BULLETIN OF THE
INSTITUTE OF MATHEMATICS
ACADEMIA SINICA
Volume 28, Number 4, December 2000

TRANSFINITE SUBDIVISION UNSTRUCTURED GRID GENERATION FOR ARBITRARY 2D DOMAIN

BY

WEI-HAN WU (吳維漢)

Abstract. In this article, we propose a hybrid approach to quickly generate a dense grid on any two-dimensional geometrical domain. In the approach, we first apply an unstructured mesher to generate an initial coarse mesh. After performing post-processing procedure to enhance the grid quality on the initial coarse grid, a transfinite method is used to locally subdivide each quadrilateral or triangular cell into many geometrically similar cells. The resulting grid is a structured-like unstructured mesh. The proposed method is simple, efficient and universal for any two-dimensional geometrical domain. Numerical examples are shown to illustrate the advantages of the hybrid approach.

1. Introduction. For many numerical computation, the first important step is to generate a triangular or quadrilateral mesh on the computational domain. Up to now, there are two major grid methods, i.e. the structured grid generation methods and the unstructured grid generation methods, developed to generate the mesh on a geometrical domain. The structured grid generation methods mainly include algebraic methods [1] and various PDE methods [2,3]. Generally speaking, the structured grid methods generate a structured mesh by mapping grid points from a simple rectangular domain to a physical geometrical domain. The mapping function is straightforward for a simple geometry. Therefore, a structured grid can be generated with high efficiency. However, for a domain with highly

Received by the editors December 21, 1999.

complicated geometry, it is difficult to find a proper mapping function for the particular geometry. In such cases, the domain has to be manually decomposed into several geometrical simpler sub-domains so that the mapping function can be constructed on each sub-domain [4]. The mesh for the entire domain can be obtained by assembling all local structured grids generated on each sub-domain together. Nevertheless, the resulting mesh in general is not satisfactory for many computational problems.

For domain with highly complicated geometry, more general approaches such as the unstructured grid generation methods are developed to generate the domain grid. Currently, there are two dominate unstructured grid generation schemes, i.e. advancing front methods [5,6] and Delaunay methods [9,10], to generate the unstructured grid. A more popular approach such as [11] is to combine advancing front methods and Delaunay methods to generate the unstructured mesh with user-specified mesh density in the domain. In general, the unstructured grid schemes generate the mesh by elaborately calculating a proper position to insert a new interior node and then making a local edge connection to obtain a better mesh on the domain. For examples, the advancing front methods generate the mesh by inserting each new interior node near the front boundary, i.e. the current mesh boundary, and updating front boundary by making an edge connection with the new node. New triangles are created during the local edge connection procedure. The triangulation for the entire domain will be generated whenever the front boundary is vanished. On the other hand, the Delaunay methods use the zero circle criterion to generate Delaunay triangulation for each node on or inside the domain boundary. Here the zero circle criterion states that the circumcircle for each triangle in the triangulation contains no other points except its vertices. However, the zero circle criterion itself does not provide a systematic way to insert interior nodes. In fact, they are the various Delaunay methods to provide algorithms to effectively compute the positions of the interior nodes so that the final mesh can be constructed efficiently. One of the most important characteristics for the unstructured grid gener-

ation methods is that it is insensitive to the geometry of domain. That is, an unstructured mesh can be always obtained regardless of the complexity of domain geometry. By controlling the nodes density to be inserted, it is always possible to generate grid with user-specified density. However, comparing to the structured mesh, the unstructured grid generation methods suffer efficiency problems. They normally take much longer time to generate the entire unstructured mesh.

Here, we propose a hybrid approach to combine the structured and the unstructured grid generation methods to automatically and quickly generate a dense structured-like unstructured mesh for any two-dimensional geometrical domain. The idea is rather simple. To be able to generate the mesh for any geometrical domain, we first use an unstructured mesher to generate a coarse triangular or quadrilateral mesh for the geometrical domain. After applying some post-processing techniques, such as smoothing or edge swapping procedures on this initial mesh, a well-distributed initial coarse mesh is obtained. Next we use transfinite method to subdivide each triangular or quadrilateral cell on the initial mesh into many equal numbers of geometrically similar cells. A dense mesh with high grid quality can be obtained very quickly.

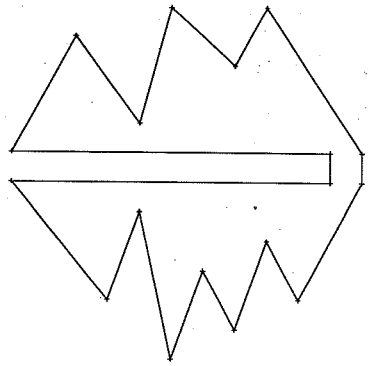
The proposed hybrid method has several advantages. First of all, it is efficient. A highly dense mesh for a complicated domain can be generated very quickly compared to the unstructured meshers. Secondly, it is automatic. An initial mesh can be always generated for any geometric domain by using unstructured methods and the final mesh is obtained by subdividing cells in the initial mesh. It won't be necessary to manually decompose the complicated domain into many simpler sub-domains in order to construct the global mesh. Finally, the generated mesh has "nice" grid quality. In fact, it is structured-like unstructured mesh. All subdivided cells generated from the same coarse cell are geometrically similar to each other.

The structure of this article is organized as follows: Following the introduction, the algorithm to generate the initial boundary mesh is discussed.

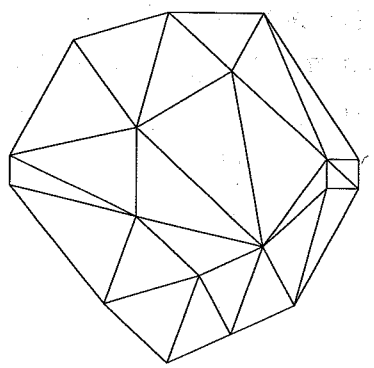
After the initial mesh is generated, the procedure to subdivide each coarse cell is described in section 3. Numerical examples are shown in section 4 and concluding remark is presented in the last section.

2. Initial Mesh. At first, we describe the geometrical boundary of domain by properly selecting boundary nodes so that the geometry of domain is sufficiently represented. To be able to generate the initial mesh for any geometrical domain, here we use a popular Delaunay method to generate the rough triangulation. In general, there are two major steps in the Delaunay methods to construct the triangulation. The first step is to construct the Delaunay triangulation for all boundary nodes. Here a Bowyer-Watson [7,8] algorithm is used to generate the mesh with boundary nodes only. It should be noted that the algorithm does not guarantee that all boundary edges on the domain are necessarily contained in the triangulation as shown in Fig.1. Some boundary edges do not exist in the Delaunay triangulation. The problem will be occurred more often when using the less representative boundary points of domain.

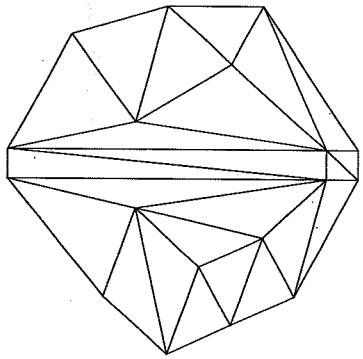
2.1. Missing boundary edge recovery. Currently, there are two major approaches to recover the missing boundary edges of domain. The first one is to apply the Delaunay method repeatly by adding additional boundary points on the missing edge until all missing boundary edges are recovered. However, the approach will certainly generate more dense mesh in the area near the missing edges than user expects. In general, it is not a preferable approach. The second approach proposed by [10] applied a random edge swapping on the intersecting edges with the missing boundary edge. Here two triangles with edge-connected can be edge-swapped if they form a convex geometry as shown in Fig. 2. The approach, in practice, can recover the missing edges within some finite steps; however, it is not considered as a good approach to randomly swap the intersecting edges along the missing boundary edge and hope that the missing boundary edge be recovered. Here, we provide a two-step algorithm as follows to recover the missing boundary edge in the Delaunay triangulation.



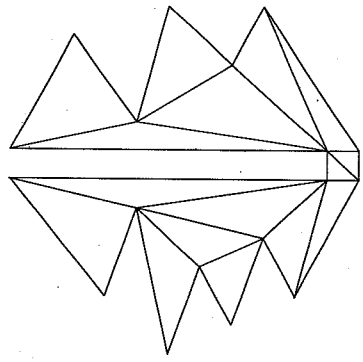
(a) domain geometry



(b) Delaunay triangulation with missing boundary edges



(c) boundary mesh after edge recovery



(d) final boundary mesh

Figure 1: Generation of boundary mesh

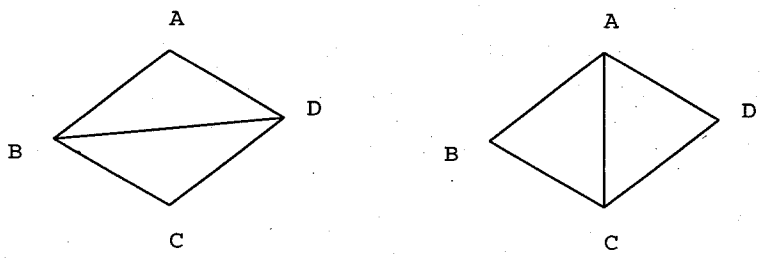


Figure 2: Edge swaping between two edge-connected triangles

Let $\boldsymbol{E}$ be one of the missing boundary edges with two end nodes as $\boldsymbol{A}$ and $\boldsymbol{B}$. All edges intersecting with the missing edge $\boldsymbol{E}$ can be classified into three different types. The first type, $T1$, consists of the intersecting edges which will not reduce the total number of intersecting edges even after attempt to swap the intersecting edge. The second type, $T2$, consists of the intersecting edges which can not be swapped. The rest of the intersecting edges are of type, $T3$. The algorithm to recover the missing edge is described as follows:

1. for each intersecting edge $\boldsymbol{e}$ along the missing edge $\boldsymbol{E}$ from node $\boldsymbol{A}$ to node $\boldsymbol{B}$
   - if $\boldsymbol{e}$ is of $T3$, swap edge $\boldsymbol{e}$
   - if $\boldsymbol{e}$ is of $T1$, append edge $\boldsymbol{e}$ into set $S1$
   - if $\boldsymbol{e}$ is of $T2$, append edge $\boldsymbol{e}$ into set $S2$
2. while sets $S1$ or $S2$ are not empty
   - for each edge $\boldsymbol{e}$ in set $S1$
     if $\boldsymbol{e}$ is of $T3$, swap edge $\boldsymbol{e}$
     if $\boldsymbol{e}$ is of $T1$, swap edge $\boldsymbol{e}$ and append new intersecting edge into set $S3$
     if $\boldsymbol{e}$ is of $T2$, append edge $\boldsymbol{e}$ into set $S4$
   - for each edge $\boldsymbol{e}$ in set $S2$
     similar procedure as above
   - assign $S1 = S3$, $S2 = S4$, clean up $S3$, $S4$ and go back to 2

In practice, the above procedure can recover missing edges very quickly. It should be noted that after the missing boundary edges are recovered, the triangulation is no longer a Delaunay triangulation but is called as a constrained Delaunay triangulation, as shown in Fig.1-(c), due to the violation of zero circle criterion after swapping edges.

2.2. Final initial mesh. After all the boundary edges are recovered, the triangles which are not inside the domain are needed to be deleted as shown in Fig.1-(d). Next the Delaunay method which provides an algorithm to properly insert interior nodes is used to construct the initial coarse mesh.

In general, there are many Delaunay methods to offer various strategies to place the interior nodes. In our current approach, we insert the interior nodes according to their positions. For each triangle in the current triangulation, we place the new node in the position which will form an equal-lateral triangle with neighboring boundary edge if it is close to the boundary; otherwise, the node is placed at the position of geometrical center of the current triangle. The density of the interior nodes to be placed can be controlled by user-provided auxiliary node-density function over the entire domain or by the node distribution on the domain boundary.

After the initial unstructured triangular mesh is generated, several indirect post-processing algorithm, such as [16,17], can be used to create the quadrilateral mesh. It should be noted that to construct a quadrilateral mesh, the total number of boundary nodes on the domain should always be even. The fact can be shown easily from the Euler's formula [12].

Finally, the initial mesh is smoothed by relocating each interior node into the area-averaged position of all connecting cells. The mesh is relatively stable after performing smooth procedure two or three times.

3. Cell Subdivision. For each cell in the initial coarse mesh, one can quickly generate a dense mesh by subdividing each cell into many smaller geometrically similar cells. The basic strategy is first to determine the node positions on each edge of cells and then subdivide each cell into many smaller sub-cells by making proper connection from the edge nodes. It should be noted that the data structure to represent the geometry data should be designed in such a way that each new additional point between any two boundary nodes should be always on the boundary of domain. Therefore, the final dense mesh after subdivision procedure would be close to the actual geometry of domain.

3.1. New edge points. An edge node is defined as an intermediate node on the edge. A vertex is not considered as an edge node. On each edge of an coarse cell, a simple approach to obtain edge nodes between any two vertices of edge is to insert a number of edge nodes uniformly. However, to

better represent the node distribution corresponding to the initial mesh, one should generate the edge nodes according to some proper criterion instead of generating edge nodes with equal distance. To begin with, we define the average edge length for a node as the average length for all edges connecting to a node. After computing average edge length for all nodes in the initial mesh, a simplified clustering distribution formula suggested by Huang [13] to compute every position of new edge points on an edge is used. It is shown as follows:

$$y_i = x_i + c\,\frac{\sin(\pi x_i)}{\pi} \qquad i \in [1..n] \tag{1}$$

where x_i is the i-th position of equally distributed edge point in $[0,1]$, c is between $[-1,1]$ and n is the total number of edge nodes to be inserted. Figure 3 shows the graphs of the formula with respect to different choices of c.

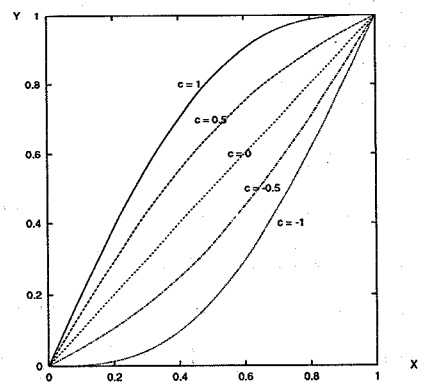


Figure 3: Clustering distribution function $y = x + c\,\frac{\sin(\pi x)}{\pi}$ with different $c \in [-1,1]$

It is easy to see that with equally distributed edge points x's, the new locations of these n edge nodes gradually shift to 1 if $c \to 1$ and they will gradually shift to 0 if $c \to -1$. When c is zero, we have equal distribution of new edge nodes. We define c as

$$c = \frac{d_1 - d_2}{d_1 + d_2} \tag{2}$$

where d_1 and d_2 are the average edge length of vertices of edge. If both vertices have the same average edge length, i.e. $d_1 = d_2$, we have the equally distributed edge nodes due to $c = 0$.

3.2. Interior nodes. After determining the locations for all new edge nodes, we now need to compute the proper positions to place the interior points corresponding to different types of cells. Since some cells with edges on geometrical boundary may have curvilinear edge, a simple approach such as directly applying linear interpolation to place the positions of new interior nodes may sometimes generate interior nodes outside the cell geometry. To overcome the potential problem, a general transfinite interpolation [14,15] is used to generate the interior nodes.

Quadrilateral cell. For a quadrilateral cell, the traditional transfinite interpolation to find the interior node can be used directly without modification. Assuming that a rectangular cell, as shown in Figure 4, has vertices N_1, N_2 , N_3 and N_4 and edge nodes N_5, N_6, N_7, N_8 on the corresponding edge. The coordinates of edge nodes N_5, N_6 , N_7 and N_8 in (s,t) coordinate system are $(s_5,0)$, $(1,t_6)$, $(s_7,1)$, and $(0,t_8)$ respectively. The intersection point P in the (s,t) coordinate system between line segment $N5$-$N7$ and line segment $N8$-$N6$ can be easily obtained by

$$(3) \qquad (s_p,t_p) = \left(\frac{s_5 + t_8(s_7 - s_5)}{1-(s_7-s_5)(t_6-t_8)} , \frac{t_8 + s_5(t_6 - t_8)}{1-(s_7-s_5)(t_6-t_8)} \right)$$

Here, it is necessary to determine the corresponding position in the (s,t) coordinate system for all edge nodes. Currently, we adopt Soni [18] approach which uses the normalized arc length along each edge to represent the coordinate in the (s,t) coordinate system, shown in Fig 5.

After the locations of all interior nodes in terms of the (s,t) coordinate system are found, it is easy to determine the corresponding positions in the (x,y) coordinate system by using usual shape function interpolations.

$$(4) \qquad P(x,y) = \sum_{i=1}^{8} N_i \, \psi_i(s_p,t_p)$$

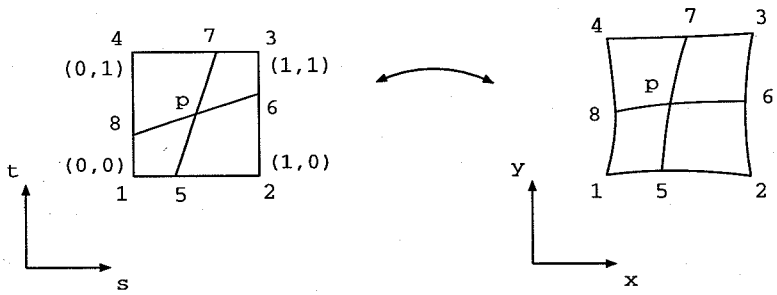


Figure 4: Transfinite Interpolation for a rectangular geometry

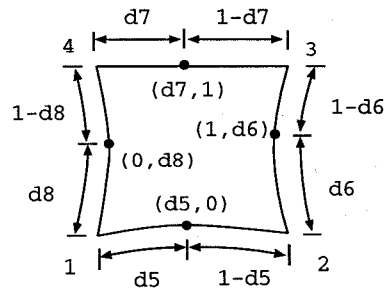


Figure 5: Normalized coordinate of edge nodes on a quadrilateral cell

where N_i represents the (x, y) coordinate of the i-th node as in Figure 4 and $\psi_i(s,t)$ is the corresponding interpolation function for each node defined as follows:

$$
\begin{aligned}
\psi_5(s,t) &= \frac{1}{s_5\,(1-s_5)}\; s\,(1-s)\,(1-t) \\
\psi_6(s,t) &= \frac{1}{t_6\,(1-t_6)}\; s\,t\,(1-t) \\
\psi_7(s,t) &= \frac{1}{s_7\,(1-s_7)}\; s\,t\,(1-s) \\
\psi_8(s,t) &= \frac{1}{t_8\,(1-t_8)}\; (1-s)\,t\,(1-t) \qquad (5) \\
\psi_1(s,t) &= (1-s)\,(1-t) - (1-s_5)\,\psi_5 - (1-t_8)\,\psi_8 \\
\psi_2(s,t) &= s\,(1-t) - s_5\,\psi_5 - (1-t_6)\,\psi_6 \\
\psi_3(s,t) &= s\,t - t_6\,\psi_6 - s_7\,\psi_7 \\
\psi_4(s,t) &= (1-s)\,t - (1-s_7)\,\psi_7 - t_8\,\psi_8
\end{aligned}
$$

The interpolation functions are the usual shape functions used widely in the finite element analysis. They will not be discussed here. For a total of n edge points to be inserted, there are n^2 interior nodes to be determined.

Triangular cell. For a triangular cell, we can use a similar procedure as above to generate the interior nodes but with minor modification. Assuming that the triangle, as shown in Figure 6, has vertices N_1, N_2 and N_3 and edge nodes N_4, N_5, N_6 and N_7 on the corresponding edge. The coordinates of edge nodes N_4, N_5, N_6 and N_7 in the (s,t) coordinate system are $(s_4,0)$, (s_5,t_5), (s_6,t_6), and $(0,t_7)$ respectively. Here N_5 and N_6 in the (s,t) coordinate system are on the line $s+t=1$. The intersection point P in terms of the (s,t) coordinate system between line segment $N4$-$N6$ and line segment $N7$-$N5$ can be obtained by

$$(s_p,t_p)=\left(\frac{s_4+nt_7}{1-mn}\,,\ \frac{t_7+ms_4}{1-mn}\right) \tag{6}$$

where $m=\dfrac{t_t-t_7}{1-t_5}$ and $n=\dfrac{s_6-s_4}{1-s_6}$.

As above, the normalized arc length along each edge of a triangular cell is used to represent coordinate of edge node as shown in Fig 7.

The location of the interior point P is

$$P(x,y)=\sum_{i=1}^{7} N_i\,\psi_i(s_p,t_p) \tag{7}$$

where N_i represents the (x,y) coordinate of the i-th node as in Figure 6 and $\psi_i(s,t)$ is the corresponding interpolation function for each node. All the interpolation functions can be easily obtained as follows:

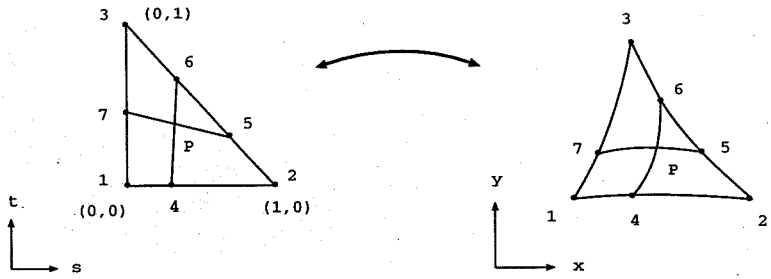


Figure 6: Transfinite interpolation for a triangular geometry

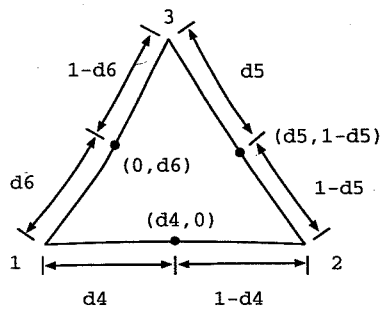


Figure 7: Normalized coordinate of edge nodes on a triangular cell

$$
\begin{aligned}
\psi_4(s,t) &= \frac{1}{s_4\,(1-s_4)}\; s\,(1-s-t) \\
\psi_5(s,t) &= \frac{1}{s_5\; t_5\,(t_5-t_6)}\; s\; t\,(t-t_6) \\
\psi_6(s,t) &= \frac{1}{s_6\; t_6\,(s_6-s_5)}\; s\; t\,(s-s_5) \\
\psi_7(s,t) &= \frac{1}{t_7\,(1-t_7)}\; t\,(1-s-t) \\
\psi_1(s,t) &= (1-s-t) - (1-s_4)\;\psi_4 - (1-t_7)\;\psi_7 \\
\psi_2(s,t) &= s - s_4\;\psi_4 - s_5\;\psi_5 - s_6\;\psi_6 \\
\psi_3(s,t) &= t - t_5\;\psi_5 - t_6\;\psi_6 - t_7\;\psi_7
\end{aligned}
\tag{8}
$$

Cell subdivision. The cell subdivision process for a quadrilateral cell is straightforward as shown in Fig. 8-(a). For a triangular cell, we can subdivide the triangular cell into two different types of cell formation as in Fig. 8-(b) and -(c) according to the maximal internal angle of the coarse

triangular cell. If the maximal internal angle of a coarse cell is too large, the cell formation will be performed by a local subdivision on the maximal angle as shown in Fig. 8-(c). This procedure is similar to the diagonal swapping procedure used to minimize the maximal angle of triangle.

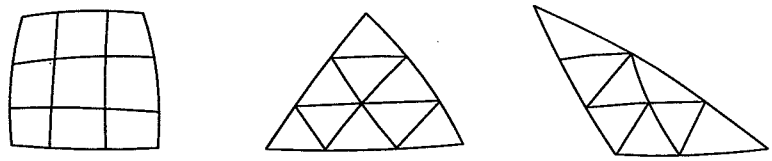


Figure 8: Subdivision of quadrilateral cell and triangular cell. There are two types of cell subdivisions for a triangular cell according to the maximal internal angle.

During the local subdivision procedure, each coarse quadrilateral cell and triangular cell both generate $(n+1)^2$ smaller cells where n is the number of inserted edge nodes. Therefore, a total of m $(n+1)^2$ cells will be generated for an initial mesh with m coarse cells after a global subdivision procedure.

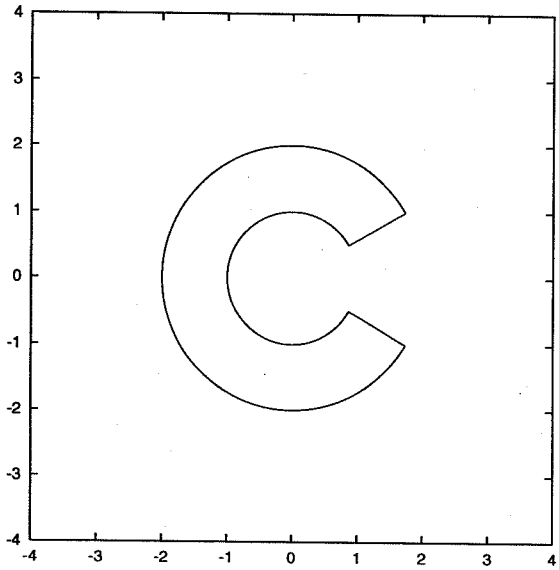
4. Numerical examples. In this section, we provide two examples toillustrate the effectness of the proposed hybrid algorithm to generate structured-like unstructured mesh. All cell type generated in the following examples is of linear triangle with three nodes; however, the hybrid procedure is successfully implemented for other types of cells such as quadratic triangle, linear quadrangle and quadratic quadrangle. The final dense meshes are shown after performing additional smooth procedure.

A square domain with C-shape body. We now consider a simple square domain with a C-shape body inside. The geometry of domain is shown in Fig. 9-(a) and the generated initial coarse mesh with 63 linear triangular cells is shown in Fig. 9-(b). In Fig 10 and Fig. 11, we generate the hybrid meshes with two and nine edge nodes inserted. For comparison, the unstructured meshes using equal number of boundary nodes generated by Delaunay method only are also shown. From the figures, it is easy to see that the grid quality of hybrid meshes is far better than that for the

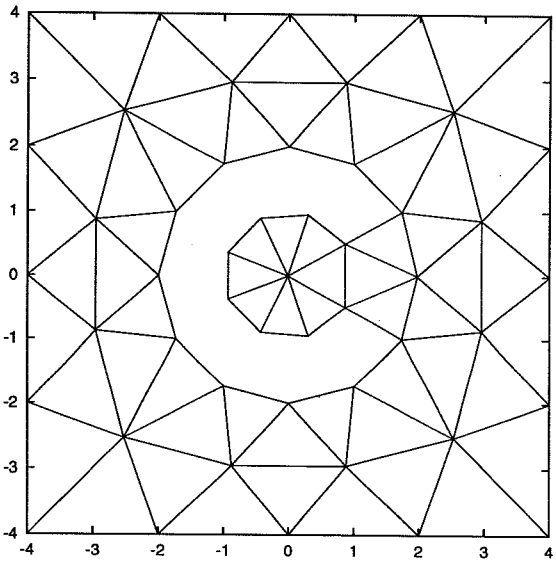
meshes generated by Delaunay method only. In terms of computational time, the time to generate the hybrid mesh is much quicker than that to generate Delaunay mesh while generating with more grid cells. Most of the CPU time spent on the subdivision process is mainly on the post-processing procedure, i.e. smoothing the final hybrid mesh. The time to actually compute the locations of edge nodes, interior nodes and generate additional smaller cells for each cell is less than 30%.

Three gear holes inside a larger gear. In this example, we consider a domain with more complicated geometry. In the geometrical domain, there are three small artificial gear holes inside a larger gear shape shown in Fig. 12-(a). The initial mesh, shown in Fig. 12-(b), generated by Delaunay method has 754 cells. A dense hybrid mesh with 6786 cells Fig. 13 is generated by inserting two edge nodes on the initial mesh. Fig. 14 shows a local enlargement on the center of geometrical domain. The CPU time to generate the initial coarse mesh is about 0.12 seconds and to perform the subdivision procedure takes another 0.13 seconds under the Ultra Sparc IIi (333 MHz CPU) system. For comparison, the CPU time to generate unstructured mesh by using the Delaunay method only with the same number of boundary nodes is 0.5 seconds. The mesh (5564 cells) and its local enlargement are shown in Fig. 15 and 16.

Conclusion. In this article, we present a hybrid algorithm to construct a structured-like unstructured mesh. The basic steps for the hybrid algorithm is first to apply the unstructured mesher to generate an initial coarse mesh and later use the transfinite interpolation to subdivide each coarse cell into many geometrically similar smaller cells. The proposed hybrid method will generate mesh with automation, better efficiency and, at the same time, good cell quality. The hybrid method can easily be extended to surface grid and three-dimensional grid without any difficulty. The studies of extension are under investigation currently.

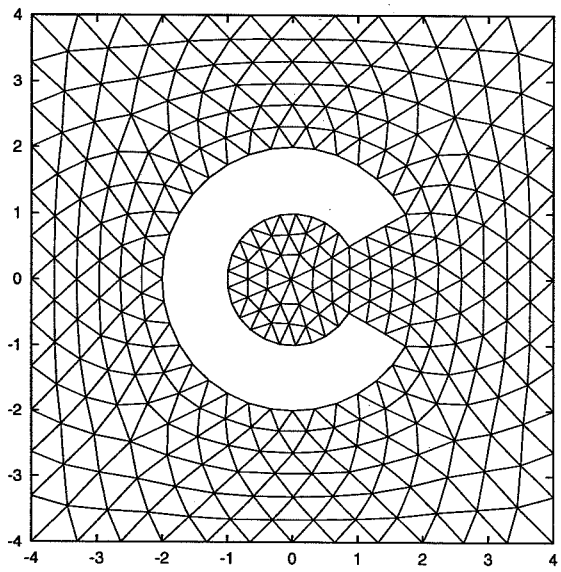


(a) Geometrical domain

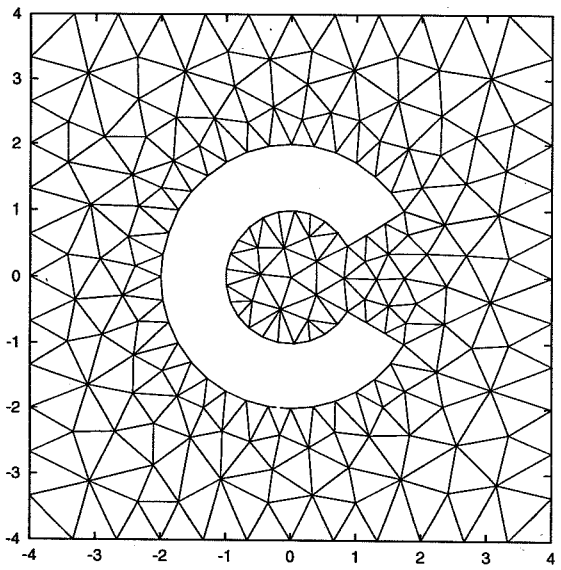


(b) Initial coarse mesh with 63 cells

Figure 9: A square domain with C-shape body and initial coarse mesh

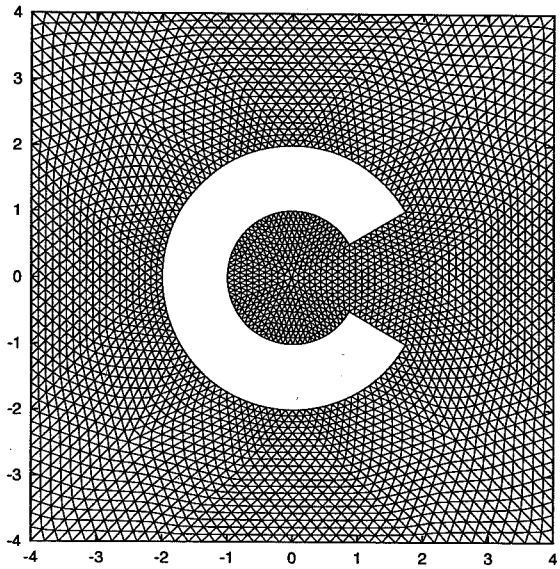


(a) Mesh (567 cells) generated by hybrid method with 2 edge nodes inserted.

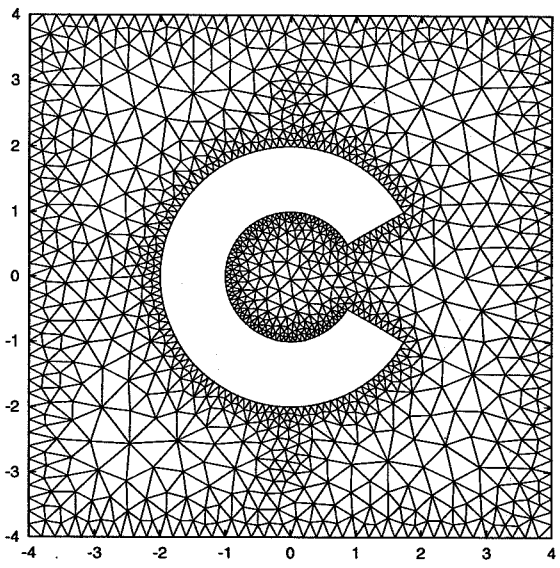


(b) Mesh (337 cells) generated by Delaunay method only.

Figure 10: A square domain with C-shape body

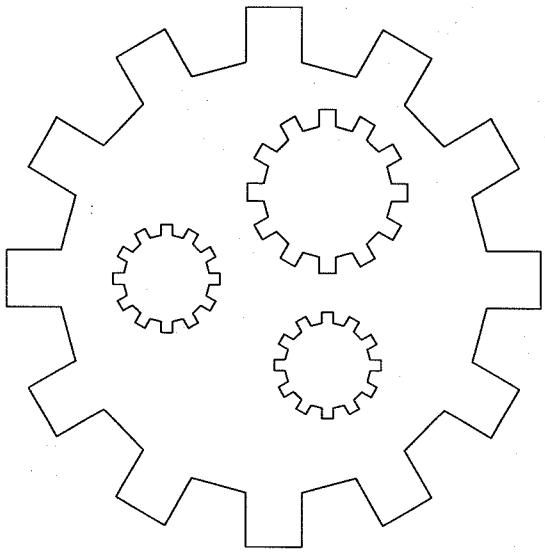


(a) Hybrid mesh (6300 cells) with 9 edge nodes inserted.

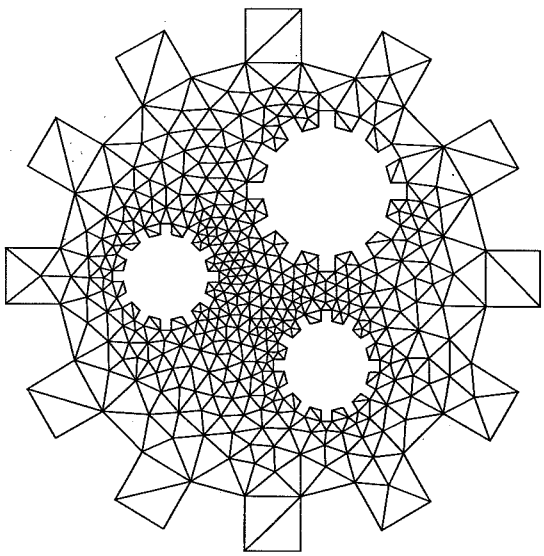


(b) Mesh (2142 cells) generated by Delaunay method only

Figure 11: A dense mesh



(a) Geometrical domain



(b) Initial coarse mesh with 754 cells

Figure 12: Gear geometry and the initial mesh

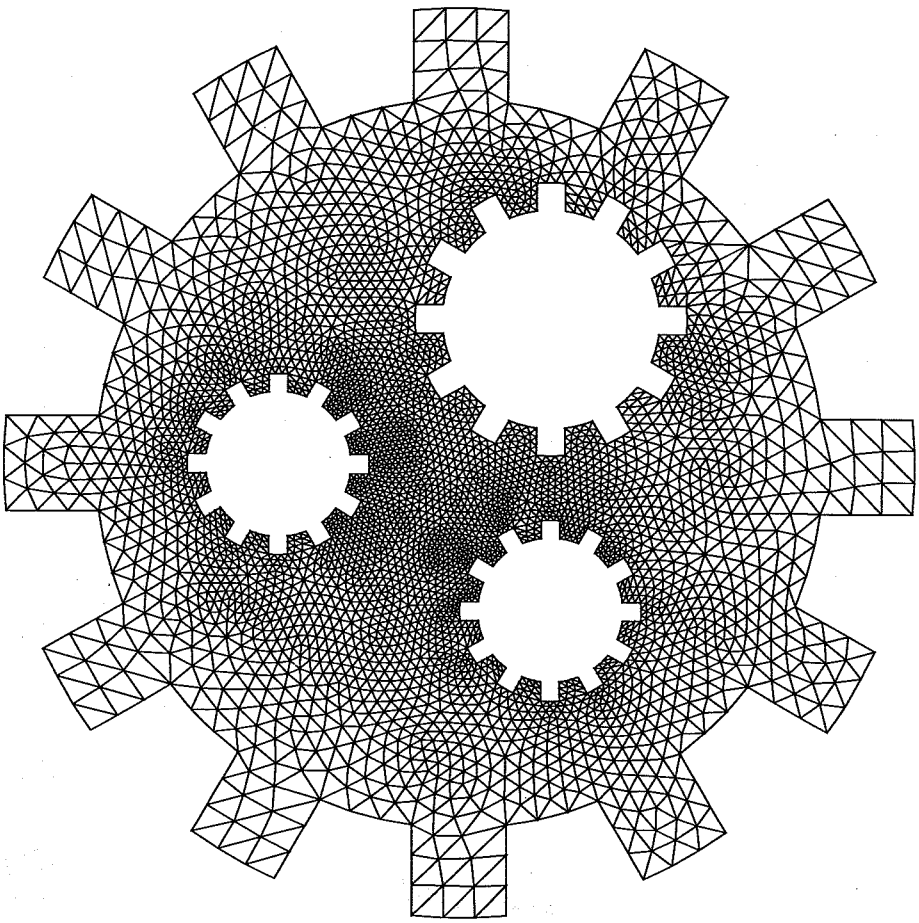


Figure 13: Hybrid mesh (6786 cells) generated by inserting two edge nodes

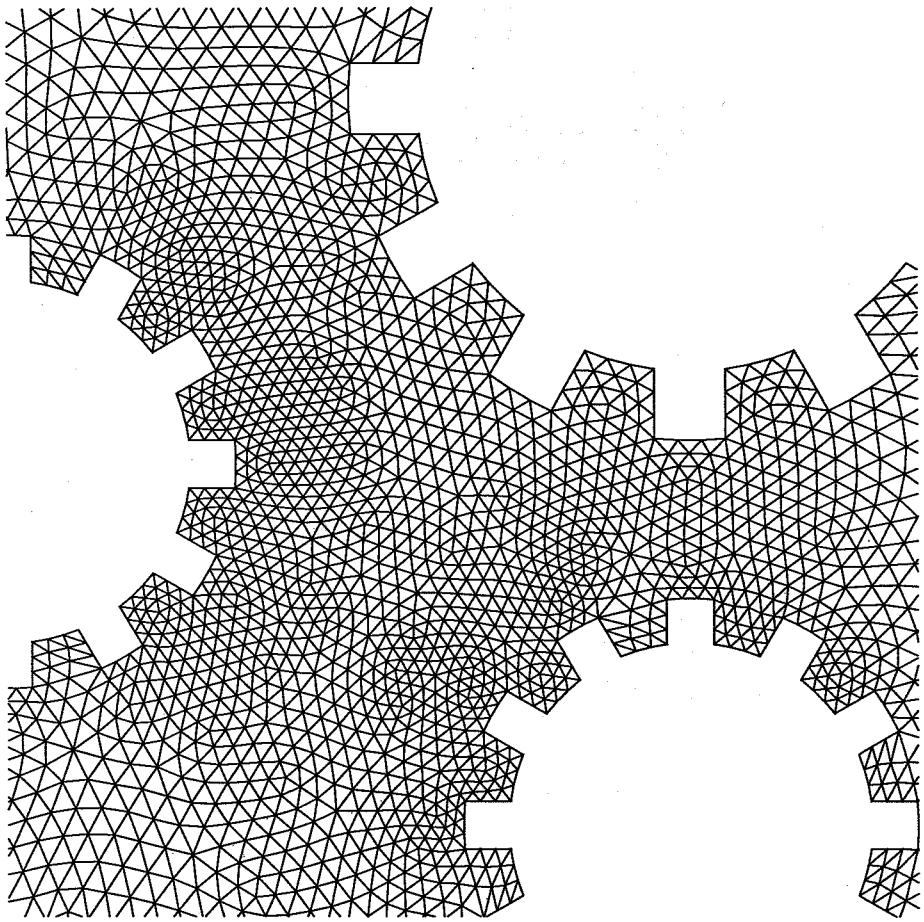


Figure 14: Local enlargement on the hybrid mesh

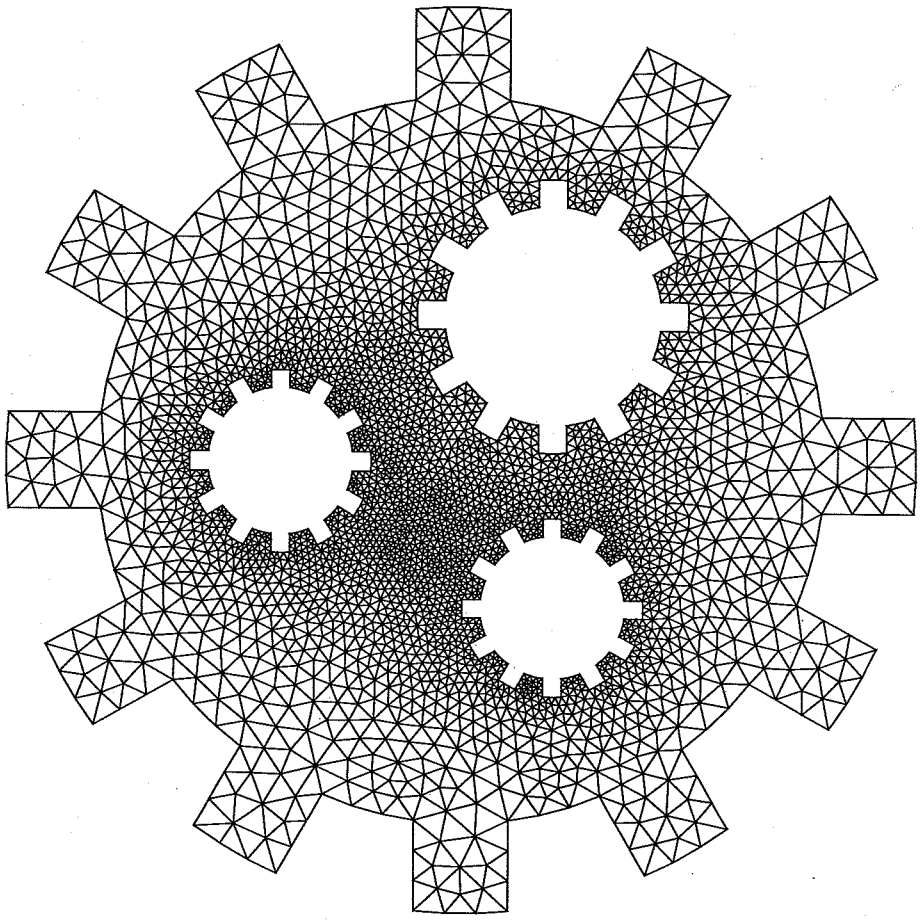


Figure 15: Mesh with 5564 cells generated by Delaunay method only

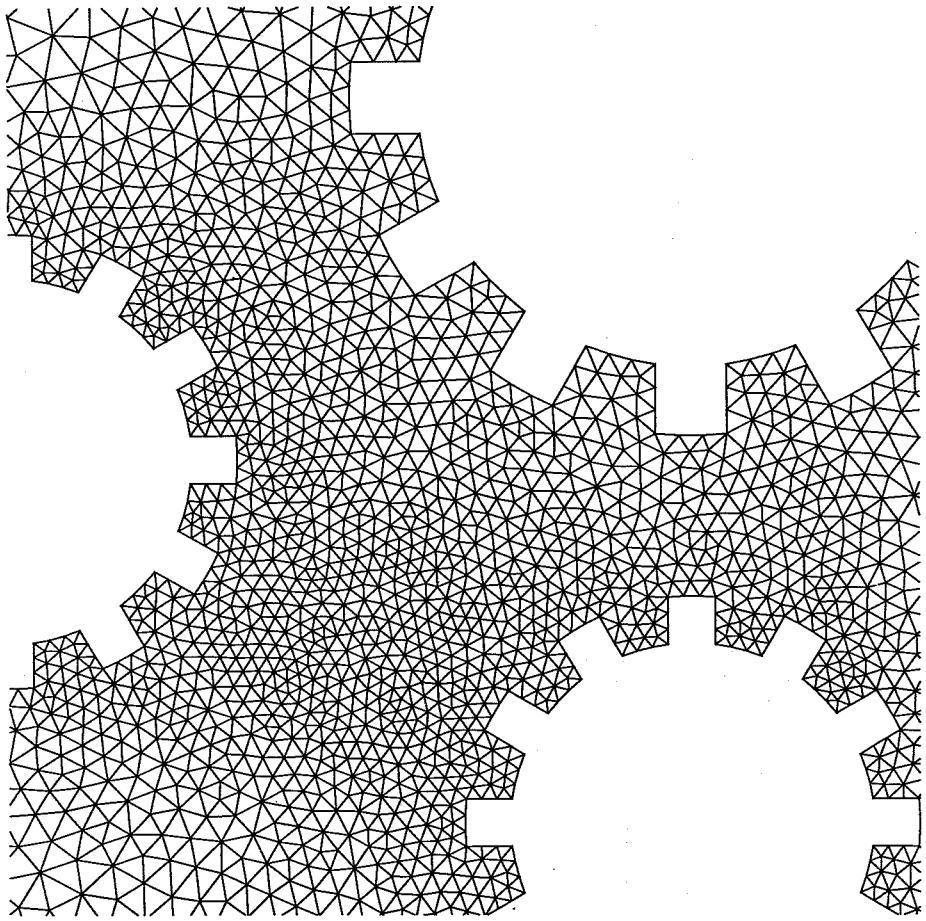


Figure 16: Local enlargement of mesh generated by Delaunay method only

References

1. J.F. Thompson, Z.U.A. Warsi and C.W. Mastin, *Numerical grid generation*, Elsevier-Science, 1985.

2. J.F. Thompson, F.C. Thames and C.W. Mastin, *Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies*, J. comp. phys, **15** (1974), 299-312.

3. S. Nakamura, *Marching grid generation using parabolic partial differential equations*, Numerical grid generation, ed. by J.F. Thompson, Elsevier-Science, 1982, 775-786.

4. J.F. Thompson, *A composition grid generation code for general 3-d regions*, AIAA-87-0275, AIAA 25th Aerospace Science Meeting, Reno, Nevada (1987).

5. S.H. Lo, *A new mesh generation scheme for arbitrary planar domains*, Int. j. numer. meth. eng., **21** (1985), 1403-1426.

6. P.L. George and E. Seveno, *The advancing-front method generation method revisited*, Int. j. numer. meth. eng., **37** (1994), 3605-3619.

7. A. Bowyer, *Computing the Dirichlet tessellation*, Comput. j., **24**(2) (1981), 162-166.

8. D.F. Watson, *Computing the Delaunay tesselation with application to Voronoi polytopes*, The Comp. j. , **24**(2) (1981), 167-172.

9. D.J. Mavriplis, *An advancing front Delaunay triangulation algorithm designed for robustness*, J. comput. phys., **117** (1995), 90-101.

10. H. Borouchaki and P.L. George, *Aspects of 2-d Delaunay mesh generation*, Int. j. numer. meth. eng., **40** (1997), 1957-1975.

11. S. Rebay, *Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm*, J. comput. phys., **106** (1993), 125-138.

12. J.A. Bondy and U.S.R. Murty, *Graph theory with application*, American Elsevier, 1976.

13. C.Y. Huang and J.T. Oden, *GAMMA2D: a multiregion/multiblock, structured/unstructured grid generation package for computational mechanics*, Comput. and struct. , **53**(2) (1994), 375-410.

14. W.J. Gordon and L.C. Thiel, *Transfinite mapping and their application to grid generation*, Numerical grid generation, ed. by J.F. Thompson, Elsevier Science, 1982, 172-192.

15. L.E. Eriksson, *Generation of boundary-conforming grids around wing-body configuration using transfinite interpolation*, AIAA J., **20**(10) (1982), 1313-1320.

16. B.P. Johnston, J. M. Sullivan Jr. and A. Kwsanik, *Automatic conversion of triangular finite element meshes to quadrilateral elements*, Int. j. numer. meth. eng., **31** (1991), 67-84.

17. S.J. Owen, M.L. Staten, S.A. Canann and S. Saigal, *Advancing front quad meshing using local triangle transformation*, Proceedings, 7th Int. Meshing Roundtable, 1998.

18. B.K. Soni, *Two- and three-dimensional grid generation for internal flow applications of computational fluid dynamics*, AIAA-85-1526, AIAA 7th Computational Fluid Dynamics Conference, Cincinnati, Ohio, 1985.

Dept. of Math., National Central University, Chung-Li, 320, Taiwan, R.O.C.