A New Triangular Mesh Generation Technique

Logah Perumal

Abstract—**Objective of this paper is to propose a new semi-automatic, adaptive and optimized triangular mesh generation technique for any domain (including free formed curves). This new technique is found by merging the generalised equations which were proposed in previous works with Delaunay triangulation method. The new technique is demonstrated for several domains with various boundaries. Initial meshes are generated for these domains, which are later optimized manually by addition, removal or replacement of sampling points. Finalized meshes consist of triangular elements with aspect ratio of less than 2 and minimum skewness of more than 45 degrees.**

Index Terms—**Generalised equations, numerical integration, Delaunay triangulation, mesh optimization, skewness and aspect ratio.**

I. Introduction

One of the steps involved in Finite Element Method (FEM) is the discretization of the problem domain (mesh creation). There are various methods proposed to generate triangular meshes for FEM. The most commonly used methods are the moving front technique also known as advancing front, Delaunay based methods and the Octree approach [1]. Recently, attempts to mesh domains with curved sides have been pursued by using higher order (HO) triangular elements [2]-[4]. In these methods, the author proposed HO triangular elements with two linear sides and one curved side. The single curved side (of the HO triangular element) is represented by a polynomial arc of arbitrary order, which is used to represent the curved boundaries of the problem domain. Later, a fully automated mesh generation code in Matlab has been developed for these HO triangular elements [5], by merging with triangular mesh generation techniques proposed by [6] or [7].

It is observed that the technique proposed by [5] has advantages such as higher accuracy in simulation results with lesser number of elements, accurate representation of curved sides of problem domain and simpler Matlab code for fully automated mesh generation (reduces the time and effort for the mesh generation). However, there are limitations in this method. The arbitrary curves of a problem domain should be able to be represented by polynomial arcs. Triangular element with high aspect ratio and skewness could also be generated as seen in Fig. 27- Fig. 29 of the paper [5], which can contribute to high stiffness for certain applications such as in solid mechanics. Therefore, mesh optimization needs to be carried out after the initial mesh generation, in order to untangle the flawed elements in the mesh. Several mesh improvement or optimization techniques are available in the literature and these techniques can be categorised as adaptive, smoothing and swapping [8].

Adaptive mesh optimization involves generation of high number of elements (fine mesh) within a specific region while the rest of the domain contains lesser number of elements (coarse mesh). This is done by using nested grid meshes. This technique does not focus on alteration of the element parameters, but rather focuses on improving the accuracy of the simulation results at regions in which the rate of change of field variables is high. This technique is also implemented to improve accuracy in geometry representation of a problem domain.

On the other hand, smoothing involves alteration of elements' shape via relocation of the nodes within the mesh without changing the element connectivity. The nodes are relocated by arithmetic mean of connected nodes (known as Laplacian approach). Another way to relocate the nodes is by using certain algorithms to meet the predefined standards that are related to mesh quality (known as optimization-based approach). Third method to relocate the nodes is by assuming that the mesh is a deformable media and forces are applied to achieve the optimal element shape (known as physics-based approach).

Swapping is yet another technique to improve mesh quality by changing the shape of elements. Alteration of the element shape is done through operations such as edge and face swapping. This technique changes the element connectivity in the mesh.

In this work, a new semi-automatic approach for triangular mesh generation is proposed by merging the generalized equations developed in previous works [9]-[11] with Delaunay triangulation algorithm in Mathematica. Elements in the initial mesh are later optimized by relocation, addition and removal of nodes. Advantages of this proposed method are that it is applicable for adaptive mesh generation, aspect ratio and skewness of the elements are within the desired range (attained through manual optimization), applicable for any order of triangular element (achieved through subparametric mapping), and the meshes can be formed for domain with any curves, including free formed curves. Disadvantages of this technique are that it requires effort and the process is time consuming (especially during the manual optimization), free formed curves are approximated by linear lines and for this case (in the case of mesh generation for

Manuscript received March 22, 2019; revised June 12, 2019. This work was supported in part by the Research Management Centre (RMC) of Multimedia University, Malaysia under Grant nos. MMUI/130070 and MMUI/160047, which enabled purchase of required software and equipment for this work.

Logah Perumal is with the Faculty of Engineering and Technology, Multimedia University, Jalan Ayer Keroh Lama, Bukit Beruang, 75450 Melaka, Malaysia (e-mail: logah.perumal@mmu.edu.my).

doi: 10.18178/ijmlc.2019.9.5.852

domain with free formed curves), sample boundary points on these free formed curves are required. The proposed approach is described in the following sections.

II. Details of the Method

A. Generalised Equations for Numerical Integration

Generalised equations were proposed in previous works [9]-[11] to enable numerical or exact integration of functions over arbitrary domains. The generalized equation for numerical integration within two dimensional (2D) domains is recalled here, as shown by (1):

$$\left.\begin{array}{l} I_1 = \int_a^b \int_{r(x)}^{s(x)} f(x,y)\,dy\,dx \\ \qquad or \\ I_2 = \int_a^b \int_{r(y)}^{s(y)} f(x,y)\,dx\,dy \end{array}\right\} = \int_L^U \int_L^U f(m_x u + c_x, m_y v + c_y)\, m_x\, m_y\, dv\, du = \sum_j^n \sum_i^n w_i w_j m_x\, m_y\, f(m_x u_i + c_x, m_y v_i + c_y)$$

where
U is upper limit
L is lower limit
w_i *and* w_j *are integration weights*
u_i *and* v_j *are integration points*
$i = 1,2,3,\ldots,n$
n is integration order

for I_1 *:* (1)

$$m_x = \frac{a-b}{L-U}; \quad m_y = \frac{r(m_x u + c_x) - s(m_x u + c_x)}{L-U};$$

$$c_x = \frac{(b\times L)-(a\times U)}{L-U}; \quad c_y = \frac{(s(m_x u + c_x)\times L)-(r(m_x u + c_x)\times U)}{L-U}$$

for I_2 *:*

$$m_x = \frac{r(m_y v + c_y) - s(m_y v + c_y)}{L-U}; \qquad m_y = \frac{a-b}{L-U};$$

$$c_x = \frac{(s(m_y v + c_y)\times L)-(r(m_y v + c_y)\times U)}{L-U}; \qquad c_y = \frac{(b\times L)-(a\times U)}{L-U};$$

Equation (1) can be used to perform numerical integration within 2D arbitrary domains, by defining the integration limits of (1) (constants *a* and *b* as well as functions *r* and *s*), according to the enclosure of the problem domain. Domain to be integrated can be enclosed by:

1) 4 constant lines
2) 3 constant lines and 1 function (or inclined line)
3) 2 constant lines and 1 function (or inclined line)
4) 2 parallel constant lines and 2 functions (or inclined line)

The numerical integration is then performed by converting the integration limits (constants *a* and *b* as well as functions *r* and *s*) to a fixed intervals *U* and *L*, according to the numerical integration scheme of choice [9]-[11]. In this work, (1) is used together with Delaunay triangulation algorithm in Mathematica to generate triangular meshes. The new triangular mesh generation technique is described in following sections.

B. Delaunay Triangulation through Linear Mapping

Delaunay triangulation generates triangular elements based on sample points which are located on the boundaries and at the interior of a domain. These sample points are connected to each other to form triangles, based on the principal that no other points should present within circumcircle associated to a particular triangle. Therefore, suitable sample points for a domain to be discretized need to be generated first. These sample points can be obtained through linear mapping described by (1). For this purpose, two types of parent rectangular domains (within range of -1 to 1) with different number of parent sample points (SP) are created and denoted as SP13 and SP41, respectively (shown in Fig. 1). These parent rectangular domains can be used interchangeably to form coarse or/and fine meshes within a domain. Details are provided in the upcoming examples 1-4.

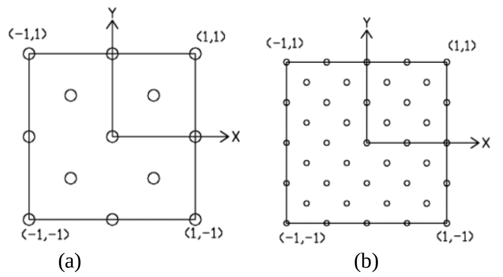


Fig. 1. Parent rectangular domains with sample points (a) SP13 (b) SP41.

C. Mesh Quality

Among various parameters which represent quality of a triangular mesh, aspect ratio and skewness are the most important aspects which should be payed attention to. Recent attempt to improve quality of triangular meshes based on aspect ratio can be seen in [12]. In this work, both aspect ratio and skewness are optimized manually, through Mathematica platform. Highest aspect ratio among the elements in all the optimized meshes considered here is less than 2 and minimum skewness angle is greater than 45 degrees. The aspect ratio (AR) is calculated based on the formulas (2) and (3):

$$\text{AR} = \frac{abc}{8(s-a)(s-b)(s-c)}, \tag{2}$$

$$s = \frac{1}{2}(a+b+c) \tag{3}$$

a, *b*, and *c* represent length of each side of a triangular element. The skewness is determined by calculating the lowest angle formed by intersection of two lines within the triangle. The first line is drawn such a way that it connects one of the triangular nodes to the midpoint of opposite side. The second line connects midpoints of the other two sides. Since there are 3 nodes for a triangular element, therefore total of three combinations of such intersection can be obtained. These combinations are shown in Fig. 2. Skewness for a particular triangular element is represented by the smallest angle which is formed by the intersections in the three configurations shown in Fig. 2.

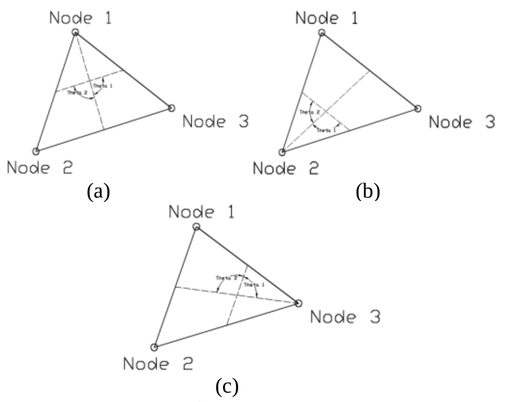


Fig. 2. Three combinations of intersections within a triangular element.

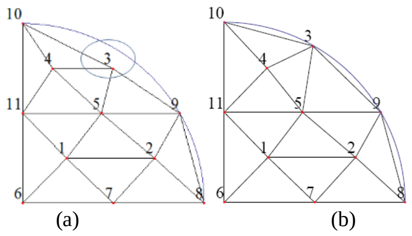


Fig. 3. Coarse triangular meshes for the quadrant of a circle (a) Initial mesh (b) Enhanced mesh.

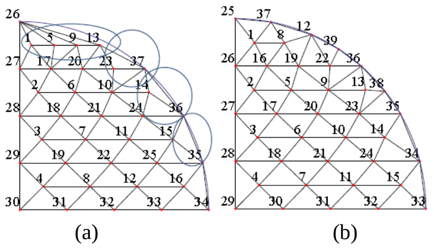


Fig. 4. Fine triangular meshes for the quadrant of a circle (a) Initial mesh (b) Enhanced mesh.

III. Results and Discussions

A. *Example 1: Quadrant of a Circle*

The technique is demonstrated here for a quadrant of a circle which consists of two linear sides and a curve. SP13 and SP41 are mapped individually to the problem domain by letting U = 1, L = -1, $r(y)$ = 0 and $s(y) = \sqrt{1-y^2}$ into (1). Initial triangular meshes are then formed by using the Mathematica command *Delaunay Triangulation*. The initial meshes together with the actual domain boundary for both coarse (obtained from mapping of SP13) and fine (obtained from mapping of SP41) meshes are shown in Fig. 3(a) and Fig. 4(a). It is seen that the actual domain boundary is not accurately captured in the coarse mesh (node 3 in Fig. 3(a)) while the fine mesh consists of elements with high aspect ratio and skewness (highlighted in Fig. 4(a)).

Therefore, the initial meshes need to be enhanced in order to ensure that the meshes are suitable for FEM. Aspect ratio and skewness of each element are calculated for the initial meshes in order to determine regions/elements which are in need for enhancement. One of the advantages of Delaunay triangulation is that the method allows addition, deletion and replacement of sample points within the domain to enhance the mesh. Hence, the affected elements/regions are then enhanced by manually adding, deleting or replacing nodes. The initial coarse mesh is improved by moving node 3 to the surface of the domain while the initial fine mesh is improved by addition and replacement of nodes at the affected regions. The improved meshes for both cases (coarse and fine meshes) are shown in Fig. 3(b) and Fig. 4(b).

B. *Example 2: Quadrant of a Hollow Cylinder*

Quadrant of a hollow cylinder with two polynomial curved boundaries is meshed by using the proposed method. The domain is partitioned into two regions (R1 and R2) as shown in Fig. 5, based on the requirements stated previously. R1 and R2 are meshed individually and later combined together. Initial meshes obtained by mapping the SP13 and SP41 are shown in Fig. 6(a) and Fig. 7(a). The initial coarse mesh is optimized by moving nodes 13 and 21, and adding node 23. The finalized meshes are shown in Fig. 6(b) and Fig. 7(b).

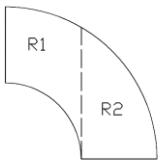


Fig. 5. Partitioning of the domain into 2 regions.

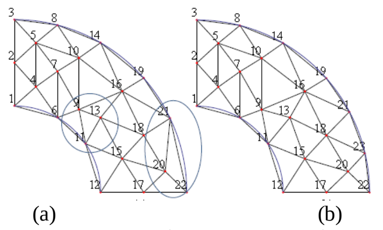


Fig. 6. Coarse triangular meshes for the quadrant of hollow cylinder (a) Initial mesh (b) Enhanced mesh.

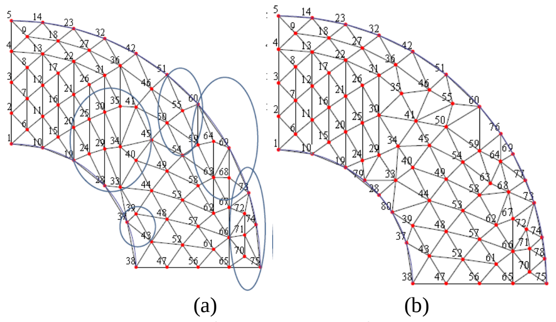


Fig. 7. Fine triangular meshes for the quadrant of hollow cylinder (a) Initial mesh (b) Enhanced mesh.

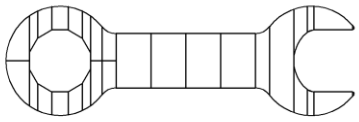


Fig. 8. Partitioning of a wrench into several regions for adaptive mesh creation.

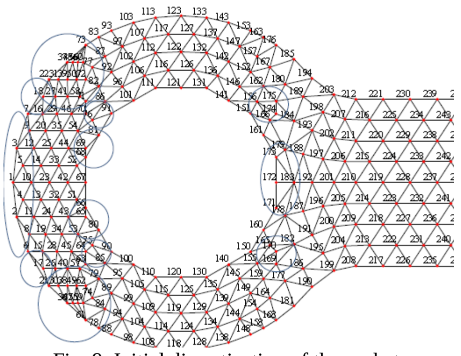


Fig. 9. Initial discretization of the socket.

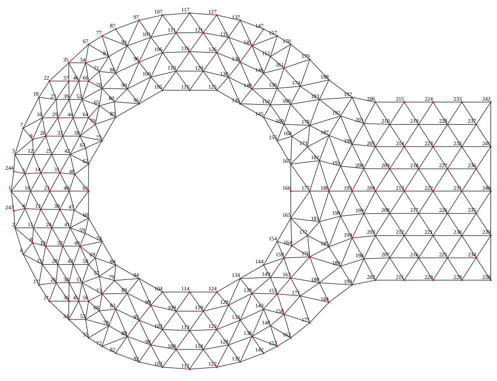


Fig. 10. Optimized discretization of the socket.

C. Example 3: Adaptive Meshing for a Wrench

A wrench, consisting of head and socket as shown in Fig. 8 is taken as an example to demonstrate adaptive meshing by using the proposed technique. The tip of the head need to be discretized by using high number of small elements in order to capture the curved tip accurately, compared to the other parts of the wrench. First, the domain is discretized into several regions as shown in Fig. 8. SP13 and SP41 are interchangeably used to obtain the sample points for these regions as well as to obtain the adaptive mesh for the tip of the head. Initial and final meshes for the socket and the head are shown in Fig. 9-14.

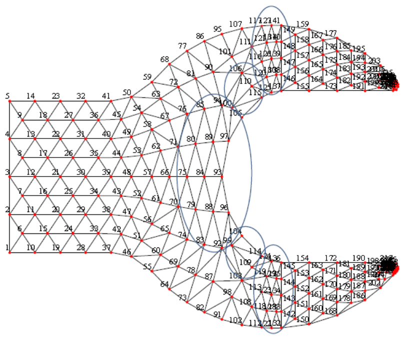


Fig. 11. Initial discretization of the head.

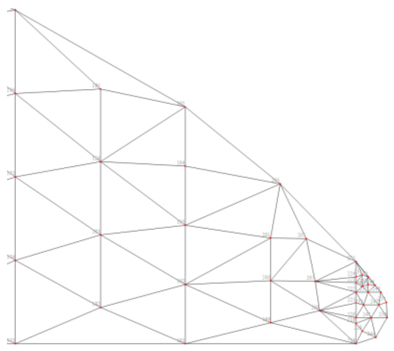


Fig. 12. Initial discretization of the tip of the head (close up view).

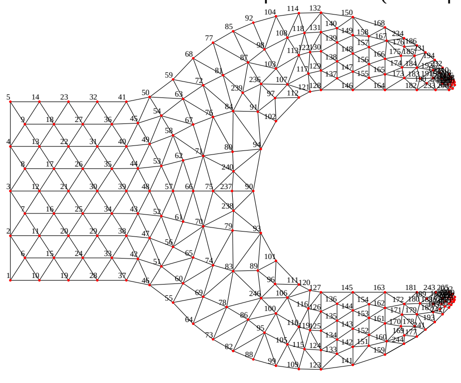


Fig. 13. Optimized discretization of the head.

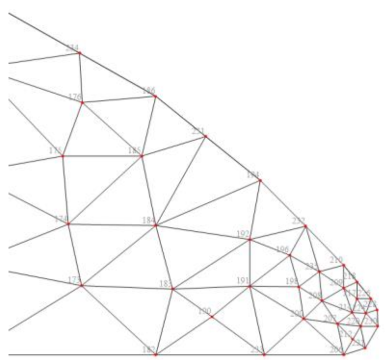


Fig. 14. Optimized discretization of the tip of the head (close up view).

D. Example 4: Free Formed Curves

Mesh creation for a domain with free formed curves (Fig. 15) is demonstrated here. The domain is partitioned into several regions (R1-R18) as shown in Fig. 16. Since the free formed curves are not represented by any functions, therefore these curves are initially approximated as linear lines to generate initial meshes. Later, the boundary surfaces (free formed curves) are discretized to generate the sample points for the optimization of the meshes near the boundaries (R1-R14). Discretization of the free formed curves is indicated by small circles in the Fig. 16. Initial meshes are optimized meshes are shown in Fig. 17 and Fig. 18.

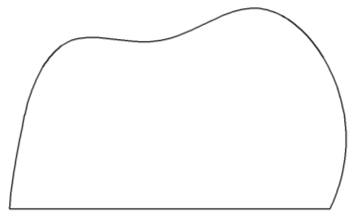


Fig. 15. A domain with free formed curves and a linear boundary.

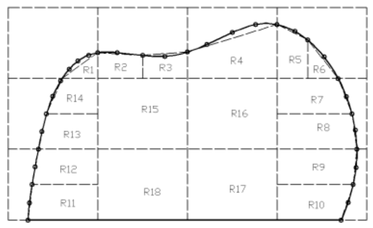


Fig. 16. Partitioning of the domain with free formed and linear boundaries.

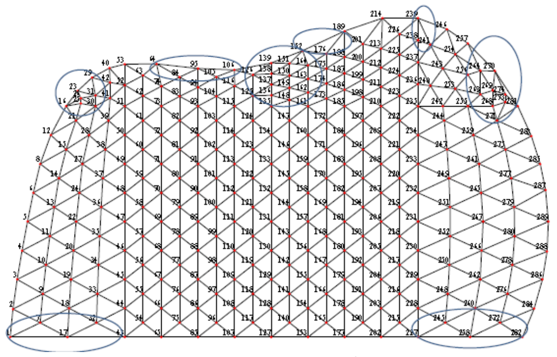


Fig. 17. Initial mesh generation for the domain.

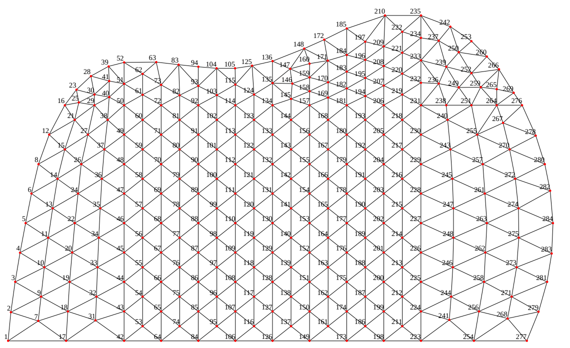


Fig. 18. Finalized mesh for the domain.

E. Mesh for Higher Order Triangular Elements

The optimized meshes created in all the examples above can be used for HO triangular elements as well, by mapping the parent element nodes to the elements in the mesh. Shape functions of a 3 nodes triangular element as shown in (1) can be used for the mapping (subparametric mapping for HO triangular elements). Examples for mapping of 3 nodes, 6 nodes and 10 nodes triangular elements are shown in Fig. 19-21.

$$x = N_1x_1 + N_2x_2 + N_3x_3; \qquad y = N_1y_1 + N_2y_2 + N_3y_3 \quad (4)$$
$$N_1 = s \qquad N_2 = 1 - r - s \qquad N_3 = r$$

N_1, N_2 and N_3 represent shape functions of parent 3 nodes triangular element in natural coordinate system (in r, s). x_i and y_i (i = 1, 2, 3) represent coordinates of the vertex of the elements in global coordinate system (in x, y).

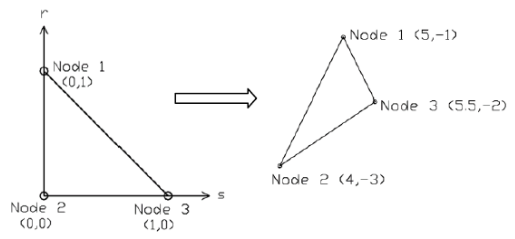


Fig. 19. Isoparametric mapping of the nodes for 3 nodes triangular element.

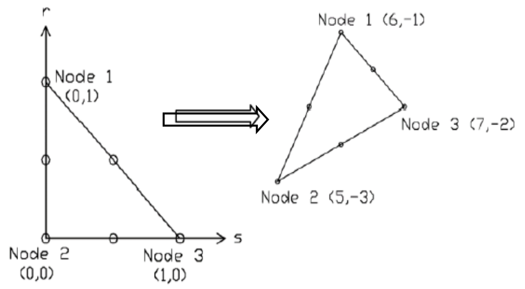


Fig. 20. Subparametric mapping of the nodes for 6 nodes quadratic triangular element.

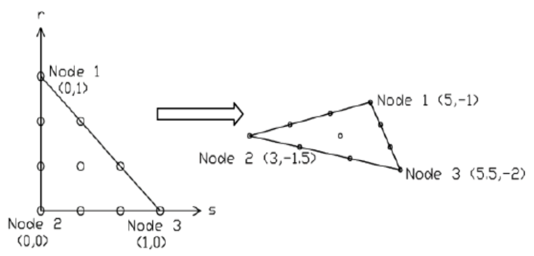


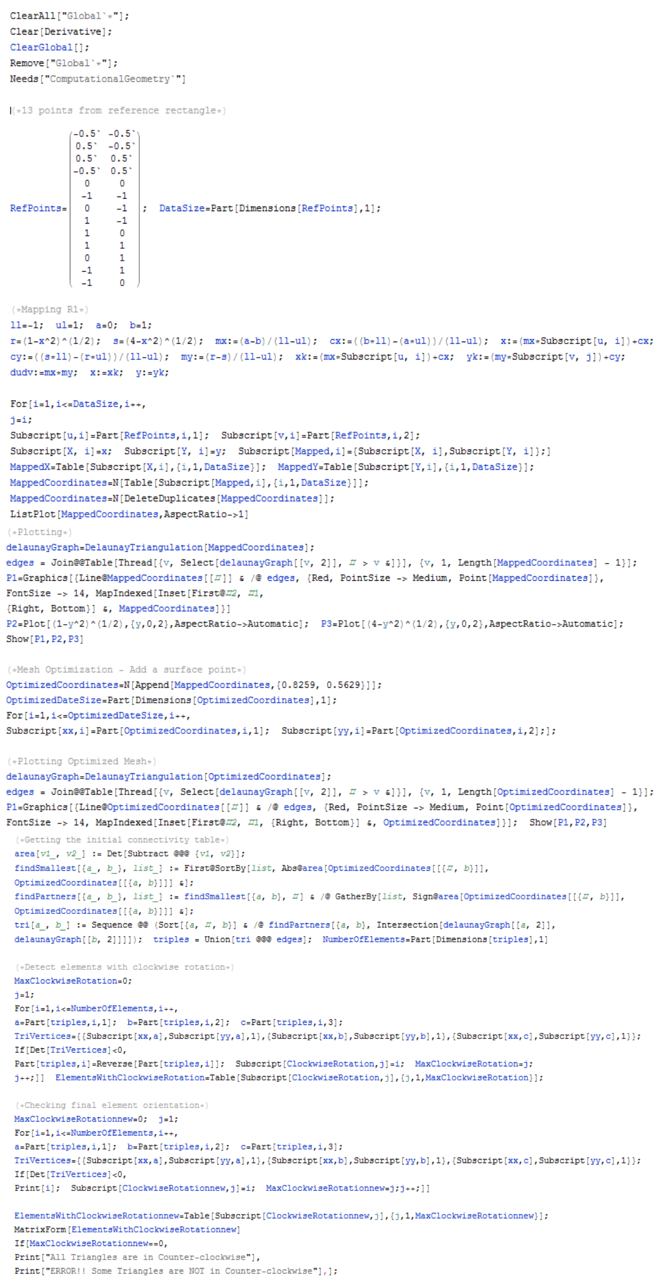
Fig. 21. Subparametric mapping of the nodes for 10 nodes cubic triangular element.

IV. Conclusions

A new technique to generate triangular meshes for arbitrary geometry is demonstrated within Mathematica platform by coupling the generalized equation (1) with Delaunay triangulation. The meshes are later optimized manually (by adding, removing or moving nodes) such that highest aspect ratio among the elements is less than 2 and minimum skewness angle is greater than 45 degrees. The method is shown to be applicable for any arbitrary domain, and for adaptive mesh. The method can be used for any order of triangular element as well, by linear mapping. Future work would be to minimize the effort and time spent on the manual optimization by incorporating mesh improvement algorithms into the execution code.

Appendix

Mathematica code to generate mesh for hollow cylinder by using the proposed technique (example 2) is provided here:



```
ClearAll["Global`*"];
Clear[Derivative];
ClearGlobal[];
Remove["Global`*"];
Needs["ComputationalGeometry`"]

(*13 points from reference rectangle*)
            -0.5`  -0.5`
             0.5`  -0.5`
             0.5`   0.5`
            -0.5`   0.5`
              0      0
             -1     -1
RefPoints=    0     -1   ;  DataSize=Part[Dimensions[RefPoints],1];
              1     -1
              1      0
              1      1
              0      1
             -1      1
             -1      0

(*Mapping R1*)
ll=-1;  ul=1;  a=0;  b=1;
r=(1-x^2)^(1/2);  s=(4-x^2)^(1/2);  mx:=(a-b)/(ll-ul);  cx:=((b*ll)-(a*ul))/(ll-ul);  x:=(mx*Subscript[u, i])+cx;
cy:=((s*ll)-(r*ul))/(ll-ul);  my:=(r-s)/(ll-ul);  xk:=(mx*Subscript[u, i])+cx;  yk:=(my*Subscript[v, j])+cy;
dudv:=mx*my;  x:=xk;  y:=yk;

For[i=1,i<=DataSize,i++,
j=i;
Subscript[u,i]=Part[RefPoints,i,1];  Subscript[v,i]=Part[RefPoints,i,2];
Subscript[X, i]=x;  Subscript[Y, i]=y;  Subscript[Mapped,i]={Subscript[X, i],Subscript[Y, i]};]
MappedX=Table[Subscript[X,i],{i,1,DataSize}];  MappedY=Table[Subscript[Y,i],{i,1,DataSize}];
MappedCoordinates=N[Table[Subscript[Mapped,i],{i,1,DataSize}]];
MappedCoordinates=N[DeleteDuplicates[MappedCoordinates]];
ListPlot[MappedCoordinates,AspectRatio->1]
(*Plotting*)
delaunayGraph=DelaunayTriangulation[MappedCoordinates];
edges = Join@@Table[Thread[{v, Select[delaunayGraph[[v, 2]], # > v &]}], {v, 1, Length[MappedCoordinates] - 1}];
P1=Graphics[{Line@MappedCoordinates[[#]] & /@ edges, {Red, PointSize -> Medium, Point[MappedCoordinates]},
FontSize -> 14, MapIndexed[Inset[First@#2, #1,
{Right, Bottom}] &, MappedCoordinates]}]
P2=Plot[(1-y^2)^(1/2),{y,0,2},AspectRatio->Automatic];  P3=Plot[(4-y^2)^(1/2),{y,0,2},AspectRatio->Automatic];
Show[P1,P2,P3]

(*Mesh Optimization - Add a surface point*)
OptimizedCoordinates=N[Append[MappedCoordinates,{0.8259, 0.5629}]];
OptimizedDateSize=Part[Dimensions[OptimizedCoordinates],1];
For[i=1,i<=OptimizedDateSize,i++,
Subscript[xx,i]=Part[OptimizedCoordinates,i,1];  Subscript[yy,i]=Part[OptimizedCoordinates,i,2];];

(*Plotting Optimized Mesh*)
delaunayGraph=DelaunayTriangulation[OptimizedCoordinates];
edges = Join@@Table[Thread[{v, Select[delaunayGraph[[v, 2]], # > v &]}], {v, 1, Length[OptimizedCoordinates] - 1}];
P1=Graphics[{Line@OptimizedCoordinates[[#]] & /@ edges, {Red, PointSize -> Medium, Point[OptimizedCoordinates]},
FontSize -> 14, MapIndexed[Inset[First@#2, #1, {Right, Bottom}] &, OptimizedCoordinates]}];  Show[P1,P2,P3]
 (*Getting the initial connectivity table*)
 area[v1_, v2_] := Det[Subtract @@@ {v1, v2}];
 findSmallest[{a_, b_}, list_] := First@SortBy[list, Abs@area[OptimizedCoordinates[[{#, b}]],
 OptimizedCoordinates[[{a, b}]]] &];
 findPartners[{a_, b_}, list_] := findSmallest[{a, b}, #] & /@ GatherBy[list, Sign@area[OptimizedCoordinates[[{#, b}]],
 OptimizedCoordinates[[{a, b}]]] &];
 tri[a_, b_] := Sequence @@ (Sort[{a, #, b}] & /@ findPartners[{a, b}, Intersection[delaunayGraph[[a, 2]],
 delaunayGraph[[b, 2]]]]);  triples = Union[tri @@@ edges];  NumberOfElements=Part[Dimensions[triples],1]

 (*Detect elements with clockwise rotation*)
 MaxClockwiseRotation=0;
 j=1;
 For[i=1,i<=NumberOfElements,i++,
 a=Part[triples,i,1];  b=Part[triples,i,2];  c=Part[triples,i,3];
 TriVertices={{Subscript[xx,a],Subscript[yy,a],1},{Subscript[xx,b],Subscript[yy,b],1},{Subscript[xx,c],Subscript[yy,c],1}};
 If[Det[TriVertices]<0,
 Part[triples,i]=Reverse[Part[triples,i]];  Subscript[ClockwiseRotation,j]=i;  MaxClockwiseRotation=j;
 j++;]]  ElementsWithClockwiseRotation=Table[Subscript[ClockwiseRotation,j],{j,1,MaxClockwiseRotation}];

 (*Checking final element orientation*)
 MaxClockwiseRotationnew=0;  j=1;
 For[i=1,i<=NumberOfElements,i++,
 a=Part[triples,i,1];  b=Part[triples,i,2];  c=Part[triples,i,3];
 TriVertices={{Subscript[xx,a],Subscript[yy,a],1},{Subscript[xx,b],Subscript[yy,b],1},{Subscript[xx,c],Subscript[yy,c],1}};
 If[Det[TriVertices]<0,
 Print[i];  Subscript[ClockwiseRotationnew,j]=i;  MaxClockwiseRotationnew=j;j++;]]

 ElementsWithClockwiseRotationnew=Table[Subscript[ClockwiseRotationnew,j],{j,1,MaxClockwiseRotationnew}];
 MatrixForm[ElementsWithClockwiseRotationnew]
 If[MaxClockwiseRotationnew==0,
 Print["All Triangles are in Counter-clockwise"],
 Print["ERROR!! Some Triangles are NOT in Counter-clockwise"],];
```

Acknowledgment

The author would like to thank Research Management Centre (RMC) of Multimedia University, Malaysia, for providing financial support through Mini Funds with Grant nos. MMUI/130070 and MMUI/160047, which enabled purchase of required software and equipment for this work.

References

[1] T. J. Baker, "Mesh generation: art or science?" *Progress in Aerospace Sciences*, vol. 41, no.1, pp. 29-63, January 2005.

[2] H. T. Rathod, K. V. Nagaraja, V. K. Naidu, and B. Venkatesh, "The use of parabolic arcs in matching curved boundaries by point transformations for some HO triangular elements," *Finite Elements in Analysis and Design*, vol. 44, pp. 920-932, August 2008.

[3] K. V. Nagaraja, V. K. Naidu, and P. G. Siddheshwar, "Optimal subparametric finite elements for elliptic partial differential equations using higher-order curved triangular elements," *International Journal for Computational Methods in Engineering Science and Mechanics*, vol. 15, no. 2, pp. 83-100, January 2014.

[4] M. Kardani, M. Nazem, J. P. Carter, and A. J. Abbo, "Efficiency of high-order elements in large-deformation problems of geomechanics," *International Journal of Geomechanics*, vol. 15, no. 6, pp. 1-10, December 2015.

[5] T.V. Smitha, K.V. Nagaraja, and J. Sarada, “MATLAB 2D higher-order triangle mesh generator with finite element applications using sub parametric transformations,” *Advances in Engineering Software*, vol. 115, no. C, pp. 327-356, January 2018.
[6] P. O. Persson and G. Strang, “A simple mesh generator in MATLAB,” *SIAM Review*, vol. 46, no. 2, pp. 329-345, June 2004.
[7] J. Koko, “A MATLAB mesh generator for the two-dimensional finite element method,” *Applied Mathematics and Computation*, vol. 250, pp. 650-664, January 2015.
[8] J. Park and S.M. Shontz, “An alternating mesh quality metric scheme for efficient mesh quality improvement,” *Procedia Computer Science,4*, 292-301, 2011.
[9] P. Logah and T.T. Mon, “Generalized equations for numerical integration over two dimensional domains using quadrature rules,” *Integration: Mathematical Theory and Applications*, vol. 3, no. 4, pp. 333-346, October 2012.
[10] P. Logah, “Integration techniques for two dimensional domains,” *International Journal of Research in Engineering and Technology*, vol. 3, no. 7, pp. 487-494, July 2014.
[11] P. Logah, C. P. Tso, and L. T. Leng, “Analysis of thin plates with holes by using exact geometrical representation within XFEM,” *Journal of Advanced Research*, vol. 7, no. 3, pp. 445-452, March 2016.
[12] H. Xuan and X. Dianna, “Aspect-Ratio Based Triangular Mesh Smoothing,” presented at the SIGGRAPH 2017 posters section, Los Angeles, California, July 30 – August 03, 2017.

Logah Perumal received his bachelor degree in mechanical engineering and master in mechanical engineering from Universiti Tenaga Nasional, Malaysia. Later, he obtained his PhD in engineering from Multimedia University, Malaysia. Currently he is working as a lecturer and attached with Faculty of Engineering and Technology at Multimedia University, Malaysia. His research interests include numerical methods, finite element method, computation and fuzzy logic. He has published research articles at national and international journals, conference proceedings, and contributed to chapter of a book.