CZ7453 Assignment 2 Report

Wu Ziqing G2105028H ziqing002@e.ntu.edu.sg

April 15, 2022

Task 1: Graph Plotting

With the help of the python library, my version of the graph for the parametric function $r(u)$ ($h = 45$) is plotted as shown in Figure 1.

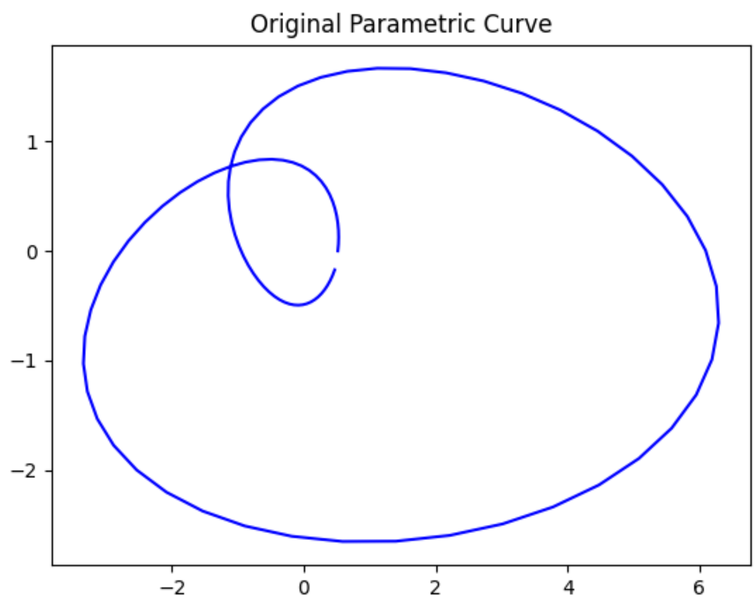

Figure 1: Graph for the Parametric Planar Curve

Task 2: Least Square Fitting

To perform the least square fitting, we first sample a list of $\mathbf{u} \in \mathbb{R}^{1\times n}$ and then locate n data points ($\mathbf{x} \in \mathbb{R}^{1\times n}, \mathbf{y} \in \mathbb{R}^{1\times n}$) from the original curve. Since the line is assumed to be cubic, we can establish two linear systems for $(\mathbf{u}, \mathbf{x})$ and $(\mathbf{u}, \mathbf{y})$ respectively in the form of $A\mathbf{c} = \mathbf{b}$, where:

$$A = \begin{bmatrix} 1, u_1, u_1^2, u_1^3 \\ 1, u_2, u_2^2, u_2^2 \\ \cdots \\ 1, u_n, u_n^2, u_n^2 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} c_1^x \\ c_2^x \\ c_3^x \\ c_4^x \end{bmatrix} \text{ or } \begin{bmatrix} c_1^y \\ c_2^y \\ c_3^y \\ c_3^y \end{bmatrix}, \mathbf{b} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \text{ or } \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}$$

Since the size of A is limited in this problem setting, to minimize the square error, we directly find the $\mathbf{c}$ which can minimise $(A\mathbf{c} - b)^2$. In the other words, we solve for $\mathbf{c}$ which can satisfy $\frac{1}{2}\frac{\partial(A\mathbf{c}-b)^2}{\partial \mathbf{c}} = A^\top A\mathbf{c} - A^\top \mathbf{b} = 0$.

Below we discuss some of the sampling method we experimented.

1. ***Sampling with Uniform Distribution***

   We sample 100 u values uniformly from the interval of $[0, 1]$. Based on the obtained $\mathbf{c}$, the parametric equation can be expressed as:

$$r(u) = \begin{cases} x(u) = -0.230 - 2.923u + 19.218u^2 - 17.168u^3 \\ y(u) = 1.017 - 13.177u + 31.000u^2 - 19.516u^3 \end{cases}$$

   and the fitted curve is shown in Figure 2.

2. ***Random Sampling***

   Then we try to randomly sample 100 data points from the curve and fit the cubic curve. The resulted graph is shown in Figure 3

3. ***Sampling with Normal Distribution***

   I also try to sample 100 u values based on a truncated normal distribution, with $\mu = 0.5, \sigma = 1$ and bounded within $[0, 1]$. The resulted graph is shown in Figure 4

4. ***Sampling with more Even Distance***

   It is noticed that all the above three sampling methods appears to heavily sample the top-left part of the curve where the velocity is faster. In order to come up with a more evenly-separated sampling scheme, I experimented the following method.

   First, I uniformly sampled 1,000 data points and calculated the total Euclidean distance between all consecutive points, which is 2.73. The purpose is to bucketize the 1,000 points into bins 1/10 in number (around 100) whose distance is even and then we can take one representative point from each bin. Thus, starting from first point $u = 0$, we pick the first point whose cumulative distance is more than 0.273 from the point and continues. In this way we picks in total 93 data points which are fairly evenly distributed along the line. The fitted curve is shown in Figure 5.

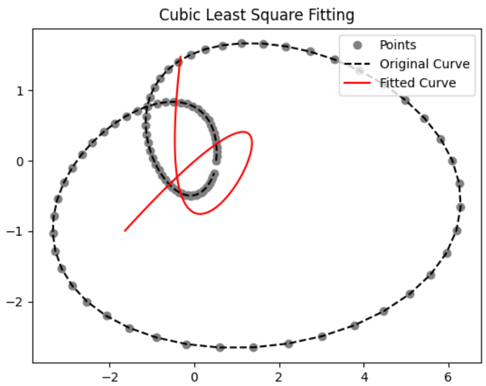

Figure 2: Uniform Sampling

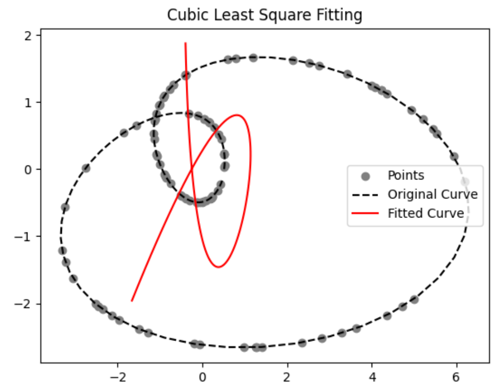

Figure 3: Random Sampling

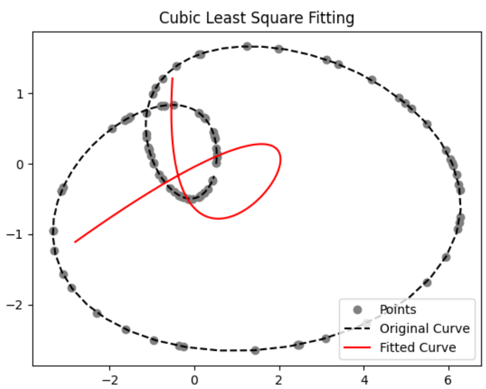

Figure 4: Normal Sampling

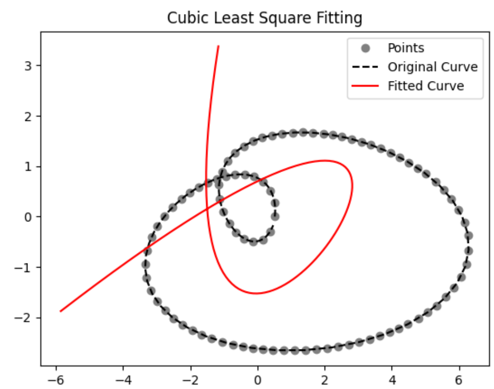

Figure 5: Even-distance Sampling

In conclusion, while uniform distribution performs quite stable, the even-distance sampling give arguably a closer approximation to the original shape. However, the least square fitting approach still fails to yield satisfactory results. It may be due to the limited expressiveness of a cubic function.

Task 3: B-spline Interpolation

With the help of the program in assignment 1, we are able to first sample a few data and then interpolate the points with a natural cubic b-spline.

The expression of the obtained b-spline curve is

$$r(u) = \begin{cases} x(u) = \sum_{i=0}^{n} Px_i N_i^k(u) \\ y(u) = \sum_{i=0}^{n} Py_i N_i^k(u) \end{cases}$$

, where knots $\mathbf{u} = \{0, 0, 0, 0, 0.0319, 0.1180, 0.2230, 0.5301, 0.6649, 0.8766, 0.9340, 0.9734, 1, 1, 1, 1\}$ and control points are

$$(Px, Py) = \begin{bmatrix} (0.518, 0.0) \\ (0.431, 0.258) \\ (0.109, 1.214) \\ (-2.382, 0.789) \\ (-4.145, -3.909) \\ (5.198, -4.375) \\ (5.133, 2.363) \\ (-0.177, 2.864) \\ (-1.289, -0.012) \\ (-0.133, -0.683) \\ (0.297, -0.319) \\ (0.47, -0.172) \end{bmatrix}$$

We further discuss the effect of sampling method, as well as sampling size with experiments.

- ***Sampling Method***

  Figure 6, 7, 8 and 9 shows the effect of the four different sampling methods described in previous section. Notice that the sampling size are all kept at 10.

  It is clear that the sampling methods play an important role in the final result. As data points with proper ordering and relatively even distribution can produce smoother b-spline, only uniform sampling or equal-distance sampling can produce decent result. What is more, since more points at the high-velocity region can produce more accurate fitting, uniform sampling is believed to have better result than equal-distance sampling.

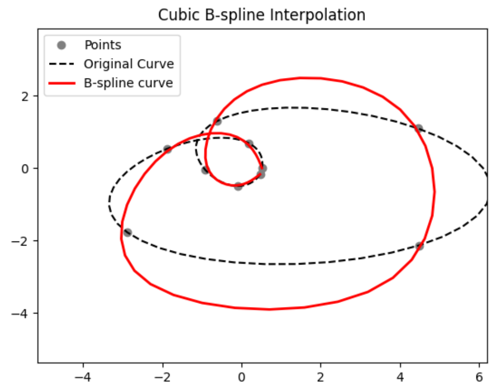

Figure 6: Uniform Sampling

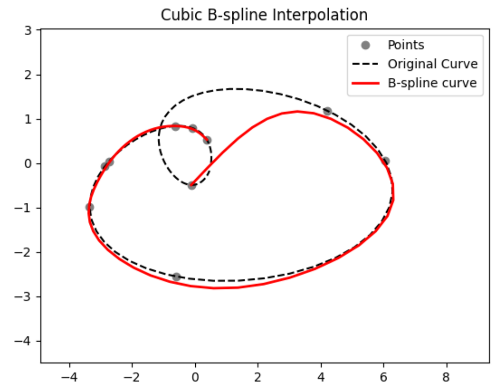

Figure 7: Random Sampling

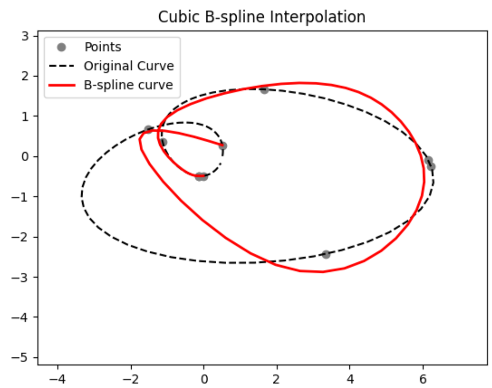

Figure 8: Normal Sampling

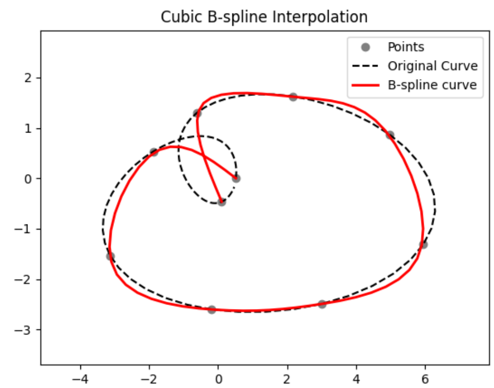

Figure 9: Even-distance Sampling

- ***Sampling Size***

  The effect of sampling size is also analyzed with experiments. We uniformly sample 10, 20, 30 and 40 data points along the original curve for interpolation. The results are shown in Figure 10, 11, 12 and 13.

  It is expected that the more data points are sampled, the more accurate the interpolation. When size=20, the fit is already very decent. When data points increases to 40, the shape of the b-spline is almost identical to the original curve.

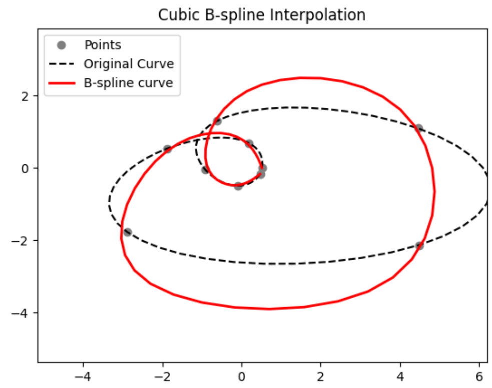

Figure 10: 10 Data Points

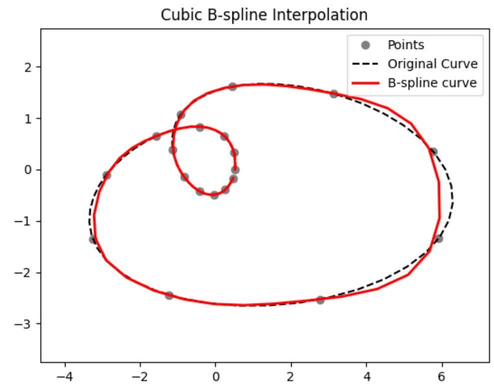

Figure 11: 20 Data Points

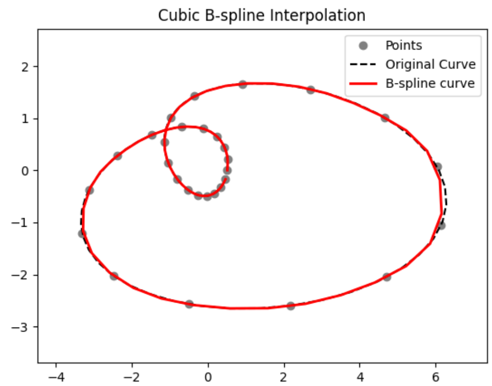

Figure 12: 30 Data Points

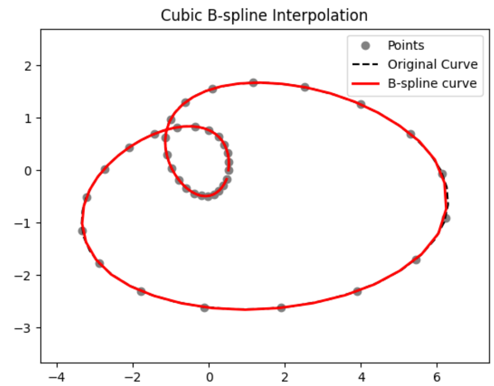

Figure 13: 40 Data Points

Task 4: Trigonometric Interpolation

In order to approximate curve $r(u)$ by a trigonometric interpolation curve that can be represented by the eight basis functions $\{1, \cos(2\pi u), \sin(2\pi u), \cos(4\pi u), \sin(4\pi u), \cos(6\pi u), \sin(6\pi u), \cos(8\pi u)\}$, I follow the trigonometric interpolation algorithm with the following steps (take x as an example and the same applies to y).

1. 8 x points are sampled from uniform u in range $[0, 1)$, i.e.,

$$\{x(0), x(\frac{1}{8}), x(\frac{2}{8}), x(\frac{3}{8}), x(\frac{4}{8}), x(\frac{5}{8}), x(\frac{6}{8}), x(\frac{7}{8})\}$$

2. $\mathbf{x}$ undergoes discrete Fourier factorization with the help of python's `fft` library, to obtain $\hat{\mathbf{x}} \in \mathbb{R}^{1\times 8}$, where $\hat{x}_j = a_j + ib_j$. Note that the python implementation does not normalize the value by $\sqrt{N}$.

3. Based on the derivation showcased in the lecture, we can obtain the parametric expression of x

in the form of

$$P_n(u) = \frac{a_0}{8} + \frac{2}{8}\sum_{k=1}^{3}[a_k \cos 2\pi uk - b_k \sin 2\pi uk] + \frac{a_4}{8}\cos 8\pi u$$

The approximation result is shown in Figure 14. While the approximation is quite smooth, the result is not yet satisfactory. If we increase the base function space to 20 functions as shown in Figure 15, the approximation is very close to the original curve.

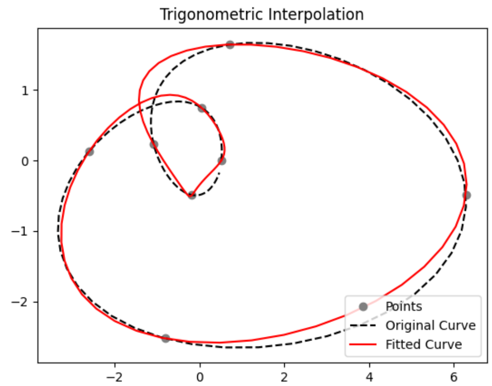

Figure 14: 8 Base Functions

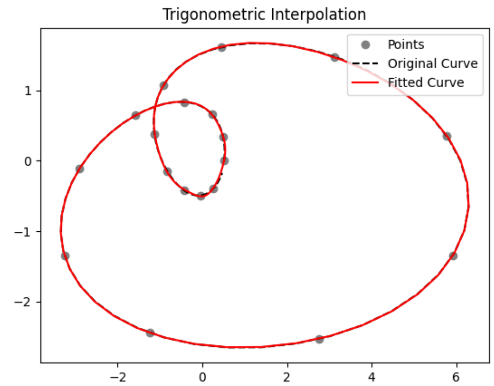

Figure 15: 20 Base Functions

Task 5: Integral Computation

To compute the integral $\int_0^1 x(u)\,du$ with composite Simpson's rule, I first split the interval $[0, 1]$ into $2m$, $(m = 100)$, pieces of equal length $h = 1/2m$. As a result, we have $2m + 1$ points on the curve, $x(0), x(u_1), ..., x(u_{200})$. Then we could follow the equation provided in the lecture,

$$\begin{aligned}
\int_0^1 x(u)\,du &= h\sum_{i=0}^{m-1}\int_{u_{2i}}^{u_{2i+2}} x(u)\,du \\
&\approx h\sum_{i=0}^{m-1}(\frac{1}{3}x(u_{2i}) + \frac{4}{3}x(u_{2i+1}) + \frac{1}{3}x(u_{2i+2})) \\
&= \frac{h}{3}(x(0) + x(1) + 4\sum_{i=0}^{m-1} x(u_{2i+1}) + 2\sum_{i=1}^{m-1} x(u_{2i}))
\end{aligned}$$

The final result of the integration is 0.35836. To obtain a more accurate approximation, we may need to increase the number of intervals, i.e., reduce each interval's length h.