Journal of *Marine Science and Engineering*

MDPI

Article

Fast High-Precision Bisection Feedback Search Algorithm and Its Application in Flattening the NURBS Curve

Kaige Zhu [1,2], Guoyou Shi [1,2,*], Jiao Liu [1,2] and Jiahui Shi [1,2]

[1] Navigation College, Dalian Maritime University, Dalian 116026, China
[2] Key Laboratory of Navigation Safety Guarantee of Liaoning Province, Dalian 116026, China
* Correspondence: sgydmu@dlmu.edu.cn

Abstract: It is important to accurately calculate flattening points when reconstructing ship hull models, which require fast and high-precision computation. However, some search algorithms, such as the bisection method, iterate near the optimal value too many times before converging in high-precision computation. The paper proposes a fast high-precision bisection feedback search (FHP-BFS) algorithm to solve the problem. In the FHP-BFS algorithm, the Newton–Raphson (NR) method is adopted to accelerate the convergence speed by considering the iteration characteristics of subintervals. Furthermore, a new feedback mechanism is proposed to control the feedback directions. In addition, an acceleration algorithm, called the interval reformation method, is used to guide the FHP-BFS algorithm for fast convergence. Finally, the flattening algorithm is improved by the FHP-BFS algorithm. In the comparative experiments, the practical efficacy of the FHP-BFS algorithm is first demonstrated, and then the optimal range of the threshold precision is determined. Next the FHP-BFS algorithm is compared to the best existing algorithms. Finally, the performance of the improved flattening algorithm is verified. The experiments demonstrate that the FHP-BFS algorithm has optimal performance among the compared algorithms, and it has an improved computation efficiency while maintaining robustness. The improved flattening algorithm reduces the computation time, ensures smoothness and meets practical engineering requirements.

Keywords: FHP-BFS algorithm; flattening algorithm; inversion; high-precision threshold; computation efficiency

Citation: Zhu, K.; Shi, G.; Liu, J.; Shi, J. Fast High-Precision Bisection Feedback Search Algorithm and Its Application in Flattening the NURBS Curve. *J. Mar. Sci. Eng.* **2022**, *10*, 1851. https://doi.org/10.3390/jmse10121851

Academic Editors: Zaojian Zou and Weilin Luo

Received: 29 October 2022
Accepted: 28 November 2022
Published: 1 December 2022

1. Introduction

1.1. Modeling and Deformation of the Ship Hull

Ship hull reconstruction is a reverse engineering application that transforms a physical model into a digital non-uniform rational B-spline (NURBS) model through computer-aided design technology [1]. In applying ship-damaged stability information, the reconstructed ship NURBS model can significantly improve computational accuracy and efficiency compared with other algorithms [2]. For example, in the calculation of water plane elements of a damaged ship hull, the triangular grid method can be used to calculate the surface intersection between the ship hull and the water plane, and the intersection line is achieved by subdividing ship stations, which can hardly obtain high-precision solutions [3]. However, the intersection line of the ship NURBS model can be directly obtained by the Newton–Raphson (NR) iteration approach [4,5] or the recursive subdivision approach [6–9], which can ensure high computational precision and efficiency. Moreover, the computation time of the residual stability of damaged ships directly determines the remaining time for rescue, and the computation precision affects the rescue measures. Therefore, a fast high-precision ship NURBS model is of practical importance for real-time calculations.

Figure 1 shows the processes of defining the ship hull NURBS model [10]. The interpolation operation is first executed to define basic curves, which describe the ship's characteristics and mainly refer to the centerline profile curve, bottom tangent curve, side

tangent curve, midship section curve, etc. Second, the cross-sectional curves along the ship length or waterline, which can be obtained through the offset or the waterline data [11,12], are defined in each frame. Third, the space curves represent complicated parts, such as the stern and stem. Fourth, the deck plane and superstructure are determined after the fairing process. Finally, the model is obtained by the ship hull fairing operation [13,14]. In these processes, the interpolating error of NURBS curves can be reduced through a deformation operation, which represents shape-preserving or geometrically constrained reconstruction using the underlying information about the reconstructed shape. The deformation operation includes flattening, bending, warping, and twisting. Flattening is the most widely used operation for the deformation of ship NURBS models, and flattening makes the bottoms or sides of the ship hull expressed as straight lines or planes [15]. However, if the flattening precision is insufficient, the calculation based on the reconstructed models will have many errors, such as ship inlet volume errors and deviation errors of the gravity center and buoyancy center. Therefore, the computational efficiency and precision of the flattening operation are particularly significant for providing quick, high-precision inversion solutions.

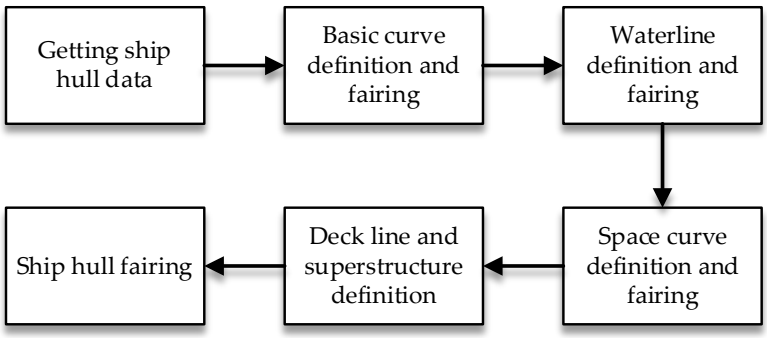


Figure 1. Processes of defining the ship hull NURBS model.

1.2. Inversion Algorithms of NURBS Curves

The inversion algorithm of the NURBS curve is divided into the compound and direct algorithms. The compound algorithms first calculate the rough solution by a method as the initial value, and other methods calculate the exact value based on the initial value. In contrast, the direct algorithms only use one method to obtain the exact value. The minimum Euclidean distance between the target and test points is usually used as the convergence criterion for calculating rough values. Ref. [16] first proposed an algorithm to achieve rough values; it calculates the maximum and minimum values of the distance by projecting values onto the boundary of the basic geometry, and the exclusion parts are the points with higher distance values than the currently obtained minimum distance. Subsequently, ref. [17] proposed an exclusion criterion based on a tangent cone. Ref. [18] divided the NURBS curves or surfaces into Bezier sub curves or surface slices. A rough solution was determined by examining the distribution between test points and control points or control grids. Ref. [19] proposed an exclusion criterion based on the Voronoi cell test. Refs. [20,21] proposed a circular or spherical clipping method to calculate the minimum distance between points and clamped B-spline surfaces. Ref. [22] improved the exclusion criterion of [20] by replacing shear circles with axis-aligned lines; subsequently, ref. [23] proposed a culling method to remove redundant curves based on the approach in [22]. However, the elimination rate is lower than that of [20,21]. Ref. [24] proposed a curvature information method to calculate the minimum distance between points and parametric curves or surfaces; nevertheless, the computation time is long due to considering second-order derivatives.

Exact values are usually calculated by algorithms with high convergence speeds, such as the NR method [15,25], the gradient descent method [26], the conjugate direction method [27] and the direct method [28]. Among these methods, the NR method is the most commonly

used due to its quadratic convergence rate; moreover, a certain reliable initial value should be provided due to the local convergence. To optimize this problem, various strategies were proposed by [26,29], such as the Levenberg–Marquardt and trust domain methods. Based on these methods, the NR methods were applied to global strategy optimization in Riemannian settings [30–33]. The NR methods with these global strategies are also called damped NR methods, where the detailed parameter settings of the linear search problem are studied in [26,34,35]. Among the various strategies, the linear search and value function is effective. A direct method called the bisection feedback search (BFS) algorithm was proposed to obtain the global optimum solution [28], which is based on the bisection method and incorporates feedback processes to obtain the ability to jump out of local optima. Additionally, the interval reformation (IR) method is proposed to provide a search direction for the BFS algorithm. Hence, the IR-BFS algorithm has an excellent single-loop capability and converges faster when the threshold precision is lower than 10^{-4}.

1.3. Problems with the IR-BFS Inversion Algorithm

The target interval of inversion in the IR-BFS algorithm is quickly locked through the IR method, which improves computation efficiency compared with other algorithms. However, if the high-precision threshold is set, for example, to 10^{-8}, the computation speed begins to slow down; that is, the model experiences continuous iteration without convergence in the neighborhood of the optimal solution, which consumes too much computation time, and this problem is called "precision refinement" in the paper.

Figure 2 shows the inversion process of the flattening point p_s based on the IR-BFS algorithm. There are two convergence processes, "Process 1" and "Process 2". A feedback process occurs, that is, the transition from "Process 1" to "Process 2". In addition, "Process A" and "Process B" are the processes of "precision refinement". In addition, the criteria, which are set to $|u_e - u_s| \leq 10^{-3}$, are used to judge whether the iteration enters the "precision refinement" process. Figure 2 shows a "precision refinement" phenomenon in each convergence process.

Taking "Process B" in Figure 2 as an example, when $t = 0.011$, the iteration enters the feedback process; when $t = 0.013$, the iteration enters "Process B"; when $t = 0.021$, the solution meets the convergence threshold, and the precision is 5.54×10^{-9}, which is the global optimum. In the iteration process, the time consumption of "Process 2" is $0.01s$, while the time consumption of "Process B" is $0.008s$, which is 80% of the time consumed by "Process 2". Therefore, much computation time is consumed in the "precision refinement" process of the IR-BFS algorithm, although the algorithm can obtain the global optimal solution with a high-precision threshold.

1.4. Research Objectives and Structure

This paper studies how to improve the computational efficiency of the inversion algorithm while ensuring computational precision, which is used to improve the computational speed of the flattening algorithm. The fast high-precision bisection feedback search (FHP-BFS) algorithm, which is proposed to solve the problem of "precision refinement", uses global convergence and the fast single iteration ability of the BFS algorithm to obtain rough values; then the NR method, which has the advantage of quadratic convergence speed, is applied to obtain the exact solution. Moreover, an appropriate threshold precision value is set for the rough value to provide a good initial value for the NR method; the optimal range of the output threshold precision of the FHP-BFS algorithm is determined experimentally to improve its scalability and to more easily apply it to practical operations. Finally, the fast high-precision inversion process of the FHP-BFS algorithm is provided for the flattening algorithm to solve the problem of long computation time. In addition, all the experiments were performed on a Windows 10 laptop with 32 gigabytes of RAM and a Core I7 processor using the Python programming language and the PyCharm IDE.

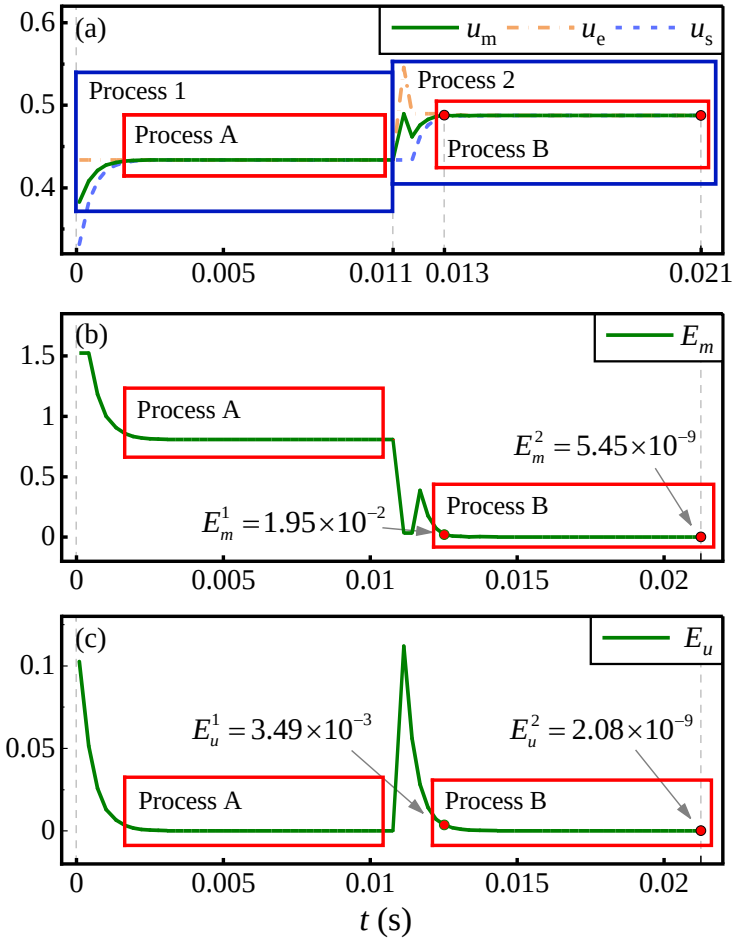


Figure 2. Inversion process of parametric value u of flattening point p_s based on the interval reformation and bisection feedback search (IR-BFS) algorithm. (**a**) The parametric value u in inversion process; (**b**) The distance error between p_m and p_s; (**c**) The distance error between u_m and u_s, where p_m denotes the curve point according to u_m; "Process 1" and "Process 2" are the processes of convergence; "Process A" and "Process B" are the processes of "precision refinement"; u_s, u_e and u_m denote the values of the left endpoint, right endpoint, and the middle point of the iteration interval, respectively; E_m denotes the distance error; and E_u denotes the range of the iterating interval.

The main contributions of this paper are as follows: (i) The FHP-BFS algorithm is proposed, and the algorithm has global convergence in NURBS curve inversion, which increases the computation efficiency while ensuring the computation precision. The higher the precision is, the greater the computational efficiency compared with other algorithms. (ii) The optimal range of the threshold parameters of the FHP-BFS algorithm is determined, which makes the algorithm easier to apply to practical engineering problems. (iii) The flattening algorithm is improved to enhance the computation efficiency in high-precision real-time modeling.

The subsequent sections are organized as follows. Section 2 introduces the mathematical background of the relevant algorithms. Section 3 introduces the framework of the proposed algorithms. Section 4 designs comparative experiments to verify the effectiveness of the proposed algorithm. Sections 5 and 6 present the discussion and conclusion.

2. Mathematical Background

2.1. NURBS Curve

NURBS is a unique mathematical method used to define the geometry of industrial products in the data exchange standard [36]. In the ship hull surface modeling task, the

NURBS method creates more realistic and vivid modeling results [37]. Generally, the expression of the parametric form of a pth-degree NURBS curve is as follows:

$$C(u) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,p}(u)}{\sum_{i=0}^{n} w_i N_{i,p}(u)} \quad (a \le u \le b), \tag{1}$$

where $\{w_i\}$ denote the weights corresponding to control points; $\{P_i\}$ denote the control points; u denotes the parametric value and $\{N_{i,p}(u)\}$ are the pth-degree B-spline basis functions defined by the knot vector U [15]. The knot vector U can be defined by Equation (2):

$$U = \left\{ \underbrace{a, \cdots, a}_{p+1}, u_{p+1}, \cdots, u_{m-p-1}, \underbrace{b, \cdots, b}_{p+1} \right\}, \tag{2}$$

and the recursion formula for $N_{i,p}(u)$ is defined by Equation (3) [38]:

$$\begin{cases} N_{i,0}(u) = \begin{cases} 1, & if \quad u_i \le u \le u_{i+1} \\ 0, & else \end{cases} \\ N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u) \\ defined \quad \frac{0}{0} = 0 \end{cases}, \tag{3}$$

The definition of Equation (3) is the most efficient form for computer implementation. In Figure 3, the NURBS curve is interpolated to the feature points of the Archimedes curve, and the values of the basis function are calculated by Equation (3). The control polygon in Figure 3 denotes a polygon formed by connecting the control points in order. In reconstructing a ship hull, the waterplane or cross-section NURBS curves are typically obtained by the interpolation algorithm.

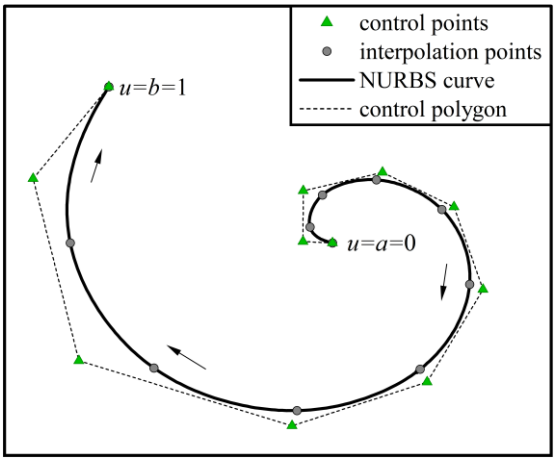


Figure 3. Non-uniform rational B-spline (NURBS) curve interpolated by Archimedes feature points, where the arrows indicate the direction in which the curve points change with the order of knot values.

In addition, the rational basis function $R_{i,p}(u)$ can be introduced and defined as follows:

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \quad (a \le u \le b), \tag{4}$$

therefore, the NURBS curve of Equation (1) can also be defined by Equation (5):

$$C(u) = \sum_{i=0}^{n} P_i R_{i,p}(u) \quad (a \le u \le b). \tag{5}$$

2.2. IR-BFS Inversion Algorithm

The inversion of the NURBS curve is the process of calculating the parametric values according to the inversion points. As shown in Figure 4, P_s and P_e are inversion points randomly selected on the NURBS curve. The curve segment between them is defined within the range of the knot vector U. The inversion process calculates the corresponding knot values u_s and u_e. In addition, the chain dotted line in Figure 4 represents the curvature of the curve point, which is used to measure the bending degree of the curve.

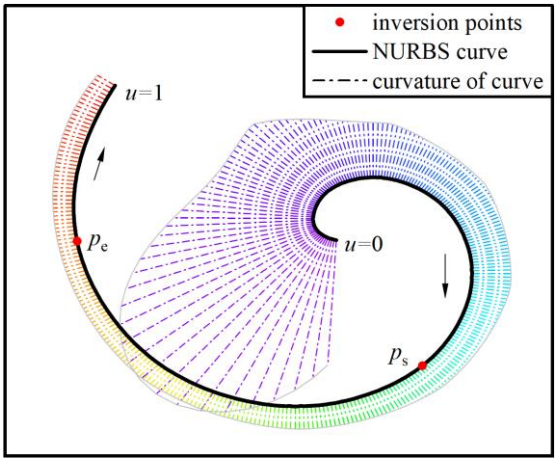


Figure 4. Process of selecting inversion points on the Archimedes non-uniform rational B-spline (NURBS) curve, where the arrows indicate the direction in which the curve points change with the order of knot values.

The IR-BFS algorithm was proposed by us to solve the low computational efficiency in the inversion of NURBS curves [28]. Figure 5 shows the flow chart of the IR-BFS algorithm, in which some operations, such as the bisection table, the inverting interval, the outputting threshold, and the feedback operations, are described. Among them, the most important are the two feedback operations, which guarantee the ability of the algorithm to jump out of local optima. The feedback operations are the feedback to the current iteration subinterval and the next subinterval. In the feedback to the current subinterval, the current iteration interval and the parametric value need to be updated according to the inverting interval criterion for calculating the next iteration. The bisection table needs to be reset in the feedback to the next subinterval, and the iteration interval and parametric values need to be updated according to the feedback criteria. The bisection table records the bisection selection of each iteration of the current interval; the inverting interval operation inversely selects the interval of the target record in the bisection table to update the current iteration parameters; and the outputting thresholds, α and β, are crucial parameters that affect the convergence speed and accuracy of the algorithm.

In addition, two major processes, the IR method and the BFS algorithm are designed in series. The IR method is responsible for reducing the search range of the BFS algorithm, and the BFS algorithm searches the target solution in ascending order in the subinterval provided by the IR method. More detailed descriptions of the parameter settings can be found in [28].

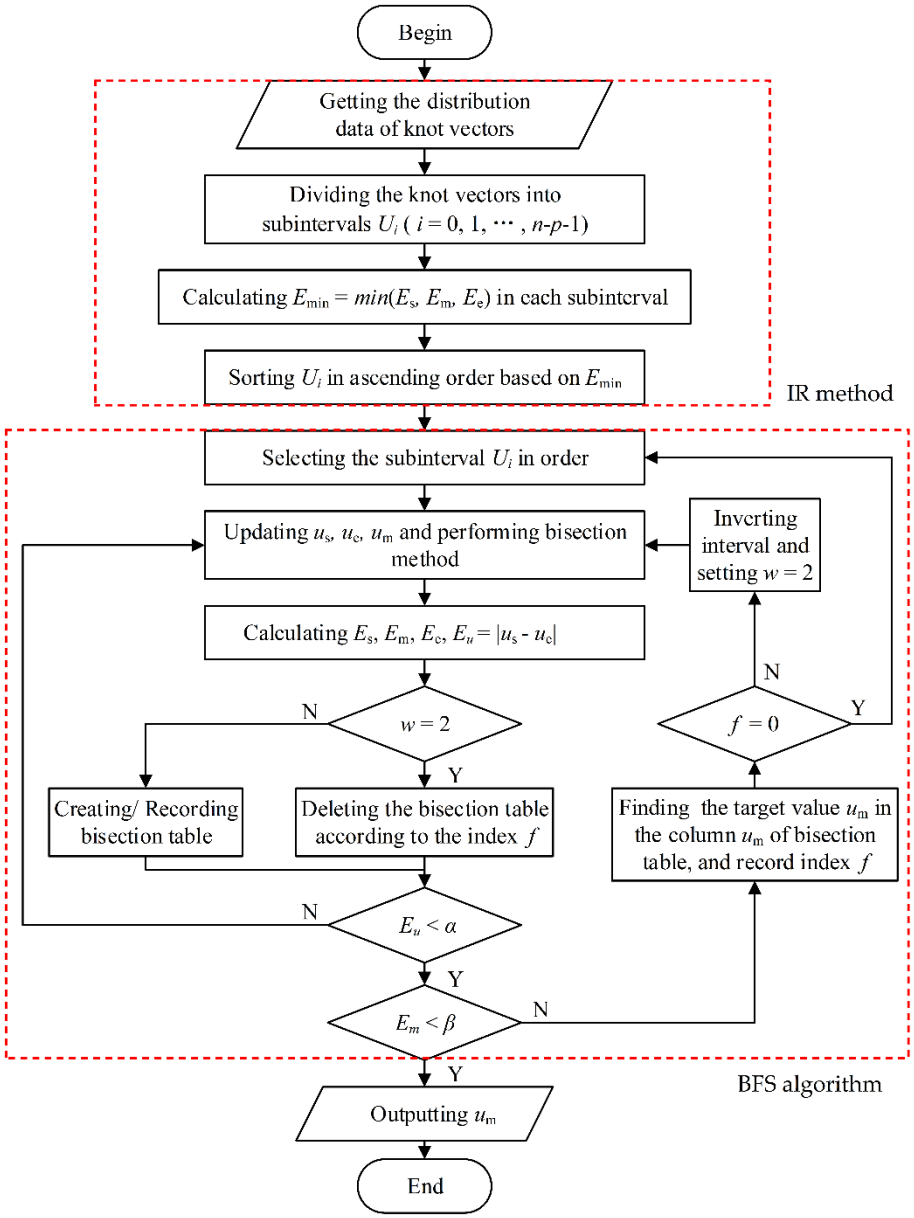


Figure 5. Overall design of the interval reformation and bisection feedback search (IR-BFS) algorithm, where w is a state parameter; f is the index of the target row in the bisection table; E is the distance error and α and β denote the knot error threshold and the distance error threshold, respectively.

2.3. Flattening Algorithm of the NURBS Curve

The flattening algorithm can quickly produce straight line segments or plane regions on NURBS curves or surfaces [39,40]. In reconstructing a ship hull, the deformations of flattening operations are usually carried out on basic models. Figure 6 shows the flattening algorithm based on the IR-BFS algorithm. The flattening algorithm first interpolates the input data to NURBS curves. Next, the parametric values are obtained by the inversion algorithm. Then, knot refinement is performed to obtain more control points in the interval affected by the flattening parametric values. After that, the control points are projected onto the flattening line based on the projection criterion and updated again. Finally, the flattening operation is completed if the number of control points successfully projected on the flattening line segment reaches at least $p + 1$. In addition, if the number of successfully projected points is less than $p + 1$, the knot refinement operation is performed again, and the number of inserted knots is increased by 1.

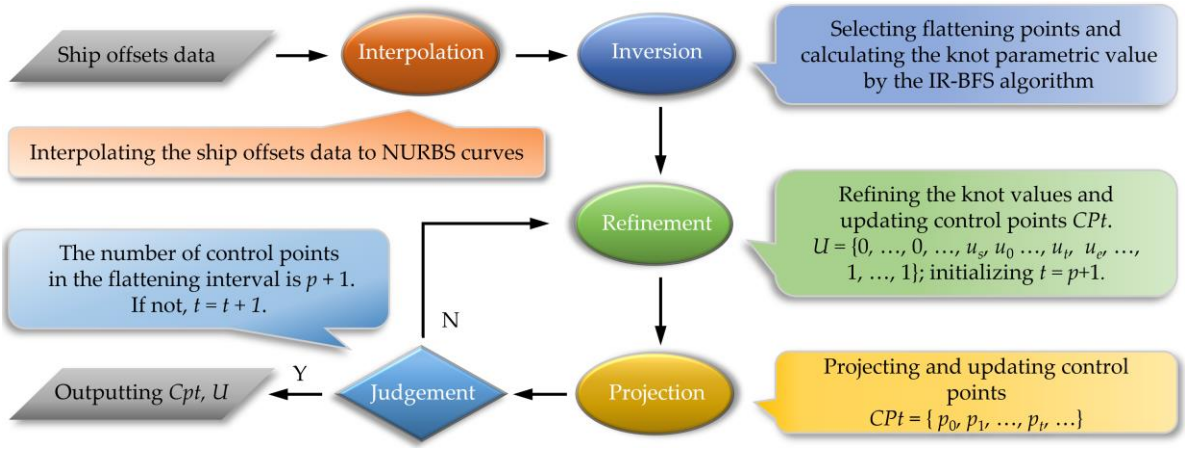


Figure 6. Flattening algorithm of the non-uniform rational B-spline (NURBS) curve based on the interval reformation and bisection feedback search (IR-BFS) algorithm.

The flattening effect is analyzed by the curvature change in the NURBS curve before and after the flattening operation. The curvature of the NURBS curve is defined by Equation (6):

$$K(u) = \frac{|C'(u) \times C''(u)|}{|C'(u)|^3}, \tag{6}$$

where $C'(u)$ and $C''(u)$ denote the first and second derivatives of the curve with respect to parameter u, respectively. If the curvature near the flattened points is gradually reduced to 0, then the segment of the flattening curve becomes a straight line, indicating a good flattening effect. Otherwise, it indicates that the effect is poor. Figure 7 shows the flattened NURBS curves with flattening points p_s and p_e. By comparing the curve before flattening in Figure 4, the curvature near the flattening parametric values of points p_s and p_e gradually reduced to zero, indicating a good flattening effect.

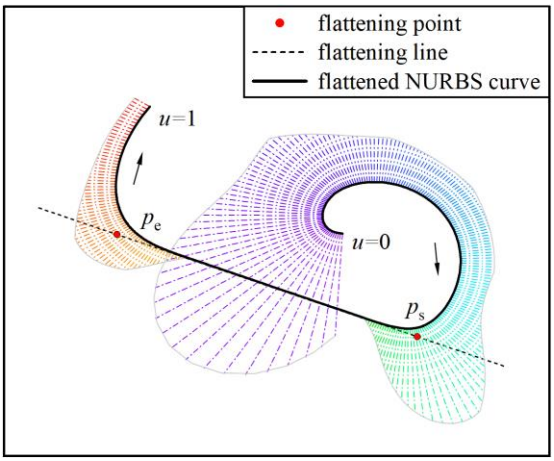


Figure 7. Flattened Archimedes non-uniform rational B-spline (NURBS) curve with flattening points p_s and p_e, where the arrows indicate the direction in which the curve points change with the order of knot values.

3. Framework of the Proposed Methodology

3.1. Overall Design of the FHP-BFS Algorithm

Figure 8 shows the overall design of the FHP-BFS algorithm. Compared with the IR-BFS algorithm in Figure 5, the NR method is integrated into the IR method and the BFS algorithm. Simultaneously, the loop mechanism of the algorithm is changed, and the

feedback mechanism and the convergence criteria for the NR algorithm are added to the FHP-BFS algorithm.

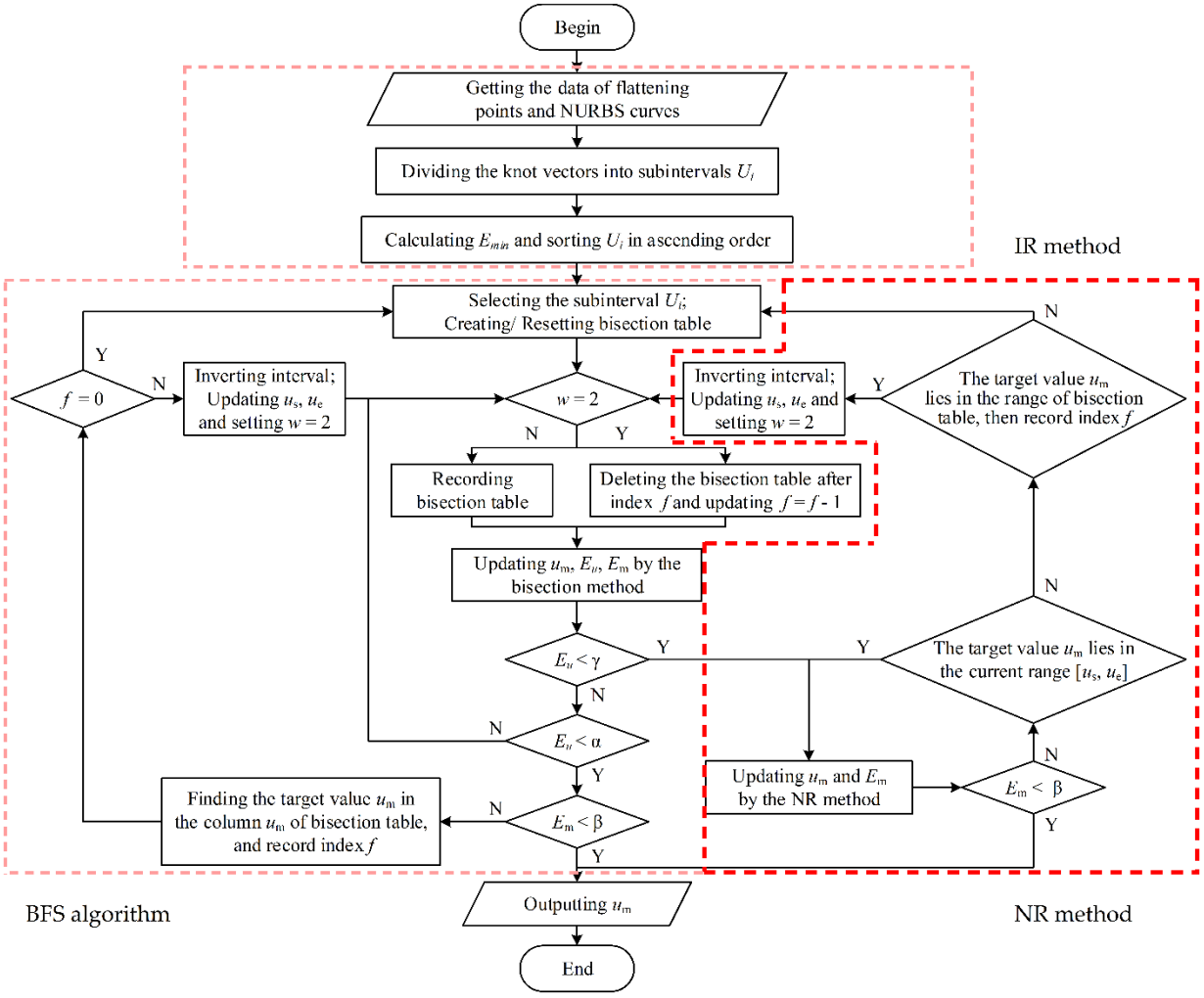


Figure 8. Overall design of the fast high-precision bisection feedback search (FHP-BFS) algorithm.

The loop mechanism of the FHP-BFS algorithm first reduces the iteration interval of possible solutions. Then, the BFS algorithm is used to provide the ability to quickly locate the range of the convergence results and realize the ability to jump out of local minimum values. Finally, the NR method is used to refine the precision of the convergence result. The solution that meets the threshold is achieved after several iterations and feedback loops.

The condition for using the NR algorithm in the FHP-BFS algorithm is judged by the length of iteration interval E_u with threshold γ. When $E_u < \gamma$, the NR method with the initial value of u_m is executed for convergence. Figure 2 shows that the "precision refinement" process begins when $E_u = 3.49 \times 10^{-3}$, and u_m is an excellent and stable initial value for the NR method. Therefore, it is recommended that γ be set to 10^{-3}. Furthermore, through our practice experiments, the recommended threshold $\gamma = 10^{-3}$ not only improves the computational efficiency of the FHP-BFS algorithm but also ensures stability.

The feedback object should be first clarified for the feedback criterion of the NR method in the FHP-BFS algorithm, that is, the feedback is provided to the current subinterval or the next subinterval. In the iteration of the IR method, if the target solution is not in the current iteration interval $[u_s, u_e]$, the bisection table is searched to judge whether the target parameter u_m is in the recording interval. Assume that the record is $[u_s, u_m, u_e, w]$; if $w = 0$ or $w = 1$, which means the interval $[u_s, u_m]$ or $[u_m, u_e]$ was chosen when recording, then the judgment of whether u_m is in the interval $[u_m, u_e]$ or $[u_s, u_m]$ is performed. In addition,

if $w = 2$, which means the interval had been fed back, then the record will be skipped. If u_m is in the recording interval, then the index f in the bisection table is recorded, and the feedback operation will be performed on the current subinterval. Conversely, if u_m is not in the recording interval, which indicates that the u_m value is not in the current subinterval, then the feedback operation will be performed on the next subinterval. Second, the updating of the iteration interval and the operations of the bisection table of the IR method in the feedback operation are as follows: The iteration interval is updated by the recording interval index in the feedback to the current subinterval; then, the recording interval is updated according to $[u_s, u_m, u_e, 2]$. In addition, a deletion operation is performed on the bisection table, that is, all subsequent records of the index f are deleted. However, the iteration interval will be directly updated by the next subinterval in the feedback to the next subinterval, and all records in the bisection table are deleted.

The iterative result u_m determines the principle of performing iteration operations or feedback operations in the NR method. If u_m is in the current iteration interval, then the probability that the convergence solution lies in the interval increases. In this case, the iteration operation continues until the condition $E_m < \beta$ is satisfied to obtain the global optimum. Conversely, if the iterative result u_m is not in the current iteration interval, which indicates that the convergence solution is out of the interval, then the feedback operation is performed.

Finally, in the FHP-BFS algorithm, the different processing methods in the NR method and the BFS algorithm should be noted. If the iteration result u_m of the NR method does not satisfy the threshold β, it is necessary to determine whether the calculated result is in the current interval before the iteration operation continues; moreover, the iteration parameter is updated by u_m directly. The iteration operation of the BFS algorithm is performed directly, and the iteration parameter is updated by the recording interval in the bisection table. In addition, the current iteration interval $[u_s, u_e]$ of the NR method is unchanged, and the iterative value u_m of the subsequent iterations is not recorded in the bisection table until u_m is fed to the BFS algorithm or the iterative value u_m of the NR method converges. However, the current iteration interval $[u_s, u_e]$ of the BFS algorithm changes according to the values of the index, and the iteration interval is permanently recorded in the bisection table.

3.2. Flattening Algorithm Based on the FHP-BFS Algorithm

The purpose of applying the FHP-BFS algorithm to the flattening algorithm is mainly to improve the computation speed. The processes of the flattening algorithm between inversion and projection are distributed in series; hence, the whole computation speed can be improved by enhancing the computation speed of the individual processes. Furthermore, the progress of the precision in the inversion process will directly reduce the error of the subsequent projection operation, indirectly affecting the updating accuracy of the control points and knot vectors.

Algorithm 1 shows the pseudocode of the improved flattening algorithm based on the FHP-BFS algorithm. The improved algorithm, which directly corresponds to the task of ship hull reconstruction, uses the data of the offsets table of the ship hull as input and then interpolates the data to half-width cross-section NURBS curves. In lines 4 and 5, the FHP-BFS algorithm inverses the flattening points; the inversion solutions u_l, u_r are involved in the knot refinement operation in line 11, and then the control points are updated according to the projection operation in line 12 based on the refined knot vectors. Therefore, the processes of control point updating and the flattening effect are affected by the improvement in the inversion based on the FHP-BFS algorithm. In addition, the NURBS interpolation algorithm and the knot refinement algorithm are used separately in lines 2 and 11. Detailed information on the two algorithms, such as the principle and parameter settings, can be found in [15].

Algorithm 1: Flattening algorithm based on the fast high-precision bisection feedback search (FHP-BFS) algorithm.

Input: Q—list of offsets table; Q_l, Q_r—location points of flattening line segment ends; p—degree of the interpolated NURBS curve.
Output: P, U— control point vector and knot vector of a flattened NURBS curve.
function $FlatteningAlg(Q, Q_l, Q_r, p)$
  $P, U = NURBSInterpolation(Q, p)$//interpolation operation of list Q
  $//u_l, u_r$ are parametric values of the endpoint of the flattening line segment
  $u_l = FHPBFS(P, U, Q_l)$//point inversion by the FHP-BFS algorithm
  $u_r = FHPBFS(P, U, Q_r)$
  $//s$ is the number of points projected successfully
  $//n$ is the knot refinement number
  $n = p$
  while $s \leq p + 1$ **do**
    $n = n + 1$
    $U = KnotVecRefine(U, u_l, u_r, n)$//knot refinement algorithm
    Projection of the control point P located on the same side of the flattening line segment
    update P, s
  end while
  return P, U
end function

4. Results

In this section, the effectiveness of the algorithms is verified by comparative experiments. In the experiments, the cross-section data of a ship hull are selected as the original data, and the flattening points are extracted as the inversion sample points. Table 1 shows the half-width cross-section data of the ship hull of selected cross-sections. Figure 9 shows the distribution of the interpolated cross-section NURBS curve, where the left-hand coordinate system is taken as the coordinate system, and the origin of the coordinates is located at the stern of the ship. From the origin of the coordinates, each cross-section is numbered in the positive direction of the x-axis, called the "station". The cross-section data of 32 stations are used in this section.

Table 1. Sample points of the half-width cross-section of the ship station.

x (Station 4)			x (Station 14)			x (Station 32)		
Index	**y**	**z**	**Index**	**y**	**z**	**Index**	**y**	**z**
1	0.000	0.731	1	0.000	0.000	1	0.000	0.932
2	0.082	0.750	2	4.490	0.000	2	0.115	1.000
3	0.179	0.821	3	5.000	0.035	3	0.267	1.241
4	0.303	1.000	4	6.000	0.313	...	...	...
5	0.426	1.287	5	6.322	0.500	10	0.000	4.306
...	...	...	6	6.541	0.710	11	0.000	7.306
23	6.191	6.953	7	6.732	1.000	12	0.164	7.371
24	6.359	7.610	8	6.850	1.319	...	...	...
25	6.531	8.630	9	6.900	1.719	14	2.000	8.877
26	6.638	9.520	10	6.900	15.00	15	2.505	9.284
27	6.638	15.00	-	-	-	16	2.505	15.00

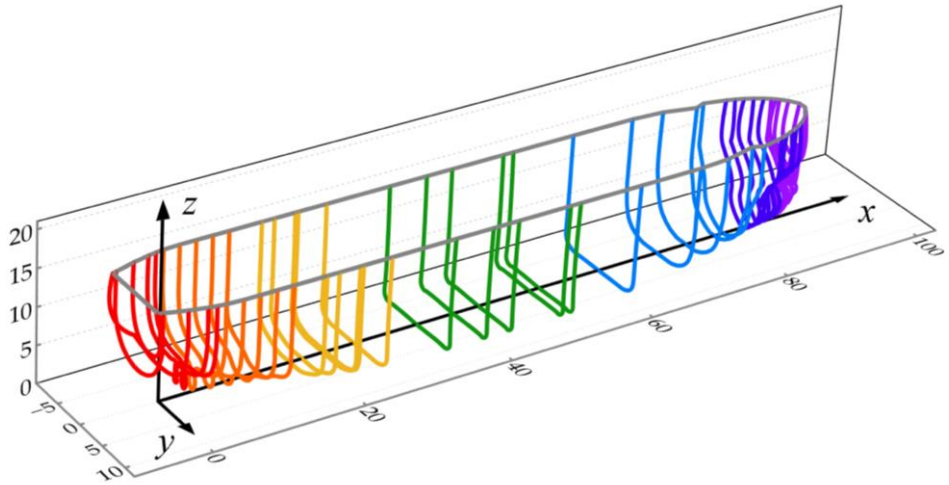


Figure 9. Distribution of the interpolated cross-section curves of 32 ship hull stations.

The method for extracting the flattening points from the sample data is as follows: for the cross-section data at the same station, if the y or z coordinates of two adjacent points have the same value and the z or y coordinates have different values, then the two adjacent points are the end of a straight line segment. Figure 10 shows the interpolated NURBS curves of station 4, station 14, and station 32. The data that can be used as sample flattening points are the 26th and the 27th points of station 4 in Table 1, which correspond to p_3 and p_4 in Figure 10a. The 1st and 2nd points of station 14 in Table 1 correspond to p_{10} and p_{11} in Figure 10b. The 15th and 16th points of station 32 in Table 1 correspond to p_{16} and p_{17} in Figure 10c. In the comparative experiments, 100 flattening points are randomly selected as the inversion sample points.

4.1. Comparison of Algorithms between FHP-BFS and IR-BFS

In the comparative experiments in this section, the parameter γ takes the recommended value of 10^{-3}, and the parameter α takes a value equal to parameter β.

4.1.1. Validation of the Practical Effectiveness of the FHP-BFS Algorithm

In this section, experiments are designed to compare the FHP-BFS algorithm and the IR-BFS algorithm with conventional and high-precision threshold values, and the computation time of the iteration process is recorded. The acceleration effect is verified by analyzing the computation time of algorithms in the "precision refinement" process. The selection criteria for the analysis point are as follows: First, 20 points with a single "precision refinement" process are selected as reference points; then, the average computation time of the reference points is calculated; finally, the reference point with a computation time near the average computation time is chosen as an analysis point. According to the criteria, p_{16} is taken as the analyzing point, and the β precision thresholds are set as 10^{-3} and 10^{-13}. In addition, we used a sample with a single "precision refinement" process for the analysis, and in practice, multiple iterations are often the superposition of numerous single processes.

Figure 11 shows the inversion processes of the analyzing point p_{16} based on the FHP-BFS algorithm and the IR-BFS algorithm, where "Process A" denotes the "precision refinement" process. Figure 11a shows the inversion process with the conventional threshold $\beta = 10^{-3}$. The total computation time of the FHP-BFS algorithm is $1.24 \times 10^{-2}s$, and the total computation time of the IR-BFS algorithm is $1.12 \times 10^{-2}s$. In this case, the time consumed by the FHP-BFS algorithm is longer than that of the IR-BFS algorithm. Figure 11b shows the inversion process based on the high-precision threshold of 10^{-13}. The total computation times of the FHP-BFS algorithm and IR-BFS algorithm are $1.72 \times 10^{-2}s$

and $3.27 \times 10^{-2}s$, respectively. In this case, the time consumed by the FHP-BFS algorithm is less than that of the IR-BFS algorithm.

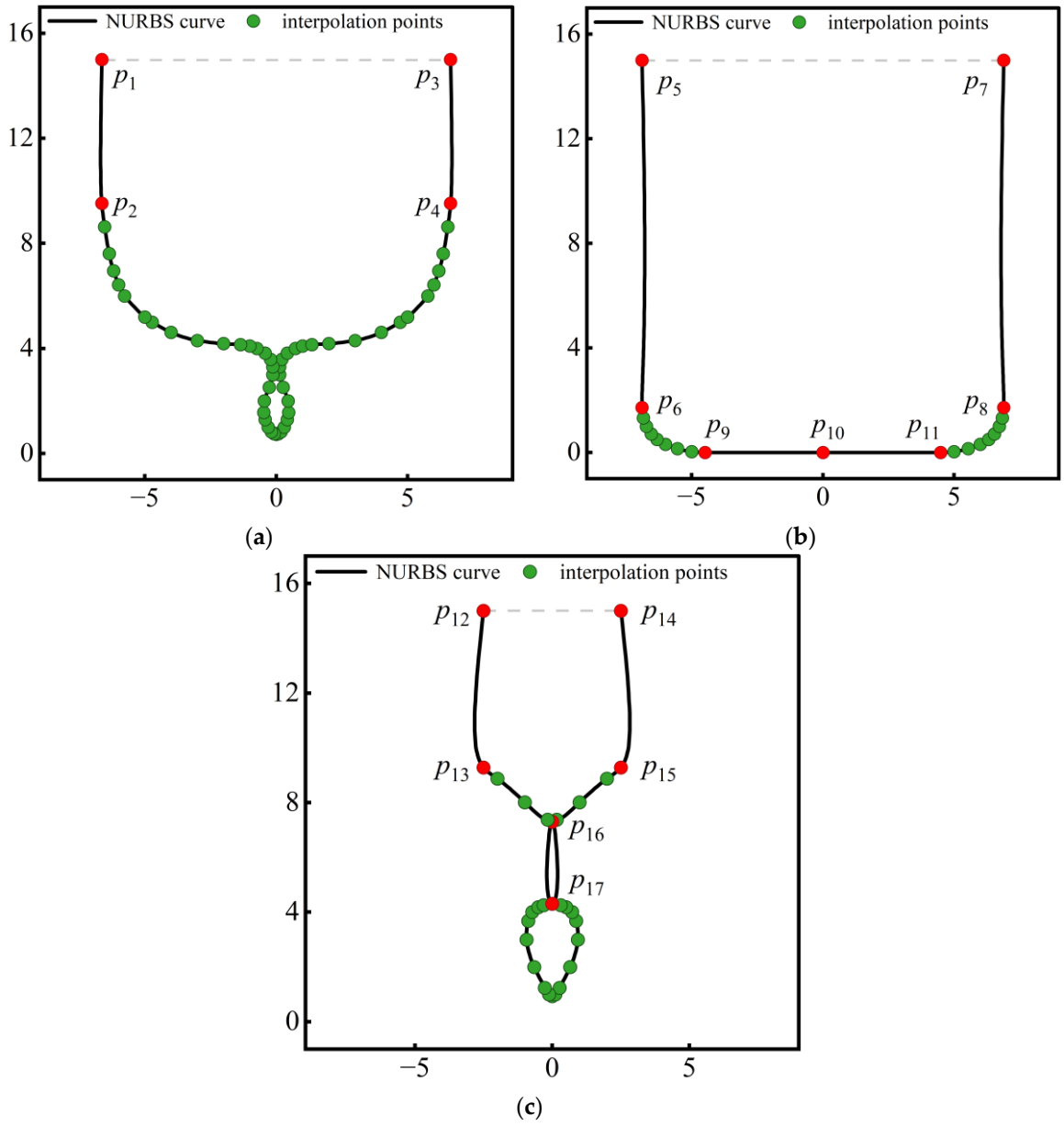
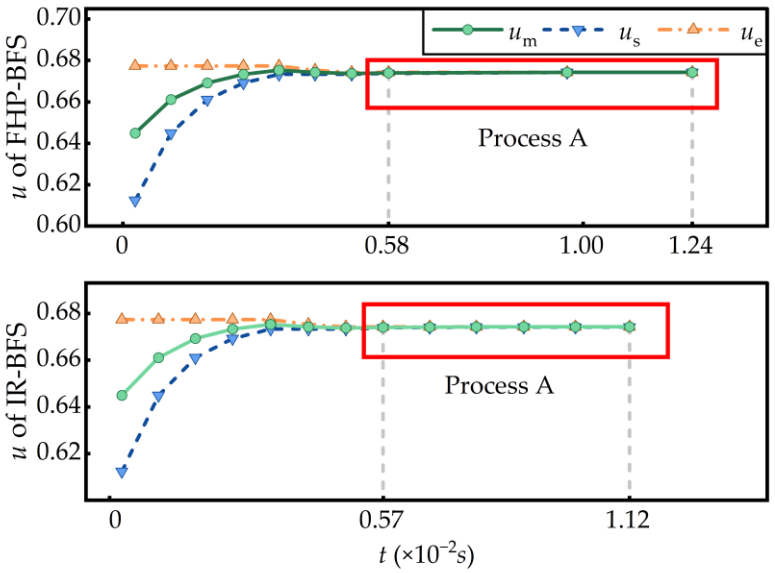
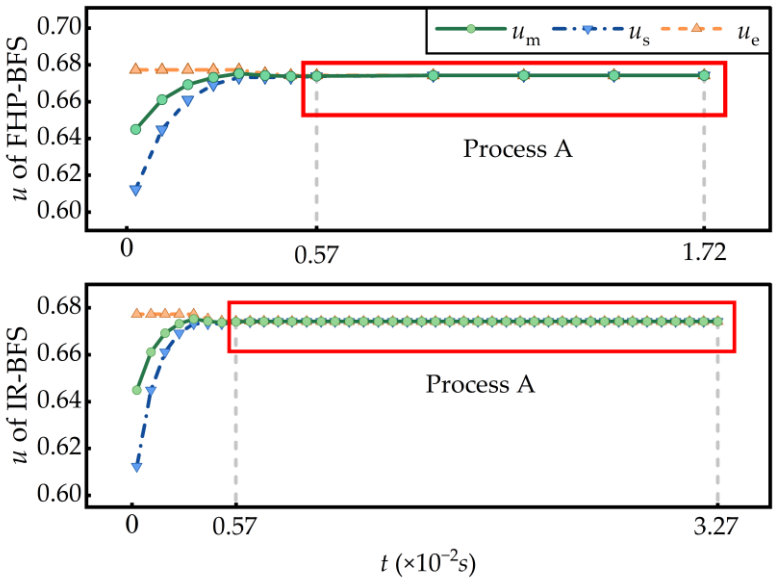


Figure 10. Interpolated cross-section curves and distribution of flattening points of different stations of the ship hull. (**a**) Station 4. (**b**) Station 14. (**c**) Station 32, where the red dots denote the flattening points.

In "Process A" of Figure 11a, the distribution of the time points of the FHP-BFS algorithm becomes longer when $t = 0.58 \times 10^{-2}$. Therefore, the NR method is used at this point, and the precision of the threshold is reached in two iterations. Similarly, the precision of the threshold in the IR-BFS algorithm is reached in five iterations. In Figure 11b, "Process A" of the FHP-BFS algorithm begins when $t = 0.57 \times 10^{-2}$, and the threshold is satisfied in four iterations; moreover, "Process A" of the IR-BFS algorithm also starts when $t = 0.57 \times 10^{-2}$ but too many iterations are needed.



(**a**)



(**b**)

Figure 11. Inversion process of the knot value u of the fast high-precision bisection feedback search (FHP-BFS) algorithm and interval reformation and bisection feedback search (IR-BFS) algorithm with analyzing point p_{16} and at different precision thresholds β. (**a**) β is 10^{-3}. (**b**) β is 10^{-13}, where "Process A" denotes the "precision refinement" processes.

In summary, the FHP-BFS algorithm performs best with a high-precision threshold. However, the advantage of the low computation time is minor with the threshold of conventional precision. Therefore, a suitable precision threshold should be set for the FHP-BFS algorithm to maintain superiority.

4.1.2. Setting the Precision of the Threshold of the FHP-BFS Algorithm

This section determines the optimal precision threshold through comparative experiments to maintain the superiority of the FHP-BFS algorithm. Figure 12 shows the computation time of the inversion process at different precision thresholds based on the FHP-BFS algorithm and the IR-BFS algorithm, which contains the curves of u_s, u_m and u_e;

the straight line segment in the $t\beta$-plane of the coordinate system is the projection of the curve of u_m.

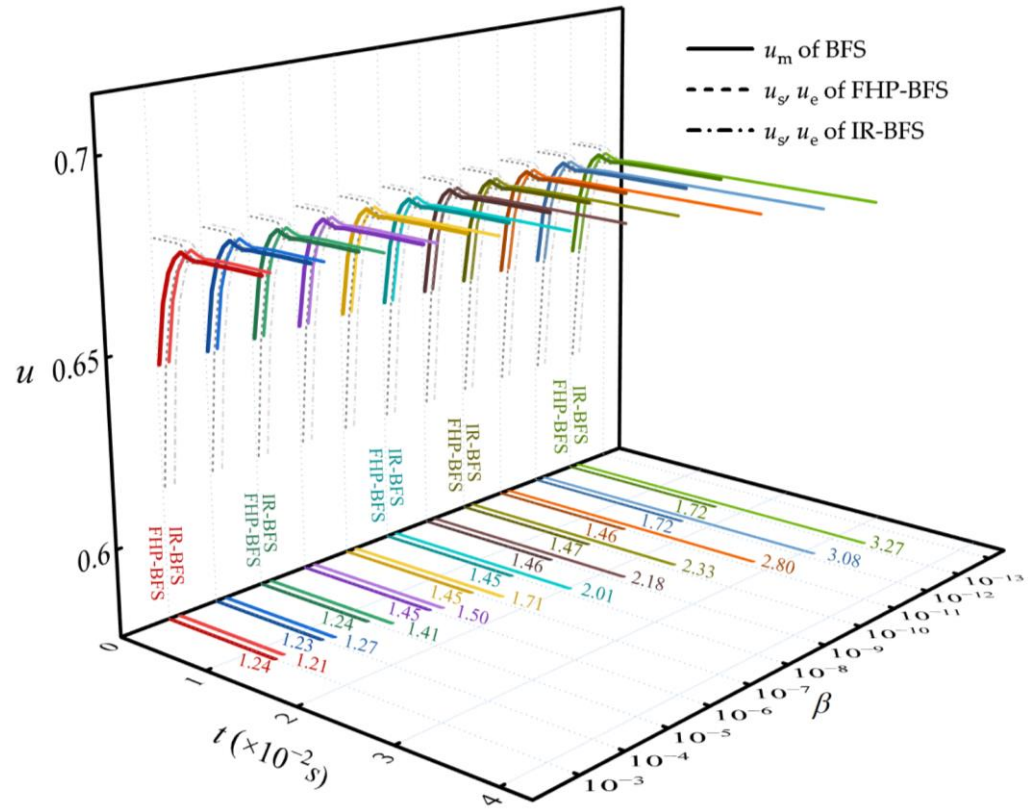


Figure 12. Inversion results of computation time t of parametric value u of sample point p_{16} based on the fast high-precision bisection feedback search (FHP-BFS) algorithm and interval reformation and bisection feedback search (IR-BFS) algorithm at different precision thresholds β.

The value of threshold β decreases from 10^{-3} to 10^{-13}, and the computation time based on the FHP-BFS algorithm is $1.24 \times 10^{-2}s$, $1.23 \times 10^{-2}s$, $1.24 \times 10^{-2}s$, $1.45 \times 10^{-2}s$,$\cdots$, $1.72 \times 10^{-2}s$ and $1.72 \times 10^{-2}s$. The minimum value $t_{\min}$ and the maximum value $t_{\max}$ are $1.23 \times 10^{-2}s$ and $1.72 \times 10^{-2}s$, respectively, and the range of the computation time $t_{\max} - t_{\min}$ is $0.49 \times 10^{-2}s$. The values of the computation time based on the IR-BFS algorithm are $1.21 \times 10^{-2}s$, $1.27 \times 10^{-2}s$, $1.41 \times 10^{-2}s$, $1.50 \times 10^{-2}s$,$\cdots$, $3.08 \times 10^{-2}s$ and $3.27 \times 10^{-2}s$. The minimum value $t_{\min}$ and the maximum value $t_{\max}$ are $1.21 \times 10^{-2}s$ and $3.27 \times 10^{-2}s$, respectively, and the range of the computation time $t_{\max} - t_{\min}$ is $2.06 \times 10^{-2}s$. Hence, the distribution of values of the computation time in the FHP-BFS algorithm fluctuates less and is relatively stable in the inversion process. In contrast, the distribution of the computation time in the IR-BFS algorithm has approximate linear growth with large fluctuations. Therefore, the computation time of the FHP-BFS algorithm has good robustness and is not significantly affected by the precision of the threshold, while the computation time of the IR-BFS algorithm is more affected by variations in the precision threshold.

In the inversion process of the sample point p_{16} in Figure 12, the computation time of the FHP-BFS algorithm is longer than that of the IR-BFS algorithm when the threshold $\beta = 10^{-3}$, while the computation time of the FHP-BFS algorithm is less than that of the IR-BFS algorithm when the threshold $\beta \leq 10^{-3}$. A comparative experiment is designed to discuss the inversion threshold that makes the computation time of the FHP-BFS algorithm better than that of the IR-BFS algorithm. To ensure the generality of the analysis results, 50 sample points are randomly selected in the experiment, and the precision of threshold β is set as $10^{-i}(i = 3, 4, 5, \cdots, 12, 13)$. In addition, to discuss the improvement in the computation time of the FHP-BFS algorithm at different precision thresholds, the parameter of

improved percentage of computation time T_{percent} is proposed to measure the computation efficiency of the FHP-BFS algorithm. The calculation formula is as follows:

$$T_{\text{percent}} = \frac{(t_{\text{IR-BFS}} - t_{\text{FHP-BFS}})}{t_{\text{IR-BFS}}} \times 100\%, \tag{7}$$

where $t_{\text{FHP-BFS}}$ and $t_{\text{IR-BFS}}$ are the computation times based on the FHP-BFS algorithm and the IR-BFS algorithm, respectively. If $T_{\text{percent}} > 0$, the computational efficiency of the FHP-BFS algorithm is higher than that of the IR-BFS algorithm at a specific threshold. If $T_{\text{percent}} < 0$, the computational efficiency of the FHP-BFS algorithm is lower than that of the IR-BFS algorithm. If $T_{\text{percent}} = 0$, the computational efficiency of the IR-BFS algorithm is nearly equal to that of the IR-BFS algorithm. Figure 13 shows the superposition of the scatter distribution and the box plot of the computation efficiency T_{percent}. Six outliers, whose values are too large or too small, are deleted.

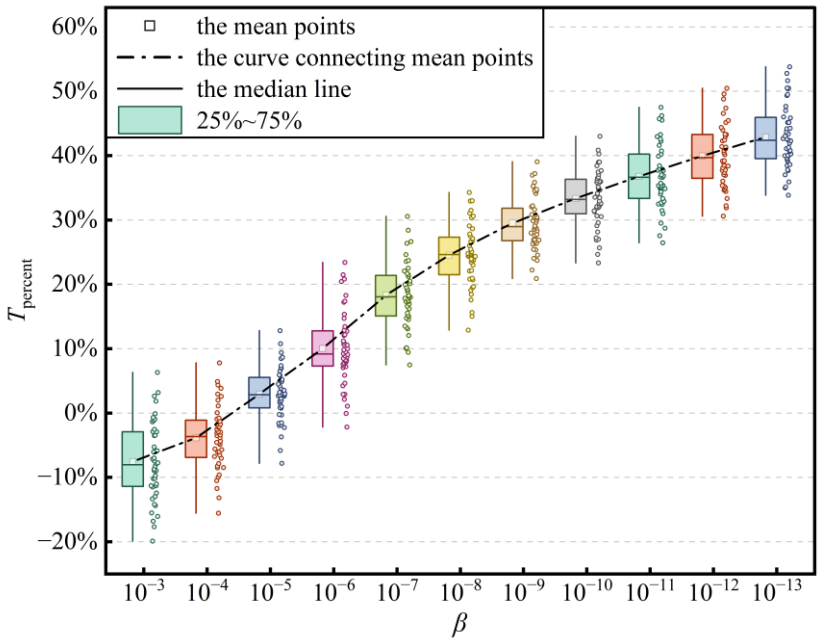


Figure 13. Superposition of the scattering distribution and box plot of T_{percent} of the fast high-precision bisection feedback search (FHP-BFS) algorithm at different thresholds β, where T_{percent} denotes the improved percentage of computation time.

Figure 13 shows that T_{percent} tends to increase as the threshold β numerically decreases from 10^{-3} to 10^{-13}, indicating that the computational efficiency of the FHP-BFS algorithm increases with the precision of the threshold. Taking the mean scatter value of T_{percent} as the analysis target, if $\beta \geq 10^{-4}$, then $T_{\text{percent}} < 0$ indicates that the computation efficiency of the FHP-BFS algorithm is lower than that of the IR-BFS algorithm; if $\beta = 10^{-5}$, then T_{percent} is slightly greater than zero, while some scatter points with values less than zero exist, indicating that the computation efficiency of the FHP-BFS algorithm is nearly equal to that of the IR-BFS algorithm; and if $\beta < 10^{-5}$, then $T_{\text{percent}} > 0$, and the overall trend is increasing, indicating that the computation efficiency of the FHP-BFS algorithm is higher than that of the IR-BFS algorithm. In this situation, the smaller the value of threshold β is, the higher the computational efficiency of the FHP-BFS algorithm. Therefore, the computational efficiency of the FHP-BFS algorithm is improved under the condition of a high-precision threshold. Simultaneously, it is recommended that the range of the threshold is $\beta \leq 10^{-5}$ to maintain the superiority of the FHP-BFS algorithm.

4.2. Comparison with Other Algorithms

Comparative experiments are designed with the best existing compound algorithms to prove the effectiveness of the FHP-BFS algorithm in this section. The compared compound algorithms are the algorithms of IR-BFS [28], Chen et al. [20], Selimovic [19] and Ma and Hewitt [18], which are abbreviated as IR-BFS, Chen, Seli, and Ma, respectively, in this section. Furthermore, the root-finding algorithms that perform well in the local interval are also compared, which include the algorithms of [41–43], and they are abbreviated as Badr, Sab, and Kim, respectively.

In the IR-BFS algorithm, the IR method is proposed to shrink the range of the target interval, and the BFS algorithm is proposed to jump out of local optima. Chen subdivided the NURBS curve into Bezier sub curves, and the rough solution was obtained when only one optimal solution was contained in the interval; the exact solution was obtained by a hybrid algorithm of the bisection method and the NR method. Seli proposed the internal knot clipping method to eliminate intervals, and a rough solution is obtained when the sufficient flatness of the subcurve is satisfied or when the range of the solution interval is less than the given tolerance; the exact solution is calculated by the NR method. Ma subdivided the NURBS curve into Bezier subintervals by finding a simple and convex control polygon, and the rough solution was obtained by the iteration of subintervals between the control polygon and the test point; the exact solution was calculated by the NR method. Badr selects the optimal iteration value by the trisection and false position methods. Sab proposed a three-way hybrid root-finding algorithm based on the previously proposed two-way algorithm. The algorithm uses the methods of bisection, false position, and NR to select the optimal iteration value. However, the problem of computing the global optimal solution is still not considered. Kim combined the NR method and the bisection algorithm to speed up the calculation and improve the local convergence ability of the algorithm. However, the algorithm cannot jump out of the optimal local solution.

Since 50 sample points have been selected in Section 4.1.2 to analyze the optimal precision range of the threshold for the FHP-BFS algorithm, to reflect the algorithm's generalization ability effectively, the other 50 sample points are selected as inversion points in this section. The inversion precision threshold is set as $10^{-i}(i = 3, 4, 5, \cdots, 12, 13)$. The test points or query points in the algorithms are the inversion sample points. The knot refinement algorithm handles all operations for converting NURBS curves to Bezier curves. The NR algorithm is considered not converged if the accuracy threshold is not satisfied after 20 iterations. In the FHP-BFS algorithm, the threshold γ for the NR method is set to 10^{-3}. The settings of the parameters of Chen in the hybrid algorithm based on the bisection method and the NR method can be found in [44]. The tolerance of the solution interval in the Seli algorithm is set to β for the sufficient flatness of the subcurves. The maximum number of iterations of the Ma algorithm in dividing the NURBS curve into Bezier curves is set to 20 to avoid unnecessary time consumption.

The root-finding algorithms easily fall into local optima if directly used to perform inversion because they cannot find the global optimal value. Therefore, some processing must be performed before these algorithms are used; that is, the previously proposed IR-BFS algorithm was used to reduce the interval of parameters within 0.1 to minimize the possibility of the root-finding algorithms falling into local optimal values in the samples. The iteration of the root-finding algorithms terminates when the iterative times arrive at 20 to avoid consuming too much time in nonconvergent samples.

Figure 14 shows a 3D heatmap and 2D contour map of the computation time t of the sample points calculated by the compound algorithms at different threshold precisions. The 2D contour map in the bottom plane denotes the projection of the 3D heatmap, the dotted line is the contour line, and the dashed line is the auxiliary line for observing the computation time of different algorithms.

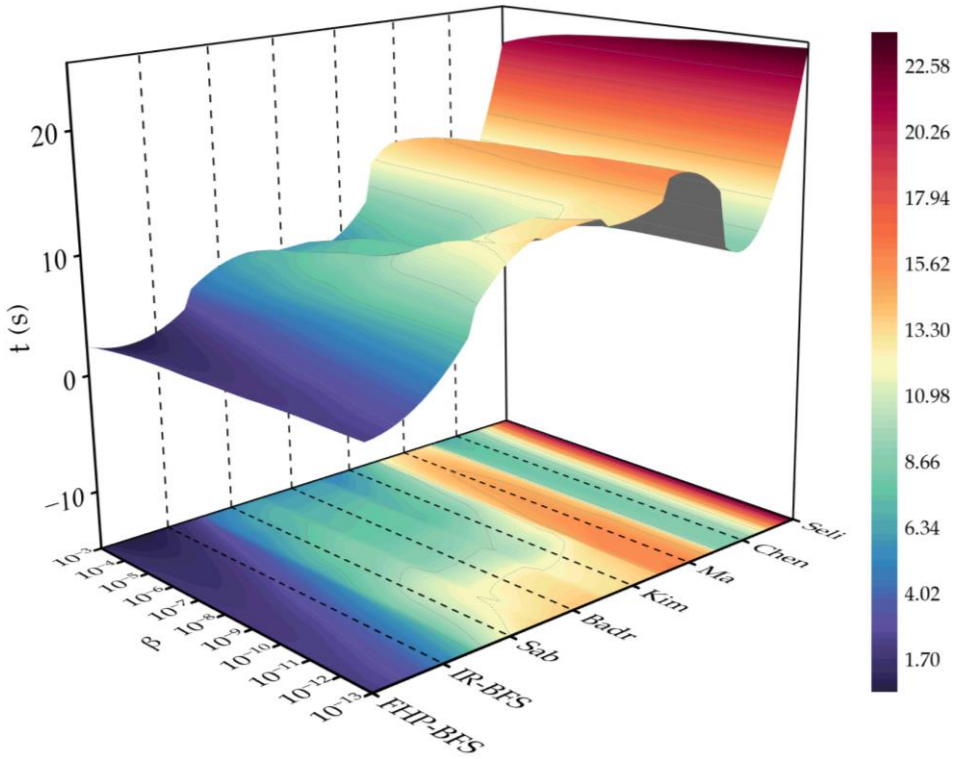


Figure 14. 3D heatmap and 2D contour map of the computation time distribution of 50 sample points based on different algorithms at different threshold precision values, where the color is the mapping of the value of calculation time t.

The 3D heatmap shows that the FHP-BFS and IR-BFS algorithms have shorter computation times, and that Seli and Ma have the highest computation times. Through the 2D contour map, the mapping colors of the IR-BFS and FHP-BFS algorithms are both dark purple when the values of threshold precision change from 10^{-3} to 10^{-5}, which indicates that the computation time of the IR-BFS algorithm is similar to that of the FHP-BFS algorithm in this case; however, when the values of threshold precision change from 10^{-5} to 10^{-13}, the mapping color of the IR-BFS algorithm gradually becomes blue, while the mapping color of the FHP-BFS algorithm remains purple and is almost unchanged, which indicates that the computation time of the IR-BFS algorithm increases in this case and gradually exceeds that of the FHP-BFS algorithm. Furthermore, the mapping colors of the Sab, Badr, and Kim algorithms change significantly with the threshold precision. The colors are light blue or green when the threshold precision varies from 10^{-3} to 10^{-6}, indicating that relatively little computation time is consumed. However, when the values of the threshold precision change from 10^{-6} to 10^{-13}, the colors gradually change to yellow or even orange, which indicates that the computation time is significantly increased. The considerable variation in computation time is because the root-finding algorithms cannot jump out of local optimal values, and the number of samples that cannot converge gradually increases after the accuracy threshold increases. Therefore, the FHP-BFS algorithm performs the best in the experiment, and the root-finding algorithms have medium-level performance among the compared algorithms.

The experimental results are analyzed in more depth to make more practical and theoretical conclusions. The computation process of the compound algorithms is divided into the processes of P_{rough} and P_{exact}, which denote the process of computing the rough solution and the exact solution, respectively. Moreover, the computation time of each process is represented as t_1 and t_2, respectively. The total computation time is represented as t; i.e., $t = t_1 + t_2$. In the FHP-BFS algorithm, t_1 denotes the computation time consumed by the IR-BFS algorithm with a precision threshold of $\beta < 10^{-3}$, and t_2 denotes the time

consumed by the NR method. In the Chen algorithm, t_2 denotes the computation time consumed by the hybrid algorithm composed of the bisection algorithm and NR method. In the algorithms of Ma and Seli, t_2 denotes the computation time consumed by the NR method. In addition, the root-finding and IR-BFS algorithms are not divided into subprocesses because they do not need to compute rough solutions. Figure 15 shows the 2D heatmap of the computation times of processes based on the algorithms at different threshold precisions in the inversion of sample points. The specific values corresponding to each color are added to the figure and rounded to two decimal places to compare the results more precisely. In addition, IR in Figure 15 denotes the IR-BFS algorithm.

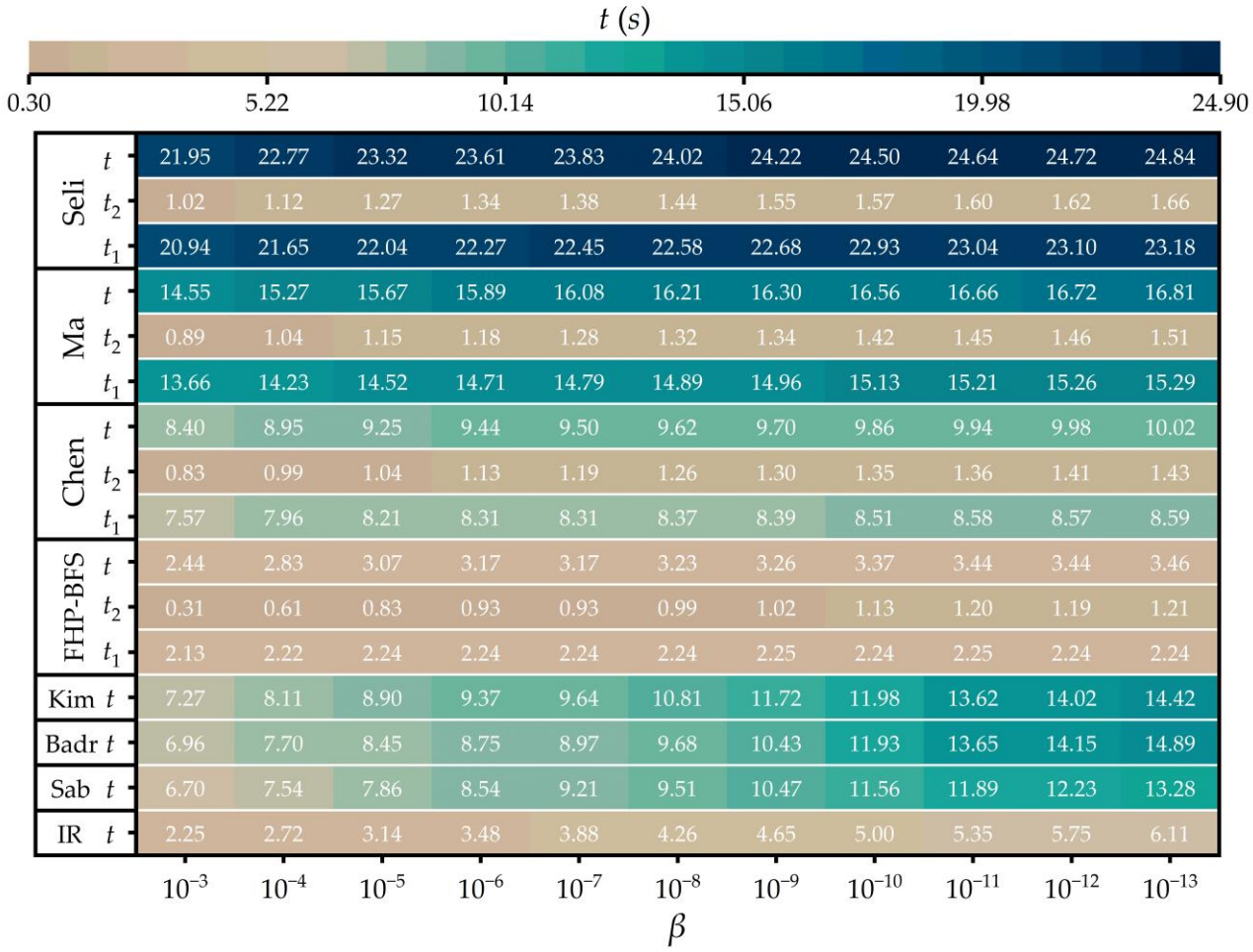


Figure 15. 2D heatmap of the computation times of the inversion of sample points calculated by the algorithms at different threshold precision values, where IR denotes the fast high-precision bisection feedback search (FHP-BFS) algorithm.

In Figure 15, when $\beta = 10^{-3}$, the computation times, t_1, are 2.13s, 7.57s, 13.66s and 20.94s in P_{rough} according to the compound algorithms of FHP-BFS, Chen, Ma and Seli, respectively, and the range of t_1 is 18.81s, which is calculated by $t_{max} - t_{min}$. In addition, the computation times, t_2, are 0.31s, 0.83s, 0.89s and 1.02s in P_{exact} according to the compound algorithms, and the range of t_1 is 0.71s. Therefore, the FHP-BFS algorithm consumes the least amount of time in both the P_{rough} and P_{exact} processes, and the range of t_1 is greater than that of t_2. When $\beta = 10^{-4}$, the computation times, t_1, are 2.22s, 7.96s, 14.23s and 21.65s in P_{rough}, and the range of t_1 is 19.43s. Additionally, the computation times, t_2, are 0.61s, 0.99s, 1.04s and 1.12s in P_{exact}, and the range of t_2 is 0.51s. Therefore, the FHP-BFS consumes the least time in the two processes, and the range of t_1 is greater than that of t_2. Similarly, it can be found that when $\beta > 10^{-4}$, the distribution of computation time t_1

is consistent with this conclusion. Therefore, the FHP-BFS algorithm consumes the least computation time compared to the other compound algorithms in both the P_{rough} and P_{exact} processes. In addition, the range of t_1 is greater than that of t_2, which indicates that the differences in the computation time are significant in P_{rough}, while the differences are minor in P_{exact}.

Moreover, the performance of the computation time with the change in threshold precision β should be noted. In the inversion of the FHP-BFS algorithm, the maximum and minimum values of t are 3.46s and 2.44s, respectively; the range of t is 1.02s, where the range of t_1 is 0.11s and the range of t_2 is 0.9s. In the inversion of Chen, Ma, and Seli, the ranges of t are 1.62s, 2.26s and 2.89s, respectively. Furthermore, the ranges of t_1 are 1.02s, 1.63s and 2.24s; the ranges of t_2 are 0.6s, 0.62s and 0.64s. Through the comparison, the FHP-BFS algorithm has the smallest range of t_1, indicating that the algorithm has the best robustness with the change in threshold precision β in P_{rough}. Moreover, the ranges of t_2 among the compound algorithms are similar, and the range of t for the IR-BFS algorithm is 3.86s, which indicates that the design of the NR method maintains better robustness in P_{exact}.

In summary, the FHP-BFS algorithm, which consumes the least computation time in both the P_{rough} and P_{exact} processes, performs the best in computation efficiency and robustness among the compound algorithms. Specifically, the FHP-BFS algorithm significantly reduces the computation time in P_{rough} and, to some extent, reduces the time in the P_{exact} process.

4.3. Evaluation of the Flattening Algorithm

This section designs experiments to verify the precision performance of the improved flattening algorithm. The flattening performance can be judged by the curvature change near flattening points before and after the flattening operation. If the curvature near the flattened points is gradually reduced to 0, the algorithm has a good flattening effect. The cross-section curves at stations 4, 8, and 27 are taken as sample curves. In addition, the threshold precisions are set as $\alpha = \beta = 10^{-8}$ and $\gamma = 10^{-3}$. Figure 16 shows the curvature of the sample curves before and after the flattening operation, where p_s and p_e denote the two flattening points. Figure 17 shows porcupine plots of the curvature distribution of the flattened ship cross-section curves. For the sake of making the curvature distribution clear, the curvature value greater than $1.5m^{-1}$ is still represented by $1.5m^{-1}$ in the plot.

Figure 16a,c show the curvature of the NURBS curve of stations 4 and 27. The two flattening points of each figure are distributed at the right end of the NURBS curve, and the curvature after point p_s is gradually reduced to 0. The curvature near the flattening point p_s increases after flattening, which not only enhances the bending degree of flattened curves near point p_s but also speeds up the rate at which the curves' curvature drops to zero after point p_s. In Figure 17a,c, the flattened NURBS curve of the right half-width cross-section of the ship hull corresponds to the curvature in Figure 16a,c), in which the curve between points p_s and p_e is converted to a straight line, and the remaining part of the half-width NURBS curve is still expressed as a curve. Therefore, the precision of inversion of flattening points satisfies the preset threshold of 10^{-8}, and the flattening algorithm performs well on the NURBS curve of stations 4 and 27.

Figure 16b shows the curvature of the NURBS curve of station 8. The two flattening points are distributed at the right end of the NURBS curve and the curvature after point p_s is gradually reduced to 0. Comparing the curvature change before and after flattening, the curvature of the flattened curve decreases at point p_s; the curvature after point p_s has a negative value and the extremum of curvature after point p_s is smaller than before flattening, where the negative sign only indicates the direction of curvature. Therefore, the curvature decrease at the flattening point also reduces the curvature change after the flattening point in the opposite direction, which eventually speeds up the curvature drops to 0 in the flattening part. In Figure 17b, the flattened curve of the right half-width corresponds to the curvature in Figure 16b, in which the flattened NURBS curve is successfully expressed

as a straight line in the flattening part. Therefore, the inversion precision of the flattening points satisfies the requirements, and the flattening algorithm performs well in this case.

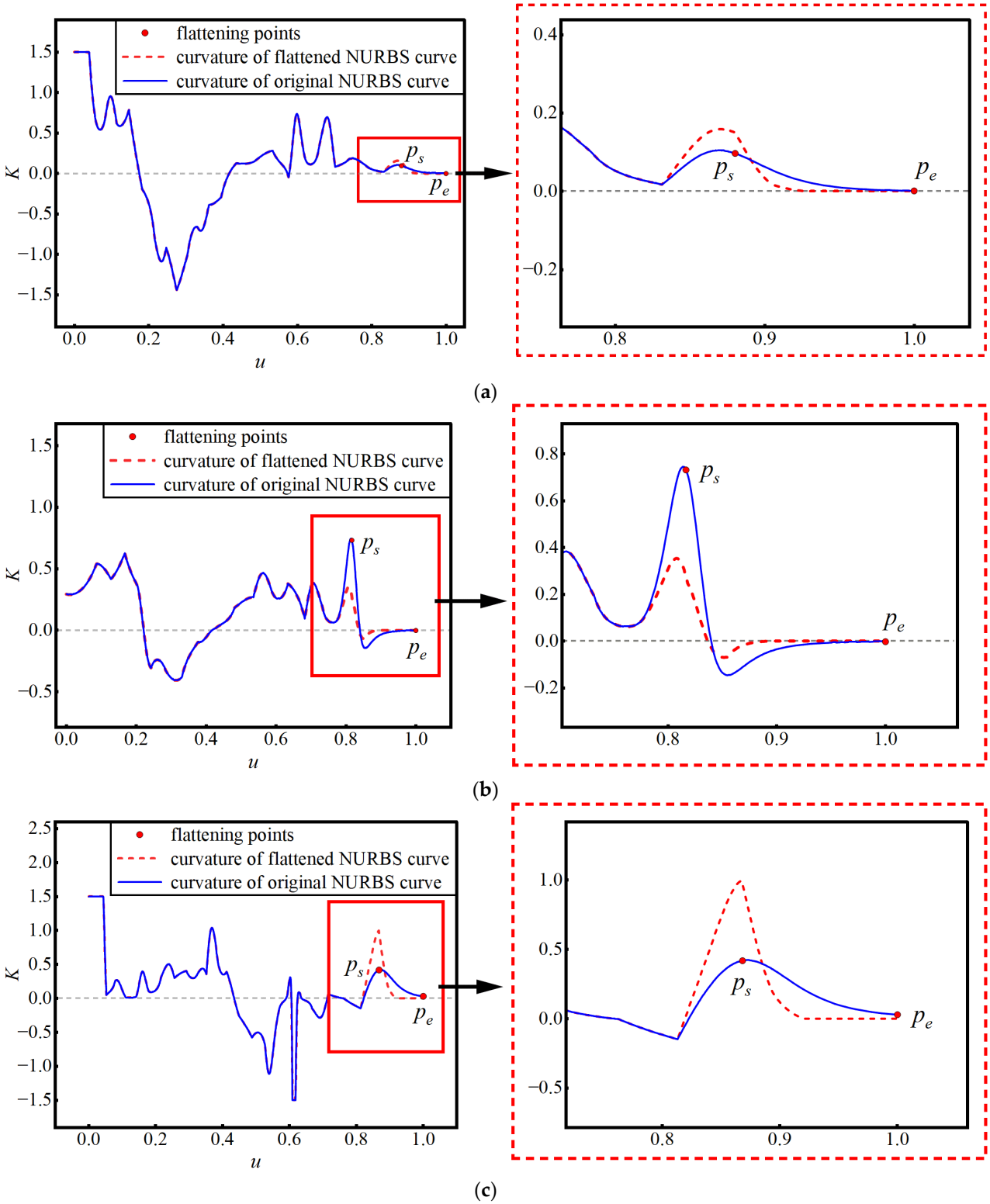


Figure 16. Curvature of the NURBS curve of the half-width cross-section of the ship hull before and after the flattening operation. (**a**) Station 4; (**b**) Station 8; (**c**) Station 27, where, p_s and p_e denote the two flattening points; and K denotes the curve curvature.

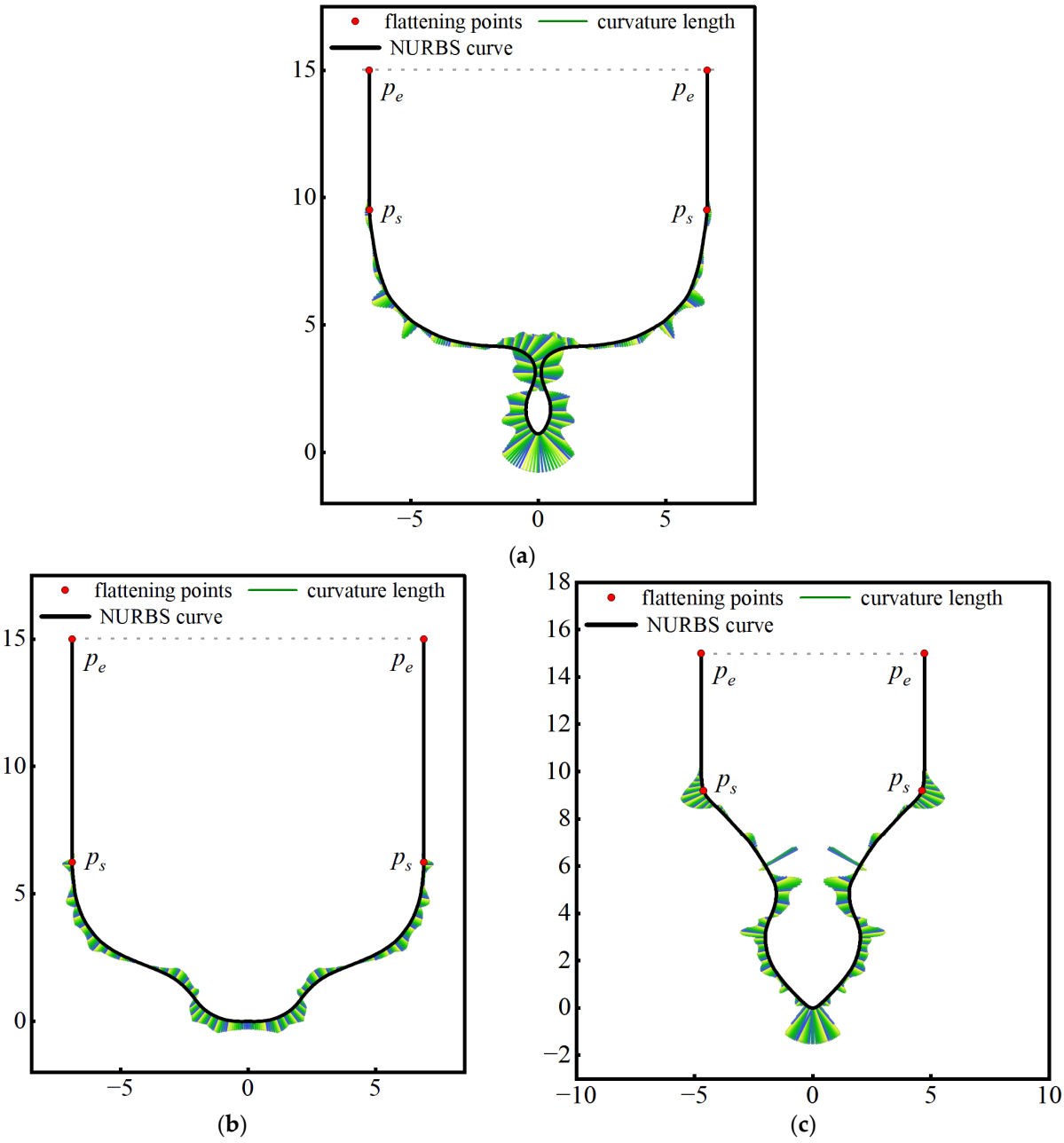


Figure 17. Porcupine plot of the curvature distribution of the NURBS curves of the ship hull cross-section. (**a**) Station 4; (**b**) Station 8; (**c**) Station 27.

In summary, the flattening algorithm based on the FHP-BFS algorithm can gradually change the curvature near the flattening point and exhibits a good flattening effect. The improved flattening algorithm, which ensures that the inversion results of the flattening points meet the high-precision threshold, can improve the computation efficiency and maintain the smoothness of the flattened curves.

5. Discussion

In this paper, a fast inversion algorithm of the NURBS curve with a high precision-threshold is proposed and applied to the NURBS curve-flattening algorithm to improve the calculation speed. Then, through a series of comparative experiments, the algorithms are verified.

Section 4.1.1 compared the proposed FHP-BFS algorithm with the IR-BFS algorithm. The IR-BFS algorithm was slightly faster in conventional precision situations, and the FHP-BFS algorithm was more rapid in high-precision cases. By comparing the number of iterations, the iterations needed for convergence of the FHP-BFS algorithm were much less than those of the IR-BFS algorithm. This indicated that the FHP-BFS algorithm is more robust in computation time and revealed the fundamental reason for the better performance in high-precision computation. Thus, although the IR-BFS algorithm has the advantage of fast single iterations, it must go through many iterations due to its low convergence speed; under this condition, the inversion process eventually leads to a long total computation time. In contrast, the NR process of the FHP-BFS algorithm converged quickly, although it has the disadvantage of a long single iteration time. Therefore, the superiority of the FHP-BFS algorithm over the IR-BFS algorithm is reflected well in the case of the long computation time. Then, a suitable precision value of threshold 10^{-5} was determined to maintain the priority of the FHP-BFS algorithm through comparative experiments in Section 4.1.2.

Section 4.2 compared the FHP-BFS compound algorithms with the algorithms of [18–20,28,41–43]. Algorithms [41–43] are local root-finding algorithms that perform very average computation time. Algorithms [18–20] consumed too much time computing the rough solution but showed a significant advantage in calculating the exact solution. The IR-BFS algorithm calculates the rough solutions of the FHP-BFS algorithm, and the comparison demonstrated that the proposed approach makes the algorithm more efficient in this process. Ref. [18] and ref. [20] calculated rough solutions by dividing the NURBS curves into subcurves with their proposed partitioning algorithms. The comparison demonstrated that this approach consumes too much time due to the intersection and interpolation of curves. Ref. [19] calculated rough solutions by dividing curves directly by the NURBS interpolation algorithm, which lacks directness to the target solution and consumes too much time. Therefore, the computation time required to obtain rough solutions significantly differs among the compound algorithms. In addition, the exact solutions of the algorithms FHP-BFS, refs. [18,19] are calculated by the NR method; Ref. [20] calculates the exact solution with the hybrid algorithm composed of the bisection algorithm and the NR method. Therefore, the computation time in obtaining exact solutions is similar among the compound algorithms due to the same convergence rate of the NR method. In brief, the FHP-BFS algorithm, which takes advantage of the IR-BFS algorithm and NR method to compensate for their respective disadvantages, performs best in terms of robustness in the two processes.

Section 4.3 verifies the smoothness of the flattened NURBS curve while ensuring the computation time at high-precision thresholds. In the flattening algorithm, the processes of inversion and projection distribute in series; hence, the overall computation efficiency of the algorithm is directly improved by reducing the computation time in the inversion process. Finally, the FHP-BFS algorithm speeds up the computation of the flattening algorithm.

In summary, the proposed FHP-BFS algorithm can improve the computation efficiency at the proposed threshold precision, especially at high precision values. However, the algorithm still needs further improvement. First, in the selection of the threshold γ, increasing the threshold γ in many samples can further improve the calculation speed, but in a few samples, the problem of increasing the computation time will also occur. To solve this problem, we can set the optimal threshold precision γ for different inversion points through the curve characteristic parameters. Second, in the interpolation process of the NURBS curve-flattening algorithm, it is still necessary to consider the influence of the number of control points of the flattened curve to avoid causing a more complicated form. Third, the algorithm is only a high-precision and fast inversion research for NURBS curves, and it still needs to be further applied to NURBS surfaces.

6. Conclusions

This paper studies how to solve the "precision refinement" problem in NURBS curve inversion based on ship hull station curves. A new compound algorithm is proposed to calculate the exact solution using the faster convergence algorithm to solve the problem. Then, the optimal values of the parameters in the algorithm are determined by experiments, and many comparison experiments are performed with other algorithms. Finally, the proposed algorithm is applied to the NURBS curve-flattening algorithm to improve the computational efficiency. The main contributions are as follows:

(1) The FHP-BFS algorithm, a compound algorithm that improves computational efficiency while guaranteeing computational accuracy, is proposed. In the algorithm, the fast single iteration of the BFS algorithm ensures the quick inversion of rough solutions, and the NR algorithm provides fast convergence to the exact solution. Then, the FHP-BFS algorithm is compared with the best existing algorithms, and the high computational efficiency of the FHP-BFS algorithm is demonstrated with high-precision thresholds.

(2) The optimal range of the threshold precision in the FHP-BFS algorithm is proposed. The computation time of the inversion solutions is compared at different threshold precisions. Then, the relationship between the improved percentage of computation time and the threshold precisions is analyzed, and the optimal range of the threshold precision is derived. Furthermore, the computation time consumption of the FHP-BFS algorithm is compared at the optimal precision threshold, and the high efficiency is verified. In addition, the proposed ranges of the precision thresholds can make the FHP-BFS algorithm easier to use in other applications.

(3) The flattening algorithm of the NURBS curve is improved based on the FHP-BFS algorithm. The precision of the improved flattening algorithm in the processes of projection and control point updating is greatly enhanced by considering the factors of high precision and low computation time in the inversion of flattening points. Moreover, the effect of the improved flattening algorithm is verified by the change in the curvature of the curves before and after flattening.

In subsequent research, the proposed algorithm will be applied to computation tasks based on ship hull reconstruction, such as the calculation of ship damage stability, ship hull strength, and ship hull viscous resistance. In addition, the FHP-BFS algorithm is general and can be applied to more research areas. In the problem of finding the intersection lines between spline surfaces, the proposed algorithm can be extended to the exaction operation of intersection solutions obtained with errors based on the partition or tracing method. In the preprocessing problem of point cloud data of ship hulls or data of ship automatic identification systems, the proposed algorithms can be implemented to identify and clean anomalies in the dataset through spatiotemporal information.

Author Contributions: Conceptualization, K.Z.; methodology, K.Z.; software, K.Z. and J.L.; validation, K.Z. and J.L.; formal analysis, K.Z. and J.L.; writing—original draft preparation, K.Z. and J.L.; writing—review and editing, K.Z. and J.L.; visualization, K.Z.; supervision, G.S. and J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China [52201414; 51579025; 51709165]; the Provincial Natural Science Foundation of Liaoning [20170540090]; and supported by the Navigation College of Dalian Maritime University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors are grateful for the support of the Key Laboratory of Navigation Safety Guarantee of Liaoning Province, China.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

FHP-BFS	Fast high-precision bisection feedback search
IR-BFS	Interval reformation and bisection feedback search
NURBS	Non-uniform rational B-spline
NR	Newton-Raphson

References

1. Nategh, M.J.; Parvaz, H. Development of computer aided clamping system design for workpieces with freeform surfaces. *Comput.-Aided Des.* **2018**, *95*, 52–61. [CrossRef]
2. Bulian, G.; Cardinale, M.; Dafermos, G.; Lindroth, D.; Zaraphonitis, G. Probabilistic assessment of damaged survivability of passenger ships in case of grounding or contact. *Ocean Eng.* **2020**, *218*, 107396. [CrossRef]
3. Sun, X.; Ni, Y.; Liu, C.; Wang, Z. A practical method for stability assessment of a damaged ship. *Ocean Eng.* **2021**, *222*, 108594. [CrossRef]
4. Martin, W.; Cohen, E.; Fish, R.; Shirley, P. Practical ray tracing of trimmed NURBS surfaces. *J. Graph. Tools* **2000**, *5*, 27–52. [CrossRef]
5. Guthe, M.; Balázs, A.; Klein, R. GPU-based trimming and tessellation of NURBS and T-Spline surfaces. *ACM Trans. Graph* **2005**, *24*, 1016–1023. [CrossRef]
6. Dokken, T. Finding intersections of B-spline represented geometries using recursive subdivision techniques. *Comput.-Aided Geom. Des.* **1985**, *2*, 189–195. [CrossRef]
7. Dokken, T.; Skytt, V.; Ytrehus, A.M. Recursive subdivision and iteration in intersections and related problems. In *Mathematical Methods in Computer Aided Geometric Design*; Academic Press: Cambridge, MA, USA, 1989; pp. 207–214.
8. Sederberg, T.W.; Nishita, T. Curve intersection using Bézier clipping. *Comput. Aided Des.* **1990**, *22*, 538–549. [CrossRef]
9. Efremov, A.; Havran, V.; Seidel, H.P. Robust and numerically stable Bézier clipping method for ray tracing NURBS surfaces. In Proceedings of the 21st Spring Conference on Computer Graph, Budmerice, Slovakia, 12–14 May 2005; pp. 127–135.
10. Lee, D.; Lee, S.S.; Park, B.J. 3-D geometric modeler for rapid ship safety assessment. *Ocean Eng.* **2004**, *31*, 1219–1230. [CrossRef]
11. Lu, C.; Lin, Y.; Ji, Z.; Chen, M. Ship hull representation with a single NURBS surface. In Proceedings of the ISOPE-2005 Conference: International Offshore and Polar Engineering Conference, Seoul, Republic of Korea, 19–24 June 2005.
12. Lu, C.; Lin, Y.; Ji, Z. Ship hull representation based on offset data with a single NURBS surface. *Ship Technol. Res.* **2007**, *54*, 81–88. [CrossRef]
13. Guo, J.; Zhang, Y.; Chen, Z.; Feng, Y. CFD-based multi-objective optimization of a waterjet- propelled trimaran. *Ocean Eng.* **2019**, *195*, 106755. [CrossRef]
14. Kuznecovs, A.; Ringsberg, J.W.; Johnson, E.; Yamada, Y. Ultimate limit state analysis of a double-hull tanker subjected to biaxial bending in intact and collision-damaged conditions. *Ocean Eng.* **2020**, *209*, 107519. [CrossRef]
15. Piegl, L.A.; Tiller, W. *The NURBS Book*, 2nd ed.; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1997.
16. Johnson, D.E.; Cohen, E. A framework for efficient minimum distance computations. In Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Leuven, Belgium, 20 May 1998; pp. 3678–3684.
17. Johnson, D.E.; Cohen, E. Distance extrema for spline models using tangent cones. In Proceedings of the GI'05: Proceedings of Graphics Interface 2005, Victoria, BC, Canada, 9–11 May 2005; pp. 169–175.
18. Ma, Y.L.; Hewitt, W.T. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Comput.-Aided Geom. Des.* **2003**, *20*, 79–99. [CrossRef]
19. Selimovic, I. Improved algorithms for the projection of points on NURBS curves and surfaces. *Comput.-Aided Geom. Des.* **2006**, *23*, 439–445. [CrossRef]
20. Chen, X.D.; Yong, J.H.; Wang, G.; Paul, J.C.; Xu, G. Computing the minimum distance between a point and a NURBS curve. *Comput.-Aided Des.* **2008**, *40*, 1051–1054. [CrossRef]
21. Chen, X.D.; Xu, G.; Yong, J.H.; Wang, G.; Paul, J.C. Computing the minimum distance between a point and clamped B-spline surface. *Graphical Models* **2009**, *71*, 107–112. [CrossRef]
22. Oh, Y.T.; Kim, Y.J.; Lee, J.; Kim, M.S.; Elber, G. Efficient point projection to freeform curves and surfaces. In Proceedings of the International Conference on Geometric Modeling and Processing, Castro Urdiales, Spain, 16–18 June 2010; pp. 192–205.
23. Oh, Y.T.; Kim, Y.J.; Lee, J.; Kim, M.S.; Elber, G. Continuous point projection to planar freeform curves using spiral curves. *The Visual Comp.* **2012**, *28*, 111–123. [CrossRef]
24. Li, X.W.; Wu, Z.N.; Hou, L.K.; Wang, L.; Yue, C.G.; Xin, Q. A geometric orthogonal projection strategy for computing the minimum distance between a point and a spatial parametric curve. *Algorithms* **2016**, *9*, 15. [CrossRef]
25. Quinlan, S. Efficient distance computation between non-convex objects. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), San Diego, CA, USA, 8–13 May 1994; pp. 3324–3329.
26. Dennis, J.E.; Schnabel, R.B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 1996.
27. Shacham, M. Numerical solution of constrained nonlinear algebraic equations. *Int. J. Numer. Methods Eng.* **1986**, *23*, 1455–1481. [CrossRef]

28. Zhu, K.G.; Shi, G.Y.; Liu, J. Improved flattening algorithm for NURBS curve based on bisection feedback search algorithm and interval reformation method. *Ocean Eng.* **2022**, *247*, 110635. [CrossRef]
29. Bertsekas, D.P. *Constrained Optimization and Lagrange Multiplier Methods*; Academic Press: Cambridge, MA, USA, 2014.
30. Ring, W.; Wirth, B. Optimization methods on Riemannian manifolds and their application to shape space. *SIAM J. Optim.* **2012**, *22*, 596–627. [CrossRef]
31. Huang, F.; Kim, H.Y.; Yang, C. A new method of ship bulbous bow generation and modification. In Proceedings of the Twenty-Fourth International Offshore and Polar Engineering Conference, Busan, Republic of Korea, 15–20 June 2014.
32. Absil, P.A.; Baker, C.G.; Gallivan, K.A. Trust-region methods on Riemannian manifolds. *Found. Comput. Math.* **2007**, *7*, 303–330. [CrossRef]
33. Absil, P.A.; Mahony, R.; Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*; Princeton University Press: Princeton, NJ, USA, 2008.
34. Ortega, J.M.; Rheinboldt, W.C. *Iterative Solution of Nonlinear Equations in Several Variables*; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 1987.
35. Deuflhard, P. *Newton Methods for Nonlinear Problems*; Springer: Berlin/Heidelberg, Germany, 2011.
36. Jiang, X.N.; Yan, L. Relevant integrals of NURBS and its application in hull line element design. *Ocean Eng.* **2022**, *251*, 111147. [CrossRef]
37. Nam, J.H.; Bang, N.S. A curve based hull form variation with geometric constraints of area and centroid. *Ocean Eng.* **2017**, *133*, 1–8. [CrossRef]
38. De Boor, C. *A Practical Guide to Splines*; Springer: New York, NY, USA, 1978.
39. McCartney, J.; Hinds, B.K.; Chong, K.W. Pattern flattening for orthotropic materials. *Comput.-Aided Des.* **2005**, *37*, 631–644. [CrossRef]
40. Takezawa, M.; Matsuo, K.; Maekawa, T. Control of lines of curvature for plate forming in shipbuilding. *Comput.-Aided Geom. Des.* **2019**, *75*, 101785.1–101785.14. [CrossRef]
41. Badr, E.; Sultan, A.; Abdallah, E.G. A Comparative Study among New Hybrid Root Finding Algorithms and Traditional Methods. *Mathematics* **2021**, *9*, 1306. [CrossRef]
42. Sabharwal, C.L. An Iterative Hybrid Algorithm for Roots of Non-Linear Equations. *Eng* **2021**, *2*, 7. [CrossRef]
43. Kim, J.; Noh, T.; Oh, W. An improved hybrid algorithm to bisection method and Newton-Raphson method. *Appl. Math. Sci.* **2017**, *11*, 2789–2797. [CrossRef]
44. Ye, Y. Combining Binary Search and Newton's Method to Compute Real Roots for a Class of Real Functions. *J. Complex.* **1994**, *10*, 271–280. [CrossRef]