

A PARALLEL ADVANCING FRONT GRID GENERATION SCHEME

Rainald Löhner

Institute for Computational Science and Informatics
M.S. 4C7, George Mason University
Fairfax, VA 22030-4444, USA

ABSTRACT

A parallel advancing front scheme has been developed. The domain to be gridded is first subdivided spatially using a relatively coarse octree. Boxes are then identified and gridded in parallel. A scheme that resembles closely the advancing front technique on scalar machines is recovered by only considering the boxes of the active front that generate small elements. The procedure has been implemented on the SGI Origin class of machines using the shared memory paradigm. Timings for a variety of cases show speedups similar to those obtained for flow codes. The procedure has been used to generate grids in excess of a hundred million elements.

Keywords. Unstructured Grid Generation, Parallel Computing, CFD.

1. INTRODUCTION

The widespread availability of parallel machines with large memory, solvers that can harness the power of these machines, and the desire to model in ever increasing detail geometrical and physical features has lead to a steady increase in the number of points used in field solvers. Grids in excess of 10^7 elements have become common for production runs in Computational Fluid Dynamics (CFD) [Bau93, Bau95, Jou98, Yos98, Mav99] and Computational Electromagnetics (CEM) [Dar97, Mor97]. The expectation is that in the near future grids in excess of $10^8 - 10^9$ elements will be required. While many solvers have been ported to parallel machines, grid generators have lagged behind. For applications where remeshing is an integral part of simulations, e.g. problems with moving bodies [Löh90, Mes93, Mes95, Bau96, Kam96, Löh98a, Has98] or changing topologies [Bau98, Bau99], the time required for mesh regeneration can easily consume more than 50% of the total time required to solve the problem. Faced with this situation, a number of efforts have been reported on parallel grid generation [Löh92, dCo94, Sho95, dCo95, Oku96, Che97, Oku97, Sai99].

The two most common ways of generating unstructured grids are the Advancing Front Technique (AFT) [Per87, Per88, Löh88a,b, Per90, Per92, Jin93, Fry94, Löh96] and the Generalized Delaunay Triangulation (GDT) [Bak89, Geo91, Wea92, Wea94, Mar95]. The AFT introduces one element at a time, while the

GDT introduces a new point at a time. Thus, both of these techniques are, in principle, scalar by nature, with a large variation in the number of operations required to introduce a new element or point. While coding and data structures may influence the scalar speed of the ‘core’ AFT or GDT, one often finds that for large-scale applications, the evaluation of the desired element size and shape in space, given by background grids, sources or other means [Löh96] consumes the largest fraction of the total grid generation time. Unstructured grid generators based on the AFT may be parallelized by invoking distance arguments, i.e. the introduction of a new element only affects (and is affected by) the immediate vicinity. This allows for the introduction of elements in parallel, provided that sufficient distance lies between them.

Several years ago, the author and his colleagues introduced a parallel AFT based on the subdivision of the background grid [Löh92, Sho95]. While used for some demonstration runs, this scheme was not general enough for a production environment. The background grid had to be adapted in order to be sufficiently fine for a balanced workload. As only background grid elements covering the domain to be gridded were allowed, complex in/out tests had to be carried out to remove refined elements lying outside the domain to be gridded. Furthermore, element size specified at CAD entities could not be ‘propagated’ into the domain, as is the case in the scalar AFT, disabling an option favoured by many users and rendering many grid generation data sets unusable. The otherwise positive experience gained with this parallel AFT prompted the search for a more general parallel AFT. The key requirement was a parallel AFT

that changes the current, evolved and mature scalar AFT as little as possible, while achieving significant speedups on common parallel machines. This implies that the parallelism should be applied at the level of the current front, and not globally.

2. PARALLEL GRIDDING SCHEME

The advancing front technique attempts to introduce an element at a time into an as yet ungridded domain by eliminating the face generating the smallest new element from the front [Löh88a,b]. Given that the introduction of an element only affects its immediate neighbourhood, one could, in principle, introduce many elements at the same time, provided they are sufficiently far apart. A convenient way of delimiting the possible zones where elements may be introduced by each processor is via boxes. These boxes may be obtained in a variety of ways, i.e. via bins, binary recursive trees, or octrees. We have found the octree to be the best of these possibilities, particularly for grids with a large variation of element size. In order to recover a parallel gridding procedure that resembles closely the advancing front technique on scalar machines, only the boxes covering the active front in regions where the smallest new elements are being introduced are considered. After these boxes have been filled with elements, the process starts anew: a new octree is built, new boxes are created and meshed in parallel. The procedure is summarized schematically for a 2-D case in Figure 1.

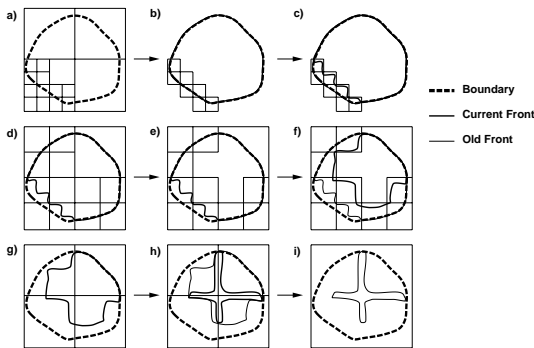


Figure 1 Parallel Grid Generation

At the end of each parallel gridding pass, each one of the boxes gridded can have an internal boundary of faces. For a large number of boxes, this could result in a very large number of faces for the active front. This problem can be avoided by shifting the boxes slightly, and then regridding them again in parallel, as shown in Figure 2. This simple technique has the effect of eliminating almost all of the faces between boxes with a minor modification of the basic parallel gridding algorithm.

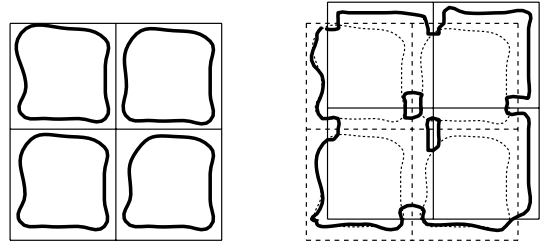


Figure 2 Shift and Regrid Technique

If we define as d_{min} the minimum element size in the active front, and as s_{min} the minimum box size in which elements are to be generated, the parallel AFT proceeds as follows:

```

WHILE: There are active faces left:
  - Form an octree with minimum octant size  $s_{min}$ 
    for the active points;
  - Retain the octants that have faces that will gen-
    erate elements of size  $d_{min}$  to  $c_l \cdot d_{min}$ ;
  - If too many octants are left: agglomerate them
    into boxes;
  - DO ISHFT=0, 2:
    - IF: ISHFT.NE.0:
      Shift the boxes by a preset amount;
    - ENDIF
    - Generate, in parallel, elements in these
      boxes, allowing only elements up to a size
      of  $c_l \cdot d_{min}$ ;
  - ENDDO
  - Increase  $d_{min} = 1.5 * d_{min}$ ,  $s_{min} = 1.5 * s_{min}$ ;
ENDWHILE
  
```

The increase factor allowed is typically in the range $c_l = 1.5 - 2.0$. The shift vectors are given by

$$s = \delta_s(1, 1, 1), \delta_s = \begin{matrix} + \\ - \end{matrix} \min(0.5 * s_{min}, 2.0 * d_{min}) .$$

We remark that the octree used to compute the boxes for parallel grid generation is very coarse compared to the element size specified by the user. The edge-length of the finest octree box is of the order of 20 to 50 times the specified element size. This implies that its construction is very fast, and can be accomplished on a single processor without discernable CPU penalty.

3. WORK ESTIMATION AND BALANCE

The procedure outlined above will work optimally if each box requires approximately the same CPU time to complete its grid. This implies that a good work estimate should be provided. Given that the boxes are not body conforming, even for uniform grids the volume to be gridded can vary drastically from box to box. A balanced workload can be obtained by starting

with many boxes, estimating the work to be done for each of them, and then gridding in parallel **groups of boxes** with similar workload.

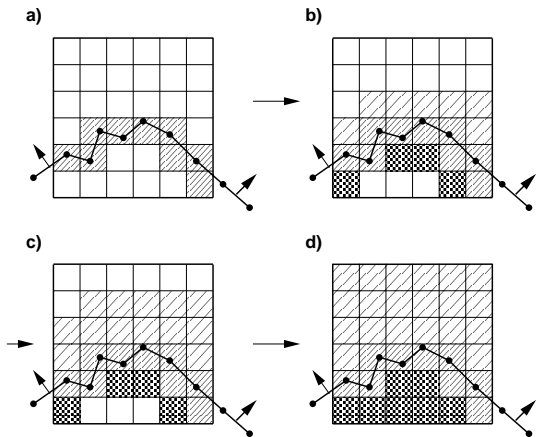


Figure 3 Estimation of Volume to be Gridded

The volume to be gridded is estimated by the marching cubes procedure shown schematically in Figure 3. Given the dimensions of the box, and the list of active faces, the box is first subdivided into voxels (i.e. small cubes). In a first pass over the faces, the voxels cut by faces are marked as ‘inside the domain’, and an average normal is computed for each cut voxel. This normal information is used in a subsequent pass over the voxels in order to mark the neighbours of cut voxels as either inside or outside the domain to be gridded. The remaining voxels are then marked as inside or outside in several sweeps over the voxels. These sweeps are carried out until no further voxels can be marked. Finally, a work estimate is obtained by summing the expected number of elements in each of the voxels marked as inside the domain to be gridded. This work estimation procedure is done in parallel.

Given the estimated work in each of the boxes, the load is balanced in such a way that each processor receives a similar amount of work. The assumption is made that the number of boxes is always larger than the number of processors. Should this not be the case, the boxes are subdivided further (8 new boxes for each box). If any given box has a work estimate that lies above the average work per processor, this box is also subdivided further. This ‘greedy’ work balancing algorithm may be summarized as follows:

a) Obtain a minimum nr. of active boxes:

WHILE: The number of boxes is smaller than the number of processors:

- Subdivide boxes (1:8);

ENDWHILE

b) Balance the work:

WHILE: Work unbalanced

- Estimate, in parallel, the work for each box;
- Obtain average work per processor;
- **IF:** A box has a work estimate above average: Subdivide it further (1:8) and balance again;
- **ENDIF**
- Attempt to group boxes in such a way that the work in each group is close to average;
- **IF:** Good work balance impossible: Subdivide boxes with highest work estimate (1:8) and balance again;
- **ENDIF**

ENDWHILE

In many instances, boxes within a group will share a common face. In order to avoid the (unnecessary) buildup of many faces at the borders of boxes, an attempt is made to agglomerate these neighbouring boxes within groups. This is done recursively by checking, for any pair of boxes, if two of the dimensions are the same and the boxes are coincident in the remaining dimension. If so, the boxes are merged. Boxes are merged recursively, until no pair of boxes passes the test outlined above.

4. MESH IMPROVEMENT

After the generation of the mesh using the parallel advancing front technique has been completed, the mesh quality is improved by a combination of several algorithms, such as:

- Diagonal swapping,
- Removal of Bad Elements, and
- Laplacian smoothing.

4.1 Diagonal Swapping

Diagonal swapping attempts to improve the quality of the mesh by reconnecting locally the points in a different way. Examples of possible 3-D swaps are shown in Figures 4,5.

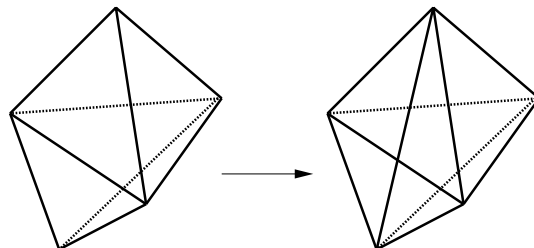


Figure 4 Diagonal Swap Case 2:3

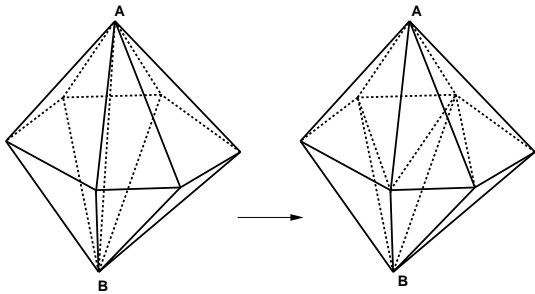


Figure 5 Diagonal Swap Case 6:8

The optimality criterion used is the one proposed by George (1999).

$$Q = \frac{h_{max}S}{V}$$

where h_{max} , S and V denote the maximum edge length, total surface area and volume of a tetrahedron. The number of cases to be tested can grow factorially with the number of elements surrounding an edge. Figure 6 shows the possibilities to be tested for 4,5 and 6 elements surrounding an edge.

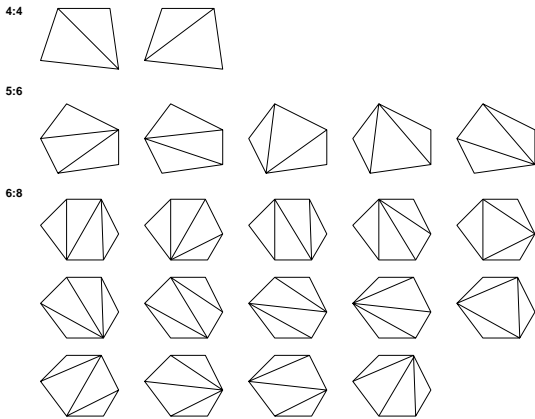


Figure 6 Swapping Cases

Given that these tests are computationally intensive, considerable care is required when coding a fast diagonal swapper. Techniques that were used in the present implementation include:

- Treatment of bad ($Q > Q_{tol}$), untested elements only;
- Processing of elements in an ordered way, starting with the worst (highest chance of reconnection);
- Rejection of bad combinations at the earliest possible indication of worsening quality;
- Marking of tested and unswapped elements in each pass.

As stated before, the computationally intensive part of any diagonal swapping is sifting through all possibilities in order to find a better local point connectivity. One observes that a very large number of tests are

required to obtain an improvement. This implies that parallelizing only the checking portion of a diagonal swapper, which can easily be done as no swapping actually occurs, will yield very good speedups. The parallel swapper can then be summarized as follows:

- **WHILE**: There are bad, untested elements in the heap:
 - Check the worst elements in parallel;
 - Reconnect if possible, and the neighbouring elements have not been reconnected (scalar, integer);
 - Remember swapped elements;
- **ENDWHILE**

4.2 Removal of Bad Elements

A straightforward way to improve a mesh containing bad elements is to get rid of them. For tetrahedral grids this is particularly simple, as the removal of an internal edge does not lead to new element types for the surrounding elements. Once the bad elements have been identified (in parallel), they are compiled into a list and interrogated in turn. An element is removed by collapsing the points of one of the edges, as shown in Figure 7.

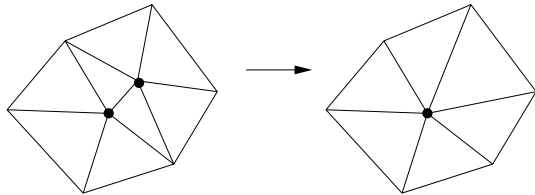


Figure 7 Removal of Element

This operation also removes all the elements that share this edge. It is advisable to make a check which of the points of the edge should be kept: point 1, point 2, or a point somewhere on the edge (e.g. the mid-point). This implies checking all elements that contain either point 1 or point 2. This procedure of removing bad elements is simple to implement and relatively fast. On the other hand, it will not tend to improve the overall quality of the mesh. It is therefore used mainly in a pre-smoothing or pre-optimization stage, where its main function is to eradicate elements of very bad quality from the mesh.

4.3 Laplacian Smoothing

A number of smoothing techniques are lumped under this name. The edges of the triangulation are assumed to represent springs. These springs are relaxed in time using an explicit time stepping scheme, until an equilibrium of spring-forces has been established. Because ‘globally’ the variations of element size and shape are smooth, most of the non-equilibrium forces are local in nature. This implies that a significant improvement in mesh quality can be achieved rather

quickly. The force exerted by each spring is proportional to its length and along its direction. Therefore, the sum of the forces exerted by all springs surrounding a point can be written as:

$$\mathbf{f}_i = c \sum_{j=1}^{ns_i} (\mathbf{x}_j - \mathbf{x}_i) \quad ,$$

where c denotes the spring constant, \mathbf{x}_i the coordinates of the point, and the sum extends over all the points surrounding the point. The time-advancement for the coordinates is accomplished as follows:

$$\Delta \mathbf{x}_i = \Delta t \frac{1}{ns_i} \mathbf{f}_i \quad .$$

At the surface of the computational domain, no movement of points is allowed, i.e. $\Delta \mathbf{x} = 0$. Usually, the timestep (or relaxation parameter) is chosen as $\Delta t = 0.8$, and 5-6 timesteps yield an acceptable mesh. The parallelization of the Laplacian smoothing is similar to that of any other field solver. The loops over the elements/edges are colored so that cash misses are minimized, pipelining does not lead to memory contingencies and cache-line overwrite is avoided [Löh98b]. The application of the Laplacian smoothing technique will yield an overall improvement of grid quality, but can result in inverted or negative elements. These negative elements are eliminated. It has been found advisable to remove not only the negative elements, but also all elements that share points with them. This element removal gives rise to voids or holes in the mesh, which are regridded using the advancing front technique.

5. IMPLEMENTATION

The procedure described above was implemented using the shared memory (i.e. `c$doacross`) paradigm on the SGI Origin 2000. Although this is a distributed-memory machine, it can be programmed as a shared-memory machine. This choice was adopted for the following reasons:

- Coding for a shared-memory environment is much simpler than for a distributed-memory environment. The operating system takes care of most of the inherent message passing, communication conflicts, etc., relieving the user from this task;
- The production codes used in conjunction with the grid generator scale well using the shared-memory paradigm [Löh98b]. Even on 32 processors, more than 50% of the theoretical speedup is achieved. Figure 8 shows the speedups obtained for a steady compressible flow problem using an edge-based, upwind solver on different computer platforms. Note that the speeds obtained using

the shared and distributed memory paradigms are comparable.

- For the last years, all of the large-scale production runs carried out by the author and his colleagues [Löh98a, Löh98c, Bau98, Bau99] were performed on these machines, using the shared-memory paradigm;
- The SGI Origin 2000 has become the dominant platform within the High-Performance Computing sites in the US; at present, there are more SGI Origin 2000 processors that those of all other vendors combined; the expectation is that this trend will not change drastically in the foreseeable future.

While some of the reasons stated above have to do with particular circumstances, the basic ideas of the proposed algorithms are general, and may be coded within a distributed memory framework with explicit message passing.

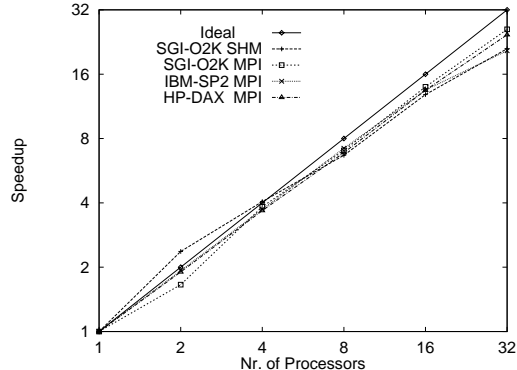


Figure 8 Performance of FEFLO on Different Platforms

6. EXAMPLES

The proposed parallel advancing front scheme has been used extensively over the last year in a production environment. We include a sampling of geometries gridded, highlighting the characteristics of the proposed scheme. The timings for all examples include surface gridding, mesh generation and mesh improvement, i.e. they can be considered representative of the timings obtained in a production environment. All timings were obtained on SGI Origin 2000 servers.

6.1 Cube: This academic example is included here to see how the procedure works in the ‘best case scenario’. The unit cube is to be gridded with a uniform mesh of approximately 1 million tetrahedra. Although this is not a large grid, the timings shown in Figure 9 are illustrative.

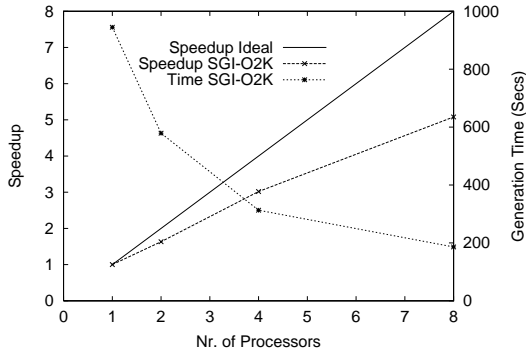


Figure 9 Cube: Speedups Obtained

6.2 Aneurism: This example is taken from a hemodynamic analysis recently conducted for this particular geometry. The outline of the domain, grid, as well as some sample results are shown in Figure 10.1.

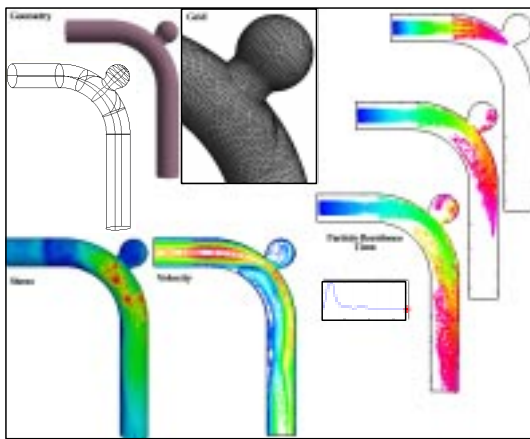


Figure 10.1 Aneurism

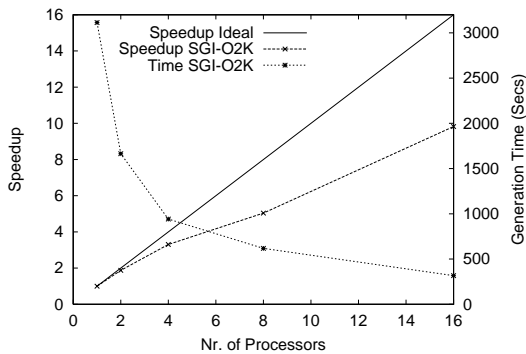


Figure 10.2 Aneurism: Speedups Obtained

The final uniform mesh had approximately 2.7 million tetrahedra. The speedup obtained is shown in Figure 10.2. As one can see, the parallel mesher was more efficient for this finer grid than for the unit cube, even though the volume to be gridded in each box can

vary widely due to geometric features.

6.3 Garage: This example was taken from a blast simulation recently carried out for an office complex. The outline of the domain is shown in Figure 11.1.

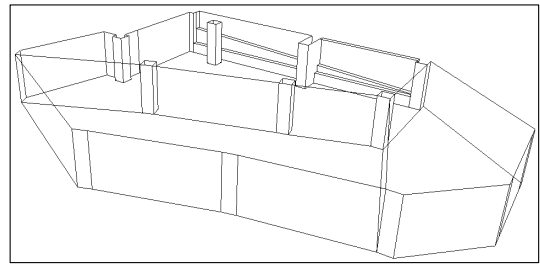


Figure 11.1 Garage: Wireframe

The final uniform mesh had approximately 9.2 million tetrahedra. The speedup obtained is shown in Figure 11.2. As before, the parallel mesher was more efficient for this finer grid than for the unit cube, even though the volume to be gridded in each box can vary widely due to geometric features.

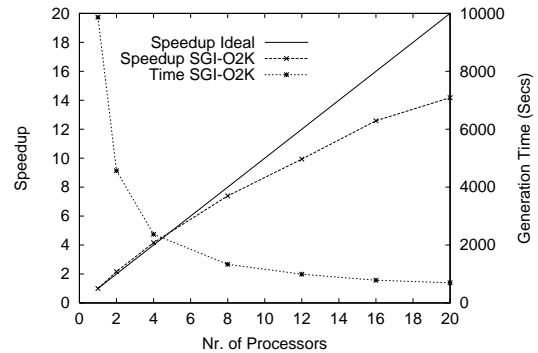


Figure 11.2 Garage: Speedups Obtained

6.4 Tyson's Corner: This example was taken from a dispersion simulation recently carried out for this well-known shopping center. The outline of the domain is shown in Figures 12.1,12.2.

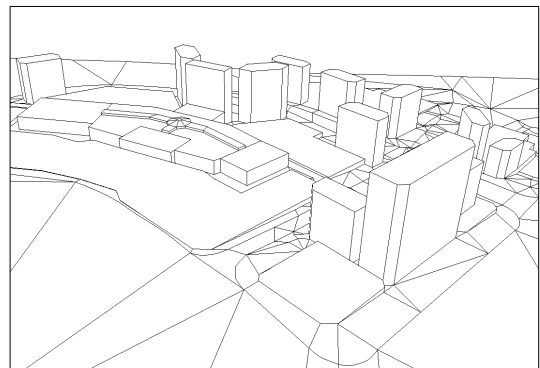


Figure 12.1 Tyson's Corner: Wireframe (W)

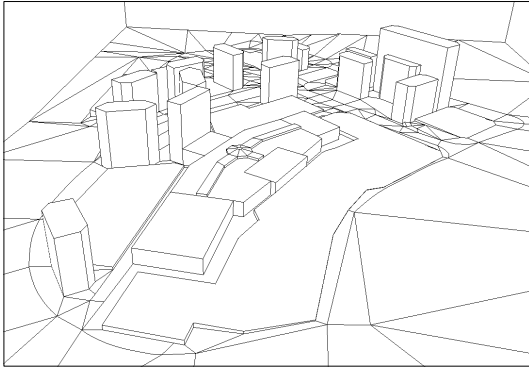


Figure 12.2 Tysons Corner: Wireframe (N)

The final mesh had approximately 16 million tetrahedra. The smallest and largest specified element side lengths were 230 cm and 1400 cm respectively. The speedup obtained is shown in Figure 12.3.

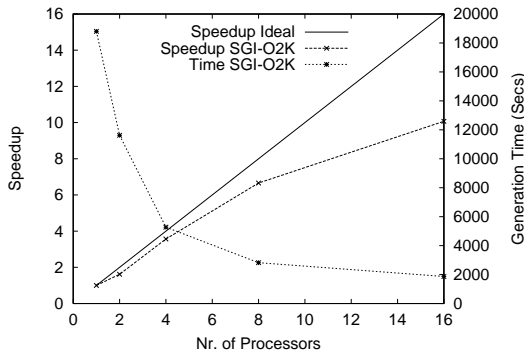


Figure 12.3 Tysons Corner: Speedups Obtained

6.5 Space Shuttle: This example is included here because it is typical of many aerodynamic data sets. The outline of the domain is shown in Figure 13.1.

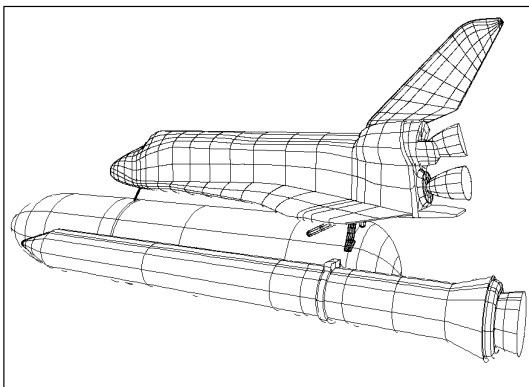


Figure 13.1 Space Shuttle: Outline of Domain

The surface triangulation of the final mesh, which had approximately 4 million tetrahedra, is shown in Figure 13.2. The smallest and largest specified element side lengths were 5.08 cm and 467.00 cm respectively,

i.e. an edge-length ratio of approximately $1 : 10^2$ and a volume ratio of $1 : 10^6$. The spatial variation of element size was specified via approximately 200 sources [Löh96].

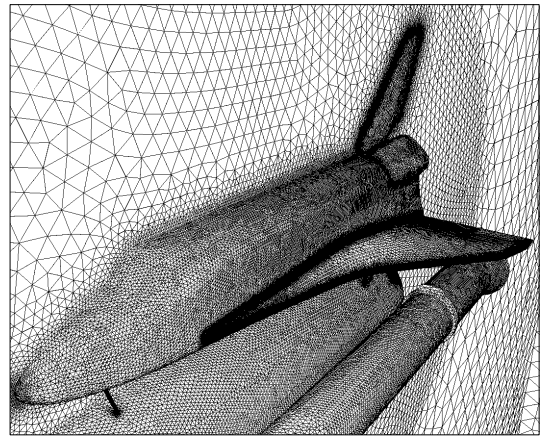


Figure 13.2 Space Shuttle: Surface Mesh

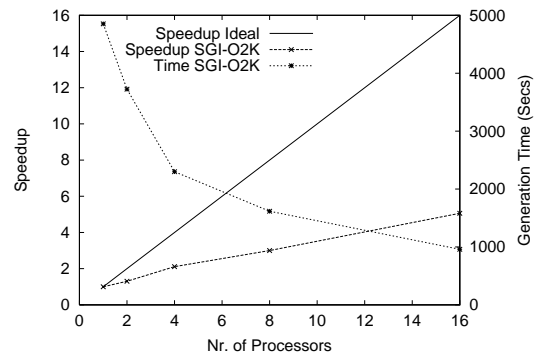


Figure 13.3 Space Shuttle: Speedups Obtained

The speedup obtained is shown in Figure 13.3. Finally, the results of an Euler run for an incoming Mach-nr. of $M = 2.0$ and angle of attack of $\alpha = 1.1^\circ$ are shown in Figure 13.4.

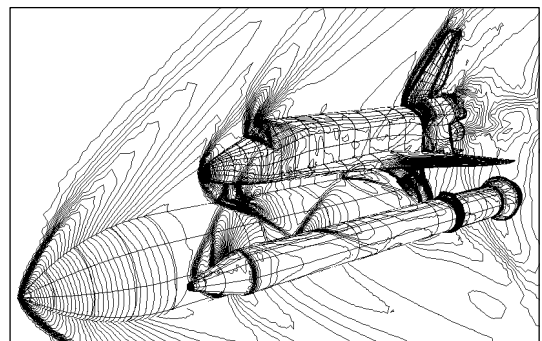


Figure 13.4 Space Shuttle: Surface Pressures

6.6 Pilot Ejecting From F18: Problems with moving bodies often require many remeshings during the course of a simulation. The case shown here was taken from a pilot ejection simulation recently conducted. The outline of the domain is shown in Figure 14.1. The surface triangulation of the final mesh, which had approximately 14 million tetrahedra, is shown in Figure 14.2. The smallest and largest specified element side lengths were 0.65 cm and 250.00 cm respectively, i.e. an edge-length ratio of approximately $1 : 4 \cdot 10^2$ and a volume ratio of $1 : 5.6 \cdot 10^7$. The spatial variation of element size was specified via approximately 110 sources [Löh96].

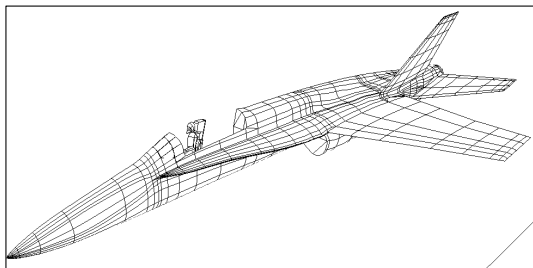


Figure 14.1 F18 Pilot Ejection: Outline of Domain

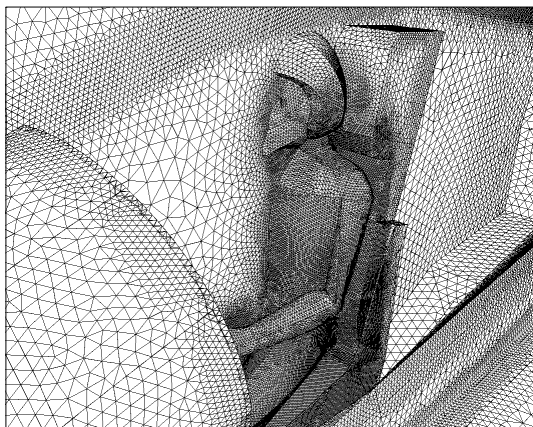


Figure 14.2 F18 Pilot Ejection: Surface Mesh (Closeup)

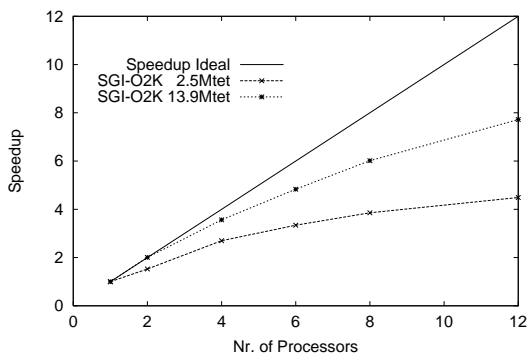


Figure 14.3 F18 Pilot Ejection: Speedups Obtained

The speedup obtained for two different grid sizes is displayed in Figure 14.3.

As could be seen from the previous examples, the grid generator certainly scales well with the number of processors. As with many other areas where parallel processing is being attempted, scalability improves with the amount of work required. The larger the grids, the better the scalability. One can also observe that for each case the ‘scalability slope’ is somewhat different, ranging from almost perfect for the garage to a 1:3 asymptote for the Shuttle. Closer inspection revealed that for the Shuttle, although the final mesh contains 4 million elements, the number of elements created in each parallel pass over increasing element sizes was rather modest, never exceeding 0.5 million elements. Thus, the inherent parallelism of this problem is rather low.

7. CONCLUSIONS AND OUTLOOK

A parallel advancing front scheme has been developed. The domain to be gridded is first subdivided spatially using a relatively coarse octree. Boxes are then identified and gridded in parallel. A scheme that resembles closely the advancing front technique on scalar machines is recovered by only considering the boxes of the active front that generate small elements. The procedure has been implemented on the SGI Origin class of machines using the shared memory paradigm. Timings for a variety of cases show speedups similar to those obtained for flow codes. The procedure has been used to generate grids for a large variety of cases, and is nearing production maturity. Current work is focusing on improved work prediction algorithms, as it is found that in going to a larger number of processors, any small imbalance incurs a heavy CPU penalty.

8. ACKNOWLEDGEMENTS

This research was partially supported by AFOSR, with Dr. Leonidas Sakell as the technical monitor.

9. REFERENCES

- [Bak89] T.J. Baker - Developments and Trends in Three-Dimensional Mesh Generation. *Appl. Num. Math.* **5**, 275-304 (1989).
- [Bau93] J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of a Blast Inside a Boeing 747; *AIAA-93-3091* (1993).
- [Bau95] J.D. Baum, H. Luo and R. Löhner - Numerical Simulation of Blast in the World Trade Center; *AIAA-95-0085* (1995).
- [Bau96] J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman - A Coupled

- Fluid/Structure Modeling of Shock Interaction with a Truck; *AIAA-96-0795* (1996).
- [Bau98] J.D. Baum, H. Luo and R. Löhner - The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies; *AIAA-98-0788* (1998).
- [Bau99] J.D. Baum, H. Luo, E. Mestreau, R. Löhner, D. Pelessone and C. Charman - A Coupled CFD/CSD Methodology for Modeling Weapon Detonation and Fragmentation; *AIAA-99-0794* (1999).
- [Che97] L.P. Chew, N. Chrisochoides and F. Sukup - Parallel Constrained Delaunay Meshing; *Proc. 1997 Workshop on Trends in Unstructured Mesh Generation*, June (1997)
- [dCo94] H.L. de Cougny, M.S. Shephard and C. Ozturan - Parallel Three-Dimensional Mesh Generation; *Computing Systems in Engineering* 5, 311-323 (1994).
- [dCo95] H.L. de Cougny, M.S. Shephard and C. Ozturan - Parallel Three-Dimensional Mesh Generation on Distributed Memory MIMD Computers; *Tech. Rep. SCOREC Rep. # 7*, Rensselaer Polytechnic Institute (1995).
- [Dar97] E. Darve and R. Löhner - Advanced Structured-Unstructured Solver for Electromagnetic Scattering from Multimaterial Objects; *AIAA-97-0863* (1997).
- [Fry94] J. Frykestig - Advancing Front Mesh Generation Techniques with Application to the Finite Element Method; *Pub. 94:10*, Chalmers University of Technology; Göteborg, Sweden (1994).
- [Geo91] P.L. George, F. Hecht and E. Saltel - Automatic Mesh Generator With Specified Boundary; *Comp. Meth. Appl. Mech. Eng.* 92, 269-288 (1991).
- [Geo99] P.L. George - Tet Meshing: Construction, Optimization and Adaptation; *Proc. 8th Int. Meshing Roundtable*, South Lake Tahoe, October (1999).
- [Has98] O. Hassan, L.B. Bayne, K. Morgan and N. P. Weatherill - An Adaptive Unstructured Mesh Method for Transient Flows Involving Moving Boundaries; pp. 662-674 in *Computational Fluid Dynamics '98* (K.D. Papailiou, D. Tsahalis, J. Périaux and D. Knörzer eds.) Wiley (1998).
- [Jin93] H. Jin and R.I. Tanner - Generation of Unstructured Tetrahedral Meshes by the Advancing Front Technique; *Int. J. Num. Meth. Eng.* 36, 1805-1823 (1993).
- [Jou98] W. Jou - Comments on the Feasibility of LES for Commercial Airplane Wings; *AIAA-98-2801* (1998).
- [Kam96] A. Kamoulakos, V. Chen, E. Mestreau and R. Löhner - Finite Element Modelling of Fluid/ Structure Interaction in Explosively Loaded Aircraft Fuselage Panels Using PAMSHOCK/ PAMFLOW Coupling; *Conf. on Spacecraft Structures, Materials and Mechanical Testing*, Noordwijk, The Netherlands, March (1996).
- [Löh88a] R. Löhner - Some Useful Data Structures for the Generation of Unstructured Grids; *Comm. Appl. Num. Meth.* 4, 123-135 (1988).
- [Löh88b] R. Löhner and P. Parikh - Three-Dimensional Grid Generation by the Advancing Front Method; *Int. J. Num. Meth. Fluids* 8, 1135-1149 (1988).
- [Löh90] R. Löhner - Three-Dimensional Fluid-Structure Interaction Using a Finite Element Solver and Adaptive Remeshing; *Comp. Sys. in Eng.* 1, 2-4, 257-272 (1990).
- [Löh92] R. Löhner, J. Camberos and M. Merriam - Parallel Unstructured Grid Generation; *Comp. Meth. Appl. Mech. Eng.* 95, 343-357 (1992).
- [Löh96] R. Löhner - Progress in Grid Generation via the Advancing Front Technique; *Engineering with Computers* 12, 186-210 (1996).
- [Löh98a] R. Löhner, C. Yang, J. Cezral, J.D. Baum, H. Luo, D. Pelessone and C. Charman - Fluid-Structure-Thermal Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; *AIAA-98-2419* [Invited] (1998).
- [Löh98b] R. Löhner - Renumbering Strategies for Unstructured- Grid Solvers Operating on Shared-Memory, Cache- Based Parallel Machines; *Comp. Meth. Appl. Mech. Eng.* 163, 95-109 (1998).
- [Löh98c] R. Löhner, C. Yang and E. Oñate - Viscous Free Surface Hydrodynamics Using Unstructured Grids; *Proc. 22nd Symp. Naval Hydrodynamics*, Washington, D.C., August (1998).
- [Mar95] D.L. Marcum and N.P. Weatherill - Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection; *AIAA J.* 33, 9, 1619-1625 (1995).
- [Mav99] D.J. Mavriplis and S. Pirzadeh - Large-Scale Parallel Unstructured Mesh Computations for 3-D High-Lift Analysis; *ICASE Rep.* 99-9 (1999).
- [Mes93] E. Mestreau, R. Löhner and S. Aita - TGV Tunnel-Entry Simulations Using a Finite Element Code with Automatic Remeshing; *AIAA-93-0890* (1993).
- [Mes95] E. Mestreau and R. Löhner - Airbag Simulation Using Fluid/Structure Coupling; *AIAA-96-0798* (1996).

- [Mor97] K. Morgan, P.J. Brookes, O. Hassan and N.P. Weatherill - Parallel Processing for the Simulation of Problems Involving Scattering of Electro-Magnetic Waves; in *Proc. Symp. Advances in Computational Mechanics* (L. Demkowicz and J.N. Reddy eds.) (1997).
- [Oku96] T. Okusanya and J. Peraire - Parallel Unstructured Mesh Generation; *Proc. 5th Int. Conf. Num. Grid Generation in CFD and Related Fields*, Mississippi, April (1996).
- [Oku97] T. Okusanya and J. Peraire - 3-D Parallel Unstructured Mesh Generation; *Proc. Joint ASME/ASCE/SES Summer Meeting* (1997).
- [Per87] J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz - Adaptive Remeshing for Compressible Flow Computations; *J. Comp. Phys.* 72, 449-466 (1987).
- [Per88] J. Peraire, J. Peiro, L. Formaggia K. Morgan and O.C. Zienkiewicz - Finite Element Euler Calculations in Three Dimensions; *Int. J. Num. Meth. Eng.* 26, 2135-2159 (1988).
- [Per90] J. Peraire, K. Morgan and J. Peiro - Unstructured Finite Element Mesh Generation and Adaptive Procedures for CFD; *AGARD-CP-464*, 18 (1990).
- [Per92] J. Peraire, K. Morgan, and J. Peiro - Adaptive Remeshing in 3-D; *J. Comp. Phys.* (1992).
- [Sai99] R. Said, N.P. Weatherill, K. Morgan and N.A. Verhoeven - Distributed Parallel Delaunay Mesh Generation; to appear in *Comp. Meth. Appl. Mech. Eng.* (1999).
- [Sho95] A. Shostko and R. Löhner - Three-Dimensional Parallel Unstructured Grid Generation; *Int. J. Num. Meth. Eng.* 38, 905-925 (1995).
- [Wea92] N.P. Weatherill - Delaunay Triangulation in Computational Fluid Dynamics; *Comp. Math. Appl.* 24, 5/6, 129-150 (1992).
- [Wea94] N.P. Weatherill and O. Hassan - Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints; *Int. J. Num. Meth. Eng.* 37, 2005-2039 (1994).
- [Yos98] S. Yoshimura, H. Nitta, G. Yagawa and H. Akiba - Parallel Automatic Mesh Generation Method of Ten-Million Nodes Problem Using Fuzzy Knowledge Processing and Computational Geometry; *Proc. 4th World CongComp. Mech.* Buenos Aires, Argentina, July (1998).