Delaunay Triangulation with Adaptive Meshing Technique for Crack Propagation Analysis

Sutthisak Phongthanapanich[a,*] Sathaporn Chatakom[b] and Pramote Dechaumphai[c]
[a,b]Department of Mechanical Engineering Technology, College of Industrial Technology, King Mongkut's Institute of Technology North Bangkok, Bangkok 10800, Thailand
[c]Mechanical Engineering Department, Chulalongkorn University, Bangkok 10330, Thailand

Abstract

The adaptive Delaunay triangulation is combined with a finite element method for analysis of two-dimensional crack propagation problems. The content includes detailed descriptions of the Delaunay triangulation with mesh generation, node creation, mesh smoothing, and adaptive remeshing using object-oriented programming. The resulting stress intensity factors and simulated crack propagation behavior are used to evaluate the effectiveness of the combined method. The sample problem for predicting the stress intensity factors of a single edge cracked plate under mixed-mode loading are used to evaluate the accuracy of the procedure. A sample problem for predicting crack growth trajectory in a single edge cracked plate with three holes is also simulated and the result assessed.

Keywords: Adaptive meshing technique, Crack propagation, Finite element method.

1. Introduction

The Delaunay triangulation, based on the concept of the Voronoi diagram [1,2], is one of the automated mesh generation algorithms that has recently gained popularity. In this paper, the domain discretization based on the algorithm proposed by Weatherill and Hassan [3] which constructs triangular mesh for crack propagation analysis is described in details. In addition, an adaptive remeshing technique is developed and incorporated into the Delaunay triangulation in order to improve the solution accuracy of the finite element method. The technique generates an entirely new mesh based on the solution obtained from the previous mesh; such that elements in regions with large changes of solution gradients become smaller and elements in areas with little changes of solution gradients grow larger. The pseudo-code procedures based on object-oriented programming concept are presented for the Delaunay triangulation algorithm, the automatic node creation procedure and the adaptive remeshing technique.

The stress intensity factor is a critical parameter in the prediction of fatigue crack growths in crack propagation problems. The standard six-node isoparametric elements are used for modeling. In order to improve the accuracy of the near-tip stress fields, elements with mid-side nodes displaced from their nominal positions to quarter points are employed near the crack tip [4]. The nodal displacements around the crack tip are then used to determine the stress intensity factors using the displacement extrapolation method [5]. The example under mixed mode loadings, is modeled to evaluate accuracy and effectiveness of the combined procedure. In addition, the capability of the proposed procedure is further demonstrated by the simulation of a crack propagation trajectory in a single edge cracked plate with three holes under mixed-mode loading. Solutions are compared with both the numerical solutions, as well as the experimental data.

2. Formulation

The stress intensity factors for the fracture modes I and II, K_I and K_{II}, represent the intensity factors under opening and

shearing modes, respectively. They may be determined from,

$$K_I = \frac{E}{3(1+\nu)(1+\kappa)}\sqrt{\frac{2\pi}{L}}\left(4(v_b - v_d) - \frac{(v_c - v_e)}{2}\right) \quad (1a)$$

$$K_{II} = \frac{E}{3(1+\nu)(1+\kappa)}\sqrt{\frac{2\pi}{L}}\left(4(u_b - u_d) - \frac{(u_c - u_e)}{2}\right) \quad (1b)$$

where E is the modulus of elasticity, ν is the Poisson's ratio, κ is the elastic parameter defined by $(3 - 4\nu)$ for plane strain and $(3 - \nu) / (1 + \nu)$ for plane stress problems, and L is the element length. The u and v are the displacement components in the x and y directions, respectively; their subscripts indicate the position as shown in Fig. 1.

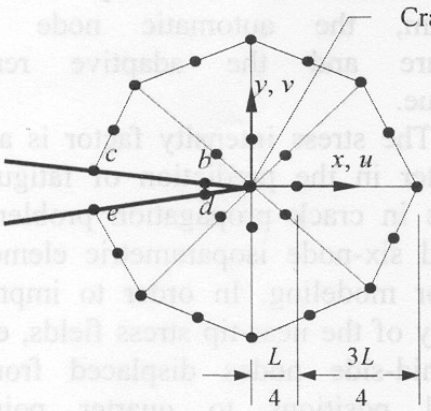

Fig. 1. Quarter-point triangular elements around the crack tip

Crack propagation in practical problems normally occurs under mixed mode loading. Based on the maximum circumferential stress theory [6], the direction of crack propagation θ may be computed from,

$$K_I \sin\theta + K_{II}(3\cos\theta - 1) = 0 \quad (2)$$

The finite element equations for determining nodal displacements and stresses can be derived from the governing partial differential equations that represent the equilibrium conditions. The six-node triangular elements with mid-side nodes displaced from their nominal positions to quarter points of the crack tip, are used to form up a circular zone surrounding the crack tip to facilitate high solution accuracy. The radius of the circular zone is specified to be no larger than one-eight of the initial crack length, with roughly one element every 30 degrees in the circumferential direction [7]. The crack growth simulation is based on the maximum circumferential stress and the increment of the crack length during each crack propagation step is specified by the user.

3. Delaunay Triangulation for Crack Propagation Analysis

For a given set of points in space $\{P_k\}$, $k = 1,\ldots, n$, the regions $\{V_k\}$, $k = 1,\ldots, n$, are boundaries assigned to each point P_k and represent the space closer to P_k than to any other points in the set. Therefore, these convex regions satisfy Eq. (3), implies that boundaries of the Voronoi diagram must lie half way between the two points on either side of the boundary. If all points which have some segments of a Voronoi boundary in common are joined, the resulting shape is a Delaunay triangulation.

$$V_k = \left\{P_i : |p - P_i| < |p - P_j|, \forall j \neq i\right\} \quad (3)$$

3.1 Delaunay Mesh Generation Algorithm

The Delaunay triangulation algorithm for constructing boundary triangles is based on the in-circle criterion. The details of the Delaunay triangulation algorithm are described by Ref. [8]. The algorithm is implemented by developing a computer program, which summarized as a pseudo-code in the algorithm I below with variables and parameters defined in next section.

Algorithm I; `DelaunayTriangulation(P, T, p)`

```
Let P0 be the collection of node
objects;
Let T0 be the collection of mesh
objects;

P0.Initialize;
T0.Initialize;
```

```
t ← T.FindTriangleContainNode(p);
T0 ← T.IncircleTriangles(t, p);
P0 ← T0.DestroyTriangles();
T.CreateNewTriangles(P0, p);
T.AssignNeighborhoodTriangles;
End;
```

3.2 Automatic Node Creation Procedure

The Delaunay triangulation algorithm described above does not explain the method of creating nodes inside the domain. So far, researchers have introduced several approaches for creating new nodes inside the domain by refining boundary triangles such that the set of boundary points guide new node placements [9,10]. The new node creation procedure for geometry with crack developed in this paper is extended from that proposed by Weatherill and Hassan. The shape and size of triangles are controlled by two coefficients, the Alpha and the Beta coefficients. The Alpha coefficient controls node density by changing the allowable shape of the formed triangles. The Beta coefficient controls the regularity of triangulation by disallowing node within a specified distance of each other to be inserted in the same sweep of the triangles within the field.

The main idea of the automatic node creation procedure is the search for the triangle that conforms to both Alpha and Beta testing criteria and a new node placement at the centroid of that triangle. New triangles can then be created by Delaunay triangulation algorithm as described in algorithm I. The key idea of the procedure is summarized as pseudo-code in the algorithm II below;

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.ComputePointDistributionFunction();
        p.dm(1:3) ← t.DistanceCentroidToVertices();
        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.FindInsertedNodeOfNearestTriangles;
            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;
```

The point distribution function, dp_i for the node, p_i, is computed from Eq. (4) where node i is surrounded by M nodes [3].

$$dp_i = \frac{1}{M}\sum_{j=1}^{M}\left| p_j - p_i \right| \qquad (4)$$

3.3 Adaptive Remeshing Technique

The adaptive remeshing technique generates an entirely new mesh based on the solution obtained from a previous mesh [11]. In this paper, the technique is modified and corporated into the Delaunay triangulation and the finite element method to analyze crack propagation problems. 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 determine proper element sizes at different locations in the domain, the von Mises stress σ are used as the indicator for computing proper element sizes. As small elements must be placed in the region where changes in the von Mises stress gradients are large, the second derivatives of the von Mises stress at a point with respect to global coordinates x and y are needed. Using the concept of principal stresses determination from a given state of stresses at a point, the principal quantities in the principal directions X and Y where the cross-derivatives vanish are determined. The maximum principal quantity is then used to compute the proper element size, h_i, by requiring the error to be uniform for all elements,

$$h_i^2 \lambda_i = h_{\min}^2 \lambda_{\max} = \text{constant} \qquad (5)$$

where λ_{max} is the maximum principal von Mises stress quantity of all elements, h_{min} is the minimum element size specified by users, and λ_i is the higher principal quantity of the element that is considered,

$$\lambda_i = \max\left(\left|\frac{\partial^2 \sigma}{\partial X^2}\right|, \left|\frac{\partial^2 \sigma}{\partial Y^2}\right|\right) \qquad (6)$$

The mesh regeneration, based on the concepts of the Delaunay triangulation and the mesh refinement as described by Algorithm I and II, is combined with the adaptive remeshing technique to generate a new mesh. The new mesh is constructed using the information from the previous mesh (background mesh). Node insertion is performed in an element which contains the background node that has point distribution function smaller than *Hmax* and the average of the distance to the three vertices of that element. With this technique, the new mesh thus consists of small elements in the regions with large changes in solution gradients, and large elements in the other regions where the changes in solution gradients are small. The process of adaptive remeshing technique is summarized as pseudo-code below;

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.ComputePointDistributionFunction();
         pq.dm(1:3) ← t.DistanceCentroidToVertices();
         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.AddNodeAsInsertedNode(pq);
      };
   };
   Do p ← P.NextInsertedNode {
      Call DelaunayTriangulation(P, T, p);
   };
} Loop Until (P.InsertedNodes <= threshold);
End;
```

3.4 Mesh Generation Implementation

This section presents the main algorithm for implementing together the mesh generation from the Delaunay triangulation (algorithm I), the mesh refinement procedure (algorithm II), and the adaptive remeshing technique (algorithm III). This main algorithm is developed using the object-oriented

programming concept that takes into account the advantages of the encapsulation, inheritance, and polymorphism capabilities.

In addition, the main algorithm has incorporated a scheme for mesh generation around the crack tips. The crack tip nodes are first removed from the node-list prior to the domain discretization. Such scheme avoids the modification of the original Delaunay triangulation (algorithm I) and the mesh refinement procedure (algorithm II) previously described. Then the rosette nodes around the crack tips, with specified angle by the parameter *angle_increment*, are generated as depicted in Fig. 2 and added to the node-list. After the domain has been discretized, the crack tip nodes are inserted to form the rosette elements around the crack tips. The implementation of the main algorithm is summarized in the algorithm IV below.

Algorithm IV; Main(*P, T, angle_increment, alpha, beta, iteration, Hmin, Hmax, threshold, isadaptive*)

```
Let BP be the collection of boundary
node objects that stored in sequence
of counter-clockwise direction for all
outside boundaries and clockwise
direction for all inside boundaries;
Let P0 be the collection of background
node objects;
Let P be the collection of node
objects;
Let T be the collection of mesh
objects;
Let angle_increment be the isosceles
triangle crack tip angle;
Let alpha be the constant that
controls shape of formed triangles;
Let beta be the constant that controls
regularity of the triangulation;
Let iteration be the number of loops
to refine meshes;
Let Hmin and Hmax be the minimum and
maximum element size, respectively;
Let threshold be the number of minimum
increasing nodes for each iteration;
Let isadaptive be the flag to generate
background or adaptive meshes;
BP.Initialize;
P0.Initialize;
P.Initialize;
T.Initialize;

If (isadaptive) {
    P0.ReadBackgroundNodes;
    BP.RediscretizeBoundaryNodes;
    };
Else {
    BP.ReadBoundaryNodes;
};
BP.MarkCrackTipNodes;
BP.CreateRosetteNodesAroundCrackTips(a
ngle_increment);
BP.CreateConvexHull;
P.AddNode(BP.p1, BP.p2, BP.p3, BP.p4);
T.AddTriangle(t1, BP.p1, BP.p2,
BP.p3);
T.AddTriangle(t2, BP.p3, BP.p2,
BP.p4);

Do p ← BP.NextBoundaryNode {
    Call DelaunayTriangulation(P, T,
p);
};
T.RemoveOutsideDomainTriangles;

Call MeshRefinement(P, T, alpha, beta,
iteration);
If (isadaptive)
        Call AdaptiveRemeshing(P, T,
P0, alpha, beta, _
Hmin, Hmax, threshold);
T.MeshRelaxation;
T.LaplaceSmoothing;

Do p ← BP.NextCracktipNode

T.CreateRosetteTriangularElements(p);
};
End;
```

A demonstration of a new point creation inside a triangle with automatic point creation procedure (Algorithm II) and the

formation of new triangles by Delaunay triangulation algorithm (Algorithm I) are shown in Fig. 3. The values of both the Alpha and the Beta coefficients that are 0.5 and 0.6, respectively.

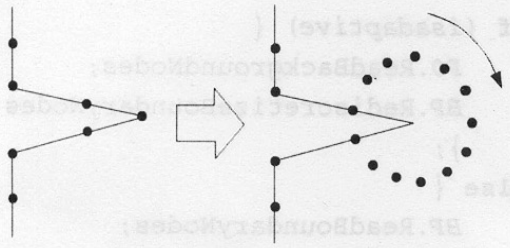

Fig. 2. Removal of the crack tip node and creation of rosette nodes around the crack tip

4. Algorithm Evaluation

The fracture mechanics simulation with finite element program is used to evaluate the efficiency of the combined Delaunay triangulation and the adaptive remeshing technique. The entire procedure is first used to determine the stress intensity factors for problems with analytical solutions or experimental data so that their results can be compared. The procedure is then employed to capture the crack trajectory by adapting the mesh automatically with the crack growth.

4.1 Determination of Stress Intensity Factors

The geometry of the single edge cracked plate and its final adaptive mesh are shown in Fig. 4. The plate has an initial crack length a = 3.5 mm. The modulus of elasticity and the Poisson's ratio are 30×10^6 units and 0.25, respectively. The plane strain condition is assumed in the analysis. The computed stress intensity factors K_I and K_{II} from the adaptive mesh are 33.99 and 4.57 comparing to the reference values of 34.00 and 4.55 [12], respectively.

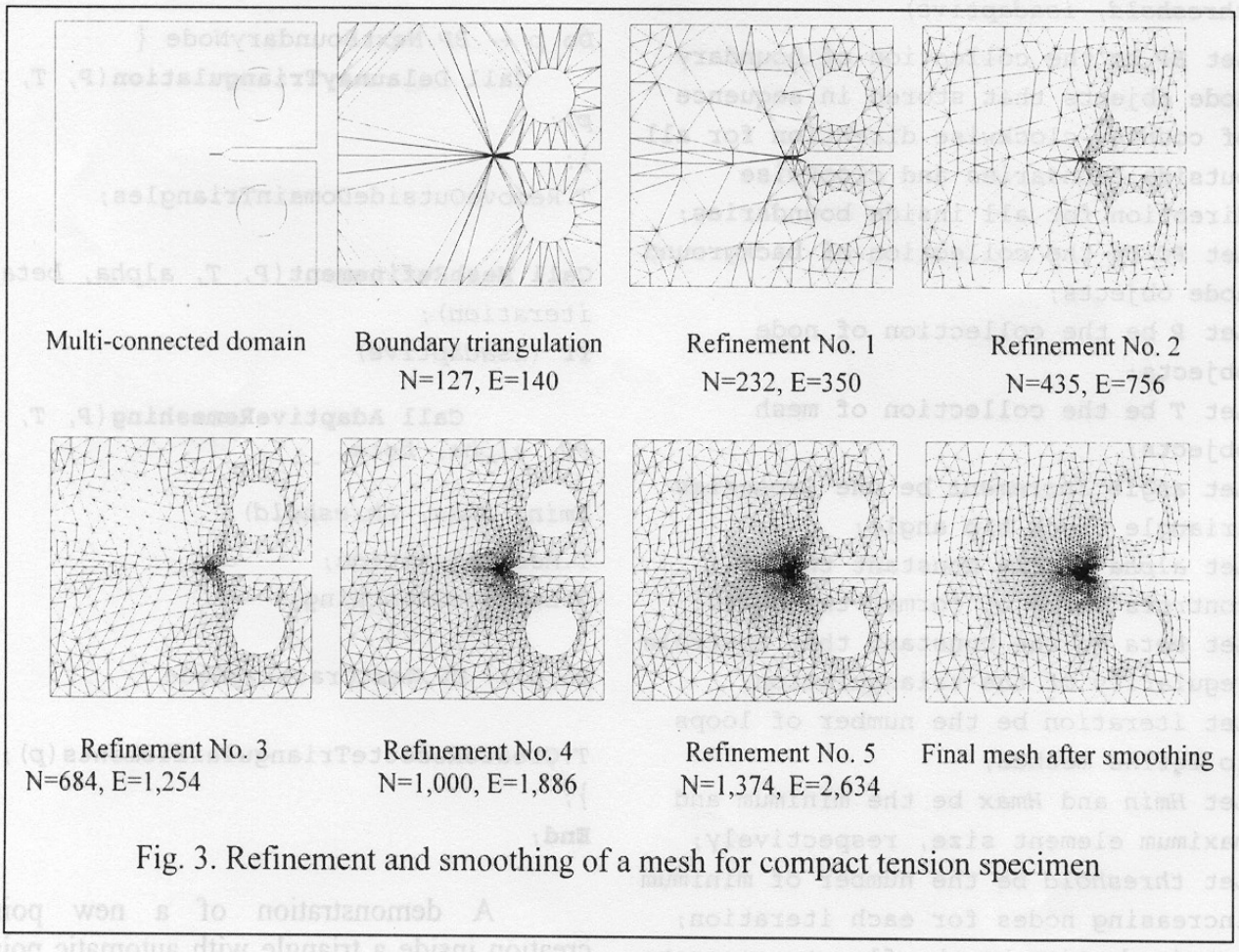

Fig. 3. Refinement and smoothing of a mesh for compact tension specimen

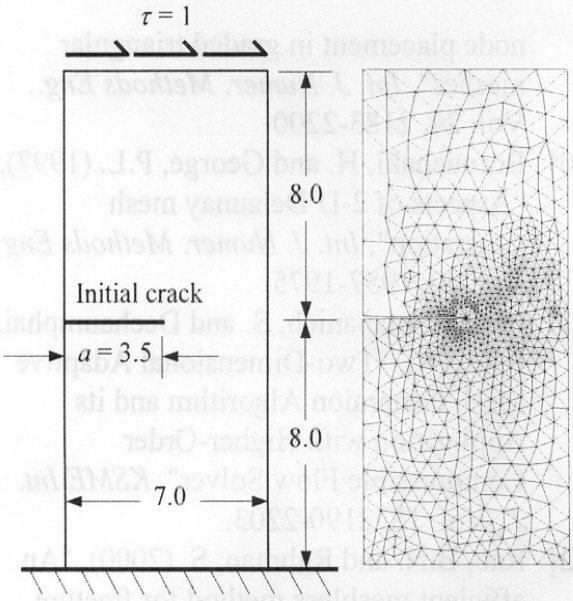

Fig. 4. Problem statement and the final mesh of the single edge crack plate

4.2 Simulation of Crack Propagation

To simulate the crack propagation, the first case of the experiment [13] was carried out in this paper. Figure 5 shows the problem statement with an initial crack length, a, and its location, b, are 1.0 and 4.0 units, respectively. The results of the adaptive finite element meshes and the crack growth trajectory are depicted in Fig. 6. The figures show that the crack growth trajectory passes near the lower hole and ended at the middle hole. The predicted crack growth trajectory resemble very well with the experimental results.

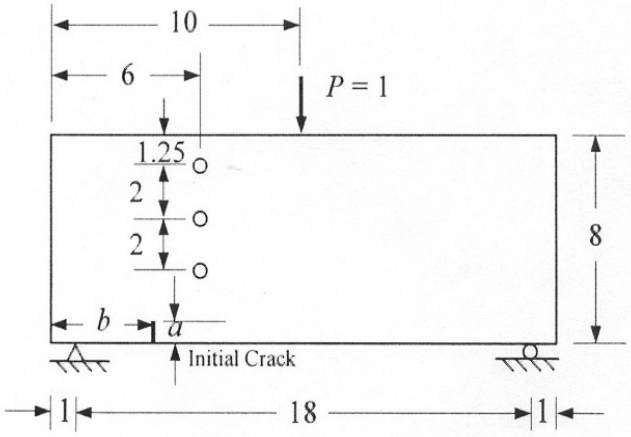

Fig. 5. Problem statement for the simulation of crack propagation

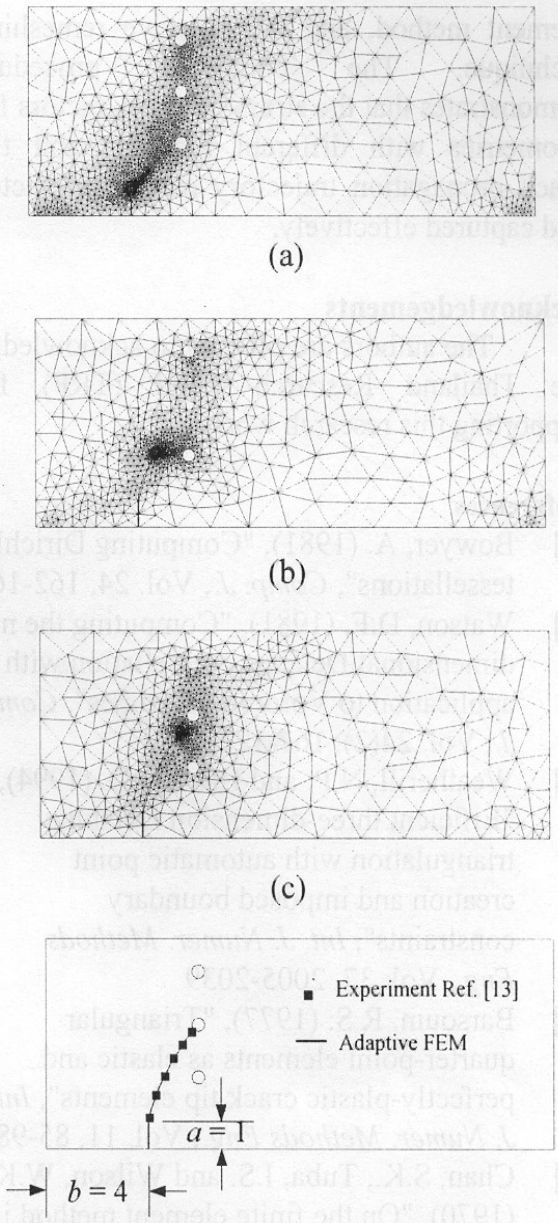

Fig. 6. Adaptive finite element meshes and the crack growth trajectory for the single edge cracked plate with three holes under mixed-mode loading

5. Conclusions

Delaunay triangulation was combined with the finite element method and the adaptive remeshing technique for analysis of crack problems under mixed mode loading. The concepts of the mesh generation and the adaptive technique for two-dimensional domain using object-oriented programming were presented in details. Several examples were employed to evaluate the accuracy of the combined Delaunay triangulation, the finite

element method, and the adaptive remeshing technique. The combined procedure demonstrates that the stress intensity factors for geometries with different loadings and the crack propagation trajectory can be predicted and captured effectively.

Acknowledgements

The authors are pleased to acknowledge the Thailand Research Fund (TRF) for supporting this research work.

References

[1] Bowyer, A. (1981), "Computing Dirichlet tessellations", *Comp. J.*, Vol. 24, 162-166

[2] Watson, D.F. (1981), "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes", *Comp. J.*, Vol. 24(2), 167-172

[3] Weatherill, N.P. and Hassan, O. (1994), "Efficient three-dimension Delaunay triangulation with automatic point creation and imposed boundary constraints", *Int. J. Numer. Methods Eng.*, Vol. 37, 2005-2039

[4] Barsoum, R.S. (1977), "Triangular quarter-point elements as elastic and perfectly-plastic crack tip elements", *Int. J. Numer. Methods Eng.*, Vol. 11, 85-98.

[5] Chan, S.K., Tuba, I.S. and Wilson, W.K. (1970), "On the finite element method in linear fracture mechanics", *Eng. Fract. Mech.*, Vol. 2, 1-17

[6] Erdogan, F. and Sih, G.C. (1963), "On the crack extension in plates under plane loading and transverse shear", *J. Basic Eng.*, Vol. 85, 519-527

[7] Guinea, G.V., Planas, J. and Elices, M. (2000), "K_I evaluation by the displacement extrapolation technique", *Eng. Fract. Mech.*, Vol. 66, 243-255

[8] Phongthanapanich, S. and Dechaumphai, P. (2004), "Evaluation of combined Delaunay triangulation and remeshing for Finite element analysis of conductive heat transfer", *Trans. of CSME*, Vol. 27, 319-339.

[9] Frey, W.H. (1987), "Selective refinement: a new strategy for automatic node placement in graded triangular meshes", *Int. J. Numer. Methods Eng.*, Vol. 24, 2183-2200

[10] Borouchaki, H. and George, P.L. (1997), "Aspects of 2-D Delaunay mesh generation", *Int. J. Numer. Methods Eng.*, Vol. 40, 1957-1975

[11] Phongthanapanich, S. and Dechaumphai, P. (2004), "Two-Dimensional Adaptive Mesh Generaion Algorithm and its Application with Higher-Order Compressible Flow Solver", *KSME Int. J.*, Vol. 18, 2190-2203.

[12] Rao, B.N. and Rahman, S. (2000), "An efficient meshless method for fracture analysis of cracks", *Comput. Mech.*, Vol. 26, 398-408

[13] Bittencourt, T.N., Wawrzynek, P.A., Ingraffea, A.R. and Sousa, J.L. (1996), "Quasi-automatic simulation of crack propagation for 2D LEFM problems", *Eng. Fract. Mech.*, Vol. 55, 321-334