shortcoming is further reordering within each of these blocks[23].

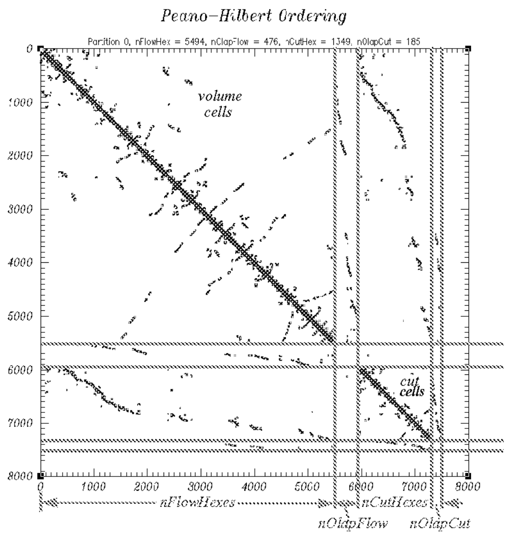


Figure 5: Connectivity matrix of a typical subdomain after partitioning with the U-ordering. Various cell types within the subdomain are labeled.

Provision is included for applying a RCM reordering[23] to the diagonal blocks in this matrix which produces a matrix with substantially reduced bandwidth. For cache-based machines, further re-ordering is also possible by coloring the data on any given cache-line so that data-dependencies are avoided when loading the data pipes on pipelining architectures, or for constructing short vectors on processors which support short vector processing.

AUTOMATIC CONSTRUCTION OF COARSE GRIDS

A central issue in the implementation of multigrid smoothers on unstructured meshes is the construction of a series of coarse grids for the smoother to act upon. However, since adaptively refined Cartesian grids are based upon successive refinements of an initial coarse grid, there is a natural path for coarse grid construction. A variety of approaches have been suggested in the literature, however, the asymptotic coarsening ratio in some of these has been insufficient to ensure that the method will extract the full benefit of multigrid. Moreover, the approach in ref. [2] permits the cells to divide anisotropically and therefore, we revisit the issue of efficient coarse mesh generation.

In contrast to coarse grid generation problems on unstructured (general) hexahedral, tetrahedral, or mixed element meshes, coarse cells in Cartesian meshes can be designed to nest *exactly* (i.e. cells on the coarse mesh are the precise boolean addition of cells on the fine mesh). In addition, the cells can be organized such that any cell in the mesh may be located uniquely by a set of integer indices[2]. The combination of these two facts lead to an novel coarse mesh generation algorithm for adaptively refined Cartesian meshes. The asymptotic complexity of this algorithm is $O(N \log N)$, where N is the number of cells in the fine mesh. This result stems from the fact that the central operation is a standard quicksort routine[1], and all other operations may

1. This result could be improved upon through the use of a radix sort, or other sort which has a better time-bound, however, quicksort is fast enough in most cases.

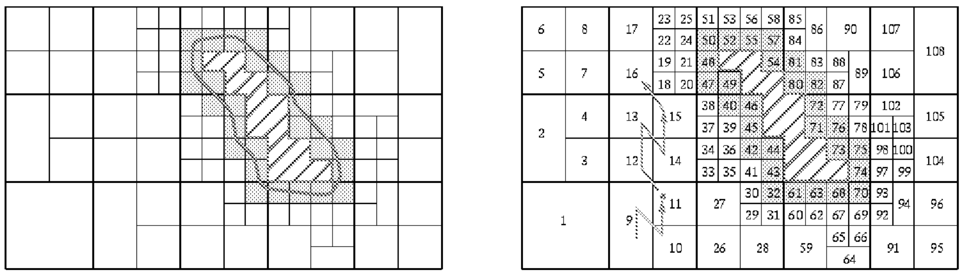


Figure 6: (Left) A two dimensional adaptively refined Cartesian mesh. Cut-cells are shown shaded. (Right) The same mesh, after reordering with a specially designed comparison operator in preparation for coarsening.

be performed in constant time.

Figure 6 displays a two dimensional, directionally refined Cartesian mesh which illustrates the coarse mesh generation strategy. The mesh shown (left) is the input or "fine mesh" which the algorithm coarsens. The boundary of a hypothetical body is indicated, and the crosshatching indicates where there are no cells in the mesh. Gray shaded cells denote cut-hexahedra. To the right of this figure lies a second view of the mesh after it has been sorted using a specially designed comparison operator. The cell indices in this mesh indicate the sorted order, which is further illustrated by a partial sketch of the path shown through cells 9–16.

The comparison operator basically performs a recursive lexicographical ordering of cells which can coarsen into the same coarse cell. Adaptively refined Cartesian meshes are formed by repeated subdivision of an initial coarse mesh (referred to as the *level 0 mesh*), therefore any cell, i, is traceable to an initial "parent cell" in the level 0 mesh. Similarly, if cell i has been refined R times, it will have parent cells at levels 0 through $(R-1)$. If a cell has never been divided, then it is referred to as a "level 0 cell" and is identical to its level 0 parent.

1. Cells on the level 0 mesh are sorted in lexicographic order using the integer coordinates of their level 0 parents as keys.
2. If a cell has been subdivided, recursively sort its children lexicographically.

This algorithm can be implemented with a single quicksort which uses a comparison operator which examines the integer indices of two input cells on a bit-by-bit basis (see ref. [12]). As noted above, its asymptotic complexity is proportional to that of the sorting method used.

After sorting the fine mesh, coarsening proceeds in a straightforward manner. Cells are processed by a single sweep through the sorted order. If a contiguous set of cells are found which coarsen to the same parent they are coalesced into that parent. Cells which do not meet this criteria are "not coarsenable" and are injected to the coarse mesh without modification.

Figure 7 (left) shows the coarse mesh resulting from one application of the coarsening algorithm, note that fine grid cells on the level-0 mesh are *fully coarsened* and do not coarsen beyond their initial size. The right

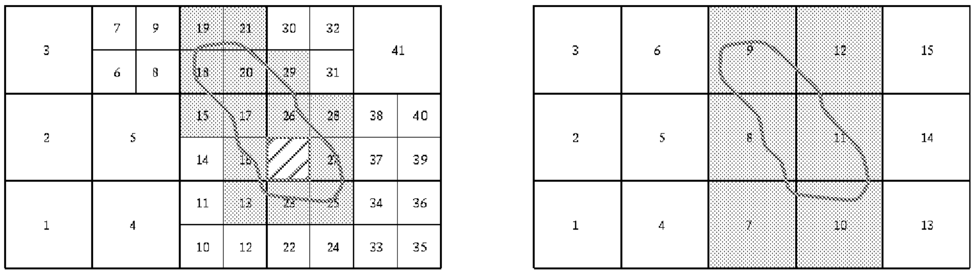


Figure 7: Left: Adapted Cartesian mesh from Figure 6 after one coarsening. Outline of geometry is indicated, and cut-cells are shown in grey. Right: Same mesh after one additional application of the coarsening algorithm

frame in this figure shows the mesh resulting from a second application of the coarsening operator.

Note that with this algorithm, cells won't coarsen in two situations: (1) if they are fully coarsened; or if coarsening is suspended because one (or more) of the children of a given parent is subdivided. Application of this algorithm to a variety of adapted Cartesian meshes on actual geometry (including a 3D wing, a Single Stage to Orbit configuration and a subsonic business jet) revealed that it consistently produces coarse meshes with coarsening ratios greater than 7:1. Finally, note that the coarse cells in fig. 7 are automatically constructed in the sorted order so that further coarsening does not require additional sorting.

One subtlety that the coarsening algorithm must contend with is indicated in Figure 8. The presence of split-cells in the domain implies that, under some coarsening situations, cut-cells on the fine mesh may coarsen into split-cells on the coarse grid. Alternatively, when fine grid split-cells coarsen into the same parent as uncut volume cells, cut- or split-cells may result. This apparent complication stems directly from the fact that the index space of the control volumes is not the same as that of the Cartesian hexahedra from which these control volumes were formed, and in three dimensions, a wide variety of such cases exist. To ensure accurate construction of the coarse mesh, our algorithm insists that two cut/split-cells with the same parent must have at least one common face to coalesce into the same control volume on the coarse grid.

PRELIMINARY RESULTS

The preliminary results presented in this section intend to investigate the global order of accuracy of the flow solver, as well as the parallel scalability of the method using the SFC mesh partitioners. An investigation of the effectiveness of the multigrid scheme will not be presented as such results are still premature. All computations were performed on 1-32 processors of an SGI Origin 2000 equipped with MIPS R10000 processors running at 250Mhz. subdomain boundary information exchange is performed using shared a shared memory programming paradigm, and care was taken to ensure that the memory required to store each subdomain is physically located on boards local to the processor which integrate each subdomain.

VERIFICATION AND GLOBAL ORDER OF ACCURACY

Before examining issues of modeling and parallel scalability, it is necessary to first verify that our implementation correctly solves the Euler equations, and to document the order of accuracy of the solver on a actual meshes. This investigation relies upon a closed-form, analytic solution to the Euler equations for a supersonic vortex model problem[24]. The presence of an exact solution permits the investigation to examine the truncation error of the discrete solution using a series of telescopic meshes. Since this is a shock-free flow, the measured order of accuracy is not corrupted by limiter action near shocks, and the behavior is indicative of the scheme's performance in smooth regions of a flow. Although this example is only two dimensional, the full three dimensional solver was run using an 3-D geometry made by extrusion.

To investigate the truncation error of the scheme, the domain was initialized to the exact solution and integrated one time step. The residual in each cell then offers a direct measure of the difference between the dis-

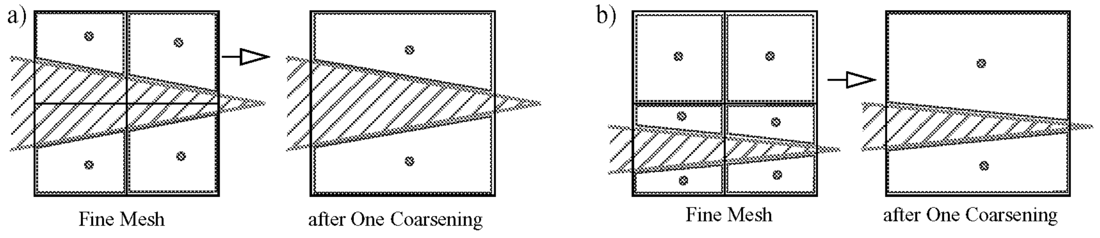
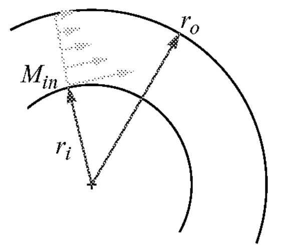


Figure 8: Mesh coarsening examples in which the index space of the control volumes differs from that of the Cartesian hexahedra from which these control volumes are formed. (a) Four cut-cells become 2 split-cells when the mesh is coarsened, (b) 2 volume cells, and 4 split cells become 2 split-cells after coarsening.

exact solution:

$$\rho(r) = \rho_i\left\{1 + \frac{\gamma+1}{2}M_i^2\left[1 - \left(\frac{r_i}{r}\right)^2\right]\right\}^{\frac{1}{\gamma-1}}$$



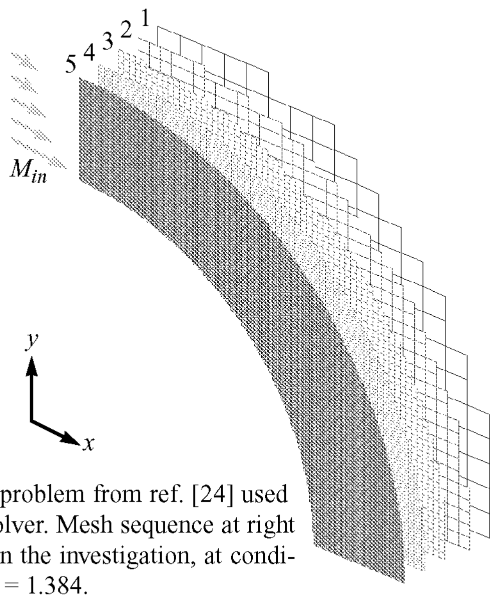


Figure 9: Overview of supersonic vortex model problem from ref. [24] used to investigate the order of accuracy of the solver. Mesh sequence at right shows series of 5 telescoping meshes used in the investigation, at conditions: $M_{in} = 2.25$, $p_{in} = 1/\gamma$, $\rho_{in} = 1$, $r_i = 1$, $r_o = 1.384$.

crete scheme and the governing equations, including the effects of boundary conditions.

Figure 9 presents an overview of the investigation. The sketch at the left shows the inviscid flow between two concentric circular arcs, while the frame at the right shows the sequence of 5 Cartesian meshes used in the investigation. The meshes were created by nested subdivision and while the coarsest of these grids had 105 cells in a 2D slice, the finest had over 21000 cells at the same station.

Figure 10 contains a plot of the L2 norm of density error resulting from this analysis. The error plot is remarkably linear over the first 4 meshes, but shows signs of a slight tailing-off on the final mesh. Over the first 4 meshes, the average order of accuracy is 1.88. If the finest mesh is included, this estimate drops to a value of 1.82. Both of these slopes are comparable to those in the investigation of reconstruction schemes on body-fitted unstructured meshes in ref. [24], and we note that the absolute magnitude of error in the present (Cartesian) scheme is more than a factor of two lower than was reported in that investigation. The slight tailing-off of the results for mesh 5 is believed to be a result of round-off error in computation of the error norm is not surprising considering the extremely low levels of error measured on this mesh. This hypothesis, however, is still under investigation.

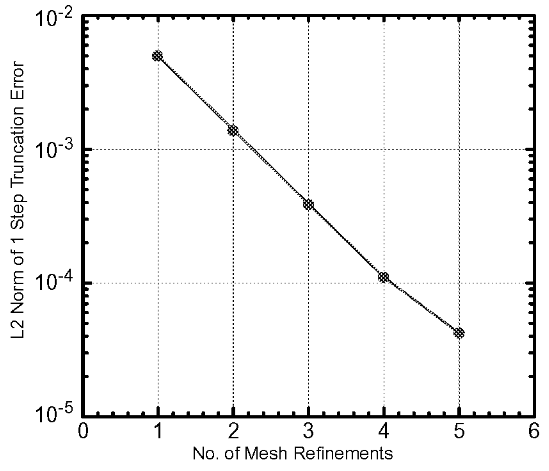


Figure 10: L2 norm of density truncation error for sequence of refined meshes shown in fig. 9.

The results shown in fig. 10 were generated using the Colella flux function, however, results with the van Leer option are essentially identical.

CONVERGENCE ON PARTITIONED DOMAINS

Adopting the domain decomposition with a single overlap cell permits a formulation which ensures that the residuals computed within each cell at every timestep with one partitioning match those for any other partitioning. Figure 11 illustrates this property by documenting convergence of the maximum residual of density for the supersonic vortex problem using a 250,000 cell mesh partitioned into 1, 2, 4, and 8 subdomains. All histories in this figure collapse to the same line to within machine precision. As the legend indicates, this test was performed using both the machine's default arithmetic (SGI Origin 2000, *cc* option *-Ofast)* and IEEE-754 compliant arithmetic.

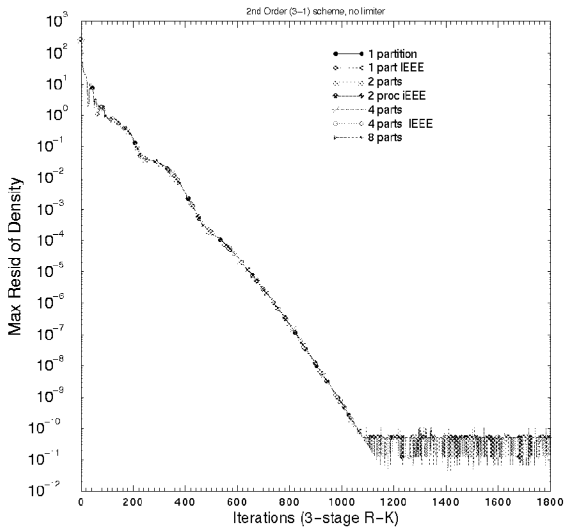


Figure 11: Comparison of convergence history using 1, 2, 4, and 8 subdomains using both default (SGI Origin 2000, *cc* option *-Ofast)* and IEEE-754 compliant arithmetic.

ONERA M6 WING

With the preliminary verification complete, focus shifts to a three dimensional example of an ONERA M6 wing which has been widely cited in the literature. This transonic $M_\infty = 0.84$, $\alpha = 3.06^\circ$ case is often used in the validation of inviscid solution techniques. This test was performed at a relatively high Reynolds number (based on root chord) of 12 x 10^{6}[25], which minimizes effects of the displacement thickness making accurate comparisons of sectional pressure distributions possible. Other viscous effects in the experimental data are limited to a slight separation filling in the C_p distribution behind the lambda shock on the lee surface.

Simulation of this test was conducted using the geometry of a wing in free air, with the far-field boundary located 30 chords from the wing. The final mesh contained 525000 control volumes, with 25000 cut-cells and 528 split-cells. The mesh was partitioned into 8 subdomains using the Peano-Hilbert ordering described

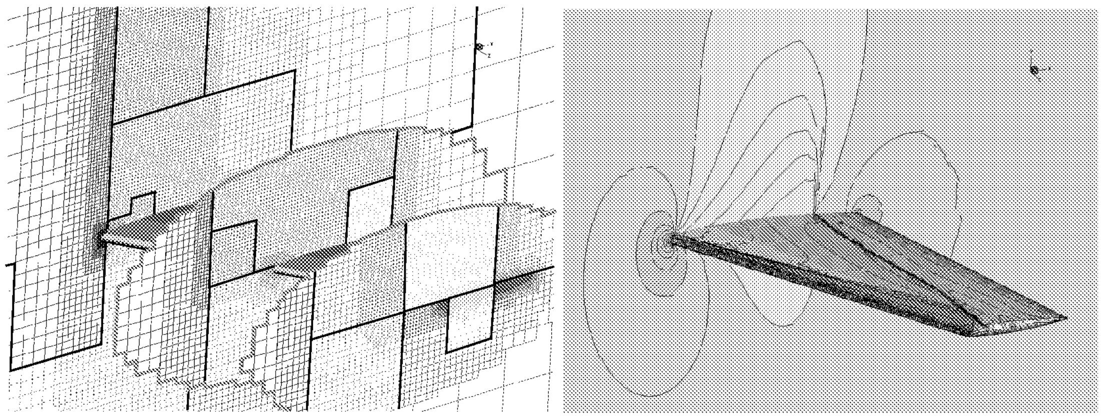


Figure 12: Partitioned mesh and C_p contours for the ONERA M6 wing example. The mesh contains 525000 cells at 9 levels of refinement, mesh partitions are shown by color-coding and outlined in heavy lines. C_p contours are plotted using a cell-by-cell reconstruction of the discrete solution. $M_\infty = 0.84$, $\alpha = 3.06$, van Leer flux.

in the preceding section. Figure 12 displays this mesh by three cutting planes. Cells on each cut plane are color coded by subdomain. Along side the mesh, fig. 12 presents C_p contours on the wing surface, and symmetry plane resulting from a simulation using the van Leer flux option. This image clearly displays the well-known lambda shock structure on the upper surface of the wing. Contours in this image were constructed cell-by-cell, using the computed gradients within each cell. This method of plotting gives a more accurate picture of the discrete solution, since fluxes are formed with this same reconstruction. The slight breaks in the contour lines in some high gradient regions are a by-product of this cell-by-cell plotting. These solution shown was converged 6 orders of magnitude (L1 norm of density) using the van Leer flux option.

Figure 13 provides a quantitative assessment of the solution quality through pressure profiles at six spanwise stations. This figure displays C_p *vs. x/c* at spanwise stations at 20, 44, 65, 80, 90, and 95% span. The inboard stations correctly display the double-shock on the upper surface, while stations at 90 and 95% confirm accurate prediction of the merging of these shocks. The experimental data at stations 20 and 44% indicate that the rear shock is followed by a mild separation bubble triggered by the shock-boundary layer interaction. As is typical in such cases, the inviscid discrete solution locates this rear shock slightly behind its experimental counterpart.

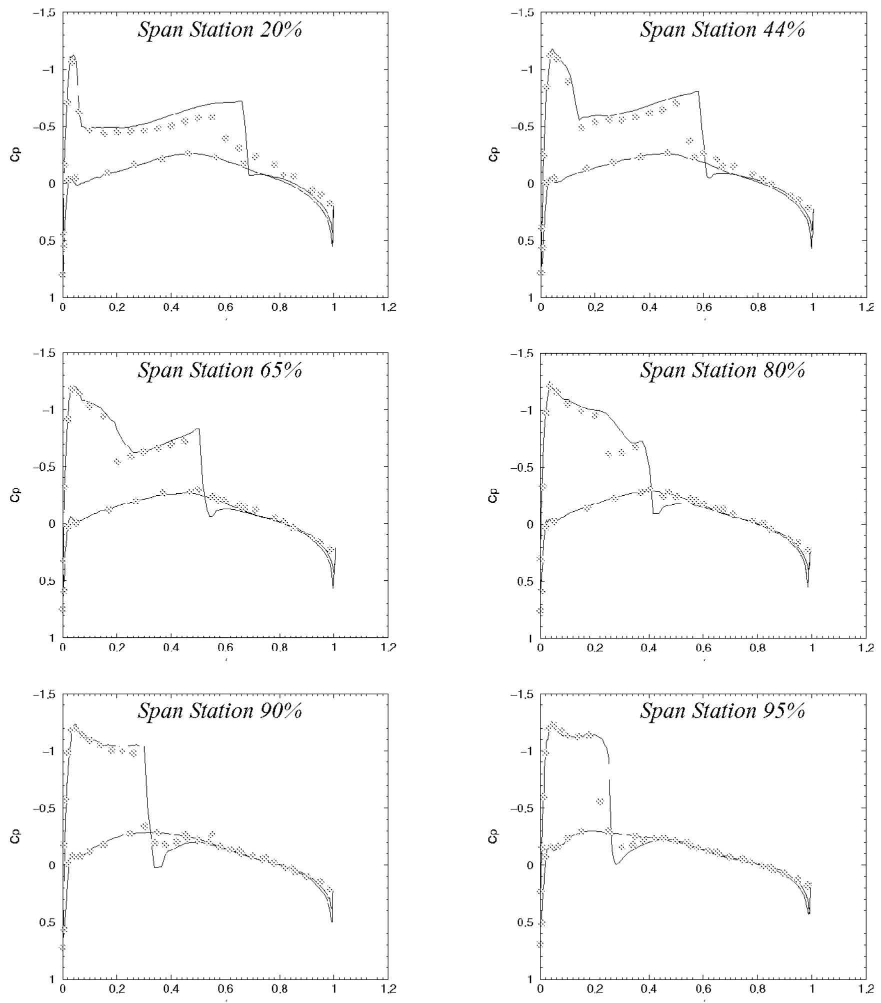


Figure 13: C_p vs. *x/c* for ONERA M6 wing example at six spanwise locations. $M_\infty = 0.84$, $\alpha = 3.06^\circ$. Experimental data from ref. [25] shown as symbols, inviscid discrete solution shown with solid line.

PARALLEL SCALABILITY AND PERFORMANCE

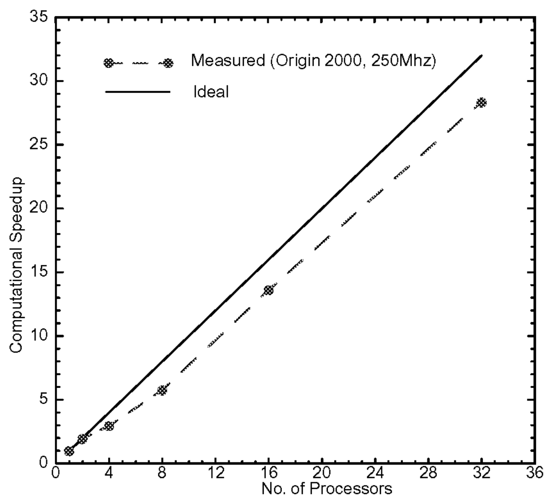


Figure 14: Preliminary investigation of parallel scalability of single mesh (no-multigrid) case. Data reflect average results from 3 runs with each partitioning.

Figure 14 contains preliminary results from scalability testing. Tests were conducted on from 1 to 32 processors on an Mips R10000 based SGI Origin 2000. The mesh in this test contained 525000 cells. Each processor of this machine has a 4Mb Level 2 cache, and two processors on each board share the same local memory. Examination of this plot shows generally good scalability, however, communication does appear to slow this particular computation on 4 and 8 processors when the problem initially gets spread over several boards within the machine. On 32 processors the timings show a "cache bubble" evidenced by the fact that the results on 32 processors are more than a factor of two faster than the timings on 16 processors. Table 1 shows the per-processor execution rate and parallel speed-up for this example. Results in this table clearly show a 4% increase in per-processor execution rate on 32 processors as each processor's L2 cache was very nearly sufficient to store the entire subdomain. The table demonstrates no substantial decrease in performance with larger numbers of subdomains, and the communication/computation ratio afforded by the partitioning does not appear to be uncompetative. Results in Table 1 and in Figure 14 were obtained by averaging the results of 3 separate sets of tests since timings on this machine are known to vary by as much as 10%.

Table 1: Parallel scalability and processing rate per processor. Results for each partitioning reflect average of three runs. 525000 control volumes, 200 iterations per test.

No. of Processors	CPU time/CPU (sec.)	Parallel Speed-up	Mflops/CPU[a]	Ideal Speedup
1	2559	1	81.4	1
2	1315	1.94	81.9	2
4	865	2.96	72.77	4
8	383	5.76	77.57	8
16	188	13.61	78	16
32	90	28.37	82	32

a. Mflops counted using R10000 hardware counters on optimized code, with single cycle MADD instruction disabled. Floating-point multiply, add, and divide each counted as one flop.

CONCLUSIONS AND CURRENT WORK

This paper presented preliminary verification and validation of a new, parallel, upwind solver for Cartesian meshes. Comparison of the scheme's one-step truncation error with an analytic solution demonstrated an achieved order of accuracy between 1.82 and 1.88. Preliminary validation by direct comparison to experimental results on a three dimensional wing configuration was also performed, demonstrating that the discrete solutions were competitive with other solution procedures. Preliminary documentation of a new on-the-fly SFC based partitioning strategy was also presented. This strategy enables reordered meshes to be presorted and stored as a single domain. This mesh can then be partitioned into any number of partitions at run time. Investigations demonstrated that this decomposition strategy produces a parallel speed-up in excess of 28 on 32 processors with no net decrease in processing rate. Details of a new coarse mesh generation algorithm for multilevel smoothers on Cartesian meshes were also presented. This algorithm generally achieves mesh coarsening ratios in excess of 7 on adaptively refined meshes.

Development of this method continues, and examples on complex configurations at elevated Mach numbers are planned in the immediate future. An investigation of multigrid efficiency for flows with complex geometry at a variety of Mach numbers is on-going.

ACKNOWLEDGMENTS

The authors would like to extend thanks to R. Löhner for pointing to literature on the use of space-filling curves as potentially useful reorderers and partitioners. M. Berger and G. Adomavicius were supported in part by AFOSR Grant F49620-97-1-0322 and DOE Grant DEFG02-92ER25139. Some of this work was performed while M. Berger was at RIACS and this support is gratefully acknowledged.

REFERENCES

[1] Melton J. E., Berger, M. J., and Aftosmis, M. J., "3D Applications of a Cartesian grid Euler method," *AIAA Paper 95-0853-CP*, Jul. 1993.

[2] Aftosmis, M.J., Berger, M.J., Melton, J.E.: "Robust and efficient Cartesian mesh generation for component-based geometry." *AIAA Paper 97-0196*, Jan. 1997.

[3] Charlton, E. F., and Powell, K. G., "An octree solution to conservation-laws over arbitrary regions." *AIAA Paper 97-0198*, Jan. 1997.

[4] Wang Z.J., "An automated viscous adaptive Cartesian grid generation method for complex geometries." in *Proceedings of the 6th International Conf. on Numerical Grid Generation in Computational Field Simulations,* Eds. Cross, M. *et al.,* Univ. Greenwich, UK., 1998.

[5] Day, M. S., Colella, P., Lijewski, M. J., Rendleman, C. A., and Marcus, D. L., "Embedded boundary algorithms for solving the Poisson equation on complex domains," Lawrence Berkeley National Laboratory, *LBNL-41811*, May, 1998.

[6] Karman, S. L., "SPLITFLOW: A 3D unstructured Cartesian/prismatic grid CFD code for complex geometries." *AIAA Paper 95-0343*, Jan. 1995.

[7] Forrer, H., "Second order accurate boundary treatment for Cartesian grid methods." Seminar for Angewandte Mathmatic, ETH Zürich, ETH Research Report 96-13, 1996.

[8] Barth, T.J., and Jespersen, D.C., "The design and application of upwind schemes on unstructured meshes." *AIAA Paper 89-0366*, Jan. 1989.

[9] Colella P, Ferguson, R., and Glaz, H., "Multifluid algorithms for Eulerian finite difference methods". Preprint 1996.

[10] Hackbusch, W., and Trottenberg, U., (eds.), *Lecture Notes in Mathematics: Multigrid Methods*, Springer-Verlag Berlin, Heidelberg, ISBN 0-387-11955-8, 1982.

[11] Riemslagh, K., and Dick, E., "A multigrid method for steady Euler equations on unstructured adaptive grids." *Proceedings of the 6th Copper Mountain Conf. on Multigrid Methods*, NASA Conference publication 3224, pp. 527-542. 1993.

[12] Aftosmis, M. J., Berger, M J., and Adomavicius, G. "A domain-decomposed multi-level method for adaptively refined Cartesian grids with embedded boundaries." *AIAA Paper 2000-0808,* Jan. 2000.

[13] Allmaras, S. R., "Analysis of semi-implicit preconditioners for multigrid solution of the 2-D compressible Navier-Stokes Equations"., *AIAA Paper 95-1651-CP*, Jun., 1995.

[14] Karypis, G., and Kumar, V., "METIS: A software package for partitioned unstructured graphs, partitioing meshes, and computing fill-reducing orderings of sparse matrices." University of Minn. Dept. of Comp. Sci., Minneapolis, MN., Nov. 1997

[15] Schloegel, K., Karypis, G., and Kumar, V., "Parallel Multilevel Diffusion Schemes for Repartitioning of Adaptive Meshes." *Tech. Rep. #97-014,* University of Minn. Dept. of Comp. Sci., 1997.

[16] Samet, *The design and analysis of spatial data structures.* Addison-Wesley Series on Computer science and information processing, Addison-Wesley Publishing Co., 1990.

[17] Salmon, J.K., Warren, M.S., and Winckelmans, G.S., "Fast parallel tree codes for gravitational and fluid dynamical N-body problems." *Internat. Jol. for Supercomp. Applic.* **8**:(2), 1994.

[18] Griebel, M., Tilman, N., and Regler, H., "Algebraic multigrid methods for the solution of the Navier-Stokes equations in complicated geometries." Int. J. Numer. Methods for Heat and Fluid Flow 26, pp. 281-301, 1998, also as SFB report 342/1/96A, Institut für Informatik, TU München, 1996.

[19] Pilkington, J.R., and Baden, S.B.,"Dynamic partitioning of non-uniform structured workloads with spacefilling curves." Jan 1995.

[20] Schrack, G., and Lu, X., "The spatial U-order and some of its mathematical characteristics." *Proceedings of the IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing.* Victoia B.C, Canada, May, 1995.

[21] Liu, X., and Schrack, G., "Encoding and decoding the Hilbert order." *Software-Practice and Experience,* **26**(12), pp. 1335-1346, Dec. 1996.

[22] Biswas, R., Oliker, L., "Experiments with repartitioning and load balancing adaptive meshes." NAS Technical Report NAS-97-021, NASA Ames Research Ctr., Moffett Field CA., Oct. 1997.

[23] Löhner, R., "Renumbering strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines." *AIAA Paper 97-2045-CP*, Jun., 1997.

[24] Aftosmis, M. J., Gaitonde, D., and Tavares, T. S., "Behavior of linear reconstruction techniques on unstructured meshes." *AIAA J.,* **33**(11), pp. 2038-2049, Nov. 1995.

[25] Schmitt, V., and Charpin, F., "Pressure distributions on the ONERA-M6-Wing at transonic Mach numbers." *Experimental Data Base for Computer Program Assessment,* AGARD Advisory Report AR-138, 1979.