Generation of graded triangular and quadrilateral mesh

Citation for published version (APA):

Document status and date:
Published: 01/01/1992

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 09. Nov. 2019
Generation of Graded Triangular and Quadrilateral Mesh

TUE-Research Report
Gaohong Xie

Oct. 1992  WPA 1409

Fac. of WPA, Dept. of Mechanical Engineering
Eindhoven University of Technology
P.O.Box 513, 5600 MB
Eindhoven, The Netherlands
GENERATION OF GRADED TRIANGULAR AND QUADRILATERAL MESH

G. Xie
Fac. of WPA, Dept. of Mechanical Engineering
Eindhoven University of Technology
P.O.Box 513, 5600 MB, Eindhoven, The Netherlands

SUMMARY

A simple and fast triangular and quadrilateral mesh generation method has been developed in this report. The method guarantees the generation of quadrilateral mesh as long as the triangular base mesh has been generated. To achieve the gradation of mesh around certain position, both internal and boundary control nodal spacing were employed. In order to ensure the quality of the quadrilateral mesh, firstly, the method try to generate equal sides triangles as possible with desired node spacing, secondly, the weighted central smoothing is applied to triangular base mesh and then transformed to quadrilateral mesh and smoothed again. Local area refinement is implemented as a post-process, which can be used to refine internal area by splitting the longest element edge into two. To demonstrate the efficiency of this method, some test results are presented for the comparing with existing ones.
## Contents

1 introduction 4

2 mesh generation 4
   2.1 generation of line elements 5
   2.2 data structure and local operations 6
   2.3 internal point control spacing 9
   2.4 post-processing 11
   2.5 implementation 11

3 program and user guide 12
   3.1 input records 15

4 results 17

5 conclusions 18
List of Figures

1 Line elements along the boundary and ordered nodes ........................................ 5
2 Topologically closing one angle, node i become inactive ........................................ 6
3 Bisect the angle, node i become inactive .............................................................. 7
4 Form a new triangle on the edge of reflex node ...................................................... 7
5 3 situations of the candidate point j ....................................................................... 9
6 Divide a triangle into three quadrilaterals ............................................................. 11
7 Flow chart of mesh generation .................................................................................. 13
8 User points to represent curve ................................................................................ 14
9 Decompose a multi-connected region into simply connected ones ......................... 15
10 left mesh and right mesh from the reference ......................................................... 20
11 Meshes from current method .................................................................................. 20
12 Mesh of the square before swapping and smoothing .............................................. 21
13 Mesh of the square after swapping and smoothing ............................................... 21
14 Mesh of the square after local refinements ............................................................ 22
15 Mesh of a triangle ................................................................................................... 22
16 Base mesh of the square ......................................................................................... 23
17 Quadrilateral mesh of the square ........................................................................... 23
18 Triangular mesh of a v-shape geometry .................................................................. 24
19 Quadrilateral mesh of a v-shape geometry ............................................................ 24
20 Triangular mesh of square-hole ............................................................................... 25
21 Quadrilateral mesh of square-hole ......................................................................... 25
22 Triangular mesh of a deformed blank ..................................................................... 26
23 Quadrilateral mesh of a deformed blank ............................................................... 26
1 introduction

Domain discretization plays an important role in finite element analysis as well as in numerous engineering analysis. As a result, various strategies and techniques for generating mesh automatically have been proposed [1] [2] [3] [4]. The latest mesh adaptation techniques has made it possible to connect domain discretization with error analysis of the results. [1] [5]. This work was initiated for the remeshing and adapting purpose during the FEM analysis with some commercial softwares, such as ABAQUS, in which the user interface it supplied do not contain sufficient information to initiate the remeshing process.

In general, there are existing three kinds of mesh generation methods, namely, mapping method, internal node method which introduce all interior nodes before constructing triangles [1] and advancing front method [6], which use a background mesh to interpolate nodal spacing, stretching direction and stretching ratio, the current method falls into the later group. In advancing front method, without the background mesh, the gradation can only be achieved through boundary nodal spacing specification which in some cases cause bad gradation across the mesh. While for complex geometries, well graded mesh can be achieved by dividing region to be meshed into sub-regions, and in each region generate fairly average spacing mesh, the current method, however, use the similar idea proposed by Lo [7] which use internal nodal spacing to control the density distribution without many necessary sub-region divisions. The idea is for each internal point, the desired element size can be estimated according to the both boundary and internal nodal spacing, which reflects the user desired nodal spacing around the specified points. In this report, the methodology and efficient techniques used in the development were described.

2 mesh generation

The problem of meshing in two-dimensional case is defined as:

Given a 2D domain $\Omega$, together with a node spacing function $h = h(x, y)$, $\forall(x, y) \in \Omega$, defined over the entire domain $\Omega$, the task of meshing is to discretize the domain $\Omega$ into triangles $T$ or quadrilaterals $Q$ or combination of
both in consistency with the given node spacing function $h$.

The node spacing function $h(x, y)$, which specifies the element size of the discretization can be defined based on the consideration of the current analysis interests e.g. loading concentrations, boundary conditions etc. The first step in mesh generation is to divide the region into several disjoint sub-regions by openings to holes or connector to holes. For simplicity, here we assume that the region is composed of only straight line and region is simply connected. The region boundaries are characterised by physical boundaries and internal boundaries (openings or connectors). For internal boundaries, the nodes on them can be repositioned during the smoothing and swapping process.

### 2.1 generation of line elements

After the definition of all boundary lines, line elements can be generated along those boundaries, which are generated in consistency with specified nodal spacing. The node insertion was done by means of geometrical progression:

$$S_n = \sum_{i=1}^{n} ad^{i-1} \quad (1)$$

where $S_n$ is the length of line segment, and $a$, $d$ and $n$ are selected such that:

$$a \approx l_f \quad and \quad ad^{n-1} \approx l_l \quad (2)$$

$l_f$ and $l_l$ are the first and last element length of segment. The nodal numbers of nodes inserted are assigned sequentially, therefore they are ordered integer set which will be used in constructing data structure, as shown in Fig.1.

![Figure 1: Line elements along the boundary and ordered nodes](image-url)
2.2 data structure and local operations

For each sub-region or surface (simply connected boundary contour loop), a temporary array is defined in such a way that the first index points to the current active nodal number, the second index points to its backward nodal number and the third index points to its forward nodal number. At the beginning of meshing process of this region, the total active nodes are the total nodes on the boundary contours of this region.

The forward and backward node are defined in pre-selected direction, e.g. count-clock wise, therefore, the unmeshed region always lie on the left side of line segments. For each active node, depending on the angle between the forward and backward node, three operations were performed:

1. Closing, if the angle is smaller than a pre-selected value. One triangle is formed, this is purely topological operation no new node is inserted. (see Fig. 2)

2. Bisecting, if the angle is smaller than 180 degrees but larger than that of closing operation, then two equal sides triangles were generated, then use the middle point of the vertices of two triangles as a new nodal point candidate. (see Fig. 3)

3. Forming, if the angle is greater than 180 degrees the node is called reflex node. One triangle is generated based on one of smaller line segments (see Figure 2: Topologically closing one angle, node i become inactive
Figure 3: Bisect the angle, node i become inactive

Figure 4: Form a new triangle on the edge of reflex node

Fig.4). Since the one triangle can be generated in the concave corner, there is no need to demand that the region is convex and no need to divide the region during the meshing process, which needs recursive programing possibility as in the paper [8].

In all operations, the mesh data is built concurrently as:

1). The total element number and total nodal number is updated.
2). The current element’s three nodal number are stored in the element node list.
3). The node’s element list, which stores all element number around the current node, is also updated. To avoid bad element formed, the maximum
elements connected to one node is prescribed, and if the number exceeded the maximum, the program stopped and prints the reason in output file.

4). The temporary array is updated, which made some nodes inactive and new active nodes added in the place of inactive nodes or in the bottom of the temporary array.

In both bisecting and forming operations, the new point, nodal point candidate was inserted. We introduce the concept of local node i’s distance as:

\[ d_i = \frac{d(i - 1, i) + d(i, i + 1)}{2} \]  \hspace{1cm} (3)

\( d(i, j) \) is defined as the distance between the point \( i \) and point \( j \). \( i - 1 \) and \( i + 1 \) are the backward and forward nodes of node \( i \). In order the candidate point become one of nodal points, three conditions must be met by the candidate:

1). The candidate point has to be inside unmeshed region.
2). The candidate point has to be far away from other nodal points.
3). If the candidate point is close to any known active node in prescribed range, the candidate point is replaced by the known active node.

The first condition ensures that the inserted point is inside the unmeshed region, which is composed of current active nodes, therefore, the edges connected to the new point do not cross with any other existing edges as explained in Fig.5.

The second condition ensures smooth transition of element size. In case of that the new point is far away from any other active nodal point, but close to known existing edge in the order of nodal spacing, the reduced size element is formed e.g. 80% of normal size as shown in Fig.5.

The third condition will prevent the generation of distort elements. It also implicitly offers the possibility to divide the region into sub-regions. Generally, the variation of line elements across the boundary lines of the region were restricted within a certain range. To achieve large variation without generating distort elements, region sub-division is necessary. The detailed information on region division can be found in reference [2] [8], here no ex-
j is far away from other points

j is replaced by n

form reduced size elements

Figure 5: 3 situations of the candidate point j

Explicit region sub-division is performed during the mesh generation, instead it is performed by replacing the candidate node by a known active node which is close to candidate node.

The process is repeated to all active nodes until total active nodes is less than two, then the whole process is shifted to other sub-regions until all sub-regions were covered. Since the mesh data were built up during the meshing process, they are explicitly available in the post-processing swapping and smoothing and will speed these processes.

2.3 Internal point control spacing

With above mentioned method, the nodal spacing of internal node can only be achieved by boundary element size which may give bad gradation inside internal area. Realizing that a small element size has to be used at a region close to boundary segments of short length, the internal control points are used to obtain a mesh with smooth transition in element size from one point to other point. The node spacing \( h \) at any point \( P(x, y) \) is to be computed by the following formula:

\[
h(x, y) = \frac{\sum_{i=1}^{N} \frac{h_i}{d_i}}{\sum_{i=1}^{N} \frac{1}{d_i}}
\]  

(4)
is the node spacing at point $P(x, y)$, $d_i$ is the distance of $P$ to user points or internal control points with nodal spacing $h_i$. Since the method to form new triangles is based on existing edges, therefore, the local node length $d_i$ is served as a local nodal spacing. When the calculated node spacing is bigger than existing edge, then a bigger (20%) element is formed else a smaller (20%) element is formed. From our test examples, the effects of internal point spacing control is not significant (see Fig.13 and Fig.14), therefore, we implement local refinements by splitting the longest edge of triangle into two around the refinement area. Local refinement is done in post-processing stage, and followed by smoothing and swapping because splitting will introduce locally bad elements in previously good shaped mesh.

Compare to conventional front advancing method, the current method do not need any searching and sorting during the meshing process. The time consuming part of the current method is the calculation of internal point nodal spacing, which is linear time algorithm. However, the program is designed in such a way that the user can suppress this possibility by change input parameter. In the current method, closing, bisecting and forming are all local operations aimed to generate equilateral triangle, and internal nodal point spacing is calculated globally, therefore, the current method try to generate the elements locally good shape, globally good gradation.

Since the mesh was generated locally good shape and globally good distribution, the resulting mesh does not need many swapping and smoothing operation, therefore, the post-processing of both swapping and smoothing is less time consuming than generating triangles. The difference between the mesh without any swapping and smoothing and the mesh after swapping and smoothing is shown in Fig.12 and Fig.13. The time consuming part of transforming the triangular mesh into quadrilateral mesh is due to the fact that the each node's element adjacency list is rebuilt after obtaining the quadrilateral mesh, which involves repeating and searching over all nodal points and all elements. The another reason is due to memory swapping, because the program is designed to use same array during the rebuilt process.
2.4 post-processing

After complete triangle generation, the mesh is smoothed by repositioning the internal nodes of existing mesh in such a way that the new position of internal nodes (including nodes on the openings or connectors) lie in the center of its polygon formed by all its neighbouring nodes (including the boundary nodes). Mesh smoothing is a repetitive process, however, 2 or 3 times smoothing is enough for practical use.

Direct quadrilateral mesh generation is generally difficult than triangular one, the algorithms involved are complex as described by Blacker [4]. In this paper, base triangular mesh is transformed into the quadrilateral mesh by dividing one triangle into three quadrilateral elements, which guarantees the generation of quadrilaterals. The method is showed in Fig. 6.

2.5 implementation

The implementation of mesh generation described here can be explained in following flow chart, see Fig.7:

Within the preprocessing stage, the user inputs were checked and were trans-
lated into boundary elements for each boundary.

In the generation stage, for each surface, the active nodes on loop contour enclosed the surface was built. For each active node, the angle (sin and cos functions) between the forward and backward nodes is calculated, depending on the value of angle, the closing, bisecting and forming operation is performed, meanwhile the data structure (node adjacency list) of the current surface and global data are also updated. The process is repeated until all active nodes become inactive, then the process is repeated to other surfaces.

In post-processing, swapping is applied to triangular mesh, then followed by central smoothing. If local refinement was chosen, the local refinement is applied to internal control points area, followed by swapping and smoothing. If the output mesh is quadrilateral mesh, the triangular mesh is transformed into quadrialteral mesh followed by smoothing.

3 program and user guide

The complete package of mesh generator is composed of 48 subroutines written in standard FORTRAN 77 computer language. The package itself will perform all input data checking and report to the output file on errors, warnings and advises to the solution. The complete source code is available via the contact to the author.

To generate mesh, the user need prepare an input file named meshin, the results without graphic possibilities are written into the file meshout which include all input and output information in text form. The input file is organized in records. To implement the inputs, the following definitions refered in the package are neccessary.

1. User point and internal control point: The user points are the points used by user to represent the curves, the user point is numbered starting from 1 to the maximum 200 (see Fig.8). The nodal point number of user points are unchanged after mesh generation. The internal control points are the internal points (not boundary points) where the mesh need to be refined. Local
Figure 7: Flow chart of mesh generation
mesh refinement will be applied to these areas. The internal control points are not included in the final mesh.

2. Curve: A curve is a straight line segment with direction from starting point to ending point, the curve is numbered from 1 to the maximum 200 (see Fig.8).

3. Surface: A surface is a simply connected polygon (convex or concave). The surface is composed of specialy ordered curves. (see Fig. 9)
If the user's problem concerned with multi-connected regions, the user has to resolve them into simply connected region by openings or connectors. Refer to Fig.9 case (a), if the curves are defined in following way:
c1 = 1 to 2
c2 = 2 to 3
c3 = 3 to 4
c4 = 4 to 1
then the surface s1 would be (c1,c2,c3,c4) enclosure
If the curves are defined in following way:
c1 = 2 to 1
c2 = 2 to 3
c3 = 3 to 4
c4 = 4 to 1
then the surface s1 would be (-c1,c2,c3,c4) enclosure.
Refer to Fig.9 case (b), if the region contains a hole (1,3,4,6), then two connectors (5,11) and (2,8) are used to divide region into two surfaces s1 and s2, if the curves are defined in following way:
c1 = 1 to 2
c2 = 2 to 3
c3 = 3 to 4

\[ i1 \rightarrow i2 \quad \text{ci=(i1,i2)} \]

\[ i1 \rightarrow -i2 \quad \text{ci= (i2,i1)} \]

i1 and i2 are user points to represent curve ci

Figure 8: User points to represent curve
Figure 9: Decompose a multi-connected region into simply connected ones

c4 = 4 to 5

c5 = 5 to 6

c6 = 6 to 1

c7 = 7 to 8

c8 = 8 to 9

c9 = 9 to 10

c10 = 10 to 11

c11 = 11 to 12

c12 = 12 to 7

c13 = 11 to 5

c14 = 2 to 8

then the surface s1 would be \((c7, c14, -c1, -c6, -c5, -c13, c11, c12)\) enclosure, the
surface s2 would be \((c8, c9, c10, c13, -c4, -c3, -c2, c14)\) enclosure.

### 3.1 input records

With the above defintions, we describe the input records in the following
steps and variables in each record is seperated by a comma ',':

The first record consists of 4 variables:
ishape, nsurf, ncurve, npoint

ishape: type of element generated. 3 for 3-node triangular element, 4 for 4-node quadrilateral element
nsurf: total number of surfaces, maximum 15
ncurve: total number of curves, maximum 100
npoint: total number of user points, maximum 200

The second record consists of 2*ncurve variables, starting from the first curve, the first node number and the second node number sequentially.

The next several records consists of curve number which enclosed the surfaces, starting from the first surface and for each surface terminated with integer 0, therefore, the program can calculate the number of curves enclose the current surface

The following record consists of npoint+1 variables

The first variable is cunit, which is used for nodal spacing calculation. If the nodal spacing is $h$ given in input file, then the real nodal spacing is approximately $cunit \times h$. The other npoint positions are the nodal spacing value of each user point

The following record consists of 2*npoint variables
Starting from the first user point sequentially, the x and y coordinates of each user point

The following record is nspint, number of internal control points
If the number of control points are not zero, then local refinement factor followed by the control points' nodal spacing have to be given in the next record with nspint values, followed by the record of the coordinates of control points in x and y sequentially. The factor determine the area of local refinement. If the distance between the internal control point to the center of the current element is $d$, and the average side length of the current element is $d_s$, then all elements whose distance to internal control points satisfy $d \leq fact \times d_s$ will be refined once.
As an example, if we want to generate a triangular mesh in the region represented in Fig.9 case (a), and demand around point 1 and point 3 dense mesh without using internal control points, the input file would be:

```
3,1,4,4
1,2,2,3,3,4,4,1
1
2
3
4
0
1.0,0.1,1.2,0.1,1.2
0,0,2,0,2,2,0,2
0
```

4 results

The resulting mesh of above inputs is shown in the Fig.16 and Fig.17, together with other tested meshes (triangular and quadrilateral meshes). Also, we compare with two existing examples from the reference [9] in Fig.10 and Fig.11 to demonstrate the efficiency of our method. The dependence of CPU time on the number of nodes is not a linear relationship in our method as shown in the table 1, rather it is influenced by the number of sub-division of the region.

<table>
<thead>
<tr>
<th>Name</th>
<th>Regions</th>
<th>Nodes</th>
<th>Elements</th>
<th>Generate</th>
<th>Swap</th>
<th>Smooth</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>34</td>
<td>123</td>
<td>192</td>
<td>9s</td>
<td></td>
<td></td>
<td>Dec Station GPX</td>
</tr>
<tr>
<td>Ours</td>
<td>1</td>
<td>193</td>
<td>323</td>
<td>0.93s</td>
<td>0.38s</td>
<td>0.38s</td>
<td>PC-80386SX-25</td>
</tr>
<tr>
<td>Right</td>
<td>36</td>
<td>136</td>
<td>208</td>
<td>11s</td>
<td></td>
<td></td>
<td>Dec Station GPX</td>
</tr>
<tr>
<td>Ours</td>
<td>3</td>
<td>217</td>
<td>369</td>
<td>0.66s</td>
<td>0.27s</td>
<td>0.5s</td>
<td>PC-80386SX-25</td>
</tr>
</tbody>
</table>

Table 1: Compare with existing meshes from the reference [9]
The other test results were shown below, the number of nodes and elements of these test examples, as well as the CPU time on a PC-80386SX-25 machine are listed in the table 2. Here nodes and elements are referred to the quadrilateral mesh.

**Table 2: Nodes and elements, CPU time listing**

<table>
<thead>
<tr>
<th>Name</th>
<th>Regions</th>
<th>Nodes</th>
<th>Elements</th>
<th>Generate</th>
<th>Swap</th>
<th>Smooth</th>
<th>Transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>1</td>
<td>2081</td>
<td>2004</td>
<td>2.57s</td>
<td>0.828s</td>
<td>0.875s</td>
<td>39.6s</td>
</tr>
<tr>
<td>Triangle</td>
<td>1</td>
<td>419</td>
<td>381</td>
<td>0.277s</td>
<td>0.164s</td>
<td>0.109s</td>
<td>1.976s</td>
</tr>
<tr>
<td>V-shape</td>
<td>1</td>
<td>1995</td>
<td>1932</td>
<td>1.37s</td>
<td>0.769s</td>
<td>0.878s</td>
<td>36.7s</td>
</tr>
<tr>
<td>Hole</td>
<td>2</td>
<td>596</td>
<td>546</td>
<td>0.49s</td>
<td>0.218s</td>
<td>0.218s</td>
<td>4.06s</td>
</tr>
<tr>
<td>Blank</td>
<td>3</td>
<td>2141</td>
<td>2067</td>
<td>1.43s</td>
<td>0.824s</td>
<td>0.933s</td>
<td>48.5s</td>
</tr>
</tbody>
</table>

**5 conclusions**

After reviewing the current existing mesh generation methods, a simple and robust triangular and quadrilateral mesh generation method has been developed in this work. The method guarantees the generation of quadrilateral mesh as long as the triangular base mesh has been generated. The gradation of mesh can be controlled by either boundary or internal control nodes or the combination of both. The local refinements is offered as an optional process combined with internal control option. Local refinement is implemented as a post-processing by splitting the longest element edge into two. The post-processing time in swapping and smoothing is less than that of generation of triangles in the current method. The processing time in generating the triangles is not linearly depending on the nodes generated, instead it is influenced by the number of divisions of the region. The efficiency of our method is demonstrated by comparing with existing results from the reference. The method is also simple in design and implementation, and it only took two month for a full time staff.
ACKNOWLEDGEMENTS

I am grateful to dr. ir. J. A. H. Ramackers for his arranging the financial support of this work.

References


Figure 10: left mesh and right mesh from the reference

Figure 11: Meshes from current method
Figure 12: Mesh of the square before swapping and smoothing

Figure 13: Mesh of the square after swapping and smoothing
Figure 14: Mesh of the square after local refinements

Figure 15: Mesh of a triangle
Figure 16: Base mesh of the square

Figure 17: Quadrilateral mesh of the square
Figure 18: Triangular mesh of a v-shape geometry

Figure 19: Quadrilateral mesh of a v-shape geometry
Figure 20: Triangular mesh of square-hole

Figure 21: Quadrilateral mesh of square-hole
Figure 22: Triangular mesh of a deformed blank

Figure 23: Quadrilateral mesh of a deformed blank