An Automated Mesh Generation Algorithm for Curved Surfaces of Ship Longitudinal Structures

Chuan Wang[1,2], Yiling Cao[1], Wei Lin[1], Liping Wang[3], Youhong Tang[3] and Chengbi Zhao[1]

[1]South China University of Technology,tccbzhao@scut.edu.cn
[2]Shenzhen Chiwan Sembawang Engineering Co. Ltd
[3]Flinders University, youhong.tang@flinders.edu.au

ABSTRACT

An automated finite element mesh generation algorithm for ship hull surfaces is proposed and applied effectively in this work. The algorithm plays an important role in dealing with the intersections between ship hull surfaces and its frames, since the operations for flexural surfaces and curved lines are complex and quite different from those for general structures. In this study, a specialized planar curve intersection method and a local refinement intersection algorithm between a non-uniform rational B-spline (NURBS) surface and an implicit surface are combined, leading to high mesh quality and close approximation to the actual ship hull structure. The proposed algorithm, with high applicability, improves the efficiency and accuracy of finite element modeling of ship hull surfaces.

Keywords: ship hull surface, finite element mesh generation, intersection of B-spline curves and surfaces.

1. INTRODUCTION

In the progress of ship hull surface finite element (FE) modeling, the quality of mesh has a significant influence on the result of FE analysis (FEA). Though the surface mesh generation technique has led to significant progress in the development of the FE method (FEM) in recent years, not all these algorithms are applicable FE mesh generation for complex ship hull surface structures. Traditionally, for hull surface manual FE modeling, the fine mesh is generated by linear interpolation and some manual modeling based on a coarse mesh composed of the transverse and longitudinal structure member intersectional points. Since the surface structures are curved as illustrated in Fig. 1(a), these traditionally generated manual FE meshes are quite different from the actual surface structures, leading to significant errors and non-ideal expression. However, the generated mesh is closer to actual situation using the method shown in Fig. 1(b), with each FE node on the hull surface and structure. The analysis result with this gird is more reliable.

On the other hand, there are many geometrical element intersections in the process of hull surface FE mesh interpolation generation. For instance, longitudinal frame structures are treated as the intersections of curved surfaces and planes, and porthole windows are treated as the intersections of curved surfaces and cylinder surfaces. The intersection points of longitudinal and transverse structure members are taken as the key points of FE meshes. These intersections spread all over the mesh, leading to a situation where intersection operations constitute most of the FE mesh process for curved surfaces. The quality of intersection operations thus plays a vital role in the accuracy and efficiency of the FE mesh. In this research, a different automated FE mesh generation algorithm is proposed, to better approximate real ship hull surface structures and to reduce the workload of manual modeling. An intersection algorithm between B-splines and an intersection algorithm between a non-uniform rational B-spline (NURBS) surface and an implicit surface are designed, to be embedded in the mesh generation method on a curved surface and to support it for accuracy and efficiency.

1.1. Surface FE mesh Generation Algorithm

There are two main ways to generate surface FE mesh. One is the direct mesh method, with meshes directly generated on a physical surface. This method is mainly represented by the surface decomposition method and the direct advancing front technique (AFT) [4,11,16] based method. The direct mesh method has a good curved surface adaptive capacity and local refinement ability, but its efficiency is lower than that of the mapping method.

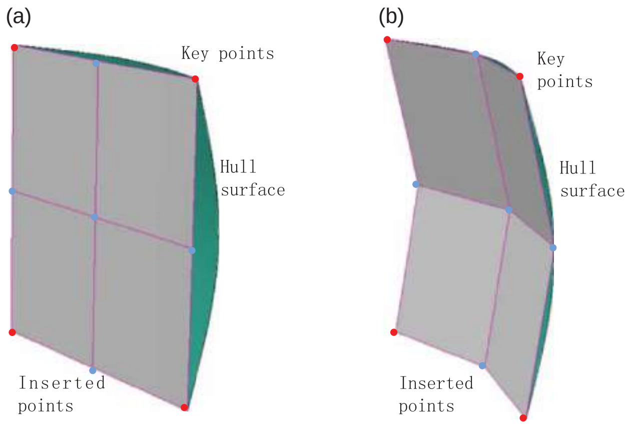

Fig. 1: Finite element mesh: (a) linear interpolation, (b) surface interpolation.

The mapping method, also called the parametric spatial method, is generated on the parametric field by the planar mesh method and mapped on the physical surface to form the surface FE mesh model. Although the high-quality triangle and quadrangle planar mesh method is quite mature, distortion often occurs in the process of mapping [2,3,13,14,17, 21,25,26].Though the researches made a great contribution to automatic surface FE mesh generation, these are not directly applicable for ship hull surface mesh generation. Since not only the curved surface related closely to the ship hull surface mesh generation, but also the transversal and vertical components on the hull.

In this research, the ship hull is described in NURBS surface. The coarse mesh is composed of the intersection points of transversal and vertical hull components. The fine mesh is obtained by the interpolation of the surface and the coarse mesh. The mesh generated by this method is closer to the actual hull surface and considered reasonably with the ship components.

1.2. Algorithm of Planar B-spline Curves Intersection

The intersection algorithm of curves is one of the key points for a complex curved surface. There are several kinds of cubic B-spline curve intersection methods, including the algebraic method, subdivision method, iteration method, etc. Goldman [5] and Sederberg et al. [18] used the algebraic method that runs quickly for low order parametric curves with elimination theory and extraction of roots. Sederberg and Nishita [19] and Koparkar and Mudur [10] used the sub-division method and solved the problem of more than one point. But that method is inefficient since large amounts of computer memory and calculation time are needed. Jiang [8] also used the subdivision method, setting the range boxes of maximum and minimum point values (max min boxes) of two B-spline curves as rectangle bounding boxes. He judged the intersection points by the two max min boxes with the repeating process of judge-divide-rejudge-subdivide, that did not stop until the curved parts of the two curves were replaced by the straight line parts and the two max min boxes had intersected. Li and Shi [12] turned the two B-spline curves into Bezier curve segments and used their bounding boxes as judgment and subdivision until the divided curve section was smooth. The methods proposed by Jiang [8] and Li and Shi [12] belong to the subdivision method. They have the advantage of helping to figure out all the intersection points and running rapidly, but they are restricted to low precision and the objects must be subdivided and calculated constantly with high precision requirements for engineering applications, a time consuming process. Among those employing iteration methods, Hoscheck and Lasser [6] used the tangent line of the points to replace the curved sections and calculated the step size by Newton projection. Jin and Wang [9] and Hu et al. [7] used the iteration method of a moving affine frame (MAF) intersection operation with the step size calculation method. Procedures using the iteration method run quickly, but they are quite sensitive to the initial iteration values. Avoiding the disadvantages of the Newton and MAF methods, Zhang and Huang [24] proposed a curvature circle iteration method based on secondary curve approximation, which is closer to the real curve than the first order. They gave the detailed algorithm of step size reliability, which was the corresponding maximum of each step size, and utilized the curvature circle to calculate the step size. In Zhang and Huang [24] research, 20 fourth order B-spline curves were generated. Each of them has 4 to 20 control points. Each intersection is iterated for 50 times. The random test results respectively under the MAF method of first-order iteration step size reliability and the curvature circle iteration method are illustrated as Table 1.

As shown in Table 1, it is easier to obtain more accurate intersection points with less iteration using Zhang and Huang [24] method compared with the MAF algorithm. But it is only applicable when two B-spline curves have a single intersection point. Moreover, there is no operation for an iteration value outside of the definition field. Meanwhile, the accuracy of step size calculation with angle variation is low when two curvature circles intersect.

The intersection algorithm of B-spline curves proposed in this study combines the segmentation intersection method and the curvature circle iteration

Method	MAF algorithm with 1.order iteration step	Curvature circle algorithm
Times	5563	5563
Success times	4762	5559
Iteration times	64998	27879
Success rate	4762/5563=85.6	5559/5563=99.9
Average iteration times	64988/5563=11.7	27879/5563=5

Tab. 1: Random test result of B-spline curves intersection [24].

method based on 2nd order approximation curves. The initial iteration value is acquired by the process of judge-divide-rejudge-subdivide. And the intersection points between two B-spline curves are located by the curvature circle iteration method based on 2. order approximation curves. The proposed method has higher success rate and less iteration times. And it is suitable for numbers of intersection points.

1.3. Algorithm of Intersection between NURBS Surface and Implicit Surface

The algorithm of intersection between a NURBS surface and an implicit surface is another key point for operations involving complex curved surface intersection. Yu et al. [22] introduced a surface intersection algorithm of parametric surface and implicit surface, based on the contour line extracted on a two-dimensional scalar field. It can calculate all the intersection lines and avoid the need for selecting initial points for iteration. More important, it is applicable to the intersection of trimmed surfaces. Zhang et al. [23] extended the algorithm of Yu et al. [22] to detailed applications including calculating the frame lines. Song et al. [20] further developed the intersection program to verify the reliability of the contour line algorithm. These algorithms deal with mesh elements one by one, according to their permutation order. Since all mesh elements should be dealt with, the efficiency is non-ideal, especially when surface curvature changes markedly or when the accuracy requirement of intersection lines is high with dense regular meshes. Furthermore, linear interpolation is used in the algorithm and the required accuracy for intersection points may not be satisfied, as well as the smoothness.

According to the isoline interpolation theory of a NURBS surface and an implicit surface, an improved local refinement intersection algorithm is proposed in section 3. The main improvements are shown as below.

(1) The calculation times of h value on vertexes were reduced by local refinement algorithm. And the efficiency of curved surface intersection algorithm was increase greatly.
(2) The quasi-Newton method was used for the intersection points search between gird cells and isolines, improving the algorithm accuracy.
(3) The intersection lines of NURBS surface and implicit curved surface are smoother with isolines matched by triple B-spline curves.
(4) The residual value is applied and set for accuracy control, making the accuracy of algorithm easier to control.

The proposed algorithm between a NURBS surface and an implicit surface and can be used to acquire ship waterlines, vertical stations, porthole and anchor span boundary lines etc. It is the base of the following work for ship hull automatic FE mesh generation.

1.4. Ship Hull Surface Modeling

A ship hull surface description is the decisive condition for the hull mesh generation. In this study, the ship hull surfaces are modeled in NURBS surfaces according to the hull offsets and the NURBS inverse operation algorithm proposed by Lu [15]. A part of a container vessel stern is modeled as an example using the NURBS inverse operation algorithm, as shown in Fig. 2. With its high accuracy and flexibility, the NURBS surface description method lays a good foundation for the quality of mesh generation in following work.

2. A PLANAR B-SPLINE CURVES INTERSECTION ALGORITHM

A circle of curvature iteration algorithm is proposed here. A rough iterative initial value is acquired by the process of judge-divide-rejudge-subdivide. The intersection point values are calculated by iteration with the circle of curvature algorithm. The range boxes of maximum and minimum point values (max min boxes) are generated before initial iteration values are obtained, which is shown as below. As illustrated in Fig. 3, there is a B-spline curve, with n discrete points distributed evenly on it. The corresponding parameters of points are $[\bar{u}_1, \bar{u}_2, \cdots, \bar{u}_n]$

Assume that the coordinates of discrete points are $\boldsymbol{p}(\bar{u}_i) = (x_i, y_i)$, $i = 1, 2, \cdots, n$. The max min box vertexes of whole curve are illustrated in Fig. 4. Then the coordinates of vertex A is $(\min([x_1, x_2, \cdots x_n]), \min([y_1, y_2, \cdots y_n]))$. Vertex C is

$(\max([x_1, x_2, \cdots x_n]), \max([y_1, y_2, \cdots y_n]))$. To dived n discrete points into k parts, the max min box vertex of each curve section could be obtained by above way, illustrated in Fig. 5.

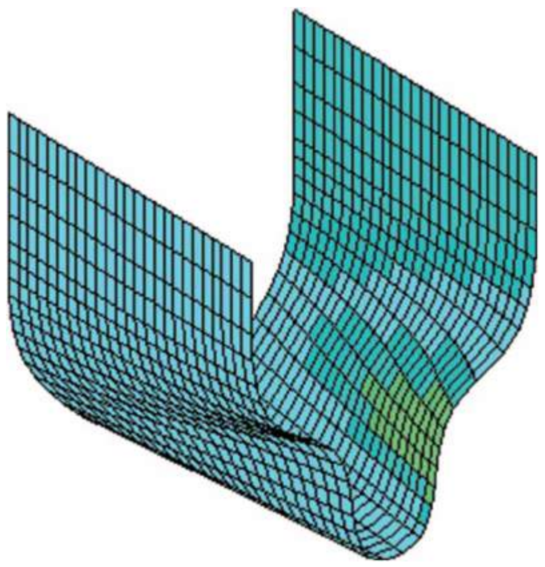

Fig. 2: Part of a container vessel stern in NURBS surface.

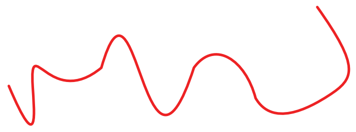

Fig. 3: The B-spline curve.

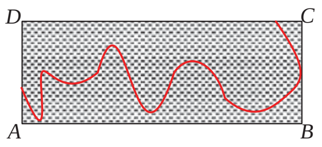

Fig. 4: The max min box of the curve.

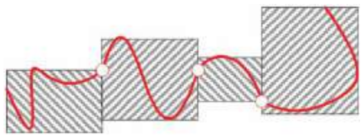

Fig. 5: The max min box after dividing.

2.1. Initial Value and Step Size

Suppose that B-spline curve $\boldsymbol{p}(u)$ intersects B-spline curve $\boldsymbol{q}(v)$. The process of obtaining the initial iteration value is shown as below.

(1) Divide B-spline curves into segments and build the max min box of each segment.
(2) Suppose that the max min box of $\boldsymbol{p}(u)$ is m_{1i}. The dealing process of a max min box is illustrated as Fig. 6.
(3) Check the entire max min box on curve $\boldsymbol{p}(u)$ as above process and renumber the subdivided boxes.
(4) Repeat (2) and (3). The process of judge-divide-rejudge-subdivide is illustrated in Fig. 6. The initial iteration values for intersection point on curve $\boldsymbol{p}(u)$ and $\boldsymbol{q}(v)$ are obtained as $u_0 = \frac{u_2-u_1}{2}$, $v_0 = \frac{v_2-v_1}{2}$. The process is suitable for the initial iteration value of other intersection points.

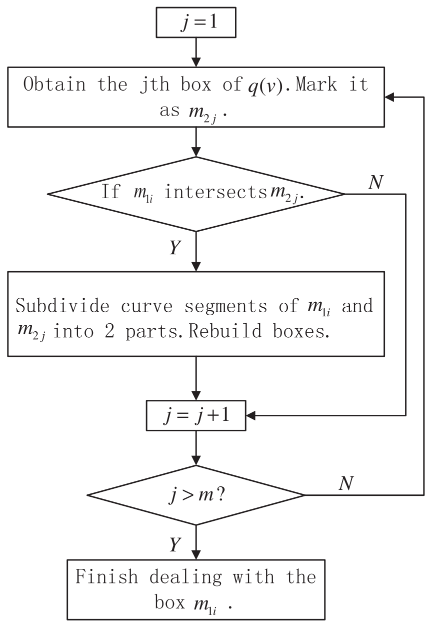
$j = 1$
Obtain the jth box of $q(v)$. Mark it as m_{2j}.
If m_{1i} intersects m_{2j}.
N
Y
Subdivide curve segments of m_{1i} and m_{2j} into 2 parts. Rebuild boxes.
$j = j+1$
$j > m$?
N
Y
Finish dealing with the box m_{1i} .

Fig. 6: The process of a max min box computation.

The max min boxes of two B-spline curves were formed after division as shown in Fig. 7(a). These boxes were judged for intersection and re-divided as shown in Fig. 7(b), then intersections was judged in the new boxes for the third time and bounds were narrowed for more accurate initial iteration values as shown in Fig. 7(c). The curve segments are replaced by the curvature circles arc calculated by the curvature circle and secondary step size reliability. The secondary step size reliability is derived as: $\Delta u_{\max}$ is marked as the secondary step size reliability; arc curvature on the point $\mathbf{p}(u_0)$ of the B-spline curve $\mathbf{p}(u)$ in the global coordinate system is indicated in the following formula; $\mathbf{x}_1$ and $\mathbf{y}_1$ are respectively the unit vectors in the $\mathbf{OX_1}$ direction and $\mathbf{OY}_1$ direction; $\mathbf{p}'(u_0)$ is the first order derivative of $(x(u_0), y(u_0))$; $\mathbf{p}''(u_0)$ is the second order derivative and $\mathbf{p}'''(u_0)$ is the third

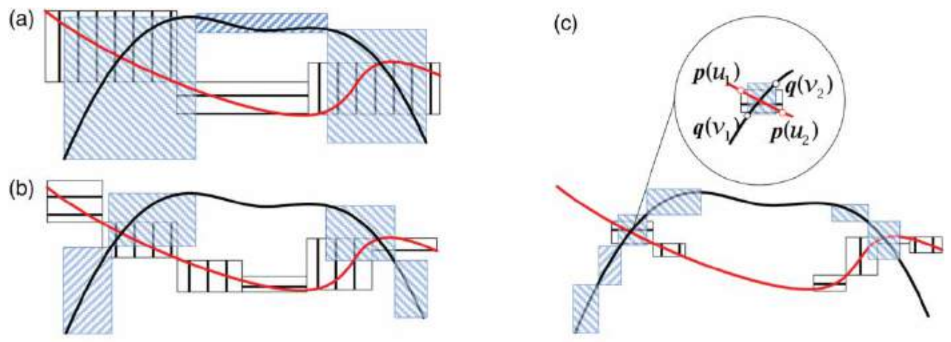

Fig. 7: The process of judge-divide-rejudge-subdivide on curves: (a) the first subdivision with corresponding max min box, (b) the second subdivision after judgment and (c) the third subdivision.

order derivative.

$$\mathbf{C}(u) = \mathbf{O} + r\cos(ku)\cdot\mathbf{x}_1 + r\sin(ku)\cdot\mathbf{y}_1,$$
$$u \in (-\Delta u_{\max}, \Delta u_{\max}1) \tag{1}$$

where the radius of curvature curve is $r = \frac{\|\mathbf{p}'(u_0)\|^3}{|\det(\mathbf{p}'(u_0),\mathbf{p}''(u_0))|}$.

The center coordinates $\mathbf{O}(x_o, y_o)$ of the curvature circle in the global coordinates system are

$$x_O = x(u_0) - y'(u_0)\frac{\|\mathbf{p}'(u_0)\|^2}{|\det(\mathbf{p}'(u_0),\mathbf{p}''(u_0))|},$$
$$y_O = y(u_0) + x'(u_0)\frac{\|\mathbf{p}'(u_0)\|^2}{|\det(\mathbf{p}'(u_0),\mathbf{p}''(u_0))|} \tag{2}$$

The values of $\mathbf{x}_1$ and $\mathbf{y}_1$ are illustrated as

$$\mathbf{x}_1 = \frac{\mathbf{p}(u_0) - \mathbf{O}}{\|\mathbf{p}(u_0) - \mathbf{O}\|} = \frac{(y'(u_0), -x'(u_0))\,\|\mathbf{p}'(u_0)\|^2}{r\,|\det(\mathbf{p}'(u_0),\mathbf{p}''(u_0))|},$$
$$\mathbf{y}_1 = \frac{\mathbf{p}'(u_0)}{\|\mathbf{p}'(u_0)\|} \tag{3}$$

The difference value between $\mathbf{C}(u)$ and $\mathbf{p}(u)$ is shown below after Taylor expansion and ignoring the remainder of $o(u^3)$:

$$R(u) = \frac{\mathbf{p}'(u_0)\cdot\mathbf{p}''(u_0)}{2\,\|\mathbf{p}'(u_0)\|}\cdot\frac{\mathbf{p}'(u_0)}{\|\mathbf{p}'(u_0)\|}u^2 + \left[\frac{\mathbf{p}'''(u_0)}{3!} - \frac{|\det(\mathbf{p}'(u_0),\mathbf{p}''(u_0))|^2}{3!\,\|\mathbf{p}'(u_0)\|^4}\mathbf{p}'(u_0)\right]u^3 \tag{4}$$

The margin of error is set as dr. The derived secondary step size reliability is shown as

$$\Delta u_{\max} = \left(\frac{dr}{\|\alpha + \beta\Delta\bar{u}_{\max}\|}\right)^{\frac{1}{2}} \tag{5}$$

where $\Delta u_{\max}$ is the secondary step size reliability, $\Delta\bar{u}_{\max}$ is the middle iteration value and dr is the maximum allowable error.

$$\alpha = \frac{\mathbf{p}'(u_0)\cdot\mathbf{p}''(u_0)}{2||\mathbf{p}'(u_0)||}\cdot\frac{\mathbf{p}'(u_0)}{||\mathbf{p}'(u_0)||} \quad \text{and}$$
$$\beta = \frac{\mathbf{p}'''(u_0)}{3!} - \frac{|\det(\mathbf{p}'(u_0),\mathbf{p}''(u_0))|^2}{3!||\mathbf{p}'(u_0)||^4}\mathbf{p}'(u_0).$$

2.2. Solving two Planar B-spline Curves Intersection Problems

Two planar B-spline curve intersection problems can be solved with curvature circle iteration algorithm according to above, and the process is shown in Fig. 8.

(1) Set the initial iterative values. In the process of judge-divide-rejudge-subdivide with the partitioning algorithm, the intersection points of two planar B-spline curves will be judged. If there is no intersection point, the algorithm terminates. If there are intersection points, the number of points can be calculated roughly, and the n iterative initial values signed as.
(2) Calculate the intersectional points using the curvature circle method.
(3) Calculate all the intersectional points during the process of (2) with n initial iteration values.
(4) Eliminate repeated points. As shown in Fig. 9, it is necessary to eliminate redundant intersectional points.

The iteration step size Δu is obtained in two situations as shown below.

(1) When there is only one intersection point on two curvature circles. As illustrated in Fig. 10, the curvature circle arc of second order confidence coefficient at the point P_1 and P_2 on the B-spline curves $\boldsymbol{p}$ and $\boldsymbol{q}$ is marked as black solid line. The points of minimum distance between two curvature circle arc are P_1' and P_2'. θ_1 and θ_2 are angle variations of the points on B-spline curves. The equation of iteration step size on B-spline curve $\boldsymbol{p}$ and $\boldsymbol{q}$ derived by angle variations are

shown as equation 6.

$$\Delta u = \frac{\theta_1}{k_2} = \frac{\left\| \boldsymbol{p}'(u_0) \right\|^2 \theta_1}{\left| \det(\boldsymbol{p}'(u_0), \boldsymbol{p}''(u_0)) \right|}$$
$$\Delta v = \frac{\theta_2}{k_2} = \frac{\left\| \boldsymbol{q}'(v_0) \right\|^2 \theta_2}{\left| \det(\boldsymbol{q}'(v_0), \boldsymbol{q}''(v_0)) \right|} \tag{6}$$

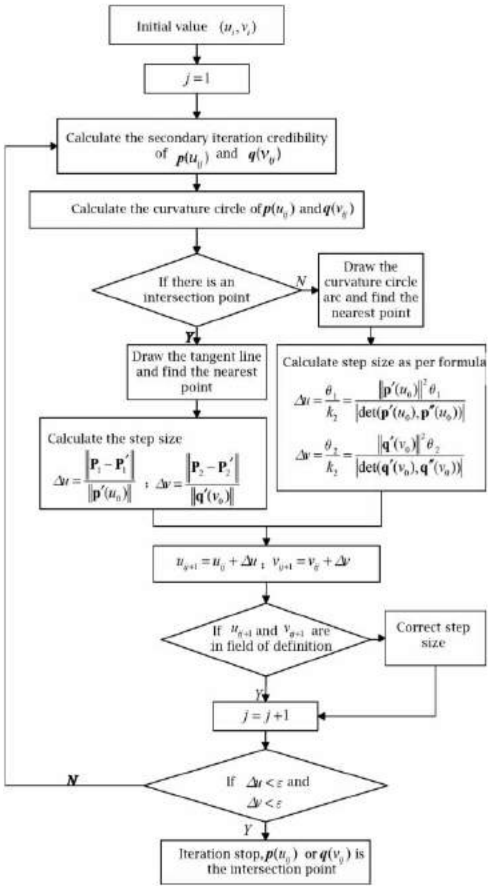
Initial value (u_i, v_i)
$j=1$
Calculate the secondary iteration credibility of $p(u_{ij})$ and $q(v_{ij})$
Calculate the curvature circle of $p(u_{ij})$ and $q(v_{ij})$
If there is an intersection point
N
Draw the curvature circle arc and find the nearest point
Y
Draw the tangent line and find the nearest point
Calculate step size as per formula
$\Delta u = \frac{\theta_1}{k_2} = \frac{\|p'(u_0)\|^2\theta_1}{|\det(p'(u_0), p''(u_0))|}$
$\Delta v = \frac{\theta_2}{k_2} = \frac{\|q'(v_0)\|^2\theta_2}{|\det(q'(v_0), q''(v_0))|}$
Calculate the step size
$\Delta u = \frac{\|P_1 - P_1'\|}{\|p'(u_0)\|}$; $\Delta v = \frac{\|P_2 - P_2'\|}{\|q'(v_0)\|}$
$u_{ij+1} = u_{ij} + \Delta u$; $v_{ij+1} = v_{ij} + \Delta v$
If u_{ij+1} and v_{ij+1} are in field of definition
Correct step size
Y
$j = j+1$
N
If $\Delta u < \varepsilon$ and $\Delta v < \varepsilon$
Y
Iteration stop, $p(u_{ij})$ or $q(v_{ij})$ is the intersection point

Fig. 8: Flow chart for solving the intersection problem with the curvature circle algorithm.

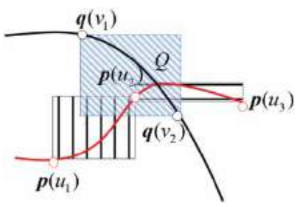

Fig. 9: Repeated intersection points.

(2) When there are two intersection points on two circles, as shown in Fig 11. If the intersection point is outside the tangent line, the endpoint is the closest point as point P_1'. If not, the closest point is the intersection point as point P_2'.

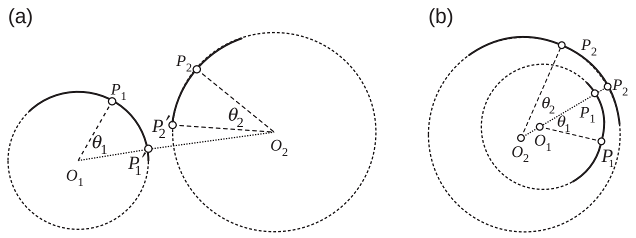

Fig. 10: There is only one point on two circles: (a) disjoint or circumscribed, (b) inscribe or contain.

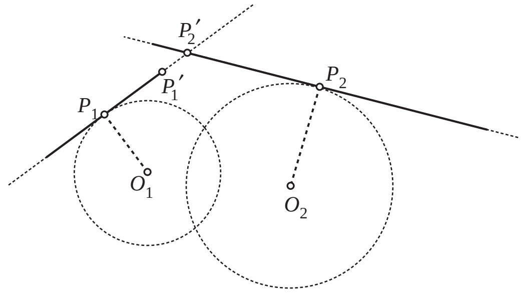

Fig. 11: Two intersection points on the curvature circles.

The iteration step size of B-spline curve $\boldsymbol{p}$ and $\boldsymbol{q}$ are derived as equ.7.

$$\Delta u = \frac{\|P_1 - P_1'\|}{\|\boldsymbol{p}'(u_0)\|} \quad \Delta v = \frac{\|P_2 - P_2'\|}{\|\boldsymbol{q}'(v_0)\|} \tag{7}$$

It should be noted that the value of u_{ij+1} (or v_{ij+1}), which $u_{ij+1} = u_{ij} + \Delta u$, may not in the define field of $u \in [0, 1]$. If not, the iteration step size should be modified as below.

$$\text{If } u_{ij+1} < 0,\ \Delta u = -\frac{u_{ij}}{2}; \quad \text{If } u_{ij+1} > 1, \Delta u = \frac{1 - u_{ij}}{2}.$$

2.3. An Example of Planar B-spline Curves Intersection

As shown in Fig. 12, we take two cubic B-spline curves $\boldsymbol{p}$ and $\boldsymbol{q}$ which intersect at (1, 2.5), (3, 4), and (8, 3) as an example. The corresponding analytical (u, v) values for intersections are (0.093, 0.071), (0.305, 0.365), (0.813, 0.820), which are the reference value of the following calculation.

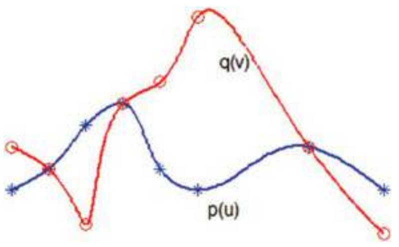

Fig. 12: The example of two B-spline curve p(u) and q(v).

Ini. Value	Result	Ini. Value	Result	Ini. Value	Result
(0.05, 0.05)	(0.093, 0.071)	(0.15, 0.05)	(0.093, 0.071)	(0.25, 0.35)	(0.305, 0.365)
(0.35, 0.35)	(0.305, 0.365)	(0.75, 0.85)	(0.813, 0.820)	(0.85, 0.85)	(0.813, 0.820)

Tab. 2: Intersection points after iteration of improved curvature circle algorithm.

After obtaining the initial iteration value by decomposition algorithm, the improved curvature circle algorithm is applied in the iteration. Each initial value is iterated several times to obtain the intersection points. The result is shown in Table 2.

Deleting repeated values, the intersection points of curve p(u) and q(v) are (0.093, 0.071), (0.305, 0.365), (0.813, 0.820), equal to the analytic values. The program running time is 0.013s by a computer with 2.8 GHz AMD Athlon quad core and 8G RAM. The intersectional algorithm of planar B-spline curves is shown to possess high accuracy automated initial iteration value, requiring little running time and demonstrating efficiency.

The intersection algorithm between B-spline curve and the straight line is the basis of locating key points on ship sections. As illustrated in Fig. 13, the straight line could be taken as a special B-spline curve. The initial iteration value could be obtained by max min box. And all intersection points could be obtained by curvature circle method.

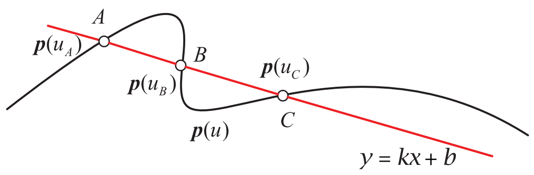

Fig. 13: Intersection between B-spline and straight line.

3. A LOCAL REFINEMENT ALGORITHM OF INTERSECTIONS

There are numerous intersection operations in the process of hull surface modeling calculation, including the intersection of hull surface and transverse sections for the body lines, and the intersection of hull surface and waterlines for the displacement and buoyancy center. The hull surfaces are classical free curved surfaces usually described by NURBS surfaces. The surfaces with which they intersect, such as transverse sections, water planes and cylindrical portholes, are described as implicit surfaces. Thus, the intersections are between NURBS surfaces and implicit surfaces.

Suppose a NURBS surface can be expressed as $x = g_1(u,v)$; $y = g_2(u,v)$; $z = g_3(u,v)$. Here the definition domain of g is Ω_g and g can be expressed as:

$$g(u,v) = \frac{\sum_{i=0}^{n_1}\sum_{j=0}^{n_2} \omega_{i,j} p_{i,j} B_{i,k_1}(u) B_{j,k_2}(v)}{\sum_{i=0}^{n_1}\sum_{j=0}^{n_2} \omega_{i,j} B_{i,k_1}(u) B_{j,k_2}(v)} \tag{8}$$

Where control vertices $p_{i,j}(i = 0, 1, \ldots n_1; j = 0, 1, \ldots n_2)$ make a topological rectangular array and form the control meshes. $\omega_{i,j}$ is the weight factor that contacts with vertices $p_{i,j}$. $B_{i,k_1}(u)$ and $B_{j,k_2}(v)$ represent standard B-spline with k_1 degrees in the u-direction and k_2 degrees in the v-direction, respectively.

The implicit surfaces are expressed as $f(x,y,z) = 0$. Here the definition domain of f is Ω_f. The equation of the intersection line can be simplified as $f(g_1, g_2, g_3) = h(u,v) = 0$. The process of solving the polynomial equation $h(u,v) = 0$ is exactly the same as the process of surface intersection.

3.1. A Whole Mesh Traversal Algorithm for a NURBS Surface and an Implicit Surface

The traversal algorithm is a common method for intersection operations between NURBS surfaces and implicit surfaces. In the method, the parameter field of a NURBS surface is divided into series of regularization cells. Each cell is judged for intersections with the contour line on an implicit surface. Then, all the intersections are collected and used for the object contour line fitting with a cubic B-spline curve. A flow chart of the method is presented in Fig. 14.

However, the limitations of the whole mesh traversing algorithm are obvious. Since all the mesh cells are traversed seriatim, the calculation consumes a large amount of time and resources. When the surface curvature changes drastically or the accuracy requirement of intersection lines is high, the mesh sizes are small and efficiency declines dramatically. An example of an intersection operation between a NURBS surface and a cylindrical surface is shown in Table 3. The NURBS surface is fitted by offsets and the whole mesh traversing algorithm is operated on a MATLAB platform with 2.8 GHz AMD Athod quad-cores and an 8G memory.

Table 3 indicates that the elapsed time for the object cell judgment calculation increases as the mesh size decreases. While the mesh density increases, the amount of calculation and the elapsed time tend to multiply. For a small mesh space, the judgment time for object cells takes up most of the time. To improve efficiency, a local refinement algorithm for the judgment calculation is proposed here.

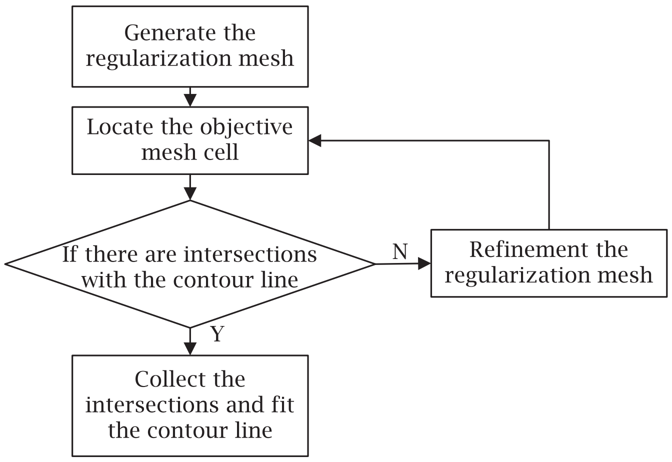
Generate the regularization mesh
Locate the objective mesh cell
If there are intersections with the contour line
N
Refinement the regularization mesh
Y
Collect the intersections and fit the contour line

Fig. 14: Flow chart of whole mesh traversal algorithm.

Mesh space	1/250	1/500	1/1000
Number of judgments	63001	251001	1002001
Elapsed T of judgment calculation	68.7	246.4	985.1
Elapsed T of intersection	77.3	261.4	1020
% of the elapsed T for judgment	89%	94%	97%

Tab. 3: Elapsed time comparison.

3.2. A Local Refinement Algorithm for a NURBS Surface and an Implicit Surface Intersection

3.2.1. *Regularization base mesh*

To begin with the local refinement algorithm, a sparse regularization mesh is generated on a NURBS surface parameter field. It is divided into n_u and n_v parts separately on the u and v dimensions. The four vertices of each cell are referred to as (u_i, v_j), (u_i, v_{j+1}), (u_{i+1}, v_{j+1}), and (u_{i+1}, v_j), corresponding with the four judgment functions $h_{i,j}$, $h_{i,j+1}$, $h_{i+1,j+1}$, and $h_{i+1,j}$ $(i = 0, 1, \dots, n_u - 1; j = 0, 1, \dots, n_v - 1)$. The base mesh algorithm is marked as Δ. If $h \le 0$, the vertex is marked "$-$", if not, "$+$". A judgment of no intersection for an object cell is made if four vertices have the same mark. Otherwise, the intersection is located by the Newton iteration method on lines with different marked vertices. After a rough whole traversal algorithm, several intersection points will be identified, as shown in Fig. 15. The h value of each vertex is stored into a matrix $h_0(i, j) = h((i-1)\Delta, (j-1)\Delta)$, $(i, j = 1, 2 \dots, 1/\Delta + 1)$.

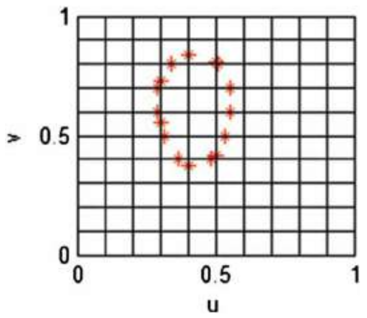

Fig. 15: Intersection points.

3.2.2. *Refinement range*

Taking the mesh space equal to Δ as an example, the method for determining the refinement range is:

Sort the intersection points and record the points as $a_1, a_2, \dots a_k, a_{k+1} \dots a_n (k = 1 \dots n)$. For the mesh element i which contains intersection points, each vertex is marked as shown in Fig. 16. A_i is set as the reference point for mesh element i, and its coordinate is (u_{Ai}, v_{Ai}), in which $u_{Ai} = floor(\min(u_k, u_{k+1})/\Delta)\Delta$ and $v_{Ai} = floor(\min(v_k, v_{k+1})/\Delta)\Delta$. In mesh element i, the coordinates of B_i, C_i, D_i can be obtained by the reference basis A_i. For example, $u_{Bi} = u_{Ai} + \Delta$; $v_{Bi} = v_{Ai}$.

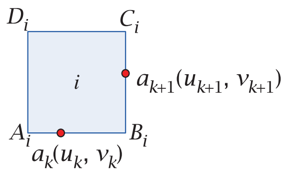

Fig. 16: An identifier in element refinement range.

Identify mesh element reference basis points $A_1, A_2, \dots A_i, A_{i+1} \dots A_m (i = 1 \dots m)$ according to order of intersection points, as shown in Fig. 17. Each mesh element containing an intersection point can be located by these reference basis points. The contour line must therefore lie in these mesh elements and the refinement range is made up of these mesh elements (see the blue frame surrounding area in Fig. 18). The flow chart for determining the refinement range is provided in Fig. 19.

3.2.3. *Mesh refinement*

Each element in the refinement range is divided evenly into four small elements. For example, the mesh before refinement is shown in Fig. 20(a) and the mesh after refinement is shown in Fig. 20(b).

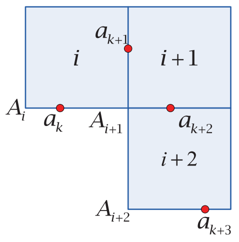

Fig. 17: Identify the vertex.

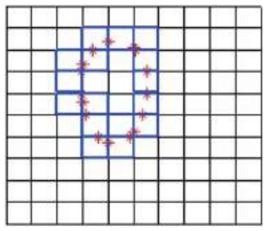

Fig. 18: Sketch map of refinement range.

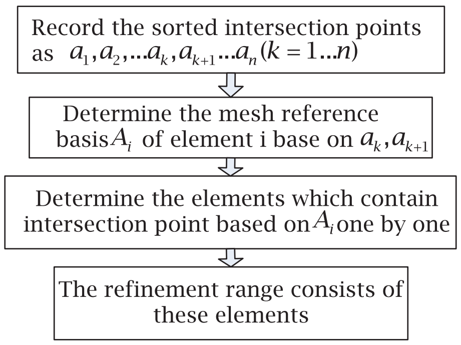
Record the sorted intersection points as $a_1, a_2, ... a_k, a_{k+1} ... a_n (k = 1...n)$
Determine the mesh reference basis A_i of element i base on a_k, a_{k+1}
Determine the elements which contain intersection point based on A_i one by one
The refinement range consists of these elements

Fig. 19: Flow chart for determining the refinement range.

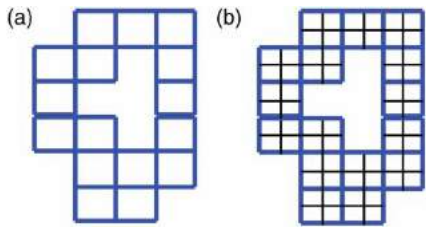

Fig. 20: Sketch map for mesh refinement: (a) before refinement and (b) after refinement.

3.2.4. Data processing

After each mesh refinement operation, the h value of new vertices must be updated. To avoid repeat calculations, some special data treatments are adopted.

- Identify the minimum rectangular domain that contains the mesh refinement area. Rectangular domain $D_1D_2D_3D_4$ is the minimum rectangular domain shown in Fig. 21. The fields surrounded by a blue wire frame in the figure are the mesh refinement areas.

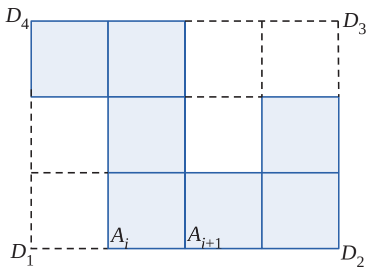

Fig. 21: Identifying the min rectangular domain.

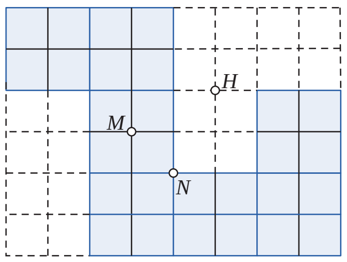

Fig. 22: Min. rectangular domain after refinement.

- Generate a judgment matrix. After the minimum rectangular domain is refined and each element is divided evenly into four small elements (Fig. 22), the judgment matrix is introduced to identify whether the vertices in the minimum rectangular domain are in the mesh refinement area or not. Suppose the judgment matrix after mesh refinement of the $\boldsymbol{K}$ times is $P_K(i,j)$, in which $i = 1, 2, \ldots 2^{K-1}(v_{D3} - v_{D1})/\Delta + 1$ and $j = 1, 2 \ldots 2^{K-1}(u_{D3} - u_{D1})/\Delta + 1$. $P_K(i,j)$ can be expressed as:

$$P_K(i,j) = \begin{cases} 0 & \textit{in the refinement area (such as M,N)} \\ 1 & \textit{else (such as H)} \end{cases}$$

- Update h values of the vertices in the minimum rectangular domain and store them into the matrix h_K. When $K = 1$, $h_1(i,j) = h_0(i + v_{D1}/\Delta, j + u_{D1}/\Delta)$ and when $K = 2, 3 \ldots$, the flow process for calculating h_K is used as shown in Fig. 23.

3.2.5. Intersection points between elements and the contour line after mesh refinement

Intersection points between mesh elements and the contour line must be freshly calculated after every mesh refinement. The details of the calculation are shown in Fig. 24.

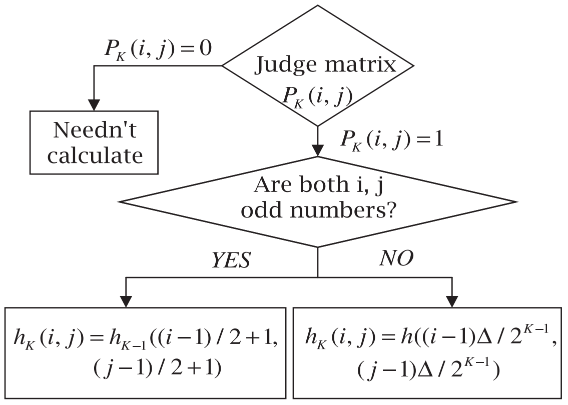
$P_K(i,j)=0$
Judge matrix $P_K(i,j)$
Needn't calculate
$P_K(i,j)=1$
Are both i, j odd numbers?
YES
NO
$h_K(i,j)=h_{K-1}((i-1)/2+1, (j-1)/2+1)$
$h_K(i,j)=h((i-1)\Delta/2^{K-1}, (j-1)\Delta/2^{K-1})$

Fig. 23: Flow chart for calculating h_K.

3.2.6. Accuracy control

The accuracy of calculation determines the quality of the intersection operation. There are n sample points extracted from the contour line and here, we suppose that $n = 200$; then, $h(u_i, v_i)(i = 1, 2, \ldots, n)$ of these sample points are calculated. Residual values are defined as $\xi = \sum_{i=1}^{n} |h(u_i, v_i)|/n$. Theoretically, the exact function value of each point on the contour line $h(u_i, v_i) = 0$, and the closer ξ is to zero, the higher accuracy of the intersection algorithm. Here, use $\xi < \varepsilon$ to control the intersection accuracy.

3.2.7. Flow of local refinement algorithm for NURBS surface and implicit surface

Details of the local refinement algorithm were introduced above. Here, we give the flow process of the local refinement algorithm for a NURBS surface and an implicit surface in Fig. 25.

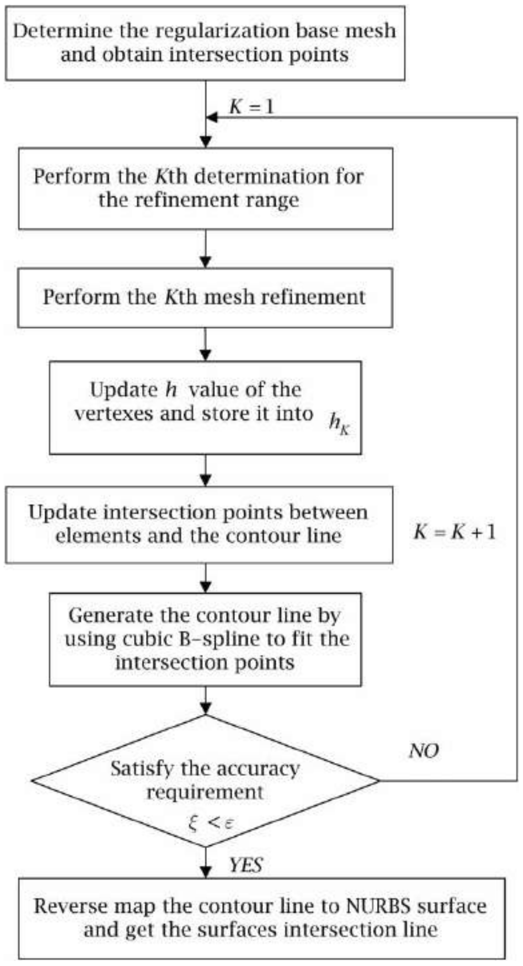
Determine the regularization base mesh and obtain intersection points
$K=1$
Perform the Kth determination for the refinement range
Perform the Kth mesh refinement
Update h value of the vertexes and store it into h_K
Update intersection points between elements and the contour line
$K=K+1$
Generate the contour line by using cubic B-spline to fit the intersection points
Satisfy the accuracy requirement $\xi<\varepsilon$
NO
YES
Reverse map the contour line to NURBS surface and get the surfaces intersection line

Fig. 25: Flow chart of local refinement algorithm.

3.3. Application of Local Refinement Algorithm for Hull Surface Intersection

A surface intersection program based on the local refinement algorithm is designed for a NURBS surface and an implicit surface intersection to hull surface

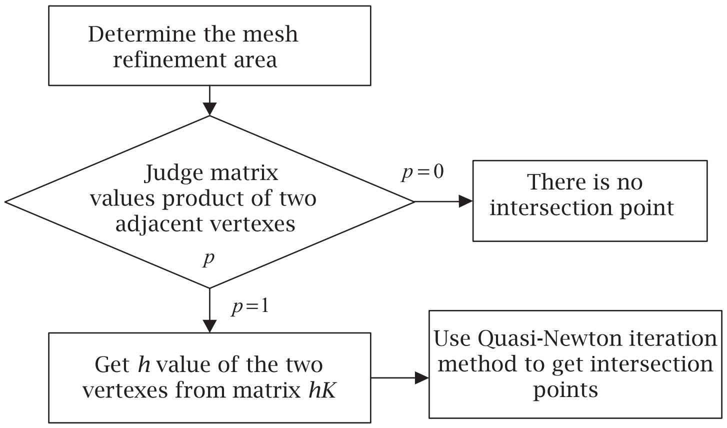
Determine the mesh refinement area
Judge matrix values product of two adjacent vertexes p
$p=0$
There is no intersection point
$p=1$
Get h value of the two vertexes from matrix hK
Use Quasi-Newton iteration method to get intersection points

Fig. 24: Flow chart for re-calculation of intersection points.

intersection. The waterline of a ship can be obtained successfully for any floating condition. Further, some advantages of the local refinement algorithm are discussed based on this hull surface intersection application.

3.3.1. Intersection between a hull surface and a wave surface

In any floating condition, the intersection line between a hull surface and the water surface is the foundation of hull 3D stability calculation [1]. As mentioned above, the hull surface is a typical free surface and it is often expressed by a NURBS surface. On the other hand, a water surface, either a static water surface or a simple wave surface, can be expressed by an implicit surface, and thus we can apply the proposed algorithm to obtain the waterline.

In our example we use the intersection of a hull surface and a wave surface. The details of the intersection process are shown in Fig. 26. Here, the hull surface is a part of a container ship stern and it is fitted by 201×401 offsets. The wave surface is a sinusoidal surface.

Firstly, the regularization base mesh and the intersection points are determined before the loop of local refinement algorithm. Secondly, each part of the loop is performed, which includes determination the refinement range, refining mesh, updating h value of the vertexes and store it into h_x, updating intersection points between elements and the contour line and generating the contour line by using cubic B-spline to fit the intersection points. Finally, the local refinement algorithm is stopped while the accuracy requirement is satisfied. And the surfaces intersection line is got by reverse mapping the contour line to NURBS surface.

3.4. Comparison of the Local Refinement Algorithm with the Whole Mesh Traversal Algorithm

In this study, we present two programs to obtain the waterline in any floating condition (see section 3.3.1.), based on the local refinement algorithm (LRA) and the whole mesh traversal algorithm (WMTA), respectively. Table 4 shows the intersection process using these two programs with $\Delta u = \Delta v = \Delta = 0.1$ It is easy to discern that LRA significantly reduces the number of calculations of the h value. The ratio of the decreased number of h value calculations to WMTA is even greater than 90%, when the minimum mesh space is less than 1/160.

Times of refinement	5	6	7
Min. mesh space	1/160	1/320	1/640
h value calculations with LRA	2937	5906	11722
h value calculations with WMTA	25921	103041	410881
Reduction in number of h value calculations	22984	97135	399159
Ratio of reduced part to WMTA	89%	94%	97%

Tab. 4: Comparison of LRA with WMTA on the number of h value calculations.

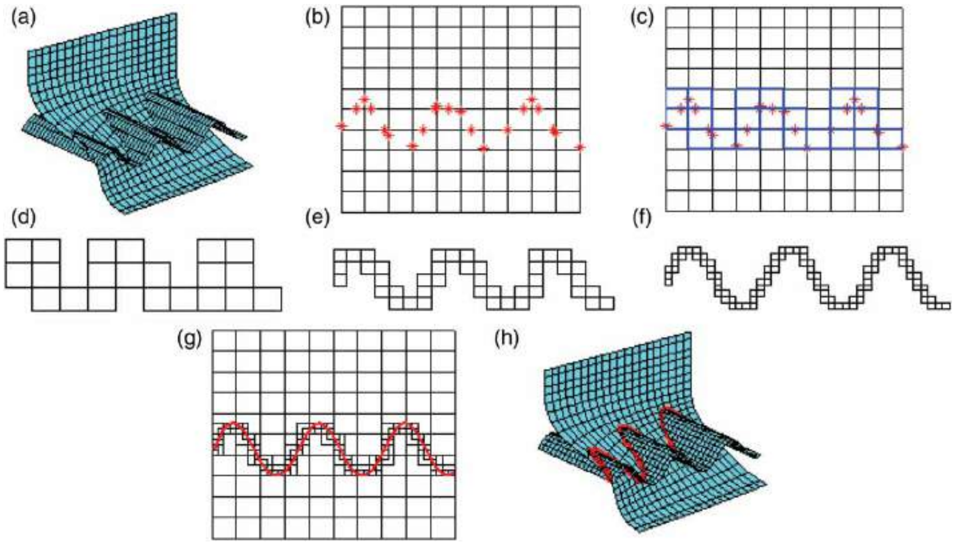

Fig. 26: Intersection of a hull surface and a wave surface: (a) hull surface and wave surface, (b) intersection points on the regularization base mesh, (c) identifying the refinement range, (d) refinement range of the first refinement ($K = 1$), (e) refinement range of secondary refinement of $K = 2$ and (f) $K = 3$, (g) the contour line on the refined mesh and (h) a waterline.

Fig. 27 shows that the elapsed time for the program using LRA is much less than that for the program using WMTA in the same minimum mesh space (the elapsed time percentage represents the ratio of elapsed time for LRA to that for WMTA.). When the minimum mesh space is less than 1/160, the elapsed time ratio of LRA to WMTA is only about 25%. Table 4 and Fig. 27 indicate that the LRA in this study greatly improves the efficiency of the surface intersection operation compared to the WMTA, due to the decreased number of calculations of *h* values, even though this method includes accuracy control and requires more judgments and cycle operations.

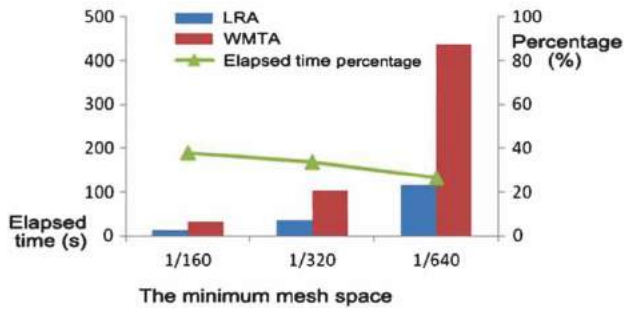

Fig. 27: Comparison LRA with WMTA on elapsed time.

4. APPLICATION OF FEM GENERATION ALGORITHM OF HULL SURFACE STRUCTURE

The automated generation of surface interpolation FE meshes plays an important role in the improvement of ship hull FE modeling efficiency and quality. A part of a container ship stern and midship structure is modeled here as an example of the automated mesh generation method in surface intersection.

4.1. Rough FE Mesh Generation

A rough FE mesh is generated by the intersect operation here. Longitudinals can be considered as the intersection lines of water planes and a hull surface, since they are parallel to the base plane in general. The intersection algorithm of a NURBS surface and a water plane is used to locate the longitudinal frame. The coordinates of key points are located rapidly by the intersection algorithm of B-spline curves and the LRA of NURBS surfaces and implicit surfaces. They are mapped on the transverse section, connected to form the rough FE meshes as shown in Fig. 28. The key points in the surface parameter field are shown in Fig. 29 and key points on the transverse section are shown in Fig. 30.

The key point coordinates are obtained and imported in the FEM code, and the hull surface and the plates of girders and frames are modeled by Quad4 (4-node shell unit). The stiffeners of bulkhead

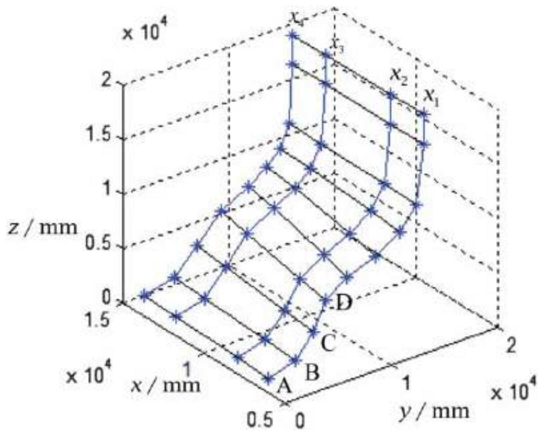

Fig. 28: Rough FE mesh.

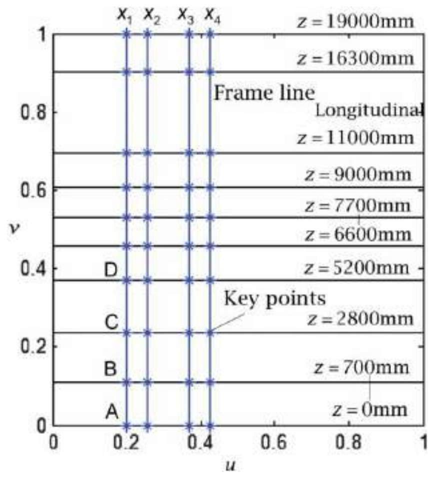

Fig. 29: Key points in the hull surface parameter field.

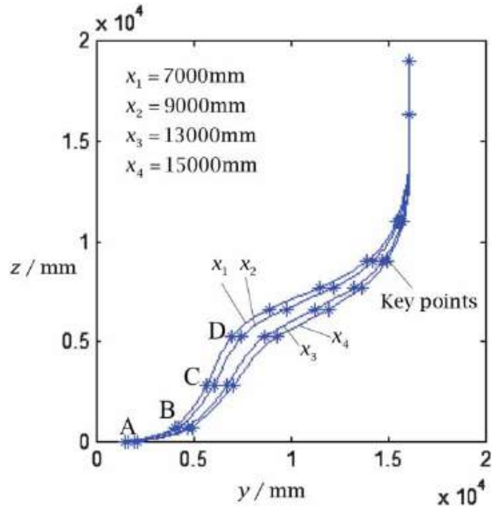

Fig. 30: The transverse sections of a ship hull surface.

and frame of web with height less than 300 mm are modeled by beam element Bar2 considering the offset. The FE model is shown in Fig. 31(b).

However, the quality of a rough FE mesh is always limited. The slenderness ratio of the red wireframe shown in Fig. 31(a) reaches 2.57, which is far beyond the ideal number of 1. Middle nodes need to be inserted into the high slenderness wireframe for further refinement and better quality.

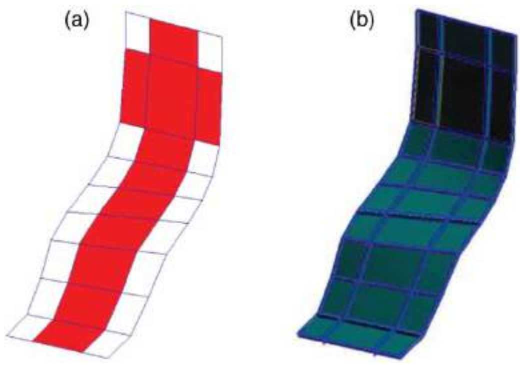

Fig. 31: FE model of rough hull surface mesh: (a) Rough FE mesh and (b) FE model.

4.2. Inserting Transverse Section and Water Plane

To refine the mesh, a series of middle points are inserted. Taking one longitudinal as an inserted object ($z = 19000$ mm), the parts after being divided by the middle point are as indicated in Fig. 32. In the same way, a frame line ($x = 7000$ mm) is taken as an object line to be inserted. After calculation, the objective inserting transverse plane is $x = 11000$ mm and the inserting water plane is $z = 13700$ mm.

4.3. Fine FE Mesh Generation

According to the transverse section and the water plane, the coordinates of the middle points are calculated by the intersection algorithm of B-spline curves and the LRA of NURBS and implicit surfaces. In Fig. 33, the middle points are represented by red points and new transverse section and water plane are represented by red lines.

By importing the point coordinates in the FEM code for MSC.PATRAN, the fine FE mesh of hull surface can be modeled as shown in Fig. 34. The border length of the unit consists of the key points and the nodes are closer to each other after insertion of the middle nodes. The aspect, warp, skew, taper and Jacobian ratio of the fine mesh tends to the ideal values. The mesh quality of the fine FE mesh after insertion is

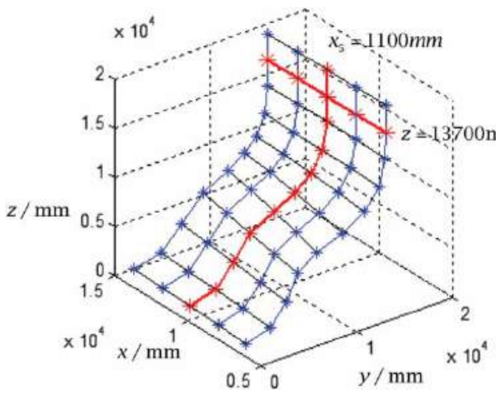

Fig. 33: The mesh consists of key points and inserted middle points.

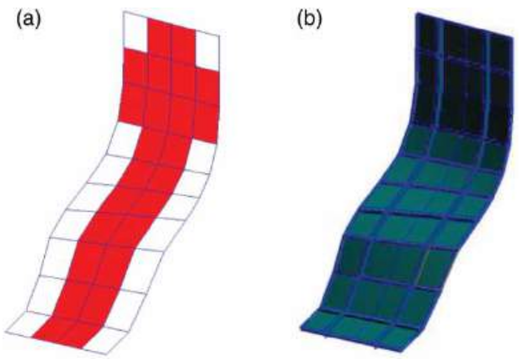

Fig. 34: The mesh consists of key points and middle Points: (a) fine FE mesh and (b) FE model.

much improved. The quality of fine and rough FE meshes is compared in Table 5.

Quadrilateral unit	Ideal value	Rough mesh	Finer mesh
Aspect	1	≤ 2.558	≤ 1.539
Warp	0	≤ 0.019	≤ 0.0145
Skew	90	≥ 71.025	≥ 71.025
Taper	0	≤ 0.105	≤ 0.059
Jacobian ratio	1	≤ 1.24	≤ 1.15

Tab. 5: Quality of fine and rough meshes.

4.4. Fine FE Mesh Generation

Although fine mesh can be used in FE calculation and analysis, refinement of the fine mesh is still

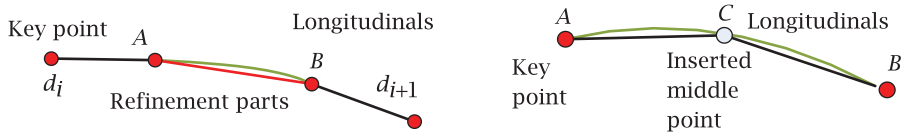

Fig. 32: Inserting middle points in a longitudinal line.

needed in the actual application. New nodes need to be inserted in each corresponding section of the water line and the transverse section line of the unit side. The method of inserting a new node and the model of fine FE mesh are shown in Figs. 35 and 36 respectively.

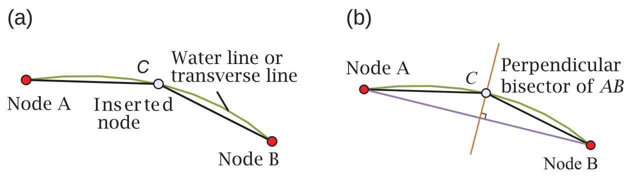

Fig. 35: Inserting new nodes (a) sketch of new node (b) location of new node C.

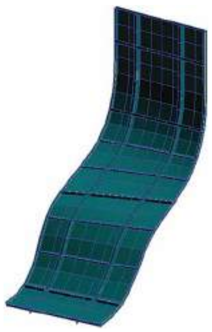

Fig. 36: Fine FE mesh of the ship hull surface structure.

4.5. Processing of the Stem and Stern Surfaces FE Mesh

The change in the longitudinal direction of the ship hull at stem and stern is quite significant, such that the key point numbers at different stations are not the same. Therefore, the triangle unit is introduced for transition in the process of modeling hull surface FE mesh for ship stem and stern. Unlike the midship surface structure, the planes of key points are not parallel to the water plane or the longitudinal section. The planes need to be formed by the transverse section structure of stem and stern. The coordinates of key points, which are still at the intersection of the hull surface and water plan, can be calculated rapidly by the LRA of a NURBS surface and a plane and the intersection algorithm of B-spline curves. The mesh generation process is shown in Fig. 37.

4.6. Data Exchange

The nodes of the coordinates are obtained and written in the TXT files and then imported into the FE code. According to coordinates of the nodes and the corresponding unit property of the hull structure drawing, a ship hull surface structure FE model can be built.

4.7. Modeling Example

Automated generation of hull surface structure FE mesh is applied in the modeling of a container ship midship and stern surface structure, as shown in Fig. 38 with $B = 1.0$, $T = 20.9$, $D = 0.0$, $P = 0.0$. All

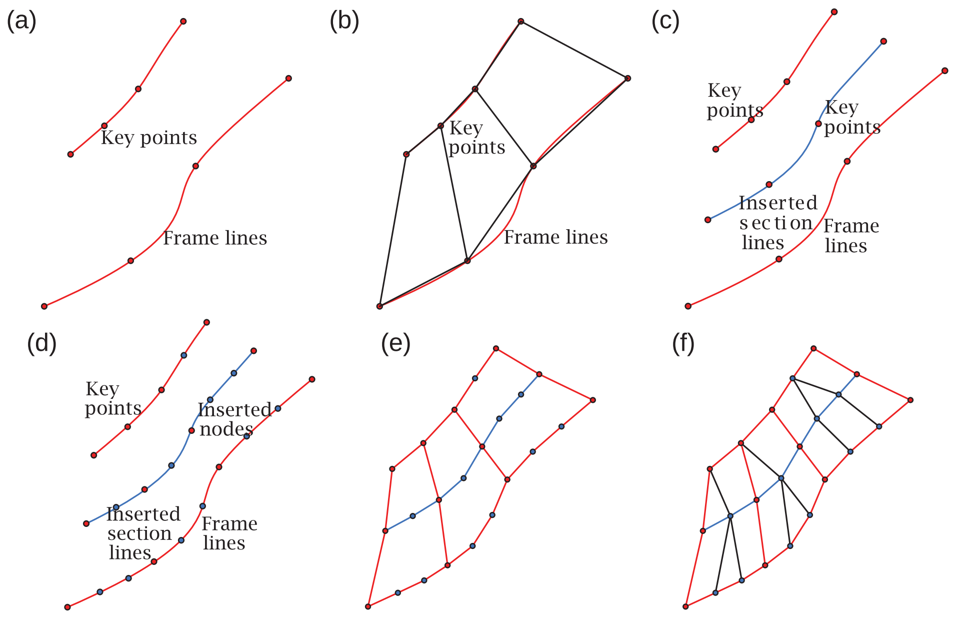

Fig. 37: Generation of fine FE meshes of stem and stern surface structures: (a) the key points on frame lines, (b) rough FE mesh, (c) insertion of transverse section, (d) insertion of new nodes, (e) connection of key points, and (f) fine FE mesh generation.

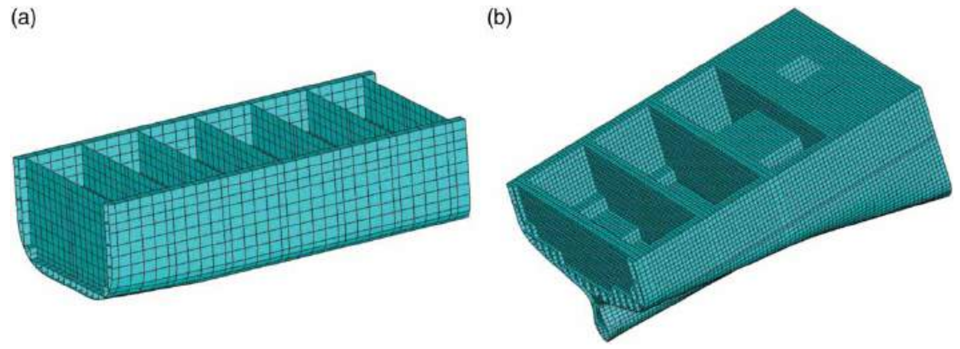

Fig. 38: Examples of FE models: (a) the container midship model and (b) the container stern model.

kinds of plates, shell structures, and the webs and bulkhead of strong frames, longitudinal girders, and girders and frame lines of planar bulkheads are modeled by 4-node shell unit Quad4. The stiffeners of bulkheads and frames of web height less than 300 mm are modeled by beam element Bar2 with consideration of offset.

5. CONCLUSION

The automated FE mesh generation algorithm of ship hull surface structure is challengeable in the long term. In this study, the designed FE mesh generation method of obtaining nodes by surface interpolation in this research creates meshes that are close to the real ship hull surface structure. The FE ship hull surface structure is meshed automatically with the generated nodes coordinates, nodes and unit numbers. The FE ship hull surface structure is modeled by importing the mesh information in the FEM code and giving the unit corresponding attribute. The intersection algorithm for curves and surfaces used largely in automated generation of ship hull surface FE mesh is studied intensively and improved. The improved intersection algorithm proposed in this study is applicable and feasible, and the LRA introduced in the intersection algorithm of a NURBS surface and an implicit surface improves the arithmetical efficiency. This work lays a foundation for parameterization of ship hull FE modeling. Meanwhile, the intersection algorithm of B-spline curves and the LRA of a NURBS surface and an implicit surface are widely applicable to CAD/CAE.

ACKNOWLEDGEMENTS

This study was partially supported by a guiding fund for the integration of industry, education and research of Guangdong province (Grant No.: 2012B091100132). Y. Tang is grateful for the research support of Industrial Transformation Training Centres (IC140100003) from the Australian Research Council.

REFERENCES

[1] Barn, H. R. E.; Kersey S. N.: A marching method for parametric surface/surface intersection, Computer Aided Geometric Design, 7, 1990, 257–280.

[2] Chen, H.; Bishop, J.; Delaunay triangulation for curved surface, In: Proceedings of the 6th International Meshing Roundtable, Utah, US, 1997, p. 115–127.

[3] Cuilliere, J. C.: An adaptive method for the automatic triangulation of 3D parametric surfaces, Computer Aided Design, 30(2), 1998, 139–149.

[4] George, P.-L; Seveno, E.: The advancing-front mesh generation method revisited, International Journal for Numerical Method Engineering, 37, 1994, 3605–3619.

[5] Goldman, R. N.: The method of resolvents: A technique for the implicitization, inversion, and intersection of non-planar, parametric, rational cubic curves, Computer Aided Geometric Design, 2(1), 1985, 237–255.

[6] Hoscheck J.; Lasser D.: Fundamentals of computer aided geometric design, A.K. Peters, Wellestey, MA: 1993.

[7] Hu, S.-M.; Sun, J.-G.; Jin, T.-G.; Wang, G.-Z.: Computing the parameters of points on NURBS curves and surfaces via moving affine frame method, Journal of Software, 11(1), 2000, 49–53.

[8] Jiang, Y.-M.: An intersection algorithm between two B-spline curves, Microelectronics & Computer, 12, 1991, 30–33.

[9] Jin, T.-G.; Wang, G.-Z.: MAF method for intersection calculation between parametric surfaces, Zhejiang University, Technical Report, No.: 8801057, 1988.

[10] Koparkar, P. A.; Mudur, S. P.: A new class of algorithms for processing of parametric curves, Computer Aided Design, 15(1), 1983, 41–45.

[11] Lau, T.-S.; Lo, S.-H.: Finite element mesh generation over analytical curved surfaces, Computer & Structures, 59(2), 1994, 301–309.

[12] Li, J.-F.; Shi, F.-Z.: A practical subdivision algorithm for intersection of B-spline curves, Journal of Machine Design, 20(11), 2003, 49–51.

[13] Lin, W.; Tang, Y.; Zhao, C.; Liu, X.; Zhu, G.; Jiang, F.: An algorithm for automatic two dimensional finite element mesh generation with line constraints, Computer-Aided Design, 43, 2011, 1803–1813.

[14] Lin, W.; Wang, C.; Zhu, G.; Tang, Y.; Zhao, C.; Liu, X.; Qiu, A.: Automatic recognition of hull transverse sections and quick finite element modelling for cargo hold longitudinal structures, Proceedings of the Institution of Mechanical Engineers - Part M: Journal of Engineering for the Maritime Environment, 2013 (Accepted). http://dx.doi.org/10.1177/1475090213507914

[15] Lu, C.-H.: Key technologies in preliminary ship design based on NURBS representation, Dalian University of Technology, 2005.

[16] Mei, Z.-Y.; Fan, Y.-Q.; Hu S.-G.: Finite element mesh generation of NURBS surfaces, Journal of Computer-aided Design & Computer Graphics, 9(1), 1997, 289–294.

[17] Rypl, D.; Krysl, P.: Triangulation of 3D surfaces, Engineering with Computers, 13(2), 1997, 87–98.

[18] Sederberg, T. W.; Anderson, D. C.; Goldman, R. N.: Implicitization, inversion, and intersection of planar rational cubic curves, Computer Vision, Graphics and Image Processing, 31(1), 1985, 89–102.

[19] Sederberg, T. W.; Nishita, T.: Curve intersection using Bezier clipping, Computer Aided Geometric Design, 22(9), 1990, 538–549.

[20] Song, H.-X; Han, Y.; Wu, C.-N.: An intersection algorithm of NURBS surface and plane based on the method of isoline, Digital Technology and Application, 7, 2011, 103–105.

[21] Xiong, Y.; Hu, Y.-J.; Zhao, J.-Y.: An algorithm of surface triangulation based on mapping and Delaunay method, Journal of Computer-aided Design & Computer Graphics, 14(1), 2002, 56–60.

[22] Yu, Z.-S.; Peng, Q.-S.; Ma, L.-Z.: An algorithm for intersection between parametric surface and implicit surface, Journal of Computer Aided Design & Computer Graphics, 11(2), 1999, 97–99.

[23] Zhang, M.-X.; Ji, Z.-S.; Lin, Y.; Implementation and application of intersection between NURBS-based surface and implicit surface, Ship Building of China, 43(3), 2002, 94–98.

[24] Zhang, S.-H.; Huang, Z.-Y.: Intersections of planar parametric curves based on curvature circle, Chinese Journal of Computers, 30(9), 2007, 1588–1593.

[25] Zheng, Y.; Lewis, R. W.; Gethin, D. T.: Three-dimensional unstructured mesh generation: Part I, Foundational aspects of triangulation and point creation, Computer Methods in Applied Mechanics and Engineering, 134(3/4), 1996, 249–268.

[26] Zheng, Y.; Lewis, R. W.; Gethin D. T.: Three-dimensional unstructured mesh generation: Part II, Foundational aspects of triangulation and point creation, Computer Methods in Applied Mechanics and Engineering, 134(3/4), 1996, 269–284.