This is the cover page and is not included in the page count



**Design and Calculating Bounded Area**

**Stage 2 Mathematics Methods**

**SACE No. 216771E**

**Introduction**

1. **Part A: Working with a Given Function and Designing with Functions**
   1. **Overview**

Part A of the task requires a graphic design that is clearly described mathematically. It also asks to integrate two given functions and into the design. A good mathematical expression given in this section will be important in Part B when evaluating the area. To achieve this, a series of methods will be introduced, and based on them, a rather complex but not overly elaborate pattern is expected.

* 1. **Calculate the area of the given region**

Given a function . The second function It is created by vertically translating down by 2 units. The region bounded by the two curves is

(see Figure 1)

To calculate its area, there are 2 methods.

|  |  |  |
| --- | --- | --- |
| This area can be evaluated by the formula:    This is a general formula to calculate the region bounded by a pair of functions in a given domain. To use this formula, the equation of is required. By the definition of translation, . |  | original  Figure 1. Region *D0* |
| An easier way: the region between 2 curves is formed by translation. Hence, when evaluating the integral in Eq. 1, every infinitesimal rectangle of the Riemann summation has the same length , so the total area is equal to the area of a 4.5 by 2 rectangle (see the orange rectangles in Figure 2). |  | Figure 2. Put all infinitesimal rectangles together |

* 1. **A rough sketch of the final design**
  2. **Design of region *D1***
     1. **Graph transformation of a function**

The task asks to integrate the given functions and into the final design, and it is encouraged to use transformations including reflection, translation, and scaling. Generally, for a function , the transformation above can be done by following operations:

1. Translate right by : ; Translate left by : ; Translate upward by : ; Translate downward by : .
2. Horizontal scaling: . Vertical scaling: .
3. Reflection across the x-axis: . Reflection across the y-axis:

Moreover, these transformations can be combined. For example, means shift right by 3, stretch vertically by a factor of 2, reflect in the x-axis, and shift up by 4.

* + 1. **Non-single-valuedness and affine transformation of a parametric curve**

The definition of the function is that

It is also known as the **single-valuedness** of functions. However, the design is expected to be **non-single-valued**. Hence, affine transformations and parametric curves are introduced. A simple definition of an **affine transformation** is to perform a linear transformation on a vector space, and additionally, a translation. Generally, an affine transformation on vector in can be written as

where is a 2 by 2 matrix representing a linear transformation, and represents a translation. The transformation of a function is more complicated. For this specific purpose, since what is transformed is the whole shape of a function , which means the actual subject of the transformation is every point on the plane, namely

This equation naturally satisfies the restriction on the **domain.** But note that after the transformation, the relation between and will probably no longer obey the single-valuedness. To better describe this kind of curve, **parametric equations** are introduced. According to Eq. 2, an affine transformation on a parametric equation is

Parameterise is simply letting , so . And , the parametric form of , is given by transformation . Now all kinds of transformation can be applied to. To adjust the given function to be a fancy frame of the design, the following transformations are applied:

|  |  |  |  |
| --- | --- | --- | --- |
| The region looks too long in the vertical direction. To compress it vertically to 1/5 of the original size and stretch horizontally to four times the original size requires the matrix  Since there is no translation, b is the zero vector. | Figure 3. Transformation 1 |  |  |
| Then the wiggling end of the curves is expected to be centered and aligned with the y-axis. Hence a flip transformation and a translation are needed. The flip matrix is .  The translating vector is | Figure 4. Transformation 2 |  |  |
| Copy a pair of curve and to make a symmetrical shape, where the same flip matrix is needed. | Figure 5. Transformation 3 |  |  |

An advantage of affine transformations is that the amount of the scaling of the area can be easily calculated by the **determinant** of the transformation matrix. This will be further discussed in Part B.

* + 1. **Non-linear transformation and Jacobian**

The shape of a frame is expected to be more “enclosed”, by curlingit into a circular shape. This requires a **non-linear transformation**. A non-linear transformation does not guarantee that the parallel lines in the original graph remain parallel, and hence do not have a constant matrix to describe the transformation for every point on the Cartesian plane. A general way is to use a **map**. For this purpose, the map is:

since the new graph’s x-coordinates have a range from -18 to 18, and the curl starts at in the polar frame. What this **curl** transformation actually does is to wrap the Cartesian grid into a polar shape (see Figure 6). For convenience, the transformed curves are named .

|  |  |
| --- | --- |
| ↓ curl    Figure 6. Curl transformation |  |

Though this transform do not have a matrix to describe it, it do behave locally as a linear transformation. This sort of linear behaviour can be measured by the Jacobian. For transformation , the Jacobian is

The method to evaluate the transformed area based on the Jacobian will be further discussed in Part B.

* + 1. **Smoothness Consideration**

Generally, the smoothness can be measured by continuity , namely the nth derivatives at the two ends are equal. However, when dealing with a parametric curve, this approach will probably fail since the rate of change in position of a point with respect to t is not only related to the smoothness of the shape but also the smoothness of the motion (change in velocity). Thereby, the geometric continuity is introduced. A curve or surface can be described as having continuity, with being the increasing measure of smoothness. Consider there are two curves, and their continuity is defined as: (1) , the curves touch at the join point; (2) , the curves also share a common tangent direction at the join point, which means the derivatives are collinear; (3) , the curves also share a common centre of curvature at the join point (Wikipedia, 2023). For graph designs, is sufficiently smooth. Also, an affine transformation will not affect the original continuity, and actually, every continuous mapping can keep the original continuity.

To construct continuous joins, copy one set of curves shown in Figure 6 and rotate them 180 degrees to get curves **,** therefore,

(for full equations, see Appendix 1).

|  |  |
| --- | --- |
| Figure 7. Region *D1* | Curves give us Region *D1* as shown in Fig. 2, where there are four joins satisfying and continuity: , , , and . Take as an example. |

Since the direction of the motion is reversible in a parametric equation, Eq. 7 actually implies there is a continuity. By applying similar approaches to the other three joins, it can be found that all joins are continuous. Now, these curves can be denoted as

And let Region be the region bounded by the piecewise smooth closed curve but not by , namely

* 1. **Design of region *D2***

|  |  |  |
| --- | --- | --- |
| Figure 8. g1(x) | This part of the design is the wing of a bird, which would be formed by some basic functions. Its shape is roughly like Figure 9, constituted by functions .  is simply a straight line. For convenience, it is put horizontally and assigned to be | C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\Content.Word\wing.png  Figure 9. D2 Rough shape |

|  |  |
| --- | --- |
| Figure 10. g2(x) | is winding and has an “S” shape. A very suitable function is Sigmoid, which is defined as  To fit this curve to the shape required, use the graph transformation in 1.4.1:  Where , , and control the dilations in and directions and horizontal translation. By adjusting them, it is found that the curve in Figure 10 is given when , , and are 2.6, 3.4, and 0.9. |
| Figure 11. g3(x) | is the upper half of the left side of the wing. It is like a log function and can be modelled as  Where , , , and control the dilations in and directions and translation in and directions. The curve in Figure 10 is given when , , , and are 1, 0.1, -0.8, and 3.8, respectively. It is expected that the and connect at with continuity. The derivative of at is |
| Figure 12. g4(x)    Figure 13. D2 no domain | is modelled by a quadratic function:  Whose derivative is . To make and connect at with continuity, the following 2 equations need to be satisfied.  Note that the equations are now hard to solve. To simplify it, let . Thus, .  3 variables and 2 equations leave 1 degree of freedom to adjust. When set , Figure 12 is given, and . |
| Figure 14. D2 | To get the joints A, B, D, to get the **domains**, it is necessary to solve the following equations:  The solution is . Hence, the domains are , , , and . |

Let the upper and the lower functions be respectively

And Region (see Figure 14).

* 1. **Design of region *D3***
     1. **Introduction to Bézier Curves**

Bézier Curves are a type of interpolation method intended to approximate a real-world shape that otherwise has no mathematical representation. The general equation of a Bezier curve of order n is

Where are the control points**.** The curves are expected to be rather wavy when designing the pattern using Bezier curves. It can be achieved by using a high-order Bezier curve, but there are problems. First, when there are many control points, it is hard to imagine the shape of the curve. Secondly, one cannot modify only a certain part of the curve, because if one control point is moved, the whole curve will change. In addition, applying a higher-order curve can cause a problem of oscillation at the edges of an interval. It is called **Runge's phenomenon**. A viable solution is a composite Bezier curve, namely, composing a curve of multiple shorter Bezier curves. To do that, the smoothness of their joints must be guaranteed. The derivative of Eq. 8 is

(for the full derivation, see Appendix 2) Put and , the conclusion can be obtained that: the tangent vector at the start point is and at the terminal point is . Thus, to ensure that the two curves have continuity at the junction is to put the four control points near the junction collinearly, namely

* + 1. **Design based on Bézier Curves**

|  |  |  |  |
| --- | --- | --- | --- |
| Figur*e* 15. D3 | Figure 16. Continuous joint between B1 and B2 | Figure 17*.* Continuous joint between B*6*and B*7* | Region D3 is enclosed by 8 Bezier curves with control points *.* The curves are connected end to end and form an enclosed shape (see Figure 15). Note that Bezier curves naturally have a domain of . As shown in Figure 16 & Figure 17, there are 2 pairs of curves that have continuous joints, which are taken as samples to show calculations (for full equations see Appendix 3). |

|  |  |
| --- | --- |
| Figure 18. B*2*    Figure 19. B*1* | (see Figure 18) has control points  Substitute into Eq. 8, there is  **To ensure G1 continuity, it must be**  That is to say,  Under this constraint, the control points of B1 (see Figure 19) are designed to be  where the k is set to be 3.90. Hence the equation is |
| Figure 20. B*4*    Figure 21. B*5* | (see Figure 20) has control points  Note there is only one turning point is required for this curve. So it only needs 3 control points, namely it is a quadratic Bezier curve. Substitute into Eq. 8, there is  **To ensure G1 continuity, it must be**  **That is to say,**  Under this constraint, B5 (see Figure 21) are designed to be |

All these 8 curves and the Regioncan be denoted as

* 1. **Design of region *D4***
     1. **Curve fitting and interpolation**

|  |  |  |
| --- | --- | --- |
| Note that what is done in the design of *D2* and *D3* is pretty much fitting a shape with some curve. In numerical analysis, it is called interpolation, which means the process of predicting the continuous value between discrete data points with some function or curve. Bezier curves are a classical method of interpolation, which is essentially a parameterised polynomial. Hence, why not just use a polynomial to fit some given pattern and get a design? | Figure 22. Handwritten letter M | Figure 23. Runge phenomenon of 10-point Lagrange interpolation |
| Given a handwritten letter M shown in Figure 22, a polynomial is used for interpolation for 20 sample points of upper and lower boundaries. Take the upper boundary as an example. Take 10 sample points , The Lagrange interpolating polynomial through them is | |

where is called the Lagrange basis for this linear combination. Each if and , which guarantees that the polynomial must go through all points. This interpolation yields the curve in Figure 24. It is not well-fitted and has severe oscillations at the edges of the domain, which is the Runge phenomenon investigated in the context of Bezier curves. The solution is the same – consider composing multiple polynomials together.

* + 1. **Piecewise interpolation polynomial and Hermite basis functions**

To construct a polynomial between each interval and ensure that adjacent ones connect with continuity, it requires:

where is a parameter we assign to the slope shared by the ends of the two adjacent polynomials. Notice that there are 4 constraints, so consider using cubic polynomials with 4 free parameters.For the convenience of the derivation, use a factor to scale the interval to by letting . Hence, the intended satisfies

If assuming , there are

For convenience, write as a linear combination of , such that

with the result in Eq. 12, it can be found that the bases (called Hermite basis functions) are

* + 1. **Design with the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)**

There are 2 parameters to adjust: the number of sample points and joint slope . For the sample, OpenCV-Python is used to extract sample pointa from the boundaries. Different numbers of sample points can influence how many details of the original graph can be caught by the PCHIP. After trials, it is found that 23 is a suitable value (see Table 1)

Table 1. Comparison between different numbers of sample points

|  |  |  |
| --- | --- | --- |
| 15 sample points | 23 sample points | 30 sample points |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Here is set to be , which is the average of the secant slopes of the adjacent points with the edge conditions . But in figures in Table 1, some unnatural wiggles are observed at sharp turns. It is because when a sample point is a stationary point, the corresponding might not be 0, making the PCHIP faking a new stationary point (see Figure 25). The solution is, when dealing with a local maximum or minimum, namely  If and have different signs, . | Figure 25. Fake peak | Figure 26. D4 |

Which yields the curves in Figure 26, which contains two sets of 22 PCHIPs , namely,

And Region . For expanded equations and Python code see Apendix 4 and Code Appendix.

* 1. **Design of region *D5***
     1. **Generate an interpolation curve from the given points based on the Discrete Fourier Transform**

The PCHIP can only be applied to points that have a function relation between ys and xs, namely, for every y, there is only one corresponding x. That is to say, it cannot be applied to parametric curves. That is why the **Fourier Transform** is introduced. Consider an arbitrary closed (end-to-end) parametric curve . Denote it as a complex function

Note that this complex function can be expressed as a sum of infinite simple orbiting complex numbers with different frequencies and complex coefficients to describe the phases and amplitude. Hence, there is

This series, known as the **Fourier series**, was proposed by Joseph Fourier in 1807. The reason why the frequency is is that for any closed curve, is true if and only if ; you can see this as a sort of period ( ). The **Fourier Transform** is used to transform a function from the original domain to the frequency domain, namely to get corresponding s for different frequencies s, which is:

Proof: According to Eqs. 17 and 15,

According to **Euler’s formula** , the complex number really describes a “revolution”. Therefore, if integrate it through t, it will give zero as:

Hence, all terms in Eq. 15 vanish except . In the real-world problem, it is unlikely to get a continuous curve, and most of the time, we have to deal with discrete sample points. Suppose there are discrete points on a curve in complex form: . The **Discrete Fourier Transform** and the Fourier series fetched are:

A higher can give a more accurate fitting of the original curve, and a relatively small can filter the high-frequency details and give a gentler (no steep wiggle) shape.

* + 1. **Design with the Fourier series**

|  |  |  |
| --- | --- | --- |
| Given an original graph like Figure 27, first take sample points with OpenCV-Python on the contour of it. Note that the contour must be enclosed for the periodic property of the Fourier series. Since this is a rather complex shape, to keep more details, 2500 sample points are taken. Next, the only parameter that needs to be adjusted is the number of terms retained from the series (the value of *M*). Different *M*s give different curves (see Figure 28). | Figure 27. D5 original graph | Figure 28. Sampled graph |

*Table 2.* Different *values of* M *and corresponding* curves

|  |  |  |
| --- | --- | --- |
| M=2 | M=10 | M=100 |
|  |  |  |
| M=200 | M=1000 | M=2500 |
|  |  |  |

The process to get the Discrete Fourier Transform from 2500 sample points requires massive calculation. Therefore, a Python program is used (see Code Appendix). Notice that when M=2500, the curve start to have sawteeth from the pixels of the original image, which is not expected. Hence, M=1000 is considered more suitable. Now, an enclosed curve is obtained. It only consists of one curve. Thus, it has continuity everywhere. Translate it from complex form to vector form:

Let , the Region (see the figure where M=1000 in Table 2).

* 1. **Assemble and arrange parts**

Now all parts are settled and they are going to be assembled. However, at this stage, are they not at the same scale, specifically, some are too big while some are too small (see Figure 29). Moreover, some final adjustments of the patterns are required. Hence, the arrangement is based on affine transformations.

|  |  |  |  |
| --- | --- | --- | --- |
| C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\Content.Word\assemble.png  Figure 29. Parts before transformation | | C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\Content.Word\assemble_D3.png  Figure 30. Assemble *D3* | First, assemble *D3* since it determines where the wing and the vine lie. *D3* is too big, so before the translation , compress it first by linear transformation . Thus, the affine transformation is |
| Figure 31. Assemble *D2* | To assemble the wing to the bird, a rotation and a scaling are required. The composed transformation is    And the translation is . The affine transformations on the parameterised functions are | | |
| C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\Content.Word\assemble_D5.png  Figure 32. Assemble *D5* | The vine needs a rotation of 90° and a scaling. The composed matrix is\  And the translation is . The affine transformation is | | |
| C:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\Content.Word\assemble_label.png  Figure 33. Final design | For the handwritten letter M in *,* copy an identical region . Different scalings and shearings are applied:  And the translations are and . The affine transformations are.  Here, all designs are done. Their areas are ready to be evaluated. | | |

**Evaluate:** The design in Figure 33 using various types of functions, including trig, log, exponential and polynomial, is a graph of high quality.

* 1. **Summary: A comparison of the mathematics methods of design used**
     1. **A comparison of the transformation methods (Green: strength. Red: limitation)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Graph transformation** | **Affine transformation** | **Non-linear transformation** |
| **Object** | Function | Vector space (hence any vector, parametric equation, and parameterised function) | Every point in the plane. |
| **Non-single-valuedness** | Not supported | Supported before and after the transformation | Supported before and after the transformation |
| **Computational complexity** | Low | Moderate. Requires matrix operations. | High, involves nonlinear equations and singularity. |
| **Type of transformation supported** | Translation, scaling, reflection. Very limited transformational capability | Moderate, supports translation, scaling, reflection, rotation, and shearing. | Most, including complex distortions, bending, and even topological changes |
| **Preserved properties** | Straightness, parallelism, ratio, continuity, monotonicity and function property (single-valuedness) | Straightness, parallelism, continuity, and ratio. | Generally does not preserve geometric properties. |
| **Intuitiveness** | High, easy to understand and visualise | Moderate. Requires intuitive understandings of matrices and vector spaces. | Low |
| **Computational complexity of the area scaling factor** | Low | Low, just need to evaluate the determinant. | High, involves partial derivatives (Jacobians) and double integrals. |

* + 1. **A comparison of the methods of curve design by adjusting**

|  |  |  |
| --- | --- | --- |
|  | **Design with functions** | **Design with Bezier curves** |
| **Outcome** | Arbitrary functions | Quadratic or cubic parametric curves |
| **Non-single-valuedness** | Not supported for a single function. Can be achieved by multiple single-valued functions. | Supported |
| **Computational complexity** | Relatively low | Moderate, needs to expand equations. |
| **Intuitiveness** | Low, needs to construct and imagine the shape in mind | High |
| **Ensure continuity** | Hard, needs to take derivatives of both at joint. | Easy |

* + 1. **A comparison of the methods of curve design by fitting**

|  |  |  |
| --- | --- | --- |
|  | **PCHIP** | **Fourier transform** |
| **Outcome** | Piecewise cubic functions | Cartesian form of a complex Fourier series |
| **Non-single-valuedness** | Not supported for a single function. Can be achieved by multiple single-valued functions. | Supported |
| **Computational complexity** | Moderate | Very high when doing summations of 1000+ terms |
| **Degree of process automation** | Relatively low, parameters such as number of sample points and the secant slope need to be assigned. | Very high, only needs to set the number of terms of the series kept. |
| **Ensure continuity** | Ensures continuity | Naturally has continuity |

* + 1. **A comparison of the design by adjusting & the design by fitting**

|  |  |  |
| --- | --- | --- |
|  | **Design by adjusting** | **Design by fitting** |
| **Strength** | Highly customised. | Highly automatic process. Can be customised by inputting hand-drawn images. |
| **Limitation** | Slow and complicated process. | Usually requires lots of calculations that have to be done using computers |

1. **Part B: Calculations of the Area** 
   1. **Overview**

The task asks to apply calculus and mathematical reasoning to analyse full designs. Specifically, calculate the area of the design using algebraic integration, justify assumptions and limitations in modelling processes, and discuss how the model simplifies real-world structures.

* 1. **Calculate region *D1***

|  |
| --- |
|  |

**Bibliography**

Wikipedia. (2023). *Smoothness*. [online] Available at: https://en.wikipedia.org/wiki/Smoothness.

3Blue1Brown (2019). *But what is a Fourier series? From heat flow to drawing with circles*. [online] YouTube. Available at: <https://www.youtube.com/watch?v=r6sGWTCMz2k&list=PL4VT47y1w7A1-T_VIcufa7mCM3XrSA5DD&index=4>.

**Appendix 1. Curves that construct *D1***

**Appendix 2. Derivative of the Bezier curves.**

We have discussed the general form of Bezier curves. But to make the index of control point sstart from 1, we didn’t use the most general equation, which is supposed to be

First, take the derivative of the th term of a Bezier curve without control points as coefficients:

Expand the binomial coefficient:

Notice that the two terms are typical Bezier curves of lower orders. Hence,

Now, apply the control points.

Expand this and reorder the terms by :

In this transformation, there are two eliminations, namely as the th term does not exist and the very last term with the binomial coefficient , which is usually considered to be zero. Write it in the form of a sum:

This form can also be translated to our convetion by letting .

**Appendix 3. Equations.**

The control points are,

According to Eq. 8,

**Appendix 4. PCHIP Equations**

For the upper boundary, the sample points are:

(23.0000,125.0000),(30.0000,126.0000),(38.0000,121.0000),(46.0000,116.0000),(54.0000,111.0000),(62.0000,105.0000),(69.0000,98.0000),(77.0000,92.0000),(85.0000,82.0000),(93.0000,70.0000),(101.0000,58.0000),(109.0000,47.0000),(116.0000,122.0000),(124.0000,116.0000),(132.0000,103.0000),(140.0000,93.0000),(148.0000,83.0000),(155.0000,69.0000),(163.0000,55.0000),(171.0000,84.0000),(179.0000,107.0000),(187.0000,122.0000),(195.0000,121.0000)

Hence, according to Eq. 12 and the Hermite basis functions,

Similarly, for the lower boundary, the sample ponits are (23.0000,120.0000), (30.0000,118.0000), (38.0000,112.0000), (46.0000,108.0000), (54.0000,102.0000), (62.0000,95.0000), (69.0000,90.0000), (77.0000,80.0000), (85.0000,68.0000), (93.0000,56.0000), (101.0000,45.0000), (109.0000,28.0000), (116.0000,25.0000), (124.0000,101.0000), (132.0000,91.0000), (140.0000,80.0000), (148.0000,66.0000), (155.0000,53.0000), (163.0000,39.0000), (171.0000,36.0000), (179.0000,80.0000), (187.0000,106.0000), (195.0000,121.0000).

Hence, according to Eq. 12 and the Hermite basis functions,