| Team Number : | 2020250010057 |
|---|---|

| Problem Chosen : | A |
|---|---|

2020 APMCM summary sheet

By studying both zigzag and contour parallel hatch printing methods and considering the efficiency of the printing program, this paper proposes a reasonable plan to meet the efficiency requirements of actual industrial applications.

For Question 1, draw a single-layer contour pattern through the Curve Interpolation Model according to the data given in the question. For zigzag parallel hatch printing, use the Spline Polynomial Root Finding Model. Introduce a set of straight lines parallel to the x-axis, intersect the parallel lines and the closed curve and sort them from the small to the large. Then print in the order of odd and even intersected points. For contour parallel hatch printing, the Plane Affine Transformation Model is used to realize the gradual shrinkage of the target curve.

For Question 2, the research object has changed from a single-layer contour pattern to a more complicated nested multilayer contour pattern. For zigzag parallel hatch printing, we use the same model for Question 1. For the contour parallel hatch printing, on the basis of the Plane Affine Transformation Model in Question 1, we set the direction of the inner nested closed curve to the opposite direction of the outer curve. Then the remaining algorithm steps keep the same in Question 1.

For question 3, in order to improve the performance and efficiency of the algorithm, we used the Linear Interpolation model and the Contour Model to improve the two methods in Question 1 and Question 2.

Finally, analyze and compare algorithm performance before and after the improvements. The experimental results indicate that the improvements greatly reduce the time complexity of the algorithms.

**Key Words:** interpolation, affine transformation, contour line, closed curve, printing

# Contents

# 1. Problem Restatement

## 1.1 Background

With the development of the times, the application of laser in real life has become more and more extensive. It can be used in industrial processing, identifying objects, making weapons and laser treatment. It is a kind of applied optics with great potential, and its application scope will continue to expand in the process of continuous exploration and invention. Laser marking machine is one of the applications of laser. It uses two methods: zigzag parallel hatch or contour parallel hatch to mark LOGO, characters, images, etc. on the product surface.

## 1.2 Problem Summary

1) Based on the data in Attachment 1, to realize the zigzag parallel and contour parallel hatch of the single-layer contour pattern under the two groups of input parameters which are 1mm and 0.1mm of the internal contraction of boundary distance and the hatch line spacing. And calculating the total length of hatching lines through two methods separately, the number of horizontal lines of zigzag parallel hatch, the number of circles of contour parallel hatch and the ratio of the average running time of the program under the two parameter groups. Importantly, only the hatch in horizontal direction (0° degree) is considered for zigzag parallel hatch.

2) Based on the data in Attachment 2, to realize the zigzag parallel and contour parallel hatch of the mutually nested multi-layer contour patterns under the two groups of input parameters which are 1mm and 0.1mm of the internal contraction of boundary distance and the hatch line spacing. And calculating the total length of hatching lines through two methods separately, the number of horizontal lines of zigzag parallel hatch, the number of circles of contour parallel hatch and the ratio of the average running time of the program under the two parameter groups. Importantly, only the hatch in horizontal direction (0° degree) is considered for

zigzag parallel hatch.

3) When performing laser marking, contour lines of any shape have extremely high requirements on the operating efficiency of the program. Therefore, improving the operating efficiency of the program is a very necessary step. Please analyze the performance and efficiency of the existing algorithm, and optimize it so that it can meet the efficiency requirements of actual industrial applications.

## 2. Problem Analysis

1) When using the data in Attachment 1 to perform two printing methods, firstly considering drawing the given scatter points, approximating the outline of the given scatter points through interpolation and then realizing two printing methods for the obtained closed curve. For the zigzag parallel hatching printing method, it is considered to be realized by multiple sets of parallel straight lines at a fixed interval parallel to the x-axis, for example, 0.1 mm is used as the fixed interval between parallel lines. And for the determination of multiple sets of parallel lines, using equidistant different y values to connect multiple points corresponding to the closed curve in a certain order. For the contour parallel section line printing method, according to the meaning of the question, we need to find the hatching lines of the contour parallel hatch, then we can get the hatching lines through considering the parallel relationship between the line segments after discretizing the boundary curve and determine the distance between the parallel lines by the internal contraction of boundary distance. Naturally, the outline shrinking direction of the closed curve as a whole is determined. The total length of hatching lines through two methods separately, the number of horizontal lines of zigzag parallel hatch, the number of circles of contour parallel hatch and the ratio of the average running time of the program under the two parameter groups can be obtained through the principle of next event advancement, that is, to record the data generated when each event occurs.

2) In the second question, on the basis of the first question, there are mutually nested multi-layer contour patterns. The basic idea remains the same. For the zigzag parallel hatch printing method, the problem can be solved by determining the connection direction of the corresponding points of the closed curve at the nesting position for the same y value. For the contour parallel hatch printing method, considering that the nested closed curve as a complex contour and after determining the direction of the closed curve as the negative direction (the direction of the inner boundary curve is opposite to the outside) to get the hatch lines by using the same idea in the first question. The calculation of the remaining data also refers to the method of the first question.

3) The third question requires that the algorithm be improved on the basis of the first and second questions to meet the requirements of industrial production. Consider improving the interpolation method and the contour convergence method to speed up the hatch circle lines' generation.

## 3. Model Hypothesis

1) When building the Parallel Line Coverage of Odd and Even Points Model, it is assumed that the tangent of a straight line and a closed curve is not considered.
2) Assuming that every time the program runs, the state of the computer itself differs very little and can be approximately ignored.

## 4. Symbol Explanation

| | |
|---|---|
| g-i sequence | a sequence of numbers from small to large |
| $A^e$ | the set of positive even numbers |
| $(A_{oi}, A_{e(i+1)})$ | the adjacent line segments with odd-numbered points as the head and even-numbered points as the end |

× a cross product between vectors,

if $\vec{A} = (x_1, y_1)$ , $\vec{B} = (x_2, y_2)$, then $\vec{A} \times \vec{B} = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}$

# 5. Establishment and Solution of the Model

## 5.1 Question1

Establishing a mathematical model to realize the zigzag parallel hatch printing method and the contour parallel hatch printing method, and calculating some parameters in the implementation process. Among them, the ratio of the average running time of the program is the ratio of parameter groups (2) to (1).

### 5.1.1 Model establishment and solution of zigzag parallel hatch method

1) Model Building

a. Curve Interpolation Model

Given a closed curve, discretize the curve firstly and replace the curve with a large number of points on the curve with a small enough spacing, denoted as $(x_1, y_1)\ldots(x_n, y_n)$. Introduce a new variable $t$ to change the point $(x, y)$ on the two-dimensional plane into the point $(x, y, t)$ on the three-dimensional space, thereby obtaining a list of scattered points in the three-dimensional space $(x_1, y_1, t_1) \cdots (x_n, y_n, t_n)$. Then, perform cubic spline interpolation on variables $t, x$ and variables $t, y$ respectively to obtain cubic spline interpolation polynomials $x = S_1(t), y = S_2(t)$. After that, using these two polynomials to approximate the actual boundary curve $x = x(t), y = y(t)$. The corresponding interpolation node $t$ can be obtained by solving $S_2(t) = y$, and $x = S_2(t)$ can be obtained accordingly.

The model introduces a new variable $t$ to establish the relationship between the horizontal and vertical coordinates of the closed curve which is inconvenient to directly write the curve equation. Its geometric meaning is the abscissa of the intersection of a straight line parallel to the x-axis and the closed curve in the

coordinate system.

b. Parallel Line Coverage of Odd and Even Points Model

Without considering the tangency of a line parallel to the x-axis with the closed curve, recording the intersection of the line and the closed curve as $A_1$, $A_2 \cdots A_n$, and the g-i sequence of the abscissa of the intersection is $\{x_1, x_2, x_3 \cdots x_n\}$, $n \in A^e$. Then, connecting line segments sequentially $(A_{oi}, A_{e(i+1)})$, and there are parallel lines without an array can approximately cover a closed figure. Among them, the g-i sequence represents a sequence of numbers from small to large; $A^e$ represents the set of positive even numbers; $(A_{oi}, A_{e(i+1)})$ represents the line segment connected by two adjacent points, and it must start with an odd point and end with an even point. The derivation of the model requires theorem 1.

**Theorem 1:** Without considering the tangency of a line parallel to the x-axis and tangent to the closed curve, the number of intersections between a line parallel to the x-axis and the closed curve in the coordinate system must be an even number.

**Proof:** According to the Jordan's Lemma in complex function [1], we know that a closed curve can divide a plane into two parts, namely the inside of the curve and the outside of the curve. The abscissa of the intersection, which are obtained by a straight line parallel to the x-axis and a closed curve in the coordinate system are arranged in order from small to large. And recording the intersection points as $A_1$, $A_2 \cdots A_n$. Take $A_0$ on the left side of $A_1$ and $A_{n+1}$ on the right side of $A_n$, then the line segment $(A_0, A_1)$ and the line segment$(A_n, A_{n+1})$ must be located outside the closed curve.

If the line segment $(A_1, A_2)$ is located outside the curve, it can be inferred that the point $A_1$ is the tangent point between the straight line and the closed curve based on the line segment $(A_0, A_1)$ located outside the closed curve. This contradicts the assumption, so the line segment $(A_1, A_2)$ must locate inside the

curve;

If the line segment $(A_2, A_3)$ is located inside the curve, it can be inferred that the point $A_2$ is the tangent point between the straight line and the closed curve based on the line segment $(\mathrm{A}_1, \mathrm{A}_2)$ located inside the closed curve. This contradicts the assumption, so the line segment $(A_2, A_3)$ must locate outside the curve;

⋮

⋮

If the line segment $(A_{n-1}, A_n)$ is located outside the curve, it can be inferred that the point $A_n$ is the tangent point between the straight line and the closed curve based on the line segment $(A_n, A_{n+1})$ located outside the closed curve, which contradicts the assumption. So the line segment $(A_{n-1}, A_n)$ must be located inside the curve;

In summary, we can conclude that n must be a positive even number by using induction. ■

Then successively connect the adjacent line segments with odd-numbered points as the head and even-numbered points as the end, and perform a pan operation on the set of line segments obtained above. These parallel lines can approximately cover the closed curve when the translation distance is small enough.

c. Spline Polynomial Root Finding Model

Since the intersection point is required in the curve interpolation model, the spline interpolation polynomial is introduced here to find the roots:

Given that the spline polynomial $y = S_1(t)$, to find all the real roots of the equation $S_1(t) = y$. The idea is to divide the domain of $t$ into sufficiently small intervals and then use the zero-existence theorem in mathematical analysis [3]. Determine whether it is a rooted interval on each cell; use the dichotomy method to solve the root of the equation on the interval on each interval after determining all

rooted intervals.

2) Problem Solving

Based on the principle of the zigzag parallel hatch method and the model established, get the printing algorithm as follows:

**STEP 1**: When printing each group of parallel lines, the Curve Interpolation Model is used to determine the points on the closed curve corresponding to different y value levels. And then use Parallel Line Coverage of Odd and Even Points Model to connect the obtained points to get a set of line segments.

**STEP 2**: By changing the value of y, generating multiple sets of parallel lines to cover the closed curve. And $y_i = y_{i-1} - b_0, i = 2, 3, \cdots, n$ (In this question, the value of $b_0$ is 1mm and 0.1mm).

To sum up that the process of successively identifying each group of line segments is equivalent to the parallel line printing process. So far, we have realized the printing process of the zigzag parallel hatch method.

3) Results for Problem Solving

a. Since the printing curve is too dense to be intuitively recognized when the internal contraction of boundary distance is 1mm or 0.1mm, the parallel printing curve when the internal contraction of boundary distance and hatch line spacing is 1cm is given, as shown in **Figure 1**.

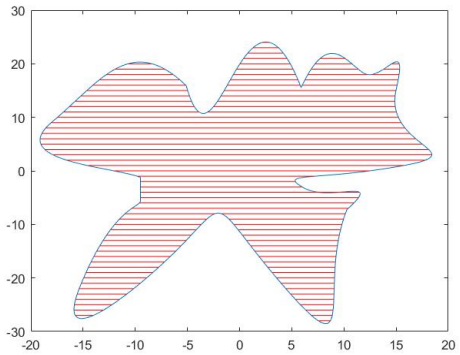


**Figure 1 The parallel printing curve at the level of 1cm**

b. When the internal contraction of boundary distance and hatch line spacing are 1mm and 0.1mm， given its print curve in **Figure 2** and **Figure 3**.

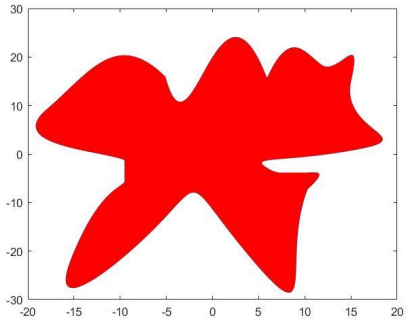


**Figure 2 1mm**

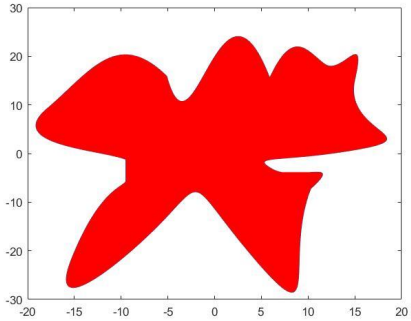


**Figure 3 0.1mm**

c. Perform five experiments under the two sets of parameters of 1mm and 0.1mm respectively, and record the program running time, the number of parallel lines and the total length of parallel lines during the experiment. **Table 1** and **Table 2** are obtained.

**Table 1 Summary table of experimental data at the level of 1mm**

| **Summary table of experimental data when the internal contraction of boundary distance and hatch line spacing are 1mm** | | | |
|---|---|---|---|
| **Experiment number** | Program running time (s) | Number of parallel lines (pieces) | Total length of parallel lines (cm) |
| **1** | 30.235 | 526 | 10130 |
| **2** | 27.663 | 526 | 10130 |
| **3** | 32.145 | 526 | 10130 |
| **4** | 29.135 | 526 | 10130 |
| **5** | 28.156 | 526 | 10130 |
| **Average** | 29.4668 | 526 | 10130 |

**Table 2 Summary table of experimental data at the level of 0.1mm**

| **Summary table of experimental data when the internal contraction of boundary distance and hatch line spacing are 0.1mm** | | | |
|---|---|---|---|
| **Experiment number** | Program running time (s) | Number of parallel lines (pieces) | Total length of parallel lines (cm) |
| **1** | 232.12 | 5260 | 101280 |
| **2** | 236.47 | 5260 | 101280 |
| **3** | 222.15 | 5260 | 101280 |

| | | | |
|---|---|---|---|
| **4** | 228.78 | 5260 | 101280 |
| **5** | 238.56 | 5260 | 101280 |
| **Average** | 231.616 | 5260 | 101280 |

According to **Table 1**, when the internal contraction of boundary distance and hatch line spacing is 1mm, the average printing time is about 29.47s, the number of horizontal lines of zigzag parallel hatch is 526 and the total length of hatching lines is 10130cm;

According to **Table 2**, when the internal contraction of boundary distance and hatch line spacing is 0.1mm, the average printing time is about 231.62s, the number of horizontal lines of zigzag parallel hatch is 5260 and the total length of parallel lines is 101280cm. Combining the data in the above two tables, the ratio of the average running time of parameter group (2) to parameter group (1) is 231.62/29.47= 7.86

### 5.1.2 Model Establishment and Solution of Contour Parallel Hatch Method

1) Model Building

a. Planar Affine Transformation Model

Given any vector $\vec{A} = (x, y)$ on the plane, the vector obtained by rotating the angle $\theta$ clockwise is $\overrightarrow{A'} = (x', y')$. And the affine transformation performed [6] is: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$. Among them, an arrow on a capital letter indicates a vector.

The model is based on the coordinates of the vector on the plane, and through the transformation of the vector coordinates to realize the vector rotation.

b. Line Intersection Model

Suppose there are two line-segments AB and CD on the plane. The necessary and sufficient conditions for the intersection of line segments AB and CD are $\left(\overrightarrow{DA} \times\right.$

$\overrightarrow{DC}) \cdot (\overrightarrow{DC} \times \overrightarrow{DB}) < 0$ and $(\overrightarrow{AC} \times \overrightarrow{AB}) \cdot (\overrightarrow{AB} \times \overrightarrow{AD}) < 0$ . Among them, an arrow on a capital letter indicates a vector; "×" indicates a cross product between vectors, that is, if $\vec{A} = (x_1, y_1)$ , $\vec{B} = (x_2, y_2)$, then $\vec{A} \times \vec{B} = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}$. This model is used to determine whether two line-segments on the plane intersect. Two important lemmas are needed to arrive at this model.

**Lemma 1:** The necessary and sufficient conditions for the intersection of two line segments are: ①The two points C and D are on both sides of the line AB; ②The two points A and B are on both sides of the line CD.

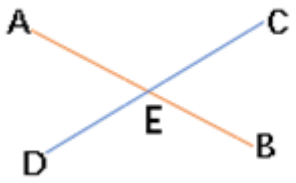


**Figure 4 Line Intersection**

**Proof:**

According to **Figure 4,**

Sufficiency: It can be inferred from ① that the line segment CD and the straight line AB intersect at a single point E, and from ② it can be seen that the line segment AB intersects the straight line CD at a single point. Because the straight line AB intersects the straight line CD, and they have only one intersection. In summary, the line segment AB intersects the line segment CD and there is a unique intersection E.

Necessity is clearly established. ■

**Lemma 2:** $\vec{A} \times \vec{B} < 0$ means $\vec{B}$ is in the clockwise direction of $\vec{A}$ (that is, $\vec{A}$ can be rotated clockwise by a certain angle $\theta$ and then it is in the same direction as $\vec{B}$, $\theta \in (0, \pi)$); $\vec{A} \times \vec{B} > 0$ means that $\vec{B}$ is in the counterclockwise direction of $\vec{A}$ (that is, $\vec{A}$ can be rotated counterclockwise by a certain angle $\beta$ and then in the

same direction as $\vec{B}, \beta \in (0, \pi)$) [7].

This Lemma is the basic conclusion of plane geometry and is not directly quoted here for proof. Let's prove the rationality of the model now.

**Proof:**

Necessity: From Lemma 1, we know that A and B are on both sides of $\overrightarrow{DC}$, and from Lemma 2, we can get that $\overrightarrow{DA} \times \overrightarrow{DC}$ and $\overrightarrow{DC} \times \overrightarrow{DB}$ have the opposite signs, so $\left(\overrightarrow{DA} \times \overrightarrow{DC}\right) \cdot \left(\overrightarrow{DC} \times \overrightarrow{DB}\right) < 0$ . And the same can be obtained $\left(\overrightarrow{AC} \times \overrightarrow{AB}\right) \cdot \left(\overrightarrow{AB} \times \overrightarrow{AD}\right) < 0$.

Sufficiency: From $\left(\overrightarrow{DA} \times \overrightarrow{DC}\right) \cdot \left(\overrightarrow{DC} \times \overrightarrow{DB}\right) < 0$ and the Lemma 2 can see that two points A and B are located on both sides of the straight line CD; from $\left(\overrightarrow{AC} \times \overrightarrow{AB}\right) \cdot \left(\overrightarrow{AB} \times \overrightarrow{AD}\right) < 0$ and the Lemma 2 can see that the two points C and D are on both sides of the line AB.

Then we know from the Lemma 1 that the line AB and the line CD intersect. This explains the rationality of the model. ■

2) Problem Solving

For the contour parallel hatch printing method, we need to find the hatch curve lines parallel to the closed curve boundary according to the meaning of the question. Then we can consider the parallel relationship between the line segments after discretizing the boundary curve and determine the distance between parallel lines by the internal contraction of boundary distance, thus the hatch curve lines obtained. The specific method for determining the hatch curve lines is as follows:

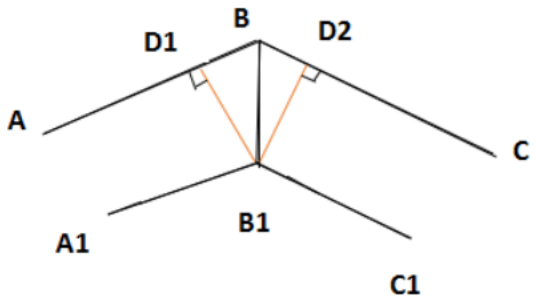


**Figure 5**

**STEP 1:** Take two parallel line segments corresponding to the original boundary curve and the hatch curve lines to study the properties of the hatch curve lines. As shown in the **Figure 5**, the line segment $A_1B_1$ is parallel to the line segment AB, the line segment $C_1B_1$ is parallel to the line segment CB, and the distance from point $B_1$ to the line segment AB and CD is equal (the extension of the line segment $BB_1$ is the angle bisector of $\angle$ABC according to the knowledge of plane geometry). Then, rotate the vector $\overrightarrow{BC}$ clockwise by the angle $\angle B_1BC$ to get the vector $\vec{a}$ which has same direction with $\overrightarrow{BB_1}$ (based on the Plane Affine Transformation Model). And after stretching and transforming, $\vec{a}$ can completely coincide with $\overrightarrow{BB_1}$ (that is, we can find the vector $\overrightarrow{BB_1}$ on the basis of the vector $\overrightarrow{AB}$, $\overrightarrow{BC}$). In the end, calculating the coordinates of $B_1$ to determine the hatch curve lines through $\overrightarrow{OB_1} = \overrightarrow{OB} + \overrightarrow{BB_1}$.

**STEP 2:** However, the hatch curve lines obtained at this time will appear as shown in **Figure 6**.

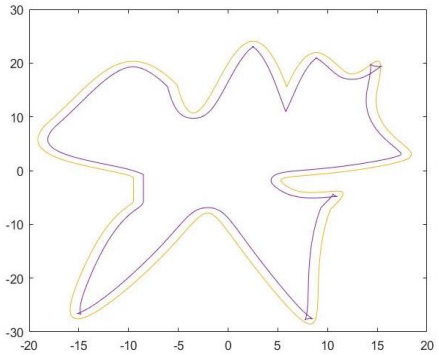


**Figure 6 Abnormal Situation**

In **Figure 6**, there will be situations where the hatch circle lines produce coincident points and intersect with the original boundary curve. Therefore, there are two ways to correct the shadow curve based on the above methods.

a. In the correction process, we need to use the Line Segment Intersection Model. Because the boundary curve and the hatch curve are essentially obtained by approximating the contours of the scattered points with interpolation, and the scattered points are connected by segments, so we can determine the line segments that produce the coincident points and intersect the boundary by the Line Segment Intersection Model after obtaining the hatch curve. Then after removing these line segments, we can get the hatch curve we need.

b. From the knowledge of topology [4], the curve has more than one convergence point during the inward contraction process. Therefore, during the overall contraction process, it is considered that the curve starts to converge with different convergence points as the center when the shadow curve appears more obvious separation (here, consider the distance between adjacent points greater than 1cm ). Then, shrink the curve in blocks is needed.

After solving the above problems, the Contour Parallel Hatch method can be realized through the continuous shrinking of the hatch curve.

3) Results for Problem Solving

a. Since the curve is too dense to be seen intuitively when the internal contraction of boundary distance and hatch line spacing are 1mm and 0.1mm, we first give the partial printing curve when the internal contraction of the boundary distance is 0.5cm as shown in **Figure 7**.

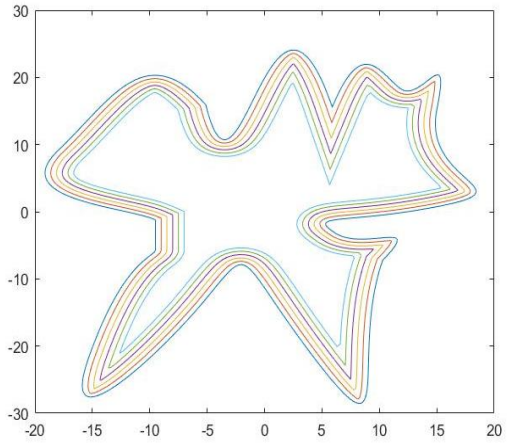


**Figure 7 Effects with the internal contraction of the boundary distance as 0.5cm**

b. Printing curve when the internal contraction of the boundary distance and hatch line spacing are 1mm and 0.1mm, as shown in **Figure 8** and **Figure 9**.

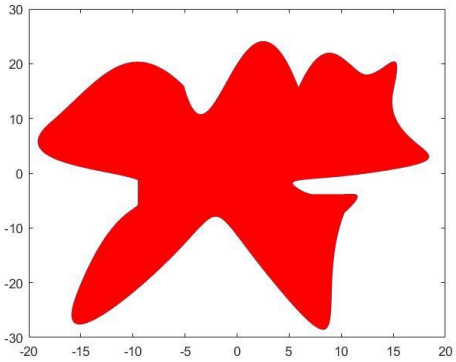


**Figure 8 1mm**

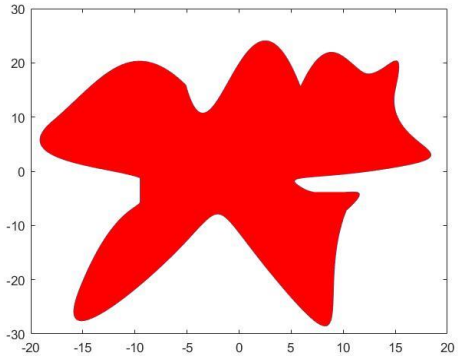


**Figure 9 0.1mm**

c. Perform five experiments under the two sets of parameters of 1mm and 0.1mm respectively, and record the program running time, the number of circles of contour parallel hatch and the total length of parallel lines during the experiment. **Table 3** and **Table 4** are obtained.

**Table 3 Summary table of experimental data at the level of 1mm**

| **Summary table of experimental data when the internal contraction of boundary distance and hatch line spacing are 1mm** | | | |
|---|---|---|---|
| **Experiment number** | Program running time (s) | The number of circles of contour parallel hatch(pieces) | Total length of parallel lines (cm) |
| **1** | 41.563 | 32 | 9326 |

| | | | |
|---|---|---|---|
| **2** | 42.636 | 32 | 9326 |
| **3** | 40.467 | 32 | 9326 |
| **4** | 39.446 | 32 | 9326 |
| **5** | 38.562 | 32 | 9326 |
| **Average** | 40.5348 | 32 | 9326 |

**Table 4 Summary table of experimental data at the level of 0.1mm**

| **Summary table of experimental data when the internal contraction of boundary distance and hatch line spacing are 0.1mm** | | | |
|---|---|---|---|
| **Experiment number** | Program running time (s) | The number of circles of contour parallel hatch(pieces) | Total length of parallel lines (cm) |
| **1** | 380.67 | 316 | 92830 |
| **2** | 375.34 | 316 | 92830 |
| **3** | 377.88 | 316 | 92830 |
| **4** | 378.15 | 316 | 92830 |
| **5** | 377.58 | 316 | 92830 |
| **Average** | 337.924 | 316 | 12260 |

According to **Table 3**, when the internal contraction of boundary distance and hatch line spacing is 1mm, the average printing time is about 40.535s, the number of circles of contour parallel hatch is 32 and the total length of parallel lines is 9326cm;

According to **Table 4**, when the internal contraction of boundary distance and hatch line spacing is 0.1mm, the average printing time is about 337.924s, the number of circles of contour parallel hatch is 316 and the total length of parallel lines is 12260cm. Combining the data in the above two tables, the ratio of the average running time of parameter group (2) to parameter group (1) is about 337.924/40.535= 8.337.

## 5.2 Question 2

For the two printings of nested multi-layer contour patterns, the same mathematical model as in question 1 is still used. Calculate some parameters in the implementation process. Among them, the ratio of the average running time of the

program is the ratio of parameter groups (2) to (1).

### 5.2.1 The Solution of the Zigzag Parallel Hatch Method Model

1) Problem Solving

Compared with Question 1, this question has changed from the original single-layer contour pattern to the nested multilayer contour pattern. The basic solution idea remains unchanged. The model and solution method established by the first question are used to realize the Printing in this case:

**STEP 1**: When printing each group of parallel lines, determine the points on the closed curve corresponding to different value levels through the curve interpolation model, and then use the Parallel Line Coverage of Odd and Even Points Model to connect the obtained points to get a group line segments.

**STEP 2**: By changing the value y, multiple sets of parallel lines are generated to cover the closed curve, $y_i = y_{i-1} - b_0, i = 2, 3, \cdots, n$ (in this question, the values are 1mm and 0.1mm).

2) Results for Problem Solving

a. When the distance between parallel lines is 1mm and 0.1mm, the printing curve is too dense to be intuitively recognized. Now the parallel printing curve when the internal contraction of boundary distance and hatch line spacing is 1cm is given, as shown in **Figure 10**.

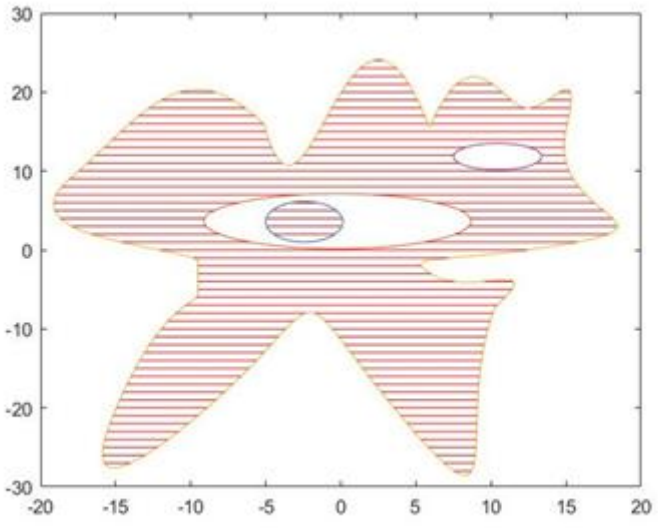


**Figure 10 Effect with the internal contraction of the boundary distance as 1cm**

b. When the distance between parallel lines are 1mm and 0.1mm, the printing curve is shown in **Figure 11** and **Figure 12**.

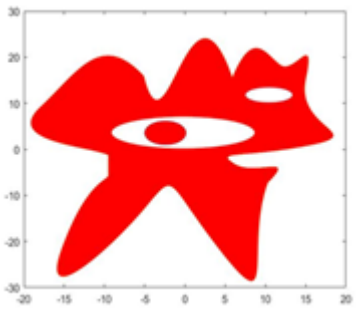

**Figure 11 1mm**

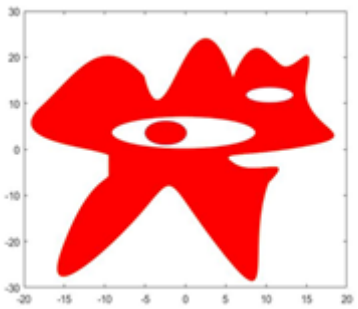

**Figure 12 0.1mm**

c. Perform five experiments under two sets of parameters of 1mm and 0.1mm, and record the program running time, the number of horizontal lines and the total length of parallel lines during the experiment, and get **Table 5** and **Table 6**.

**Table 5 Summary table of experimental data at the level of 1mm**

| **Summary table of experimental data when the internal contraction of boundary distance and hatch line spacing are 1mm** | | | |
|---|---|---|---|
| **Experiment number** | Program running time (s) | Number of parallel lines (pieces) | Total length of parallel lines (cm) |
| **1** | 40.143 | 526 | 9205 |
| **2** | 37.691 | 526 | 9205 |
| **3** | 42.635 | 526 | 9205 |
| **4** | 39.935 | 526 | 9205 |
| **5** | 38.752 | 526 | 9205 |
| **Average** | 39.8312 | 526 | 9205 |

**Table 6 Summary table of experimental data at the level of 0.1mm**

| **Summary table of experimental data when the internal contraction of boundary distance and hatch line spacing are 0.1mm** | | | |
|---|---|---|---|
| **Experiment number** | Program running time (s) | Number of parallel lines (pieces) | Total length of parallel lines (cm) |
| **1** | 383.12 | 5260 | 92036 |
| **2** | 378.76 | 5260 | 92036 |
| **3** | 386.45 | 5260 | 92036 |
| **4** | 381.28 | 5260 | 92036 |
| **5** | 376.25 | 5260 | 92036 |

| | | | |
|---|---|---|---|
| **Average** | 381.172 | 5260 | 92036 |

According to **Table 5**, when the internal contraction of boundary distance and hatch line spacing is 1mm, the average printing time is about 39.83s, the number of horizontal lines of zigzag parallel hatch is 526 and the total length of hatching lines is 9205cm;

According to **Table 6**, when the internal contraction of boundary distance and hatch line spacing is 0.1mm, the average printing time is about 381.17s, the number of horizontal lines of zigzag parallel hatch is 5260 and the total length of parallel lines is 92036cm. Combining the data in the above two tables, the ratio of the average running time of parameter group (2) to parameter group (1) is 381.17/39.83= 9.57.

### 5.2.2 Establishment and Solution of Contour Parallel Hatch Method Model

1) Problem Solving

According to the nature of the complex boundary curve and based on the Plane Affine Transformation Model in question 1, set the direction of the inner nested closed curve to the opposite direction of the outer curve. The remaining steps are the same as the first question.

2) Results for Problem Solving

a. When the internal contraction of boundary distance and hatch line spacing is 1mm and 0.1mm, the curve is too dense to be intuitive. The partial printing curve when the internal contraction boundary distance is 0.2cm is given, as shown in **Figure 13**.

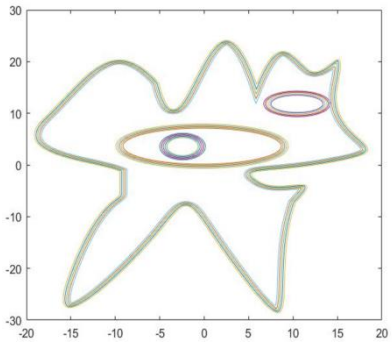


**Figure 13 Effects with the internal contraction of the boundary distance as 0.2cm**

b. When the distance between parallel lines are 1mm and 0.1mm, the printing curve is shown in **Figure 14** and **Figure 15**.

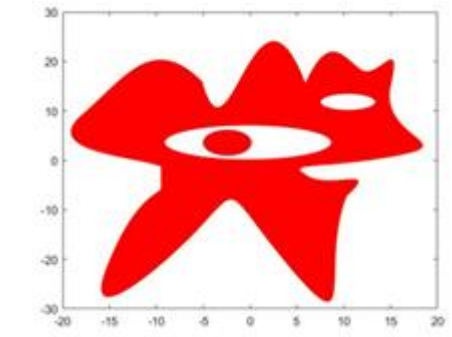

**Figure 14 1mm**

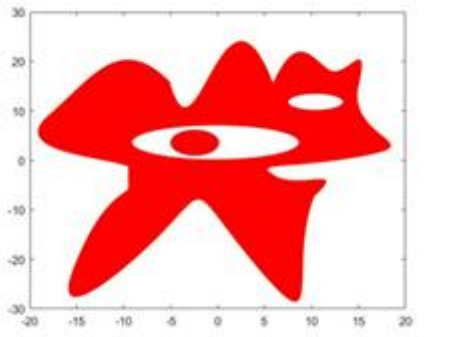

**Figure 15 0.1mm**

c. Perform five experiments under the two sets of parameters of 1mm and 0.1mm, and record the program running time, the number of circles of contour parallel hatch, and the total length of the parallel lines during the experiment.

**Table 7 Summary table of experimental data at the level of 1mm**

| **Summary table of experimental data when the internal contraction of boundary distance and hatch line spacing are 1mm** | | | |
|---|---|---|---|
| **Experiment number** | Program running time (s) | The number of circles of contour parallel hatch(pieces) | Total length of parallel lines (cm) |
| **1** | 51.373 | 28 | 9146 |
| **2** | 54.331 | 28 | 9146 |
| **3** | 49.287 | 28 | 9146 |
| **4** | 52.176 | 28 | 9146 |
| **5** | 48.476 | 28 | 9146 |
| **Average** | 51.1286 | 28 | 9146 |

**Table 8 Summary table of experimental data at the level of 1mm**

| **Summary table of experimental data when the internal contraction of boundary distance and hatch line spacing are 0.1mm** | | | |
|---|---|---|---|
| **Experiment number** | Program running time (s) | The number of circles of contour parallel hatch(pieces) | Total length of parallel lines (cm) |
| **1** | 488.26 | 272 | 89748 |

| | | | |
|---|---|---|---|
| **2** | 482.57 | 272 | 89748 |
| **3** | 491.36 | 272 | 89748 |
| **4** | 487.18 | 272 | 89748 |
| **5** | 483.77 | 272 | 89748 |
| **Average** | 486.628 | 272 | 89748 |

According to **Table 7**, when the internal contraction of boundary distance and hatch line spacing is 1mm, the average printing time is about 51.13s, the number of circles of contour parallel hatch is 28 and the total length of parallel lines is 9146cm;

According to **Table 8**, when the internal contraction of boundary distance and hatch line spacing is 0.1mm, the average printing time is about 486.63s, the number of circles of contour parallel hatch is 272 and the total length of parallel lines is 89748cm. Combining the data in the above two tables, the ratio of the average running time of parameter group (2) to parameter group (1) is about 486.63/51.13= 9.52.

## 5.3 Question 3

### 5.3.1 Improvement of the Model

Based on the model established by the first question, an improved method is proposed to reduce the time complexity of the algorithm. The improvement measures are as follows:

a. Linear Interpolation Model

For the zigzag parallel hatch method, the spline polynomial interpolation is improved to linear interpolation, because the essence of linear interpolation is to connect two adjacent points with a line segment. Then we don’t need to use the dichotomy method when finding the intersection point. Instead, we can use the nature of the line segment to directly determine the zero point, which greatly reduces the time complexity of the algorithm.

b. Contour Model

For the contour parallel hatch method, firstly introduce the theorem of topology.

**Theorem 2**: Convex curve can converge to a point on the plane uniformly according to the contour line [5].

So the contour model can be derived from **Theorem 2**: Divide the plane figure into some convex areas, then determine the center of each convex area. Assign the height of the boundary point to 0 and the height of the center point to 1. By three-dimensional curve fitting, contour lines are drawn on the curve and then projected onto a two-dimensional plane to determine the shadow line of the boundary curve.

### 5.3.2 Problem Solving

According to the Linear Interpolation Model and Contour Model to improve the zigzag parallel hatch method and the contour parallel hatch method. Obtain the average experimental data under different parameters through multiple experiments to check whether the algorithm is improved.

### 5.3.3 Results for Problem Solving

a. Carry out the experiment under multiple sets of parameters, record the average running time of the linear interpolation program and the average running time of the spline interpolation program during the experiment, and get **Table 9**.

**Table 9**

| Internal contraction of boundary distance and hatch line spacing (mm) | Average running time of linear interpolation model program (s) | Average running time of spline interpolation model program (s) |
|---|---|---|
| **1** | 16.225 | 30.235 |
| **0.8** | 20.136 | 37.503 |
| **0.6** | 26.713 | 50.627 |
| **0.4** | 41.113 | 76.364 |
| **0.2** | 81.362 | 150.167 |

b. Carry out the experiment under multiple sets of parameters, record the average running time of the Contour Model program and the Parallel Contour Model program during the experiment, and get **Table 10**.

**Table 10**

| Internal contraction of boundary distance and hatch line spacing (mm) | Average running time of contour model program (s) | Average running time of parallel contour model program (s) |
|---|---|---|
| **1** | 12.146 | 41.563 |
| **0.8** | 15.372 | 52.136 |
| **0.6** | 21.667 | 68.443 |
| **0.4** | 31.492 | 106.142 |
| **0.2** | 57.113 | 206.762 |

Through the analysis of the above data, the newly established Linear Interpolation Model and Contour Model indeed play a role in reducing the time complexity of the algorithms.

# 6. Model Evaluation

## 6.1 Model Advantages

1) Parallel Line Coverage of Odd and Even Points Model ensures that the program can quickly extract the line segments located inside the curve, greatly improving the rate of hatch curve lines' generation.

2) The Plane Affine Transformation Model converts the problem of finding the angle bisector into a vector rotation transformation, which solves the problem of equations that need to be solved in the usual model of finding the angle bisector.

3) The Line Intersection Model converts the point on the outside of the curve and the key problem of curve generation into judging whether the line segments intersect, which solves the problem of whether the equation needs to be solved in the traditional curve-line intersection model.

4) The Linear Interpolation Model changes the interpolation method on the basis of the spline polynomial interpolation model, which solves the problem of the dichotomy that needs to be used when the spline polynomials intersect, and greatly reduces the algorithm time complexity.

5) The Contour Model uses contour lines to ensure that the generated shadow

lines are all inside the curve, which solves the problem of judging whether or not intersecting line by line segment when the contour shrinks.

6) The solution method is universal. Two different methods can be used to generate shadow lines for any kind of boundary curve, which has a strong promotion significance.

## 6.2 Model Shortcomings

1) The contour shrink program cannot be recognized immediately after the curve is clearly separated. It needs to be run 1 or 2 times to recognize it. However, the solution of this problem requires a lot of data attempts and cannot be solved in a short time.

2) When a part of the curve is convex to a certain degree (such as an acute angle), we need to smooth the part of the boundary curve (insert some points), otherwise it will cause this part of the curve to be automatically truncated during the program running , But it is not common for the curve to be extremely convex in real life.

# 7. References

[1]钟玉泉，复变函数论[M],北京：高等教育出版社，2012，26-27；

[2]金一庆，陈越，王冬梅，数值方法[M],北京：机械工业出版社，2000，10-14；

[3]欧阳光中，朱学炎，金福临，陈传璋，数学分析上册[M]，北京：高等教育出版社，2007，53-65；

[4]尤承业，基础拓扑学[M],北京：北京大学出版社，1997，24-36-

[5]沈继红，数学建模[M],哈尔滨：哈尔滨工程大学出版社，1996，34-75；

[6]刘琼荪，数学实验[M],北京：高等教育出版社，2004，26-47

[7]杨桂元，黄己立，数学建模[M]，合肥：中国科技大学出版社，2008，18-67

# 8. Appendix

The appendix programs do not contain data import statements, see supporting materials for details.

**1）Question 1**

**a. The program of zigzag parallel hatch method**

yasai2.m

```
t=linspace(0,10,1195);
pp=csape(t,x(:,2));
pp1=csape(t,x(:,1));
t0=linspace(0,10,2000);
plot(x(:,1),x(:,2));
hold on
x1=[];
deta1=0.001;
deta2=0.001;
for y= -30:0.1:30
    x2=[];
    x0=[];
for i=1:1999
    if (ppval(pp,t0(i))-y)*(ppval(pp,t0(i+1))-y)<0
```

```
            x0=[x0;t0(i),t0(i+1)];
        end
    end
    for j=1:size(x0,1)
        if size(x0,1)==0
            break
        end
        a1=x0(j,1);
        b1=x0(j,2);
        c1=(a1+b1)*0.5;
        for k=1:100000
            if abs(ppval(pp,c1)-y)<deta1 && b1-a1<deta2
                x2(j)=c1;
                break
            end
            if k==1000000
                x2(j)=inf;
                break
            end
            if (ppval(pp,a1)-y)*(ppval(pp,c1)-y)<0
                a1=a1;
                b1=c1;
```

```
            c1=(a1+b1)*0.5;
        elseif (ppval(pp,c1)-y)*(ppval(pp,b1)-y)<0
            a1=c1;
            b1=b1;
            c1=(a1+b1)*0.5;
        end
    end
end
    if size(x0,1)==0
        x2=x2;
    else
        x2=ppval(pp1,x2);
        x2=[sort(x2),zeros(1,8-size(x2,2))];
        a1=[x2(1),x2(2)];a2=[x2(3),x2(4)];
        a3=[x2(5),x2(6)];a4=[x2(7),x2(8)];
        x1=[x1;x2];
        plot(a1,[y,y],'r',a2,[y,y],'r',a3,[y,y],'r',a4,[y,y],'r');
        hold on
    end
end
b=sum(x1(:,2)-x1(:,1))+sum(x1(:,4)-x1(:,3))+sum(x1(:,6)-x1(:,5))+sum(x1(:,8)-x1(:,7
    ))
```

**b. The program of contour parallel hatch method**

yasai1.m

```
y1=x(1194,2);

y2=x(1195,2);

y=linspace(y1,y2,100);

y=[x(1195,1)*ones(100,1),y'];

x=[x([1:1193],:);y];

plot(x(:,1),x(:,2));

hold on

for i0=1:4

x1=x([2:size(x,1)],:)-x([1:size(x,1)-1],:);

x2=[x1([2:size(x1,1)],:);x1(1,:)];

cos2theta=-(x1(:,1).*x2(:,1)+x1(:,2).*x2(:,2))./(sqrt(x1(:,1).^2+x1(:,2).^2).*sqrt(x2(:,1
    ).^2+x2(:,2).^2));

q=[];

for k =1:size(x2,1)

      q(k)=sign(det([x1(k,:);x2(k,:)]));

end

u=[];

for i=1:size(x2,1)

if cos2theta(i)==1

      u1(i)=0;

else
```

```
u1(i)=1/sqrt((1-cos2theta(i))./(1+cos2theta(i)))*q(i);
end
end
x3=[];
for i=1:size(x2,1)
    x3(i,:)=x(i+1,:)-x2(i,:)/sqrt(x2(i,1)^2+x2(i,2)^2)*[u1(i),1;-1,u1(i)];
end
p=1;
while size(p)~=0
    p=[];
    n=size(x3,1);
for j=1:n-1
        d1=x3(j,:);
        d2=x3(j+1,:);
    for i=1:n-1
        d3=x3(i,:);
        d4=x3(i+1,:);
        e1=d1-d2;
        e2=d3-d2;
        e3=d4-d2;
        f1=d4-d3;
        f2=d1-d3;
```

```
            f3=d2-d3;
            if det([e1;e2])*det([e1;e3])<0 && det([f1;f2])*det([f1;f3])<0
                p=[p;i,j];
                break
            end
        end
end
p1=[];
for j0=1:size(p,1)
p1=[p1;[p(j0,1):p(j0,2)]'];
end
x3(p1,:)=[];
end
x3=[x3;x3(1,:)];
plot(x3(:,1),x3(:,2))
hold on
x=x3;
end
```

**2) Question 2**

**a. The program of zigzag parallel hatch method**

yasai4.m

```
plot(x1(:,1),x1(:,2),x2(:,1),x2(:,2),x3(:,1),x3(:,2),x4(:,1),x4(:,2));
hold on
t1=linspace(0,10,size(x1,1));
t2=linspace(0,10,size(x2,1));
t3=linspace(0,10,size(x3,1));
t4=linspace(0,10,size(x4,1));
pp1=csape(t1,x1(:,1));
pp2=csape(t1,x1(:,2));
pp3=csape(t2,x2(:,1));
pp4=csape(t2,x2(:,2));
pp5=csape(t3,x3(:,1));
pp6=csape(t3,x3(:,2));
pp7=csape(t4,x4(:,1));
pp8=csape(t4,x4(:,2));
t0=linspace(0,10,2000);
z0=[];
deta1=0.001;
deta2=0.001;
for y= -30:0.1:30
    z2=[];
    z1=[];
for i=1:1999
```

```
    if (ppval(pp2,t0(i))-y)*(ppval(pp2,t0(i+1))-y)<0
        z1=[z1;t0(i),t0(i+1)];
    end
end
for j=1:size(z1,1)
    if size(z1,1)==0
       break
    end
    a1=z1(j,1);
    b1=z1(j,2);
    c1=(a1+b1)*0.5;
    for k=1:100000
        if abs(ppval(pp2,c1)-y)<deta1 && b1-a1<deta2
            z2(j)=c1;
            break
        end
        if k==1000000
            z2(j)=inf;
            break
        end
        if (ppval(pp2,a1)-y)*(ppval(pp2,c1)-y)<0
            a1=a1;
```

```
            b1=c1;
            c1=(a1+b1)*0.5;
        elseif (ppval(pp2,c1)-y)*(ppval(pp2,b1)-y)<0
            a1=c1;
            b1=b1;
            c1=(a1+b1)*0.5;
        end
    end
end
    z3=[];
    z4=[];
for i=1:1999
    if (ppval(pp4,t0(i))-y)*(ppval(pp4,t0(i+1))-y)<0
        z3=[z3;t0(i),t0(i+1)];
    end
end
for j=1:size(z3,1)
    if size(z3,1)==0
        break
    end
    a1=z3(j,1);
    b1=z3(j,2);
```

```
    c1=(a1+b1)*0.5;
    for k=1:100000
        if abs(ppval(pp4,c1)-y)<deta1 && b1-a1<deta2
            z4(j)=c1;
            break
        end
        if k==1000000
            z4(j)=inf;
            break
        end
        if (ppval(pp4,a1)-y)*(ppval(pp4,c1)-y)<0
            a1=a1;
            b1=c1;
            c1=(a1+b1)*0.5;
        elseif (ppval(pp4,c1)-y)*(ppval(pp4,b1)-y)<0
            a1=c1;
            b1=b1;
            c1=(a1+b1)*0.5;
        end
    end
end
  z5=[];
```

```
    z6=[];
for i=1:1999
    if (ppval(pp6,t0(i))-y)*(ppval(pp6,t0(i+1))-y)<0
        z5=[z5;t0(i),t0(i+1)];
    end
end
for j=1:size(z5,1)
    if size(z5,1)==0
        break
    end
    a1=z5(j,1);
    b1=z5(j,2);
    c1=(a1+b1)*0.5;
    for k=1:100000
        if abs(ppval(pp6,c1)-y)<deta1 && b1-a1<deta2
            z6(j)=c1;
            break
        end
        if k==1000000
            z6(j)=inf;
            break
        end
```

```
            if (ppval(pp6,a1)-y)*(ppval(pp6,c1)-y)<0
                a1=a1;
                b1=c1;
                c1=(a1+b1)*0.5;
            elseif (ppval(pp6,c1)-y)*(ppval(pp6,b1)-y)<0
                a1=c1;
                b1=b1;
                c1=(a1+b1)*0.5;
            end
        end
end
    z7=[];
    z8=[];
for i=1:1999
        if (ppval(pp8,t0(i))-y)*(ppval(pp8,t0(i+1))-y)<0
            z7=[z7;t0(i),t0(i+1)];
        end
end
for j=1:size(z7,1)
        if size(z7,1)==0
            break
        end
```

```
a1=z7(j,1);
b1=z7(j,2);
c1=(a1+b1)*0.5;
for k=1:100000
    if abs(ppval(pp8,c1)-y)<deta1 && b1-a1<deta2
        z8(j)=c1;
        break
    end
    if k==1000000
        z8(j)=inf;
        break
    end
    if (ppval(pp8,a1)-y)*(ppval(pp8,c1)-y)<0
        a1=a1;
        b1=c1;
        c1=(a1+b1)*0.5;
    elseif (ppval(pp8,c1)-y)*(ppval(pp8,b1)-y)<0
        a1=c1;
        b1=b1;
        c1=(a1+b1)*0.5;
    end
end
```

```
end
   xx1=ppval(pp1,z2);          xx2=ppval(pp3,z4);          xx3=ppval(pp5,z6);
    xx4=ppval(pp7,z8);
   xx=sort([xx1,xx2,xx3,xx4]);
   if size(xx,1)==0
        z0=z0;
   else
   xx=[xx,zeros(1,10-size(xx,2))];
   z0=[z0;xx];

    plot([xx(1),xx(2)],[y,y],'r',[xx(3),xx(4)],[y,y],'r',[xx(5),xx(6)],[y,y],'r',[xx(7),xx(8)],
    [y,y],'r',[xx(9),xx(10)],[y,y],'r');
         hold on
    end
end
b=sum(z0(:,2)-z0(:,1))+sum(z0(:,4)-z0(:,3))+sum(z0(:,6)-z0(:,5))+sum(z0(:,8)-z0(:,7))
    +sum(z0(:,10)-z0(:,9))
```

**b. The program of contour parallel hatch method**

yasai5.m

```
plot(x1(:,1),x1(:,2),x2(:,1),x2(:,2),x3(:,1),x3(:,2),x4(:,1),x4(:,2));
hold on
for i0=1:4
```

```
y1=x2([2:size(x2,1)],:)-x2([1:size(x2,1)-1],:);

y2=[y1([2:size(y1,1)],:);y1(1,:)];

cos2theta=-(y1(:,1).*y2(:,1)+y1(:,2).*y2(:,2))./(sqrt(y1(:,1).^2+y1(:,2).^2).*sqrt(y2(:,1
    ).^2+y2(:,2).^2));

q=[];

for k =1:size(y2,1)

    q(k)=sign(det([y1(k,:);y2(k,:)]));

end

u1=[];

for i=1:size(y2,1)

if cos2theta(i)==1

    u1(i)=0;

else

u1(i)=1/sqrt((1-cos2theta(i))./(1+cos2theta(i)))*q(i);

end

end

y3=[];

for i=1:size(y2,1)

    y3(i,:)=x2(i+1,:)-0.2*y2(i,:)/sqrt(y2(i,1)^2+y2(i,2)^2)*[u1(i),1;-1,u1(i)];

end

p=1;

while size(p)~=0

    p=[];
```

```
    n=size(y3,1);
for j=1:n-1
        d1=y3(j,:);
        d2=y3(j+1,:);
    for i=1:n-1
        d3=y3(i,:);
        d4=y3(i+1,:);
        e1=d1-d2;
        e2=d3-d2;
        e3=d4-d2;
        f1=d4-d3;
        f2=d1-d3;
        f3=d2-d3;
        if det([e1;e2])*det([e1;e3])<0 && det([f1;f2])*det([f1;f3])<0
            p=[p;i,j];
            break
        end
    end
end
p1=[];
for j0=1:size(p,1)
p1=[p1;[p(j0,1):p(j0,2)]'];
```

```
end
y3(p1,:)=[];
end
y3=[y3;y3(1,:)];
plot(y3(:,1),y3(:,2))
hold on
x2=y3;
yy1=x4([2:size(x4,1)],:)-x4([1:size(x4,1)-1],:);
yy2=[yy1([2:size(yy1,1)],:);yy1(1,:)];
cos2theta=-(yy1(:,1).*yy2(:,1)+yy1(:,2).*yy2(:,2))./(sqrt(yy1(:,1).^2+yy1(:,2).^2).*sqr
    t(yy2(:,1).^2+yy2(:,2).^2));
q=[];
for k =1:size(yy2,1)
      q(k)=sign(det([yy1(k,:);yy2(k,:)]));
end
uu1=[];
for i=1:size(yy2,1)
if cos2theta(i)==1
      uu1(i)=0;
else
uu1(i)=1/sqrt((1-cos2theta(i))./(1+cos2theta(i)))*q(i);
end
end
```

```
yy3=[];
for i=1:size(yy2,1)
    yy3(i,:)=x4(i+1,:)-0.2*yy2(i,:)/sqrt(yy2(i,1)^2+yy2(i,2)^2)*[uu1(i),1;-1,uu1(i)];
end
p=1;
while size(p)~=0
    p=[];
    n=size(yy3,1);
for j=1:n-1
        d1=yy3(j,:);
        d2=yy3(j+1,:);
    for i=1:n-1
        d3=yy3(i,:);
        d4=yy3(i+1,:);
        e1=d1-d2;
        e2=d3-d2;
        e3=d4-d2;
        f1=d4-d3;
        f2=d1-d3;
        f3=d2-d3;
        if det([e1;e2])*det([e1;e3])<0 && det([f1;f2])*det([f1;f3])<0
            p=[p;i,j];
```

```
                break
            end
        end
end
p1=[];
for j0=1:size(p,1)
p1=[p1;[p(j0,1):p(j0,2)]'];
end
yy3(p1,:)=[];
end
yy3=[yy3;yy3(1,:)];
plot(yy3(:,1),yy3(:,2))
hold on
x4=yy3;
yyy1=x1([2:size(x1,1)],:)-x1([1:size(x1,1)-1],:);
yyy2=[yyy1([2:size(yyy1,1)],:);yyy1(1,:)];
cos2theta=-(yyy1(:,1).*yyy2(:,1)+yyy1(:,2).*yyy2(:,2))./(sqrt(yyy1(:,1).^2+yyy1(:,2).
    ^2).*sqrt(yyy2(:,1).^2+yyy2(:,2).^2));
q=[];
for k =1:size(yyy2,1)
    q(k)=sign(det([yyy1(k,:);yyy2(k,:)]));
end
uuu1=[];
```

```
for i=1:size(yyy2,1)
if cos2theta(i)==1
    uuu1(i)=0;
else
uuu1(i)=1/sqrt((1-cos2theta(i))./(1+cos2theta(i)))*q(i);
end
end
yyy3=[];
for i=1:size(yyy2,1)

    yyy3(i,:)=x1(i+1,:)-0.2*yyy2(i,:)/sqrt(yyy2(i,1)^2+yyy2(i,2)^2)*[uuu1(i),1;-1,uuu
    1(i)];
end
p=1;
while size(p)~=0
    p=[];
    n=size(yyy3,1);
for j=1:n-1
        d1=yyy3(j,:);
        d2=yyy3(j+1,:);
    for i=1:n-1
        d3=yyy3(i,:);
        d4=yyy3(i+1,:);
```

```
            e1=d1-d2;
            e2=d3-d2;
            e3=d4-d2;
            f1=d4-d3;
            f2=d1-d3;
            f3=d2-d3;
            if det([e1;e2])*det([e1;e3])<0 && det([f1;f2])*det([f1;f3])<0
                p=[p;i,j];
                break
            end
        end
end
p1=[];
for j0=1:size(p,1)
p1=[p1;[p(j0,1):p(j0,2)]'];
end
yyy3(p1,:)=[];
end
yyy3=[yyy3;yyy3(1,:)];
plot(yyy3(:,1),yyy3(:,2))
hold on
x1=yyy3;
```

```
yyyy1=x3([2:size(x3,1)],:)-x3([1:size(x3,1)-1],:);

yyyy2=[yyyy1([2:size(yyyy1,1)],:);yyyy1(1,:)];

cos2theta=-(yyyy1(:,1).*yyyy2(:,1)+yyyy1(:,2).*yyyy2(:,2))./(sqrt(yyyy1(:,1).^2+yyy
    y1(:,2).^2).*sqrt(yyyy2(:,1).^2+yyyy2(:,2).^2));

q=[];

for k =1:size(yyyy2,1)

      q(k)=sign(det([yyyy1(k,:);yyyy2(k,:)]));

end

uuuu1=[];

for i=1:size(yyyy2,1)

if cos2theta(i)==1

      uuuu1(i)=0;

else

uuuu1(i)=1/sqrt((1-cos2theta(i))./(1+cos2theta(i)))*q(i);

end

end

yyyy3=[];

for i=1:size(yyyy2,1)


    yyyy3(i,:)=x3(i+1,:)-0.2*yyyy2(i,:)/sqrt(yyyy2(i,1)^2+yyyy2(i,2)^2)*[uuuu1(i),1;-
    1,uuuu1(i)];

end

p=1;
```

```
while size(p)~=0
    p=[];
    n=size(yyyy3,1);
for j=1:n-1
        d1=yyyy3(j,:);
        d2=yyyy3(j+1,:);
    for i=1:n-1
        d3=yyyy3(i,:);
        d4=yyyy3(i+1,:);
        e1=d1-d2;
        e2=d3-d2;
        e3=d4-d2;
        f1=d4-d3;
        f2=d1-d3;
        f3=d2-d3;
        if det([e1;e2])*det([e1;e3])<0 && det([f1;f2])*det([f1;f3])<0
            p=[p;i,j];
            break
        end
    end
end
p1=[];
```

```
for j0=1:size(p,1)

p1=[p1;[p(j0,1):p(j0,2)]'];

end

yyyy3(p1,:)=[];

end

yyyy3=[yyyy3;yyyy3(1,:)];

plot(yyyy3(:,1),yyyy3(:,2))

hold on

x3=yyyy3;

end
```