Regression Model based Automatic Finite Element Mesh Generation

Jie Jin

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

February 2008

Your file Votre référence
ISBN: 978-0-494-40884-1
Our file Notre référence
ISBN: 978-0-494-40884-1

ABSTRACT

Regression Model based Automatic Finite Element Mesh Generation

Jie Jin

A typical mesh generation problem consists of generating triangles, quadrilaterals, tetrahedron or hexahedron elements based on the predefined piece-wised boundary of a domain. A finite element mesh is a discrete representation of a geometric domain, resulting from the subdivision of the domain into patches referred to as elements.

In recent years, a variety of methods have been introduced to generate 2D quadrilaterals for the boundary by using some predefined 'if-then' rules until the whole domain is filled with required elements. However, it is difficult to define the rules to generate a good-quality element based on all boundary information. This thesis proposes a regression model-based element extraction method for automatic finite element mesh generation that needs information from the boundary as few as possible. The method represents the 'if-then' element extraction rules and trains the relationship behind these rules. The input for the regression model includes the coordinates of some boundary points while the output defines the parameters for extracting an element. To generate good-quality elements while keeping the updated problem still solvable, the design and definition of the regression model is more complex than those in the traditional classification methods. Numerical experiments on quadrilateral mesh generation based on design of experiments demonstrate the effectiveness of the proposed method in comparison with the results obtained from existing algorithms.

Acknowledgements

I would like to express my appreciation to my supervisors Dr. Yong Zeng and Dr. A. Ben Hamza. Without their support and guidance, I could not have finished this research. Their knowledgeable and helpful suggestions and their constant encouragement have helped me overcome numerous difficulties. They have offered me precise guidance and creative comments throughout my Master's study. I have benefited enormously from the discussions with them, especially in research methodology, modes of thinking, and analytical strategies.

My gratitude also goes to Guangqing He and Shengji Yao, who have helped me so much during my research.

Table of Contents

List of Figures

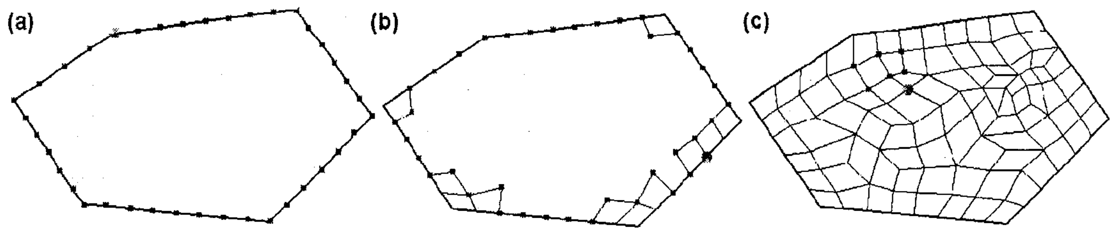
List of Tables

Chapter 1
Introduction

1.1 Background

In recent years, the finite element method (FEM) has become a prevalent technique for analyzing physical phenomena in the field of structural, solid and fluid mechanics, as well as for the solution of many field problems. Various plane, surface and volume meshing algorithms have been developed. Usually they are variants of the following techniques: mapping transformations [8] [27]; explicit solution of partial derivative equations [22]; node insertion and connection [5] [14] [30]; regular grid overlay approaches [7] [9]; paving and progressive front techniques [3] [4]; and geometric decomposition [16] [17] [21].

Each finite element computation needs the input of large amounts of data. Manual preparation of these data, as done in the first part of the 1970s, is a tedious and error-prone task. To improve the accuracy and efficiency of structural analysis we need sophisticated and flexible preprocessor programs to make the process of modelling and input data preparation quicker and therefore more profitable. The benefit of the system for many areas including structural analysis and optimization, computational fluid dynamics, and geometric modeling has kept it an active research. The problem is illustrated in Figure 1.



(a) A predefined piece-wised boundary. (b) Generating element. (c) Finish generating.

Figure 1. Mesh generation problem.

Manual mesh generation is suitable for domains with a simple geometry where the user defines the elements by their vertices. The resulting mesh will often be transformed to obtain some more accurate meshes. A mesh with previously known connectivity can also be used for the semi-automatic generation. This approach is suitable for domains with hexahedrons or for cylindrical geometries where, from a 2D mesh of a section, the corresponding volumetric elements are defined layer by layer with the help of a given transformation.

Because of the importance and underlying challenges, many automatic mesh generation methods and algorithms have been developed for 3D *triangular* [6] [19] [37] [39] [40] and 2D *quadrilateral* [5] [14] [18] [24] [30] . There are also some methods based on recursive [1] [15], intelligent [35] and arbitrary topology [20] [29] in both 2D and 3D area which are interesting and helpful. At present, commercial software kits in automatic mesh generation field are widely used as a result in industry projects [8] [10] [25]. However, most of the currently available mesh generation methods are heuristic. The methods can be applied only in solving some specific problems. The mesh generation of geometric shapes with complex boundary is still an opening question.

Element extraction recognized as a flexible method to process geometric shapes with complex boundaries [2] [12] [13] [34]. These methods base on the feature of the boundary and the knowledge of the researcher. The meshes generated by these methods usually have high quality elements in the area close to the geometric boundary. This feature is critical for complex engineering problems. The high quality elements on the boundary usually followed by poor elements in the middle of the geometric domain which mainly because the robust rules for element extraction are extremely difficult to acquire. This drawback made it difficult to be widely accepted as a robust mesh generation method.

Based on a mathematical analysis of the problem underlying the element extraction method, many methods based on neural network have been published [41] [42] [43]. In that kind of papers, the artificial neural network has been used to extend rules for extracting elements. Those rules generate elements of good-enough quality around both the boundary and the middle of the domain. However, poor elements will be generated if the situations are not covered in the training samples. Therefore, the critical factors and the regression model should be developed to fix those kinds of problems.

1.2 Objective

A finite element mesh is structured if it has a regular pattern of connections between elements. The boundary of a domain can be regular or irregular. The complex irregular domains have unstructured meshes.

For the sake of simplicity, we will use two-dimensional quadrilateral mesh generation as an example. At best, the generation process should be fully automatic, not requiring complex interaction from the user. The problem under consideration is how to create a quadrilateral mesh if the boundary of the region is a closed sequence of straight segments. We only use the simple (without 'holes', 'arcs' etc.) regions since a multiple connected can be decomposed to simple regions.

The goal of this thesis is to generate a two-dimensional quadrilateral mesh from a predefined piece-wised boundary, satisfying the following requirements:

- Inner corners of each quadrilateral element should be between 45° and 135°;
- The aspect ratio (the ratio between opposite edges) and taper ratio (the ratio between adjacent edges) of each quadrilateral element should be between 0.1 and 10;
- The mesh around the short boundary segments should be dense and vice versa;
- The transformation from dense mesh to coarse mesh should be smooth.

The generated quadrilateral elements calculated from the points on the predefined piece-wised boundary. To find the optimum number of points to be picked up from the boundary as critical factors and find the regression model to generate the mesh automatically are the two main goals of the thesis. The algorithm must be able to cope with the generation of large meshes within reasonable compile time. The algorithm also should be able to cope with difficult situations (large variations in the step size, sharp corners, etc.)

1.3 Literature review and challenges

Manual mesh generation is suitable for domains with a simple geometry where the user defines the elements by their vertices. The resulting mesh will often be transformed to obtain more accurate meshes. A mesh with previously known connectivity can also be used for the semi-automatic generation. This approach is suitable for domains with hexahedrons or for cylindrical geometries where, from a 2D mesh of a section, the corresponding volumetric elements are defined layer by layer with the help of a given transformation. Three methods will be introduced in this section.

1.3.1 RESM [14]

RESM is based on a novel application of the Delaunay triangulation algorithm. Then the quadrilateral mesh can be generated as easily as triangular meshes for complex domains.

Basic steps: for a 2D quadrilateral mesh, it can be shown that

$$NB = 2(ND + NO - NE - 1)$$

Where NB is the number of nodes on the geometry boundary, ND is the total number of nodes, NE is the number of elements and NO is the number of openings within the domain.

Step 1: Fully automatic quadrilateral mesh generation is transforming curved boundaries into straight segments and making sure that the number of nodes on the boundaries is even.

Step 2: The automatic generation of a geometry specifying initial quadrilateral mesh covering the domain.

Step 3: To locate and insert new nodes into the initial mesh by using the RESM for mesh control. See Figure 2.

Step 4: The generated mesh will be optimized through nodal smoothing and diagonal swapping techniques.

Step 5: Mesh Optimization.

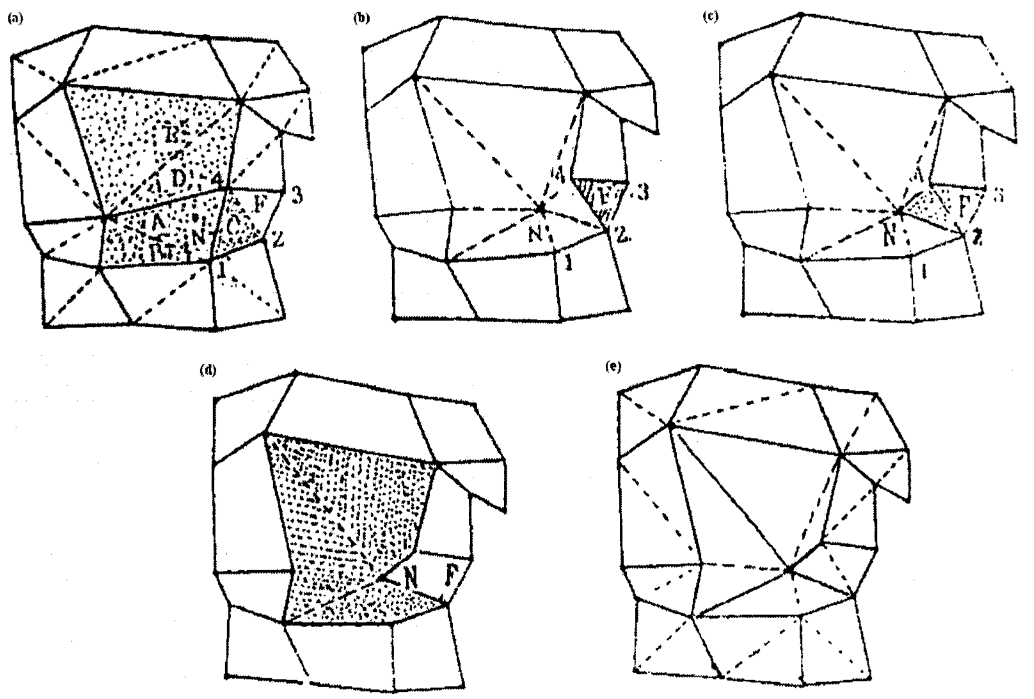


Figure 2. Insert the new node *N* to an old quadrilateral mesh.

When insert a new node into an old quadrilateral mesh $NS = NT + 2$ where NT is the number of triangles inside the CIP, NP is the number of pending triangles and NB is the number of sides of the CIP.

$$NS' = NS + NP = (NS + NP) + 2 = even$$

Regarding the quadrilateral mesh as a triangular mesh, it can be shown that

$$H_a(X) = \frac{2}{\sqrt{3}} \sqrt{\int_\Omega \frac{1.0}{H_r^2(X)} \Big/ NET} \times H_r(X)$$

where *Ha(X)* is the absolute element size function. *Ha(P)* stands for the actual element size around point *P*.

At first, $C_0 = \int_{\Omega} \frac{dx}{H_x^2(X)}$ during mesh generation, for each element *i*, $C_i = \int_{\Delta_i} \frac{dx}{H_r^2(X)}$ where Δ_i stands for triangle *i*. If $C_i \geq K_0 \geq C_0$, then a node will be added into this element. This process continues until all elements pass this check. The final mesh will have approximately *NET* element.

For mesh smoothing, the well known Laplacian smoothing technique has been used, which repositions the internal node at the centriod of the polygon formed by its neighboring nodes. An example of generated mesh is shown in Figure 3.

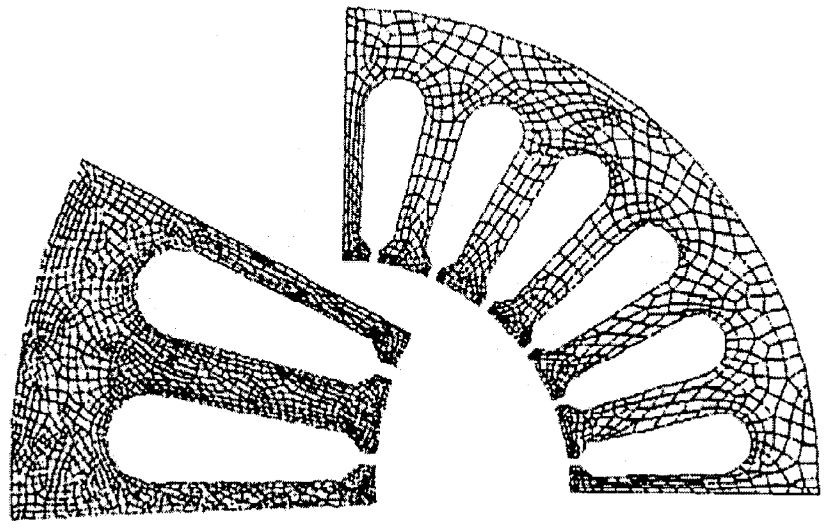


Figure 3. Quadrilateral mesh generated by RESM algorithm.

1.3.2 Extension of TP algorithm [30]

Figure 4 is the flow chart of this introduced algorithm. Suppose a set of nodes numbered from *1* to n_t in order (and in the anticlockwise direction) is placed on the boundary of the region. Steps between nodes may vary from one pair of nodes to another. We must introduce criteria to determine how many elements should adjoin angle α with node i as a vertex, and that depends on the size of the angle.

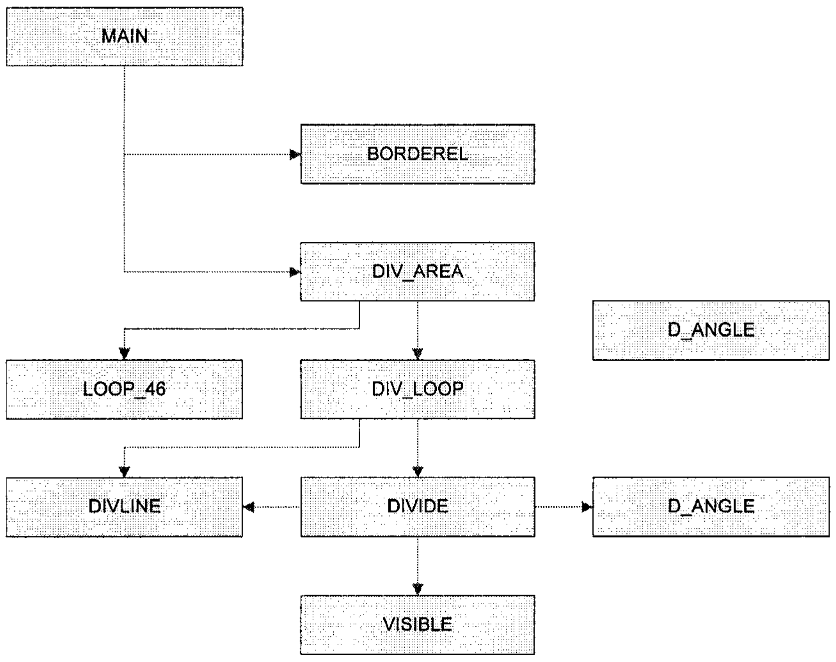


Figure 4. Flow chart of extension of TP algorithm.

The general idea of the algorithm is quite straight-forward. An initial loop should be split into two loops, then each of them should be split again and so on, until all generated loops must be divided into several (2, 3 or 4) quadrilateral elements. The most complicated problem is how to select the best splitting line as Figure 5. Several requirements have to be satisfied simultaneously.

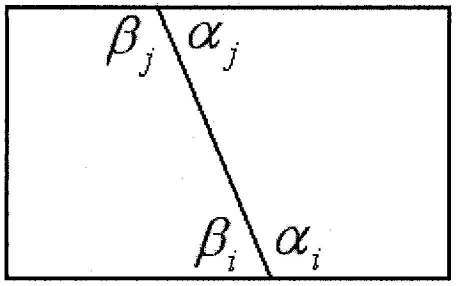


Figure 5. The best selected splitting line.

The splitting line should be selected in a way that the new angles obtained, α_i, β_i, α_j, β_j are as close to the right angle as possible.

Coordinate of new nodes should be generated such that any four elements adjoining those angles should be as close to rhombs as possible.

The number of nodes on each of the new loops must be even (otherwise, a division into a set of quadrilateral elements may not exist.)

The splitting line must not cross the boundary of the loop; i.e., if a splitting line connects node i and j, the node j must be 'visible' from i.

Suppose a set of nodes numbered from 1 to n_r in order (and in the anticlockwise direction) is placed on the boundary of the region. Steps between nodes may vary from one pair of nodes to another. It must be introduced criteria to determine how many elements should adjoin angle α with node i as a vertex and that depends on the size of the angle. Three possible cases are shown in Figure 6.

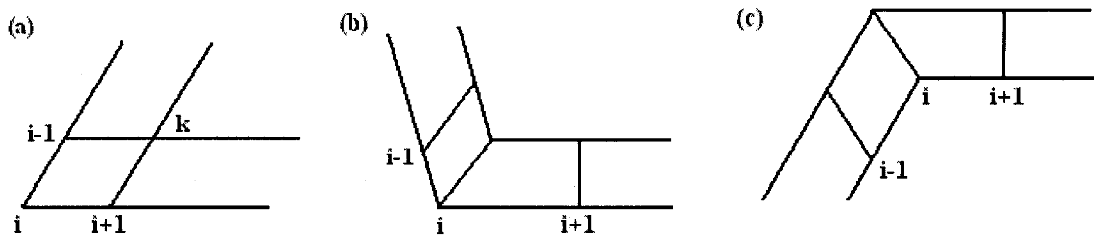


Figure 6. Three possible cases for elements adjoined as a vertex.

The result generated by the algorithm can be shown below:

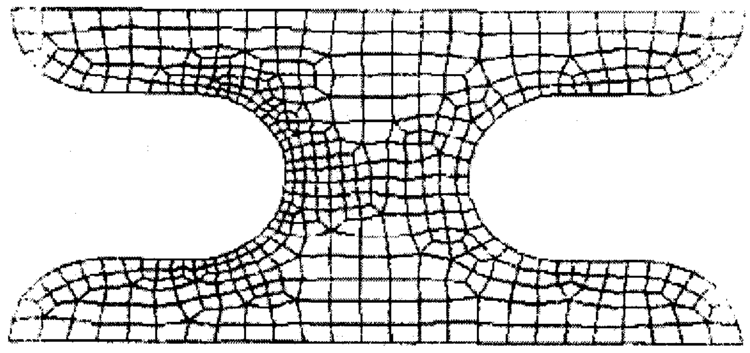


Figure 7. Quadrilateral mesh of the algorithm.

1.3.3 ILA algorithm [35]

ILA algorithm can well control both size and aspect ratio of quadrilateral element in a two-dimensional plane. The algorithm can be extended straightforwardly to hexahedral elements in a three-dimensional solid. The elements are created sequentially, considering local information on geometrical constraints and user's demand on quality of elements. Figure 8 shows the flow chart of ILA implemented based on the hypothesis.

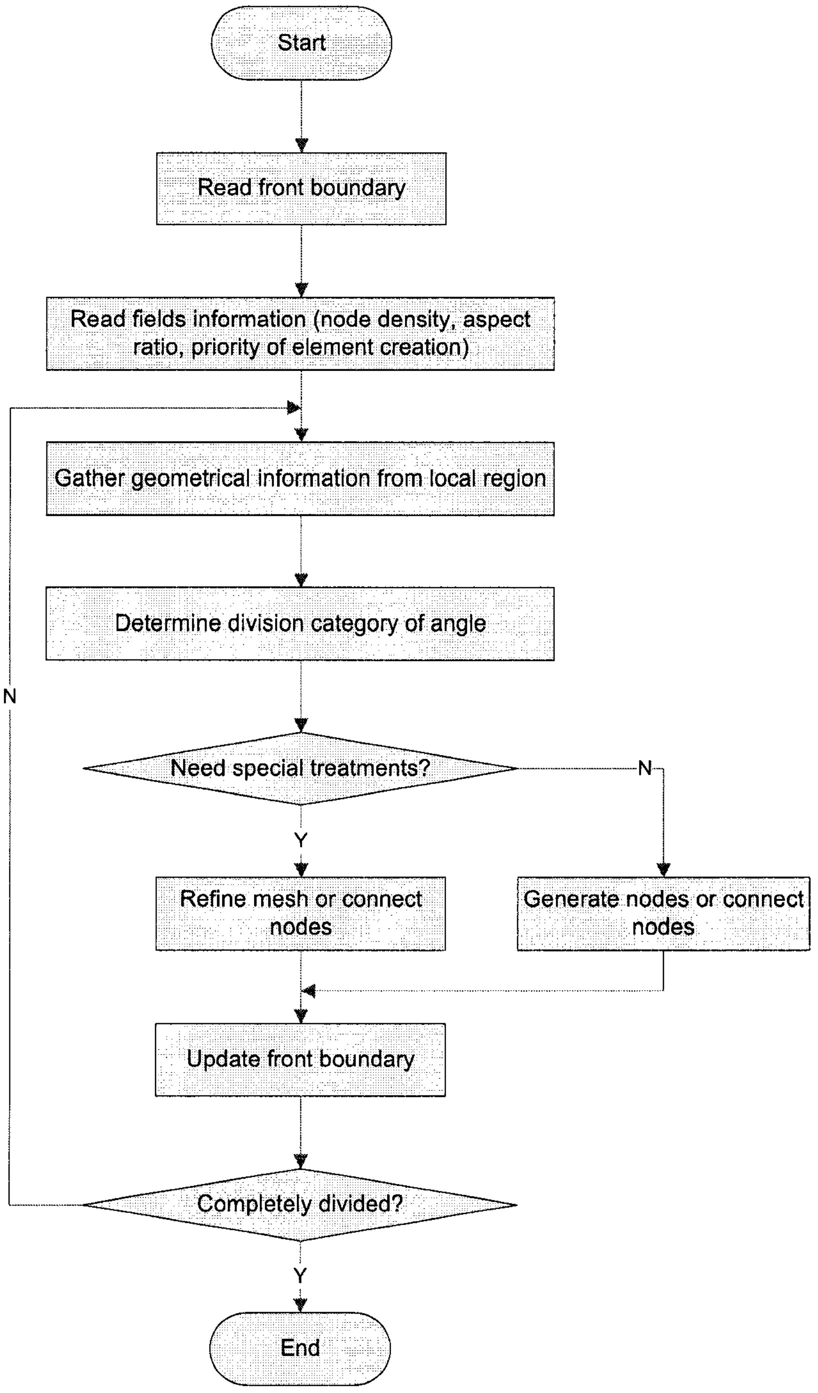
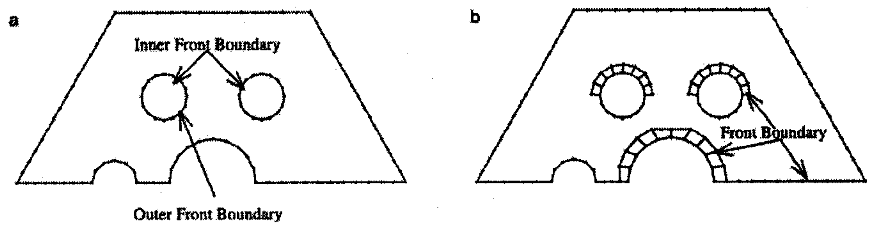

Start
Read front boundary
Read fields information (node density, aspect ratio, priority of element creation)
Gather geometrical information from local region
Determine division category of angle
Need special treatments?
Y
N
Refine mesh or connect nodes
Generate nodes or connect nodes
Update front boundary
Completely divided?
N
Y
End

Figure 8. Flow chart of ILA algorithm.

Step 1: Front boundary. An initial front boundary should be defined as a set of segments that represent a geometry model shown below:



(a) Initial front boundary. (b) Front boundary during mesh generation.

Figure 9. Front boundary.

Step 2: Collection of local geometrical information. In the case of quadrilateral mesh generating problems, the lengths of segments and angles of two adjacent segments should be collected. In the case of hexahedral mesh generation problem, the areas of faces and angles of two adjacent faces should be collected.

Step 3: Categorization of angle division. Considering the local geometrical information collected as described in step 2, Table 1 summarizes five kinds of division categories.

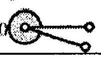
Division category	Number of divisions	An angle of two segments (degree)
C1	1	45–105
C12	1 or 2	105–155
C2	2	155–200
C3	3	200–300
C4	4	300–360 

Table 1. Definition of division categories.

Step 4: Judgement for special treatment and switch to step 5 or step 6.

Step 5: Determination of node location using fuzzy knowledge processing.

1) Geometrical parameters for goodness of element shape as shown below:

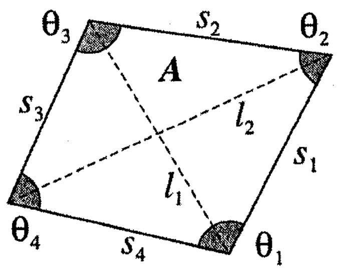
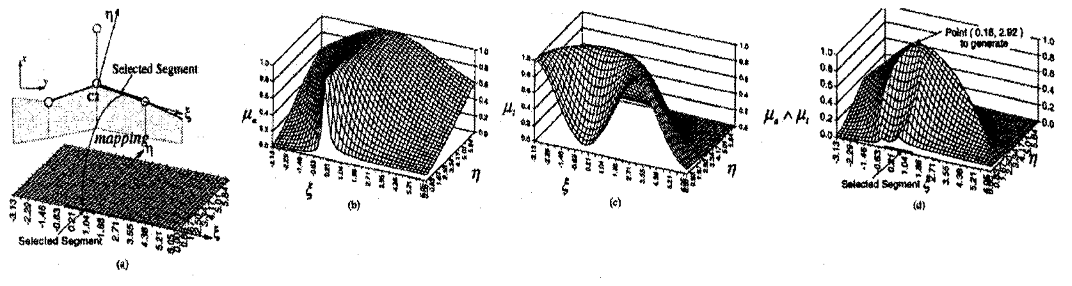


Figure 10. Geometrical parameters of quadrilateral elements.

2) Information for control of mesh subdivision.

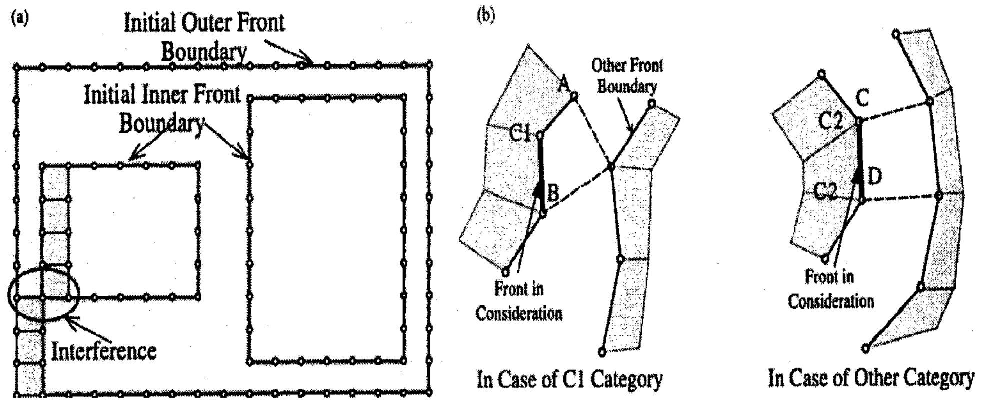
3) Fuzzy knowledge processing as shown below:



(a) Mapping from local; *x-y* to $\xi - \eta$. (b) μ_a distribution over the grid. (c) μ_1 distribution over the grid. (d) $\mu_a \wedge \mu_1$ distribution derived from fuzzy product operation.

Figure 11. Node generation based on fuzzy knowledge processing.

Step 6: Special treatments.



(a) Example of interference between different parts of front boundary.

(b) Treatment of releasing interference.

Figure 12. Interference treatment of front boundaries.

Step 7: Updating front boundary.

Step 8: Smoothing process.

Step 9: Object-oriented system design and implementation. The flow chart is shown in Figure 13. Figure 14 is an example of generation by using this method.

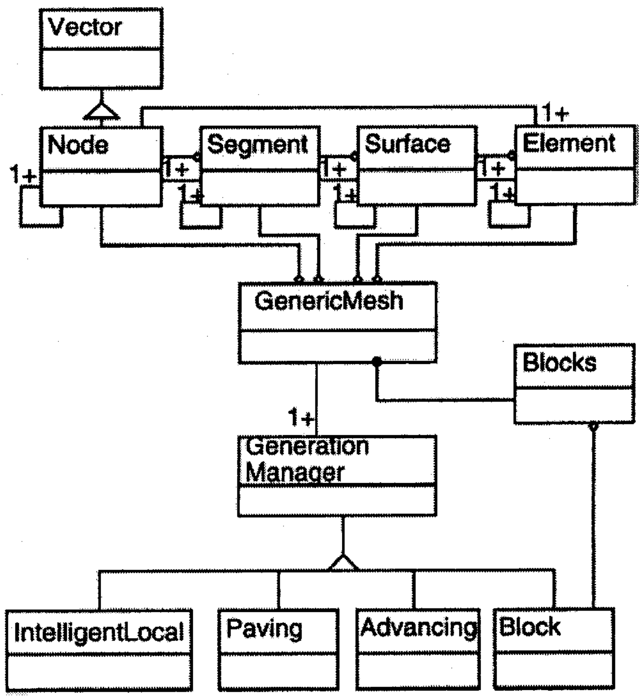
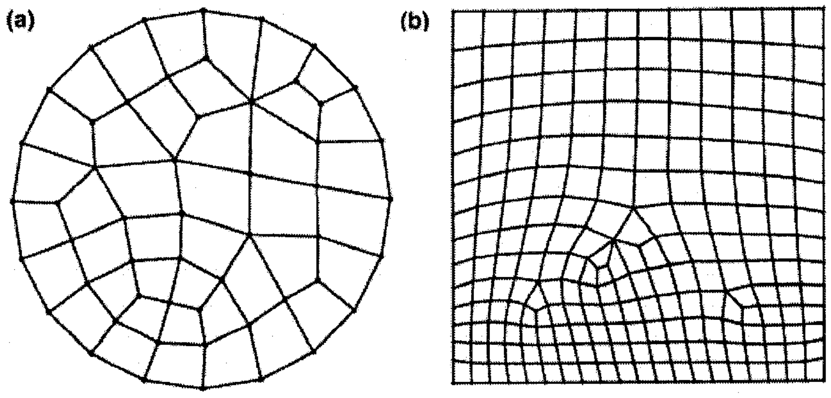


Figure 13. Object diagram of mesh generation.



(a) Mesh in circle region. (b) Mesh with size change in square region.

Figure 14. Example of quadrilateral meshes by ILA algorithm.

1.3.4 Challenges

These methods can be applied only in solving some specific problems. The optimal rules for element extraction are difficult to acquire which make the high quality elements on the boundary usually followed by poor elements in the middle of the geometric domain. This drawback made the element extraction rules difficult to be widely accepted as a robust mesh generation method.

1.4 Contributions

The major contributions in the present thesis may be summarized as follows:

- ➢ Defined the rules for collecting training sample data from two-dimensional quadrilateral meshes with different number of leading points.
- ➢ Developed a friendly user graphic program for collecting training sample data automatically.
- ➢ Found the critical factors for generating an element for a two-dimensional quadrilateral mesh.
- ➢ Proposed the optimal regression models to be used for any closed boundary to generate a two-dimensional quadrilateral mesh.

1.5 Thesis organization

The rest of the present thesis organized as follows:

- ▪ Chapter 2 reviews the *Knowledge-based* and *ANN-based* element extraction method for quadrilateral mesh generating. Some basic knowledge of mesh generation will also be introduced in this chapter.

- Chapter 3 discusses the definitions and the collection of training samples from a quadrilateral mesh.
- Chapter 4 uses *UGO* algorithm of *Auto2Fit* software to find the critical factors when generating a quadrilateral mesh. This chapter also introduces the regression models for mesh generation based on the collected training sample data.
- Chapter 5 discusses the case study and the comparison results between the mesh generated by the regression model and by other existing element extraction methods.
- Chapter 6 summarizes the main research results based on the present thesis and points out future research directions.

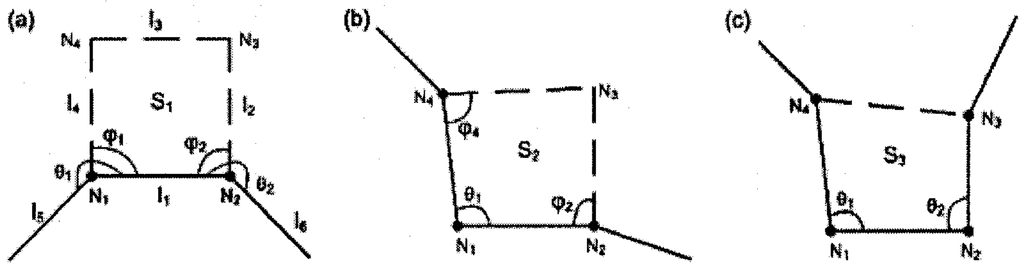
Chapter 2
Knowledge-based and ANN-based Element Extraction Method

Existing mesh generated algorithms are reviewed in this chapter, including knowledge-based by Zeng and Cheng (1993) and ANN-based by Yao and Zeng (2005) element extraction method and the concepts used in other chapters.

2.1 Preliminaries

2.1.1 Basic element extraction rules

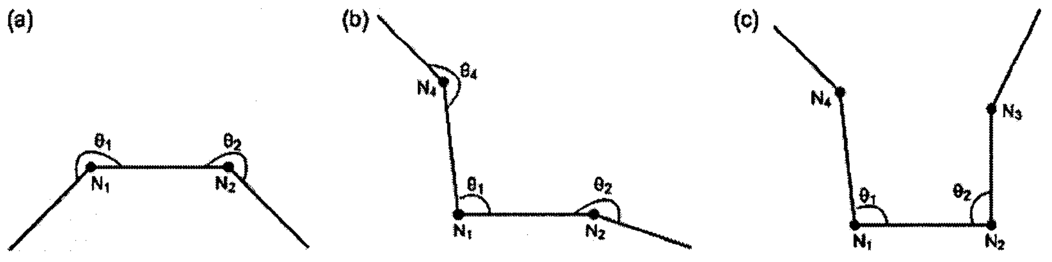
For the two-dimensional quadrilateral mesh generation problem, there are three basic design actions in R, which are shown in Figure 15. They extract an element from the domain by adding three, two, or one line(s), respectively.



(a) r1: Adding three lines. (b) r2: Adding two lines. (c) r3: Adding one line.

Figure 15. Basic design actions.

Corresponding to the three design actions in Figure 15, there are three basic types of boundary components as is shown in Figure 16. Those three types of components define basic components for representing any two-dimensional domains in terms of the actions given in Figure 15 that named primitive boundary components.



(a) #1:b_1. (b) #2:b_2. (c) #3:b_3.

Figure 16. Basic boundary components.

The precedents of these actions are the coordinates of a group of boundary points whereas the conclusions include the number of lines to be added and the parameters defining these lines. The following lists of the actions used in a system developed by Zeng and Cheng (1993).

If $\theta_1 \geq U_c, \theta_2 \geq U_c$, then $\phi_1 = \theta_1/2, \phi_2 = \theta_2/2, l_2 = (l_1 + l_6)/(2\sin\phi_2), l_4 = (l_5 + l_1)/(2\sin\phi_1)$, as shown in Figure 16.(a)

If $\theta_1 \leq U_c, \theta_2 \geq 180^\circ, \theta_4 \geq 180^\circ$ then, $\phi_2 = \pi - \theta_1, \phi_4 = \pi - \theta_1$, as shown in Figure 16.(b)

If $\theta_1 \leq U_c, \theta_2 \leq U_c$, then link N_3 and N_4, as shown in Figure 16.(c) where U_c represents the upper bounds for an element inner corner.

2.1.2 Selection of reference point

To generate a finite element mesh, a point is chosen from the existing boundary points. This point is called the reference point, around which the new element is generated. Surrounding this reference point, a number of points (called leading points) that have more influence than others on generating a new element are collected from the existing boundary.

A reference point is the point where we start to extract an element. Not every point on the existing boundary can be taken as a reference point. In our study, if the angle at a boundary point is between 45° and 135°, it should be taken as a reference point. Once a new element generated, the next reference point needs to be selected. Various criteria can be used to identify the next reference point.

In Figure 17, *2N* points are picked up around a reference point as the leading points. The number of leading points will be selected based on the complexity of the domain. Theoretically, the more complicated the domain is, the more leading points will be included for the input data. The coordinate of the reference point and the leading points of that reference point compose the input of the regression model as $[P(_{l,N}), P(_{l,N-1}), \ldots, P_{l2}, P_{l1}, P_0, P_{r1}, P_{r2}, \ldots, P(_{r,N-1}), P(_{r,N})]$.

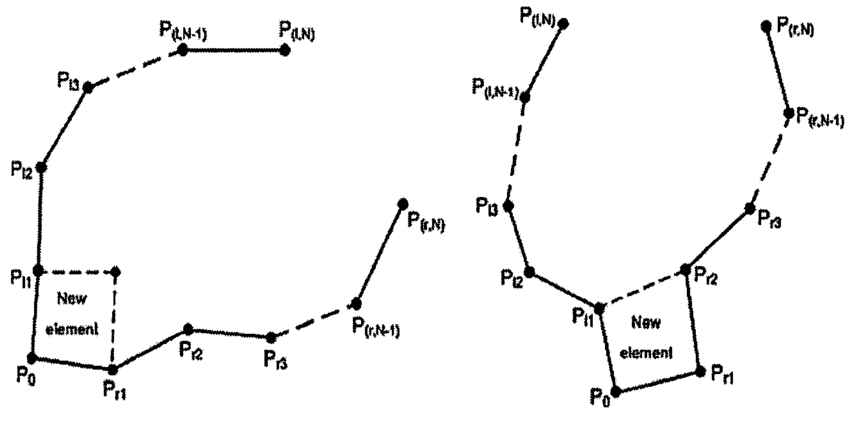


Figure 17. Leading points.

The output includes the parameters to define element. In our study, we consider three types of extractions as shown in Figure 18. *Type 0* represents the case where an element is extracted by adding a new point on the base of three boundary points. *Types 1* and *Type 2* represent the case where an element extracted without adding any new point. The only difference between *Type 1* and *Type 2* is the relative location of the reference point. We use *[type, P_n]* to represent a new element. The variable *type* should take the values of *0*, *1*, and *2*. P_n is the generated point in the new element for *Type 0*. There is no new point generated for *Type 1* and *Type 2*. P_n is called output point.

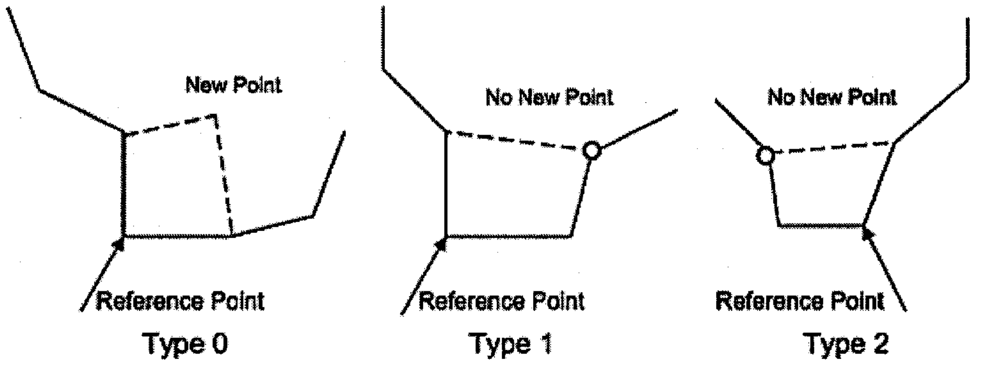


Figure 18. Three types of new elements.

The output model is shown below:

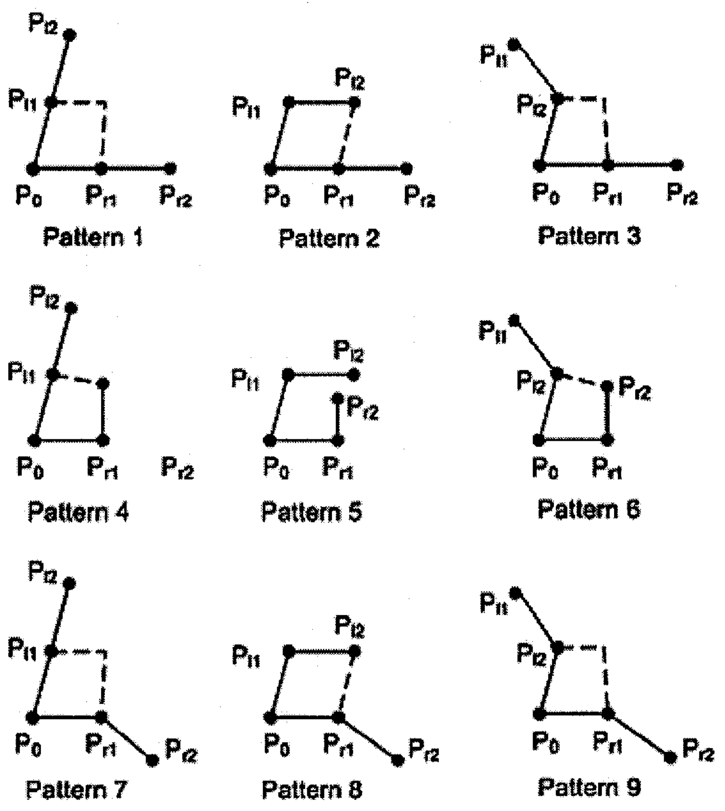
$$[\text{type}, P_n] = [P(_{l,N}), P(_{l,N-1}), \ldots, P_{l2}, P_{l1}, P_0, P_{r1}, P_{r2}, \ldots, P(_{r,N-1}), P(_{r,N})]$$

2.1.3 Initial training samples for regression model-based method and ANN based method

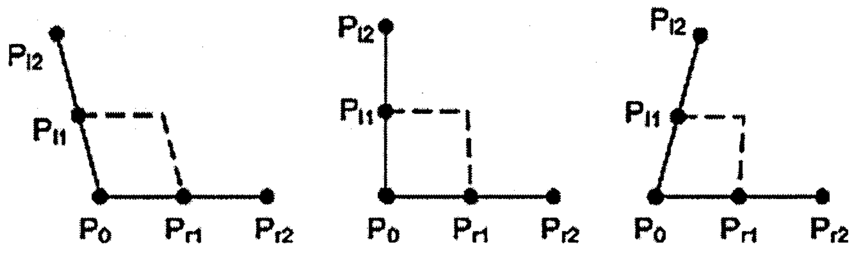
The initial training samples should combine the basic relationship of input and output. We ignore the influence of the relative length of boundary components and consider only three levels: *small (0° to 90°)*, *medium (90° and 180°)* and *large (180° and 270°)* of angles. Therefore, if *n* is the number of leading points, there are $3^{(n-2)}$ different patterns given. Four leading points will correspond to nine patterns as shown in Figure 19 (a).

Obviously, *pattern 5* is not a valid sample and no initial training samples will be generated for that pattern. For every other patterns, we use a different angle at the reference point to generate a number of training samples which shown in Figure 19 (b).

Different forms of *pattern 1* can be indicated that the number of training samples for each pattern is the same. With the equal number of samples, each pattern will play an equal role in the training process of the regression model and ANN model.



(a) Nine patterns of input–output relationship.



(b) Different forms of pattern 1.

Figure 19. Patterns define.

2.1.4 Final pattern

A final pattern is needed when the number of points left on the boundary is less than the number of input points in the mesh generation method. If five boundary points are used to extract a new element, then the last boundary will have four points left, which form a quadrilateral element. Thus no final pattern is needed. For seven points taken as the input, however, there are only six points left on the last boundary. These six points have to be dealt with separately. For this final pattern, three situations will be considered to complete the mesh generation as shown in Figure 20.

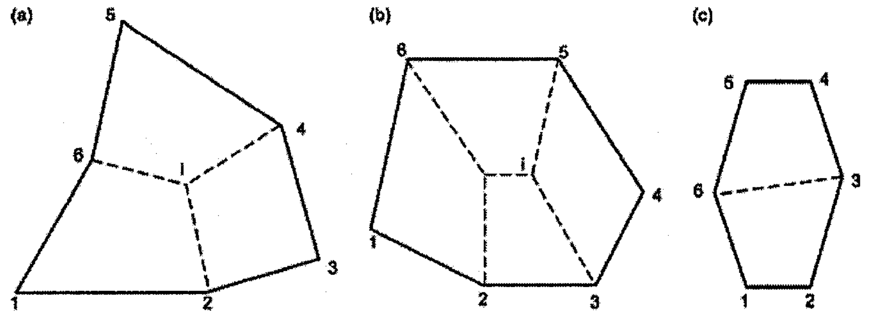


Figure 20. Three types of final pattern.

For the pattern shown in Figure 20(a), the coordinates of the i^{th} node is

$$\begin{cases} x_i = \frac{1}{3}(x_2 + x_4 + x_6) \\ y_i = \frac{1}{3}(y_2 + y_4 + y_6) \end{cases}$$

For (b), the coordinates of the i^{th} node is

$$\begin{cases} x_i = \frac{1}{8}(3x_3 + 3x_5 + x_2 + x_6) \\ y_i = \frac{1}{8}(3y_3 + 3y_5 + y_2 + y_6) \end{cases}$$

For (c), connect point 3 and point 6.

If we use nine points or more as the input to extract a new element, the final pattern becomes more complex since more different situations have to be taken into account to complete the mesh generation. For more input point case, the final pattern should be more complex.

2.2 Introduction of Knowledge-based and ANN-based extraction method

2.2.1 Knowledge-based element extraction method [13]

Knowledge-based extraction method generates a finite element mesh by extracting elements along the domain boundary. The element can be a triangle, quadrilateral, tetrahedron, or hexahedron depending on the given mesh generation problem. In this mesh generation process, one element usually cut from the domain according to a set of predefined '*if-then*' rules. Each time an element removed from the domain, the domain boundary will be updated for further element extraction. This process continues until the domain becomes a valid element.

Figure 21 is an example of element extraction applied to two-dimensional quadrilateral mesh generation. The points are the discrete points on the boundary.

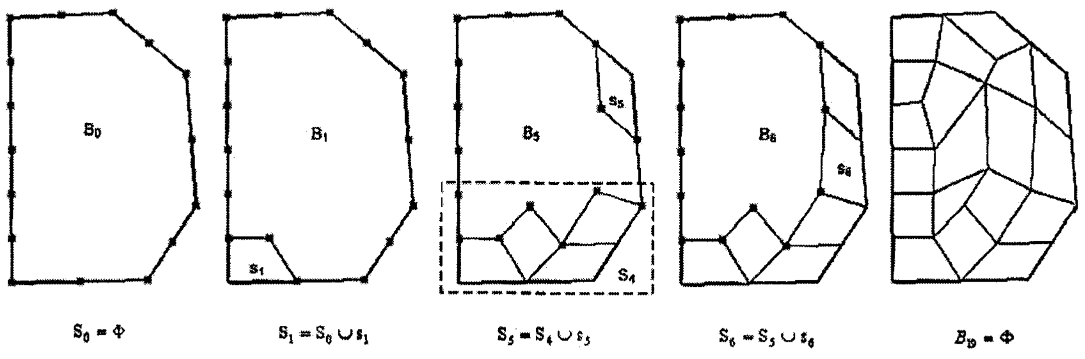


Figure 21. Quadrilateral mesh generation by knowledge-based method.

Figure 22 is the flow chart of this knowledge-based method when generating elements for the existing boundary.

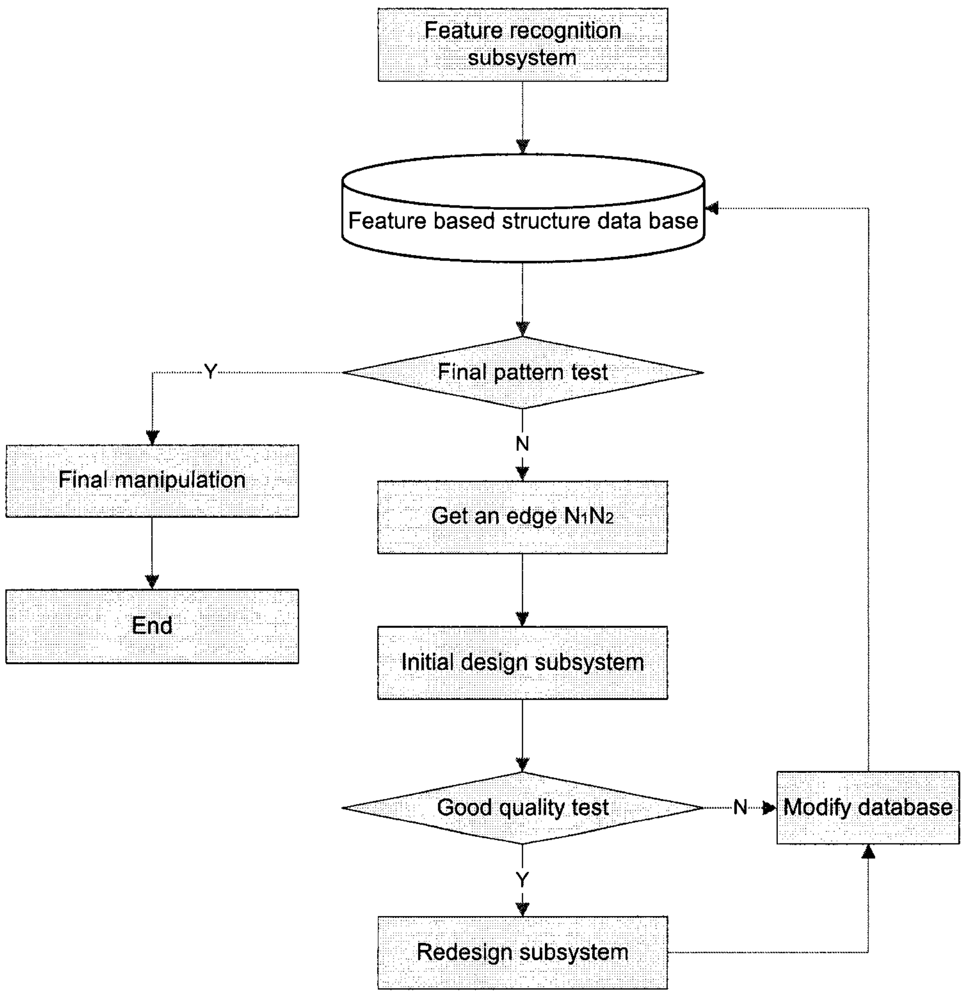


Figure 22. Mesh generation flow chart of knowledge-based method.

2.2.2 ANN-based element extraction method [43]

Figure 23 is the structure of ANN model. The input data including the coordinates of the reference point and its corresponding leading points introduced in Section 2.1.2 Figure 17 are fed into the initially trained ANN model. The output of the ANN model is the parameters that can represent the new element to be extracted. The type as shown in Figure 18 and the coordinate of the output point will be the output of ANN model.

According to the output parameters, a new element is extracted around the reference point and leading points. If the new element is not good enough, the training samples will be adjusted and the ANN model will be retrained until a good-quality element is extracted. Then the boundary will be updated for generating the next element. This procedure is repeated until the final pattern is satisfied on the existing boundary.

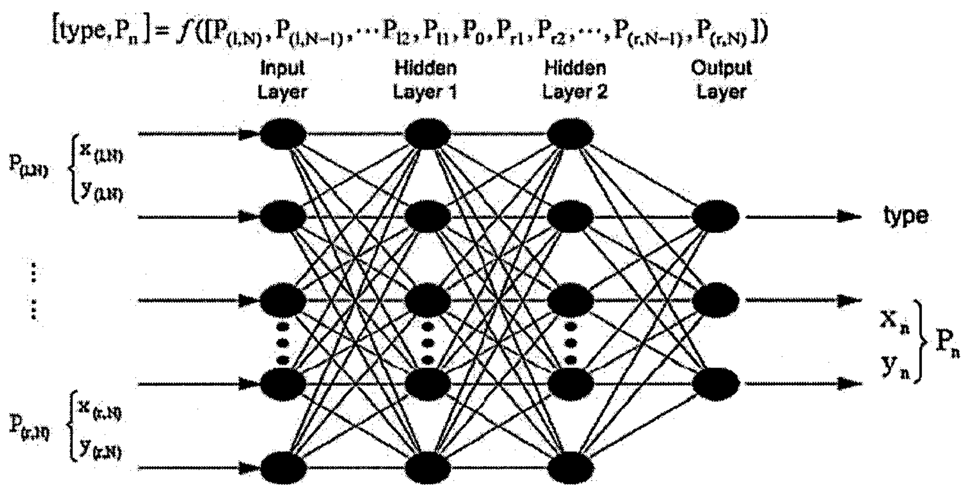

$[\text{type}, P_n] = f([P_{(l,N)}, P_{(l,N-1)}, \cdots P_{l2}, P_{l1}, P_0, P_{r1}, P_{r2}, \cdots, P_{(r,N-1)}, P_{(r,N)}])$
Input Layer
Hidden Layer 1
Hidden Layer 2
Output Layer
$P_{(l,N)}$ $x_{(l,N)}$ $y_{(l,N)}$
$P_{(r,N)}$ $x_{(r,N)}$ $y_{(r,N)}$
type
x_n y_n P_n

Figure 23. ANN model structure.

The procedures of the ANN-based mesh generation are outlined in Figure 24. After getting enough training sample data (more than 8000), an ANN model is trained to identify the relationship between input and output using initial training samples. Training of the ANN model is referred as the calculation of the weight matrix and the bias terms between neurons in different layers. In the study, Back-Propagation (BP) learning algorithm is adopted for the training.

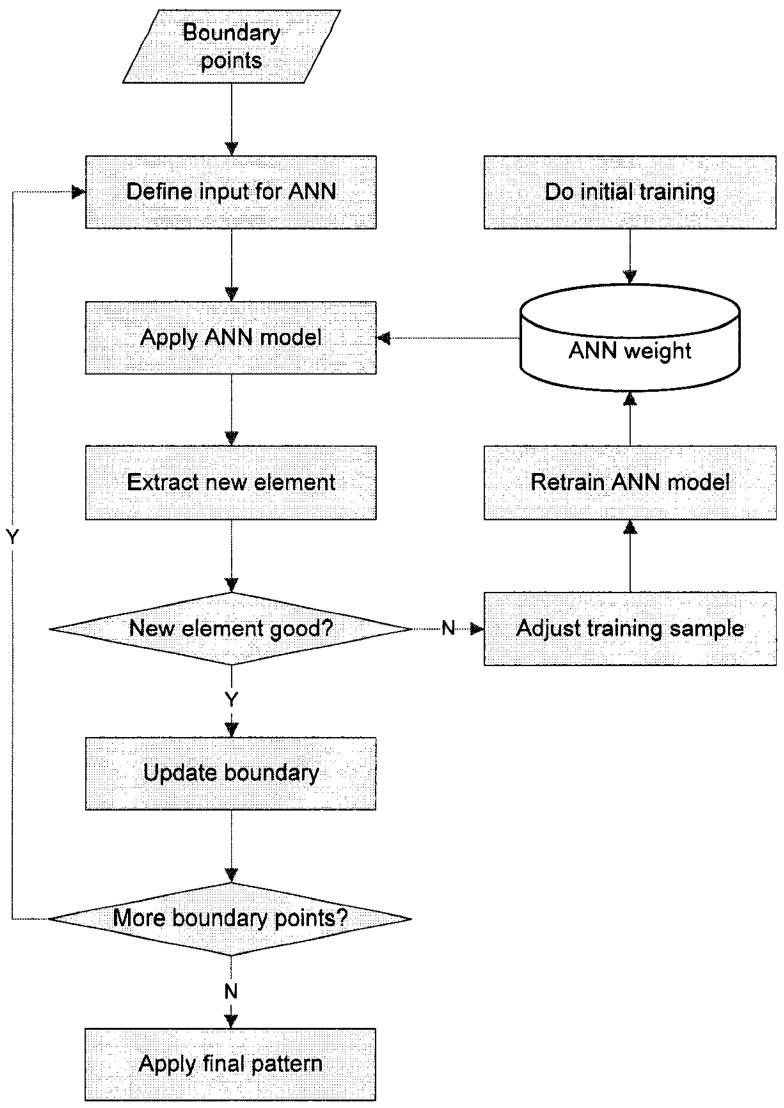


Figure 24. Mesh generation flow chart of ANN-based method.

The case studies collect the training samples based on the patterns shown in Figure 19 which means each sample will have 5 input points (1 reference point and 4 leading points).

When processing a mesh generation request using the trained result, it is easy to get a bad element because of the interaction of the initial patterns. To correct those bad elements, we need to correct the bad element manually and add the pattern of the corrected element into the training samples. Meanwhile, more samples will be generated and added to the existing sample set to improve the impact of this pattern.

2.3 Restriction of Knowledge-based and ANN-based extraction method

2.3.1 Knowledge-based element extraction method [13]

The necessary and sufficient condition for the rules in $\boldsymbol{R}$ to make a mesh generation algorithm is that they are close for the state U_i of mesh generation. In Figure 25 (a), for example, $P_1P_2P_3P_4$ is a good element, but the angle $P_5P_1P_4$ is much smaller than the lower bound for an inner corner of quadrilateral element. As a result, the boundary component $P_5P_1P_4$ does not belong to the primitive boundary components. In contrast, Figure 25 (b) is a result of the closed operation.

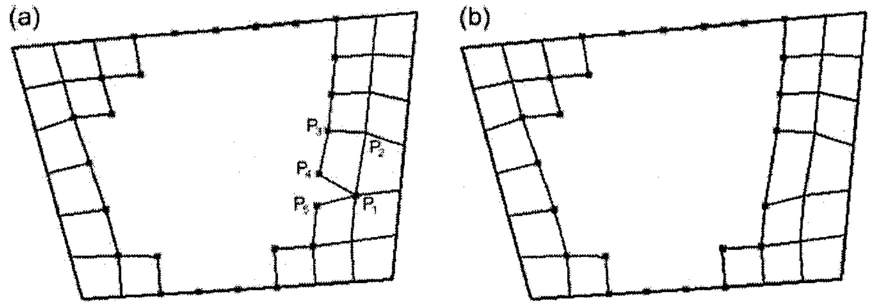


Figure 25. Boundary updating problem of knowledge-based method.

By comparing Figure 25 (a) and (b), it can be found that the reason of making an operation not closed is that the quality of the element created by this method is too good to keep the remaining boundary within the range defined by the primitive boundary components. To solve this problem, a set of balanced element extraction rules must be acquired that can generate both a good-quality element and remaining boundary components. This is a challenging task due to the complex combination of various boundary components. Figure 26 lists such an example.

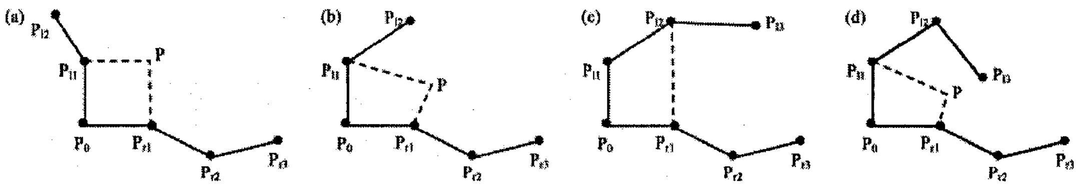


Figure 26. Challenges in defining element extraction rules.

The acquisition of these element extraction rules is usually done through an interactive trial-and-error process. The success of this interactive process largely depends on the insight and luck of the algorithm developer. The difficulty of this approach is reinforced as the complexity of the problem rises. An automatic rule acquisition method should be essential and even indispensable for the element extraction method to be applied to a wider range of problems.

2.3.2 ANN-based element extraction method [43]

As the eight patterns are only a small portion of all possibility of patterns, the training samples do not include enough information for training the ANN model for sure. Poor elements will be generated in the situations are not covered in the training samples as shown in Figure 27.

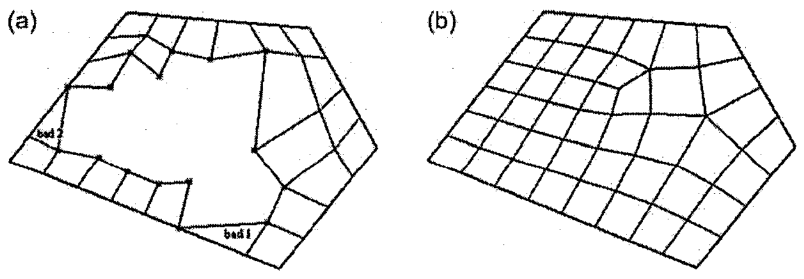


Figure 27. Boundary updating problem of ANN-based method.

In the case of 7 input points (1 reference point and 6 leading points), there are $3^4 = 81$ patterns. In the case of 9 input points (1 reference point and 8 leading points), the number of patterns is $3^5 = 243$. Each pattern should have enough samples for training. The more patterns we choose, the more training sample we should take for each pattern and the more retrain work we may need to do. Although 6 or 8 leading points can give more information on the boundary to generate better meshes, the mutual interactions between those patterns would be greatly conflicting. Retrain the ANN model is obviously. Add samples manually and retrain the ANN model will give researchers too much work additionally.

Therefore, the critical factors should be found and the optimal regression model should be developed to fix those kinds of problems.

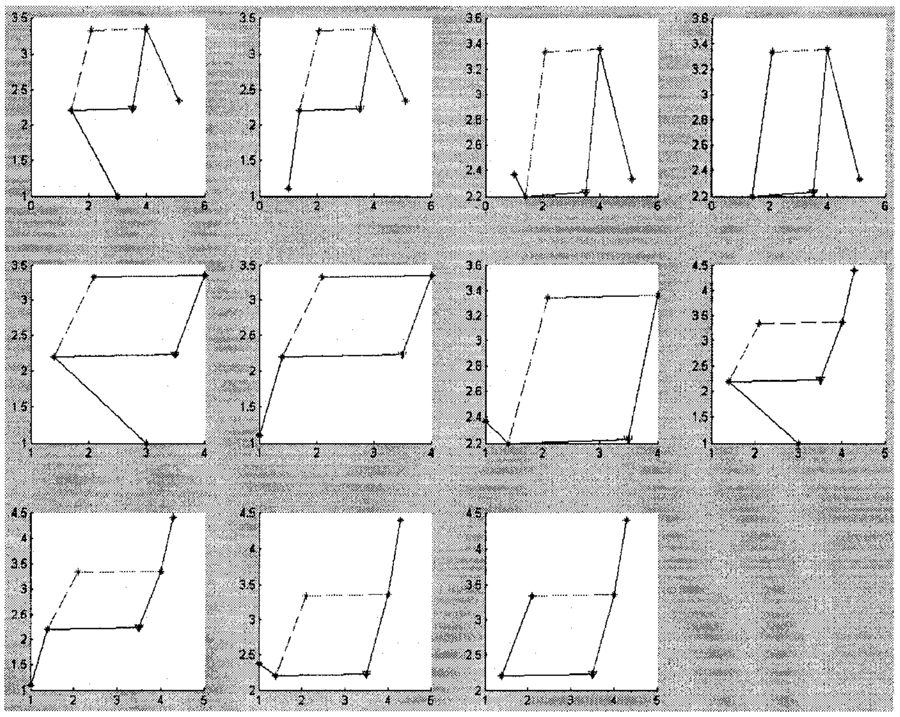
Chapter 3
Definition and Collection of Samples

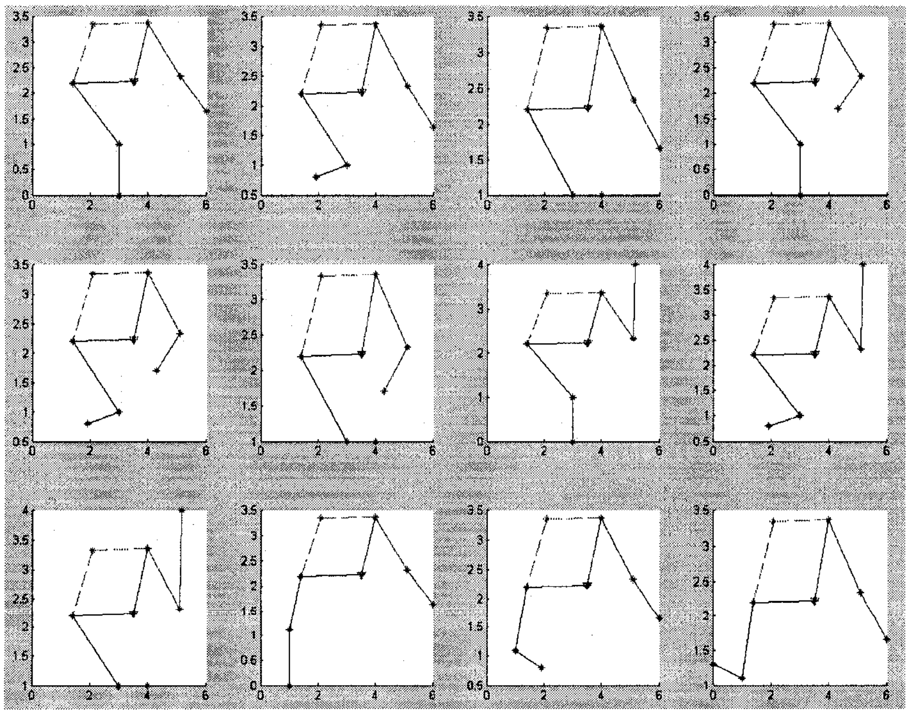
The training samples have to be representative and inclusive so that the training results can be extrapolated to other problems. Therefore, the quality of the elements is especially critical for mesh generation. The amount of the training samples is also very important because the more training samples generated the less regress error we will get.

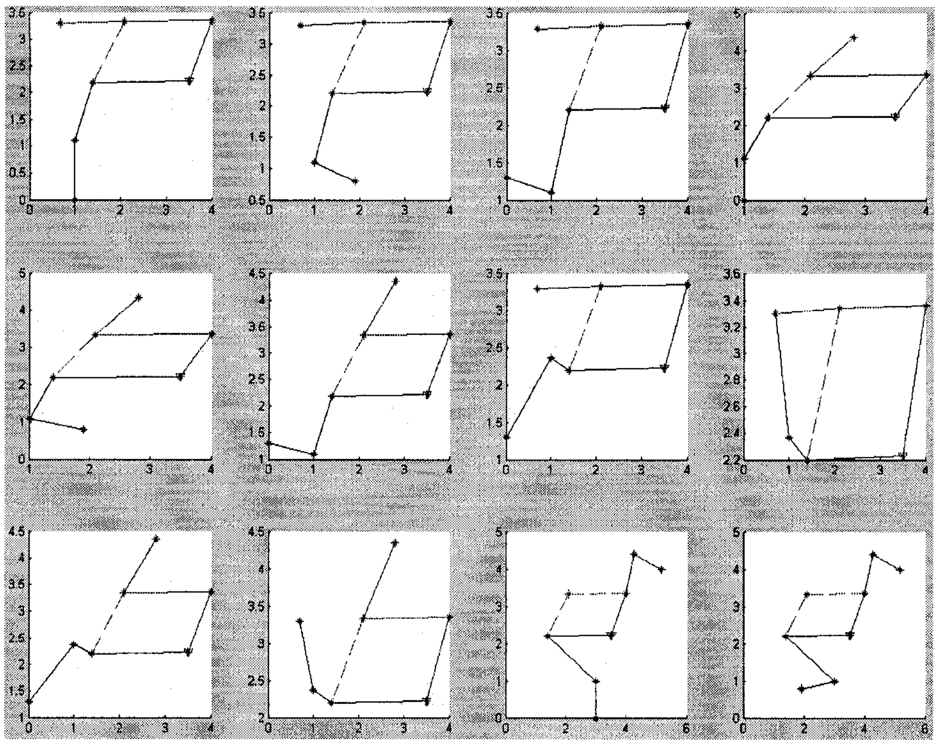
3.1 Generation of samples

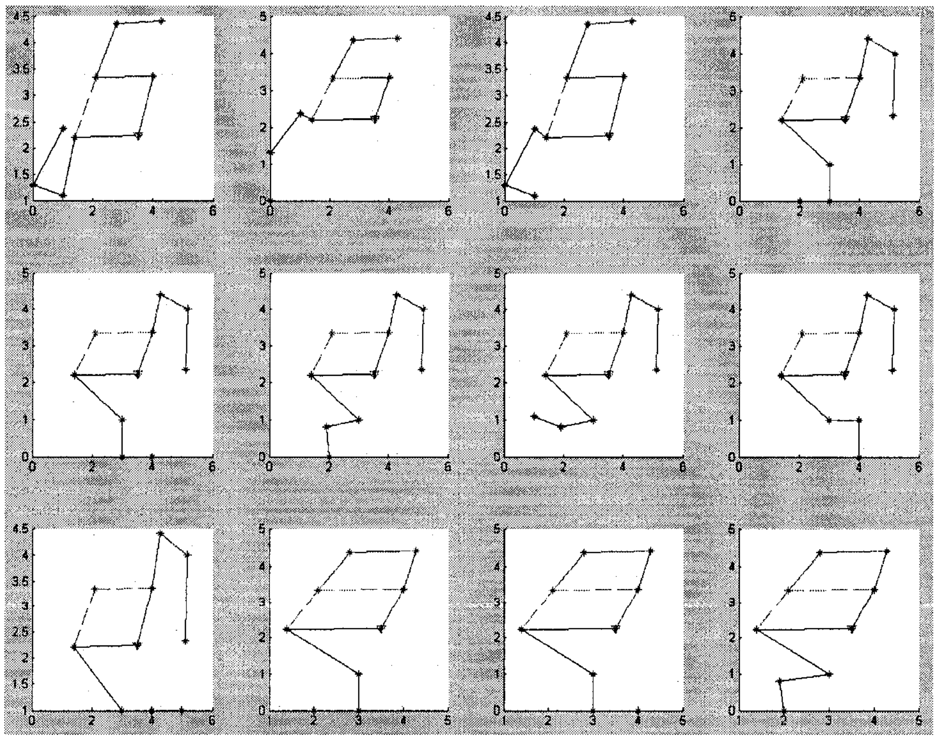
3.1.1 Initial samples

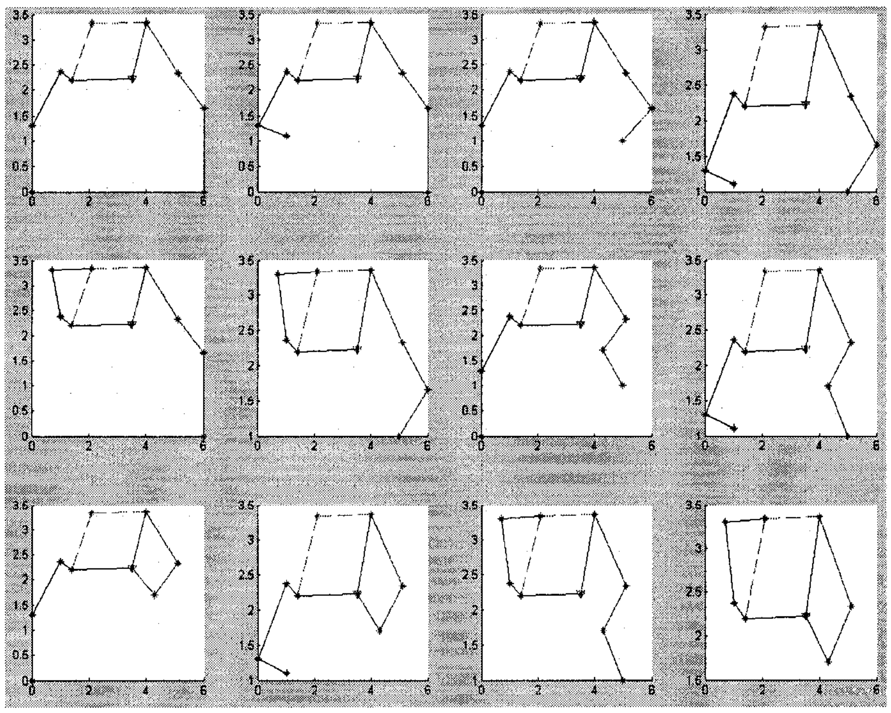
If we use 7 input points (1 reference point and 6 leading points) or more as the input to extract a new element, the pattern becomes more complex since many different situations have to be taken into account. Following is an example for the same reference point and output point with different input points. 4, 6 and 8 leading point cases are shown in Figure 28. The dark lines connect the left side leading points while the undertone lines connect the right side leading points. The downward-pointing triangle is the reference point. The connected dashed replace the old boundary and compose a new boundary with other predefined piece-wised boundary.


(a) 4 leading points #1.

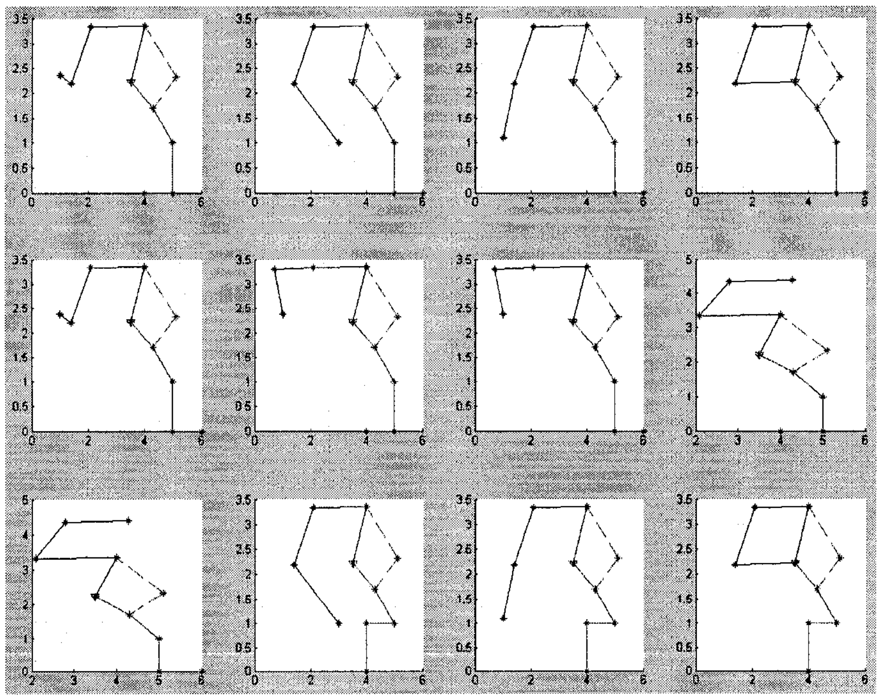

(b) 6 leading points #1.


(c) 6 leading points #2.


(d) 8 leading points #1.



(e) 8 leading points #2.

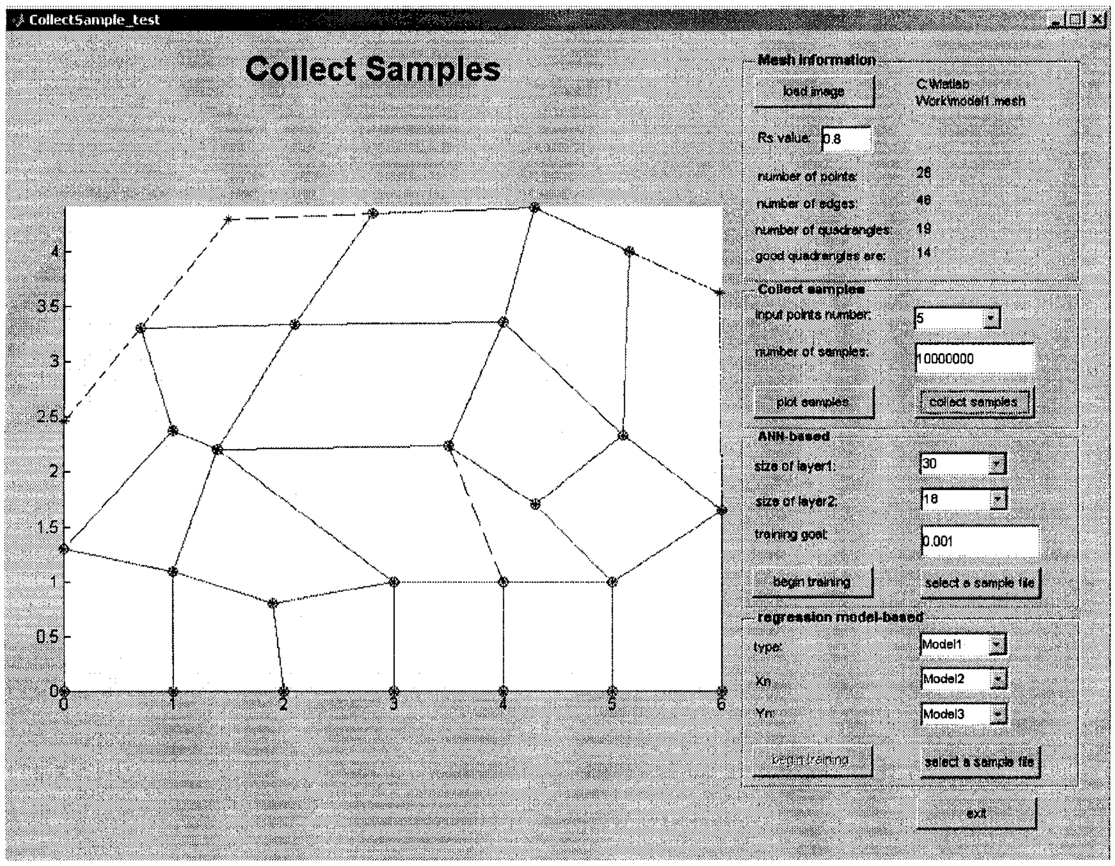


(f) 8 leading points #3.

Figure 28. Samples with different number of input leading points.

3.1.2 Collection of training samples

To generate a training sample from an existing mesh, a set of points including leading points, reference point and the output point can be picked up automatically. A user interface had been designed to pick up training samples as Figure 29. The following flow chart shown in Figure 30 is the flow chart of sample data collection. The coordinates of the input and output points and the type of the new element will be taken into a training sample.



Mesh information
C:\Matlab
Work\model1.mesh
Rs value: 0.8
number of points: 28
number of edges: 46
number of quadrangles: 19
good quadrangles are: 14

Collect samples
input points number: 5
number of samples: 10000000

ANN-based
size of layer1: 30
size of layer2: 18
training goal: 0.001

regression model-based
type: Model1
Xn: Model2
Yn: Model3

Figure 29. Interface of sample collection program.

The quadrilateral elements with real line edges are the good-quality elements that can be used to find the critical factors and the optimal regression models. The dashed edged elements are the quadrilateral with bad quality that will not be used in our training.

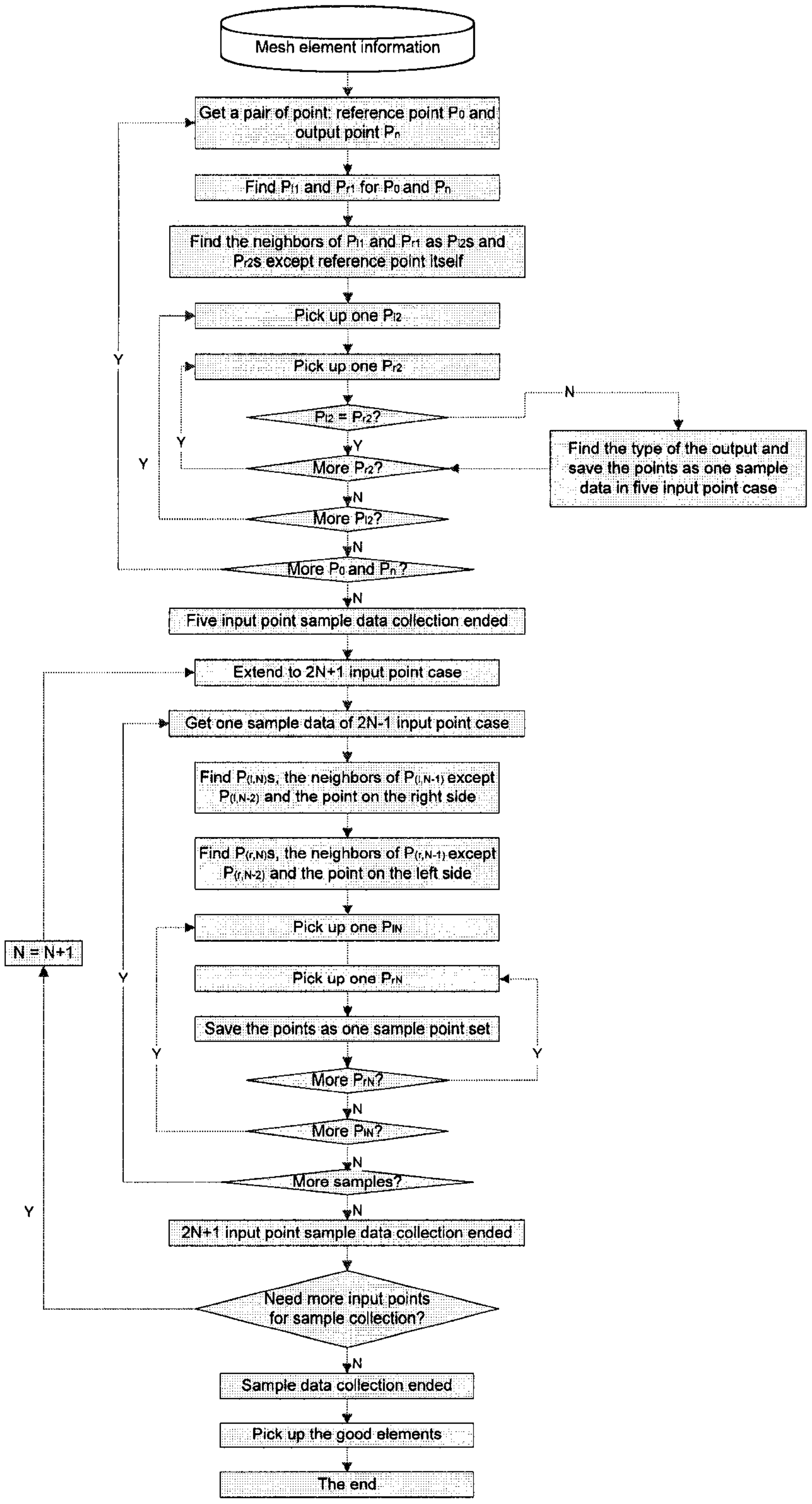

Mesh element information
Get a pair of point: reference point P0 and output point Pn
Find Pl1 and Pr1 for P0 and Pn
Find the neighbors of Pl1 and Pr1 as Pl2s and Pr2s except reference point itself
Pick up one Pl2
Pick up one Pr2
Pl2 = Pr2?
N
Y
Find the type of the output and save the points as one sample data in five input point case
More Pr2?
Y
N
More Pl2?
Y
N
More P0 and Pn ?
Y
N
Five input point sample data collection ended
Extend to 2N+1 input point case
Get one sample data of 2N-1 input point case
Find P(l,N)s, the neighbors of P(l,N-1) except P(l,N-2) and the point on the right side
Find P(r,N)s, the neighbors of P(r,N-1) except P(r,N-2) and the point on the left side
Pick up one PlN
Pick up one PrN
Save the points as one sample point set
More PrN?
Y
N
More PlN?
Y
N
More samples?
Y
N
2N+1 input point sample data collection ended
Need more input points for sample collection?
Y
N = N+1
N
Sample data collection ended
Pick up the good elements
The end

Figure 30. Flow chart of training sample collection.

3.1.3 Data collection rules

To collect sample data from a mesh should follow the rules listed below:

- In order to collect more data, each of point in good-quality elements should be taken as reference point, leading point and new generated point.
- Sample data collection procedure will find all the ways for generating one element. It means every point in the mesh could be a reference point P_0 and its neighbors should be P_{l1} and P_{r1}. The neighbors' neighbors will be P_{l2} and P_{r2} and so on. Based on different generated element for P_0, P_{new} is different and the leading points will be taken in different way.
- When collect sample data of 7 input point (1 reference point and 6 leading points) or more, the left side leading points cannot overlap with the right side leading point and vice versa.
- The angle between the adjacent boundary components should be in the range [0 ° 270°]. In our data collection procedure, we take the angle parameter as in [35° 235°].
- The outermost pair points can overlap except 4 leading point case. It means if P_{l2} overlap with P_{r2}, the new element cannot be generated anymore.

3.1.4 Define the quality of the generated element

The quality of the generated element is very important when collecting training samples. Unqualified elements will get an inaccurate result that may generate a bad-quality quadrilateral. Once this bad-quality element generated, it will influence its neighbors.

Many articles discussed mesh quality and mesh generation control [11] [28] [31] [38]. Only the mesh with good-quality elements can be taken to collect training samples. Sample data should match the following two conditions in our experiment:

First, elements must have shapes as close to the rectangular shape as possible; otherwise the elements become ill-conditioned.

We define R_s is the parameter which can indicate the quality of the quadrilateral to be generated.

$$R_s = \frac{A_s}{A_m},\ A_m = e_m{}^2$$

A_s is the area of the quadrilateral element.

e_m is the mean of the quadrilateral element's edges.

A_m is the area of a square whose edge is mean edge of the quadrilateral element.

We only take elements whose $R_s \geq 0.8$ as the good-quality samples.

The other rule is each angle of quadrilateral elements should be in the range $[45^\circ\ \ 135^\circ]$. Taking into account some complex objects which may have sharp boundary, we broaden the range to $[35^\circ\ \ 145^\circ]$ in our study. Table 2 gives an example of 30 training samples we have collected for 4 leading point case.

Input										Output		
X_{l2}	Y_{l2}	X_{l1}	Y_{l1}	X_0	Y_0	X_{r1}	Y_{r1}	X_{r2}	Y_{r2}	type	X_n	Y_n
1	1.1	0	1.3	0	0	1	0	2	0	2	1	1.1
1	2.37	0	1.3	0	0	1	0	2	0	0	1	1.1
1	2.37	0	1.3	0	0	1	0	1	1.1	1	1	1.1
0	1.3	0	0	1	0	1	1.1	1.9	0.8	2	0	1.3
0	1.3	0	0	1	0	1	1.1	1.4	2.2	2	0	1.3
1.9	0.8	1	1.1	1	0	2	0	3	0	2	1.9	0.8
0	1.3	1	1.1	1	0	2	0	3	0	0	1.9	0.8
1.4	2.2	1	1.1	1	0	2	0	3	0	0	1.9	0.8
0	1.3	1	1.1	1	0	2	0	1.9	0.8	1	1.9	0.8
1.4	2.2	1	1.1	1	0	2	0	1.9	0.8	1	1.9	0.8
0	0	1	0	2	0	1.9	0.8	3	1	0	1	1.1
1	1.1	1	0	2	0	1.9	0.8	3	1	2	1	1.1
0	0	1	0	2	0	1.9	0.8	1	1.1	1	1	1.1
3	1	1.9	0.8	2	0	3	0	4	0	2	3	1
1	1.1	1.9	0.8	2	0	3	0	4	0	0	3	1
1	1.1	1.9	0.8	2	0	3	0	3	1	1	3	1
1	0	2	0	3	0	3	1	1.9	0.8	1	1.9	0.8
1	0	2	0	3	0	3	1	4	1	0	1.9	0.8
1.9	0.8	2	0	3	0	3	1	4	1	2	1.9	0.8
1	0	2	0	3	0	3	1	1.4	2.2	0	1.9	0.8
1.9	0.8	2	0	3	0	3	1	1.4	2.2	2	1.9	0.8
1.9	0.8	3	1	3	0	4	0	5	0	0	4	1
4	1	3	1	3	0	4	0	5	0	2	4	1
1.4	2.2	3	1	3	0	4	0	5	0	0	4	1
1.9	0.8	3	1	3	0	4	0	4	1	1	4	1
1.4	2.2	3	1	3	0	4	0	4	1	1	4	1
2	0	3	0	4	0	4	1	3	1	1	3	1
2	0	3	0	4	0	4	1	5	1	0	3	1
3	1	3	0	4	0	4	1	5	1	2	3	1
3	1	4	1	4	0	5	0	6	0	0	5	1

Table 2. Original data for input 4 leading points.

3.2 Normalization of input and output

It can be found in Table 2 that the coordinate of points for input and output data could be any values. However, what matters between input and output are the relative positions of all points in each sample. Furthermore, it is the relative position that can be applied independent of the problems from which the samples were collected. To capture the generic relationship between these points, it is essential to transform all the coordinates into a uniform coordinate system without distorting the relations among the input and output. In two-dimensional quadrilateral mesh generation field, the reference point is taken as the origin of the new coordinate system. The vector from the reference point to the first leading point on the right side of the reference point is set as a unit vector along the positive direction of x axis. All the input and output points in the same sample are then transformed into a new coordinate system by translation, scaling and rotation. This transformation is shown in Figure 31.

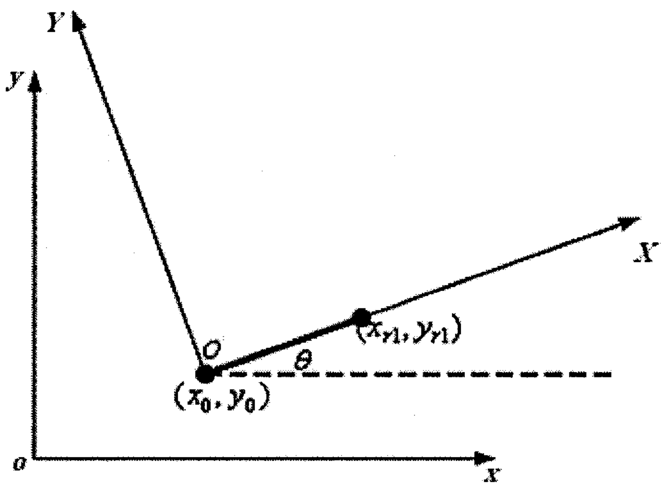


Figure 31. Coordinate transformation.

Given the reference point (x_0, y_0) and the first leading point on the right, (x_{r1}, y_{r1}), we will have the new coordinate system *XOY* as is shown in Figure 31. All the points including leading points, reference point, and the output point in the new element in each sample will be transformed into the new coordinate system *XOY*.

3.2.1 Translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -x_0 \\ -y_0 \end{pmatrix}$$

3.2.2 Scaling

$$d = \sqrt{(x_0 - x_{r1})^2 + (y_0 - y_{r1})^2}$$

$$\begin{pmatrix} x'' \\ y'' \end{pmatrix} = \begin{pmatrix} 1/d & 0 \\ 0 & 1/d \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

3.2.3 Rotation

$$\begin{pmatrix} x''' \\ y''' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x'' \\ y'' \end{pmatrix}$$

Using homogeneous coordinates, this transformation can be represented as

$$\begin{Bmatrix} X \\ Y \\ 1 \end{Bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/d & 0 & 0 \\ 0 & 1/d & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

The coordinates of the 30 training samples after the coordinate transformation are shown in Table 3. As we can see from the table, the coordinates of P_0 and P_{r1} remain the same for all samples. In this way, the coordinates of all the sampling points are defined in the same scale and their relationship are standardized. The actual input will reduce two points P_0 and P_{r1} when finding the critical factor and optimal regression models.

Input										Output		
x_{l2}	y_{l2}	x_{l1}	y_{l1}	x_0	y_0	x_{r1}	y_{r1}	x_{r2}	y_{r2}	**type**	x_n	y_n
1	1.1	0	1.3	0	0	1	0	2	0	2	1	1.1
1	2.37	0	1.3	0	0	1	0	2	0	0	1	1.1
1	2.37	0	1.3	0	0	1	0	1	1.1	1	1	1.1
1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	2	1.18	0.91
1.18	0.91	0	0.91	0	0	1	0	2	-0.36	2	1.18	0.91
0.9	0.8	0	1.1	0	0	1	0	2	0	2	0.9	0.8
-1	1.3	0	1.1	0	0	1	0	2	0	0	0.9	0.8
0.4	2.2	0	1.1	0	0	1	0	2	0	0	0.9	0.8
-1	1.3	0	1.1	0	0	1	0	0.9	0.8	1	0.9	0.8
0.4	2.2	0	1.1	0	0	1	0	0.9	0.8	1	0.9	0.8
0.31	2.46	0.15	1.23	0	0	1	0	1.08	-1.38	0	1.51	1.06
1.51	1.06	0.15	1.23	0	0	1	0	1.08	-1.38	2	1.51	1.06
0.31	2.46	0.15	1.23	0	0	1	0	1.51	1.06	1	1.51	1.06
1	1	-0.1	0.8	0	0	1	0	2	0	2	1	1
-1	1.1	-0.1	0.8	0	0	1	0	2	0	0	1	1
-1	1.1	-0.1	0.8	0	0	1	0	1	1	1	1	1
0	2	0	1	0	0	1	0	0.8	1.1	1	0.8	1.1
0	2	0	1	0	0	1	0	1	-1	0	0.8	1.1
0.8	1.1	0	1	0	0	1	0	1	-1	2	0.8	1.1
0	2	0	1	0	0	1	0	2.2	1.6	0	0.8	1.1
0.8	1.1	0	1	0	0	1	0	2.2	1.6	2	0.8	1.1
-1.1	0.8	0	1	0	0	1	0	2	0	0	1	1
1	1	0	1	0	0	1	0	2	0	2	1	1
-1.6	2.2	0	1	0	0	1	0	2	0	0	1	1
-1.1	0.8	0	1	0	0	1	0	1	1	1	1	1
-1.6	2.2	0	1	0	0	1	0	1	1	1	1	1
0	2	0	1	0	0	1	0	1	1	1	1	1
0	2	0	1	0	0	1	0	1	-1	0	1	1
1	1	0	1	0	0	1	0	1	-1	2	1	1
-1	1	0	1	0	0	1	0	2	0	0	1	1

(a) Normalized data for input 4 leading points.

Input														Output		
X_{l3}	Y_{l3}	X_{l2}	Y_{l2}	X_{l1}	Y_{l1}	X_0	Y_0	X_{r1}	Y_{r1}	X_{r2}	Y_{r2}	X_{r3}	Y_{r3}	type	X_n	Y_n
1.9	0.8	1	1.1	0	1.3	0	0	1	0	2	0	3	0	2	1	1.1
1.4	2.2	1	1.1	0	1.3	0	0	1	0	2	0	3	0	2	1	1.1
1.9	0.8	1	1.1	0	1.3	0	0	1	0	2	0	1.9	0.8	2	1	1.1
1.4	2.2	1	1.1	0	1.3	0	0	1	0	2	0	1.9	0.8	2	1	1.1
1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	3	0	0	1	1.1
0.7	3.3	1	2.37	0	1.3	0	0	1	0	2	0	3	0	0	1	1.1
1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	0	1	1.1
0.7	3.3	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	0	1	1.1
1.4	2.2	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	1	1	1.1
0.7	3.3	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	1	1	1.1
1.4	2.2	1	2.37	0	1.3	0	0	1	0	1	1.1	1.4	2.2	1	1	1.1
0.7	3.3	1	2.37	0	1.3	0	0	1	0	1	1.1	1.4	2.2	1	1	1.1
2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0	-0.91	2	1.18	0.91
2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0.91	-1.82	2	1.18	0.91
2.15	0	1.18	0.91	0	0.91	0	0	1	0	2	-0.36	0.91	-1.82	2	1.18	0.91
2.15	0	1.18	0.91	0	0.91	0	0	1	0	2	-0.36	2.03	-2.27	2	1.18	0.91
2.15	0	1.18	0.91	0	0.91	0	0	1	0	2	-0.36	2.15	0	2	1.18	0.91
2.15	0	1.18	0.91	0	0.91	0	0	1	0	2	-0.36	3.04	-1	2	1.18	0.91
2	1	0.9	0.8	0	1.1	0	0	1	0	2	0	3	0	2	0.9	0.8
2	1	0.9	0.8	0	1.1	0	0	1	0	2	0	2	1	2	0.9	0.8
-1	0	-1	1.3	0	1.1	0	0	1	0	2	0	3	0	0	0.9	0.8
0	2.37	-1	1.3	0	1.1	0	0	1	0	2	0	3	0	0	0.9	0.8

-1	0	-1	1.3	0	1.1	0	0	1	0	2	0	2	1	0	0.9	0.8
0	2.37	-1	1.3	0	1.1	0	0	1	0	2	0	2	1	0	0.9	0.8
2	1	0.4	2.2	0	1.1	0	0	1	0	2	0	3	0	0	0.9	0.8
2.5	2.23	0.4	2.2	0	1.1	0	0	1	0	2	0	3	0	0	0.9	0.8
0	2.37	0.4	2.2	0	1.1	0	0	1	0	2	0	3	0	0	0.9	0.8
1.1	3.34	0.4	2.2	0	1.1	0	0	1	0	2	0	3	0	0	0.9	0.8
2	1	0.4	2.2	0	1.1	0	0	1	0	2	0	2	1	0	0.9	0.8
2.5	2.23	0.4	2.2	0	1.1	0	0	1	0	2	0	2	1	0	0.9	0.8

(b) Normalized data for input 6 leading points.

Input																		Output		
X_{l4}	Y_{l4}	X_{l3}	Y_{l3}	X_{l2}	Y_{l2}	X_{l1}	Y_{l1}	X_0	Y_0	X_{r1}	Y_{r1}	X_{r2}	Y_{r2}	X_{r3}	Y_{r3}	X_{r4}	Y_{r4}	type	X_n	Y_n
2.1	3.34	0.7	3.3	1	2.37	0	1.3	0	0	1	0	2	0	3	0	3	1	0	1	1.1
3	1	1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	3	1	0	1	1.1
1	1.1	1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	3	1	0	1	1.1
3.5	2.23	1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	3	1	0	1	1.1
2.1	3.34	1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	3	1	0	1	1.1
3	1	1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	1	1.1	0	1	1.1
1	1.1	1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	1	1.1	0	1	1.1
3.5	2.23	1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	1	1.1	0	1	1.1
2.1	3.34	1.4	2.2	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	1	1.1	0	1	1.1
2.1	3.34	0.7	3.3	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	3	1	0	1	1.1
2.1	3.34	0.7	3.3	1	2.37	0	1.3	0	0	1	0	2	0	1.9	0.8	1	1.1	0	1	1.1
3	1	1.4	2.2	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	2	0	1	1	1.1
3.5	2.23	1.4	2.2	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	2	0	1	1	1.1
2.1	3.34	1.4	2.2	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	2	0	1	1	1.1
3	1	1.4	2.2	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	3	1	1	1	1.1

3.5	2.23	1.4	2.2	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	3	1	1	1	1.1
2.1	3.34	1.4	2.2	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	3	1	1	1	1.1
2.1	3.34	0.7	3.3	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	2	0	1	1	1.1
2.1	3.34	0.7	3.3	1	2.37	0	1.3	0	0	1	0	1	1.1	1.9	0.8	3	1	1	1	1.1
2.1	3.34	0.7	3.3	1	2.37	0	1.3	0	0	1	0	1	1.1	1.4	2.2	3	1	1	1	1.1
2.1	3.34	0.7	3.3	1	2.37	0	1.3	0	0	1	0	1	1.1	1.4	2.2	3.5	2.23	1	1	1.1
2.1	3.34	0.7	3.3	1	2.37	0	1.3	0	0	1	0	1	1.1	1.4	2.2	2.1	3.34	1	1	1.1
2	-0.36	2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0	-0.91	0	0	2	1.18	0.91
3	0.27	2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0	-0.91	0	0	2	1.18	0.91
2	-0.36	2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0	-0.91	0	-1.82	2	1.18	0.91
3	0.27	2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0	-0.91	0	-1.82	2	1.18	0.91
2	-0.36	2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0.91	-1.82	0	-1.82	2	1.18	0.91
3	0.27	2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0.91	-1.82	0	-1.82	2	1.18	0.91
2	-0.36	2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0.91	-1.82	0.91	-2.73	2	1.18	0.91
3	0.27	2.15	0	1.18	0.91	0	0.91	0	0	1	0	0.73	-0.82	0.91	-1.82	0.91	-2.73	2	1.18	0.91

(c) Normalized data for input 8 leading points.

Table 3. Normalized sample data.

Chapter 4
Finding Critical Factors and Regression Models

After data collection, the next step is to find influence of each leading points and how many leading points should be taken as the critical factors to generate the regression model. Then the regression models will be generated based on the critical factors. The non-significant leading points will be dropped in order to improve the efficiency of the regression models. We use *Auto2Fit*, a mature software kit, to find the critical factors and optimal regression model for mesh generation.

4.1 Introduction of Auto2Fit software kit

Auto2Fit is an optimization analysis comprehensive software kit developed by 7D-Soft High Technology Inc, P. R. China. In non-linear regression, curve fitting, non-linear complex projects model parameters estimate solution, etc., the kit shows excellent influence in the result. The core of the software is based on a general global optimization algorithm named Universal Global Optimization (*UGO*) algorithm published by the 7D-Soft High Technology Inc. The main characterization of this algorithm is to overcome the limitation when using the repetitive process in optimized computation, an appropriate initial value have to be given at first. If the initial value given in an unreasonable range, it is very difficult to approach the convergence value that means the correct result will not be obtained finally. The appropriate initial value is quite difficult to be estimated especially in the cases with too many input parameters. An initial value of parameters does not need to be indicated manually but can be given by the *UGO* algorithm of *Auto2Fit* software kit stochastically. Finally, the user can get the optimal solution. In nearly 90% of the cases, *Auto2Fit* software kit can obtain the correct result with any stochastic initial value based on its fault-tolerant and optimization property.

4.2 Critical factors analysis

As illustrated in Section 2.1.2, the input of element extraction method includes reference point and corresponding leading points. Ideally, all the points on the existing boundary should be taken as the input data for generating a new element. However, too many nodes on the input layer may increase the running time when generating the training sample and the regression model. It also increases the difficulty in standardizing the character of input data. We have to make the number of leading points as few as possible. Only those leading points that play an important role in extracting a new element will be retained as critical factors. Base on the leading points retained, we can standardize the input data.

4.2.1 Linear model introduction

We drop $P_0'(X_0, Y_0) = (0, 0)$ and $P_{r1}'(X_{r1}, Y_{r1}) = (1, 0)$ from the input columns since they are constants after normalized.

When finding the critical factors, we can drop the pair of leading points that have less than 5% contribution to the new point.

For example, when the input points (1 reference point with even leading points) are 11, the input data set of the method has 18 input columns: X_{l5}, Y_{l5}, X_{l4}, Y_{l4}, X_{l3}, Y_{l3}, X_{l2}, Y_{l2}, X_{l1}, Y_{l1}, X_{r2}, Y_{r2}, X_{r3}, Y_{r3}, X_{r4}, Y_{r4}, X_{r5} and Y_{r5}. The regression models need at least 4 leading points with 1 reference point to generate the quadrilateral element since define the type of the output need 5 input points.

Since we only need the rough weight of each leading point, the input, output and the linear model based on *UGO* algorithm are shown below:

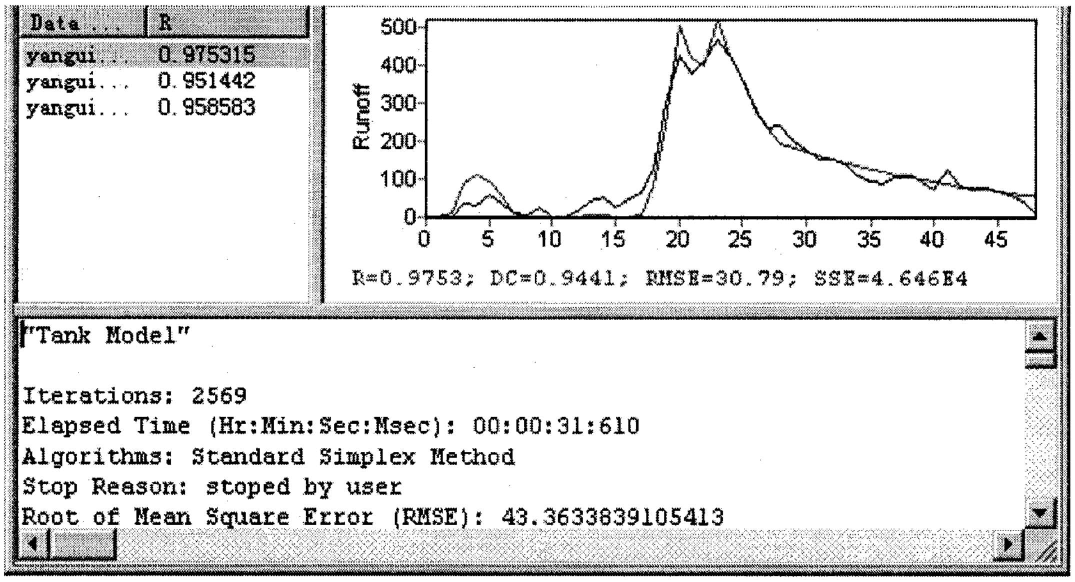
Input : $P_{(l,N)}', P_{(l,N-1)}', \ldots, P_{l2}', P_{l1}', P_0', P_{r1}', P_{r2}', \ldots, P_{(r,N-1)}', P_{(r,N)}'$

Output : $[type, P_n'(X_n, Y_n)]$

$$P_{type,x_n,y_n}' = \sum_{i=1, i\neq N+m}^{i=(2N+1)} (\alpha_i X_i + \beta_i Y_i), m = 1,2$$

X_i and Y_i are the coordinates of *Pi'*, α_i and β_i are the weight of each coordinate.

They might be negative since the coordinate of each leading point might be negative after normalized. The larger the absolute value of the weight is, the more significant the input coordinate value is. Since X_i and Y_i means the coordinate of x axis and y axis for one input point, we take $\frac{\alpha_i + \beta_i}{2}$ as the weight of that input point. Figure 32 is the interface of the software.



```
yangui...   0.975315
yangui...   0.951442
yangui...   0.958583
```

```
R=0.9753; DC=0.9441; RMSE=30.79; SSE=4.646E4
```

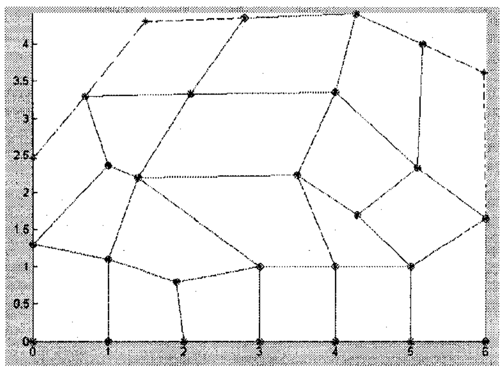
```
"Tank Model"

Iterations: 2569
Elapsed Time (Hr:Min:Sec:Msec): 00:00:31:610
Algorithms: Standard Simplex Method
Stop Reason: stoped by user
Root of Mean Square Error (RMSE): 43.3633839105413
```

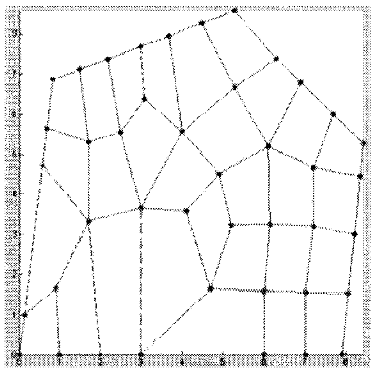
Figure 32. Interface of Auto2Fit software.

4.2.2 Experiment result

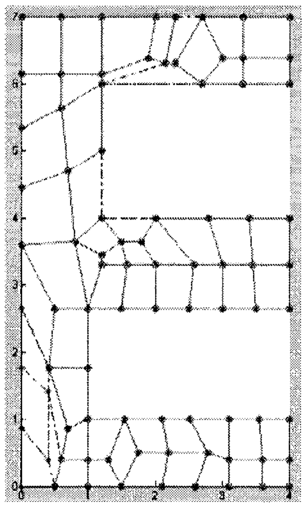
We used six generated meshes shown in Figure 33 to extract the training sample data. Base on the data sets, we did some experiments to find the weight of each leading point and discard the non-significant points.

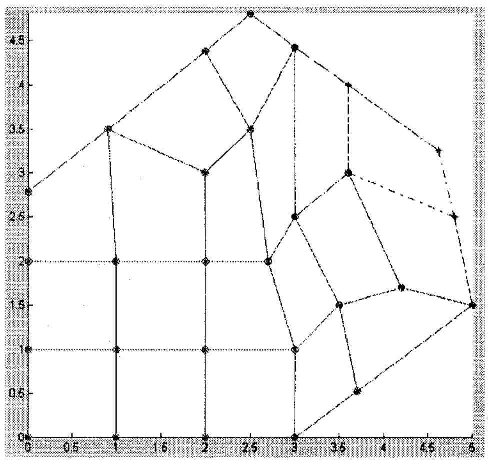


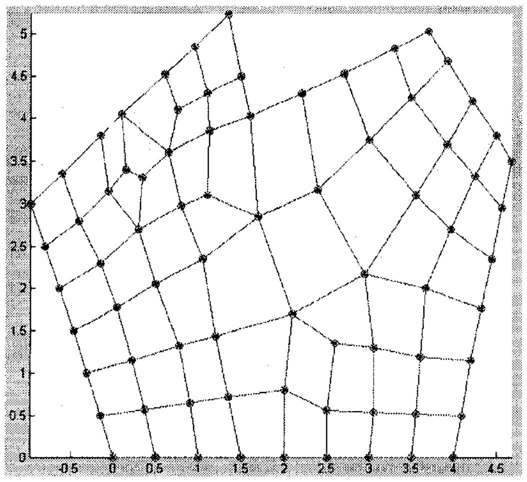
(a) Mesh #1.



(b) Mesh #2.


(c) Mesh #3.


(d) Mesh #4.


(e) Mesh #5.

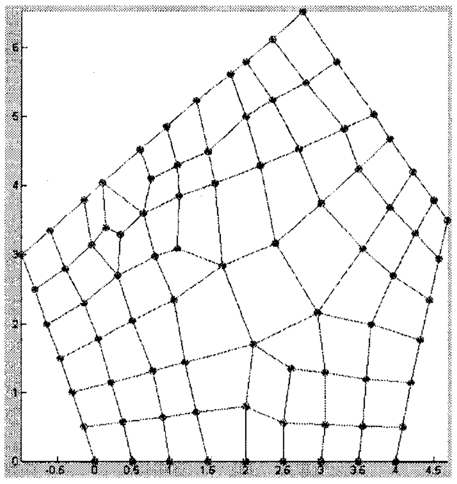

(f) Mesh #6.

Figure 33. Six generated meshes for extracting sample data.

The quadrilaterals with real line boundary are the good-quality elements that will be trained in the experiments. The dashed boundary quadrilaterals are the bad elements that cannot be used in training.

Table 4 is the number of samples of each mesh with different number of input points.

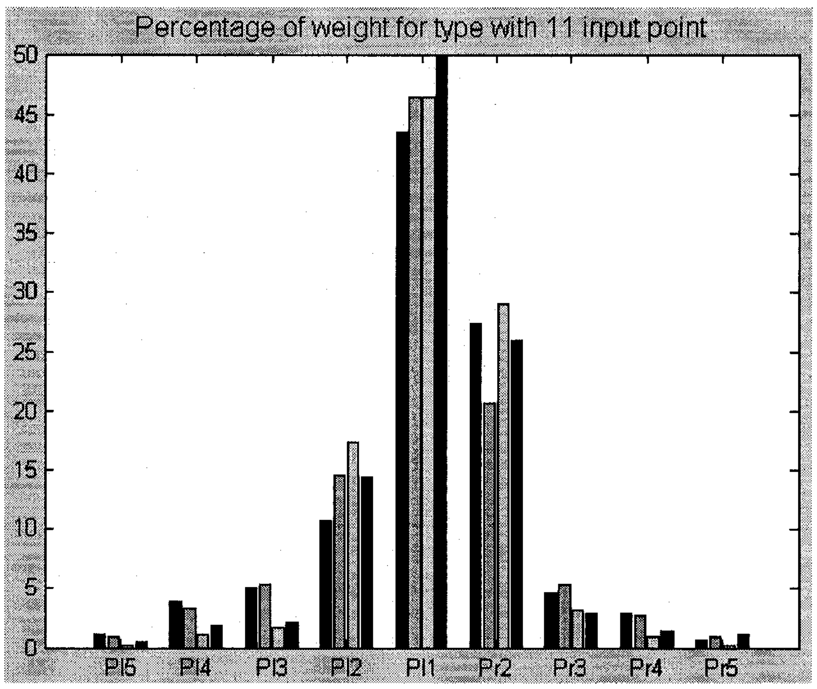
Input point num / Mesh num	11	9	7	5
Mesh #1	12463	3486	933	256
Mesh #2	27400	7095	1694	420
Mesh #3	52155	12614	2991	798
Mesh #4	35641	4938	1250	294
Mesh #5			8390	1416
Mesh #6			10150	1660

Table 4. Samples number of each mesh with different number of leading points.

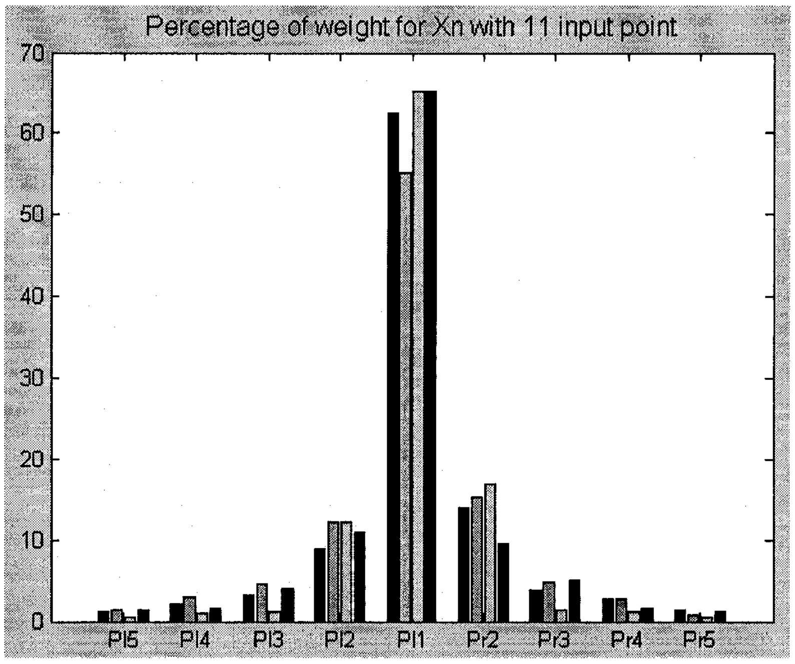
We only do the experiments for Mesh #1, Mesh #2, Mesh #3 and Mesh #4 for 9 and 11 input point case. After that, in order to prove the result strongly, we add 2 more meshes (Mesh #5 and Mesh #6) to do the experiments for 7 and 5 input point case.

1. Result of 11 input points (input of 9 points to the linear model):

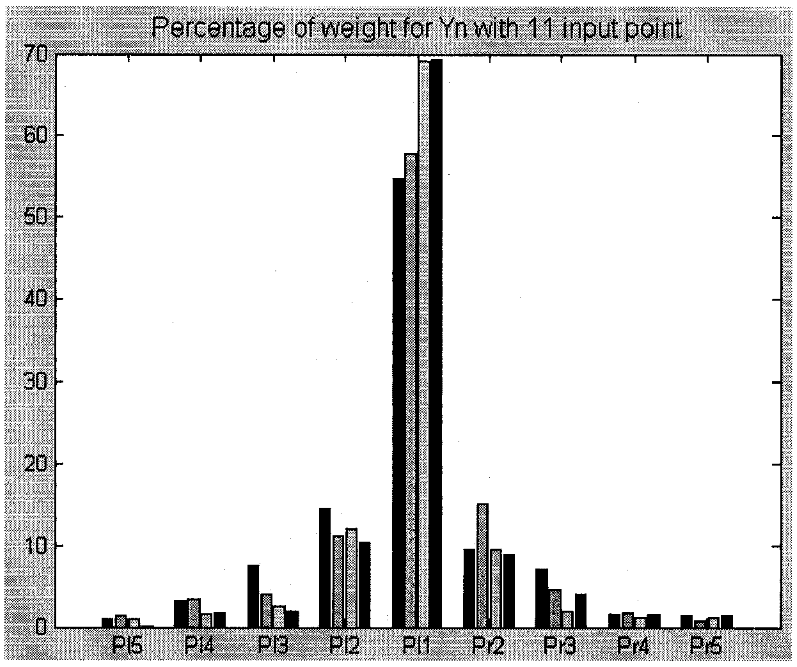
Figure 34 is the percentage of weight for each output. From the figure we can see, the weight of the outermost pair leading points P_{l5}' and P_{r5}' is less than 2.5%. It means that P_{l5}' and P_{r5}' are non-significant points when generate mesh.



(a) Percentage of weight for *type*.



(b) Percentage of weight for X_n.

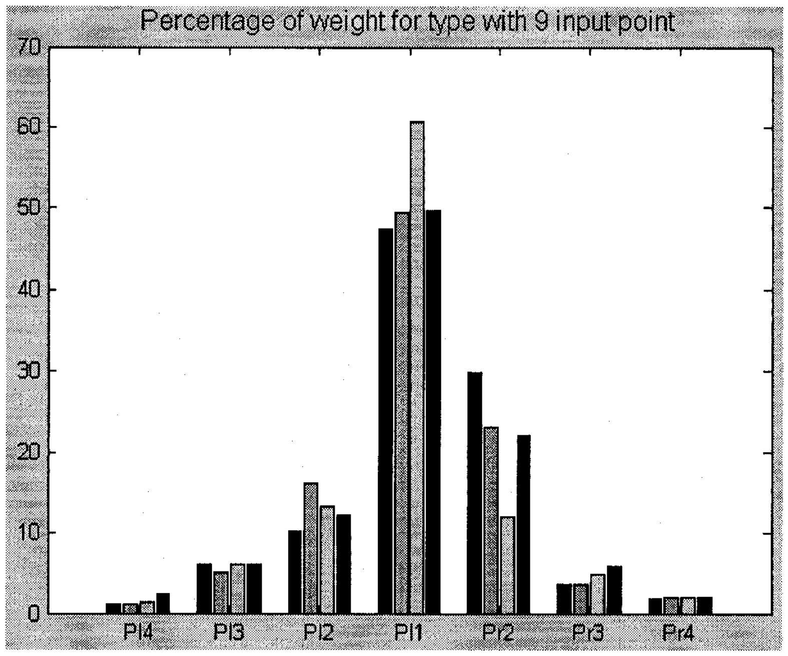


(c) Percentage of weight for Y_n.

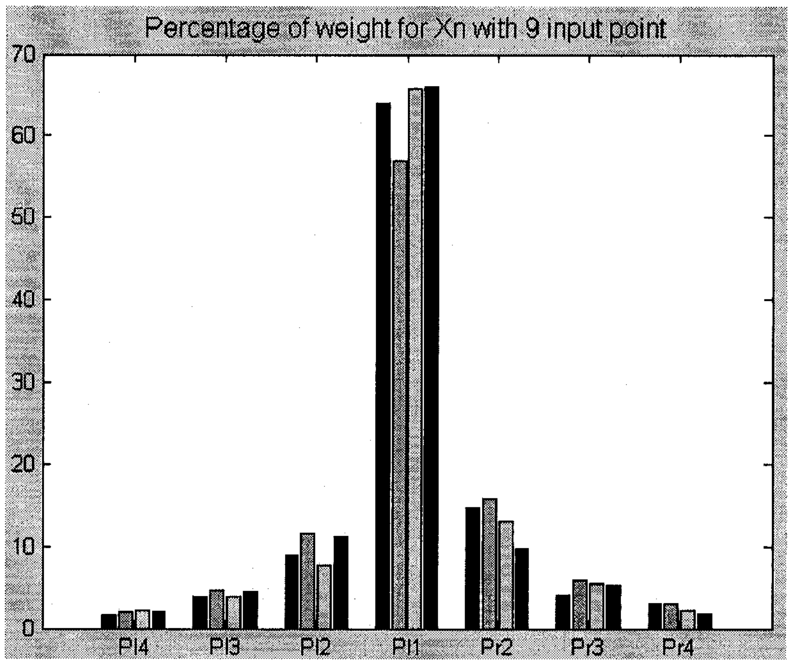
Figure 34. Percentage of weight for *type*, X_n and Y_n with 11 input points.

2. Result of 9 points (input of 7 points to the linear model):

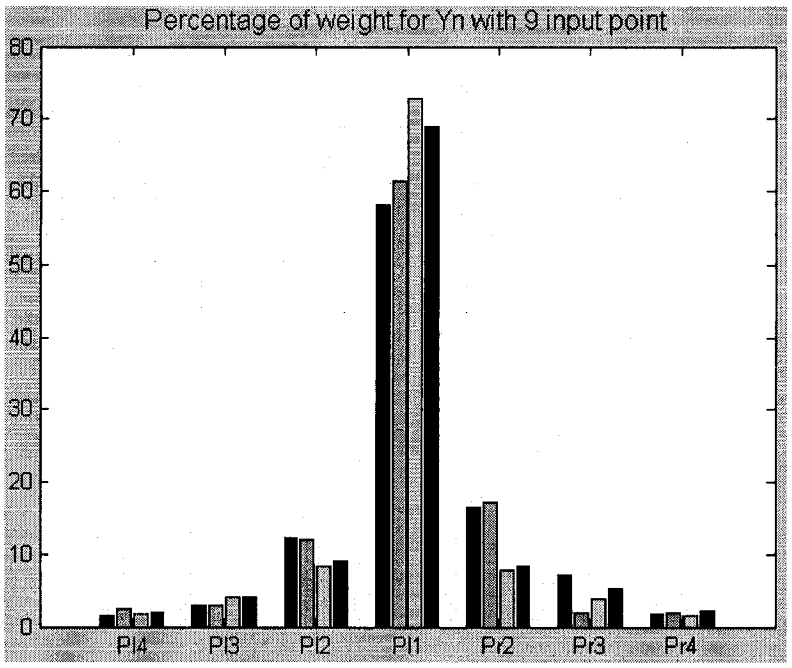
Figure 35 is the percentage of weight for each output. From the figure we can see, the weight of the outermost pair leading points P_{l4}' and P_{r4}' is less than 5%. So P_{l4}' and P_{r4}' are non-significant points when generate mesh.



(a) Percentage of weight for *type*.



(b) Percentage of weight for X_n.

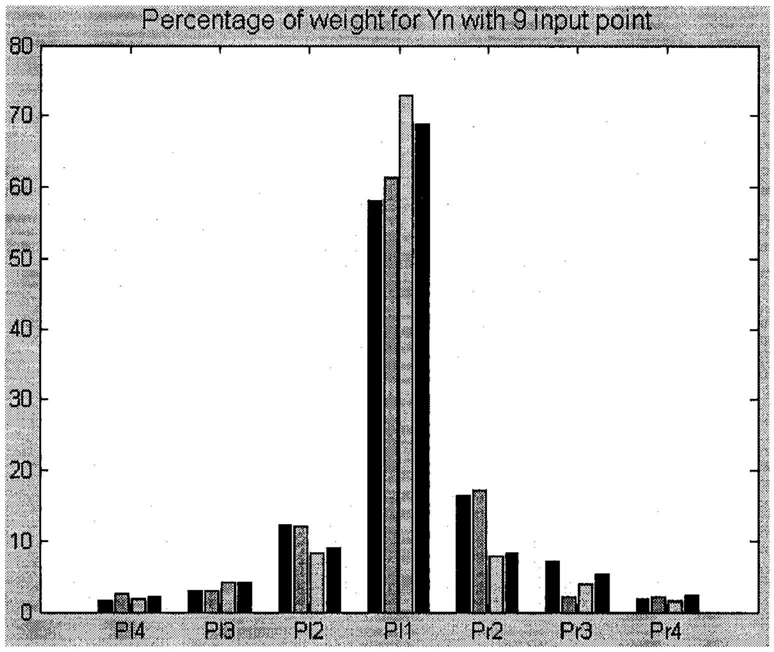


(c) Percentage of weight for Y_n.

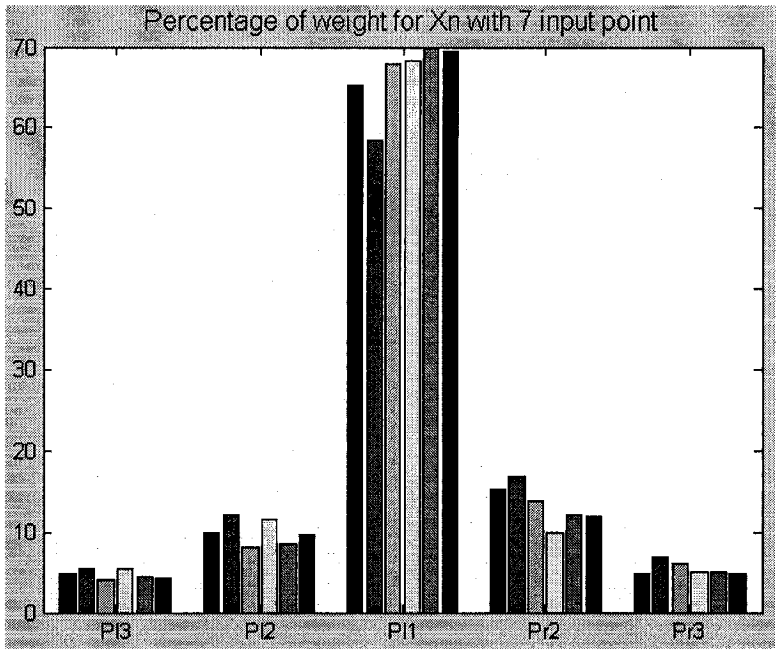
Figure 35. Percentage of weight for *type*, X_n and Y_n with 9 input points.

3. Result of 7 input points (input of 5 points to the linear model):

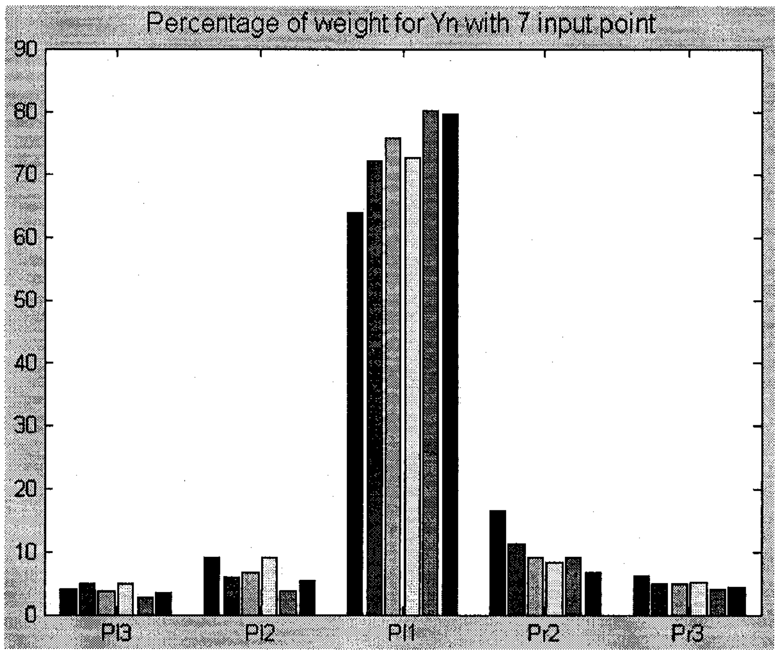
Figure 36 shows the percentage of weight for each output. From the figure we can see, the weight of the outermost pair leading points P_{l3}' and P_{r3}' is around 8-15%. P_{l3}' and P_{r3}' need to be kept in order to generate mesh elements more accurate.



(a) Percentage of weight for *type*.



(b) Percentage of weight for X_n.

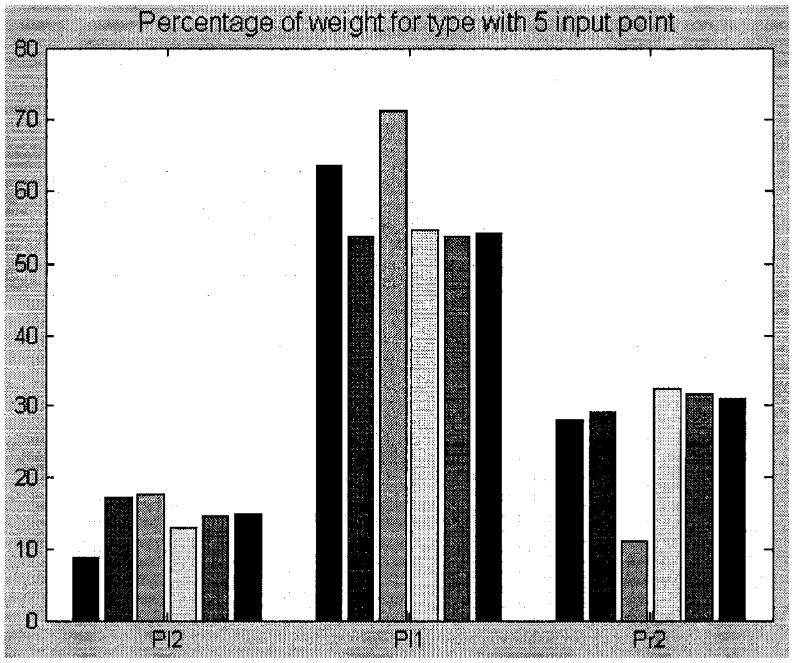


(c) Percentage of weight for Y_n.

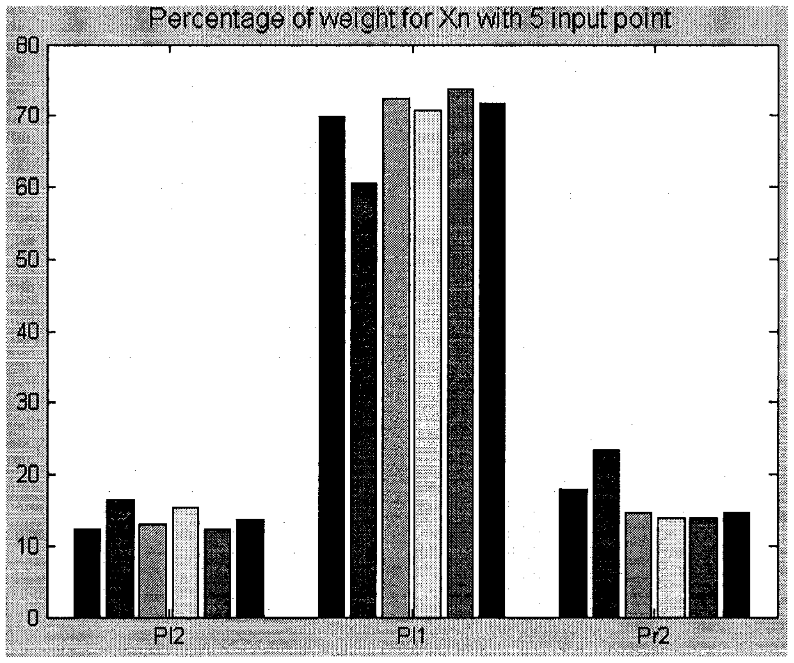
Figure 36. Percentage of weight for *type*, X_n and Y_n with 7 input points.

4. Result of 5 input point (input of 3 points to the linear model):

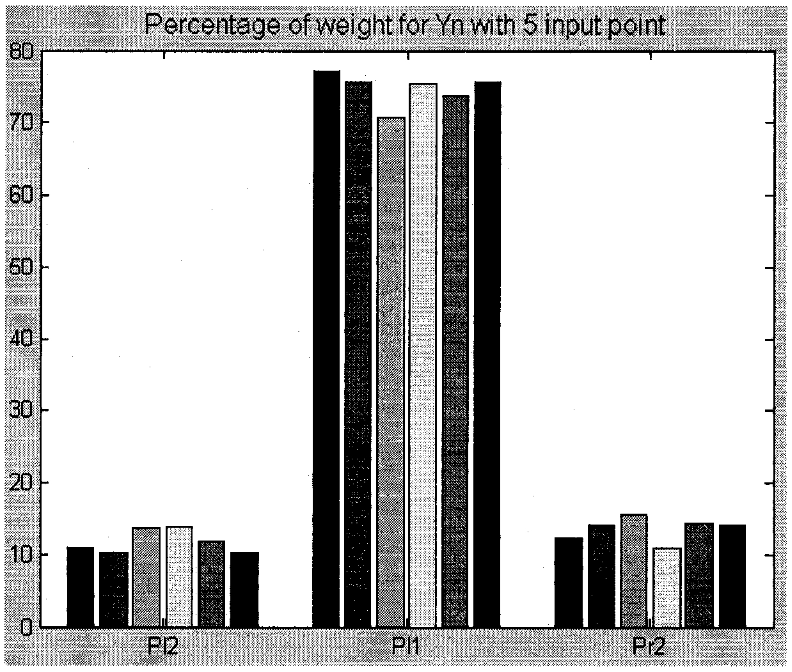
We also plot the percentage of weight for five input point case shown in Figure 37. For generating regression models, at least five input point have to be used since the output *type* needs five input point to determine. In Figure 37 we can see, P_{l1}', the first leading point on the left side is the most principal components when generate mesh element. P_{r1}', the symmetrical leading point of P_{l1}', who also play an important role is normalized as a constant (1,0) and not shown in the figure.



(a) The percentage of weight for *type*.



(b) The percentage of weight for X_n.



(c) The percentage of weight for Y_n.

Figure 37. Percentage of weight for *type*, X_n and Y_n with 5 input points.

4.2.3 Conclusion

The critical factors for each sample should be 7 input points (1 reference point and 6 leading points) when generating elements for mesh. The *correlation coefficient* of each regression models shown in Section 4.3.2 can also prove this conclusion. The regression model will be as shown below:

$$[type, P_n'(X_n, Y_n)] = f[P_{l3}', P_{l2}', P_{l1}', P_0', P_{r1}', P_{r2}', P_{r3}']$$

4.3 Optimal regression model

4.3.1 Non-linear model introduction

With the linear model, we can find the critical factors for the training sample data. That model cannot give us an accurate regression model to calculate the coordinates of P_n'. When generate regression models, we need some multi-variance non-linear regression models such as sigmoid, polynomial, exponential model, etc.

Based on the result given by linear model, we can see the most significant leading point pair is P_{l1}' and P_{r1}' (P_{r1}'normalized as constant and do not need to be taken into account when finding the non-linear regression model). P_{l1}' should have a high exponential coefficient in the polynomial regression model.

For the output *type*, the value is a constant 0, 1 or 2. The *type* value depends on the position of P_{l2}', P_{l1}' and P_{r2}'. For the value of output X_n and Y_n should be calculated by P_{l3}', P_{l2}', P_{l1}', P_{r2}' and P_{r3}'.

The sample data should be comprehensive. The more sample data we input, the more precision we will get. The bad element will be generated when there are not enough that kind of training samples which can generate good element for those input points.

The following parameters can indicate the effectiveness of the regression model.

RMSE: Root of mean square error

SSE: Sum of square error.

R: Correlation Coefficient which indicated the similarity between the output given by the training sample directly and the output given by the regression model. R=1 means the two outputs are same.

DC: Determination coefficient

4.3.2 Experiment result

1. Regression model for *type* with 5 input points:

As mentioned before, the output parameter *type* determined by P_{l2}, P_{l1}, P_0, P_{r1} and P_{r2}. We only need to take P_{l2}', P_{l1}' and P_{r2}' to generate the regression model for *type* although we will take seven input points to calculate the coordinate of the output point. The regression model we used is $P_{type}' = \mu_0 + \sum_{i=1, i\neq N+m}^{2N+1} \frac{\lambda_i}{1+e^{-\mu_1 X_i}} + \frac{\gamma_i}{1+e^{-\mu_1 Y_i}}, m = 1,2$

We did six experiments based on the six 2D meshes shown in Figure 33. Table 5 is the result calculated by the non-linear model of *UGO* algorithm. As the correlation coefficient *R* is around 0.77-0.85, the validity of the regression model is approach the goal of the thesis.

The value of parameter *type* calculated from the regression model should be transform to 0, 1 or 2. There might be some error when transform the *type* value. In our study, we transform the result as shown below:

$$P_{type}' = \mu_0 + \sum_{i=1, i\neq N+m}^{2N+1} \frac{\lambda_i}{1+e^{-\mu_1 X_i}} + \frac{\gamma_i}{1+e^{-\mu_1 Y_i}}, m = 1,2 \qquad \begin{array}{l} P_{type} = 0,\ P_{type} \le 0.75 \\ P_{type} = 1,\ 0.75 < P_{type} \le 1.5 \\ P_{type} = 2,\ \text{otherwise} \end{array}$$

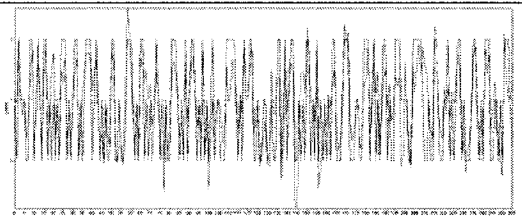
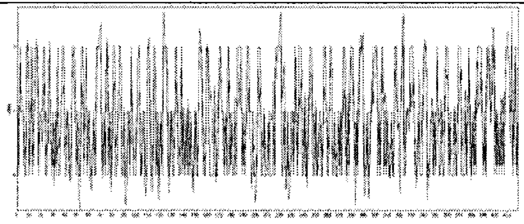
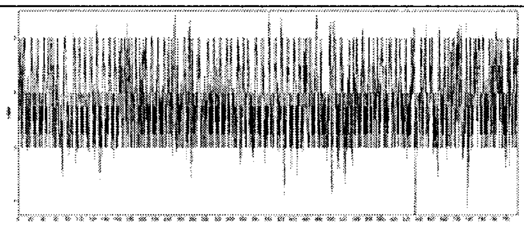
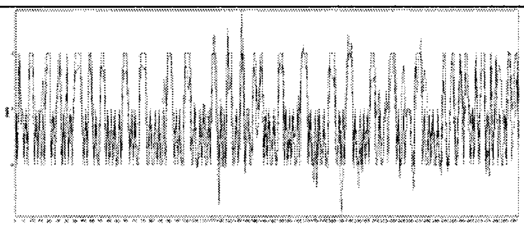
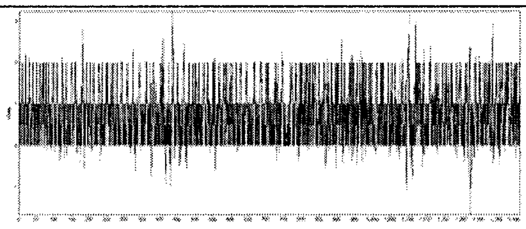
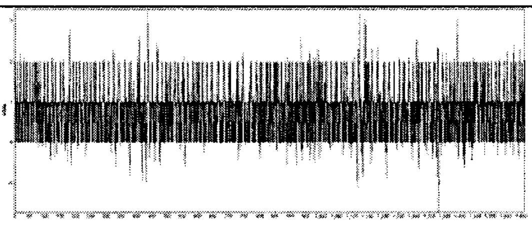
Mesh no.	Name	Value	Fitting image
Mesh #1	RMSE	0.5330	
	SSE	72.7263	
	R	0.7654	
	DC	0.5858	
Mesh #2	RMSE	0.4823	
	SSE	97.7086	
	R	0.8149	
	DC	0.6640	
Mesh #3	RMSE	0.4851	
	SSE	187.7522	
	R	0.8112	
	DC	0.6580	
Mesh #4	RMSE	0.4527	
	SSE	60.2518	
	R	0.8388	
	DC	0.7036	
Mesh #5	RMSE	0.4447	
	SSE	280.0795	
	R	0.8455	
	DC	0.7149	
Mesh #6	RMSE	0.4396	
	SSE	320.8431	
	R	0.8493	
	DC	0.7212	

Table 5. Six experiments for regression model of *type* with 5 input points.

In the fitting image shown in Table 5, the x axis shows sample number (in Table 4) and the y axis shows the value of *type*. The formal line is the output getting from the samples and the scraggly line is the output given by the non-linear regression model.

2. Regression model for X_n with 7 input points:

We take P_{l3}', P_{l2}', P_{l1}', P_{r2}' and P_{r3}' to generate the regression model for the output parameter X_n. Below is the regression model for X_n with 7 input point:

$$P_{x_n}' = \lambda_0 + \lambda_1 X_N^3 + \lambda_2 Y_N^3 + \lambda_3 X_N^2 + \lambda_4 Y_N^2 + \sum_{i=1, i \neq N+m}^{2N+1} (\varepsilon_i X_i + \tau_i Y_i)$$

$$m = 1, 2$$

We did six experiments based on the six meshes shown in Figure 33. Table 6 is the result calculated by the non-linear model of *UGO* algorithm. As the correlation coefficient *R* is around 0.69-0.85, the validity of the regression model is almost approach the goal of the thesis.

In the fitting image shown in Table 6, the x axis shows sample number (in Table 4) and the y axis shows is the value of X_n. The formal line is the output getting from the samples and the scraggly line is the output given by the non-linear regression model.

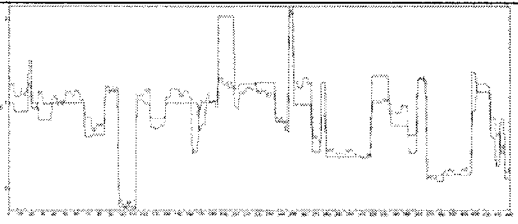
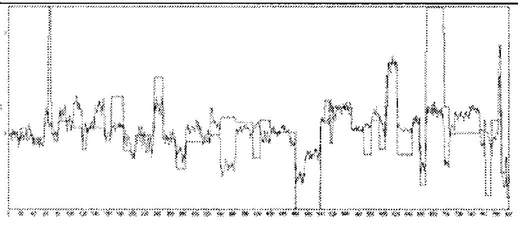
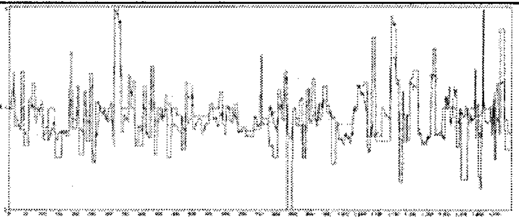
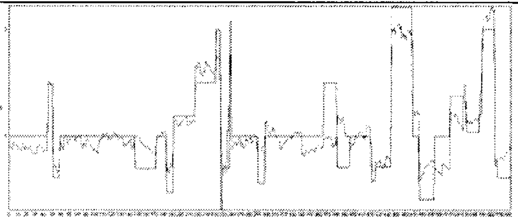
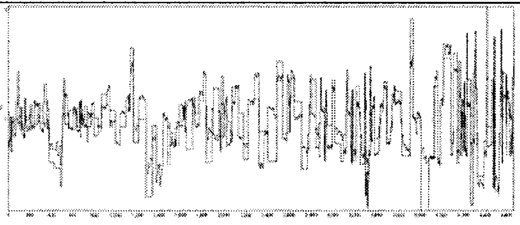
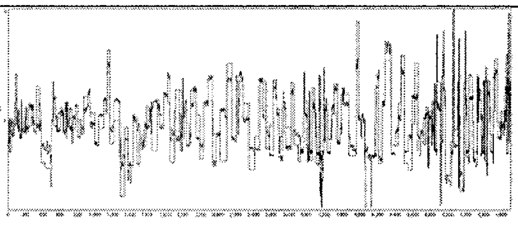
Mesh no.	Name	Value	Fitting image
Mesh #1	RMSE	0.2239	
	SSE	22.1158	
	R	0.8805	
	DC	0.7752	
Mesh #2	RMSE	0.3007	
	SSE	72.4169	
	R	0.5721	
	DC	0.3273	
Mesh #3	RMSE	0.2265	
	SSE	76.9642	
	R	0.6986	
	DC	0.4881	
Mesh #4	RMSE	0.1874	
	SSE	20.4675	
	R	0.8801	
	DC	0.7746	
Mesh #5	RMSE	0.2028	
	SSE	191.6953	
	R	0.7826	
	DC	0.6125	
Mesh #6	RMSE	0.1996	
	SSE	227.3746	
	R	0.7871	
	DC	0.6194	

Table 6. Six experiments for regression model of X_n with 7 input points.

3. Regression model for *Y_n* with 7 input points:

We take P_{l3}', P_{l2}', P_{l1}', P_{r2}' and P_{r3}' to generate the regression model for the output parameter Y_n. Below is the regression model for Y_n with seven input point:

$$P_{y_n}' = \eta_0 + \eta_1 X_N^3 + \eta_2 Y_N^3 + \eta_3 X_N^2 + \eta_2 Y_N^2 + \sum_{i=1, i \neq N+m}^{2N+1} (\chi_i X_i + \varphi_i Y_i)$$

$$m = 1, 2$$

We did 6 experiments based on the 6 meshes shown in Figure 33. Table 7 is the result calculated by the non-linear model of *UGO* algorithm. As the correlation coefficient *R* is around 0.72-0.92, the validity of the regression model is approach the goal of the thesis.

In the fitting image shown in Table 7, the x axis shows sample number (in Table 4) and the y axis shows is the value of Y_n. The formal line is the output getting from the samples and the scraggly line is the output given by the non-linear regression model.

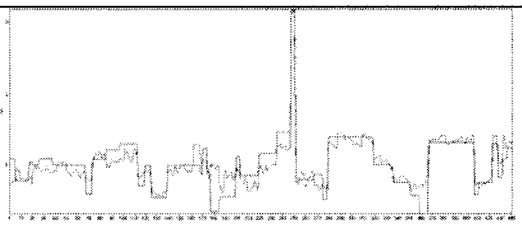
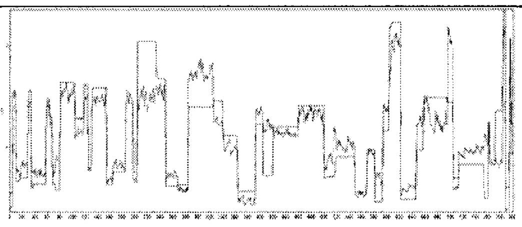
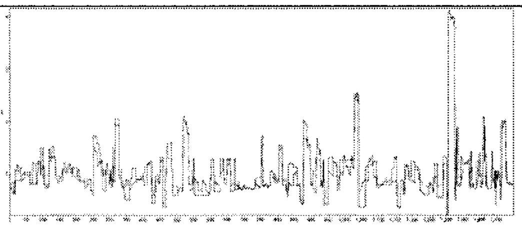
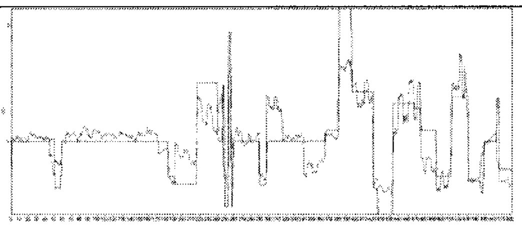
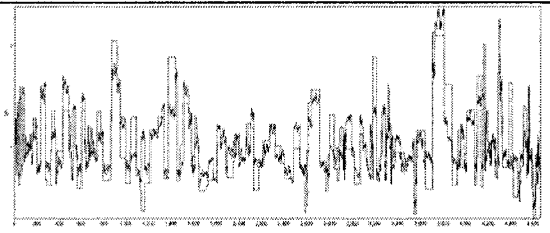
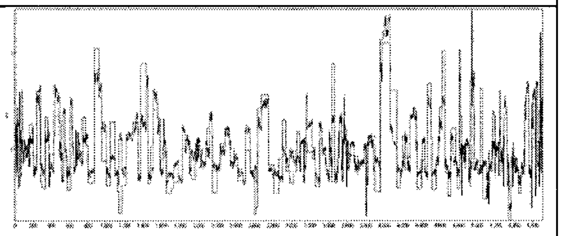
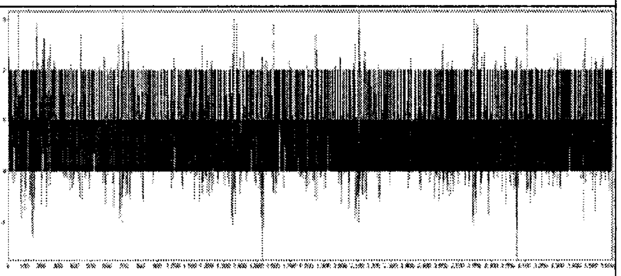
Mesh no.	Name	Value	Fitting image
Mesh #1	RMSE	0.1969	
	SSE	36.1870	
	R	0.9016	
	DC	0.8129	
Mesh #2	RMSE	0.2593	
	SSE	113.9005	
	R	0.8418	
	DC	0.7086	
Mesh #3	RMSE	0.1959	
	SSE	114.7615	
	R	0.9254	
	DC	0.8564	
Mesh #4	RMSE	0.1910	
	SSE	21.2608	
	R	0.8064	
	DC	0.6503	
Mesh #5	RMSE	0.2266	
	SSE	239.3993	
	R	0.8135	
	DC	0.6618	
Mesh #6	RMSE	0.2259	
	SSE	291.1988	
	R	0.8166	
	DC	0.6668	

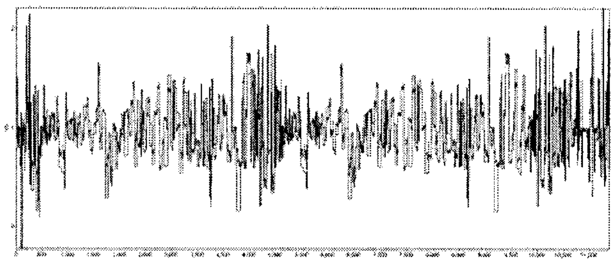
Table 7. Six experiments for regression model of Y_n with 7 input points.

4. Regression model of combined sample data for X_n and Y_n with 7 input points:

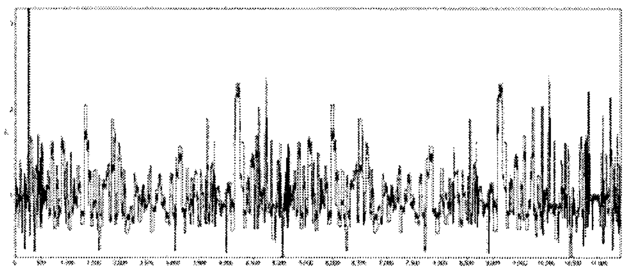
In order to get accurate regression model, we combined the sample data sets to get more input data. Mesh #1, Mesh #4 and Mesh #6 were taken to do the experiment since the error of Mesh #2 and Mesh #3 are higher than others and Mesh #5 is similar with Mesh #6. Table 8 is the result for *type*, X_n and Y_n. The formal line in the fitting image is the output getting from the samples and the scraggly line is the output given by the non-linear regression model.

Name	Value	Parameters	Fitting image
RMSE	0.4563	λ_1=1.8022 γ_1=-1.3188 λ_2=-1.2238 γ_2=1.9398 λ_5=-3.9728 γ_5=4.0131 μ_1=1.2008 μ_0=0.8325	 The x axis shows sample number from 1 to 12293 The y axis shows the value of *type*
SSE	755.1154		
R	0.8367		
DC	0.7000		

(a) Coefficients of the regression model for output *type*.

Name	Value	Parameters	Fitting image
RMSE	0.2116	λ_0=0.4567 λ_1=-0.2856 λ_3=0.0255 λ_2=0.0741 λ_4=-0.2922 ε_1=-0.0006 ε_2=0.0029 ε_3=-0.0301 ε_6=0.7916 ε_7=0.0385 τ_1=0.0018 τ_2=0.0052 τ_3=0.0239 τ_6=0.5909 τ_7=-0.0656	 The x axis shows sample number from 1 to 12293 The y axis shows the value of X_n
SSE	509.7539		
R	0.7787		
DC	0.6064		

(b) Coefficients of the regression model of output X_n.

Name	Value	Parameters	Fitting image
RMSE	0.2350	η_0=1.0933 η_1=-0.2256 η_2=-0.5996 η_3=-0.0729 η_4=2.2923 χ_1=0.0025 χ_2=0.0163 χ_3=-0.0469 χ_6=-0.1483 χ_7=0.0392 φ_1=-0.0025 φ_2=0.0019 φ_3=0.0011 φ_6=-1.9436 $\varphi 7$=-0.0611	 The x axis shows sample number from 1 to 12293 The y axis shows is the value of Y_n
SSE	628.8842		
R	0.7935		
DC	0.6296		

(c) Coefficients of the regression model of output Y_n.

Table 8. Experiment with more sample data for regression model.

The correlation coefficient which indicated the similarity of the regression models for 7 input point is shown in Figure 38. For each mesh, it has three bars. The first bar is the correlation coefficient for *type*, the second is for X_n and the third is for Y_n.

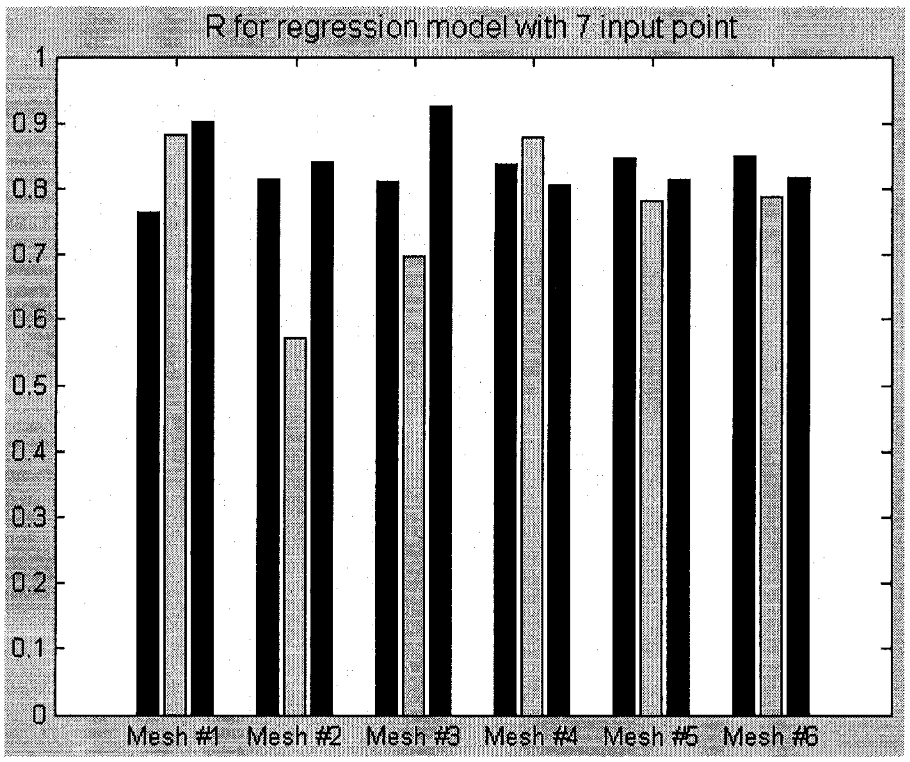


Figure 38. Correlation coefficient (R) of regression models with 7 input point.

We also test the regression model for 5 input point (1 reference point and 4 leading points) case and did some experiments to get the result for X_n and Y_n with 5 input points. The correlation coefficient R of the two regression models is around 0.70-0.75 that may not give us a good enough result.

4.3.3 Conclusion

After having tried many methods, the three models introduced in Section 4.3.2 will be taken as the optimal regression models and put into the mesh generate application. The correlation coefficient *R* of the three regression models are around 0.77-0.83.

Chapter 5
Case Study and Comparison

5.1 Introduction

On the foundation of the multivariate non-linear regression model in Section 4.3, a regression model based algorithm is proposed in this chapter.

For a given boundary, when generating elements, the algorithm will update reference point with new point based on the input points given by the boundary. The coordinate of new points will be calculated by the optimal regression model. After get the coordinate of new point, the reference point will be erased and the new point will be inserted to the boundary. The boundary will be updated for generating the next element. When a bad element is generated, we need to add that kind of samples to training sample data set and retrain the regression model. Then apply the regression model again. This procedure is repeated until the final pattern is satisfied on the existing boundary. After generation, the mesh should be de-noise in order to let the transformation from dense mesh to coarse mesh be smooth. In order to compare the result clearly, we use original mesh generated by each method.

Figure 39 is the flow chart of the mesh generation algorithm based on the regression models we get in Section 4.3.3.

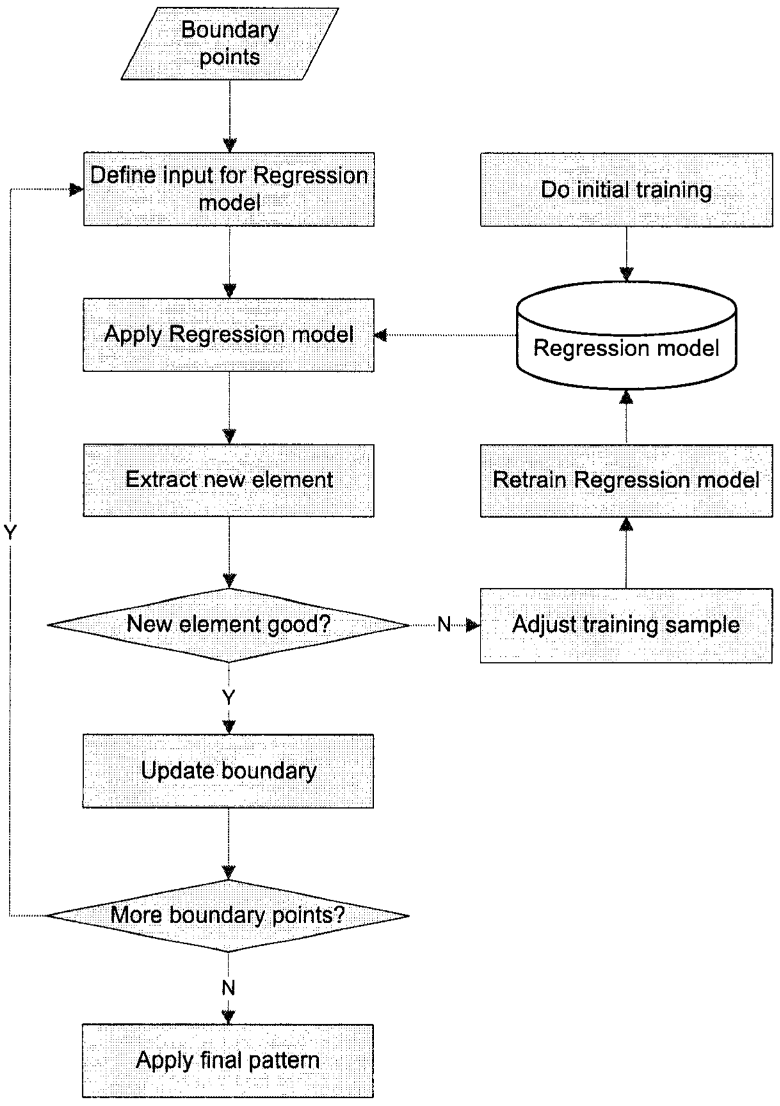
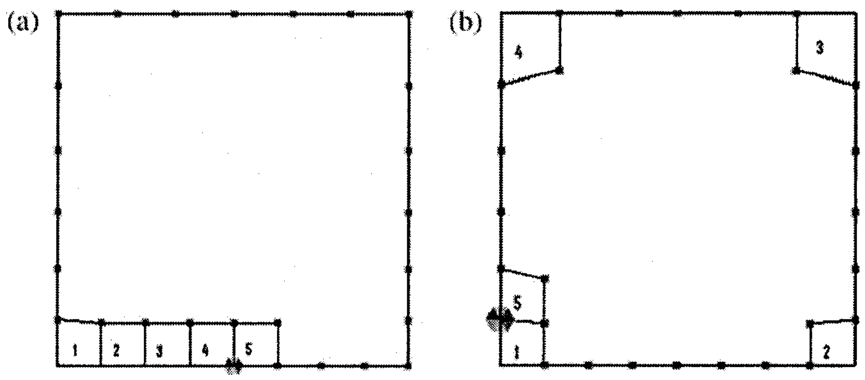


Figure 39. Flow chart of regression model-based mesh generation algorithm

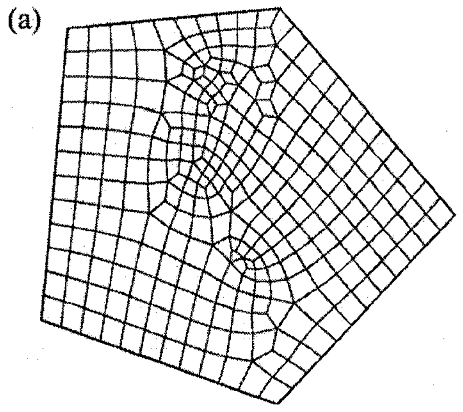
Once a new element is generated, the next reference point needs to be selected. Various criteria can be used to identify the next reference point. Figure 40 shows 2 examples. The method in Figure 40 (a) always chooses the first valid reference point next to the current one from the updated boundary. In another method, the updated boundary will not be processed until all the valid reference points in the original boundary have been processed. This is shown in Figure 40 (b). In the figure, the first four elements are generated from the original boundary and element 5 is generated in the new boundary since no more points meet the requirements of a reference point on the original boundary. If no reference point has been found, then a square element will be extracted around the shortest boundary segment.



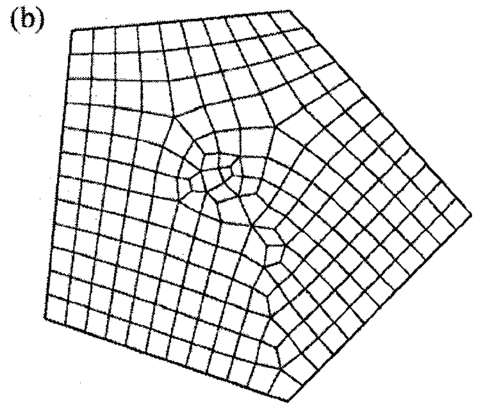
(a) Newer boundary first. **(b) Older boundary first.**

Figure 40. Finding reference point.

Using the two different methods to find reference points for the same domain, different meshes can be generated as shown in Figure 41. From the figure, it can be seen that the second method can generate more good-quality elements than the first one. The elements close to the periphery have no obvious difference, but there are more bad elements in the center using the first method. In the first method, the local boundary will become bad with the appearance of a bad element and continue to generate bad elements. However, the second method considers more boundary information by choosing all possible reference points from the original boundary and updates the original boundary with all newly generated elements.



(a) Newer boundary first.



(b) Older boundary first.

Figure 41. Generated meshes by using different methods to find reference point.

5.2 Generate mesh by using regression model

Figure 42 shows some numerical example generated by the regression models.

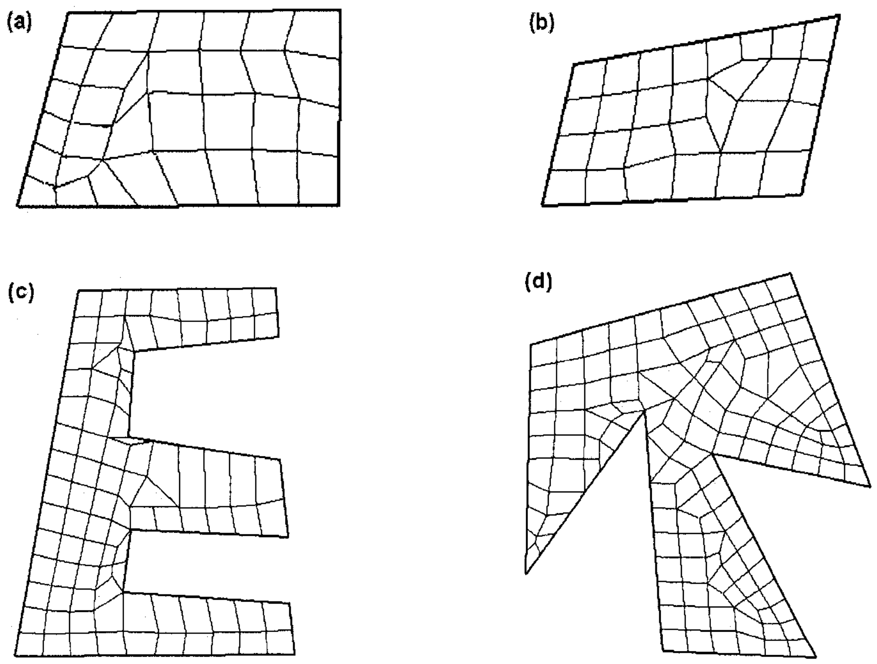
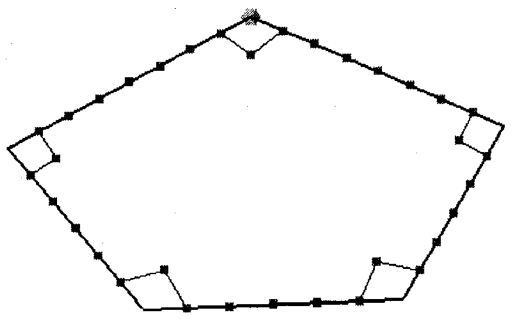
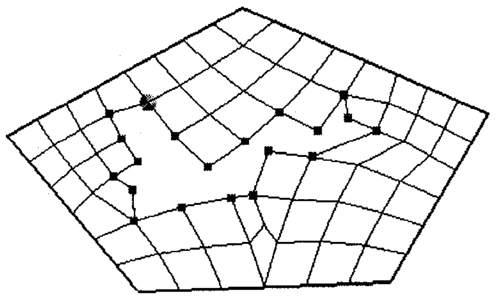
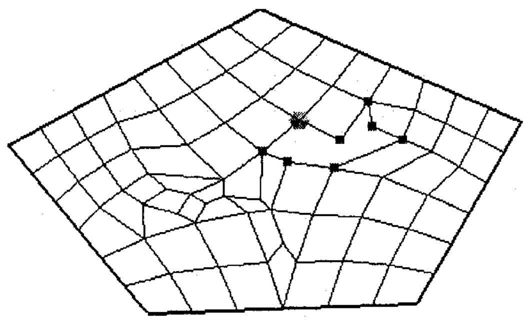


Figure 42. Meshes generated by regression model-based method.

Figure 43 shows the procedure when generate a mesh. For some simple boundary, the regression model method can finish generation automatically in one time. When the boundary is irregular or complex, choose reference point and generate some elements manually may be needed.


(a) The first 5 generated elements.


(b) More generated elements.


(c) Final pattern left.

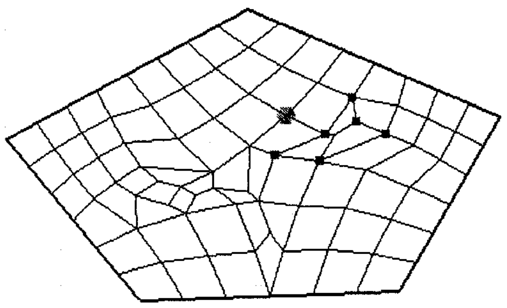

(d) The completed mesh.

Figure 43. Regression model-based mesh generation process of an example.

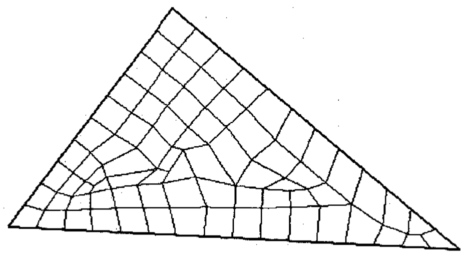
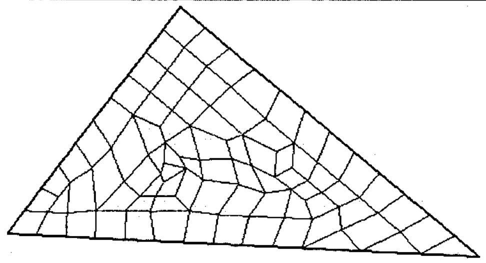
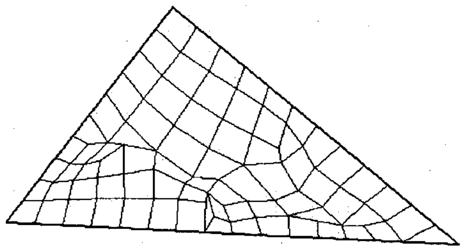
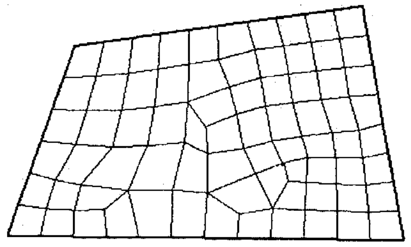
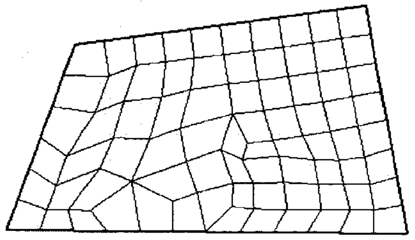
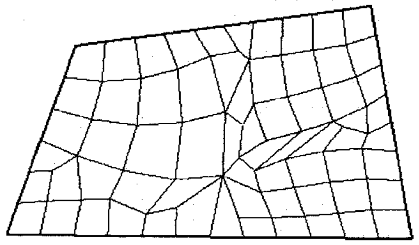
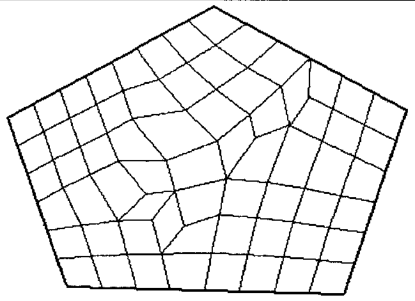
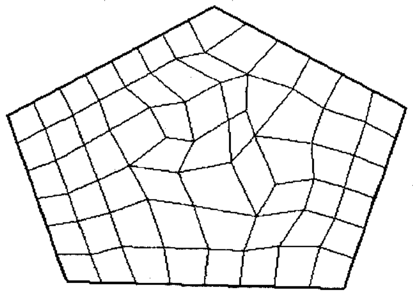
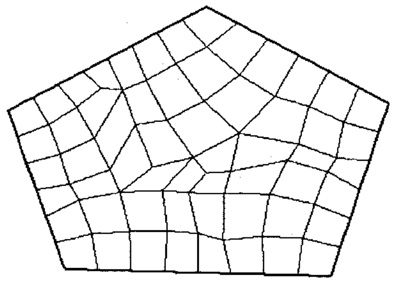
5.3 Comparison with existing algorithm

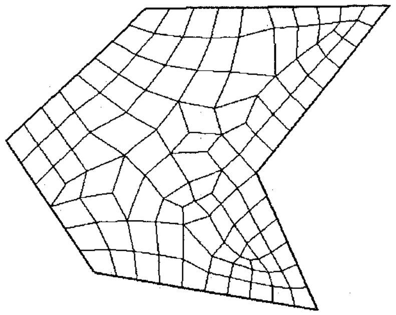
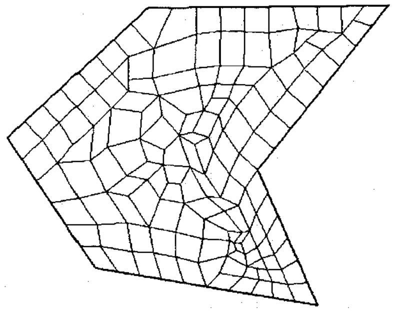
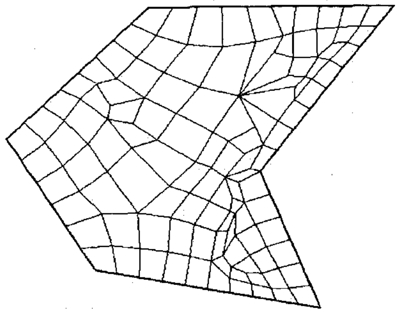
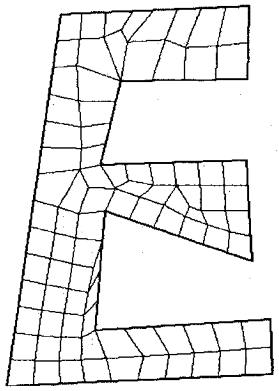
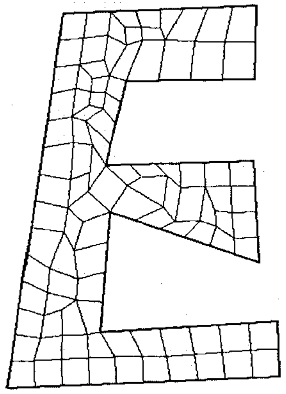
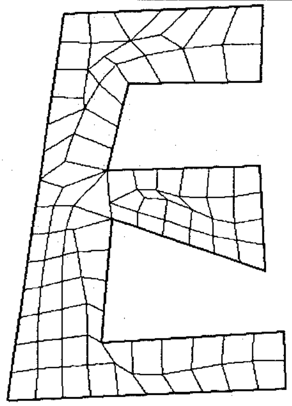
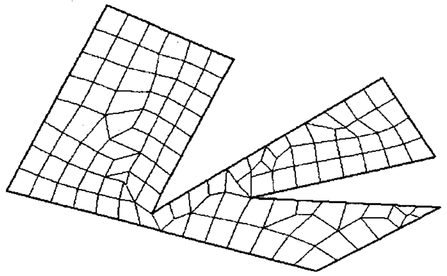
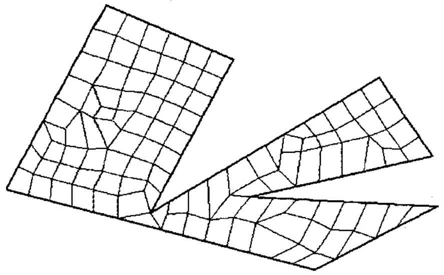
As mentioned earlier, knowledge-based method has the advantage of guaranteeing the quality of elements close to the domain boundary, but the disadvantage is also serious and obvious. To implement the this method, we need to consider the relationships of angles and segment lines to find the basic rules of creating an element based on local boundary features in a heuristic manner. The rules are used to reflect the relationship between boundary information and the element to be extracted. There are many ways to improve the quality of the generated mesh [23] [26]. If the quality is not satisfied, remeshing may need to be done as well [32] [33] [36]. However, it is difficult to define the rules to generate a good-quality element based on all boundary information.

For ANN-based method, presently, there are no general rules to determine the number of hidden layer for optimal network architecture although there is some research work contributed in this area. When processing a mesh generation request using the trained result, it is easy to get a bad element because of the interaction of the initial patterns. To correct those bad elements, we need to correct the bad element manually and add the pattern of the corrected element into the training samples. Meanwhile, more samples will be generated and added to the existing sample set to improve the impact of this pattern. It will take a long time to compile the training program when the sample data are huge. From Figure 44 we can see, ANN-based method cannot deal with the complex boundary and the quality of the elements generated by ANN-based method are not as good as we expect in Section 1.2. The transformation from dense mesh to coarse mesh is not as smooth as other methods. For some complex boundaries, the method cannot finish generating elements in the middle of the mesh.

For the purpose of our study, regression model method attempts to solve the problems that other cannot deal with. The input of the optimal regression model is a set of leading points that represent the significant information of the existing boundary. The output is the parameters used to describe an element [*type* P_{new}]. Using regression model method, we no longer need to develop basic rules to extract an element from the boundary feature manually. A number of training samples are used to learn the relationship between boundary information and the extracted element. We can decompose a complicated domain by correcting bad elements and adding more patterns into the training samples. Follow this way, mesh generation is also a self-learning process by adjusting training samples and much better than ANN-based method especially on the running time and the accuracy of the result. Regression model method provides an alternative method in mesh generation.

Figure 44 shows simple examples and complex examples generated by the above 3 methods, respectively. From the results, we can see that the elements around the periphery are still good since the boundary feature is simple. When the boundary feature becomes complex in the center, the basic rules in the element extraction method still consider local boundary information, so they cannot continue to generate good-quality elements. However, with regression method, we can continue to generate good-quality elements by adding more patterns into the training samples when the boundary condition becomes bad.

Regression model based method	Knowledge-based method	ANN-based method
		
		
		

		
		
		Cannot finish generating

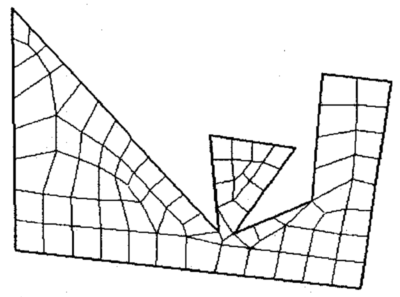
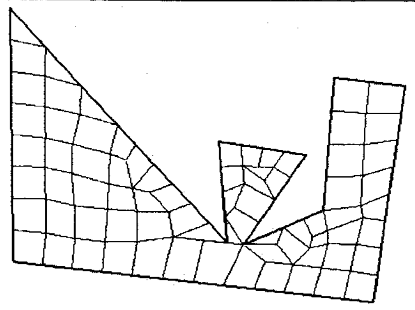
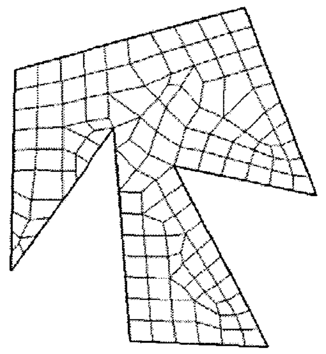
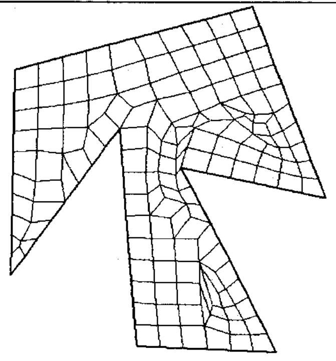
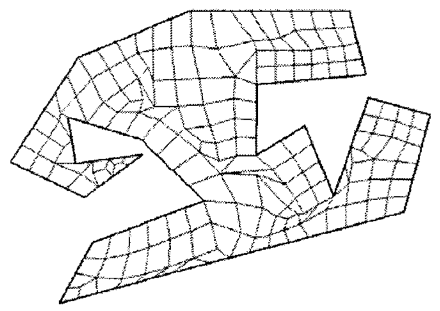
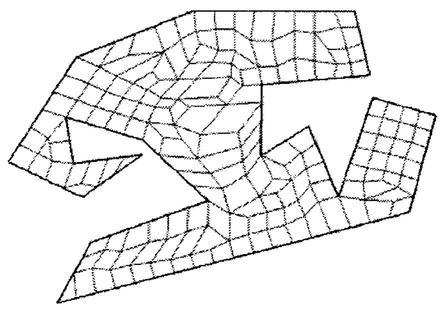
		Cannot finish generating
		Cannot finish generating
		Cannot finish generating

Figure 44. Mesh generation examples by different methods.

Because of the interaction of the initial patterns, bad element may be generated as shown in Figure 45 . After retraining the samples and applying the regression model again, we can get the correct element.

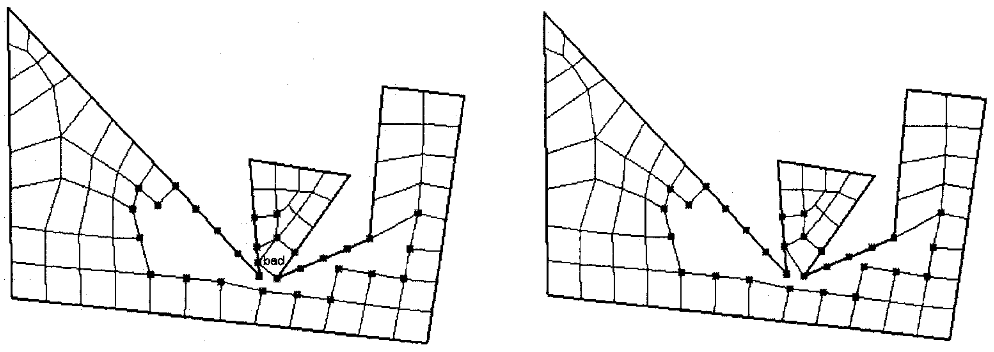


Figure 45. A bad element corrected by using regression model-based method.

For the mesh with sharp corner in Figure 46, the first element is especially important. The vertex on the corner should be taken as the first reference point. Otherwise, the boundary will become too complex at that corner and cannot generate good element anymore.

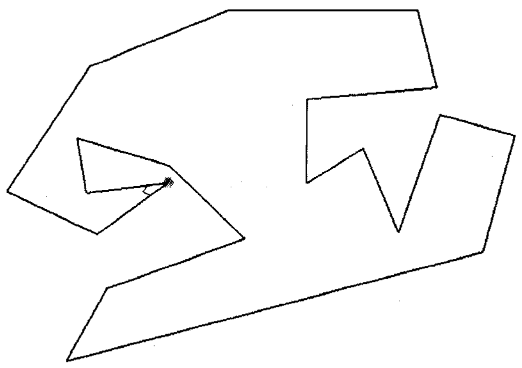


Figure 46. The first element generated for a sharp boundary.

5.4 Conclusions

The non-linear regression model has some advantages and disadvantages in comparison with other algorithms:

- This algorithm is good at dealing with any convex boundary automatically especially the complex one.
- This algorithm can handle the scraggy boundary that including the one with sharp corner at the corner. If training sample has enough kind of data, the algorithm can generate mesh automatically without any special operation.
- This algorithm cannot estimate the accurate value of the output *type* that may cause some error when generating an element.
- This algorithm cannot treat intersection boundary.

Chapter 6
Conclusions and Future Work

The present thesis aims at designing a new model that can generate a two-dimensional quadrilateral mesh from a predefined piece-wised boundary and does not need to find the basic rules of creating an element based on boundary features in a heuristic manner. It is difficult to define the rules for generating a good-quality element based on all boundary information. In this thesis, we introduced a regression model by using methods from Design of Experiments (DOE) and multivariate regression modeling. To test the performance of the proposed method, we implemented a regression model-based mesh generation algorithm.

The present thesis proposed a regression model-based mesh generate algorithm based on multivariate regression model from experiments. The major contributions made in the thesis may be summarized as follows:

- Defined the rules for collecting training sample data from 2D two-dimensional quadrilateral meshes with different number of leading points.
- During the development and research of the data collection algorithm, we implemented a program in order to collect data automatically. The program can print the sample data independently also.
- Found the critical factors when generating an element for a two-dimensional quadrilateral mesh.

- Found the influence of each leading point and derive the optimal leading point which should be taken into account when generating a mesh.
- Proposed the optimal regression models that are used for generating a two-dimensional quadrilateral mesh.
- Used the model derived from experiments in order to implement the regression model-based mesh generation algorithm. As a result, the mesh generation algorithm is much more effective in dealing with boundaries having sharp corners.

Our future research work directions will be focused on other non-linear regression models for mesh generation, especially for the output parameter *type*. This may help solve sharp corners problems. Further, additional research should be performed with more experiments to study the relationship between the leading points and the new point.

Publications

Publications in Refereed Conferences

1. Jie Jin, Guang Qing He, A. Ben Hamza, Yong Zeng (2007), Using DOE Method to Determine the Precision of a 3D Scanner, FAIM2007 Conference

2. Jie Jin, Yong Zeng, A. Ben Hamza, Guangqing He (2008), Regression model based element extraction method for 2-dimensional automatic quadrilateral mesh generation, Expert Systems with Applications (in preparation for submission).

References

[1] M. Saxena, and R. Perucchio, *Element extraction for automatic meshing based on Recursive Spatial Decomposition.* Proceedings of the ASME International Computers in Engineering Conference and Exposition, Anaheim, CA; UNITED STATES;, 1989: p. 151-161.

[2] J.A. Talbert, and A.R. Parkinson, *Development of an Automatic Two-Dimensional Finite Element Mesh Generator Using Quadrilateral Elements and Bezier Curve Boundary Definition.* International Journal for Numerical Methods in Engineering, John Wiley & Sons, 1990. **29**: p. 1551-1567.

[3] T.D. Blacker, M.B. Stephenson, and S. Canann, *Analysis automation with paving: a new quadrilateral meshing technique.* 1991 conference on design productivity, Honolulu, HI (USA), 3-9 Feb 1991, 1990 **13**(5/6): p. 332-337.

[4] T.D. Blacker, and M.B. Stephenson, *Paving: a new approach to automated quadrilateral mesh generation.* Int. J. Num. Meth. Eng., 1991. **32**(4): p. 811-847.

[5] J. Z. Zhu, O. C. Zienkiewicz, E. Hinton, and J. Wu, *A new approach to the development of automatic quadrilateral mesh generation.* Int. J. Num. Meth. Eng., 1991. **32**(4): p. 849 - 866.

[6] B.P. Johnston, J.M. Sullivan, and A. Kwasnik, *Automatic conversion of triangular finite element meshes to quadrilateral elements.* International Journal for Numerical Methods in Engineering, Wiley, 1991. **31**: p. 67-84.

[7] K.L. Lin, and H.J. Shaw, *Two-dimensional orthogonal grid generation techniques.* Computers and Structures., 1991 **41**(4): p. 569-583.

[8] D. Zhang, G. Cheng, and Y. Gu, *Automatic generation of quadrilateral mapping elements and applicability of shape optimization software.* Computers & Structures, 1992. **45**(4): p. 697-705.

[9] A. Allievi, and S.M. Calisal, *Application of Bubnov-Galerkin formulation to orthogonal grid generation.* Journal of Computational Physics, 1992 **98**(1): p. 163-173.

[10] D.E. Barker, *An object-oriented hierarchical paradigm for integrated parametric design and automated two-dimensional quadrilateral mesh generation.* University of Utah, 1993.

[11] J. Hugger, *The theory of density representation of finite element meshes. Examples of density operators with quadrilateral elements in the mapped domain.* Comp. Meth. Appl. Mech. Eng., , 1993. **109**(1/2): p. 17-39.

[12] J. Z. Zhu, E. Hinton, and O.C. Zienkiewicz, *Mesh enrichment against mesh regeneration using quadrilateral elements.* Commun. Num. Meth. Eng., 1993. **9**(7): p. 547-554.

[13] Y. Zeng and G. Cheng, *Knowledge based free mesh generation of quadrilateral elements in two dimensional domains.* International Journal of Microcomputers in Civil Engineering, 1993. **8**(4): p. 259-270.

[14] J.M. Zhou, K.R. Shao, K.D. Zhou and L.R. Li, *A new approach to automatic quadrilateral mesh generation.* Magnetics, IEEE Transactions on, 1993. **29**(2): p. 1910-1914.

[15] M. Bern, D. Eppstein, and J. Gilbert, *Provably good mesh generation.* J. Computer Syst. Sci. Int., 1994. **48**(3): p. 384-409.

[16] C.K. Lee and S.H. Lo, *A New Scheme for the Generation of a Graded Quadrilateral Mesh.* Computers and Structures, Pergammon, 1994. **52**(5): p. 847-857.

[17] B. Joe, *Quadrilateral mesh generation in polygonal regions.* Computer-Aided Design,, 1995. **27**(3): p. 209-222.

[18] W. Min, *A new approach to fully automatic mesh generation.* Journal of Computer Science and Technology, 1995. **10**(6): p. 491-508.

[19] G. Subramanian, *An algorithm for two- and three-dimensional automatic structured mesh generation.* Comp. Struct., 1996. **61**(3): p. 471-477.

[20] L. Kobbelt, *Interpolatory subdivision on open quadrilateral nets with arbitrary topology.* Comp. Graphics Forum, 1996. **15**(3): p. 409-420.

[21] G.D Cheng and H. Li, *New method for graded mesh generation of quadrilateral finite elements.* Comp. Struct., 1996. **59**(5): p. 823-829.

[22] K.T. Miura and F. Cheng, *Mesh generation based on a new label-driven subdivision.* International journal of the Japan Society for Precision Engineering (Int. j. Jpn soc. precis. eng.), 1996. **30**(4): p. 353-358.

[23] F. Neugebauer and R. Diekmann, *Improved mesh generation: not simple but good.* Fifth International Meshing Roundtable, 1996: p. 257-270.

[24] M. Bern and D. Eppstein, *Quadrilateral meshing by circle packing.* 6th Int. Meshing Roundtable, Park City, Utah, 1997: p. 7-20.

[25] M. Halpern, *Industrial requirements and practices in finite element meshing: a survey of trends.* 6th International Meshing Roundtable, Sandia National Laboratories, 1997: p. 399-411.

[26] P. Kinney, *CleanUp: improving quadrilateral finite element meshes.* Proceedings, 6th International Meshing Roundtable, Sandia National Laboratories, 1997: p. 437-447.

[27] D.E. Barkera and S.A. Lantzb, *Automating node and element assignments on conforming four-sided sections defining a domain for mapping quadrilateral elements.* Comp. Struct., 1998. **62**(2): p. 373-380.

[28] H. Borouchaki, F. Hecht and P.J. Frey, *Mesh gradation control.* Int. J. Num. Meth. Eng., 1998. **43**(6): p. 1143-1165.

[29] S.A. Canann, S.N. Muthukrishnan and R.K. Phillips, *Topological improvement procedures for quadrilateral finite element meshes.* Eng. with Comp., 1998. **14**(2): p. 168-177.

[30] S. B. Petersen, J. M. C. Rodrigues and P. A. F. Martins, *Automatic generation of quadrilateral meshes for the finite element analysis of metal forming processes.* Finite Elements in Analysis and Design, 2000. **35**(2): p. 157-168.

[31] M. Berzins, *Mesh quality: a function of geometry, error estimates or both.* Engineering with Computers, 1999. **15**(3): p. 236-247.

[32] J.C. Cuilliere, *Automatic meshing and remeshing applied to model modification.* 3rd Int. Conf. Ind. Autom., Montreal, 1999: p. 12/13-6.

[33] M. Gotoh and J. Zhu, *Automatic remeshing of 2D quadrilateral elements and its application to continuous deformation simulation: part II – applications.* Journal of Materials Processing Technology, 1999. **87**(1): p. 179-191(13).

[34] A. Hocknell, S. Mitchell, D. Underwood and R. Jones, *Feature based quadrilateral mesh generation for sculptured surface products.* IIE Transactions, 1999. **31**(7): p. 627-637.

[35] S. Yoshimura, Y. Wada and G. Yagawa, *Automatic mesh generation of quadrilateral elements using intelligent local approach.* Comp. Meth. Appl. Mech. Eng., 1999. **179**(1/2): p. 125-38.

[36] J. Zhu and M. Gotoh, *Automatic remeshing of 2D quadrilateral elements and its application to continuous deformation simulation: part I – remeshing algorithm.* J. Mater. Process. Technol, 1999. **87**(1/3): p. 165-178.

[37] B. Couteau, *The mesh-matching algorithm: an automatic 3D mesh generator for finite element structures.* Journal of Biomechanics, 2000. **33**(8): p. 1005-1009.

[38] D.A. Field, *Qualitative measures for initial meshes.* 2000. **47**(4): p. 887-906.

[39] V. Francois and J.C. Cuilliere, *3D automatic remeshing applied to model modification.* Computer-Aided Design, 2000. **32**(7): p. 433-444(12).

[40] M.K. Nathan, S. Prasanna and G. Muthuveerappan, *Three-dimensional mesh generation using principles of finite element method.* Advances in Engineering Software 2000. **31**(1): p. 25-34.

[41] A. Çinar and A. Arslan, *Neural Networks Based Mesh Generation Method in 2-D.* Lecture Notes in Computer Science, 2002. **2510**: p. 395-401.

[42] D.G. Triantafyllidis and D.P. Labridis, *A finite-element mesh generator based on growing neural networks.* IEEE Transactions on Neural Networks, 2002. **13**(6).

[43] S. Yao, B. Yan, B. Chen, and Y. Zeng, *An ANN-based element extraction method for automatic mesh generation.* Expert Systems with Applications, 2005. **29**(1): p. 193-206.