Delaunay Adaptive Remeshing Technique for Finite Element/Finite Volume Methods

Sutthisak Phongthanapanich*

Department of Mechanical Engineering Technology, College of Industrial Technology,
King Mongkut's University of Technology North Bangkok, Bangkok 10800, Thailand
*E-mail: sutthisakp@kmutnb.ac.th

Abstract

This paper describes the concept of a dynamic algorithm for constructing two-dimensional triangular meshes using the Delaunay triangulation with adaptive remeshing feature. The complexity of the geometry both simply connected and multi-boundaries domains are completely arbitrary. Laplacian smoothing technique is applied to further improve the shape and size of triangular meshes. Some examples are presented to highlight capability of the proposed algorithm.

Keywords: Adaptive Remeshing, Delaunay triangulation, Mesh generation

1. Introduction

The computational mechanics applications using finite element or finite volume methods require discretization of the domain over which a set of governing equations is to be solved. Because of arbitrarily shape of domain, improved general-purpose mesh generation algorithms have been still in great demand especially for 3D applications [1-2]. Because of varieties arbitrarily shape of domain, the unstructured mesh can provide multiscale resolution and conformity to complex geometries comparing to the structured mesh.

On the unstructured mesh approach, two methods have proved particularly successful and are widely used today. Firstly, the advancing front method, the triangles are built progressively inward from the boundaries of domain until the domain area is filled with triangles [3]. Secondly, the Delaunay triangulation method, the popular meshing technique [4] that utilizes the Delaunay criterion. The Delaunay criterion in itself, is not a meshing technique. It provides the criteria for which to connect a set of existing points in space, normally are boundary points. Therefore, the point creating technique is required in addition, to refine the triangles. The refinement technique

presented in this paper, follows the Marcum and Weatherill approach [5] which is widely used in engineering applications. The triangle aspect ratios are improved by applying the Laplacian smoothing technique that moves each node of triangles to the centroid of all triangles around the node. Then, the adaptive meshing technique developed herein generates small elements in regions with large change in solution gradients, and at the same time, larger elements in the other regions to reduce the required computer memory and the computational time.

To demonstrate the advantages of the method proposed, this paper first describes the concept behind the Delaunay triangulation. The mesh generation procedure is then proposed with automatic point creation procedure. The Laplacian smoothing technique is then described to perform mesh smoothing. A number of complex geometries are then used to evaluate the capacity and effectiveness of the proposed method. Then the adaptive meshing technique with new implementation procedure in an objected-oriented programming is described in detail. Finally, efficiency of the combined procedure is evaluated by analyzing several computational mechanics examples.

2. Delaunay Triangulation and Mesh Adaptation

Dirichlet [6-7] proposed a method to construct Dirichlet tessellation or Voronoi diagram, where as a domain could be decomposed into a set of packed convex triangles. For a given set of points in space, $\{P_k\}, k = 1,\ldots,n$, the regions $\{V_k\}, k = 1,\ldots,n$, are the boundaries which can be assigned to each point $\{P_k\}$, represent the space closer to $\{P_k\}$ than to any other points in the set. Therefore, these regions satisfy,

$$V_k = \left\{ p \in \mathcal{R} \times \mathcal{R} : \left| p - P_k \right| < \left| p - P_j \right|, \forall k \neq j \right\} \quad (1)$$

where $p \in \mathcal{R} \times \mathcal{R}$. If all the points which have some segment of a Voronoi boundary in common are joined, the result is a Delaunay triangulation. In Graph theory, Delaunay triangulation could be defined that the graph which any circle in the plane is said to be empty if it contains no vertex in its interior. This defining characteristic of Delaunay triangles, in Fig. 1, is called the empty circumcircle property.

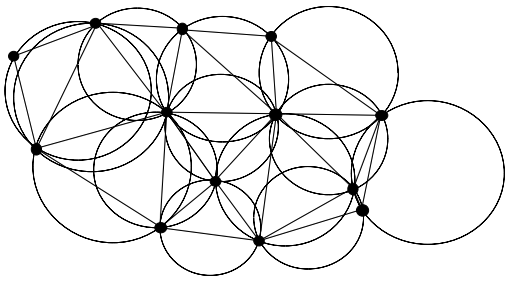

Fig. 1. Delaunay triangles have an empty circumcircle property

To ensure that all triangles are satisfy the Delaunay property, the every edge of all triangles must be locally Delaunay as shown in Fig. 2. The edge *ab* is locally Delaunay if: (a) it belongs to only one triangle and therefore bounds the convex hull and (b) it belongs to only two triangles, *abc* and *adb* and *d* lies out of the circumcircle of *abc*.

Figure 3 shows two triangles, *adc* and *bcd* are not locally Delaunay and the corresponding internal angles are $\alpha_1+\alpha_2, \delta_2, \gamma_2$, and $\beta_1+\beta_2, \gamma_1, \delta_1$, respectively. Meanwhile two triangles, *abc* and *adb* are locally Delaunay and the corresponding internal angles are $\alpha_1, \beta_1, \gamma_1+\gamma_2$, and $\alpha_2, \delta_1+\delta_2, \beta_2$, respectively. By using some geometry relations, the relations between these internal angles are

$$\begin{aligned} \alpha_1 &\geq \delta_1 \\ \beta_1 &\geq \delta_2 \\ \gamma_2 &\geq \beta_2 \\ \gamma_1 &\geq \alpha_2 \end{aligned} \qquad (2)$$

This property of locally Delaunay triangle is called Max-Min angle property [8].

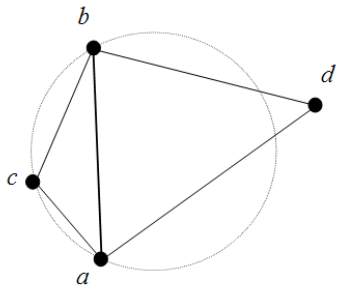

Fig. 2. The edge *ab* is locally Delaunay

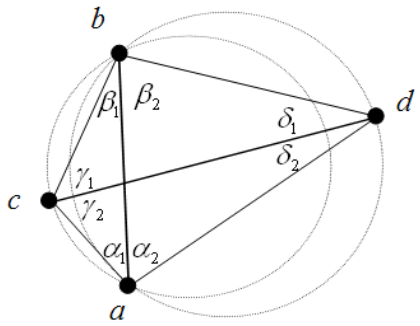

Fig. 3. Internal angles of triangles

The prior works that brought Delaunay triangulation into practical were introduced by Bowyer [6] and Watson [7] called Bowyer/ Watson algorithm. In this algorithm, when a new vertex is inserted, each triangle whose circumcircle contains the new point is no longer Delaunay (in-circle criterion) and is thus deleted all other triangles remain Delaunay are left undisturbed. Each point of the insertion polyhedron is then connected to the new point creating a new edge. The algorithm used to generate Delaunay triangulation has two steps for two-dimensional domain. Firstly, forming triangles by connected points on the boundaries of domain called boundary triangle generation. Secondly, creating points inside domain to refine triangles of previous step to conform our desired in both shape and size. The Delaunay triangulation algorithm that used to construct boundary triangle in this paper is based on the in-circle criterion according to Bowyer. The

object-oriented algorithm is described as the algorithm I below.

Algorithm I:

DelaunayTriangulation(***P***, ***T***, *p*)

Let***P0*** be the collection of node objects;
Let***T0*** be the collection of triangle objects;

P0.Initialize;
T0.Initialize;
t ⟵ ***T***.FindTriangleContainNode(*p*);
T0 ⟵ ***T***.IncircleTriangles(*t*, *p*);
P0 ⟵ ***T0***.DestroyTriangles();
T.CreateNewTriangles(***P0***, *p*);
T.AssignNeighborhoodTriangles;

End;

The Delaunay triangulation algorithm that described above does not suggest how to create points inside the domain. Many researchers introduced approaches how to create points inside the domain, to refine boundary triangles that number of methods use the set of boundary points to guide point placement [9-10]. The automatic point creation procedure in this paper derived from the algorithm suggested by Marcum and Weatherill [5]. The shape and size of triangles or density of points inside domain that created by this scheme control by two coefficients. Alpha coefficient controls point density by changing the allowable shape of formed triangles and Beta coefficient controls the regularity of the triangulation by not allowing point within a specified distance of each other to be inserted in the same sweep of the triangles within the field. The combination of Alpha and Beta coefficients cause shape and size triangles varies. The suggested values of Alpha and Beta coefficients for coarse and refine triangular mesh are 0.8 and 0.9, and 0.5 and 0.6 respectively. The implementation of automatic point creation scheme are described in algorithm II.

Algorithm II:

MeshRefinement(*P*, *T*, ***alpha***, ***beta, iteration***)

Let***P0*** be the collection of node objects;

```
For i=1 To iteration {
  Do t ⟵ T.NextTriangle {
      p ⟵ t.ComputeTriangleCentroid();
      p.dp ⟵ t.ComputePointDistribution();
      p.dm(1:3) ⟵ t.CentroidToVertices();
      p.Rejected = FALSE;
      For j=1 To 3 {
        If (p.dm(j) < (alpha * p.dp)) {
            p.Rejected = TRUE;
          Break;
        };
      };
      If (Not p.Rejected) {
        P0.Initialize;
```

```
            P0 ← T.FindInsertedNodeNearestTriangles;
            Do p1 ← P0.NextNode {
             If (distance(p, p1) < (beta * p.dp)) {
                    p.Rejected = TRUE;
                    Break;
                    };
             };
          };
          If (Not p.Rejected)
               P.AddNodeAsInsertedNode(p);
       };
       Do p ← P.NextInsertedNode {
            Call DelaunayTriangulation(P, T, p);
       };
};

End;
```

Since the proposed algorithm above does not guarantee the good mesh topology, the mesh relaxation [12] based on an edge-swapping technique is highly recommended for well-shaped mesh improvement. The objective of this method is to make the topology of elements closer to equilateral triangles by swapping edges to equalize the vertex degrees (number of edges linked to each point) toward the value of six.

Finally, shapes and sizes of triangles formed from the previous step can be improved by applying a mesh smoothing technique. This paper uses the Laplacian smoothing technique because of less computational time requirement. The point repositioning formula is derived from the finite difference approximation of the Laplace's equation. Each interior node is moved successively to the centroid of the area which is formed by connecting neighbouring nodes. Several passes are made through the entire set of all interior nodes to produce optimized shape and size of the triangles. The new node locations using the Laplacian smoothing are computed from,

$$x_{ic} = \frac{\sum_{i=1}^{M} x_i}{M} \quad and \quad y_{ic} = \frac{\sum_{i=1}^{M} y_i}{M} \quad i = 1,2,...,M \qquad (3)$$

Lastly the remeshing technique generates an entirely new mesh based on the solution obtained from a previous mesh. The technique was first introduced and applied for high-speed compressible flow analysis [11]. There are two main steps in the implementation of the adaptive remeshing technique; the first step is the determination of proper element sizes and the second step is the new mesh generation.

To capture steep gradients of the solution, small elements are needed along that region in the domain. The proper element size h_i is computed by requiring that the error should be uniform for all elements:

$$h_i^2 \;_i = h_{\min}^2 \;_{\max} = \text{constant} \qquad (4)$$

where $_i = \max\left(\left|\frac{\partial^2\phi}{\partial X^2}\right|, \left|\frac{\partial^2\phi}{\partial Y^2}\right|\right)$ is the higher principal quantity of the element considered, and ϕ is the selected solution indicator. The regions, which will be refined or coarsened by *AdaptiveMeshing* algorithm below, are identified by a dimensionless error indicator using the pressure-switch coefficient. The indicator at node *I* is given by,

$$E_I = \frac{\sum_{e\in I}\left|2\phi_I - \phi_J - \phi_K\right|}{\sum_{e\in I}(A^* + B^*)} \qquad (5)$$

where *J* and *K* are the other two nodes of the triangle, *e*, $A^* = \max(\left|\phi_I - \phi_J\right|, \alpha(\phi_I + \phi_J))$ and $B^* = \max(\left|\phi_I - \phi_K\right|, \alpha(\phi_I + \phi_K))$. The value of α is used to identify the solution discontinuity or numerical oscillation. According to numerical experiment especially for the proposed scheme that will be explained later, the value of α is prescribed as 0.005 in this paper. This means $A^* = 0.005(\phi_I + \phi_J)$ and $B^* = 0.005(\phi_I + \phi_K)$ if ϕ_J and ϕ_K are oscillated within 1% of ϕ_I, respectively.

Practical experience found that this type of error indicator for complex problems where regions such as shock or discontinuity have different strength may cause inaccurate solution from inadequate refinement because the point spacing is scaled according to the maximum value of the second derivatives. To overcome this problem, an element size scaling function, which scales the point spacing of point P_i between minimum and maximum element sizes, $h_{\min}$ and $h_{\max}$ within the range of $\chi_{\min}$ and $\chi_{\max}$ has been used,

$$\chi_i = \text{Scale}\left(\frac{h_{\max} - dp_i}{h_{\max} - h_{\min}}, 0, 1, \chi_{\min}, \chi_{\max}\right) \qquad (6)$$

where dp_i is nodal distribution value of node *i*. The coefficient χ_i controls the point insertion in the regions of high solution gradient and eliminates undue distortion of the triangle regularity. The value of $\chi_{\min}$ limits number of points insertion in high gradient region such as shock, while the value of upper limit $\chi_{\max}$ allows more points to be inserted into the lower solution gradient region. As shapes of adapted elements generated by this function may be distorted, the Alpha and Beta coefficients are incorporated as coefficients of such function to control point density and the regularity of triangulation. The concept of how to implementation Eqs.(4)-(6) can be illustrated by Fig. 4.

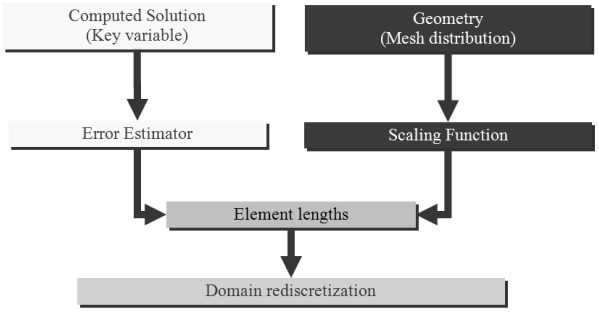

Fig. 4. Element size determination concept

Algorithm III:

```
AdaptiveRemeshing(P, T, P0, Hmin, Hmax, threshold)
Do {
  Do p ← P0.NextInteriorNode {
      If (p.hi ≤ Hmax) {
      t ← T.FindTriangleContainNode(p);
      pq ← t.ComputeTriangleCentroid();
     pq.dp ← t.ComputePointDistribution();
        pq.dm(1:3) ← t.CentroidToVertices();
        pq.Rejected = FALSE;
        For j=1 To 3 {
          If (pq.hi > pq.dm.Average Or pq.dm(j) <
Hmin) {
                pq.Rejected = TRUE;
            Break;
           };
        };
        If (Not pq.Rejected) P.AddNode(pq);
        };
  };

  Do p ← P.NextInsertedNode {
      Call DelaunayTriangulation(P, T, p);
  };
} Loop Until (P.InsertedNodes <= threshold);
End;
```

3. Examples

To evaluate the performance of the adaptive meshing technique with the Delaunay triangulation, the specification of element size, h_i, is given as an analytic function defined for two-dimensional domain. The adaptive mesh generation process starts from an initial mesh generated in the domain, then the values of the element sizes at all points are computed by the given function. The mesh generation coupled with the adaptive meshing procedure is iterated until the resulting mesh becomes globally stable. The iteration process is terminated if the total node increment is fewer than the specified number. The one example of mesh generation and two examples of adaptive mesh generation with the analytical function for specifying element sizes presented herein are: (1) Airfoil NACA 0012 (2) adaptive meshes along the centerline of a rectangular domain, and (3) an alpha-shape adaptive meshes in a square domain.

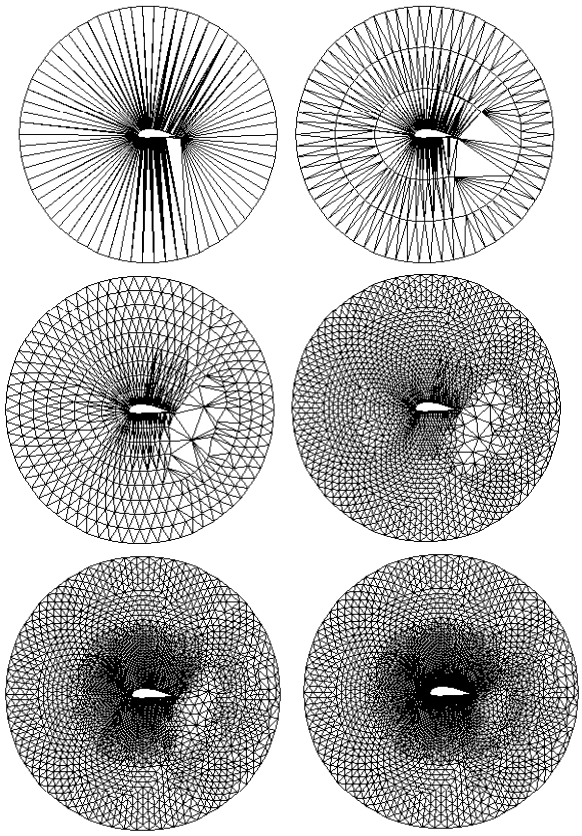

Fig. 5. Mesh generation of airfoil NACA 0012 configuration

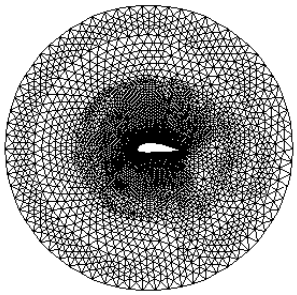

Fig. 6. Mesh improvement with Laplacian smoothing technique

3.1 Airfoil NACA 0012

To demonstrate the efficiency of the Delaunay triangulation algorithm and the Laplacian smoothing technique, Fig. 5 shows the progress of the domain discretization refinement for the airfoil NACA 0012. The geometry consists of one airfoil shape surrounding by circular boundary of the domain. Figure 6 shows the mesh quality improvement with the Laplacian smoothing technique.

3.2 Adaptive Meshes along the Centerline of a Rectangular Domain

The second example presents an adaptive mesh generation in a 3.0 × 5.0 rectangular domain. The element sizes at points in the domain are given by the distribution function,

$$h(y) = 0.42 - \frac{1}{\sqrt{2\pi}\sigma} e^{-\left[\frac{y-\mu}{2\sigma}\right]^2} \qquad (7)$$

where *y* is the variable and the values of μ and σ are constants equal to zero and one, respectively. Figure 7 shows the series of adaptive meshes generated by three iterations based on a coarse initial mesh. The value of mesh generation coefficients, α, β, $\chi_{\min}$, $\chi_{\max}$ are 0.5, 0.6, 0.75, and 1.10, respectively. Due to the prescribed distribution function in Eq. (7), small element sizes are specified around the centerline of the domain. The figure shows that size similarity of the adaptive meshes is generated along the narrow band around the centerline of the domain. The value of $\chi_{\min}$ limits the number of point insertion along the centerline of the domain, while the value of $\chi_{\max}$ allows more nodes to be inserted into the other regions. The specification of scale range and

$\chi_{\min}$, $\chi_{\max}$ have strong effects on the resulting meshes as shown in Fig. 7. Without the scale range, the mesh is composed of small elements concentrated around centerline with progressively larger elements outwards as $h_a < h_b, h_c$. Hence, a mesh consisting of relatively uniform elements in a wider centerline band of the domain may be generated. This mesh has better physical correlation with the behaviors of shocks. The scale range function sorts the nodal spacing values into prescribed intervals according to $\chi_{\min}$ and $\chi_{\max}$. In each interval, the generated element sizes are relatively uniform.

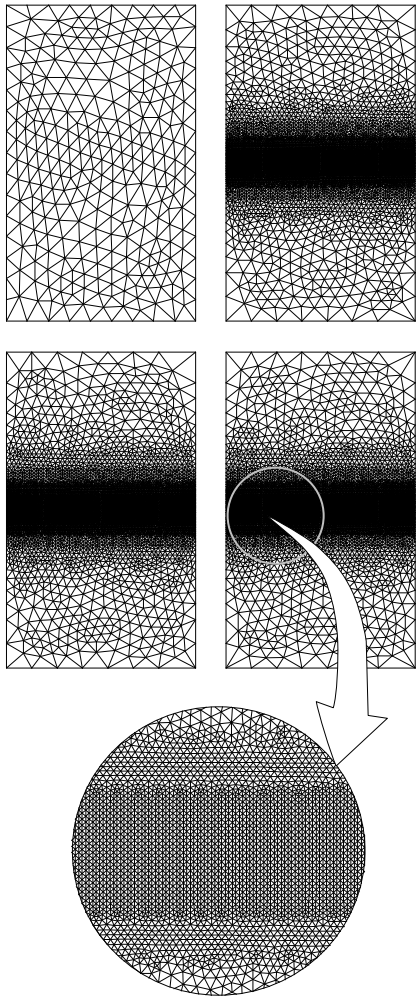

Fig. 7. Adaptive Meshes along the Centerline of a Rectangular Domain

3.3 Alpha-Shape Adaptive Meshes in a Square Domain

The third example presents an alpha-shape adaptive mesh generation in a square domain. The alpha shape function [13] is used to calculate element sizes in an 8 × 8 square domain:

$$h(x,y) = \begin{cases} \min\left(0.2(\lambda-1)^3 + 0.005, 1.0\right) & \text{if } \lambda \geq 1 \\ \min\left(0.2(\lambda-1)^2 + 0.01, 1.0\right) & \text{if } \lambda < 1 \end{cases} \quad (8)$$

where the value of parameter λ is determined from $x^3 - y^2 + 2 - 3\lambda x = 0$. Figure 4 shows the

sequence of four adaptive meshes generated from a coarse initial mesh. The value of mesh generation coefficients, $\alpha, \beta, \chi_{\min}, \chi_{\max}$ are 0.5, 0.6, 0.5, and 0.85, respectively. The smaller elements are generated along the alpha-shape in the domain while larger elements are generated in the other regions.

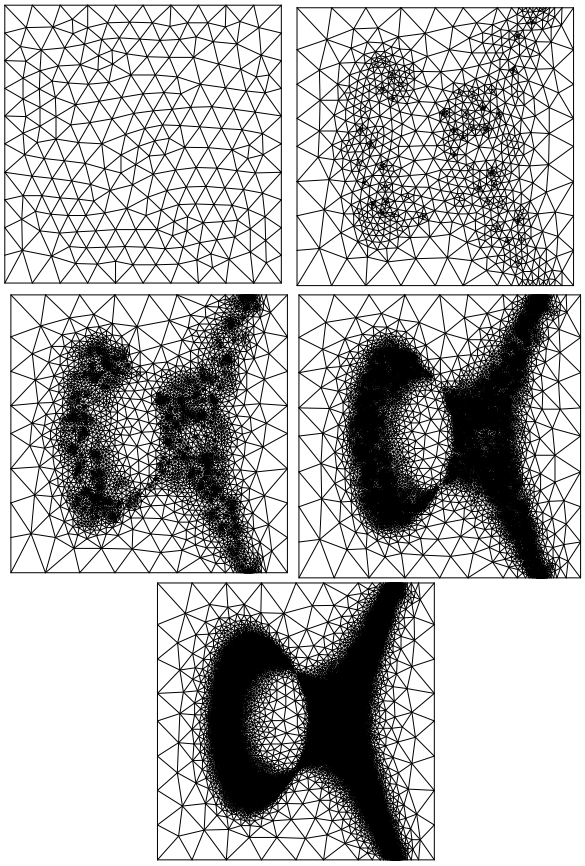

Fig. 4. An alpha-shape adaptive meshes in a square domain

The application of finite element and finite volume methods for solving continuum problems such as solid mechanics, heat transfer, and fluid flow problems incorporate with the mesh generation algorithms and adaptive remeshing technique as described above had been reported by Refs.[14-20].

4. Conclusions

This paper has discussed a method to construct unstructured mesh of triangles based on Delaunay triangulation. Mesh refinement algorithm has been suggested by automatic point creation scheme. The adaptive remeshing technique was described in detail with the pseudo-code presented in object-oriented programming concept. To capture fast variations of the solution effectively, a new element size scaling function was introduced into the adaptive remeshing technique. The combined algorithm was evaluated by generating triangular meshes and adaptive meshes for three examples with prescribed element size functions.

Acknowledgements

The authors are pleased to acknowledge the National Metal and Materials Technology Center (MTEC) for supporting this research work.

References

[1] Geuzaine, C. and Remacle, J.F. (2009) "Gmsh: A Three-Dimensional Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities", *International Journal for Numerical Methods in Engineering*, Vol. 79, No. 11, 1309-1331.

[2] Kakosimos, K.E. and Assael, M.J. (2009) "An Efficient 3D mesh Generator based on Geometry Decomposition", *Computers and Structures*, Vol. 87, No. 1-2, 273-8.

[3] Lo, S.H. (1985), "A New Mesh Generation Scheme for Arbitrary Planar Domains", *International Journal for Numerical Methods in Engineering*, Vol. 21, 1403-1426.

[4] Owen, S.J. (1998), "A Survey of Unstructured Mesh Generation Technology", Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh.

[5] Marcum, D.L. and Weatherill, N.P. (1995), "A Procedure for Efficient Generation of Solution Adapted Unstructured Grids", *Computer Methods in Applied Mechanics and Engineering*, Vol. 127, 259-268.

[6] Bowyer, A. (1981), "Computing Dirichlet Tessellations", *The Computer Journal*, Vol. 24, No. 2, 162-166.

[7] Watson, D.F., (1981) "Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes", *The Computer Journal*, Vol. 24, No. 2, 167-172.

[8] Edelsbrunner, H. (2000) "Triangulations and meshes in Computational Geometry", Acta Numerica, Vol. 9, 133-213

[9] Jin, H. and Wiberg, N.E. (1990), "Two-Dimensional Mesh Generation, Adaptive Remeshing and Refinement", *International Journal for Numerical Methods in Engineering*, Vol. 29, 1501-1526.

[10] Ruppert, J. (1995), "A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation", *Journal of Algorithms*, Vol. 8, 548-585.

[11] Peraire, J., Vahdati, M., Morgan, K. and Zienkiewicz, O.C. (1987), "Adaptive Remeshing for Compressible Flow Computations", *Journal of Computational Physics*, Vol. 72, 449-466.

[12] Frey, W.H. (1991), "Mesh Relaxation: A New Technique for Improving Triangulations", *International Journal for Numerical Methods in Engineering*, Vol. 31, 1121-1133.

[13] Borouchaki, H., George, P.L. and Mohammadi, B. (1997), "Delaunay Mesh Generation Governed by Metric Specifications. Part II. Application", *Finite Elements in Analysis and Design*, Vol. 25, pp. 85-109.

[14] Phongthanapanich, S. and Dechaumphai, P. (2006), "Heat Transfer Analyses by Means of Flux-based Formulation and Mesh Adaptation", *Engineering Journal of Siam University*, Vol. 12, 1–8.

[15] Phongthanapanich S., Boonmalert, P. and Dechaumphai, P. (2006), "Characteristic-based Split Finite Element Algorithm for Viscous Incompressible Flow Problems", *Engineering Journal of Siam University*, Vol. 13, 26-31.

[16] Phongthanapanich S., Chatakom, S. and Dechaumphai, P. (2006), "Delaunay Triangulation with Adaptive Meshing Technique for Crack Propagation Analysis", *Engineering Journal of Siam University*, Vol. 13, 1-8.

[17] Phongthanapanich, S. (2009), "Nodeless Variable Adaptive Finite Element Methods for Steady-state Heat Transfer Problems", *Engineering Journal of Siam University*, Vol. 19, 1–9.

[18] Phongthanapanich, S. and Dechaumphai P. (2009), "Adaptive Finite Element Method for Heat Transfer Analysis by Means of Linear Flux-based Formulation", *The Journal of KMUTNB*, Vol. 19, 306–314.

[19] Theeraek P., Phongthanapanich S., Dechaumphai P., (2009), "Combined Adaptive Meshing Technique and Finite Volume Element Method for Solving Convection-Diffusion Equation", *The Asian International Journal of Science and Technology*, Vol. 2, 51-58.

[20] S. Phongthanapanich (2010), "*J*-Domain Integral Technique for Crack Analysis with Adaptive Finite Element Method", *The Journal of KMUTNB*, Vol. 20, 189-195.