Heat and Mass Transfer 35 (1999) 361–370 © Springer-Verlag 1999

Automatic generation of unstructured grids with Delaunay triangulation and its application

B. Yu, M. J. Lin, W. Q. Tao

Abstract This paper is consisted of two parts. In the first part, a method is described which generates two-dimensional triangle mesh using the Delaunay triangulation criterion. An automatic algorithm was proposed which combines several advantages of the existing methods. Local mesh refinement can also be easily performed with this method. Examples of generated grids were presented for several convex, non-convex and multi-connected domains to demonstrate the effectiveness and feasibility of the proposed method. In the second part, the turbulent heat transfer in an annular space finned by wave-like longitudinal fins was numerical simulated. The proposed technique was adopted to generate the grid in the cross-section. The standard K-ε model in conjuction with wall function method was used to simulate the fluid flow and heat transfer in the complex geometry. The discretization of the governing equations was described. The computational results were compared with the authors' test data and the agreement was reasonably good.

1 Introduction

For handling complex geometries in computational heat transfer and fluid flow, a structured single block grid is often not desirable, and quite often even impossible. Many different approaches have been proposed to get around this obstacle. Of those two major categories are the multiblock grids, which utilize structured grids within blocks, and the unstructured grids, which consist of an assembly of triangles (2D) or tetrahedra (3D). Although the computer time required for the unstructured-grid solver is usually much larger than that of structured grid, the step of appropriate connections between blocks for the multiblock is much time-consuming and needs more man effort. Recent advances show that computation of fluid flow and heat transfer on unstructured grids for complex geometries are more competitive with those of structured grids.

There are two dominant methods of generating unstructured grids. The first one is the advancing front method [1–3]. In this method, giving initial boundaries of points and triangles, points are created within the field and triangles formed, which advance into the computational domain. The other method, which is addressed in this paper, is the Delaunay triangulation [4–8]. This technique triangulates a set of points in a unique way such that the minimum angle of each triangle formed in the grid system is maximized. This technique has the advantage that the resulting triangles are optimal for the given set of points in that they usually do not contain many extremely skewed cells, which is a very desirable property from the viewpoint of computational accuracy. Moreover, such techniques based on the Delaunay method are particularly suited to a self-adaptive solution strategy, because the Delaunay construction can conveniently add new points to an existing triangulation with no need to remesh the whole domain. Therefore, this technique has been calling for more and more attention in computational fluid dynamics and heat transfer.

Although the Delaunay triangulation provides a unique way with which to connect points, it does not provide the method of generating a point within a computational domain. Furthermore, the resulting meshes may not boundary-conforming, i.e. the boundaries of resulting triangles may not coincide with the boundary of a computational domain if no special method is applied during the triangulation. Therefore, the emphasis of recent investigations on the Delaunay triangulation are mainly focused on these two subjects [6–9]. A simple and convenient way to make the mesh boundary conforming is to start the triangulation process with a simple square which contains the boundary points of the computational domain [6]. The square is divided into two triangles and its four corner points are located at sufficient distance from all the given boundary points. Using the Delaunay triangulation technique (which will be briefly addressed later) to form triangles, a very coarse mesh may be obtained with triangle vertices only at the given boundary points and the four corners. Then the cells whose centroids are outside the given domain are removed, resulting in an initial triangulation for the given boundary points. The next step is to create points within the domain. This method for boundary conforming will be

Received on 9 July 1998

B. Yu
M. J. Lin
W. Q. Tao
School of Energy & Power Engineering
Xi'an Jiaotong University
Xi'an, Shaanxi 710049, PRC

Correspondence to: W. Q. Tao

This work was supported by the National Natural Science Foundation of China (No. 59676019). Discussion with Dr. Shang, H. M. from Engineering Science Inc., Huntsvill, AL, U.S.A is also grateful.

adopted in this paper. For the point creation many methods have been proposed, of which the methods proposed recently by Anderson [6] and Weatherill and Hassan [8] will be briefly reviewed. Anderson defined a parameter which he called aspect ratio as the ratio of the circumcircle radius to twice the incircle radius of any existing triangle. It is required that this aspect ratio should be less than a predetermined tolerance which is generally 1.5. A point is first added at the circumcircle of the triangle whose aspect ratio is the largest among the existing triangles. Then using the Delaunay triangulation technique to connect this point and the related points, forming a series of new triangles. This process is performed until the aspect ratios of all the triangles in the domain are less than the tolerance. Since for the equilateral triangle its aspect ratio is equal to 1, the use of the aspect ratio can effectively control the shape of the resulting triangles. However, the information contained in the boundary points which usually reflects a desire variation slope may not be effectively transferred into the domain by this method. In the method proposed by Weatherill and Hassan [8], a typical length scale for the boundary points is first determined, which is computed as the average of the two lengths of the connected edges. It is desired that around the given boundary points and any existing interior points no points should be placed within a distance comparable to the defined length scale. For each point *i*, a region Γ_i, is determined within which no interior point should be placed. In this method, the new points are placed at the centroids of the formed triangles and then a test is performed to determine if any of the new points locate within any Γ_i. If a point lies within Γ_i, it must be rejected, otherwise, it is accepted and connected using the Delaunay triangulation algorithm. Two additional parameters, α and β, are introduced to control the mesh structure, of which one for the vertices of formed triangles and the other for the other points to be inserted. Obviously, in this method, the information contained in the boundary points can be smoothly transported into the domain, while the control in shape of the resulting triangles may not be so direct as that of Anderson's method. Furthermore, the choices of α and β are somewhat arbitrary, and their appropriate values may be problem-dependent and should be determined by try-and-error.

Thus, it can be seen that if an algorithm can combine the advantages of the methods mentioned above, it will certainly prove to be efficient and convenient for the Delaunay triangulation technique.

The purpose of this paper is to present a new, simple and easy-to-implement, fully automatic point creation algorithm. The procedure for creating interior points of convex, non-convex and multiple-connected domains hardly differs. In addition, the proposed method may be easily used to refine local mesh within the domain. In the following, the general procedure for generating unstructured grids in an arbitrary domain with Delaunay triangulation will first be introduced, followed by the detailed description of the proposed point creation algorithm. Since the judge whether a point is within the domain or outside it is frequently performed in the proposed algorithm, a practical convenient method for this judge will also be presented. Finally, several grid generation examples and the preliminary computational results for a fluid flow problem will be provided.

2
General procedure of unstructured grid generation by Delaunay triangulation

A general procedure for generating an unstructured mesh by Delaunay triangulation may be summarized into following four major steps:

- Giving the grid points and connective information on the boundary(ies) of the computational domain. It is assumed that the given boundary grid point distribution reflects the desired coarseness-and-fineness of the grids around different part of the domain.
- Setting up the initial triangles by Delaunay triangulation. This procedure includes the forming of related square which includes all the given boundary points, the creating of initial triangles by Delaunay technique and the removing of the cells whose centroids are outside the given domain.
- Creating points into the domain automatically according to the predetermined criterion.
- Refining mesh (if necessary)

The detailed description of Delaunay initial triangulation can be found in Ref. [6], and here only automatic interior point creation method, especially the algorithm we proposed in the present paper is described in details as follows.

3
Automatic interior point creation

In this section a new method for creating a point into the interior of a computational domain will be presented, which is flexible, easy to implement, requires minimum user input and provides good grid quality.

3.1
Three terminologies

To proceed, three terminologies will first be defined as follows.

Control background network How to obtain grid control information efficiently is essential in the mesh generation. Many approaches have been developed, one of which is the control back-ground [11]. The control back-ground grid is a way to associate grid parameters to grid points all over the domain. Many implementation techniques are proposed, some of which are quite time-consuming. In this paper, we adopt the initial triangulation as the control background network, which can be obtained directly from the initial Delaunay triangulation for the boundary points as described above. Based on the back-ground network, we can define the length scale for any location within the domain.

Length scale The length scale of a boundary point is defined as $\sqrt{3}/2$ times the average distance between the point and its two neighboring boundary points.
For any prospective point to be inserted, its length scale is determined by an interpolation based on reciprocal rule.

This may be illustrated by an example shown in Fig. 1. Let Q be the point within the domain, it will be easy to find which triangle of the control background network encompasses this point (in the example, $\Delta 145$). Compute the distance between Q and the three vertices of the triangle and denote them by l_1, l_4, and l_5 respectively. Then the length scale of point Q is defined as:

$$L(Q) = \frac{L(1)/l_1 + L(4)/l_4 + L(5)/l_5}{1/l_1 + 1/l_4 + 1/l_5} \tag{1}$$

where $L(1)$, $L(4)$ and $L(5)$ are the length scale of the vertices 1, 4 and 5 respectively. It should be noted that for all interior points, their length scales are always computed by the reciprocal interpolation equation. Furthermore, whenever a new grid point is added, we only use the control background network to find the triangle within which the point is located, regardless of how many new triangles have been formed in the computation domain. Therefore, the information of the initial triangulation is always kept in the computer memory during the procedure of grid generation. The reason for this will be clearly understood from the later discussion.

Dimensionless circumcircle radius For each triangle Δk obtained via Delaunay triangulation, its dimensionless circumcircle radius R_k is defined as:

$$R_k = \frac{r_k}{L(x_k, y_k)} \tag{2}$$

where r_k is the circumcircle radius of Δk, $L(x_k, y_k)$ is the length scale of the circumcircle center. At this point, a few words may be added to the above description. The adoption of the reciprocal interpolation (Eq. (1)) is based on the consideration that the nearer the grid point to be added to its neighboring points, the stronger their effects on it. This interpolation rule may enhance the effect of the boundary point distribution character to the interior node coarseness. And it is for this purpose that we insist that for any newly created point its related triangle should be found from the initial triangulation whose vertices are all on the boundary. The value of the dimensionless circumcircle radius represents, in some extent, the size and shape of a triangle. Obviously, at the beginning of interior point creation the existing triangles are usually quite large and very skewed, hence its circumcircle radius is much greater than the length scale of its center. And the bigger the triangle, the larger the value of its dimensionless circumcircle radius. Thus an interior point should first be added into the triangle whose value of R_k is the largest among the existing triangles. And it is interesting to note that for the simplest case where only one equilateral triangle exists, its dimensionless circumcircle radius is 2/3.

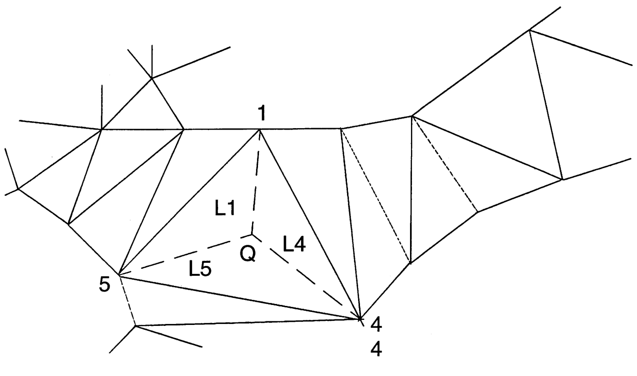


Fig. 1. Illustration for definition of length scale

3.2 Goodness criterion

Each triangle is considered bad, except that its dimensionless circumcircle radius is near 2/3. In this paper, if the dimensionless circumcircle radius R_k of a triangle is less than 1, we think the triangle is acceptable.

Keeping above discussion in mind, attention is now turned to the implementing process of point creation into the initial triangles. The details of the implementation are as follows:

3.3 Interior point creation algorithm

(a) Computing the length scale via the interpolation equation for the circumcircle center of all the existing triangle;
(b) determining the value of R_k for all the existing triangles;
(c) Ordering the triangles according to their values of R_k (from large to small);
(d) Creating a new point at the center of the circumcircle of the triangle at the top of the ordered list (i.e., the triangle with the largest value of R_k);
(e) Regenerating triangles by Delaunay triangulation technique;
(f) Deleting the triangles broken in the triangulation process from the triangle list and inserting the newly generated triangles into the list according to their values of R_k;
(g) If max $(R_k) > 1$, returning to the step d, else, stop.

3.4 Local fineness of the grids

Attention is now turned to the control of the local fineness of the grids. The local coarseness or fineness of the generated mesh can be controlled by many ways. The simplest and most convenient way is to control the grid distribution on the boundary. Ideally any method which automatically creates points (into the domain) should ensure that the boundary point distribution character is extended into the domain in a spatially smooth manner. By introducing the length scale and its particular interpolative rule, this smooth extension of the boundary point distribution character into the domain is guaranteed. For some cases, the grids generated with the above described technique are not fine enough at some local regions within the domain and can not be expected to be sufficient for accurate computations. We can easily define point, line and block sources to provide local fine grids for the unstructured mesh.

This is, to some extent, analogous to the point sources as control functions with elliptic differential equations [10]. In order to refine the local meshes around some places within the domain we take these regions as refinement sources and their positions are first assigned as input data with the boundary points. Besides, the length scales of these points are also assigned a priori, rather than interpolated by the reciprocal rule. The magnitude of the length scale of the point sources may be one order or so less than the average length scale of the boundary points, depending on the problem studied. When set up the initial triangles by the Delaunay triangulation, these points are treated as the boundary points and each inner point should be connected to form triangles. The rest of the implementing procedure is the same as outlined in the above algorithm. Some examples will be illustrated in later discussion.

3.5 Data structure

An appropriate data structure is essential to the implementation of the above described algorithm. The data structure used to implement this algorithm involves an array, *triangle*, which contains the nodes of each triangle (the so-called forming points), and a tree structure, *neighbor*, which, for each triangle, points to the three neighbor triangles. The circumcenter, *coordc*, and *radius*, radius of each circle which passes through the three nodes of each triangle are also required. Hence, the arrays required are *Triangle* (*nt,3*), *neighbor* (*nt,3*), *radius* (*nt*), *coordc* (*nt,2*), *coorp* (*nt,3,2*) where nt denotes the number of the existing triangle meshes and coorp represents the Cartesian co-ordinates of the points.

4 Practical method for judging a point-position

As outlined above, in the implementation of the unstructured grid generation, it is frequently needed to determine whether a point is in the computation domain or not. The judgment is first encountered at the beginning of triangulation, when the extended region should be discarded by judging whether the centroid of each resulting triangle is located within the domain or not. When a new point is to be added at the circumcenter of a triangle, it is also necessary to make sure that this circumcenter is in the computation domain. In this work, a simple and convenient way was developed for this judgment. It can be applied to any kind of domain, whether simply-connected or multi-connected. Take the domain shown in Fig. 2 as an example. It is a very complicated (non-convex, multi-connected) domain. Let p is an arbitrary point with coordinates (x_p, y_p). It is required to check whether point p located within the domain.

An algorithm of judging whether a point is in the computation domain or not is proposed as follows:

(a) From the input data of the boundary points, forming a series of piecewise linear arcs, each of which is ended with two neighbor boundary points. It should be noted that the data of the point coordinates of the same boundary (inner or outer) may be input in a random way, but points for different boundaries are identified.

(b) Drawing a vertical line $x = x_p$, If this line does not intersect any linear arcs mentioned above, point p must be outside the computation domain. If this line intersects any of these linear arcs, it is easily to understand that the number of these intersected arcs must be an even one. In Fig. 2, these arcs are 19–20, 29–30, 34–35, 87–88, 79–80 and 66–67 (the number is 6).

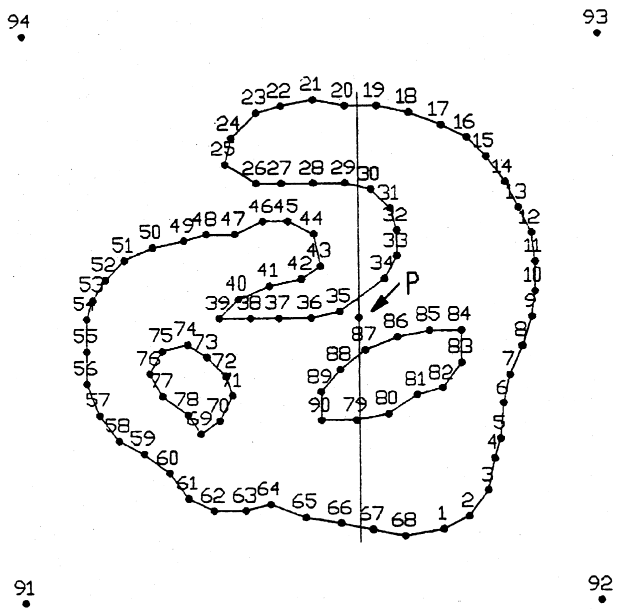


Fig. 2. Practical method for judging point location

(c) Calculating the y-coordinates of the intersected points.

(d) According to the value of y-coordinate of the intersected points, reordering the linear arcs in a range from the lowest to the highest. For the example given in Fig. 2, these arcs are: 66–67, 79–80, 87–88, 34–35, 29–30 and 19–20. They will be denoted as $(i = 1, 2, \ldots 6)$.

(e) Deciding whether the point $p(x_p, y_p)$ locates between any two linear arcs and whether the number of lower linear arc of the two linear arcs is an odd one. If the two answers are both yes, p locates in the domain, else, p outside the domain. To make this judge process efficiently, a special data structure is designed.

5 Grid generation examples

In this section, a few examples of domain triangulations are presented to illustrate the effectiveness and robustness of the method described above. In all cases user interaction with our algorithm was limited to provide necessary boundary information. (Local fineness sources were also treated as boundaries). The algorithm was performed by a special data structure and the time consumed for triangulation was very small, about 3000 triangles could be generated per minute by a personal computer (5 X 86, 166 MHz).

A multi-connected domain (see Fig. 3) with one outer boundary and three inner boundaries was selected as a complex domain to be triangulated by the above algorithm. Even though the domain is very complex, the resulting cells are of good quality, i.e. most of them are closed to equilateral.

Figure 4 illustrated unstructured grid generations by automatic point insertion with sources. Point sources, line source and block source were used to refine local meshes. Figure 4a is an example of domain triangulation with three point sources. Figure 4b presents triangulation with one reflexed line source. Figure 4c shows the case with one point source and one reflexed line source. Finally Fig. 4d shows the triangulation of a region with a block source. All the examples generated by using the approach proposed above indicate that the quality of the triangles generated is good, thus the local fineness method proposed above is effective.

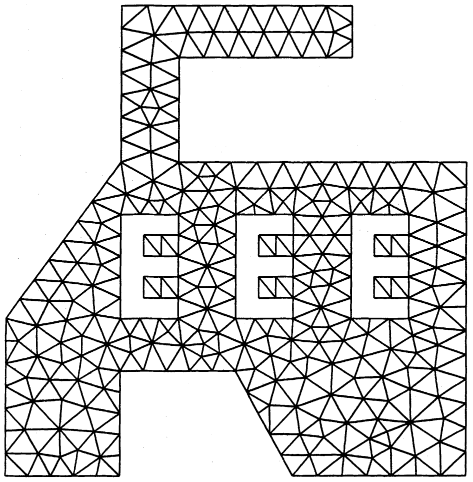


Fig. 3. Generation of triangle mesh for multi-connected regions

Figure 5 represents the triangulation of a practical multi-connected domain. In the cases, the inner boundaries are airfoils, the length scale of the inner boundary is much less than the outer boundary, especially at the top and the tail of the airfoil. The grids generated in the vicinity of the airfoils are much denser than those near the outer circle boundaries, and smoothly extend to the outer boundaries, which illustrates that the dimensionless circumcircle radius is an effective parameter and may be adopted as a good criterion for Delaunay triangulation.

In the previous sections a mesh generated method using Delaunay triangulation criterion has been presented in detail. In the following paragraphs the proposed method is

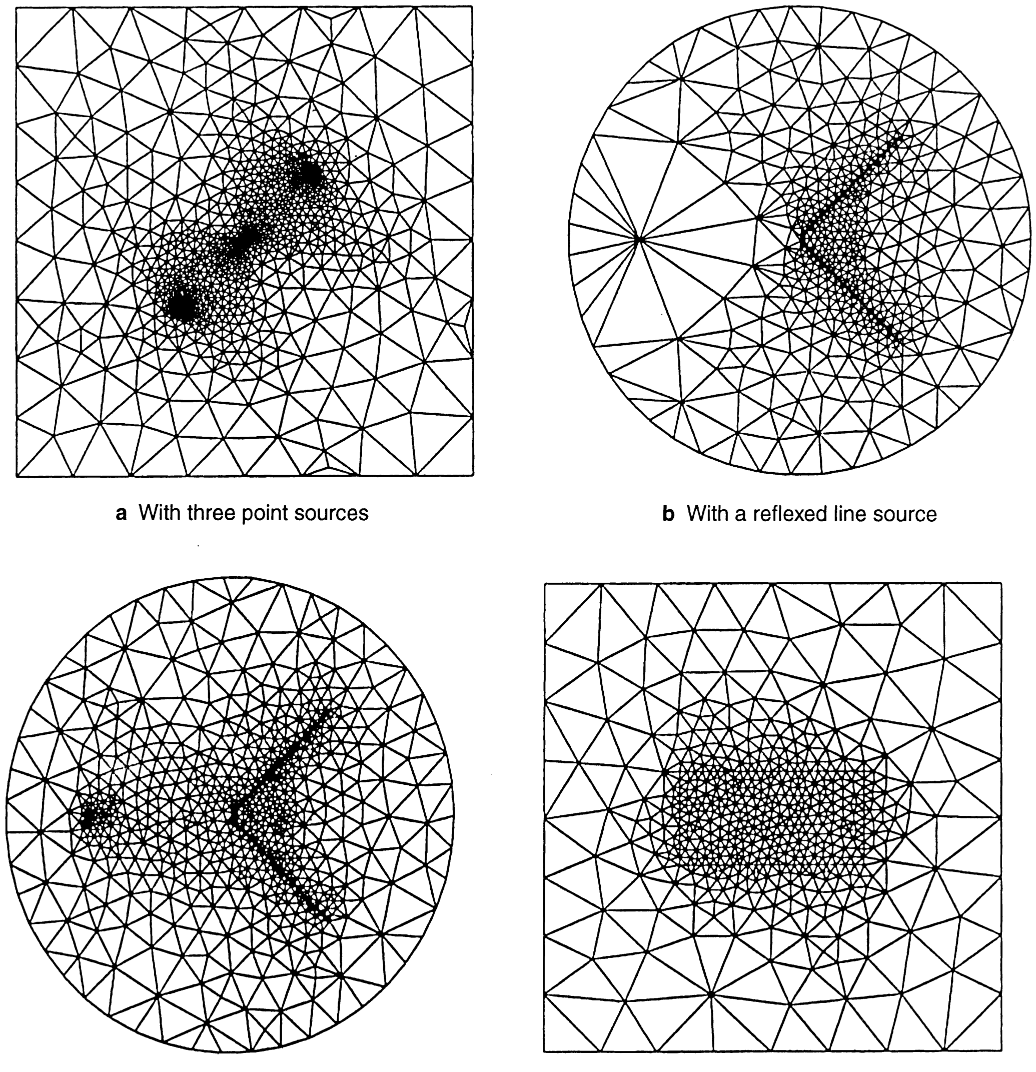
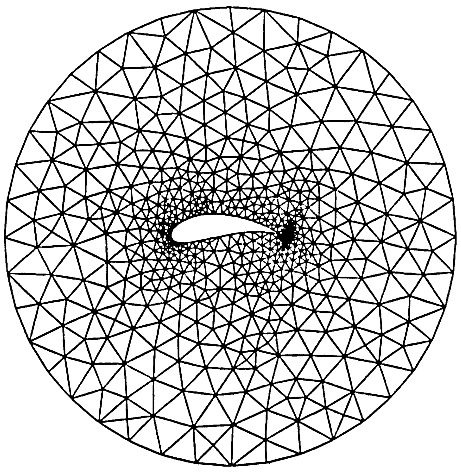

a With three point sources
b With a reflexed line source
c With a point source and a reflexed line source
d With a block source

Fig. 4a–d. Illustration of unstructured grid generation by automatic point insertion with sources. **a** with three point sources; **b** with one reflexed line source; **c** with one point source and one reflexed line source; **d** with block source



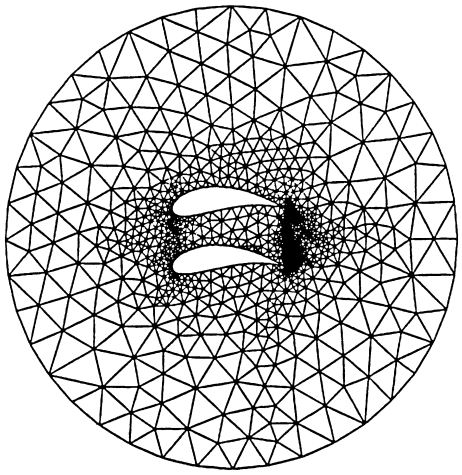


Fig. 5a,b. Triangulation of practical multi-connected domain. **a** with one airfoil inner boundary; **b** with two airfoil inner boundaries

adopted to generate an unstructured grid for simulating turbulent flow and heat transfer in a tube with wave-like longitudinal fins. Numerical methods to discretize governing equations on the unstructured grid will be briefly presented, followed by numerical results observation and comparison with test data.

6 Numerical methods

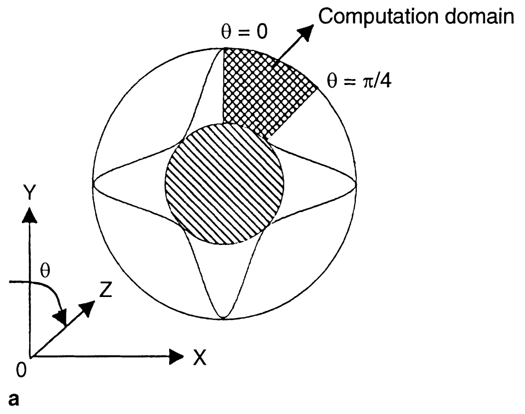
The cross-section of the tube is shown in Fig. 6a. As seen there, longitudinal fins consisted of 4 waves is attached at the tube inner wall. A blocked inner tube is inserted as a cartridge of the fin. From the symmetry of the configuration, only the shaded area is taken as the computation domain. Since a part of the wave-like fin is included in the computation domain, unstructured grids generated by the proposed method were used in the cross-section, while structured grids were adopted in streamwise-direction. The resulting control volume is a pentahedron (Fig. 6b).

Attention is now turned to discretize the equations governing fluid flow and heat transfer in a 3-D domain. The turbulent model adopted in the present study is the standard K-ε model in conjunction with the wall function method. The governing equations for K and ε are:

$$\frac{\partial}{\partial x_j}(\rho u_j K) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\left(\frac{\partial K}{\partial x_j}\right)\right] + P_K - \rho\varepsilon \qquad (3)$$

$$\frac{\partial}{\partial x_j}(\rho u_j \varepsilon) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\left(\frac{\partial \varepsilon}{\partial x_j}\right)\right] + c_1 \frac{\varepsilon}{K} P_K - c_2 \frac{\varepsilon^2}{K} \qquad (4)$$

where



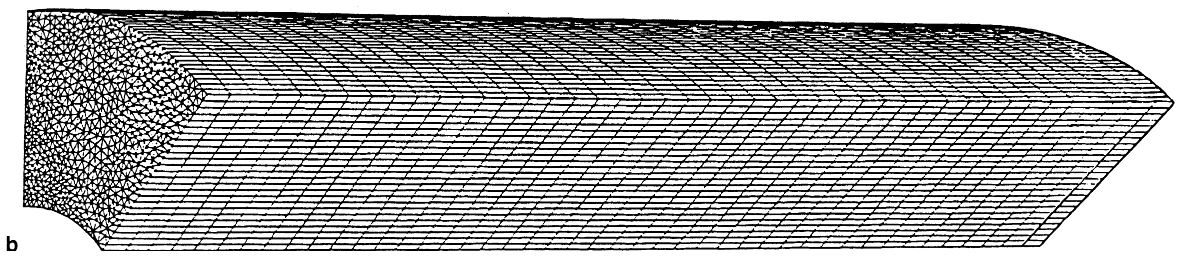


Fig. 6a,b. The finned tube and the computational grid

$$\mu_t = c_\mu \rho \frac{K^2}{\varepsilon} \quad P_K = \left[\mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]\frac{\partial u_i}{\partial x_j} \tag{5}$$

and the values of the empirical coefficients used were taken as follows [11]:

$$C_\mu = 0.09, \quad C_1 = 1.44, \quad C_2 = 1.92$$

$$\sigma_K = 1.0, \quad \sigma_\varepsilon = 1.3, \quad \sigma_T = 0.9$$

For simplicity of discretization, the governing equations for velocities, temperature, turbulent kinetic energy and dissipation are cast into a general transport equation with appropriate choice Φ, Γ and S_Φ:

$$\frac{\partial}{\partial x_j}(\rho u_j \Phi) = \frac{\partial}{\partial x_j}\left[\Gamma\left(\frac{\partial \Phi}{\partial x_j}\right)\right] + S_\Phi \tag{6}$$

The numerical boundary conditions were taken as follows.

1. Inlet boundary

$$u_{\text{in}} = 0, \quad v_{\text{in}} = 0, \quad w_{\text{in}} = w_m,$$

$$T_{\text{in}} = \text{constant}, \quad K_{\text{in}} = iw_m^2, \quad \varepsilon = K_{\text{in}}^{3/2}/(\lambda D_h) \tag{7}$$

where i is the turbulent intensity ($i = 0.25\%$), and $\lambda(\lambda = 0.05)$ is the length scale constant.

2. Outlet boundary
Axial length l of the computation domain was as large as 180 times of D_h, and one-way behavior at the outlet boundary was assumed.

3. Wall
The wall function method was used, which consisted of following treatments:

$$u_w = 0, \quad v_w = 0, \quad w_w = 0,$$

$$T_w = \text{constant}, \quad \left(\frac{\partial K}{\partial n}\right)_P = 0, \quad \varepsilon_P = \frac{C_\mu^{3/4} K_P^{3/2}}{\kappa y_P}(\kappa = 0.4) \tag{8}$$

4. Symmetrical line

$$\frac{\partial \vec{U}}{\partial n} = 0, \quad \frac{\partial T}{\partial n} = 0, \quad \frac{\partial K}{\partial n} = 0, \quad \frac{\partial \varepsilon}{\partial n} = 0 \tag{9}$$

A colocated (non-staggered) variable arrangement was adopted, for which all dependent variables are stored in the centroid of the control volume. Integration of Eq. (6) [12] on an arbitrary control volume yields:

$$\sum_{j=1}^{n} F_j \Phi_j = \sum_{j=1}^{n} D_j + (S_\Phi \Delta V)_{p_0} \tag{10}$$

where F_j is the control volume face mass flux, D_j is the transport due to diffusion (diffusive flux) through the face j, and Φ_j is the value of the cell face. To avoid the checkerbroad pressure field, a special interpolation practice for cell-face velocity proposed by Rhie and Chow [13] was adopted to calculate the mass flux at each cell face.

To determine the face value Φ_j, CDS (Central Difference Scheme) is used in the present study:

$$\Phi_j = g_P \Phi_{P_0} + (1 - g_P)\Phi_{P_j} \tag{11}$$

where g_P is an interpolated factor. In the solution of resulting algebraic equations, the deferred correction method suggested by Khosla and Rubin [14] was used to avoid iterative instability

The diffusion term at the cell face is:

$$D_j = D_j^m + D_j^c \tag{12}$$

where D_j^m and D_j^c represent main diffusion and cross diffusion flux, respectively. The main diffusion flux can be directly determined by:

$$D_j^m = \Gamma_j \frac{\Phi_{P_j} - \Phi_{P_0}}{|\vec{d_j}|} \frac{\vec{d_j}}{|\vec{d_j}|} \cdot \vec{A} \tag{13}$$

where Φ_{P_0}, Φ_{P_j} are the values of variable Φ at the centroids of control volume P_0 and P_j, respectively. Γ_j is the diffusivity of the cell face, $\vec{d_j}$ is a distant vector from the centroid of control volume P_0 to that of P_j and $\vec{A}$ is normal vector of the cell face.

In order to avoid the use of face tangents and the related nodes, the cross diffusion flux is written as the difference between the total diffusion and the main diffusion component. That is:

$$D_j^c = \Gamma_j \left(\overline{\nabla \Phi_j} - \frac{\Phi_{P_j} - \Phi_{P_0}}{|\vec{d_j}|} \frac{\vec{d_j}}{|\vec{d_j}|}\right) \cdot \vec{A} \tag{14}$$

where $\overline{\nabla \Phi_j}$ is the gradient at the cell face j which was taken as the average of the gradients at the two adjacent cells. In the present work a least-squares method was used to calculate the three components of the gradient of Φ at centroids P and then their values were averaged to obtain $\overline{\nabla \Phi_j}$.

The resulting linear algebraic equations can be cast into the following form:

$$a_P \Phi_P = \sum_{j=1}^{n} a_{\text{nb}} \Phi_{\text{nb}} + S_P \tag{15}$$

To minimize storage requirements, a segregated solution strategy was adopted, with pressure and velocity coupled by the SIMPLE algorithm [15]. Gauss-seidel iteration method was used to solve the discretized equations. The discretized computational domain and the control volumes which are pentahedra in shape are shown in Fig. 6b. There are 45 nodes in the streamwise direction and 1000 nodes in the cross-section, resulting total 45 000 nodes in simulation. For the use of the K-ε model in conjuction with wall function, the near wall nodes should be far enough from the wall. In the present study, the range of y^+ was generally from 20 to 90 for all Reynolds number which guaranteed validity of K-ε model with wall function. The convergence criterion used in this study was that the maximum mass flux residual in each control volume was less than 1.0e-5 and the maximum relative deviations of both the most representative velocity and temperature were less than 1.0e-4.

7
Numerical results and comparison

Our final purpose is to obtain the relation between Nu and Re. Hydraulic diameter was used as the characteristic length which was determined as follows:

$$D_h = \frac{\pi(D_i^2 - d_o^2) - 4c\delta}{\pi(D_i + d_o) + 2c} \qquad (16)$$

where D_i is the inner diameter of the test tube, d_o is the outer diameter of the inner tube, c is the expansion length of the wave-like fin within annulus, δ is the thickness of the fin.

The local heat flux q_z and local heat transfer coefficient h_z are defined as:

$$q_z = k_t \frac{t_w - t_p}{\Delta r_P} \qquad (17)$$

$$h_z = \frac{q_z}{t_w - t_b} \qquad (18)$$

where t_w is the temperature of the wall surface, t_p is the air temperature of the control volume adjacent to the wall surface, k_t is turbulent thermal diffusivity, Δr_P is the normal distance from the point P which is the centroid of control volume P. The turbulent thermal diffusivity is related to μ_t by the following equation:

$$\sigma_T = \frac{\mu_t C_P}{k_t} \qquad (19)$$

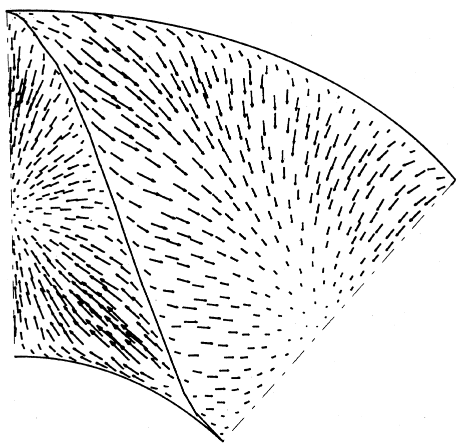
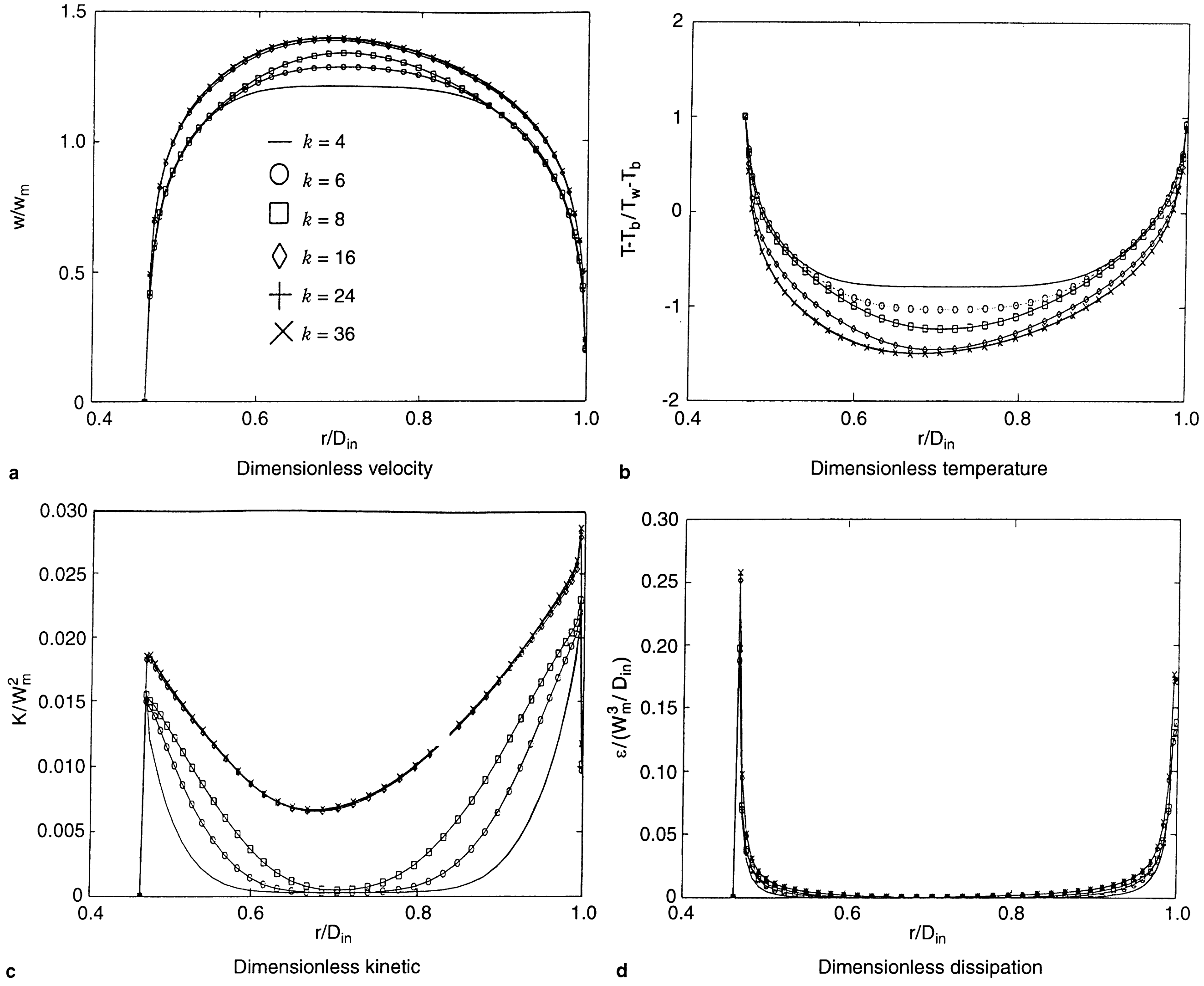


Fig. 7. Velocity vector diagram in the developing region of the X Y plane

The average heat transfer coefficient over the surface (include wall and fin) is defined as:

Fig. 8a–d. Distribution of dimensionless element at the symmetric line ($\theta = \pi/4$) (k-axial index of cross section)



$$h_m = \int_A h \, \mathrm{d}A/A \quad (20)$$

Two important dimensionless parameters Re, Nu are defined as follows:

$$\mathrm{Re} = \frac{w_m D_h}{\nu}, \quad \mathrm{Nu} = \frac{h D_h}{k} \quad (21)$$

In the computation the Reynolds number was varied in the range of 3300–30 000.

Attention is now turned to the computational results which will be presented in following sequence: (1) the cross-section flow field; (2) the distribution of velocity, temperature, turbulent kinetic energy and dissipation; (3) local Nusselt number distribution; (4) comparison with experimental results.

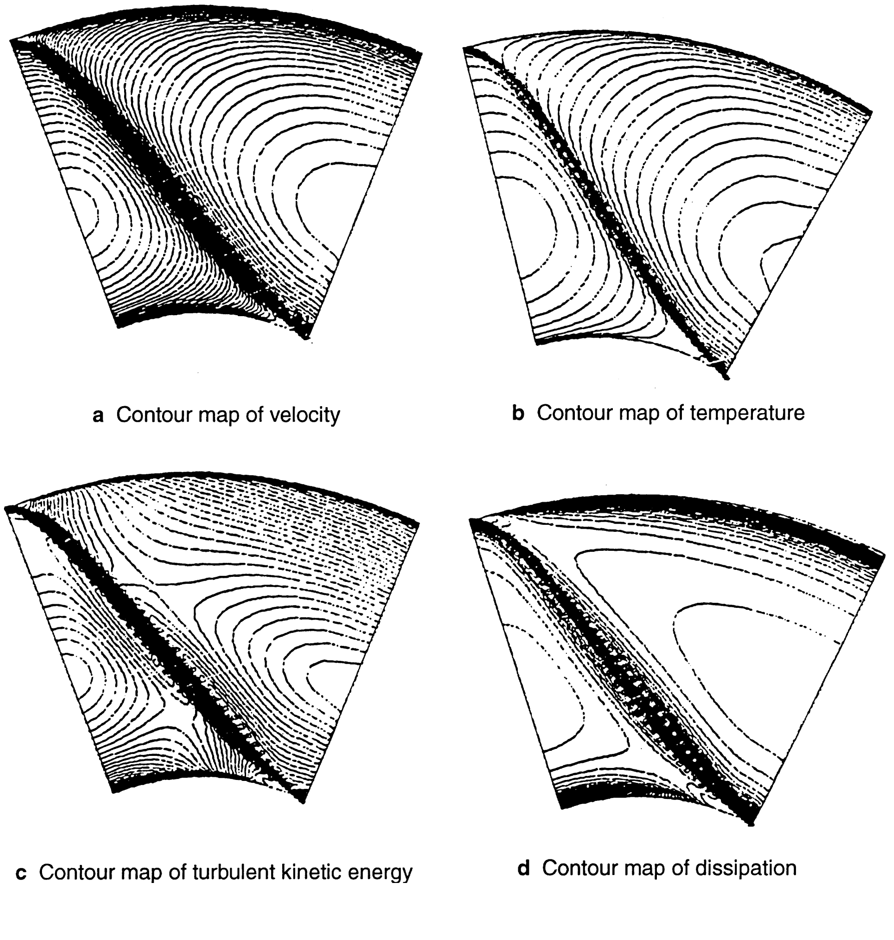
(1) The cross-section flow field
A typical flow field at a cross section with $z/D_h = 12.3$ (Re = 16 875) is shown in Fig. 7, It can be found from the vector diagram that the fluid flows away from the wall and fin. No secondary cross flow was found in parameter range computed in this study.

(2) The distribution of velocity, temperature, turbulent kinetic energy and dissipation
The distributions of dimensionless main flow velocity, dimensionless temperature, dimensionless kinetic energy and dimensionless dissipation at the symmetrical line $\theta = \pi/4$ (Fig. 6a) of several representative cross section are shown in Fig. 8a–d, respectively. It can be distinctly found that the flow in the finned tube at Re = 16875 is in the developing region before $k = 24$ (i.e. $l_z < 99D_h$), after that position the flow can be considered fully developed; furthermore both the kinetic energy and dissipation adjacent to the wall are much greater than their counterpart at the center of the symmetrical line. The contour maps of the computational velocity, temperature, kinetic energy and dissipation for the fully developed region in the case of Re = 16875 are given in Fig. 9.

(3) Local Nusselt number distribution
The local Nusselt number is plotted against the axial coordinate Z as shown in Fig. 10. It can be found that the thermal entrance length is about 40–110D_h for the problem studied, which agrees well with our experimental results.

(4) Comparison with experimental results
The Nusselt numbers in the fully developed region are presented and compared with the experimental results in Fig. 10. where the black triangles and squares represent the calculating data and the experimental data, respectively [16]. It can be seen from the figure that both the experimental and computational Nusselt number can be well-correlated by a power-law equation.

The numerical results agree fairly well with the experimental ones with a maximum deviation of 25%. The main reasons for the difference may be attributed to the fol-



a Contour map of velocity

b Contour map of temperature

c Contour map of turbulent kinetic energy

d Contour map of dissipation

Fig. 9a–d. Contour maps of the computational velocity, temperature, kinetic and dissipation in the fully developed region

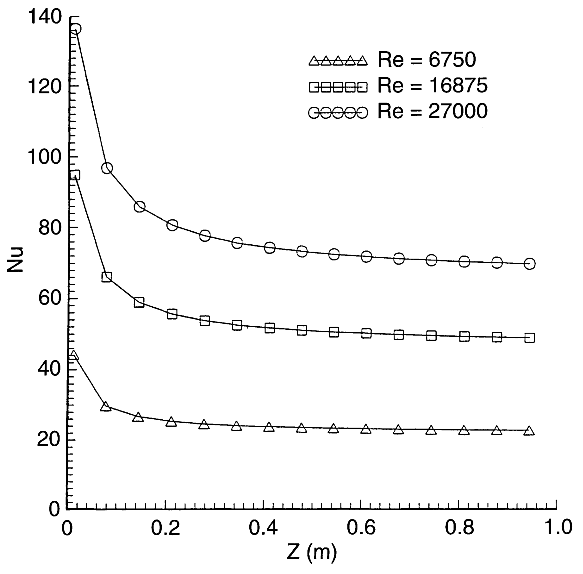


Fig. 10. Local Nusselt number distribution

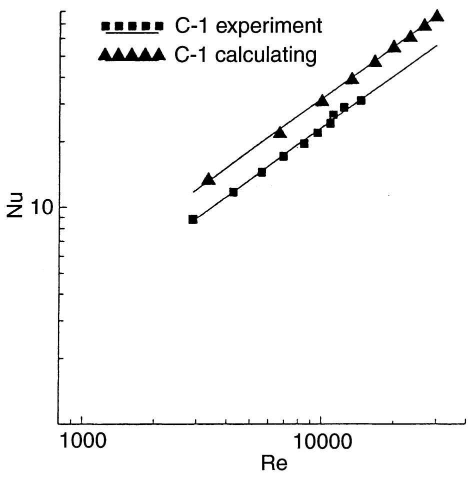


Fig. 11. Comparison of predicated Nusselt number with experimental results

lowing three aspects: (1) experimental uncertainty (2) the simulation condition is not quite the same as the experimental ones. For example, uniform inlet velocity was assumed in the present study, thus the flow rate of the two parts separated by the fin, inner and outer tube, were proportional to the areas of the two parts, which, apparently, might not match our test condition and (3) the drawback inherent in the turbulent model adopted.

8 Conclusions

In this paper a robust, simple and fully automatic unstructured grids generation for a two-dimensional region is presented. The Delaunay triangulation method is used. A new conception, dimensionless circumcircle radius, is proposed as a good criterion for Delaunay triangulation. And an automatic point creation algorithm is proposed, which can smoothly extend the boundary points distribution character into the interior. This criterion can keep the resulting cells as equilateral as possible. Local mesh fineness can be easily implemented by regarding the region sources as boundaries. A practical simple algorithm for judging a point location is also presented. Several general meshes are presented to show the effectiveness and feasibility of the proposed method.

The proposed method was used to generate an unstructured grid system to simulate turbulent flow and heat transfer in a tube with wave-like longitudinal fins. The discretization process on the unstructured grid was briefly described. The numerical results show that the entrance length of the tube studied varied with Reynolds number, ranging from 40 to 110 in the Reynold number range from 3300–30 000. The average Nusselt numbers in the fully developed region agree fairly well with author's test data with a maximum deviation of 25%.

References

1. **Lobner R; Parikh P** (1988) Generation of 3-D unstructured grids by advancing front method. Int J Num Meth Fluids 8: 1135–11453
2. **Peraire J; Morgan K; Peiro J** (1996) Unstructured mesh generation by advancing front method. In: Numerical Grid Generation, von Karman Institute for Fluid Dynamics Lecture Series, von Carman Institute, Brussels, Belgium pp. 37–68
3. **Parzadeh S** (1993) Structured background grids for generation of unstructured grids by advancing front method. AIAA J 31: 257–265
4. **Bowyer A** (1981) Computing Dirichlet Tessellations. The Computer Journal 4(2): 162–166m
5. **Lo SH** (1989) Delaunay Triangulation of non-convex planar domain. Int J Num Meth Fluids 28: 2695–2707
6. **Anderson WK** (1994) A grid generation and flow solution method for the Euler equations on unstructured grids. J Comput Phys 110: 23–38
7. **Sjubramanian G; Raveendra VVS; Geopalakrishna K** (1994) Robust boundary triangulation and Delaunay triangulation of arbitary planar domains, Int J Num Meth Eng 37: 1775–1788
8. **Weatherill NP; Hassan O** (1994) Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints, Int J Num Meth Eng 37: 2025–2039
9. **Holmes GD; Snyder DD** (1988) The grid generation of unstructured triangulation methods using Delaunay triangulation, In: Num Grid Generation in Computational Fluid Dynamics, pp. 643, Pineridge Press, Swasea, UK
10. **Thompson JF; Warrsi ZUA; Mastin CW** (1985) Numerical Grid Generation, Foundations and Applications. North-Holland, New York
11. **Tao WQ** (1988) Numerical Heat Transfer. Xi'an Jiaotong University Press, Xi'an, (in Chinese)
12. **Baliga BR; Pham TT; Patankar SV** (1983) Solution of some two-dimensional incompressible fluid flow and heat transfer problems using a control volume Finite-Element Method. Num Heat Transfer 6: 263–282
13. **Rhie CM; Chow WL** (1983) Numerical study of the turbulent flow past an artificial with trailing edge separation. AIAA J 121: 1525–1532
14. **Kbosla PK; Rubins SG** (1974) A diagonally dominant second order accurate implicit scheme. Comput Fluids 2: 207–209
15. **Patankar SV** (1980) Numerical Heat Transfer and Fluid Flow, McGraw-Hill, New York
16. **Yu B** (1998) Experimental and Numerical study on the convective heat transfer of internally finned tubes and finite Volume Method for unstructured meshes. Doctorate Thesis, Xi'an Jiaotong University