AIAA 2004-Abstract

# Applications of Space-Filling-Curves to Cartesian Methods for CFD

**M. J. Aftosmis**
Mail Stop T27B
NASA Ames Research Center
Moffett Field, CA 94404

**M. J. Berger**
Courant Institute
251 Mercer St.
New York, NY 10012

**S. M. Murman**
ELORET
NASA Ames Research Center
Moffett Field, CA 94404

**42nd Aerospace Sciences Meeting and Exhibit**
10-13 January 2004 / Reno NV

# A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries

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

**M. J. Berger**[‡]
Courant Institute
251 Mercer St.
New York, NY 10012
*berger@cims.nyu.edu*

**S. M. Murman**
ELORET
Moffett Field, CA 94404
*smurman@nas.nasa.gov*

**The proposed paper presents a variety novel uses of Space-Filling-Curves (SFCs) for Cartesian mesh methods in CFD. While these techniques will be demonstrated using non-body-fitted Cartesian meshes, most are applicable on general body-fitted meshes – both structured and unstructured. We demonstrate the use of single $\mathcal{O}(N \log N)$ SFC-based reordering to produce single-pass ($\mathcal{O}(N)$) algorithms for mesh partitioning, multigrid coarsening, and inter-mesh interpolation. The intermesh interpolation operator has many practical applications including "warm starts" on modified geometry, or as an inter-grid transfer operator on remeshed regions in moving-body simulations. Exploiting the compact construction of these operators, we further show that these algorithms are highly amenable to parallelization. Examples using the SFC-based mesh partitioner show nearly linear speedup to 512 CPUs even when using multigrid as a smoother. Partition statistics are presented showing that the SFC partitions are, on-average, within 10% of ideal even with only around 50,000 cells in each subdomain. The inter-mesh interpolation operator also has linear asymptotic complexity and can be used to map a solution with $N$ unknowns to another mesh with $M$ unknowns with $\mathcal{O}(\max(M,N))$ operations. This capability is demonstrated both on moving-body simulations and in mapping solutions to perturbed meshes for finite-difference-based gradient design methods.**

## 1 Introduction

THE literature on numerical solution methods for PDE's contains a wealth of diverse approaches for improving cache performance, performing domain decomposition, mesh coarsening and inter-mesh solution transfer. These topics are frequently considered separately and usually require different data structures and infrastructure to perform each task. For example, a high-performance distributed memory unstructured mesh code may use RCM ordering for improving cache-performance, recursive spectral bisection, or multilevel nested dissection for domain decomposition[1][2], a graph-based mesh coarser[4][3], and a variety of fast spatial search data structures for inter-mesh interpolation, or solution transfer to re-gridded regions of a subdomain.

For each task, these techniques offer superb performance, and many of them are amenable to some degree of parallelization. Nevertheless, development and maintenance of such a diverse collection of tools requires considerable investment, and when a substantial overhaul is necessary (*e.g.* moving a code to distributed-memory parallelization) the level of effort required will be significant.

The proposed paper examines the use of space-filling-curves as a unifying data structure for all of these ends. Construction of these curves is extremely inexpensive as the SFC index for any voxel in space may be computed using only local information, and thus computation of indices may be performed in parallel. The asymptotic time complexity of constructing the curve is therefore bounded by the sort algorithm which orders the mesh along the curve. This sort can be performed with standard sorting routines such as a quicksort which produces a method with $\mathcal{O}(N \log N)$ running time.

## 2 Space-Filling-Curves

The central operation in using space-filling curves (SFC) for our purposes is a reordering of the mesh using either the Morton or Peano-Hilbert order[22]. Both orderings have been explored in scientific computing in a variety of roles, including the parallel solution of $N$-body problems in computational physics[23], algebraic multigrid[24], mesh generation,[25] in the solution of PDEs and dynamic repartitioning of adaptive methods[26]. Figure 1 shows both Peano-Hilbert and Morton SFCs 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. Similar properties exist for the Morton ordering which uses an "N" shaped curve as its basic building block.

In three spatial dimensions the curves follow the same basic construction rules, but the basic building block extends into the third dimension with additional U or N-shaped turns. Figure 2 illustrates this 3-D construction for the U-order showing: (a) the basic 2x2x2 building block (b) the same mesh after one uniform refinement, each segment of the curve is replaced with an appropriately rotated basic building block, (c) the curve after additional adaptive refinement of cells near the back-south-west corner. Properties and $d$-dimensional

[†] Research Scientist, Senior Member AIAA
[‡] Professor, Member AIAA
[¥] Member AIAA.

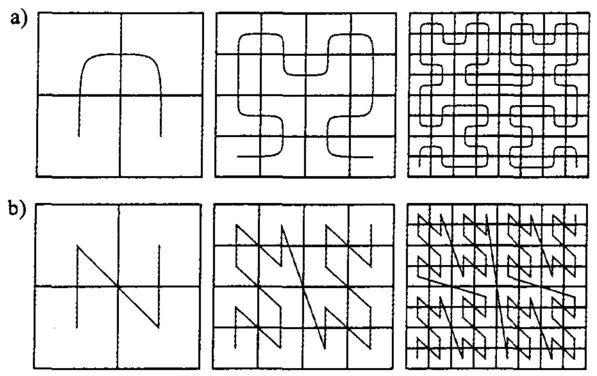
*Figure 1:* Space-filling curves used to order three Cartesian meshes in two spatial dimensions: a) Peano-Hilbert or "U-ordering", b) Morton or "N-ordering".

construction rules for these space-filling curves are discussed extensively in refs. [27] and [28].

Both orderings have locality properties which make them attractive as mesh partitioners[26][25]. For the present, we note only that such orderings have 3 important properties.

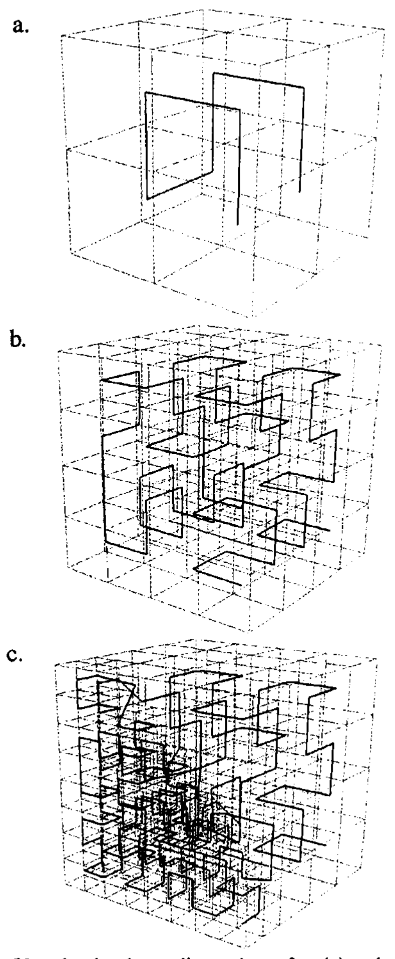
*Figure 2:* U-order in three dimensions for (a) a basic 2x2x2 block of cells, (b) the same block after uniform subdivision (c) cell order after refinement near the south-west-back cor-

1. **Mapping** $M : \mathcal{R}^d \rightarrow \boldsymbol{U}$: The U and N orderings each provide unique mappings from the $d$-dimensional physical space of the problem domain $\mathcal{R}^d$ to a one-dimensional hyperspace, ***U***, which one traverses following the curve. In the U-order, two of a cell's face neighbors in physical space will remain adjacent to the cell in the 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[22].
3. **Compactness**: Encoding and decoding the Hilbert or Morton order requires only local information. Following the integer indexing for Cartesian meshes outlined in ref. [5], a cell's 1-D index in ***U*** may be constructed using only that cell's integer coordinates in $\mathcal{R}^d$ and the maximum number of refinements that exist in the mesh. This aspect is in marked contrast to other partitioning 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 index 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 $(x_i, y_i, z_i)$ is the bitwise representation of the integer coordinates $(x_i, y_i, z_i)$ using $m$-bit integers. The bit sequence $\{x_i^1 y_i^1 z_i^1\}$ 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 location in ***U*** as the $3m$-bit integer $\{x_i^1 y_i^1 z_i^1 x_i^2 y_i^2 z_i^2 \ldots x_i^m y_i^m z_i^m\}$. Thus, simply by inspection of a cell's integer coordinates, we are able to directly calculate its location, $M(i)$, in the one-dimensional space ***U*** without any

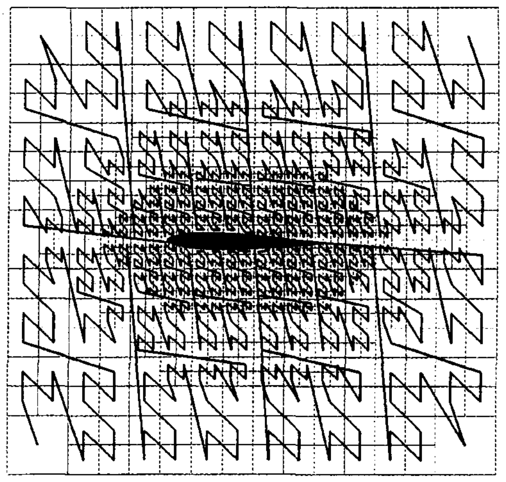
*Figure 3:* Morton order of an adaptively refined Cartesian mesh around a 2-D airfoil.

additional information. Similarly compact construction rules exist for the U-order[28].

In an *h*-refined Cartesian mesh, the finest cell can be used to define the dimensions of a single voxel. All coarser cells may then be reinterpreted as collections of this basic building block. This interpretation leads to an integer indexing scheme which can be used to address all cells in the computational domain. With this integer indexing scheme, the construction rules in the previous paragraph can then be applied to generate the Morton index, $M(i)$, of all the cells in the mesh. Figure 3 shows an example of the N-ordering on an adaptively refined mesh around a 2-D airfoil.

Construction of the Peano-Hilbert index, $H(i)$, follows a similarly compact procedure. After computing the SFC indices $M(i)$ or $H(i)$ for all the cells in the mesh, one simply takes these indices as sort keys and applies any one of the standard sorting algorithms (we use the quickersort algorithm from the C standard library). Since all the other data required for construction is local, the sort establishes the asymptotic bound for runtime of the reordering and choosing quick/quicker sort gives runtimes of $\mathcal{O}(N \log N)$.

## 3 Mesh Coarsening

The recursive nature of the SFC ordering makes it a natural choice for a mesh coarsener for *h*-refined meshes. In the same way that higher-order SFCs are generated by replacing segments with the basic U or N building block, the children produced by *h*-refinement replace their parent cell in the mesh. Since these children are sorted according to the local SFC they will all be nearest-neighbors on a contiguous segment of the SFC in the space of the curve $U$.

Figure 4 illustrates this observation in two-dimensions. Figure 4a shows the baseline multilevel mesh as it comes out of either the mesh generator or adaptation module. Frames (b) and (c) show the same mesh after two successive coarsenings. Cells in these meshes are ordered using the N-ordering, and their indices in the SFC order are shown. In fig. 4a cells 22, 23, 24, and 25 all lie adjacent to one another in the 1-dimensional hyperspace of the curve, $U$.

The coarsening algorithm proceeds with a single traversal of $U$ with a running index $i$ and testing if the cell at $U(i)$ is contained within the same parent as the cell at $U(i + 1)$. When a contiguous group of cells are found that all coarsen into the same parent cell they are agglomerated into that parent. If any of the siblings in a contiguous set are further subdivided, then all siblings of the set are "not coarsenable" and get injected into the coarse mesh without modification.

The first coarsening of the mesh in fig. 4a produces the mesh in fig. 4b. Cells 22-25 are coarsened in this first pass to produce cell 10 on the mesh in frame (b). Note that while cells 20, 21 and 26 are all children of the same parent, they are not coarsenable since 22-25 are one-level finer. The next coarsening pass produces the mesh in frame (c). Since cells 8-9 on mesh (b) were all siblings, they now coarsen to produce cell 5 on the mesh shown in frame (c). Further details of this algorithm are available in Ref. [30] with details of the treatment of split-cells described in [31].

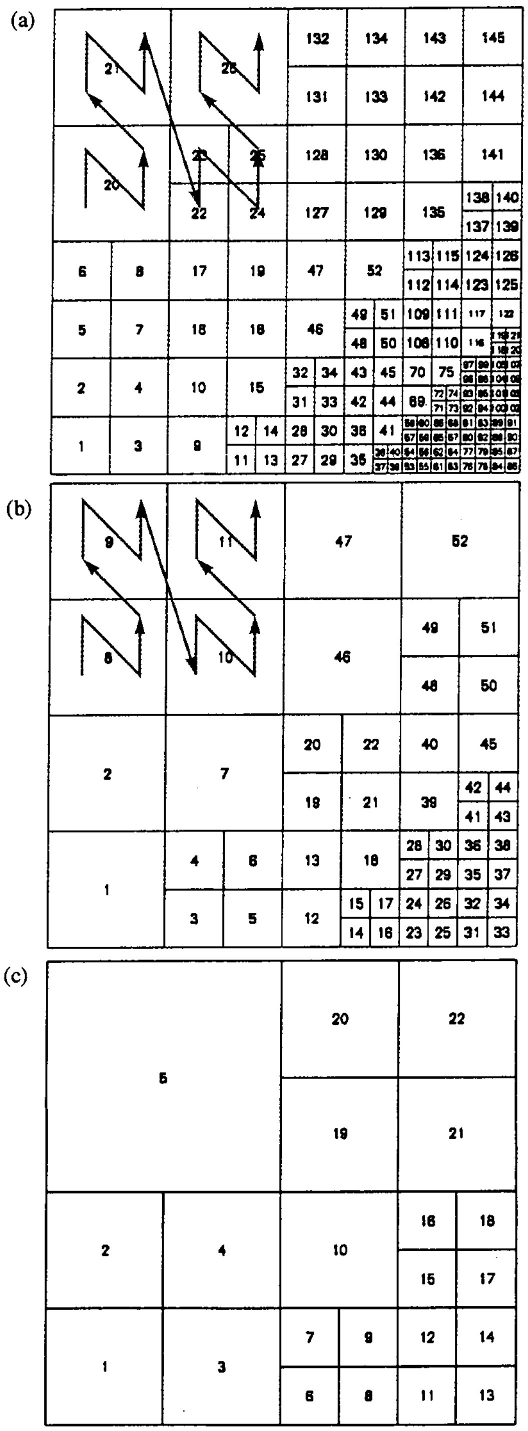
*Figure 4:* Mesh coarsening example using Morton ordering. The fine mesh in (a) is coarsened 2 times using the same space-filling-curve.

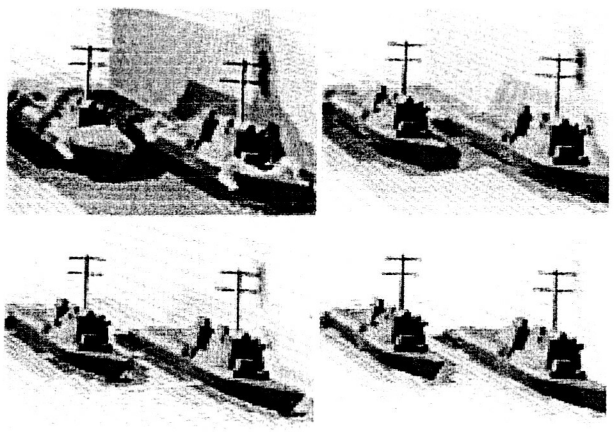
*Figure 5:* Mesh coarsening example in 3D using agglomeration along the SFC. The finest mesh contains 4.5 M cells while the coarsest contains 4500 cells after 4 coarsenings.

Figure 5 shows an example in three dimensions. In this example, a 4.5 M cell, adaptively refined mesh around two surface ships has undergone 4 cycles of coarsening by agglomeration along the SFC. The final mesh shown in the lower right frame of the figure contains 4500 cells. In practice, typical coarsening ratios for the algorithm are in excess of 7 on realistically complex problems.

## 4 Domain decomposition

The mapping and locality properties that are exploited for the single-pass mesh coarsener described above make SFCs a natural choice for partitioners on hierarchical meshes.[24][25][30][31] Figure 6 illustrates these mapping and locality properties for an adapted two-dimensional Cartesian mesh, partitioned into three subdomains. The figure points out 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.

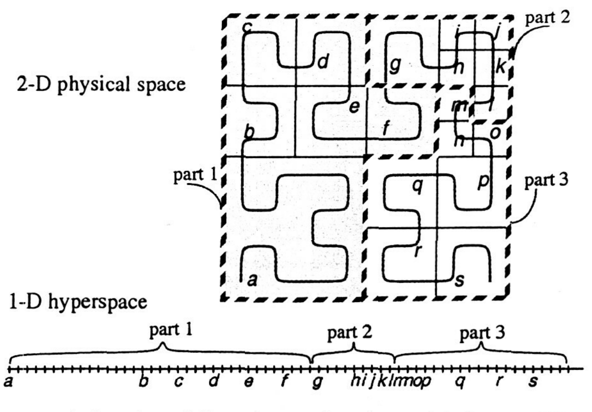

*Figure 6:* An adapted Cartesian mesh and associated space-filling curve based on the U-ordering of $M : \mathcal{R}^d \rightarrow U$ with the U-ordering illustrating locality and mesh partitioning in two spatial dimensions. Partitions are indicated by the heavy dashed lines in the sketch.

The quality of the partitioning resulting from U-ordered meshes have been examined in ref. [26] 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 coarsening simply increases or decreases the population of $U$ while leaving the relative order of elements away from the adaptation unchanged. Remapping 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[29]. 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. At run-time, this single mesh may then be partitioned into any number of subdomains on-the-fly as it is read into the flow solver from mass storage. In a heterogeneous shared 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.

The SFC reordering pays additional dividends in cache-performance. The locality property of the SFC produces a connectivity matrix which is tightly clustered regardless of the number of subdomains. Our numerical experiments suggest that SFC reordered meshes have about the same data cache hit-rate as those reordered using reverse Cuthill-McKee.

Figure 7 shows an example of a three dimensional Cartesian mesh around the full Space Shuttle Launch Vehicle (SSLV) configuration. This complex configuration includes the orbiter, external tank, solid rocket boosters, and fore and aft attach hardware. The computational mesh contains about 4.4 M cells at 14 levels of refinement, and is indicated by a single cutting plane passed through the mesh just behind the SSLV geometry. The mesh is displayed partitioned into 16 subdomains using the U-order. Reordering this mesh with the algorithm of §2 required 40 sec. on a 2 Ghz Pentium 4, and preparing 4 levels of coarser meshes for multigrid required 30 sec. on the same machine. In determining partition boundaries in this particular example, cut-cells were weighted 2.1x as compared to un-cut Cartesian hexahedra.

The partitioning in fig. 7 demonstrates how, even on adaptively refined meshes, the partitioning tends to produce subdomains that are largely rectilinear. On a uniform mesh, an appropriately chosen number of partitions would result in a cubical decomposition of the computational domain. This "best case" establishes the minimum ratio of communication to computation (surface/volume ratio) for SFC partitioned meshes. For any given number of cells, one can conceive of an idealized cubical partitioner which would communicate across some number of faces, $F_c$ given by:

$$F_c = 6\left(\frac{N_{\text{Cells}}}{N_{\text{Partions}}}\right)^{2/3} \qquad (1)$$

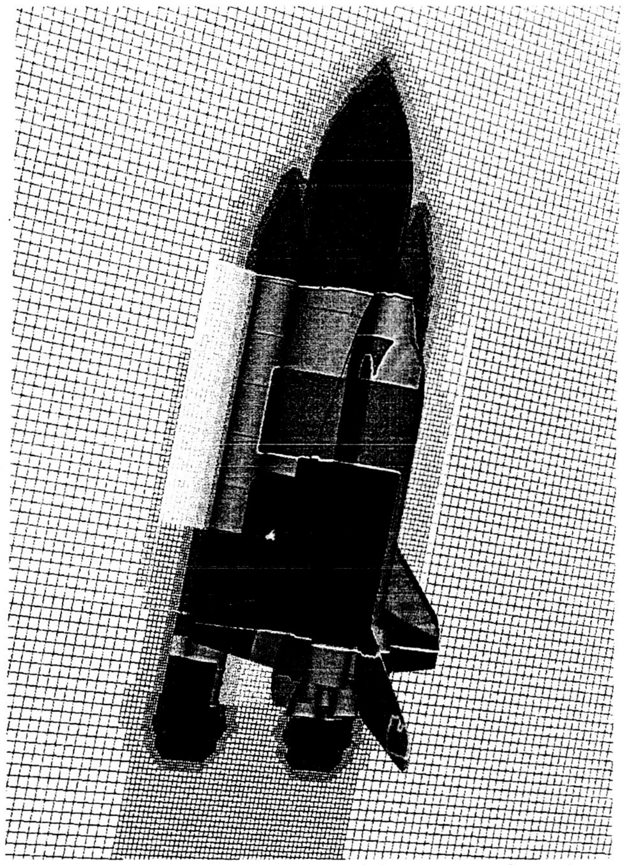
*Figure 7:* 4.4 M cell mesh around full SSLV configuration including orbiter, external tank, solid rocket boosters, and fore and aft attach hardware. Mesh color indicates partitioning (16 partitions, U-ordering).

where $N_{cells}$ is the total number of cells in the domain and $N_p$ is the number of subdomains. With this as a guide one can examine the quality of the mesh partitioning provided by the N and U-orderings.

Figure 8 examines the average partition statistics for a 1.6M cell adaptively refined mesh decomposed into 64 subdomains. Even with this large number of subdomains on a relatively small mesh, this figure shows that both SFC partitioners perform well, with the Peano-Hilbert decomposition showing only a 4% overhead. *(in the final paper more exhaustive statistics will be gathered, showing how these results vary with number of partitions, and mesh size. We will also examine the average AND worst partitions to provide better insight into predicted runtimes).*

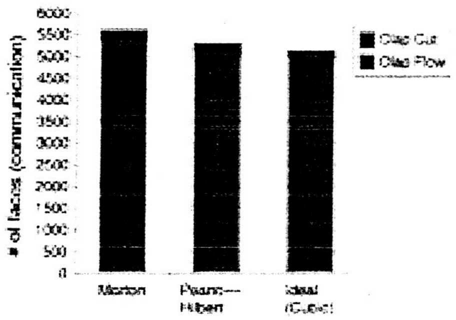

*Figure 8:* Partitioning statistics for a 1.6M cell adapted cartesian mesh decomposed into 64 subdomains with two different SFCs and the idealized cubical partitioner described in the text.

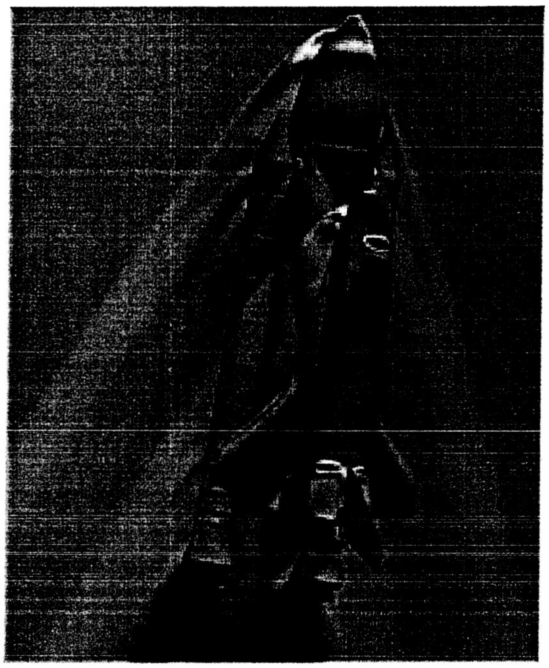
*Figure 9:* Isobars in discrete solution for SSLV configuration at $M_\infty = 2.6$, $\alpha = 2.09°$, $\beta = 0.8°$. This case was used for scalability results.

## 4.1 Scalability and Performance

Scalability tests were conducted using the 4.4M cell mesh from fig. 7. Flow conditions for the test had the SSLV at $M_\infty = 2.6$, $\alpha = 2.09°$, $\beta = 0.8°$. Isobars of the discrete solution are shown in fig. 9. The inviscid multilevel solver from [30] and [31] was used with the MPI based communication outlined in [32] and now extended to support multigrid accelleration.

Figure 10 shows scalability results for this case on an SGI Origin 3000 (600Mhz, Mips R14000 CPUs) using multigrid with MPI. Performance is essentially linear across the experiment with parallel speedups of about 430 showing on 512 cpus. The measured performance starts to tail-off above 128 CPUs due to the decrease in the ratio of computation to communication as the subdomains shrink. On 512 CPUs there are only about 8500 cells in each partition leading to an excessive amount of communication. On 128 CPUs the code is achieving parallel speedups of approximately 120, and there

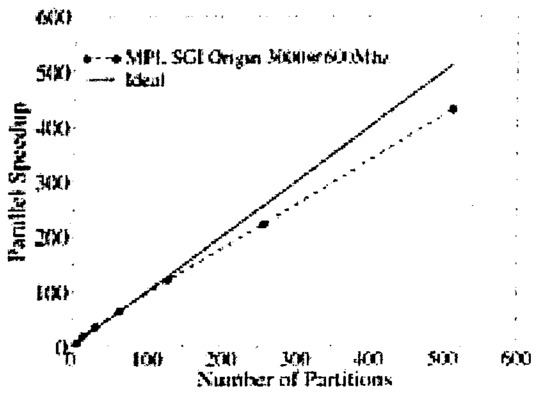

*Figure 10:* Parallel scalability on 4.4 M cell SSLV test case on SGI Origin 3600 with 600Mhz Mips R14000 processors.

are just under 35,000 cells per partition. *(More detailed analysis will be included in the final paper).*

A second example with this configuration was run on a much smaller distributed memory machine to assess performance on more commonplace hardware. Figure 11 shows scalability on a 16 CPU cluster of dual-processor Pentium 4 processors each clocked at 2.4Ghz. The eight boxes are connected by a dedicated gigabit LAN through a eight port switch. This example used a smaller test case with only 1.8 M cells in the mesh (same SSLV configuration). On this architecture, performance remains linear, and parallel speedups between 12 and 13 are achieved on 15 CPUs. *(further analysis planned for final paper)*

## 5 Inter-mesh interpolation

*The final paper will include a description of a completely new linear time 3D interpolation algorithm including runtimes. This algorithm permits solution transfer between remeshed solution domains even when the geometry and mesh are radically changed. The space filing curve permits this 3D interpolation problem to be solved in seconds on a PC. Applications will include:*

- *demonstration of algorithm as interpolation operator when geometry is perturbed. e.g. control surface deflections & design applications*
- *demonstration of use as an interpolation operator for remeshed solution domains in time-dependent moving-body simulations.*

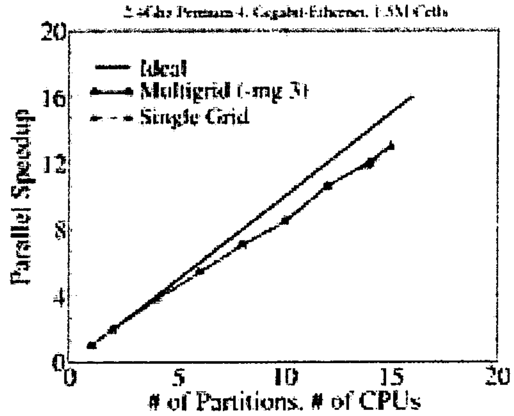

*Figure 11:* Parallel scalability on 1.6 M cell SSLV test case on dual-CPU PC cluster with gigabit interconnect.

## 6 References

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

[2] 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.

[3] Ollivier-Gooch, C., "Robust Coarsening of Unstructured Meshes for Multigrid Methods." Presented at the 14th AIAA Computational Fluid Dynamics Conference, Norfolk, Virginia, June, 1999.

[4] Venkatakrishnan, V and Mavriplis, D. J.,"Agglomeration multigrid for the three-dimensional Euler equations." *NASA/CR-191595*, 1995.

[5] 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.

[6] 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.

[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] Melton, J.E., *Automated Three-Dimensional Cartesian Grid Generation and Euler Flow Solutions for Arbitrary Geometries*, Ph.D. thesis, Univ. CA.-Davis, Davis CA, 1996.

[9] Berger, M.J and Melton. J.E., "An accuracy test of a Cartesian grid method for steady flows in complex geometries." *Proc. 5th Intl. Conf. Hyperbolic Prob.*, Stonybrook NY, Jun. 1994. Also RIACS report 95-02, NASA Ames, Feb., 1995.

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

[11] Coirier, W.J., "An Adoptively-Refined, Cartesian, Cell-Based Scheme for the Euler Equations," *NASA TM-*

*106754*, Oct., 1994. also Ph.D. Thesis, Univ. of Mich., Dept. of Aero. and Astro. Engr., 1994.

[12] 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.

[13] De Zeeuw, D., and Powell, K., "An adaptively-refined Cartesian mesh solver for the Euler equations," *AIAA Paper 91-1542*, Jun., 1991.

[14] 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.

[15] Jameson, A.,"Solution of the Euler equations for two dimensional transonic flow by a multigrid method." Applied Mathematics and Computations, **13**:327-356. 1983.

[16] Anderson, W.K., Thomas, J.L., and van Leer, B., "A comparison of finite volume flux vector splittings for the Euler equations" *AIAA Paper 85-0122,* Jan. 1985

[17] Venkatakrishnan, V., "On the accuracy of limiters and convergence to steady state solutions." *AIAA Paper 93-0880,* Jan. 1993.

[18] Kreiss, H.O., Manteuffel, T.A., Swartz, B., Wendroff, B., and White, A.B., "Supraconvergent schemes on irregular grids." *Math Comp.* **47**, 1986.

[19] Wendroff, B., and White, A.B. "A supraconvergent scheme for nonlinear hyperbolic systems." Comput. Math. Appl., **18**, 1989.

[20] 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.

[21] van Leer, B., Chang-Hsien, T., and Powell, K., "Design of optimally smoothing multi-stage schemes for the Euler equations." *AIAA Paper 89-1933-CP,* Jun. 1989.

[22] Samet, *The design and analysis of spatial data structures.* Addison-Wesley Series on Computer Science and Information Processing, Addison-Wesley, 1990.

[23] 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.

[24] 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 SFB report 342/1/96A, Institut für Informatik, TU München, 1996.

[25] Behrens, J., and Zimmermann, J., "Parallelizing an unstructured grid generator with a space-filling curve approach, in Euro-Par 2000 Parallel Processing, 6th International Euro-Par Conference, Munich, Germany, August/September 2000, Proceedings, A. Bode, T. Ludwig, W. Karl, R. Wismüller (Eds.), Lecture Notes in Computer Science 1900, Springer-Verlag, 2000, 815-823.

[26] Pilkington, J.R., and Baden, S.B., "Dynamic partitioning of non-uniform structured workloads with spacefilling curves." *IEEE Trans. on Parallel and Distrib. Sys.* 7(3), Mar. 1996

[27] 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.

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

[29] 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.

[30] Berger, M. J, Aftosmis, M. J., Adomavicius, G., "Parallel multigrid on Cartesian meshes with complex geometry"., Proceedings of the 8th International Conference on Parallel CFD, Trondheim Norway, Jun. 2000.

[31] Aftosmis, M. J., Berger, M. J, and Adomavicius, G., "A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries." *AIAA Paper 2000-0808*, Jan. 2000.

[32] Marshall, D. D., Aftosmis, M.J., and Ruffin, S.M., "Study of Parallelization Enhancements for Cartesian Grid Solver", Proceedings of Parallel CFD 2002, Jaeri-Kri, Kansai Science City, Japan., May 2002.

This work is an extended abstract to be submitted to the 42nd Aerospace sciences meeting and exhibit session on applied CFD. The conference will be in Reno NV Jan 6-11, 2004.

The proposed paper presents a variety novel uses of Space-Filling-Curves (SFCs) for Cartesian mesh methods in CFD. While these techniques will be demonstrated using non-body-fitted Cartesian meshes, most are applicable on general body-fitted meshes - both structured and unstructured. We demonstrate the use of single O(N log N) SFC-based reordering to produce single-pass (O(N)) algorithms for mesh partitioning, multigrid coarsening, and inter-mesh interpolation. The intermesh interpolation operator has many practical applications including "warm starts" on modified geometry, or as an inter-grid transfer operator on remeshed regions in moving-body simulations. Exploiting the compact construction of these operators, we further show that these algorithms are highly amenable to parallelization. Examples using the SFC-based mesh partitioner show nearly linear speedup to 512 CPUs even when using multigrid as a smoother. Partition statistics are presented showing that the SFC partitions are, on-average, within 10% of ideal even with only around 50,000 cells in each sub-domain. Results are also presented on distributed computing clusters. The inter-mesh interpolation operator also has linear asymptotic complexity and can be used to map a solution with N unknowns to another mesh with M unknowns with O(max(M,N)) operations. This capability is demonstrated both on moving-body simulations and in mapping solutions to perturbed meshes for finite-difference-based gradient design methods.