Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 00 (2015) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

24th International Meshing Roundtable (IMR24)

Parallel Two-Dimensional Unstructured Anisotropic Delaunay Mesh Generation for Aerospace Applications

Juliette Pardue, Nikos Chrisochoides, and Andrey Chernikov

Department of Computer Science, Old Dominion University, Norfolk, VA, USA

Abstract

A bottom-up approach to parallel anisotropic mesh generation is presented by building a mesh generator from the principles of point-insertion, triangulation, and Delaunay refinement. Applications focusing on high-lift design or dynamic stall, or numerical methods and modeling test cases use two-dimensional domains. Our push-button parallel mesh generation approach, meaning the user only needs to start the program by specifying the initial geometry, anisotropic gradation, and ray angle constraint, can generate high-fidelity unstructured meshes with anisotropic boundary layers for use in the computational fluid dynamics field.

Keywords: mesh generation; parallel; anisotropic; delaunay

1. Introduction

Mesh generation is a step in the pipeline for designing aerospace structures, see Fig. 1. After an analysis of the partial differential equations (PDE) solution on the mesh, the mesh is refined to yield a more favorable error estimation. The refinement process aims to incrementally add more resolution to poor-quality regions of the mesh, with respect to the PDE. For executing a pipeline of tasks, Amdahl's law is considered, which states that the speedup of a program is limited by the sequential fraction of the program. Consider a program where 25% of the program needs to be performed sequentially. If the remaining 75% of the program is parallelized yielding infinite speedup for the 75% portion, then the largest overall speedup we would be able to achieve is four.

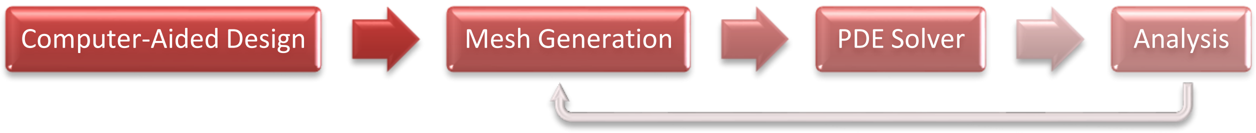


Fig. 1. Development pipeline

Parallel mesh generators by Lammera and Burghardt [1], Kadow [6], and Globisch [14] exist which handle the isotropic cases do not perform well for parallel anisotropic mesh generation. Isotropic mesh generators which focus on solution-based adaptation create many unnecessary elements where there is a high degree of gradation in the flow velocities in one direction. These anisotropic gradations in the flow velocities require anisotropic elements, typically with a 10,000:1 aspect ratio, so representing these regions with isotropic elements incurs a 10,000 fold increase in the number of elements. Chrisochoides and Nave [4] and Ito et al. [13] have applied efforts to generating meshes in parallel for the uniform isotropic and graded isotropic case. Zagaris et al. [10, 11], in parallel, and Loseille et al. [8] and Zhang et al. [12] sequentially, have begun developing anisotropic mesh generation paradigms to facilitate computational fluid dynamics (CFD) simulations.

In this paper, we present an algorithm which: generates a high-fidelity, anisotropic boundary layer mesh in parallel based on a user-defined growth function; generates a globally Delaunay, graded, isotropic inviscid region mesh in parallel; and resolves potential interpolation errors in the boundary layer caused by the local density of the mesh. The examples presented are for the NACA 0012 Airfoil, which is used to illustrate the fundamentals of the algorithm. These fundamentals hold for multi-element and concave airfoils. The anisotropic Delaunay kernel is not needed to generate the pseudo-structured boundary layer because of the extrusion-based point insertion paradigm where alignment and orthogonality of the highly stretched elements is implicit. The extension of the projection-based partitioning algorithm to three-dimensional domains holds for the property that each element of the Delaunay triangulation/tetrahedralization in a d-dimensional space corresponds to a facet of the convex hull of the projection of the points onto a (d+1)-dimensional paraboloid.

2. Algorithm

Efficient meshes for aerospace applications are comprised of two different mesh types: a pseudo-structured anisotropic boundary layer and an unstructured isotropic inviscid region. The pseudo-structured anisotropic boundary layer is generated through an extrusion-based method, presented by Aubry et al. [9]. The unstructured isotropic inviscid region is generated using a graded decoupled approach presented by Linardakis and Chrisochoides [5] along with Delaunay refinement.

2.1. Anisotropic Boundary Layer

Physical phenomena such as boundary layers in fluid mechanics are anisotropic in nature. There is a high degree of gradation in the flow velocities normal to the surface, thus it is beneficial to discretize the mesh in a way that efficiently captures these anisotropic flow velocities in order to yield substantial CPU savings without compromising accuracy. This dictates that the mesh should be refined in the direction normal to the surface where these strong gradients exist. These characteristics allow for the extrusion-based point insertion along the normal at the surface where each vertex is treated as an endpoint for a ray while the normal is treated as the direction of the ray.

2.1.1. Parallel Anisotropic Boundary Layer Point Insertion

If the angle between two rays is too large, then the distance between vertices of neighboring rays will grow at excessively rapid rates, affecting the density of the mesh, causing interpolation errors when the PDE solution is computed. Refining rays are added where the angle between two rays is larger than the user-defined angle constraint. This process is done in parallel where each thread accesses a shared queue of vertices and computes the normal at the vertex. The angle between the current ray and the forward neighboring vertex’s ray is computed to determine if refining rays need to be added. Fig. 2a shows two large angles easily visible at the trailing edge after the ray-based point insertion, while Fig. 2b includes the augmented points. Rays are checked for intersections with other rays, then each thread accesses a shared queue of rays and calculates the new points along the ray direction with respect to the growth function. New vertices are inserted until the number of vertices for the current ray equals the number of layers desired, or until the intersection point.

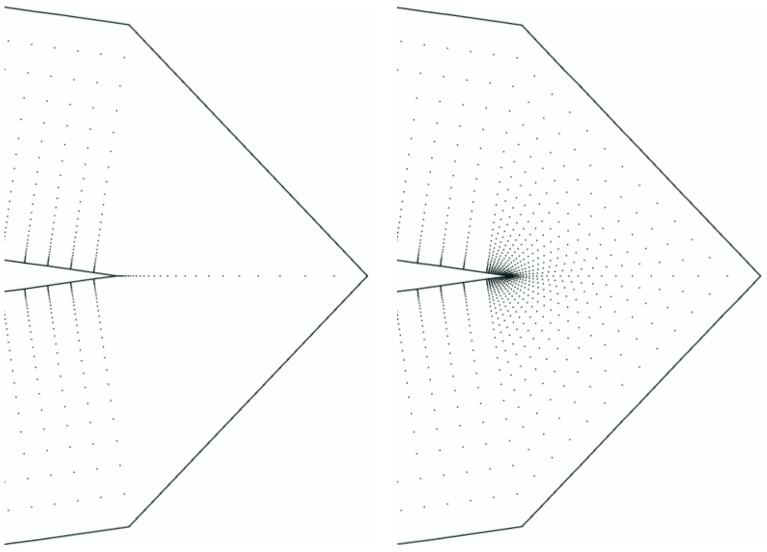


Fig. 2. (a) Trailing edge points for original rays; (b) Trailing edge points with refining rays added

2.1.2. Parallel Triangulation of Boundary Layer

After the point insertion step, the vertices in the boundary layer are triangulated using the algorithm presented by Blelloch et al. [2], which utilizes the duality between the two-dimensional Delaunay triangulation and the three-dimensional lower convex hull of a paraboloid, see Fig. 3a. This approach was chosen because the dividing path created between subdomains corresponds to edges of Delaunay triangles in the final triangulation. Unlike other algorithms [7] which use user-defined dividing paths to arbitrarily partition the domain by introducing undesired, new points which may not have been present in the final triangulation.

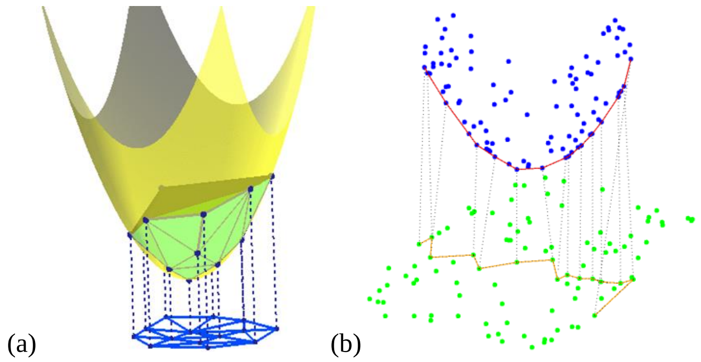


Fig. 3. (a) Paraboloid (yellow), lower convex hull (green) with corresponding Delaunay triangulation (blue), points on Cartesian plane, and projected points on paraboloid; (b) Cartesian point set (green) and Delaunay path (brown) with corresponding flattened projection of the point set (blue) and lower convex hull (red).

Using the median vertex, the vertices are projected onto a paraboloid centered at the median vertex and then flattened onto the vertical plane, see Fig. 3b. The lower convex hull of the vertices on the vertical plane is computed and used to create the dividing path. Two subdomains are created from the original subdomain by partitioning the vertices with respect to the median line. Vertices on the dividing path are added to both subdomains. A subdomain is sufficiently decomposed if: it contains no internal vertices, the number of vertices is less than a tolerance, or the decomposition's recursive level reaches a threshold. Fig. 4 shows the boundary layer decomposed into 64 Delaunay subdomains, which can all be triangulated independently by Triangle [3].

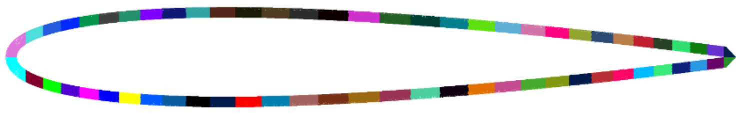
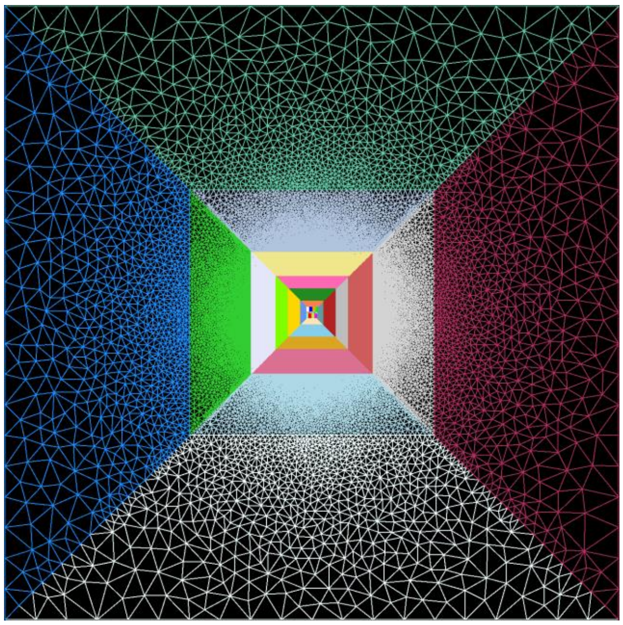


Fig. 4. Decomposed Delaunay subdomains

2.2. Isotropic Inviscid Region

For generating the isotropic inviscid region, we use a sizing function along with Triangle's ability to use a user-defined area constraint for Delaunay refinement to provide a smooth gradation of triangle size. To facilitate the parallel refinement, we generate graded Delaunay decoupling paths, presented by Linardakis and Chrisochoides [5], by computing the value $k = 1/2\,\sqrt{(A/\sqrt{2})}$, where A is the area of the desired element at the given vertex, for each of the four vertices on the corners of a subdomain. Using the k value for each of the four vertices, for each edge, E, comprised of vertex u and vertex v, a graded Delaunay decoupling path is created by discretizing the path from vertex u to vertex v. Assume $k_u \leq k_v$ for the k values at u and v respectively. We choose an integer in the range of $|E|\,/\,(2k_u)$ and $\sqrt{3}|E|\,/\,(2k_v)$ to be the number of segments that edge E will be split into. Fig. 5a shows the decoupled subdomains after Delaunay refinement. Fig. 5b shows the smooth gradation of elements from the outer layer of the boundary layer to the near-body inviscid region.



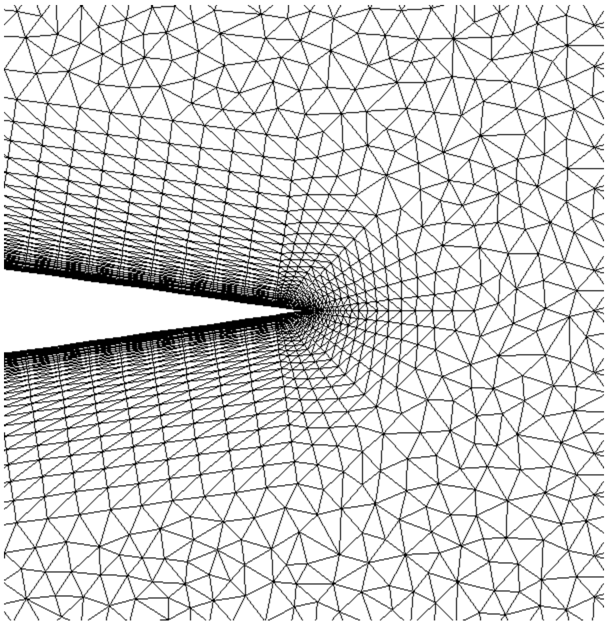


Fig. 5. (a) Inviscid region; (b) Boundary layer to near-body inviscid region

3. Evaluation

We executed our application on two machines: a Concurrent-Read Exclusive-Write Parallel Random Access Machine (CREW PRAM) with eight Intel Xeon CPU E7-4830 at 2.13 GHz processors and 32 GB of memory, and a Cache-Coherent Non-Uniform Memory Access (CC-NUMA) machine with eight Intel Xeon CPU X5560 at 2.80 GHz processors and 16 GB of memory. We measured the speed up, the ratio of the execution time of the fastest sequential algorithm, Triangle, to the execution time of the parallel algorithm; and efficiency, the ratio of speedup to the number of threads used. The execution times do not include input and output times. Our application performed favorably with an efficiency of 90% for four threads, see Fig. 6b. The max speedup we achieved is six for eight threads on the PRAM architecture, see Fig. 6a.

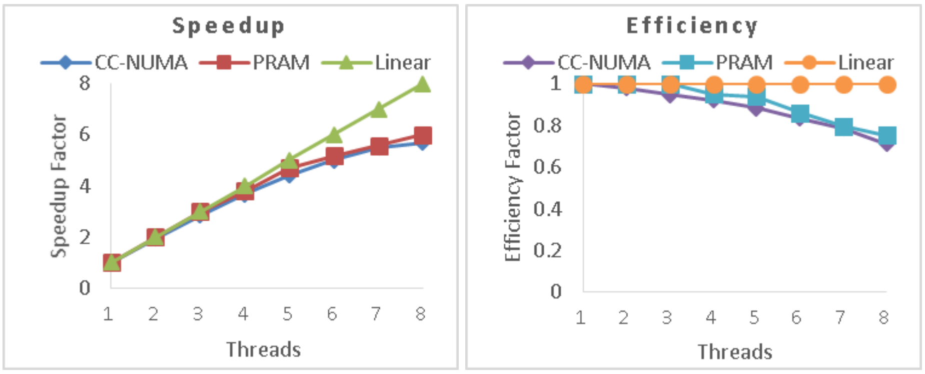


Fig. 6. (a) Speedup compared to Triangle; (b) Efficiency compared to Triangle

4. Conclusion

We have developed a practical approach to generate high-quality, two-dimensional unstructured meshes comprised of an anisotropic, high-fidelity boundary layer and decoupled inviscid region in parallel for use in CFD simulations on shared-memory machines. We plan to extend our approach to handle multi-element airfoils and to generate high-quality three-dimensional meshes in parallel.

Acknowledgements

This work was supported (in part) by the National Institute of Aerospace (http://www.nianet.org) and by the NSF grant CCF-1439079. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIA or the NSF.

References

[1] L. Lammera and M. Burghardt, Parallel Generation of Triangular and Quadrilateral Meshes, Advances in Engineering Software. 31 (2000) 929-936.
[2] G.E. Blelloch, G.L. Miller, D. Talmor, Developing a Practical Projection-Based Parallel Delaunay Algorithm, 12th Annual Symposium on Computational Geometry. (1996) 186-195.
[3] J.R. Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, Applied Computational Geometry: Towards Geometric Engineering. 1148 (1996) 203-222.
[4] N. Chrisochoides, D. Nave, Parallel Delaunay Mesh Generation Kernel, International Journal for Numerical Methods in Engineering. 58 (2003) 161-176.
[5] L. Linardakis, N. Chrisochoides, Graded Delaunay Decoupling Method for Parallel Guaranteed Quality Planar Mesh Generation, SIAM Journal on Scientific Computing. 30 (2008) 1875-1891.
[6] C. Kadow, Parallel Delaunay Refinement Mesh Generation, Ph.D. thesis, Carnegie Mellon University. (2004).
[7] L.P. Chew, N. Chrisochoides, F. Sukup, Parallel Constrained Delaunay Meshing, Symposium on Trends in Unstructured Mesh Generation. (1997) 89-96.
[8] A. Loseille, D. Marcum, F. Alauzet, Alignement and Orthogonality in Anisotropic Metric-Based Mesh Adaption, 53rd AIAA Computational Fluid Dynamics Conference. (2015).
[9] R. Aubry, K. Karamete, E. Mestreau, D. Gayman, S. Dey, Ensuring a Smooth Transition from Semi-Structured Surface Boundary Layer Mesh to Fully Unstructured Anisotropic Surface Mesh, 53rd AIAA Computational Fluid Dynamics Conference. (2015).
[10] G. Zagaris, S. Pirzadeh, A. Chernikov, N. Chrisochoides, Parallel Mesh Generation For CFD Simulations of Complex Real-World Aerodynamic Problems, US National Congress on Computational Mechanics. (2007).
[11] G. Zagaris, S. Pirzadeh, N. Chrisochoides, A Framework for Parallel Unstructured Grid Generation for Practical Aerodynamic Simulations, 47th AIAA Aerospace Sciences Meeting. (2009).
[12] R. Zhang, K.P. Lam, Y. Zhang, Conformal and Adaptive Hexahedral-Dominant Mesh Generation for CFD Simulation of Architecture Applications, Winter Simulation Conference. (2011).
[13] Y. Ito, A. Shih, A. Erukala, B. Soni, A. Chernikov, N. Chrisochoides, K. Nakahashi, Parallel Unstructured Mesh Generation Using an Advancing Front Method, Mathematics and Computers in Simulation. 75 (2007) 200-209.
[14] G. Globisch, PARMESH – A Parallel Mesh Generator, Parallel Computing. 21 (1995) 509-524.