Domain Decomposition By the Advancing-Partition Method for Parallel Unstructured Grid Generation

Shahyar Z. Pirzadeh*
NASA Langley Research Center, Hampton, Virginia, 23681

and

George Zagaris†
College of William and Mary, Williamsburg, VA 23187

A new method of domain decomposition has been developed for generating unstructured grids in subdomains either sequentially or using multiple computers in parallel. Domain decomposition is a crucial and challenging step for parallel grid generation. Prior methods are generally based on auxiliary, complex, and computationally intensive operations for defining partition interfaces and usually produce grids of lower quality than those generated in single domains. The new technique, referred to as "Advancing Partition," is based on the Advancing-Front method, which partitions a domain as part of the volume mesh generation in a consistent and "natural" way. The benefits of this approach are: 1) the process of domain decomposition is highly automated, 2) partitioning of domain does not compromise the quality of the generated grids, and 3) the computational overhead for domain decomposition is minimal. The new method has been implemented in NASA's unstructured grid generation code VGRID.

Nomenclature

A	=	cell aspect ratio
$\boldsymbol{g}_j\{1,2,3,\ldots m_j\}$	=	j^{th} partition-grid containing a set of m field elements in partition j
$\boldsymbol{G}_i\{1,2,3,\ldots m_i\}$	=	i^{th} sub-grid containing a set of m field elements in subdomain i
$\boldsymbol{G}\{1,2,3,\ldots m_G\}$	=	global grid containing a set of m field elements in the main domain
k	=	empirical parameter used for computing the center of mesh density
L	=	level of binary domain partitioning/decomposition
m	=	number of grid elements in the field
m_i	=	mass of the i^{th} particle; also number of grid elements
N	=	total number of subdomains/sub-grids
n_G	=	number of surface mesh elements
n	=	number of particles; number of boundary grid elements
$\bar{r}_i$	=	position vector of the i^{th} particle
$\bar{R}$	=	position vector of the center of mass (center of mesh density)
s_i	=	local grid spacing at the i^{th} grid node
x,y,z	=	Cartesian coordinate axes
$\mathcal{F}$ and $\mathcal{R}$	=	mapping functions used for local/global grid renumbering
ω_i	=	weighting function used for computing the center of mesh density
Ω_1, Ω_2	=	binary subdomains produced at each level L of partitioning

* Senior Research Engineer, Configuration Aerodynamics Branch, NASA Langley Research Center, Mail Stop 499.
† Graduate Student, Department of Applied Science.

I. Introduction

THANKS to recent advances in the field of Computational Fluid Dynamics (CFD), engineers and researchers are nowadays able to solve large-scale, real world aerodynamic problems that had not been tractable a few years ago. Among the enabling factors contributing to this success are: 1) advanced grid generation capabilities to discretize complex computational domains, 2) efficient and accurate flow solution algorithms, and 3) widespread availability of powerful computational resources; especially, parallel computer architectures.

Generation of large size grids on complex configurations constitutes a substantial portion of a typical CFD activity (in terms of time and effort) and requires considerable amounts of computational resources. The increasing demand for more accurate Navier-Stokes (NS) flow simulations on complex models requires grids of tens (and soon even hundreds) of million elements. Generation of such grids in a single large domain using one computer Central Processing Unit (CPU) poses a challenge because of the speed and memory limitation of typical computers available to most CFD practitioners. In addition, experience has shown that generation of NS grids with minuscule length scales in very large domains become more difficult due to floating-point operations and numerical precision issues that affect the grid quality and the generation robustness. Domain decomposition is an enabling approach that substantially facilitates the process of grid generation by dividing a large problem into smaller components and meshing each subdomain either with a single CPU sequentially or on a cluster of computers in parallel.

Although parallel computation has been in use for solving complex problems in aerodynamics and other disciplines with great success for quite some time, it has not been fully utilized for the purpose of grid generation in a large-scale production level to date. There are some obstacles that have prevented the routine application of parallel computations for generating grids, especially in three dimensions (3D). One of the common requirements for parallel computation is domain decomposition. While the decomposition of discretized (meshed) domains can be easily accomplished using conventional methods for parallel computation of fluid flow problems, partitioning of an "empty" domain for grid generation has proven to be a non-trivial task. There are several technical issues involved that make it especially difficult to decompose a domain prior to generation of grid in parallel. Among the challenges are:

1) Automatic construction of partition interfaces in such a way that they intersect the domain boundaries (and other partition interfaces) seamlessly.
2) Discretization of partitions with high-quality grid elements compatible with those generated in the volume (no introduction of anomalies and inconsistencies).
3) Estimation of loads such that approximately equal numbers of grid elements are generated in the sub-domains and uniform amount of work is distributed among the processors (load balancing).
4) Ensuring that the partition interfaces will not interfere with the natural process of grid generation in the field and, thus, will not compromise the robustness of the underlying grid generation method nor the quality of the generated grids.
5) Maintaining a low level of computational overhead for domain decomposition so that the cost of partitioning and decomposition will not overwhelm the cost of grid generation.

Furthermore, the implementation of such a complex process automatically and without human intervention is a challenge in itself.

There are a variety of techniques in the literature for domain decomposition (e.g., Ref. 1-6). Some of these techniques are based on preprocessing operations[4,5], and many require some type of auxiliary grids for defining partitions/subdomains[2,5]. The implementations of these approaches are usually complicated, and the methods are computationally inefficient[1]. The procedural complexities and additional operations often lead to excessive overhead for domain decomposition and result in unsatisfactory performance of the parallel grid generation methods[5]. In addition, extensions of these approaches are too complicated for the decomposition of domains for generating thin-layered, anisotropic, NS grids, mainly due to the obstacle items 1 and 2 above.

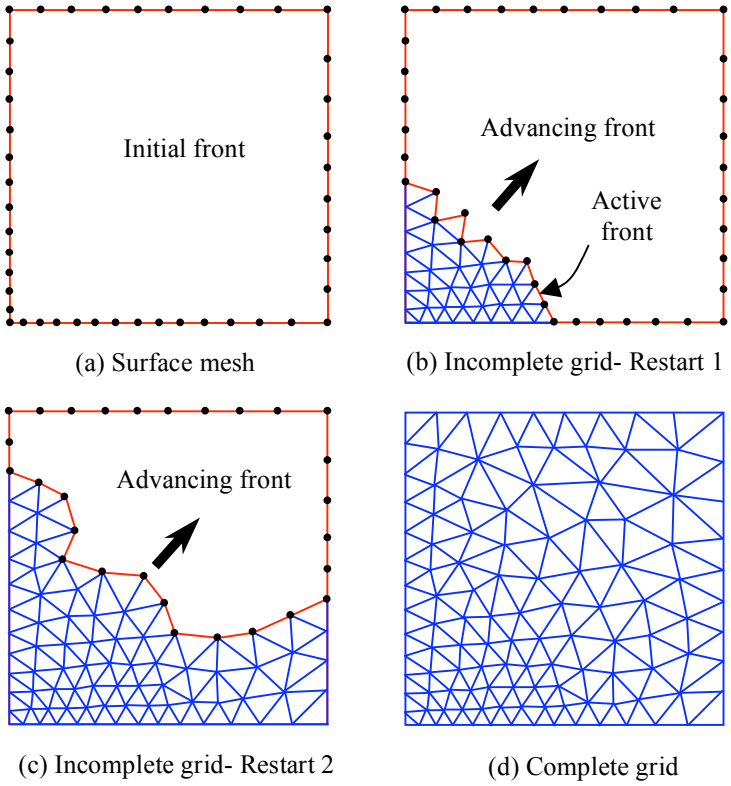
The present work is an attempt to alleviate some of the above shortcomings by consolidating the processes of domain decomposition and mesh generation in a compatible and natural way. The new method is referred to as Advancing-Partition[7] and has been implemented in the unstructured grid generation software VGRID[8,9]. The focus of this paper is on two crucial and fundamental issues concerning parallel grid generation: 1) partitioning and decomposition of computational domains and 2) load estimation for balanced distribution of parallel computation. The full implementation and presentation of the new techniques in the context of a parallel grid generation framework are beyond the scope of this paper and are reported in an accompanying paper[10] separately.

II. The Advancing-Partition Method

The present method of domain decomposition is an extension of the meshing methodology incorporated in the grid generation software VGRID and is best described by first presenting the underlying grid techniques followed by their implementation for domain decomposition.

A. Background Methodology

The unstructured grid generation software VGRID, developed at the NASA Langley Research Center, is based on two marching techniques referred to as the Advancing Front[11] (AF) and Advancing Layers[12] (AL) methods for generating Euler (inviscid) and NS grids, respectively. A grid is generated in these methods by forming tetrahedral cells in the field one by one or one layer of cells at a time. The triangular (linear in 2D) faces of cells exposed in the field form a front over which new cells are constructed. The process starts from the surface mesh (which acts as an initial front) and progressively adds new grid elements in the computational domain as illustrated in 2D in Figure 1. During this process, old faces on the front are covered by newly formed cells and become inactive while new faces are created and introduced to the active front. The growing and marching process continues until the entire domain is filled with tetrahedral cells. At this point, no active face remains on the front, and the AF/AL algorithms terminate.



(a) Surface mesh (b) Incomplete grid- Restart 1

(c) Incomplete grid- Restart 2 (d) Complete grid

Figure 1. Schematic of grid generation with the Advancing-Front method through multiple restart runs.

During the mesh generation process, a grid spacing function (derived from a set of user-prescribed source elements) is used to determine the resolution and other characteristics of the mesh.[13,14] Although the grid elements are theoretically generated in a random order during the AF process, the spacing function normally guides the advancement of the front from regions of fine grid towards areas of coarser mesh resolution as shown in Fig. 1.

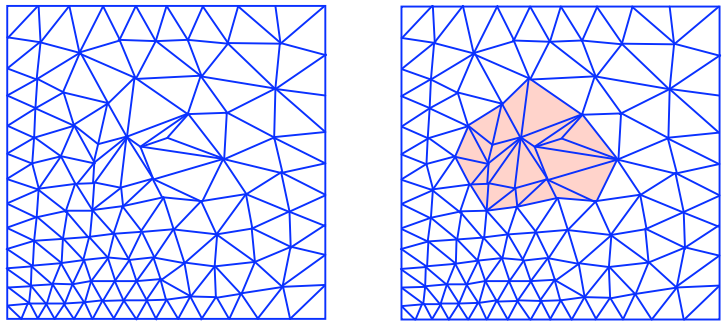
A distinctive characteristic of the Advancing-Front method is its inherent flexibility for generating grids and the ease of restructuring the grid makeup at anytime. Part of this flexibility is due to the ability of the method to introduce nodes and cells in the field simultaneously, i.e., elements are constructed locally and independently from the grid structure in other locations in the field. Consequently, a section of the mesh can be altered and regenerated at any time without affecting the grid in other areas.

During the AF process, the computational domain is divided into two regions at any instance: 1) the area that has already been meshed and 2) the remainder of the domain that is currently void. These two sections are separated by a common interface consisting of triangular or linear (in 2D) faces (see Figs. 1b and c). Once an active face on the front is covered with a new cell, it becomes dormant and will not affect the generation process anymore. The characteristics of new cells being formed depend only on the configuration of the front and the information that influence the empty portion of the domain. This means that the domain of dependence is always limited to the unmeshed section bounded by the current front. This is similar to an initial value problem in which a solution is obtained by marching outward from a prescribed initial condition.

Since a section of a mesh once generated can never influence the rest of the grid, the process of AF can be interrupted at any time and restarted without "carrying" the generated grid elements to the next stage. In other words, the problem of generating a large grid can be divided into several smaller independent sub-problems. This unique feature of the AF method facilitates two important and enabling operations for grid generation: grid restarting and local remeshing. These two capabilities are among the salient features of the VGRID software system and are currently employed for the following purposes.[15]

1) Postprocessing operation for grid closure and quality improvement in which sections of a grid with poor quality elements can be removed and remeshed as depicted in Figure 2.
2) Restart capability for generating large grids on small computers through several separate restart runs.

The restart capability implemented in VGRID is based on a recursive local/global renumbering system that drastically reduces the computer memory requirement in each run. With renumbering, the grid indices (counters) start from the cell number 1 in each restart run rather than the accumulated global number of the previous runs. In other words, each restart run becomes an independent smaller grid generation problem in which memory is allocated for array dimensions based on the number of elements to be generated locally and not the global number of the mesh elements. A local/global mapping of the grid indices is created and saved at each restart run for bookkeeping of the grid segments generated separately. The mapping is used at the end of the grid generation process to renumber and merge all the separate grid segments into a final global (single) mesh.



(a) Grid containing distorted cells

(b) Pocket of distorted cells

(c) Removal of pocket of cells for local re-meshing

(d) Restored grid

Figure 2. Schematic of local re-meshing for improving grid quality.

B. Domain Decomposition

The Advancing-Front (AF) method as well as the restart and local-remeshing features described above lay the foundation for a novel domain decomposition concept referred to as the "Advancing-Partition" (AP) method. Unlike many conventional techniques that rely on a "physical" surface discretization for partitioning the domain prior to the volume grid generation, the new method employs an "imaginary" partitioning plane in the field to define the location across which the domain will be partitioned as part of the volume grid generation.

The new method is called Advancing-Partition because of its similarity to the conventional AF technique for generating tetrahedral cells. Figure 3 illustrates the process of domain decomposition in 2D for visual clarity. In this method, the actual partitioning of the domain takes place through an initial stage (partitioning phase) of the volume grid generation. During this phase, tetrahedral cells are generated similarly to the marching process described earlier. However, the new procedure adds only one layer of tetrahedral cells advancing in the volume in two dimensions along the partition-plane. This strategy is in contrast to the conventional AF method in which all faces remain active on the front and cells are added in three dimensions with no directional restrictions. During the new marching process, the front is made of only the faces of new cells that intersect the partition-plane. The remaining faces on either side of the plane are designated as "inactive" and do not participate in the marching process. Consequently, only tetrahedral cells that intersect the partition-plane are formed and advance in the field one by one (Fig. 3b). The marching process continues until no "active" face remains on the front. The outcome of this operation is a dividing wall, made of one layer of tetrahedral cells at the location of the partitioning plane, which splits the domain (or a subdomain) into two segments. The partial mesh generated in this initial phase of volume grid generation is called the "partition-grid" and serves as a separator for domain decomposition (Fig. 3c).

The processes of partitioning and domain decomposition with the AP method as well as the generation of volume grids in multiple subdomains are summarized below and in the flowchart in Figure 4.

a) Generate a surface mesh $\boldsymbol{G}\{1,2,3,...n_G\}$ as in a regular (sequential) grid generation process. The surface mesh serves as the initial front for volume grid generation.

b) Determine the location of a Cartesian plane in the x, y, or z direction that hypothetically divides the domain/subdomain into two sections (partition-plane). The location of the partition-plane is determined in such a way that the numbers of grid elements in the subdomains will approximately be equal (load estimation).

c) Identify the triangular faces on the surface mesh that intersect the partition-plane (initial partition front).

d) Generate a partial grid ($\boldsymbol{g}_j$) using the AP method to divide the domain/subdomain at the location of the partition-plane (partitioning phase). The generated partition-grid is a subset of the final global mesh $\boldsymbol{G}$ and is saved in a file or CPU for integration with other mesh segments generated in the subdomains, i.e.,

$$\mathscr{T}(\boldsymbol{g}_j) \subset \boldsymbol{G}, \text{ where } \mathscr{T}: \boldsymbol{g}_j \rightarrow \boldsymbol{G}.$$

The function $\mathscr{T}$ maps the partition-grid subset $\boldsymbol{g}_j\{1,2,3,...m_j\}$ into the global grid set $\boldsymbol{G}\{1,2,3,...m_G\}$.

e) Collect the triangular faces on either side of the partition-plane (including those on to the surface mesh and on the partition-grid) in two separate subsets Ω_1 and Ω_2 (binary domain decomposition phase).

f) Renumber each set of grid elements locally and construct a local/global mapping function for each subdomain ($\mathscr{R}$).

g) Repeat steps (b-f) through L levels of binary partitioning/decomposition. The total number of subdomains (N) is equal to 2^L, and the number of partitions is $N\text{-}1$.

h) Use the conventional AF process to mesh each subdomain ($\boldsymbol{G}_i$) either sequentially or in parallel. The sub-grid $\boldsymbol{G}_i$ is a subset of the final global mesh, i.e.,

$$\mathscr{R}(\boldsymbol{G}_i) \subset \boldsymbol{G}, \text{ where } \mathscr{R}: \boldsymbol{G}_i \rightarrow \boldsymbol{G}.$$

This step also includes a postprocessing operation for grid closure/repairing as described below (meshing phase).

i) Merge all grid segments including N sub-grids ($\boldsymbol{G}_i$) and $N\text{-}1$ partition-grids ($\boldsymbol{g}_j$) using the local/global mapping functions generated in steps (d) and (f), i.e.,

$$\boldsymbol{G}\{1,2,3,...,m_G\} = \sum_{i=1}^{N} \mathscr{R}(\boldsymbol{G}_i\{1,2,3,...,m_i\}) + \sum_{j=1}^{N-1} \mathscr{T}(\boldsymbol{g}_j\{1,2,3,...,m_j\}) \tag{1}$$

This domain re-composition phase provides the final grid in the original single domain.

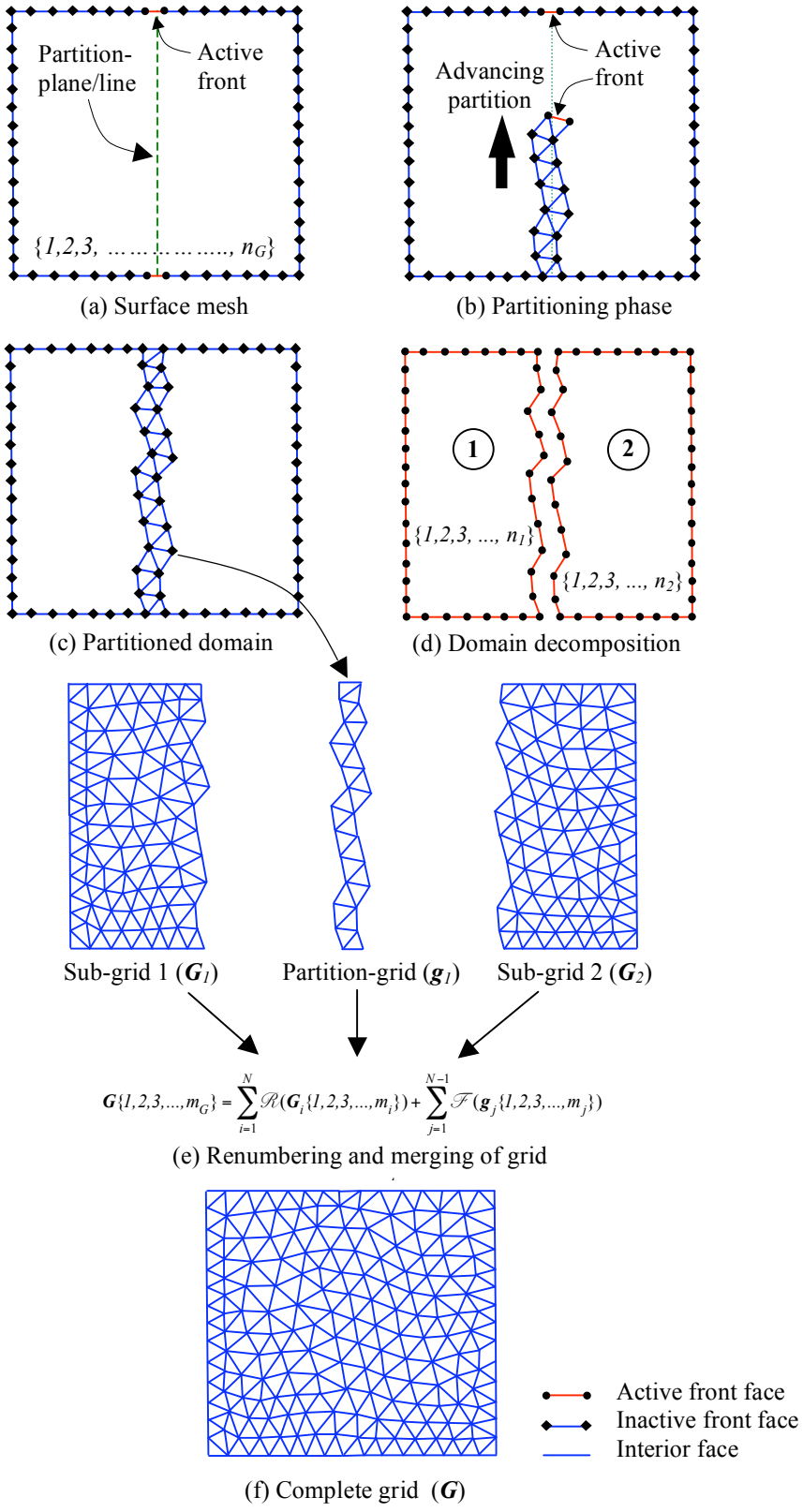


Figure 3. Schematic of domain decomposition by the Advancing-Partition method.

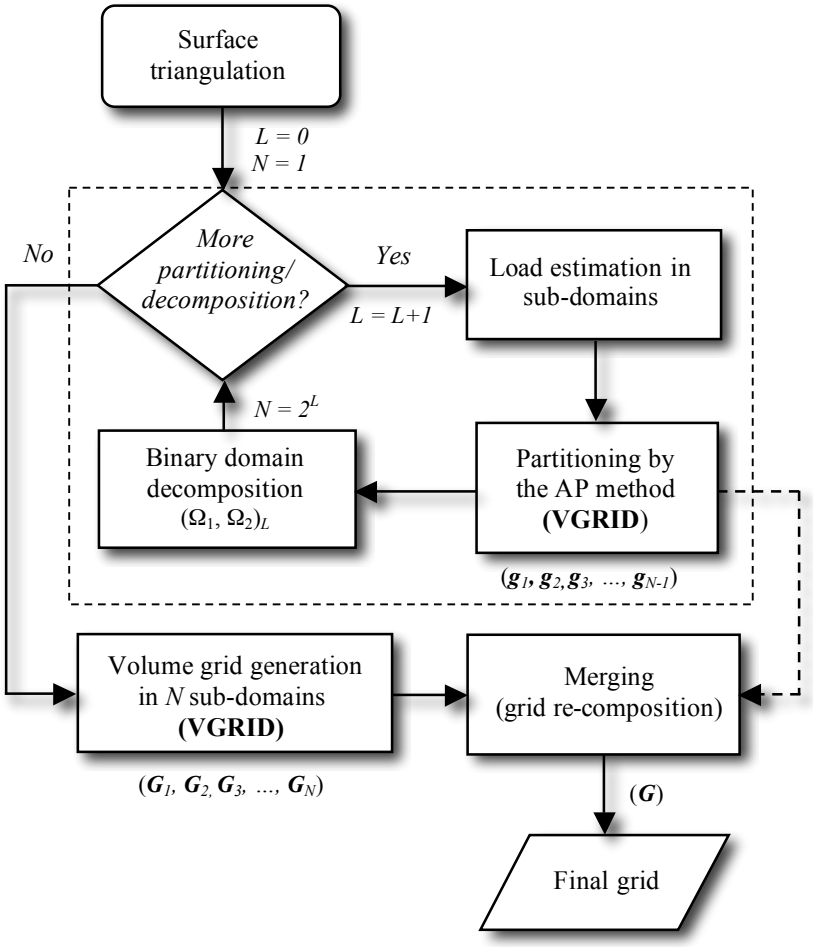


Figure 4. Flowchart of domain decomposition in *L* levels using the Advancing-Partition method and volume grid generation in *N* sub-domains with VGRID.

During the regular grid generation process with VGRID, the AF algorithm is sometimes unable to properly connect (close) the network of cells at some isolated locations. These irregularities usually occur when the front becomes severely entangled due to: 1) geometric complexities, 2) inconsistencies in the size and shape of the faces on the front, and/or 3) multiple fronts approaching from different directions. In such cases, the process simply skips the problematic faces and leaves the front open. These defective pockets are normally repaired through a postprocessing operation (using a separate code called POSTGRID), which removes the entangled cells and remeshes the defective areas in a restart run (see Figure 2).

For generating the partition grids, the new AP algorithm is self-sufficient and includes the above cleaning and remeshing operations. Therefore, no postprocessing operation is required after the partitioning phase. For the test cases examined thus far, the method has successfully partitioned the domains even for generating highly stretched grids on complex configurations. Furthermore, the AP algorithm is more robust and efficient than the regular AF method because generation of cells in a semi-2D fashion involves fewer search and check for cross-face operations.

C. Load Estimation and Center of Mesh Density

One of the crucial factors that directly affects the performance of a parallel computation scheme is the issue of load balancing. Unlike the conventional domain decomposition methods for solvers that exploit the mesh under consideration as guidance for load balancing, a decomposition method for grid generation must rely on other means to estimate loads prior to generation of grid. Load balancing for unstructured grid generation is a complicated issue as the exact number of grid elements cannot be readily determined in advance, especially, for grids generated with

the AF method. Obviously, for synchronized parallel grid generation, a domain should ideally be divided into subdomains that would contain approximately equal number of grid elements. In the present domain decomposition method, the location of the partition-plane determines the distribution of grid elements among subdomains and, thus, the loads being allocated to the computer processors.

The concept of Center of Mass, as reported in Ref. 6, provides a reasonably accurate prediction of load distribution in subdomains. In the present work, a variation of this concept referred to as the "Center of Mesh Density" (CMD) is employed to determine the position of the partition-planes. The CMD is analogous to center of mass of a heterogeneous system of particles in a uniform gravity field. The center of mass is a function of the positions and masses of individual particles in the system. It represents a unique point at which the effective mass of the system is concentrated (for the purpose of calculating the "first moment") and, therefore, the system is perfectly in balance, i.e.,

$$\bar{R} \sum_{i=1}^{n} m_i = \sum_{i=1}^{n} m_i \bar{r}_i \tag{2}$$

where $\bar{R}$ is the position vector of the center of mass, $\bar{r}_i$ is the position vector of the i^{th} particle with a mass of m_i, and n is the total number of particles. For a non-uniform unstructured grid consisting of a number of tetrahedral elements of different sizes, the mass can be replaced by a weighting function representative of the mesh resolution (density).

In the absence of a mesh, prior to generation of the volume grid, the position of the partition-plane can be approximated using a CMD function based on the resolution of surface mesh. Load estimation based on the surface mesh is reasonably accurate as long as the grid distribution on the surface resembles the variation of mesh in the volume. Obviously, the accuracy decreases when the distribution of grid density in the field is not reflected on the surface. Alternatively, an auxiliary medium such as a cloud of points in the field or a secondary simple mesh (e.g., an octree mesh) may be employed as a more accurate representation of the actual unstructured grid for load estimation. Such an auxiliary mesh is constructed using the same spacing function that provides for generation of the unstructured grid.

For generating an anisotropic unstructured grid, the CMD is approximated in this work using the function

$$\bar{R} = \frac{\sum_{i=1}^{n} \bar{r}_i / \omega_i}{\sum_{i=1}^{n} 1 / \omega_i}. \tag{3}$$

The weighting function ω_i in Eq. (3) represents an effective grid length scale defined as

$$\omega_i = s_i A_i^k \tag{4}$$

where s_i is the local grid spacing computed at the i^{th} node of the surface mesh (or an auxiliary grid) using the spacing functions that determines the unstructured grid resolution[13,14], and A_i is the corresponding cell aspect ratio. The variable n in Eq. (3) is the total number of surface mesh or auxiliary grid nodes, and the exponent k in Eq. (4) is an empirical parameter set to a value of 1/3 for estimates based on the actual surface triangulation and 4/3 for computations based on an auxiliary isotropic grid. In the present work, an octree mesh generated by recursive refinement using VGRID spacing functions is optionally used as an auxiliary grid for load estimations.

As mentioned earlier, the surface mesh option provides accurate results if the distribution of grid on the surface approximates that in the volume reasonably well. An octree mesh provides a better load estimation when the unstructured volume grid varies in the field differently from the surface mesh. The resolution of an octree auxiliary mesh is governed by the same spacing function used in VGRID and, thus, closely represents the grid distribution in the field similar to the actual isotropic unstructured grid. An advantage of octrees is that their generation is automatic and relatively fast. However, a drawback is due to the fact that they are made of Cartesian equilateral hexahedrons and do not reflect multi-dimensional anisotropy of unstructured grids. Therefore, the number of their elements in Eq. (3) is not an accurate representation of the number of grid elements in a highly stretched unstructured mesh. To compensate for the absence of anisotropy and to improve the accuracy of octrees for load

estimation, the exponent k in Eq. (4) is set to 4/3. This augmented value of k, in effect, supplements additional weight for the local cell aspect ratio (A_i) in the estimation.

Although the CMD function is based on the same spacing functions used in VGRID and provides reasonably good results for most cases, it may still lack the desired accuracy in predicting balanced loads among subdomains for complex problems. Load estimation is a challenging problem for partitioning of domains prior to generation of unstructured grids and requires further investigations.

III. Sample Result

The new method of domain decomposition by Advancing Partition has been implemented in the unstructured grid generation code VGRID. Furthermore, the development of a parallel grid generation framework based on the present capability is currently underway for generating grids on multiple CPU's in parallel. The preliminary experience with a number of test cases has produced promising results. While the demonstration and discussion of parallel grid generation is beyond the scope of this paper, a simple example of domain decomposition is presented in this report to illustrate the concept and the capability of the method.

Figure 5 depicts a tetrahedral grid generated around a sphere inside a cubical domain that is partitioned in two segments. The sphere is positioned in the cube slightly off center to remove the geometric symmetry of the overall configuration for testing the load-balancing ability of the CMD function. Only one level of domain decomposition has been applied in this case for the purpose of demonstration and visual clarity. Otherwise, the partitioning and decomposition operations can be repeated for each subdomain recursively, with the direction of the partition-planes automatically determined for the best possible load balancing. Figure 5b shows the partition-grid (separator) generated at the location of the partition-plane, which intersects the sphere in the middle. After the faces on either side of the partition-grid are identified and extracted, the domain is decomposed and meshed in each segment separately (Fig. 5c). Finally, the component grids are assembled to retrieve the complete discretized domain. As mentioned earlier, the partition-grid not only acts as the separator but is also part of the final mesh as illustrated in Figure 5d.

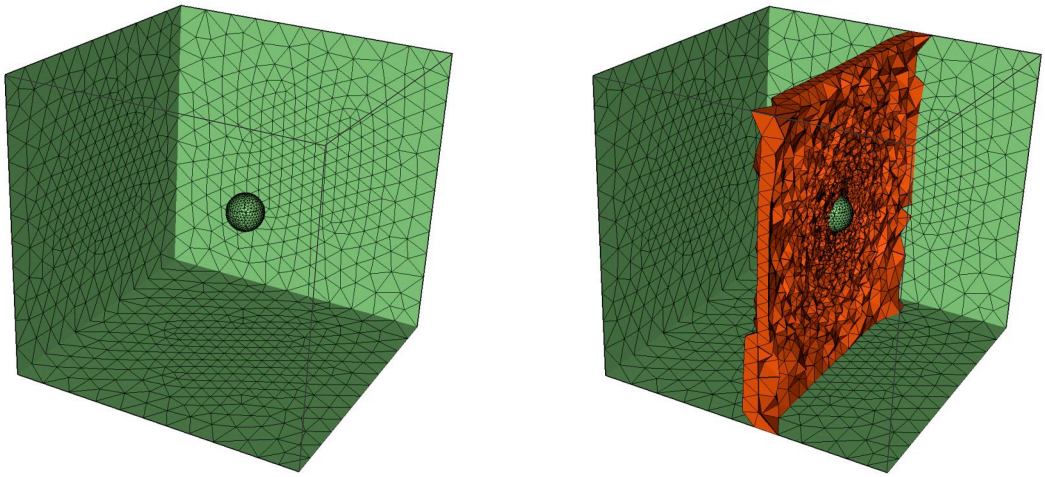
The grid shown in Fig. 5 contains about 195,000 tetrahedral cells and is intentionally made coarse for the purpose of visual clarity. Table 1 provides the grid statistics and the generation time for a finer version of the grid, also generated in two subdomains.

Table 1. Grid Statistics for the Sphere-in-Cube Example

Components	Nodes	Tetrahedra (Triangles)	CPU Time (min.)
Surface Mesh	110,577	(221,146)	1.32
Partition-Grid	42,573	125,802	0.23
Sub-Grid 1	1,127,114	6,477,826	13.11
Sub-Grid 2	1,128,289	6,484,713	13.14*
Merging			0.24
Post-processing			0.16
Total	2,408,553	13,088,341	15.09

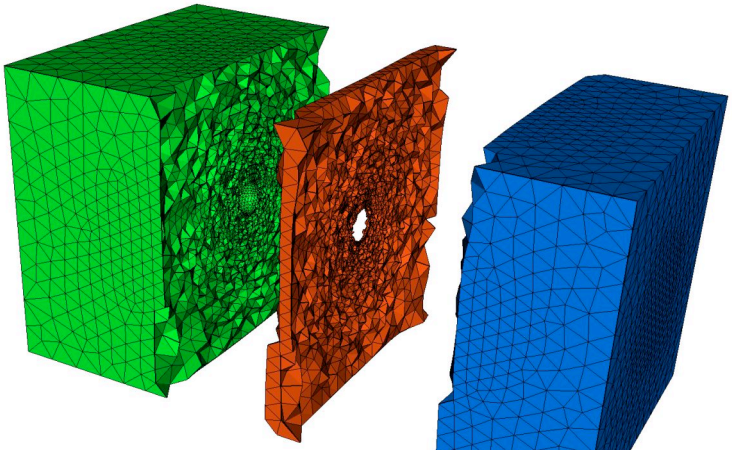
* The largest of CPU time for generating sub-grids is used to compute the total parallel grid generation time.

The numbers of grid elements in the two sub-grids and the corresponding generation times in Table 1 indicate that the CMD function has produced a good balance of loads and, thus, synchronization of grid generation in the subdomains. A balanced load distribution is important for scalable performance of parallel grid generation. Note that the computational time spent for partitioning of the domain (partition grid) is a fraction of the total grid generation time. Nevertheless, it is considered as part of the grid generation time and not as an overhead in this method.

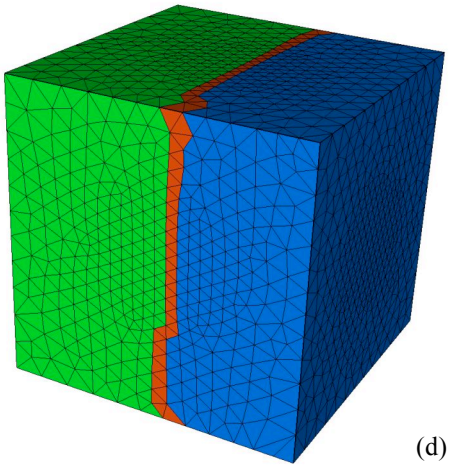


(a) Surface mesh

(b) Partitioned domain



(c) Components of volume grid generated in subdomains



(d) Complete (merged) grid

Figure 5. Example of domain decomposition by the Advancing-Partition method for a Sphere-in-Cube configuration.

This grid was generated using two processors on a Macintosh (Mac Pro) computer with two Dual-Core Intel Xeon 3-GHz processors. A similar grid containing 13,062,394 tetrahedra was also generated in a single domain using one processor on the same computer for the performance comparison. The total generation time for the single domain case is 26.90 minutes compared to 15.09 minutes for the parallel run - a speedup of 1.78 for parallel grid generation using two partitions/processors.

As noted earlier, the objective of this paper is to address two important issues concerning parallel grid generation: domain decomposition and load estimation. Other aspects of parallel grid generation such as design of a parallel framework, efficient strategies for load distribution among processors, self-balancing techniques, speedup performances for larger number of partitions and processors, and results on more complex configurations are the subjects of a separate report[10].

IV. Concluding Remarks

A new method of domain decomposition has been developed and implemented in the grid generation code VGRID for generating tetrahedral unstructured grids in subdomains. In addition, a new function for estimating the center of mesh density is introduced for balancing loads among subdomains. The method benefits from the following salient features.

1) The new domain decomposition technique is compatible with the underlying grid generation method. The partition interfaces are generated as part of the marching process, similar to Advancing-Front, in a natural and consistent way. Therefore, no artificial mesh elements are introduced in the field, and no external anomalies "disturb" the normal process of grid generation. This important feature is especially critical for generating anisotropic (highly stretched) cells as well as grids in narrow gaps and sharp corners where the presence of any artifact would impair the robustness of the grid generation process.
2) The partition interfaces conform to the domain boundaries consistently and automatically because they are constructed over the surface mesh as part of the volume grid generation.
3) The presence of partition interfaces in the domain introduces no irregularities that could, otherwise, degrade the mesh quality. In fact, the quality of the final grid generated in multiple segments is equivalent to that of a similar mesh generated in the original single domain. The component grids merge into the final mesh seamlessly as if the entire grid is generated in one segment.
4) The method is automatic and requires no time-consuming and tedious pre-processing operations to construct the partitions.
5) Since the partition interfaces are generated during (and as part of) the volume grid generation, the decomposition overhead is minimal for parallel grid generation. Except for initial load estimation and local/global renumbering, no additional operations are required beyond the normal grid generation process.
6) The method is extendible for generating "Navier-Stokes" grids using the Advancing-Layers method.

Future work will include full implementation of the method in the context of an efficient parallel grid generation framework, decomposition/parallelization of surface mesh generation, improvements to the load estimation technique, and extension of the methodology for generating thin-layered "viscous" grids suitable for solving the Navier-Stokes equations.

Acknowledgments

This work is supported by the NASA Fundamental Aeronautics Program, Subsonic Fixed Wing project. The authors would like to thank Dr. Richard A. Wahls (Project Scientist) for his encouragement and support during the course of this study.

References

[1]Löhner, R.; and J.R. Cebral, J.R.: "Parallel Advancing Front Grid Generation", *Proceedings, 8th International Meshing Roundtable,* 67-74, 1999.

[2]Löhner, R.: "Parallel Advancing Front Grid Generation Scheme, ", AIAA-00-1005, 2000.

[3]de Cougny, H.L.; and Shephard, M.S.: "Parallel volume meshing using face removals and hierarchical repartitioning," *Computer Methods in Applied Mechanics and Engineering*, 174, 1999, 275-298.

[4]Larwood, B.G.; Weatherill, N.P.; Hassan, O.; and Morgan, K.: "Domain decomposition approach for parallel unstructured mesh generation", *International Journal for Numerical Methods in Engineering*, 58, 2003, 177-188.

[5]Ito, Y.; Shih, A.M.; Erukala, A.K.; Soni, B.K.; Chernikov, A.; Chrisochoides, N.P.; and Nakahashi, K.: "Parallel Unstructured Mesh Generation by an Advancing Front Method," Mathematics and Computers in Engineering, 2005.

[6]Ivanov, E.; Andrä, H.; and Kudryavtsev, A.: “Domain Decomposition Approach for Automatic Parallel Generation of Tetrahedral Grids,” Fraunhofer-Institut fur Techno- und Wirtschaftsmathematik ITWM, Bericht 87, 2006, ISSN 1434-9973.

[7]Pirzadeh, S.Z.: "Domain Decomposition By the Advancing-Partition Method,” NASA/TM-2008-215350.

[8]Pirzadeh, S.Z.: "Robust Unstructured Grid Generation with VGRID," Proc. of NASA Workshop on Unstructured Grid Generation Techniques and Software, " NASA CP 10119, 1993.

[9]Pirzadeh, S.Z.: "Progress Toward A User-Oriented Unstructured Viscous Grid Generator," AIAA-96-0031, 34th Aerospace Sciences Meeting & Exhibit, Reno, NV, 1996.

[10]Zagaris, G.; Pirzadeh, S.Z.; and Chrisochoides, N.P.: "A Framework for Parallel Unstructured Grid Generation for Practical Aerodynamic Simulations,” AIAA-2009-0980.

[11]Löhner, R.; and Parikh, P.: “Generation of Three-Dimensional Unstructured Grids by the Advancing Front Method,” *International Journal for Numerical Methods in Engineering,* 8, 1988, 1135-1149.

[12]Pirzadeh, S.Z.: "Three-Dimensional Unstructured Viscous Grids by the Advancing -Layers Method," *AIAA Journal*, Vol. 34, No. 1, 1996, pp. 43-49.

[13]Pirzadeh, S.Z.: “Structured Background Grids for Generation of Unstructured Grids by Advancing-Front Method,” *AIAA Journal*, Vol. 31, No. 2, 1993, pp. 257-265.

[14]Pirzadeh, S.Z.: "Advanced Unstructured Grid Generation for Complex Aerodynamics Applications,” AIAA-2008-7178, 26th AIAA Applied Aerodynamics Conference, Honolulu, Hawaii 18-21 August 2008.

[15]Pirzadeh, S.Z.: "Recent Progress in Unstructured Grid Generation,” AIAA-92-0445, AIAA 30th Aerospace Sciences Meeting, Reno, NV, January 6-9, 1992.