

A Grid Coarsening Algorithm for Unstructured Multigrid Applications

Jacob Waltz and Rainald Löhner
Institute for Computational Sciences and Informatics
George Mason University
Fairfax, VA 22031

January 13, 2000

Abstract

An algorithm for the construction of nested coarse tetrahedral meshes given an initial fine tetrahedral mesh is described. This algorithm, termed Dynamic Graph Reduction with Swapping (DGRS), is able to produce nested coarse meshes suitable for unstructured multigrid applications. The capabilities of the DGRS approach are demonstrated through several examples.

1 Introduction

One of the difficulties associated with the use of multigrid methods on unstructured meshes has been the generation of the coarse meshes [2, 6, 7, 8, 9, 10, 17]. On a Cartesian mesh this problem has a rather simple solution: the coarsened mesh can be constructed by removing every other point of the fine mesh. This process can be continued until the desired number of meshes have been obtained.

With fully unstructured meshes, where each point can have an arbitrary number of neighbors and the elements are nonuniform, the problem is much more difficult, particularly in three dimensions. To date, two approaches towards grid coarsening on unstructured meshes have become prevalent: the agglomeration method [11, 12, 17] and disjoint meshing [2, 7, 10, 14]. Automated procedures which produce a series of nested tetrahedral meshes have been developed with limited success. The most promising approach developed to date appears to be the edge-collapsing method developed by M. Giles and others [6, 8]. Point-based methods, however, appear to be at a much lower level of maturity due to their very limited discussion in current literature [2, 9, 13].

The goal of the current work has been to develop an automated grid coarsening procedure suitable for

3D unstructured multigrid applications. The algorithm described herein can produce nested coarse tetrahedral meshes of reasonable quality. Several important features of this algorithm are noted:

- Because the procedure is automated, it can be fully integrated into the flow solver and enabled at run-time. No additional preprocessing time is required by the user.
- The time required to coarsen a fine mesh is significantly less than the time required to generate a coarse mesh “from scratch”. Therefore the grid coarsening procedure can be used in problems which involve transient remeshing.
- Variations in fine mesh size are automatically captured in the coarse mesh.
- Because the coarse meshes are tetrahedral no modifications need to be made to the core flow solver above and beyond the implementation of the actual multigrid scheme.

Perhaps the most important of the above features is the ability to use the algorithm in unsteady problems. To date, the majority of 3D unstructured multigrid applications have involved static meshes.

The remainder of this paper is organized as follows. Section 2 describes the automated grid coarsening procedure. Section 3 provides a sampling of coarse meshes obtained with this procedure. Example multigrid applications are presented in Section 4. Concluding remarks and an outlook for future work are summarized in Section 5.

2 Grid Coarsening Algorithm

The grid coarsening procedure can be broken down into two basic parts: point selection and element reconnection. Each is described below.

2.1 Point Selection

The point selection criterion used in this work is to require the coarse mesh points to form a Maximally Independent Set (MIS) of the fine mesh points. Given a set of simply connected points, a subset of those points forms a maximally independent set iff:

1. No two points in the subset are connected
2. Each point not in the subset is connected to at least one point which is in the subset

When translated into the language of meshes, the above criterion ensure that variations in fine mesh edge size are properly represented in the coarse mesh. Simple drawing exercises will show that violation of either criteria results in artificial variations in element size.

Several approaches can be used to generate an MIS from a given set of points. The method used in this work to generate an MIS is a pseudo-advancing front procedure based on an hierarchy of points. The algorithm can be summarized as follows:

POINT SELECTION ALGORITHM

1. Order the points as corner, line, surface, and volume
 2. Sort each group of points by the volume surrounding each point
 3. **Do** for each corner point `ipoint`
 4. Mark `ipoint` as in the MIS
 5. Mark each neighbor of `ipoint` as not in the MIS
 6. **End Do**
 7. **Do** for each line, surface, and volume point `ipoint`
 8. **If** `ipoint` has not been marked
 9. Mark `ipoint` as in the MIS
 10. Mark each neighbor of `ipoint` as not in the MIS
 11. **Else If** `ipoint` has not been marked
 12. Continue to next point
 13. **End If**
 14. **End Do**
-

Several of the steps in the above algorithm are not needed if only an MIS is desired. However, in this case a procedure which simply generates an MIS is insufficient; the boundaries of the computational domain must be preserved. To facilitate element reconnection a series of marking rules are used. Specifically, no point is allowed to mark a neighboring point which is higher up in the hierarchy. For example, surface points are not allowed to mark line

points. Furthermore, boundary points are allowed to mark other boundary points if and only if both points are of the same type and both points lie on the same boundary segment. For example, surface points may mark other surface points which lie on the same surface. Volume points are allowed to mark only other volume points.

The sorting of points by volume is used strictly to improve the quality of the coarse mesh. This feature has the largest impact towards the end of the point selection process when few unmarked points remain.

2.2 Element Reconnection

As with point selection, numerous approaches are possible for element reconnection. The most efficient approach is dynamic reconnection, *i.e.* reconnection during the point marking procedure. Given a point `ipoint` which is in the MIS, the reconnection procedure consists of the following steps:

ELEMENT RECONNECTION ALGORITHM

1. **Do** for each neighbor `jpoint` marked as not in the MIS
 2. **Do** for each element `ielem` surrounding `jpoint`
 3. Replace `jpoint` with `ipoint` in `ielem`
 4. Verify positive volume
 5. Verify surface consistency
 6. **End Do**
 7. Verify volume surrounding `jpoint`
 8. **If** any checks failed
 9. Disallow reconnection
 10. **End If**
 11. **End Do**
-

The dynamic reconnection allows a substantial improvement in coarsening speed when compared to stand-alone procedures for point marking and element reconnection. At the end of the entire procedure, the list of elements will contain a large portion of collapsed elements. These are elements elements which contain one or more identical points. The collapsed elements and the removed points are filtered out. The points and elements which remain constitute the coarser mesh.

The marking rules described in the previous section as well as the volume/surface checks serve to maintain the boundaries of the domain during the element reconnection procedure. For example, the reconnection of a surface point to a volume point would distort the boundary and therefore must be

disallowed. On curved surfaces the consistency checks must be modified slightly to account for the small changes in volume that occur. An additional check that must be performed in this case is that changes in the direction of the outward normals of the boundary faces must be limited to some prescribed maximum.

2.3 The Full Algorithm

While the algorithm as described works quite elegantly in 2D, the situation becomes much more complex in 3D. Since not every possible reconnection is valid, at the end of the coarsening procedure a small number of unremoved points (points which were marked for removal but could not be removed) will remain. This behavior can be overcome through the use of diagonal swapping for tetrahedron and post-processing improvement passes. The application of diagonal swapping and improvement passes results in the near or total elimination of unremoved points.

The full algorithm, which we refer to as Dynamic Graph Reduction with Swapping (DGRS) begins with marking and reconnection for corner points:

DGRS ALGORITHM: CORNER POINTS

1. Order the points as corner, line, surface, and volume
 2. Sort each group of points by the volume surrounding each point
 3. **Do** for each corner point *ipoint*
 4. Mark *ipoint* as in the MIS
 5. **Do** for each neighbor *jpoint ipoint*
 6. Mark *jpoint* as removed
 7. Attempt element reconnection for *jpoint*
 8. **If** reconnection fails
 9. **Do** for each neighbor *kpoint* of *jpoint*
 10. **If** point types are compatible
 11. Attempt to reconnect *jpoint* to *kpoint*
 12. **End If**
 13. **End Do**
 14. **End If**
 15. **End Do**
 16. **End Do**
-

The second phase is to perform the marking and reconnection of non-corner points:

DGRS ALGORITHM: NON-CORNER POINTS

17. **Do** for each line, surface, and volume point *ipoint*:
 18. **If** *ipoint* is unmarked
 19. Mark *ipoint* as in the MIS
 20. **Do** for each neighbor *jpoint* of *ipoint*
 21. **If** point types are compatible
 22. Attempt element reconnection for *jpoint*
 23. **End If**
 24. **If** reconnection fails
 25. **Do** for each neighbor *kpoint* of *jpoint*
 26. **If** point types are compatible
 27. Attempt to reconnect *jpoint* to *kpoint*
 28. **End If**
 29. **End Do**
 30. **End If**
 31. **End Do**
 32. **Else If** *ipoint* has been marked
 33. Continue to next point
 34. **End If**
 35. **End Do**
 36. Filter out collapsed points and elements
 37. Perform diagonal swapping across entire mesh
-

At this stage in the procedure, all points have been marked either as “to be kept” or “to be removed” and a large number of elements will have been collapsed. As mentioned previously, a number of points will remain which were marked for removal but could not be removed (hereafter referred to simply as unremoved points). Testing indicates that this number is typically 2-5% of the total number of points marked for removal. Reconnections which fail typically do so because either a negative element would have been created or the surface topology would have been altered.

The next phase of the algorithm is the improvement phase. In this phase an attempt is made to remove the *nunremv* points that were not removed. A series of improvement passes are performed until either all offending points are removed or the number of improvement passes reaches a specified maximum *npass*.

```

38.  ipass = 1
39.  Do While ipass < npass Or nunremv > 0:
40.    If ipass > 5
41.      Allow forced swaps
42.    End If
43.    Mark elements for diagonal swapping
44.    Apply diagonal swapping
45.    Do for each unremoved point ipoint
46.      Form list of points surrounding ipoint
47.      Do for each neighbor jpoint of ipoint
48.        If point types are compatible
49.          Attempt to reconnect ipoint
                    to jpoint
50.        If reconnection succeeds
51.          Goto 54
52.        Else
53.          Goto 55
54.        End If
55.      End If
56.    End Do
57.  End Do
58.  Filter out collapsed points and elements
59.  Mark elements for diagonal swapping
60.  Apply diagonal swapping
61.  ipass = ipass + 1
62. End Do
63. Perform diagonal swapping across entire mesh

```

An important observation is that in practice, most unremoved points are eliminated within the first 2-3 improvement passes. Any unremoved points which remain at this stage generally cannot be removed without a new course of action. This behavior is the reason that swaps are forced when the number of improvement passes exceeds 5. Diagonal swaps are forced by allowing any valid swap to occur even if element quality is degraded. Any distorted elements which occur are dealt with in the diagonal swapping at the end of the improvement pass. This approach eliminates unremoved points very effectively. Nonetheless, in practice a small number of points sometimes cannot be removed without an exorbitant amount of effort. This number is typically less than 1% of the total number of points in the coarse mesh.

The most time-consuming portion of this algorithm is the diagonal swapping procedure. At the end of the improvement passes only elements in the region surrounding unremoved points are marked for swapping. If all elements are marked for swapping the computational time increases substantially while the overall mesh quality increases only slightly.

2.4 Diagonal Swapping

The diagonal swapping procedure plays an important role in the grid coarsening algorithm. The obvious benefit of diagonal swapping is that the quality of the coarser meshes is improved. Less obvious but just as important is that the diagonal swapping improves the ability of the algorithm to removed unwanted points. This is accomplished by "mixing up" the mesh and thereby increasing the number of potential reconnections available.

Due to its importance a significant effort was devoted to maximizing the capabilities of the diagonal swapping procedure. The types of diagonal swapping cases used form two general classes which we refer to as face swaps and edge swaps. The general face swap consists of two or more tetrahedron which share a set of faces. These faces are either reconnected (the internal reconnection face swap) or replaced by a shared edge (the face-edge swap). The general edge swap consists of a ring of elements around an edge. The shared edge is removed and replaced with a set of interior faces. This swap, referred to as the edge-face swap, is the direct inverse of the face-edge swap. The total number of cases implemented in the diagonal swapping procedure is 120. Of these cases, 53 were ported from a 3D unstructured grid generation code. The remainder were developed specifically for this work.

The figures below provide a sampling of the types of diagonal swaps implemented in this work.

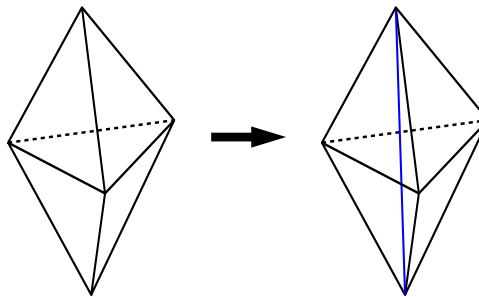


Figure 1: Face-edge swap from 2 to 3 elements.

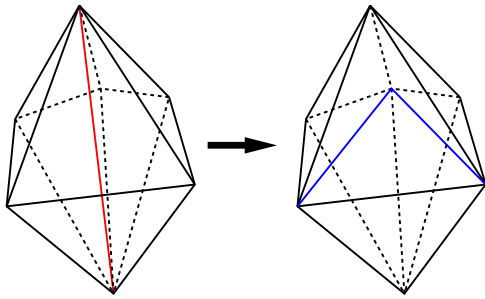


Figure 2: Edge-face swap from 5 to 6 elements.

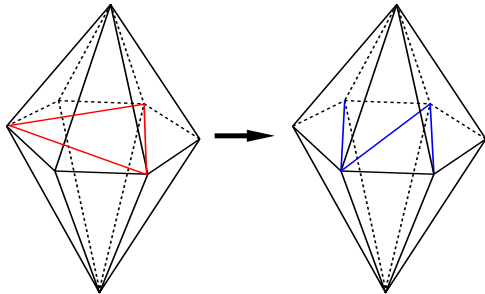


Figure 3: Internal reconnection face swap from 8 to 8 elements.

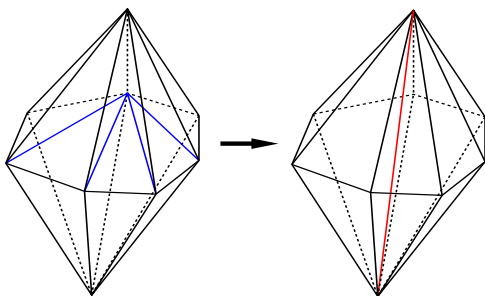


Figure 4: Face-edge swap from 10 to 7 elements.

3.1 Building Geometry

The first test case involves a building geometry used in the simulation of low-speed environmental flows and atmospheric dispersion. Figure 5 shows the initial fine mesh. The two coarse meshes are shown in Figures 6 and 7. Table 1 summarizes parameters for all meshes.

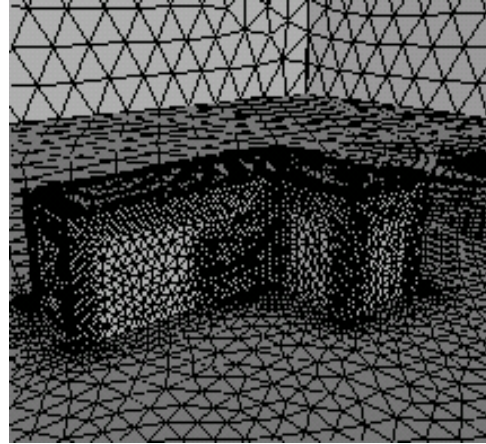


Figure 5: Initial fine mesh for building geometry.

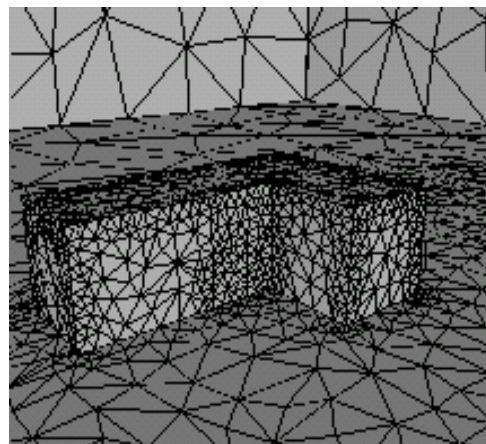


Figure 6: First coarse mesh for building geometry.

3 Coarsening Results

We first present a sample of coarse meshes generated with the aforementioned procedure.

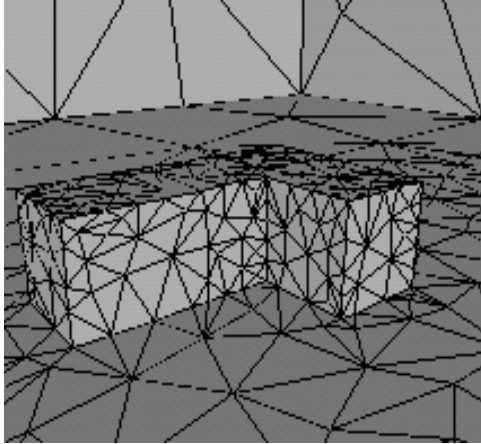


Figure 7: Second coarse mesh for building geometry.

Table 1: Summary of mesh parameters for building test case (`cfact` = coarsening factor).

Mesh	nelem	npoin	cfact
1	276682	51781	–
2	33973	7058	7.34
3	4476	1075	6.57

3.2 Intersecting Cylinders

The second test cases involves two intersecting cylinders. The meshes are shown in Figures 8-10. A summary of mesh parameters is given in Table 2.

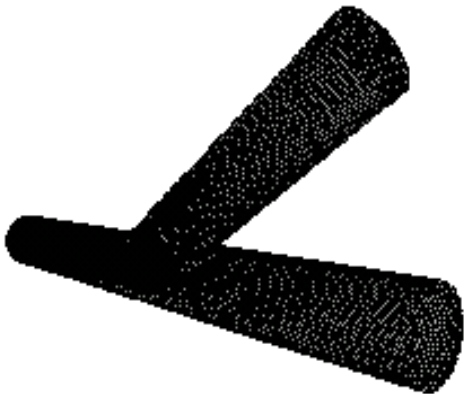


Figure 8: Initial fine mesh for intersecting cylinders.

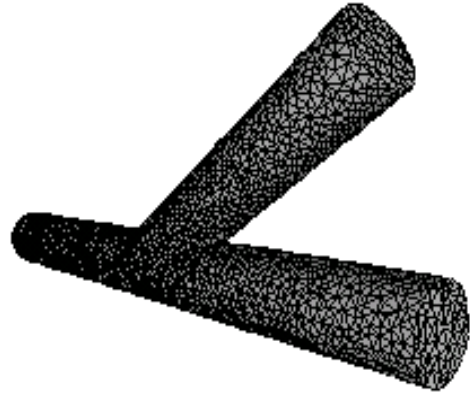


Figure 9: First coarse mesh for intersecting cylinders.

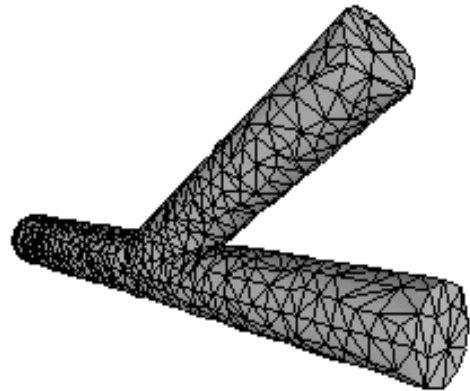


Figure 10: Second coarse mesh for intersecting cylinders.

Table 2: Summary of mesh parameters for intersecting cylinders.

Mesh	nelem	npoin	cfact
1	451104	84437	–
2	52630	11008	7.67
3	6408	1579	6.97

4 Sample Applications

The sample applications were performed using the Coarse Grid Correction scheme [3, 4] with a simple Jacobi smoother [1, 15]. The cycling algorithm used is that of [14], with `icycle = nmesh - 1` in all cases.

4.1 Poisson Equation

The multigrid solver was implemented in a prototype application which solves the Poisson equation in a cubic geometry. This test case is intended to establish the performance of our methodology using a pure Laplacian operator. Four coarse meshes were used in the calculation. The parameters for each mesh are summarized in table 3. The individual meshes are shown in Figures 11-15.

The numerical solution after 1000 multigrid cycles is shown in Figure 16. The convergence history is shown in Figure 17; the log of the residual is plotted as a function of the number of multigrid cycles. To better quantify the performance, the number of multigrid cycles required to reach a fixed level of residual and the relative speed-ups are summarized in 4. The reference level of residual is the value after 1000 steps with a single-grid solver.

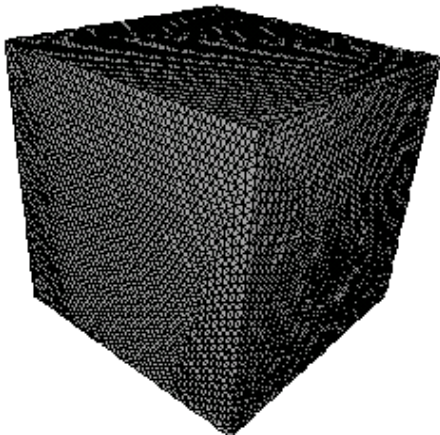


Figure 11: Initial fine mesh for cube geometry.

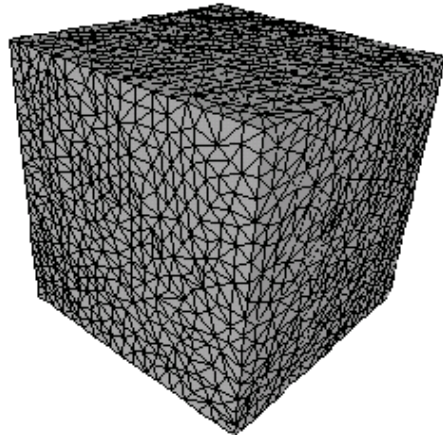


Figure 12: First coarse mesh for cube geometry.

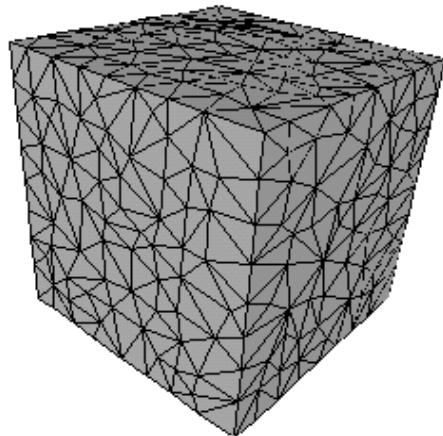


Figure 13: Second coarse mesh for cube geometry.

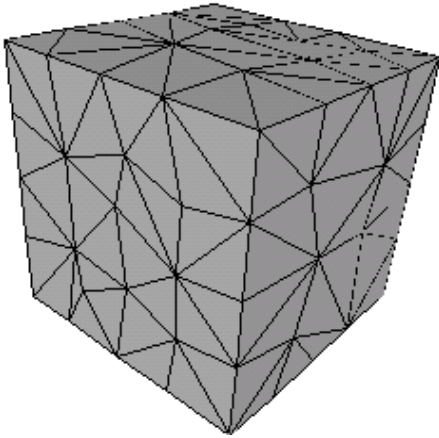


Figure 14: Third coarse mesh for cube geometry.

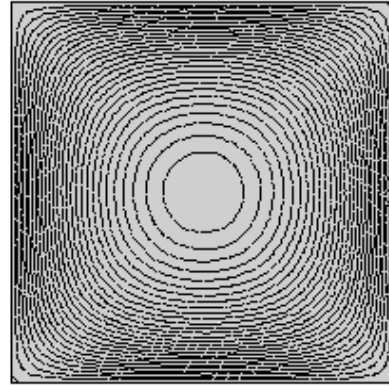


Figure 16: Potential contours for prototype problem.

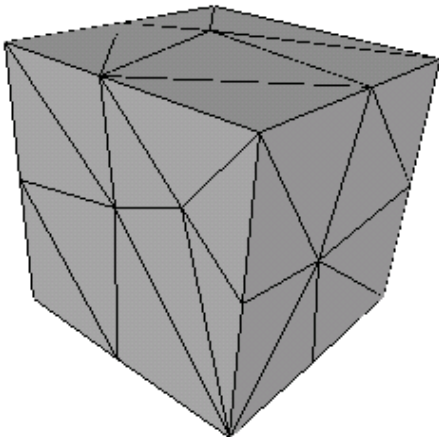


Figure 15: Fourth coarse mesh for cube geometry.

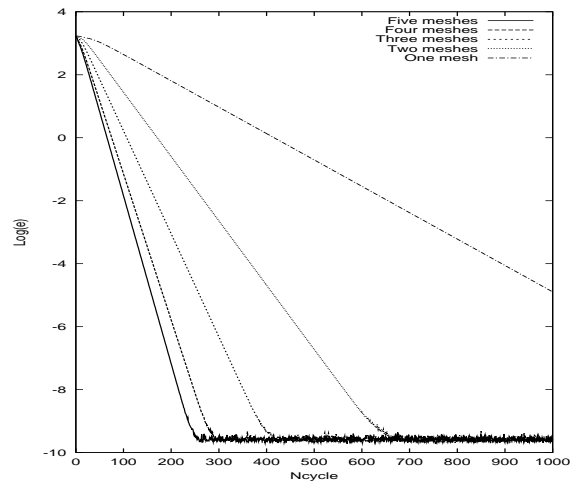


Figure 17: Sample convergence history for the pressure correction.

Table 3: Summary of mesh parameters for prototype application.

Mesh	nelem	npoin	cfact
1	319472	58261	—
2	38074	7496	7.77
3	4563	1012	7.41
4	553	149	6.79
5	78	31	4.81

Table 4: Speed increases for prototype problem.

nmesh	nstep	Speed up
1	1000	—
2	410	2.44
3	256	3.91
4	181	5.52
5	157	6.37

4.2 NACA-0012 Airfoil

The second sample application is the low speed incompressible flow over a NACA-0012 airfoil. The projection method [5, 16] was used to solve the incompressible Euler equations. Multigrid was used to solve the Poisson-like equation for the pressure correction at each time step.

Three meshes were used for this test case. The initial fine mesh is shown in Figure 18. Figures 19 and 20 show the coarse meshes. Mesh parameters are summarized in Table 5. Pressure contours for the steady state solution obtained with the multigrid solver are shown in Figure 21.

A typical convergence history in the calculation of the pressure correction is shown in Figure 22. A decrease of four orders of magnitude in the L_2 norm of the residual vector was specified as the convergence criterion. The speed-up in the convergence is typically a factor of 2.5 with two meshes and 3.5 with three meshes. Note that these values decrease as the solution approaches the steady state. The total speed-up for a run of 250 time steps in terms of clock time is 2.1 for two meshes and 2.7 for three meshes. These values include the overhead associated with the grid coarsening procedure.

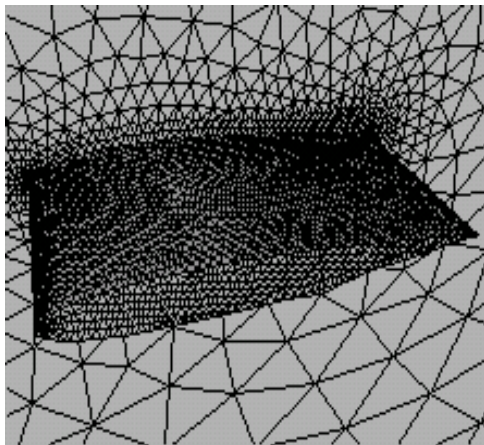


Figure 18: Initial fine mesh for NACA-0012.

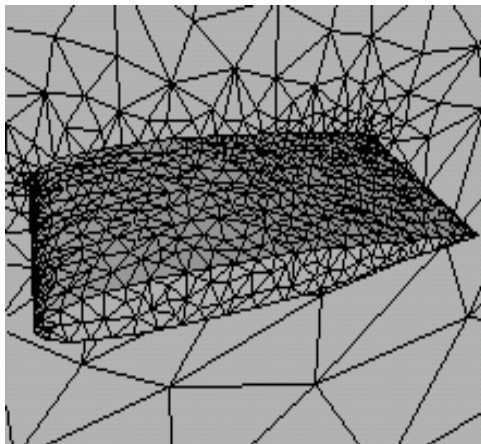


Figure 19: First coarse mesh for NACA-0012.

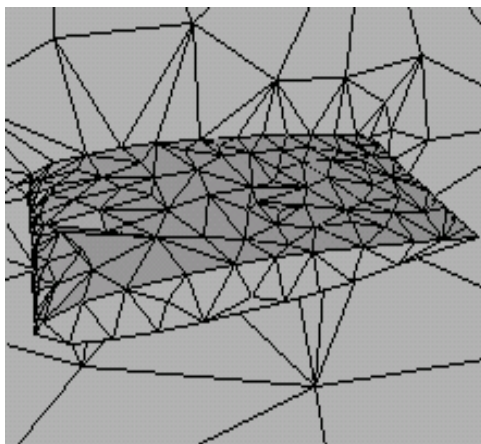


Figure 20: Second coarse mesh for NACA-0012.

Table 5: Summary of mesh parameters for NACA-0012.

Mesh	nelem	npoin	cfact
1	113838	21876	—
2	14784	3208	6.82
3	2050	515	6.23

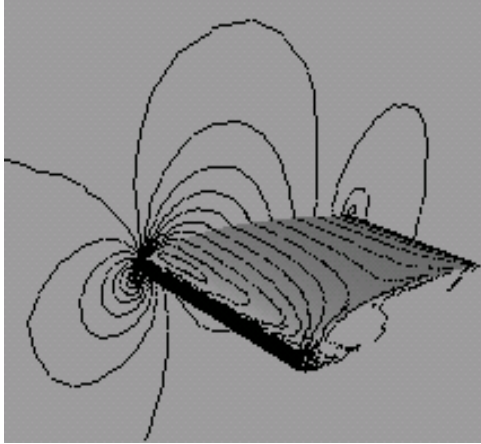


Figure 21: Steady-state pressure contours over NACA-0012.

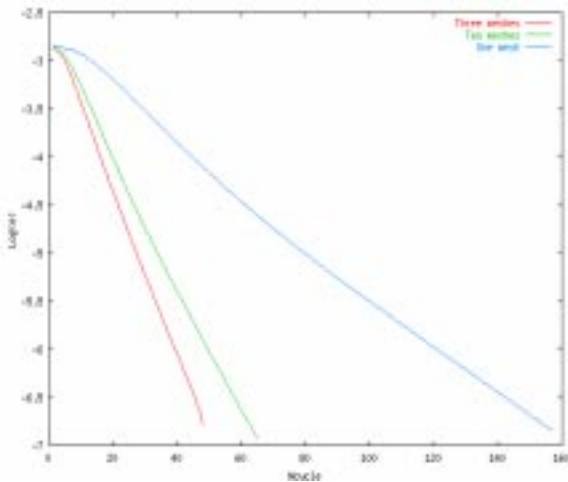


Figure 22: Convergence history for NACA-0012 pressure correction.

4.3 Micro Air Vehicle

The final sample application is the incompressible flow over a Micro Air Vehicle (MAV). For this demonstration case two meshes were used. The initial fine mesh is shown in Figure 23. The coarse mesh is shown in Figure 24. Parameters for both meshes are summarized in Table 6. The steady-state solution is shown in Figure 25.

Figure 26 shows a sample convergence history for the pressure correction. The speed-up for the history

shown is approximately 2.7. As with the NACA-0012 test case the speed-up varies over the course of the simulation. The net speed-up in terms of clock time for this test case is approximately 2.4.

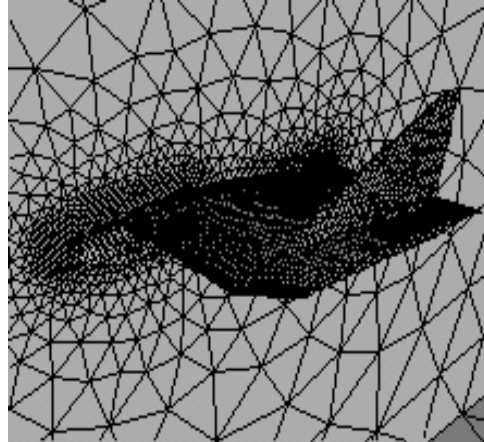


Figure 23: Initial fine mesh for MAV

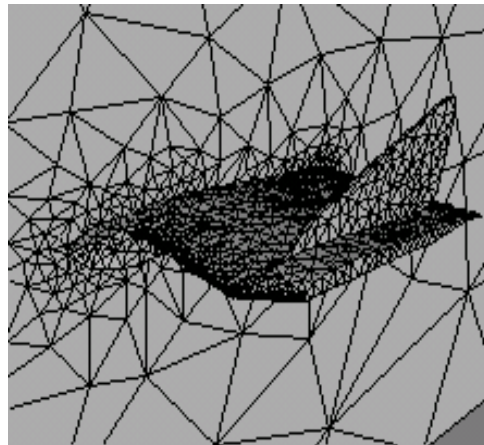


Figure 24: Coarse mesh for MAV.

Table 6: Summary of mesh parameters for MAV.

Mesh	nelem	npoin	cfact
1	231845	45330	-
2	30745	6861	6.61

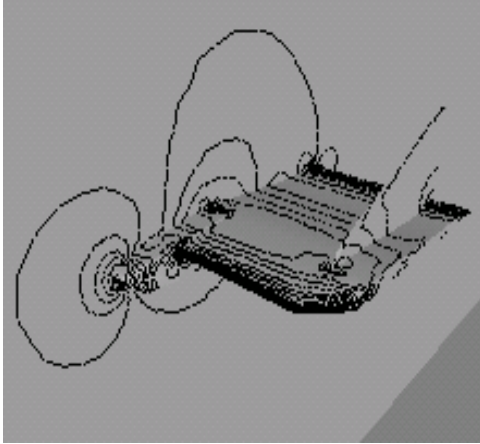


Figure 25: Steady-state pressure contours over MAV.

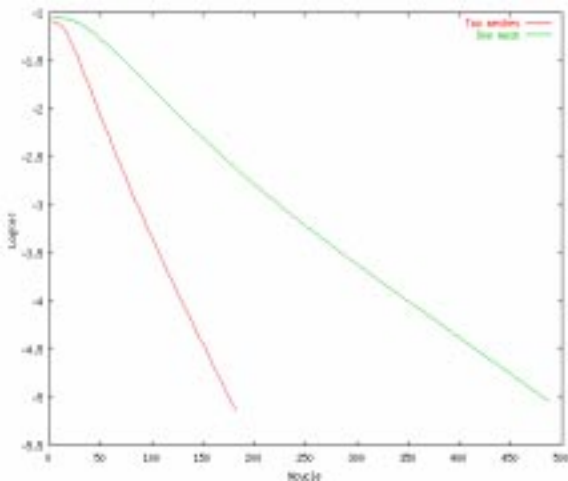


Figure 26: Convergence history for MAV pressure correction.

5 Conclusions

An automated procedure for the generation of coarse tetrahedral meshes has been presented. This DGRS procedure generates coarse tetrahedral meshes of acceptable quality given an initial fine mesh. The capabilities and flexibility of this approach have been demonstrated. Of particular importance is the automated nature of the DGRS algorithm, which eliminates additional preprocessing

and allows the procedure to be fully integrated into the flow solver.

Future work will focus on improvements to the algorithm in the areas of robustness and the ability to handle more complex geometries. Improved efficiency of the diagonal swapping procedure also will be of importance. The next major goal in this work is to apply this methodology to unsteady problems with transient remeshing and moving bodies on parallel architectures.

References

- [1] W.F. Ames, *Numerical Methods for Partial Differential Equations* (Academic Press, London, 1992).
- [2] T.J. Barth, "Randomized Multigrid", *AIAA-95-0207* (1995).
- [3] J.H. Bramble, *Multigrid Methods* (Longman Group, 1993, UK).
- [4] W.L. Briggs, *A Multigrid Tutorial* (Society for Industrial and Applied Mathematics, Philadelphia, 1987).
- [5] C.A.J. Fletcher, *Computational Techniques for Fluid Dynamics, vols. 1 and 2* (Springer-Verlag, Berlin, 1997).
- [6] P.I. Crumpton and M.B. Giles, "Implicit Time Accurate Solutions on Unstructured Dynamic Grids", *AIAA-95-1671* (1995).
- [7] P.I. Crumpton and M.B. Giles, "Aircraft Computations Using Multigrid and an Unstructured Parallel Library", *AIAA-95-0210* (1995).
- [8] M.B. Giles, P. Moinier, and J. Müller, "Edge-based Multigrid and Preconditioning for Hybrid Grids", *AIAA-99-3339* (1999).
- [9] C. Gooch, "Robust Coarsening of Unstructured Meshes for Multigrid Methods", *AIAA-99-3250-CP*.
- [10] R. Löhner and K. Morgan, "An Unstructured Multigrid Method for Elliptic Problems", *Int. J. Num. Meth. Eng.*, 24 (1987).
- [11] D.J. Mavriplis, "Multigrid Strategies for Viscous Flow Solvers on Anisotropic Unstructured Meshes", *ICASE Report No. 98-6* (1998).

- [12] D.J. Mavriplis, “Directional Agglomeration Multigrid Techniques for High-Reynolds Number Viscous Flows”, ICASE Report No. 98-7 (1998).
- [13] E. Morano and A. Dervieux, “Looking for $O(N)$ Navier-Stokes Solutions on Non-Structured Meshes”, ICASE Report No. 93-26 (1993).
- [14] J. Peraire and J. Peiro, “A 3D Finite Element Multigrid Solver for the Euler Equations”, *AIAA-92-0449* (1992).
- [15] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C* (Cambridge University Press, New York, 1992).
- [16] J.C. Tannehill, D.A. Anderson, and R.H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, (Taylor & Francis, Washington D.C., 1997).
- [17] V. Venkatakrishnan and D.J. Mavriplis, “Agglomeration Multigrid for the Three-Dimensional Euler Equations”, *AIAA-94-0069* (1994).
- [18] J. Waltz, ”Unstructured Multigrid for Time-Dependent Incompressible Fluid Flow”, Ph.D. Thesis, unpublished (2000).