

Two-dimensional finite element mesh generation algorithm for electromagnetic field calculation

Chun-Feng Zhang(章春锋), Wei Wang(汪伟), Si-Guang An(安斯光), and Nan-Ying Shentu(申屠南瑛)

Citation: Chin. Phys. B, 2021, 30 (**1**): 010101. DOI: 10.1088/1674-1056/abaedf
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

An approximation for the boundary optimal control problem of a heat equation defined in a variable domain

Yu Xin, Ren Zhi-Gang, Xu Chao
Chin. Phys. B, 2014, 23 (**4**): 040201. DOI: 10.1088/1674-1056/23/4/040201

Second-order two-scale computations for conductive–radiative heat transfer problem in periodic porous materials

Yang Zhi-Qiang, Cui Jun-Zhi, Li Bo-Wen
Chin. Phys. B, 2014, 23 (**3**): 030203. DOI: 10.1088/1674-1056/23/3/030203

A complex variable meshless local Petrov-Galerkin method for transient heat conduction problems

Wang Qi-Fang, Dai Bao-Dong, Li Zhen-Feng
Chin. Phys. B, 2013, 22 (**8**): 080203. DOI: 10.1088/1674-1056/22/8/080203

An improved local radial point interpolation method for transient heat conduction analysis

Wang Feng, Lin Gao, Zheng Bao-Jing, Hu Zhi-Qiang
Chin. Phys. B, 2013, 22 (**6**): 060206. DOI: 10.1088/1674-1056/22/6/060206

The discontinuous Petrov-Galerkin method for one-dimensional compressible Euler equations in the Lagrangian coordinate

Zhao Guo-Zhong, Yu Xi-Jun, Guo Peng-Yun
Chin. Phys. B, 2013, 22 (**5**): 050206. DOI: 10.1088/1674-1056/22/5/050206

Two-dimensional finite element mesh generation algorithm for electromagnetic field calculation*

Chun-Feng Zhang(章春锋), Wei Wang(汪伟), Si-Guang An(安斯光)†, and Nan-Ying Shentu(申屠南瑛)

Department of Mechanical and Electrical Engineering, China Jiliang University, Hangzhou 310016, *China*

(Received 14 July 2020; revised manuscript received 30 July 2020; accepted manuscript online 13 August 2020)

Two-dimensional finite element mesh generation algorithm for electromagnetic field calculation is proposed in this paper to improve the efficiency and accuracy of electromagnetic calculation. An image boundary extraction algorithm is developed to map the image on the geometric domain. Identification algorithm for the location of nodes in polygon area is proposed to determine the state of the node. To promote the average quality of the mesh and the efficiency of mesh generation, a novel force-based mesh smoothing algorithm is proposed. One test case and a typical electromagnetic calculation are used to testify the effectiveness and efficiency of the proposed algorithm. The results demonstrate that the proposed algorithm can produce a high-quality mesh with less iteration.

Keywords: mesh generation, smoothing function, finite element, electromagnetic calculation

PACS: 01.50.hv, 02.70.Dh, 07.55.–w, 96.15.Gh **DOI:** 10.1088/1674-1056/abaedf

1. Introduction

With its favorable ability to adapt the geometric shape of the field boundary and the variation of the physical properties of the medium, the finite element method has become one of the powerful numerical calculation methods for numerical analysis and optimal design of electromagnetic field.[1,2] Generally speaking, the finite element analysis is divided into three stages: modeling, calculation and solution, result processing and evaluation. Their proportions of time spent in each stage are 40%–60%, 5%–10%, 30%–50%, respectively. For complex geometry and fine meshes, it may take too long to generate a qualified mesh. The mesh generation time may account for more than 90% of the total CPU time in the whole numerical simulation process.[3–5] Therefore, how to effectively improve the speed and quality of mesh generation is crucial to the finite element method.

To improve the efficiency and quality of finite element meshing, endeavors have been made mainly in parallel mesh generation, mesh smoothing methods and mesh optimization process. Parallel mesh generation techniques can be classified as discrete domain decomposition (DDD) parallel meshing and continuous domain decomposition (CDD) parallel meshing.[6–9] The DDD parallel meshing generates coarse meshes sequentially for the input boundaries firstly and uses the mesh or graphics division technique to refine the coarse mesh. In the CDD method, more methods such as quadtree/octree, spatial sorting can be used to decompose the domain. Parallel mesh generation can increase the speed to mesh the solving domain based on decomposition. However, most parallel mesh generation methods need lots of memory to generate large-scale meshes; data dependence, load balancing, and the robustness of the method are still needed considering.

Currently, there are three main approaches for mesh smoothing: Laplacian and Winslow, optimization-based method, and physics-based method.[10] In the Laplacian and Winslow method, nodal positions are updated by the arithmetic mean of connected nodes and minimizing specific mesh quality metrics, respectively, to prevent mesh from folding.[11–13] In the optimization-based smoothing method, the nodal positions can be derived by solving local or global optimization problems with different types of objective functions.[14] In the physics-based method, the mesh is considered as physics system to improve the shape of the element.[15,16] The mesh smoothing method can improve both the average quality of the mesh and the efficiency the mesh generation. However, iterations are often involved in the mesh smoothing which is time-consuming. A fast mesh smoothing method is a persisting pursuit for the mesh generation algorithm.

Techniques and schemes are also proposed in the mesh optimization process to promote the performance of the mesh generation. A mesh quality metric alternation scheme is used in the mesh optimization process to obtain the high-quality mesh.[17,18] Some literature proposes a novel automated high-order unstructured triangular mesh generation method for the two-dimensional domain.[19,20] A mesh-generation framework for image representation based on data-dependent triangulation was proposed.[21] These methods have improved the performance of mesh generation to some extent.

In order to improve the convergence speed of mesh iteration and ensure the quality of mesh generation in finite el-

*Project supported by the National Natural Science Foundation of China (Grant Nos. 52077203 and 61701467) and the Natural Science Foundation of Zhejiang Province, China (Grant No. LY19E070003).

†Corresponding author. E-mail: annsg@126.com

© 2021 Chinese Physical Society and IOP Publishing Ltd

ement analysis to a forward step, several improvements are made based on the Persson's algorithm. An image boundary extraction algorithm based on path optimization is introduced to avoid the interaction between the computer-aided design (CAD) and the finite element analysis software, which provides an alternative method to build the geometric boundary for the solving domain. Based on Persson's algorithm, identification algorithm for the location of nodes in polygon area is proposed to determine the state of the boundary node. Finally, a novel force-based mesh smoothing algorithm is proposed to improve the mesh quality and convergence speed of the mesh generation method.

2. Persson's algorithm

In the Persson's algorithm, the signed distance functions is used to specify and describe the geometric regions to be meshed.[15] The signed distance functions specify the shortest distance from any point in the space to the boundary of the domain, where the sign of the function is positive outside the region, negative inside, and zero on the boundary. This definition is used to identify whether the point is inside or outside the geometric domain. The geometric field Ω can be implicitly defined as

$$\Omega = \left\{ x = (x,y) \in R^2 | d(x) < 0 \right\}. \tag{1}$$

The Persson's algorithm is based on the analogy between triangular meshes and two-dimensional truss structures. The edges of the triangle correspond to the bars, and the vertexes correspond to the nodes of the truss. Each of the bars in the truss structure has a force function $f(l,\ l_0)$.

$$f(l,l_0) = \begin{cases} \alpha(l_0 - l), & \text{if } l < l_0, \\ 0, & \text{if } l > l_0, \end{cases} \tag{2}$$

Persson's algorithm determines that the relationship depends on length l and unextended length l_0. At every boundary node, there is a reaction force acting normally on the boundary. This force is just large enough to prevent the node from moving outward. α is usually set to be 1, it is reasonable to require $f = 0$ for $l = l_0$. The positions of the joints are found by solving for a static force equilibrium in the structure.

The equilibrium problem corresponds to a nonlinear equation: $F(p) = 0$, where $F(p)$ is an N-by-2 array in which the first column contains x-components of the forces, and the second column contains the y-components of the forces. A simple way to solve the equation is to introduce an artificial time-dependent solution. For some $p(0) = p_0$, the ODES system in non-physical units is considered as follows:

$$\frac{\mathrm{d}p}{\mathrm{d}t} = F(p(t)),\ \ t \geq 0. \tag{3}$$

If a constant solution is found, the system $F(p) = 0$ is satisfied. The system uses forward Euler method for approximation. In the case of discrete time $t_k = k\Delta t$, the approximate solution $p^k \approx p(t_k)$ is updated as follows:

$$p^{k+1} = p^k + \Delta t F(p^k). \tag{4}$$

Persson's algorithm is simple and compatible. For some regions with geometric and mathematical background, it is relatively easy to create signed distance functions; however, for other regions, distance functions cannot be expressed quickly and accurately, and some are even difficult to express. For mesh smoothing, Persson's algorithm tries to optimize the node locations by a force-based smoothing procedure. This method usually generates high-quality mesh elements, but sometimes extremely low-quality elements are obtained, which will seriously affect the accuracy of the finite element calculation; and due to the fact that the force balancing method used in this algorithm has only a repulsive force, it is inevitable that frequent iterations will occur, which may lead low efficient mesh to be generated.

3. The proposed algorithm

Due to the lack of Persson's algorithm in the definition of complex polygons, and room for improvement in the efficiency and quality of mesh generation, some improvements are developed to speed up the convergence and also guarantee the quality of the output mesh.

3.1. Image boundary extraction algorithm based on path optimization

The solving domain of finite element analysis is built generally by the CAD or the modeling operation in finite element analysis software. Modern CAD systems tend to integrate finite element analysis tools into the design environment, but their boundary representation (B-Rep) usually contains a large number of faces due to the presence of certain surface elements that may be very narrow or smaller than the mesh size, this will result in poor element quality or incorrect local representation. The Persson's algorithm is easier to solve the boundary with a general geometric background, but it is difficult to effectively and accurately determine the complex and irregular boundary regions. Therefore, an image mapping strategy is used to provide an alternative way to determine the solving domain in this paper. In order to avoid the boundary unsmoothness caused by pixel extraction and ensure the accuracy of image boundary detection, an image boundary extraction algorithm based on path optimization is proposed to obtain the image boundary information. The process of the proposed algorithm is as follows: firstly, the image is grayed and the gray image adaptive multi-threshold segmentation is performed; secondly, the polygon boundary is extracted according to the gray value; finally, the polygon contour is optimized by path optimization algorithm. The horizontal coordinate x and vertical coordinate y of a point $p(x,y)$ are recalculated from the following equations:

$$\begin{cases} x = \dfrac{1}{2}(1-t)^2 p(i,1) + \dfrac{1}{2}\left(-2t^2+2t\right) p(i+1,1) + \dfrac{1}{2} p(i+2,1), \\ y = \dfrac{1}{2}(1-t)^2 p(i,1) + \dfrac{1}{2}\left(-2t^2+2t\right) p(i+1,1) + \dfrac{1}{2} p(i+2,1), \end{cases} \tag{5}$$

where i is the number of the points, t is a proportional value in a range between 0 and 1, and p is an N-by-2 array of node coordinates.

For the boundary of the geometric domain obtained by the above method, it is necessary to determine the internal boundary when generating the finite element mesh, which fully meets the needs of solving the differential equation with discontinuous material coefficient. The internal boundary divides the whole finite element solution domain into several subdomains, which are connected only by common nodes on the common edge. Generally speaking, a bottom-up method is often used, that is, before the whole domain is meshed, the boundaries are meshed separately, and the positions of the generated nodes and boundary elements are fixed. This method relies heavily on artificial factor, and there is room for further improvement in intelligence and efficiency. In this paper, the implicit distance function is used to represent the internal boundary, and this function is used to project the internal boundary points. The method is slightly different from the previous one, since there are nodes on both sides of the internal common edge, and in this paper the fixed points of the internal boundary are presupposed. In the subsequent iterative optimization process, these fixed nodes are tracked in the direction perpendicular to the inner boundary edge, and the tolerance deps $= 0.5h_0$ (h_0 denotes the initial mesh size) is given. For the newly generated nodes that meet the deps criteria, if they try to cross the boundary, they will be added to the list of boundary nodes and projected on the boundary. The expression is indicated as follows:

$$\begin{cases} \Delta = \dfrac{\text{grad}(p,p_i)}{\text{deps}} \left\| fd(p) \right\|, \\ p(x_0,y_0) = p(x-\Delta, y-\Delta), \end{cases} \tag{6}$$

where p is the newly generated node which meets the deps criterion, p_i is the fixed point of the internal boundary edge, fd is the distance function, and $p(x_0,y_0)$ is the internal boundary node after projection processing. It is important to note that additional quality control is required to avoid overlapping mesh elements.

3.2. Identification algorithm for the location of nodes in polygon area

By introducing fixed points and boundary checking algorithm based on crossing number, Persson's algorithm determines that the points are located in the polygon region, and the points are inside, outside or on the boundary. A simple implementation requires checking each boundary intersection of each point, resulting in O($N \times M$) complexity. In order to improve the efficiency of node position state identification, in this paper we propose to estimate the number of segments of the data obtained from the image to the boundary, and segment according to the endpoint of the segmented line chart. Then, starting from the first vertex that intersects the x range of the boundary, the location can be found by the search algorithm, and the piecewise distance function is abstracted based on the coordinates of the head and tail nodes of the boundary, so that the position can be identified and the search process can be terminated when the x coordinate of vertex is beyond the x range of this segment. In the search algorithm, the selection of keynodes as the basis of division is crucial to the efficiency of the entire search algorithm. In this paper, the formula for selecting the keynodes of the record in the middle is updated as follows:

$$\text{mid} = a[i]\,\text{low} + \frac{\text{key} - a[i]\,\text{low}}{a[i]\,\text{high} - \text{key}} \left(a[i]\,\text{high} - a[i]\,\text{low}\right), \tag{7}$$

where $a[i]$ can be a segment obtained according to image processing, i represents the number of segments, 'low' and 'high' denote the head and tail data of each segment, and 'key' is the node that needs to identify the position. For the look-up table with large table length and uniform keynodes distribution, the average performance of the data search algorithm is better.

3.3. Novel force-based mesh smoothing algorithm

Appropriate mesh smoothing algorithms can optimize the quality of generated meshes and improve the efficiency of mesh generation. The Persson's algorithm only describes the repulsive force to help distribute the nodes to the entire Ω domain, but it is inevitable to be iterated frequently. In this paper, a novel force-based mesh smoothing algorithm is proposed, which regards the mesh as a spring-like deformable medium. The whole process can be divided into two stages: measurement of the quality of each element and the average quality of the overall mesh; mesh smoothing operation based on mesh quality.

For the first stage, the mesh quality calculation method is introduced. In the finite element applications, the quality measure of the triangular element mesh can be regarded as the quality measure of the geometric element, which can be calculated in the following way: the upper limit of error depends on the minimum angle in the mesh. To quantify mesh quality, the commonly used quality measure is the relationship between the maximum inscribed circle radius and the minimum circumscribed circle radius, that is,

$$q = \frac{(b+c-a)(c+a-b)(a+b-c)}{abc}, \tag{8}$$

where a, b, and c are the side lengths. According to the rule of thumb, if the $q_{\min}$ values of all elements are greater than 0.5, the mesh is favorable.

It is worth noting that in Persson's algorithm, the force balance state is achieved by fine-tuning the position of mesh vertex. After a certain number of mesh iterations, when the fine-tuning moving distances of all internal nodes are less than the set convergence accuracy, the algorithm must be terminated. Since the quality of the mesh in the mesh generation may already meet the finite element calculation requirements, using the convergence accuracy as the termination condition in the original Persson's algorithm may cause additional iterations. A threshold value of mesh quality is set as the termination criterion of the proposed mesh generation algorithm. When the quality values of all meshes are greater than 0.5, the mesh generation algorithm will be terminated, which will effectively avoid excessive mesh iteration.

For the second stage, in this paper, each edge is set to have an $l_{ij} = [p_i, p_j]$ variable spring constant, $f(p_i, p_j)$ depending on its current length $||l_{ij}||$ and the desired length e_{ij}. In this paper, we adopt the normalized length concept, and define it on an edge by

$$s_{ij} = \frac{\|l_{ij}\|}{e_{ij}}. \tag{9}$$

Then a triangular element mesh with normalized edge length of 1 is created, and the bar force function $f(l, l_0)$ in Persson's algorithm is changed into

$$f(s_{ij}) = \begin{cases} 1 - s_{ij}, & \text{if } s_{ij} < 1, \\ 0, & \text{if } s_{ij} \geq 1. \end{cases} \tag{10}$$

This function only has a repulsive force in mesh smoothing. If the distance between two vertexes is too close, they will repel each other. For the situation that the distance between vertexes may be far away, the function does not consider it, which may cause unnecessary iteration. In order to avoid extra iterations, in this paper we consider giving each bar an attractive force or a repulsive force when performing mesh smoothing iteration.

According to $f(s_{ij})$, with $s = 1$ as the boundary, when $s \in (0, 1)$, the value of the constructed function is greater than 0 and less than 1; when $s = 1$, the value of the constructed function is equal to 0, which conforms to the principle of force-based smoothing function, in order to satisfy the smoothness of the following constructed function and be close to $f(s_{ij})$, the preliminary construction is $f(s) = (1 - s^2)$; when $s > 1$, the value of the constructed function is less than 0, and approaches to 0 before $s = 2$, which is more in line with the size requirements of the generated mesh.

Based on the above requirements, in this article, we combine $f(s_{ij})$ and use the exponential function as the basis for description, and then we use $f(s_{ij})$ function curve as the boundary to spread points appropriately, and conduct the curve fitting experiments In order to try to construct several eligible smoothing functions as shown in Fig. 1.

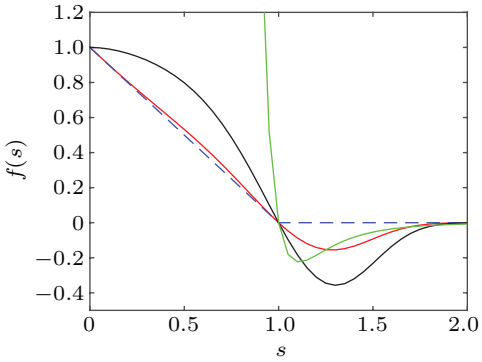


Fig. 1. Several curves of smoothing function $f(s)$, with blue line denoting the results from Persson's algorithm, red line representing $(1-s)\exp(s^3 - s^4)$, black line referring to $(1-s^2)\exp(s^3 - s^4)$, and green line indicating $s^{-13} - s^{-7}$.

The blue dotted line in Fig. 1 represents the smoothing function adopted by Persson's algorithm, which only has a repulsive force; the green line is $f(s) = s^{-13} - s^{-7}$, for mesh generation purpose, this model suffers numerical instabilities, since $f(0^+) \to \infty$. Performing the iterative tests on the smoothing function represented by the red and black lines, in this article we find that a smoothing function $f(s) = (1-s^2)\exp(s^3 - s^4)$ with this attraction/repulsion behavior can produce better results. If two nodes are too close to each other, they repel, and if too distant, they attract. This is different from Persson's smoothing where nodes always repel each other regardless of distance.

Then, a novel smoothing function is constructed based on $f(s) = (1-s^2)\exp(s^3 - s^4)$. The novel smoothing function (NSE) Eq. (11) constructed in this paper has both repulsion and attraction, in addition, the smoothing function can accurately constrain the magnitude of the force in the smoothing process through quality monitor. After experimental verification, the function is numerically stable, this will allow the smoothing process to proceed steadily. Specifically, the NSE is expressed as

$$f(s_{ij}) = \frac{1-q}{1-q_{\text{avg}}}\left(1 - s_{ij}^2\right)\exp\left(s_{ij}^3 - s_{ij}^4\right), \tag{11}$$

where q is the element quality of the finite element, q_{avg} is the average of the overall mesh, $s_{ij} = ||l_{ij}||/e_{ij}$, with $||l_{ij}||$ being the current length, and e_{ij} the desired length.

In this paper, we adopt the first order motion Eq. (3), with $p(0) = p_0$ for the novel smoothing algorithm. Equation (3) is approximated by using the forward Euler method. At the discretized (artificial) time $t_k = k\Delta t$, and p_i^k is set to be the position of node i at the k-th time step, the approximate solution $p^k \approx p(t_k)$ is updated by

$$p_i^{k+1} = p_i^k + \Delta t \sum_{j \in N_i} f\left(p_i^k, p_j^k\right)\frac{l_{ij}^k}{s_{ij}^k}, \quad i = 1, \ldots, N. \tag{12}$$

In order to reach a static state, the algorithm in this paper will be iterated through Eq. (12) to update the node position. In Eq. (12), the bar force is applied to a vector of length e_{ij}, e_{ij} being the desired edge length, and N_i is the neighborhood of node i.

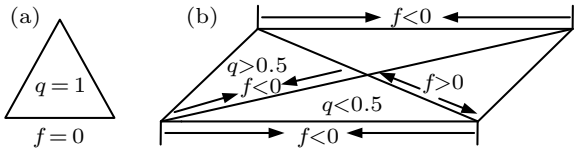


Fig. 2. (a) Ideal triangle shape, (b) schematic diagram of smoothing function.

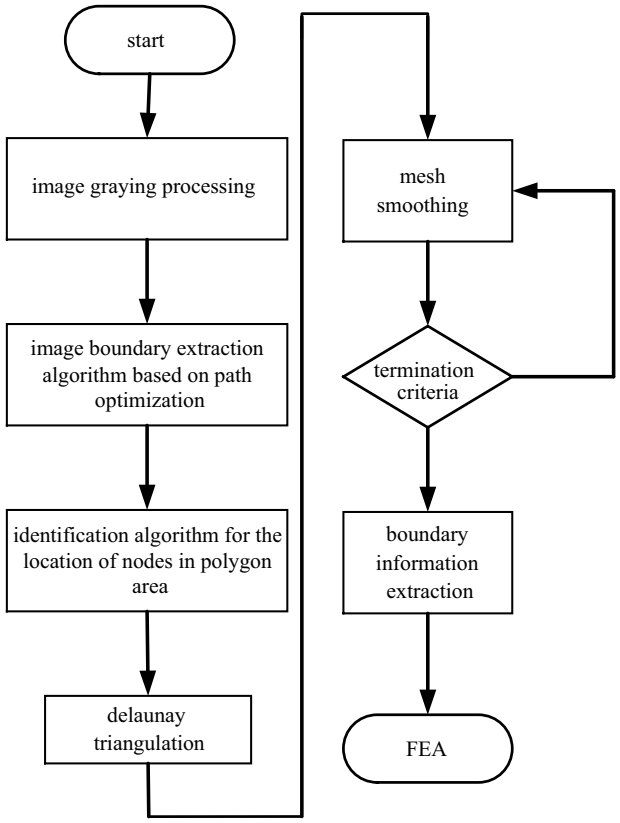


Fig. 3. Algorithm flow chart.

As shown in Fig. 2(b), mesh smoothing is performed. When the distance between two vertexes in a triangle element of the mesh is too close, $s < 1$, the NSE will exert a repulsive force on the bar formed by connecting the two vertexes; when the distance between the two vertexes is far, $s > 1$, the NSE will give bar attractive force. Since the NSE can generate an attractive force or a repulsive force according to the value of s, when it is used for the entire mesh, the repulsive force or attractive force on the edges of adjacent meshes will produce mutual assistance and can reduce the adjacent mesh iteration time, which will reduce the overall mesh iteration time. And the applied force is limited based on the mesh quality, which will avoid generating the extremely poor quality meshes.

The process of mesh smoothing proposed in this paper is as follows. After Delaunay mesh generation is ended, the mesh quality is measured; then according to the quality of the mesh, each element performs a corresponding mesh smoothing. For each element, a set of nodal forces preserves the global topology in order to transform the element itself into the best element, and the displacement constraint at the boundary of the domain should be selected correctly.

4. Numerical examples and application

The following tests are for verifying the effectiveness of the proposed algorithm for improving the mesh generation algorithm.

Experimental conditions: computer configuration is as follows:

Intel (R) Core (TM) i5-7500 CPU @3.40 GHZ

4.1. Alpha Lake of Marcus Garvey

The more complex Alpha Lake of Marcus Garvey is taken as the test case. This area is of a polygon with one polygonal hole. Due to the limitations of the conditions, the image of Alpha Lake is taken from the image provided in the Distmesh. In order to facilitate the gray processing, the surface of Alpha Lake is filled with black as shown in Fig. 4(a). Without being processed by the path optimization algorithm, the image extraction is shown in Fig. 4(b), the fit between the boundary of the image and the boundary of the geometric domain is obviously not high; after the path optimization algorithm is used, it can be seen from Fig. 4(c) that the image boundary extraction algorithm based on path optimization can improve the fit.

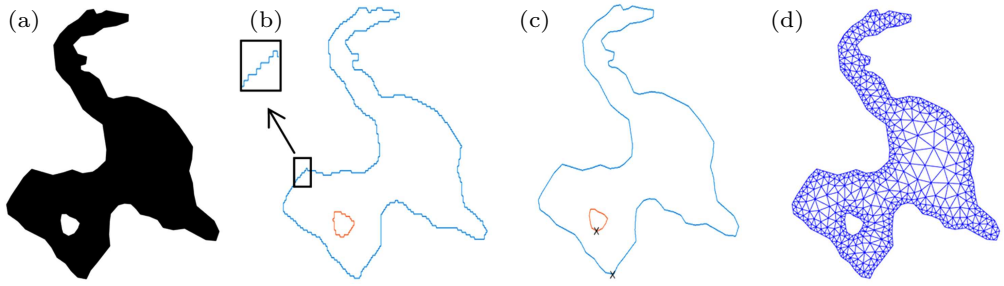


Fig. 4. Alpha Lake ($h_0 = 10$). (a) Gray image. (b) Geometric contour map based on boundary extraction algorithm. (c) Geometric contour map processed by path optimization algorithm. (d) Effect diagram of mesh generation.

To test the influence of identification algorithm for locating the nodes in polygon area and the NSE smoothing function on Persson's algorithm, for the visualization requirements of Alpha Lake meshing, take $h_0 = 10$, 9, 8, 7, where h_0 is the initial defined mesh size.

As shown in Table 1, the number of iterations required to optimize the mesh to reach a steady state using the NSE smoothing function is about 50% of the number of iterations required for Persson's algorithm, and the overall mesh generation speed is increased by 40%.

Table 1. CPU time (in unit s) (composite geometry).

h_0	10	9	8	7
Persson's algorithm	368	465	517	673
Iter/time (s)	13.91	17.29	28.00	44.32
Proposed algorithm	192	235	283	341
Iter/time (s)	7.39	10.13	16.59	21.68

The average mesh quality and the minimum mesh quality generated by the Persson's algorithm are compared with those generated by the proposed algorithm ($h_0 = 10$), and the results are shown in Tables 2 and 3, respectively.

As shown in Table 2, the average quality of the mesh generated by the proposed algorithm is roughly the same as that by the Persson's algorithm, all being above 0.8. It can be seen from Table 3 that the mesh quality generated by the Persson's algorithm is usually high, but the mesh quality will sometimes be low because of the instability of the algorithm. The proposed algorithm performs well in this respect and ensures that the $q_{\min}$ is more than 0.5.

Table 2. Average quality of generated mesh.

Algorithm	L1	L2	L3	L4
Persson's algorithm	0.91	0.89	0.90	0.93
Proposed algorithm	0.87	0.84	0.87	0.85

Table 3. Minimum quality of mesh generation.

Algorithm	L1	L2	L3	L4
Persson's algorithm	0.280	0.279	0.612	0.253
Proposed algorithm	0.504	0.501	0.513	0.507

4.2. TEAM workshop problem 30A

The proposed algorithm is then applied to the numerical solution of the TEAM workshop problem 30A. The TEAM problem 30A is to simulate a three-phase induction motor. The physical parameters are given in Fig. 5(a).[22] The winding in the stator is excited at a frequency $f = 60$ Hz. The rotor is made of steel (with electric conductivity $\sigma = 1.6 \times 10^6$ S/m) and aluminum (with electric conductivity $\sigma = 3.72 \times 10^7$ S/m), with the rotor steel surrounded by the rotor aluminum. The stator steel is laminated and its conductivity is set to be $\sigma = 0$.

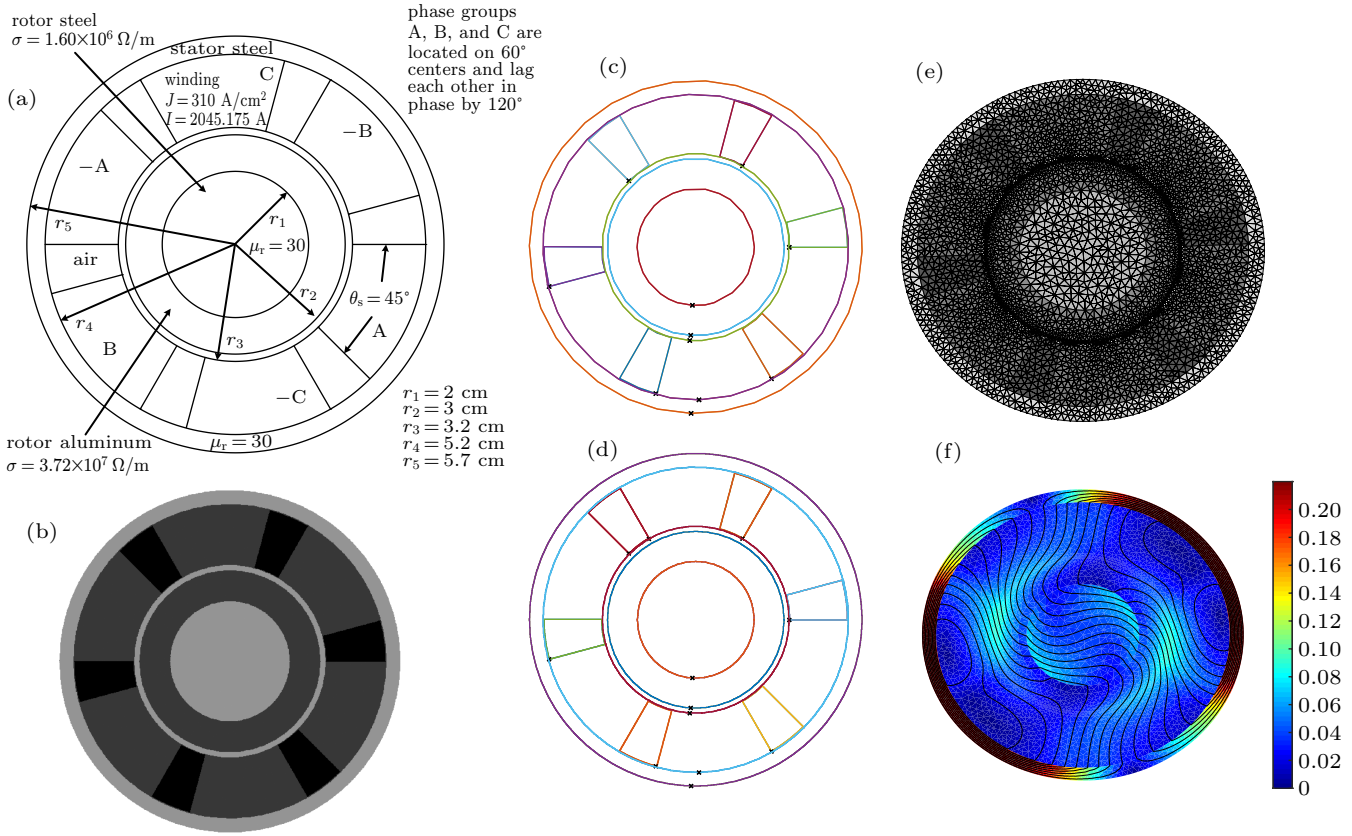


Fig. 5. (a) Setting of TEAM workshop problem 30A. (b) Gray image. (c) Geometric contour map based on boundary extraction algorithm. (d) Geometric contour map processed by path optimization algorithm. (e) Mesh for TEAM workshop problem 30A. (f) Flux density and flux lines at 600 rad/s.

Fill the Fig. 5(a) with equal difference RGB values. On the basis of gray processing, the image adaptive Multi-threshold segmentation (Otsu) is used for gray segmentation, and then the image geometric edge is extracted by image edge detection technology. It can be seen from Fig. 5(c) that the extracted geometric boundary is not smooth, which may have a limited influence on the accuracy of finite element analysis. Through the boundary smoothing algorithm based on the path optimization proposed in this paper, the extracted geometric edge is smoothed and optimized as shown in Fig. 5(d). Obviously, the smoothness of geometric boundary is improved, which helps to improve the accuracy of mesh generation. Finally, using the mesh generation algorithm proposed in this paper, the mesh generation graph is shown in Fig. 5(e). It can be seen from the figure that the performance of the inner boundary processing method adopted in this paper is outstanding, and the inner boundary is in a smooth state.

It can be seen from Table 4 that after the introduction of the NSE smoothing function, the number of iterations of the algorithm is about 50% of the Persson's algorithm's, and the running speed is increased by 50%. Comparing with the common meshing algorithms Gmsh, the speed is about 20% higher than that of the Gmsh. The mesh quality obtained by the proposed algorithm is stable. Comparing with the original algorithm, the mesh quality has not decreased significantly.

Table 4. Results obtained by different algorithms.

Algorithm	Time (s)	Number of iterations	$q_{\min}$	q_{aver}
Persson's algorithm	33.17	1573	0.308	0.965
Gmsh	21.03	1072	0.516	0.842
Proposed algorithm	16.34	736	0.641	0.927

Then the finite element boundary information obtained by the proposed algorithm is applied to the numerical solution of TEAM workshop problem 30A. Let N_{A} represent the turns' density for the phase winding in Fig. 5(a), where the current in the phase winding is indicated. Using a similar nomenclature for phase windings and B and C windings, Fourier space decomposition can write the current density of three-phase winding as

$$\begin{aligned} J(\theta,t) = & \sum_{m=1}^{\infty} \frac{4}{m\pi} \sin\left(\frac{m\theta_{\text{s}}}{2}\right) \Bigg\{ I_{\text{A}}(t) N_{\text{A}} \cos(m\theta) \\ & + I_{\text{B}}(t) N_{\text{B}} \cos m\left(\theta - \frac{2\pi}{3}\right) \\ & + I_{\text{c}}(t) N_{\text{c}} \cos m\left(\theta - \frac{4\pi}{3}\right) \Bigg\}. \end{aligned} \tag{13}$$

Assume that the three phases have the traditional time harmonic distribution, each phase with the same current density IN, then we will have

$$\begin{cases} I_{\text{A}} N_{\text{A}} = IN \cos(\omega t), \\ I_{\text{B}} N_{\text{B}} = IN \cos\left(\omega t - \dfrac{2\pi}{3}\right), \\ I_{\text{A}} N_{\text{A}} = IN \cos\left(\omega t - \dfrac{4\pi}{3}\right). \end{cases} \tag{14}$$

Using Euler's rule to represent the cosinusoidal dependence, equation (13) can be written as

$$J(\theta,t) = \text{Re} \sum_{m=1}^{\infty} \left(\tilde{J}_{+} \, \text{e}^{\text{j}(\omega t - m\theta)} + \tilde{J}_{-} \, \text{e}^{\text{j}(\omega t + m\theta)} \right), \tag{15}$$

where

$$\begin{aligned} \tilde{J}_{\pm} = & \, IN \frac{2}{m\pi} \sin\left(\frac{m\theta_{\text{s}}}{2}\right) \left[1 + \exp\left(-\text{j} \frac{2\pi}{3} (1 \mp m) \right) \right. \\ & \left. + \exp\left(-\text{j} \frac{4\pi}{3} (1 \mp m) \right) \right]. \end{aligned} \tag{16}$$

In this form, it is clear that the three-phase winding yields two counter-rotating waves. In a single-phase machine, equation (15) may take the following form:

$$J(\theta,t) = \text{Re} \sum_{m=1}^{\infty} \left(\tilde{J}_{+} \, \text{e}^{\text{j}(\omega t - m\theta)} + \tilde{J}_{-} \, \text{e}^{\text{j}(\omega t + m\theta)} \right), \tag{17}$$

where

$$J_{\pm} = IN \frac{2}{m\pi} \sin\left(\frac{m\theta_{\text{s}}}{2}\right). \tag{18}$$

The problem of Fig. 5(a) consists of multiple piecewise homogeneous regions. The solution can be developed in each region by using the transfer relation concept given by Melcher.[23,24] In nonconducting regions, the magnetic vector potential is assumed to have a coulomb gauge dependence and satisfies the Poisson's equation

$$\nabla^2 A = -\mu J. \tag{19}$$

Then, by using the finite element method in the time domain to solve the following time-domain eddy current equation represented by Euler's formula and homogeneous Dirichlet boundary conditions, the magnetic field in TEAM workshop problem 30A three-phase induction motor can be simulated. The flux density and flux line of 600 rad/s in Fig. 5(f) can be obtained by solving the following equation:

$$\sigma \frac{\partial A}{\partial t} + \sigma \boldsymbol{v} \cdot \nabla A - \frac{\partial}{\partial x}\left(v \frac{\partial A}{\partial x} \right) - \frac{\partial}{\partial y}\left(v \frac{\partial}{\partial y} \right) = J. \tag{20}$$

Taking the center of motor as the origin of coordinate system, the unit coordinate length is 1m. The following four points are selected to verify the flux density: P1: (0.0151, 0.0131), P2: (−0.0283, 0.0131), P3: (−0.0534, −0.0122), and P4: (0.0510, −0.0214), which are shown in Table 5.

Table 5. Influences of different algorithms on solution of potential.

Methods position	P1	P2	P3	P4	Time (s)
Commercial software	0.0720	0.0863	0.2890	0.1909	23.75
Gmsh	0.0725	0.0867	0.2839	0.1974	23.14
Persson's algorithm	0.0703	0.0825	0.2824	0.1872	35.01
Proposed algorithm	0.0727	0.0864	0.2895	0.1932	18.72

By obtaining random points from the results of commercial finite element software and combining the numerical values solved by the proposed algorithm, a clear comparison can be made to prove the correctness of the algorithm calculation results. Comparing the times taken by the four finite element solution methods, it can be seen from Table 5 that the finite element processing based on this algorithm is about 20% more efficient than commercial software. Comparing with the finite element processing based on Persson's algorithm, the time is reduced by 50%; comparing with Gmsh, the speed is increased by 20%.

5. Conclusions

Two-dimensional unstructured mesh generation based on Persson's algorithm is proposed for calculating the electromagnetic field in this paper. The numerical results demonstrate that compared with the original Persson's algorithm and commercial software, the algorithm can generate high-quality meshes in the solving domain, the iteration time is reduced by about 50%, and comparable solutions can be obtained in electromagnetic calculations. As the structures of electromagnetic devices become more complex, electromagnetic models are built in three dimensions to obtain a precise solution. In the future work the authors will focus on improving the efficiency of mesh generation in three-dimensional problems.

References

[1] Hao K S, Huang S L, Zhao W and Wang S 2011 *Chin. Phys. B* **20** 068104
[2] He X B, Wei B, Fan K H, Li Y W and Wei X L 2019 *Chin. Phys. B* **28** 074102
[3] Zhao Y P, Xiao X Y and Xu W Y 2018 *IEEE Trans. Ind. Electron.* **65** 98259833
[4] Yang H H, Gunzburger M and Ju L L 2018 *J. Comput. Phys.* **367** 235252
[5] Chen L C and Fan W H 2012 *Chin. Phys. B* **21** 104101
[6] Chen J J, Xiao Z F, Zheng Y, Zou J F, Zhao D W and Yao Y F 2018 *Adv. Eng. Softw.* **121** 131146
[7] Ito Y, Shih A M, Erukala A K, Soni B K, Chernikov A, Chrisochoides N P and Nakahashi K 2006 *Math. Comput. Simul.* **75** 200209
[8] Yu W Y and Li X 2015 *Procedia Engineering* **124** 4456
[9] Canann S A, Tristano J R and Staten M L 1998 *IMR* p. 479
[10] Liu X Y and Fu W N 2018 *IEEE Trans. Magn.* **54** 2857850
[11] Niu S X, Zhao Y P, Ho S L and Fu W N 2011 *IEEE Trans. Magn.* **47** 29432946
[12] Fortunato M and Persson P O 2016 *J. Comput. Phys.* **307** 114
[13] Jiao X, Wang D and Zha H 2011 *Eng. Comput.* **27** 8194
[14] Durand R, Pantoja-Rosero B G and Oliveira V 2019 *Finite Elements in Analysis & Design* **158** 1730
[15] Persson P and Strang G 2004 *Siam Review* **46** 329345
[16] Park J and Shontz S M 2011 *Procedia Computer Science* **4** 292301
[17] Roberts K J, Pringle W J and Westerink J J 2019 *Geosci. Model Dev.* **12** 18471868
[18] Smitha T V and Nagaraja K V 2018 *Adv. Eng. Softw.* **115** 327356
[19] Rathod H T, Nagaraja K V, Naidu V K and Venkatesudu B 2008 *Finite Elements in Analysis & Design* **44** 920932
[20] Noguchi S, Nobuyama F and Igarashi H 2014 *IEEE Trans. Magn.* **50** 453456
[21] Li P and Adams M D 2013 *IEEE Trans. Image Process.* **22** 20042018
[22] Davey K R *Induction Motor Analysis-International TEAM Workshop Problem* 30
[23] Huang W, Ren Y and Russell R D 1994 *SIAM J. Numer. Anal.* **31** 709730
[24] Zhou Y, Bao X H, Ma M N and Wang C Y 2018 *Chin. Phys. B* **27** 088401