CE7453 Numerical Analysis

Assignment 1 Report

XU SHIFENG G2104007A [shifeng001@e.ntu.edu.sg](mailto:shifeng001@e.ntu.edu.sg)

# Cubic B-Spline interpolation Algorithm

I implement the cubic B-spline interpolation algorithm by C# on Windows platform, with Windows form GUI. And here is how it looks like:

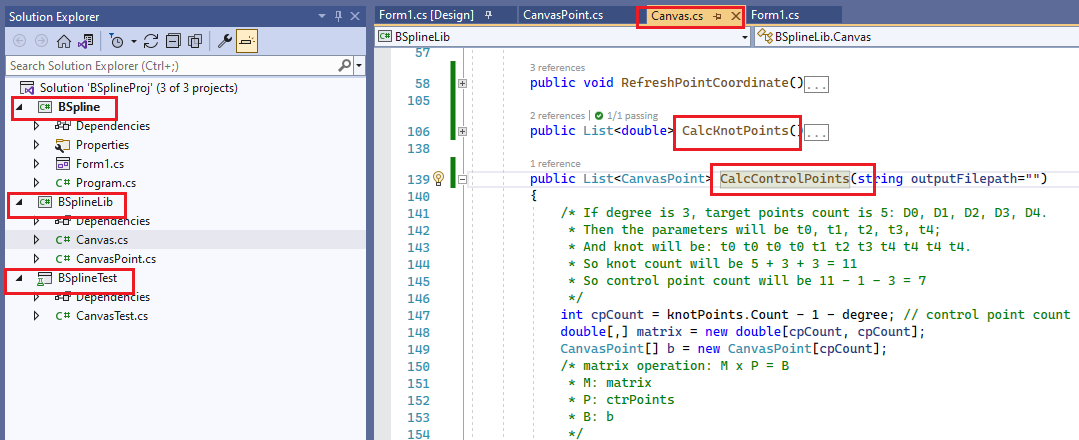
Graphical user interface

Description automatically generated

The source code is a Windows C# solution, with 3 projects as below.

|  |  |
| --- | --- |
| **Project** | **Remarks** |
| BSpline | The GUI. Windows form app. |
| BSplineLib | Handle non-UI task, such as calculating the knots, calculating the control points. |
| BSplineTest | Unit testing. |

Here is the layout in Visual Studio 2022.



The interpolation algorithm is implemented in “Canvas.cs” of the BSplineLib project.

# Explicit Linear Equations

Given the input target points below, we need to calculate some B-Spline curve to interpolate them. We are to define the range, find the knots, and calculate the control points.

|  |  |
| --- | --- |
| **Notation** | **coordinate** |
| D0 | 0 0 |
| D1 | 0 2 |
| D2 | 2 2 |
| D3 | 2 0 |
| D4 | 4 0 |

Here are the steps for the linear equations.

## Construct the matrix based on formula

Following the convention in our course, we have n = 4. This means we have (n+1) target points to interpolate. So, the range is t0, t1, t2, t3, t4. Because degree is 3, the knots count will be n+1+3+3 = 11. And the knots u0 ~ u10:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Knot** | **u0** | **u1** | **u2** | **u3** | **u4** | **u5** | **u6** | **u7** | **u8** | **u9** | **u10** |
|  | t0 | t0 | t0 | t0 | t1 | t2 | t3 | t4 | t4 | t4 | t4 |
| **real value** | 0 | 0 | 0 | 0 | 0.25 | 0.5 | 0.75 | 1 | 1 | 1 | 1 |

So, the control point count is: 11 – 3 – 1 = 7. Another quicker way to get this number is: n + 3 = 7. Let’s assume the control points are: P0, P1, P2, P3, P4, P5, P6.

For k = 0, 1, 2, 3, 4, the point on B-Spline curve matches:

Which is:

Please keep in mind here, for k = 0, 1, 2, 3, 4, . Therefore, the above equation can be rewritten as:

In my personal view, the latter equation is easier to understand, as it reflects the relationship between “u” and “”.

Since we choose multiple knots at t0 and t4, the first and last control points P0, P6 interpolate D0, D4. It is easy to verify that

Based on the 5 target points, we have 5 equations:

Then we add the endpoint conditions.

And insert the 2 equations at the second and penultimate lines. We get the final matrix equations:

Then for 7 unknown variables, we got 7 linear equations. And the matrix above is a tri-diagonal matrix, which can be resolved in O(n) time complexity.

## Hard point

In fact, during my task, the most difficult point is not the equations above, but the manipulations of the indexes. As shown below, the cubic B-Spline basis function is complicated and has so many subscripts (indexes of variable “”). I will explain how I struggled with it later.



## Result it

Given the above formula, and with the help of my program, I got the matrix as below:

|  |
| --- |
| 1 0 0 0 0 0 0 ( 0 , 0 )  96 -144 48 0 0 0 0 ( 0 , 0 )  0 0.25 0.58 0.17 0 0 0 ( 0 , 2 )  0 0 0.17 0.67 0.17 0 0 ( 2 , 2 )  0 0 0 0.17 0.58 0.25 0 ( 2 , 0 )  0 0 0 0 48 -144 96 ( 0 , 0 )  0 0 0 0 0 0 1 ( 4 , 0 ) |

From the tri-diagonal matrix above, we can remove one “sub-diagonal”:

|  |
| --- |
| 1 0 0 0 0 0 0 ( 0 , 0 )  0 1 -0.33 0 0 0 0 (-0 , -0 )  0 0 1 0.25 0 0 0 ( 0 , 3 )  0 0 0 1 0.27 0 0 ( 3.20, 2.40)  0 0 0 0 1 0.46 0 ( 2.72, -0.74)  0 0 0 0 0 1 -0.58 ( 0.79, -0.21)  0 0 0 0 0 0 1 ( 4 , 0 ) |

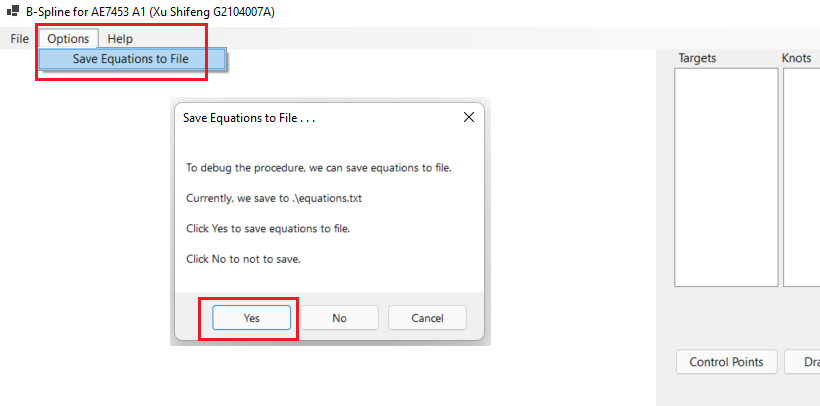
Then remove another “sub-diagonal”:

|  |
| --- |
| 1 0 0 0 0 0 0 ( 0 , 0 )  0 1 0 0 0 0 0 (-0.24, 0.79)  0 0 1 0 0 0 0 (-0.71, 2.36)  0 0 0 1 0 0 0 ( 2.86, 2.57)  0 0 0 0 1 0 0 ( 1.29, -0.64)  0 0 0 0 0 1 0 ( 3.10, -0.21)  0 0 0 0 0 0 1 ( 4 , 0 ) |

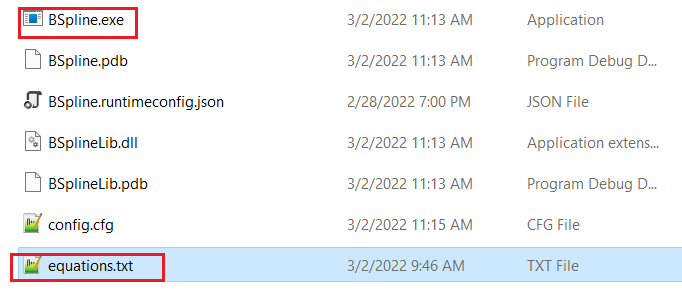
Then we get the result for control points:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **P0** | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** |
| (0, 0) | (-0.24, 0.79) | (-0.71, 2.36) | (2.86, 2.57) | (1.29, -0.64) | (3.10, -0.21) | (4, 0) |

In fact, the above numerical equations are by-product of the program. It is some output when calculating the control points. To enable such output: Options -> Save Equations to File



And the equations will be saved at “current work” directory, usually the same as the BSpline.exe file.



# Examples with 10+ data points

Here are some more examples with more than 10 data points.

## Example 1

|  |  |
| --- | --- |
| **Input** | **Output** |
| 0 0  0 2  2 2  2 0  4 0  4 2  5 2  5 4  6 4  5 5  3 6 | 3  13  0 0 0 0 0.11 0.23 0.34 0.45 0.57 0.62 0.74 0.79 0.87 1 1 1 1  0 0  -0.24 0.79  -0.73 2.37  2.93 2.52  1 -0.47  5.06 -0.66  3.14 2.47  5.89 1.49  4.05 4.49  6.69 3.69  4.49 5.48  3.57 5.80  3 6 |

Such data is also saved in attached file: input-example-1.txt, output-example-1.txt.

A picture containing diagram

Description automatically generated

## Example 2

|  |  |
| --- | --- |
| **Input** | **Output** |
| 0 0  1 0  1 1  0 1  -1 1  -1 0  -1 -1  0 -1  1 -1  2 -1  2 0  2 1  2 2  1 2  0 2 | 3  17  0 0 0 0 0.07 0.14 0.21 0.29 0.36 0.43 0.50 0.57 0.64 0.71 0.79 0.86 0.93 1 1 1 1  0 0  0.40 -0.11  1.20 -0.34  1.19 1.38  0.03 0.83  -1.29 1.29  -0.85 -0.01  -1.31 -1.26  0.10 -0.95  0.90 -0.94  2.29 -1.29  1.94 0.10  1.94 0.90  2.28 2.29  0.93 1.93  0.31 1.98  0 2 |

Such data is also saved in attached file: input-example-2.txt, output-example-2.txt.

Diagram

Description automatically generated with medium confidence

# Some Discussions

I will discuss about:

1. the number of the input data points is less than 4; and
2. there are two points in the input point list have the same coordinates.

## Only 3 data points

If only 3 data points to interpolate, then we have 5 control points. The B-Spline can still interpolate well.

Graphical user interface, application

Description automatically generated

## Two points have the same coordinates

Now let’s come to discuss the situation: Of the target points, two points have the same coordinates. Given such situation, there are still several cases.

### Case 1: the overlapping points are neighbours

If the overlapping points are neighbours in the target points. Such as D1 = D2:

|  |  |
| --- | --- |
| **Notation** | **coordinate** |
| D0 | 0 0 |
| **D1** | **0 2** |
| **D2** | **0 2** |
| D3 | 2 0 |
| D4 | 4 0 |

This means that, in the parameters, t1 = t2. And in the knots, u4 = u5. Then we have:

And thus, we get the following matrix:

|  |
| --- |
| 1 0 0 0 0 0 0 ( 0 , 0 )  54 -108 54 0 0 0 0 ( 0 , 0 )  **0 0 0.50 0.50 0 0 0 ( 0 , 2 )**  **0 0 0.50 0.50 0 0 0 ( 0 , 2 )**  0 0 0 0.25 0.50 0.25 0 ( 2 , 2 )  0 0 0 0 27 -81 54 ( 0 , 0 )  0 0 0 0 0 0 1 ( 2 , 0 ) |

Please note, the high-lighted red rows are identical. So the coefficient matrix is **not full-rank**. We cannot get unique solution for the above equations.

What’s more, this will cause issue when resolving the equations, as it will encounter the typical “divided-by-zero” issue. We should check the points before processing, and make sure to avoid such condition.

And in fact, the above case does not make sense in real life. Our target is to interpolate the points. And it’s meaningless to interpolate two successive overlapping points. Therefore, we should avoid such case when making the input.

### Case 2: the overlapping points have distance 2

Here we define the point distance as the difference of their indexes (subscripts). For example, if input is D0, D1, D2, D3, D4, D0 and D1 have distance 1; and D2 and D4 have distance 2.

If two target points have the same coordinates and have distance 2, we can interpolate them, although a bit weird. For instance, if D1 = D3.

|  |  |
| --- | --- |
| **Notation** | **coordinate** |
| D0 | 0 0 |
| **D1** | **0 2** |
| D2 | 2 2 |
| **D3** | **0 2** |
| D4 | 4 0 |

Graphical user interface

Description automatically generated with medium confidence

### Case 3: the overlapping points have distance 3

If the overlapping points have distance 3.

|  |  |
| --- | --- |
| **Notation** | **coordinate** |
| D0 | 0 0 |
| **D1** | **0 2** |
| D2 | 2 2 |
| D3 | 2 0 |
| **D4** | **0 2** |

We can interpolate well:

Graphical user interface

Description automatically generated

### Case 4: the overlapping points have distance 4 or more

If the overlapping points have distance 4.

|  |  |
| --- | --- |
| **Notation** | **coordinate** |
| D0 | 0 0 |
| **D1** | **0 2** |
| D2 | 2 2 |
| D3 | 2 0 |
| **D4** | **0 2** |

We can interpolate well also:

Graphical user interface

Description automatically generated

### Overlapping points conclusion

Seems only neighbouring overlapping point will cause problem during interpolation, and other cases won’t cause issue.

# Appendix 1: Struggle with indexes

At least 3 days totally, I had been struggling with the indexes of the formula. For example, the course mentioned the second order of the B-Spline function derivatives. It’s easy to notate, but very tedious the implement.

Text, letter

Description automatically generated

Here is how I made it. Thanks to online derivative calculator <https://www.derivative-calculator.net/> , I don’t have to calculate the derivate manually. But still, I need to take very good care of the indexes. Here are some notes I took.

|  |  |  |
| --- | --- | --- |
| **Range** | **Original function** | **2nd order** |
|  | Text  Description automatically generated with medium confidence |  |
|  | Text, letter  Description automatically generated |  |
|  | Text, letter  Description automatically generated |  |
|  | Diagram, text  Description automatically generated |  |

I even made screenshot to backtrack the process.

|  |  |  |
| --- | --- | --- |
| Text  Description automatically generated | Text  Description automatically generated | Text  Description automatically generated |
| Text  Description automatically generated | Text  Description automatically generated | Text  Description automatically generated |

Neither it’s not easy to calculate the final result of the 2nd order derivative:

Graphical user interface, application

Description automatically generated

# Appendix 2: I like B-Spline

A doggerel 😊:

Three weeks the program take,

To show it I cannot wait.

If Prof does not appreciate,

I would wail at the school gate.

Thanks, and regards.