Report for the Project

Wenchong Huang

Date: December 25, 2022

1 Introduction

This project is a simple implemention of piecewise spline interpolation, including linear and cubic. Both are implemented with two different algorithm: ppForm and B-SPline.

For the cubic spline interpolation, five different bondary conditions are supported.

- *natural*: $s''(t_1) = s''(t_N) = 0$.
- *complete*: $s'(t_1) = f'(t_1),\ s'(t_N) = f'(t_N)$.
- *second-derivatives-at-end*: $s''(t_1) = f''(t_1),\ s''(t_N) = f''(t_N)$.
- *periodic*: $s'(t_1) = s'(t_N),\ s''(t_1) = s''(t_N)$.
- *not-a-knot*: $s'''(t_2)$ and $s'''(t_{N-1})$ exist.

natural, *complete*, *second-derivatives-at-end* and *periodic* are supported in both ppForm and B-Spline. And *not-a-knot* is only supported in ppForm.

For how to use the interpolators, see the document. For how to test, firstly **run** `make` **in the sorce code directory**, then read this report to see how to test.

2 Function Test

2.1 Function Fitting

In this part, we use Runge's function as the example. The results see figure 1-4. Run

```
./runge_ppForm > ppForm.txt
./runge_BSpline > BSpline.txt
```

to get the numerical results in two text files. Copy all the text in `ppForm.txt`, replace line 3-7 of `draw_ppForm_cubic_function.m`. Then run the latter code with **matlab**. You will get figure 1.

To get figure 2, replace line 21-22 of `test_ppForm_cubic_function.cpp` with

```
const int n = 11;
const double xvalue[] = {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5};
```

Then run make again, and run `./runge_ppForm > ppForm.txt` again. Do the same work with **matlab**, you will see figure 2.

To get figure 3 and figure 4, do the similar work to `test_BSpline_cubic_function.cpp` and `draw_BSpline_cubic_function.cpp`.

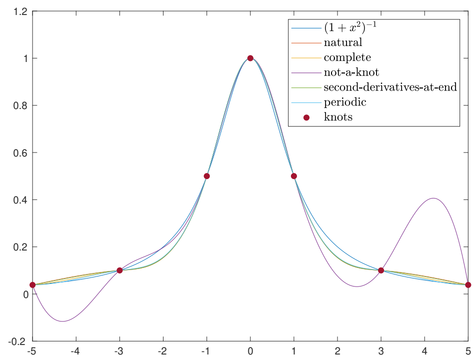

Figure 1: ppForm, 7 knots.

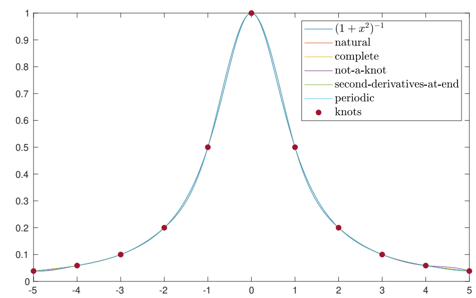

Figure 2: ppForm, 11 knots.

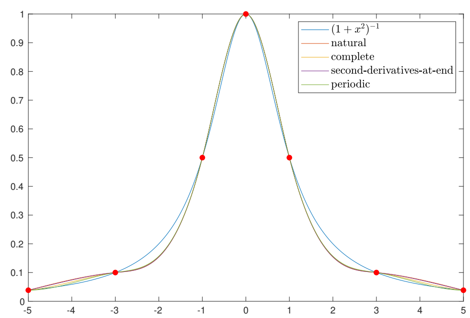

Figure 3: BSpline, 7 knots.

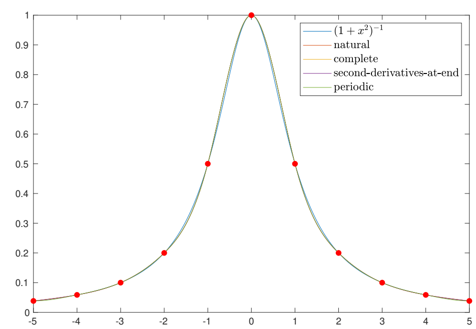

Figure 4: BSpline, 11 knots.

Actually, for the same bondary condition, the interpolation results of ppForm and BSpline are the same. There's no significant difference of bondaries *natural*, *complete*, *second-derivatives-at-end* and *periodic*. But the bondary *not-a-knot* performs poor when we only use 7 knots.

2.2 Curve Generating

In this part, we connect some discrete points in 2D plane with a cubic spline curve. Run

```
./curve > curve.txt
```

to get the numerical in a text file. Copy the text in `curve.txt` and replace line 1-9 of `draw_random_curve.m` with it. Run the latter code with **matlab** then you will see the result.

The result sees figure 5.

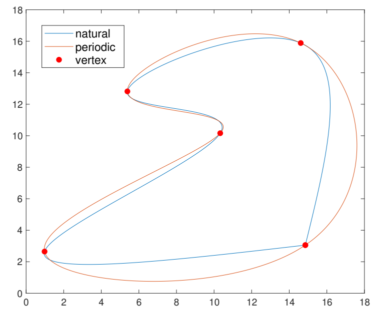

Figure 5: Generated cueve with discrete points.

Two bondaries are supported. To generate closed curve, we suggest using bondary *periodic*. It gives a more smooth curve than bondary *natural*.

2.3 Open Curve Fitting

In this part, we use Helix curve $\rho = \theta$ as the example. Run

```
./helix natural > natural.txt
./helix complete > complete.txt
./helix second-derivatives-at-end > sdae.txt
```

to get the numerical in text files. Copy the text and replace line 4-7 of `draw_curve_helix.m` with it. Run the latter code with **matlab** then you will see the result. Use the different text to get the figure of different bondaries.

The results see figure 6-8.

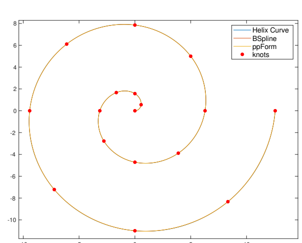

Figure 6: *complete*

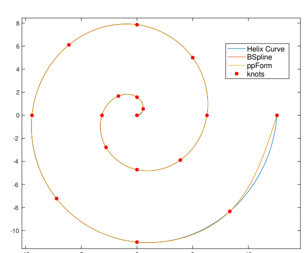

Figure 7: *natural*

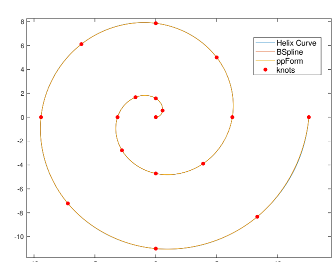

Figure 8: *second-derivatives-at-end*

The bondaries *complete* and *second-derivatives-at-end* performs better.

2.4 Closed Curve Fitting

In this part we use Cardioid curve as the example. Run

```
./cardioid natural > natural.txt
./cardioid complete > complete.txt
./cardioid second-derivatives-at-end > sdae.txt
./cardioid periodic > periodic.txt
```

to get the numerical in text files. Copy the text and replace line 4-7 of `draw_curve_cardioid.m` with it. Run the latter code with **matlab** then you will see the result. Use the different text to get the figure of different bondaries.

The results see figure 9-12.

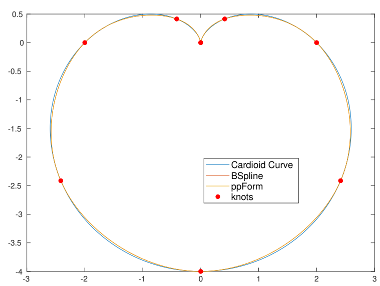

Figure 9: *complete*

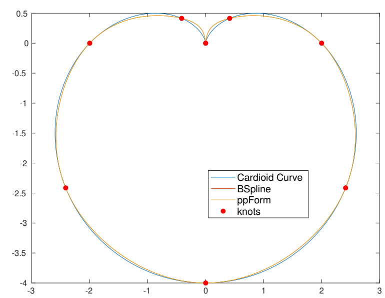

Figure 10: *natural*

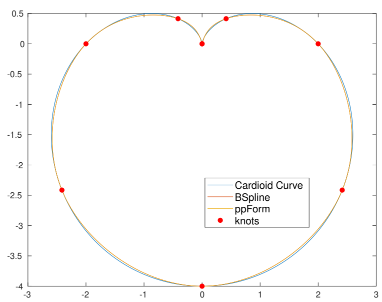

Figure 11: *second-derivatives-at-end*

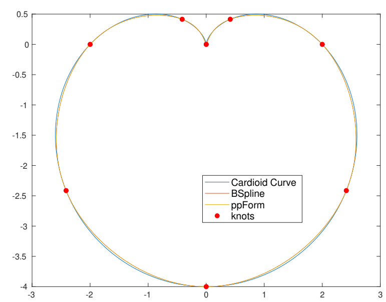

Figure 12: *periodic*

The bondary *natural* performed worse than others. The local behavior at the end sees figure 13-16.

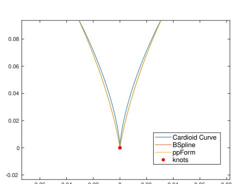

Figure 13: *complete*

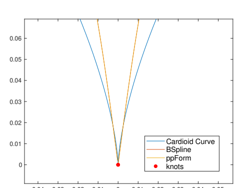

Figure 14: *natural*

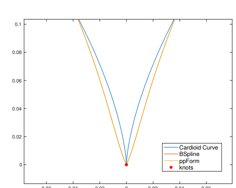

Figure 15: *second-derivatives-at-end*

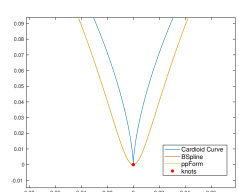

Figure 16: *periodic*

The end of periodic curve is smooth while others are sharp.

3 Programming Assignments

3.1 Assignment A

Run

```
./A N > A.txt
```

to get the numerical result, where N are the number of knots. Copy the text and replace line 3-4 of `draw_A.m`, then run the latter code with **matlab**. You will see the figures as following.

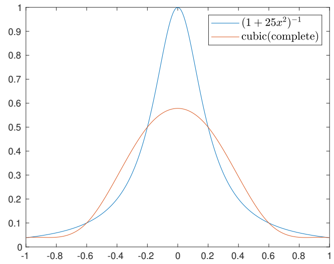

Figure 17: $N = 6$

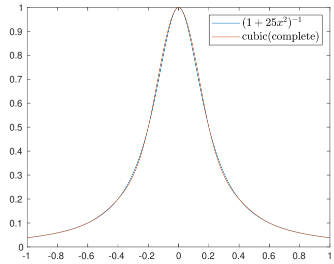

Figure 18: $N = 11$

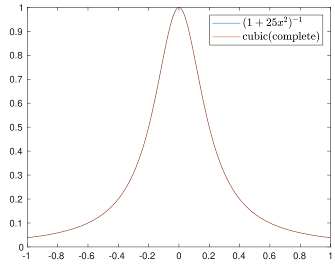

Figure 19: $N = 21$

The max error sees the following table.

N	6	11	21	41	81	501
max error	0.421696	0.0205293	0.00316894	0.000275356	1.609e-05	1.64707e-06

We can plot the logarithm of max error. And compare it with logarithm of N^{-2} and N^{-3}. As the following figure shows.

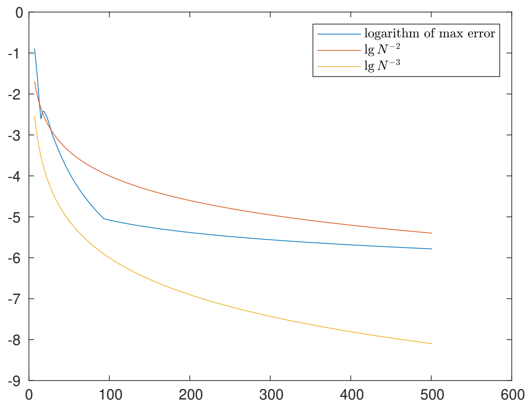

Figure 20: logarithm of max error and N^{-2}, N^{-3}.

So the convergence rate may appoximate to be of order two.

3.2 Assignment C

Run

```
./C > runge.txt
```

to get the numerical result in a text file. Copy the text and replace line 3-4 of `draw_C.m`, then run the latter code with **matlab**. You will see the figure as following.

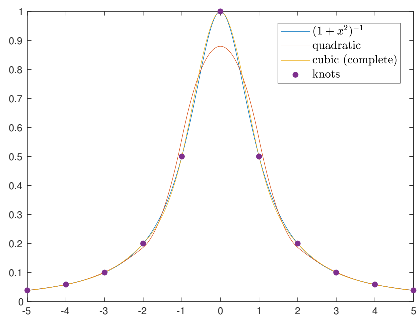

Figure 21: The interpolating result of quadratic B-spline and cubic(complete) B-spline.

3.3 Assignment D

Run `./D` directly to see the result, as the following table shows.

point	-3.5	-3	-0.5	0	0.5	3	3.5
E_S(quadratic)	0	0.00141838	0	0.120238	1.11022e-16	0.00141838	0
E_S(cubic)	0.000505852	0	0.0205266	0	0.0205266	0	0.000505852

The points $-3.5, -0.5, 0.5, 3.5$ are knots of quadratic B-spline, hence the error are close to machine precision. So does $-3, 0, 3$ to cubic B-spline. We could see that the cubic B-spline is more accurate.

3.4 Assignment E

We use the following parametrization (not unit-speed).

$$\gamma(\theta) = \left(\sqrt{3}\sin\theta,\ \ \frac{2}{3}\left(\sqrt{3}\sin\theta + \left(\sqrt{3}|\cos\theta|\right)^{\frac{1}{2}}\right)\right)$$

where $\theta \in [-\frac{\pi}{2}, \frac{3\pi}{2}]$.

We can numerically compute the length of curve between $\gamma(\theta_l)$ and $\gamma(\theta_r)$ with **divide-and-conquer strategy**. So we can use **bisection method** to find $\theta_0, \cdots, \theta_N$, such that the length of curve between $\gamma(\theta_i)$ and $\gamma(\theta_{i+1})$ are equal for $i = 0, ..., N-1$.

Run

```
./E natural 10 > E.txt
```

to get the numerical result of natural cubic B-Spline with 10 vertexes. Also replace 10 with 40, 160 and replace `natural` with `periodic` to get other results. Copy the text to `draw_E.m` to generate the figures as following.

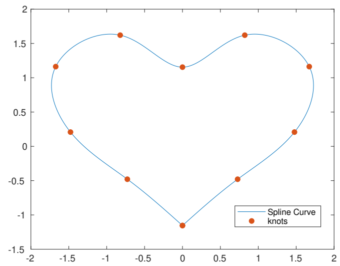

Figure 22: natural, 10 knots.

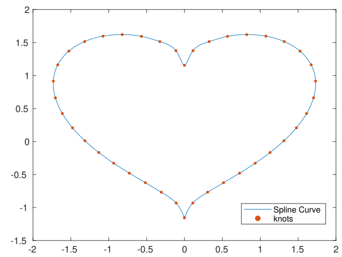

Figure 23: natural 40 knots.

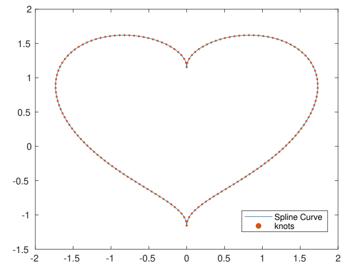

Figure 24: natural, 160 knots.

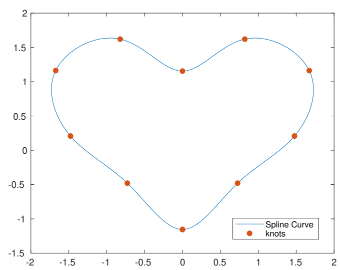

Figure 25: periodic, 10 knots.

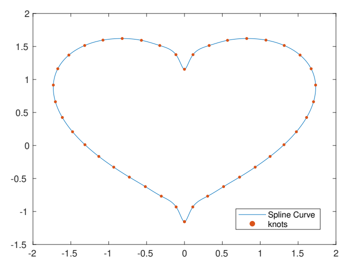

Figure 26: periodic 40 knots.

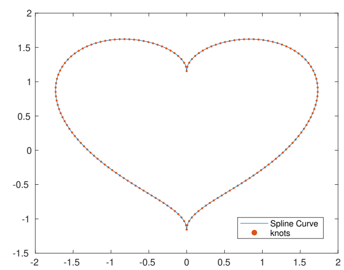

Figure 27: periodic, 160 knots.

As we can see, fit the shape with 40 knots is enough.

Morever, notice the end of the shape is sharp. But boundary *periodic* always gives smooth end. So use boundary *natural* is more appropriate.