Assignment 3 Report

Zhang Jiehuang

G1842648F

T1:

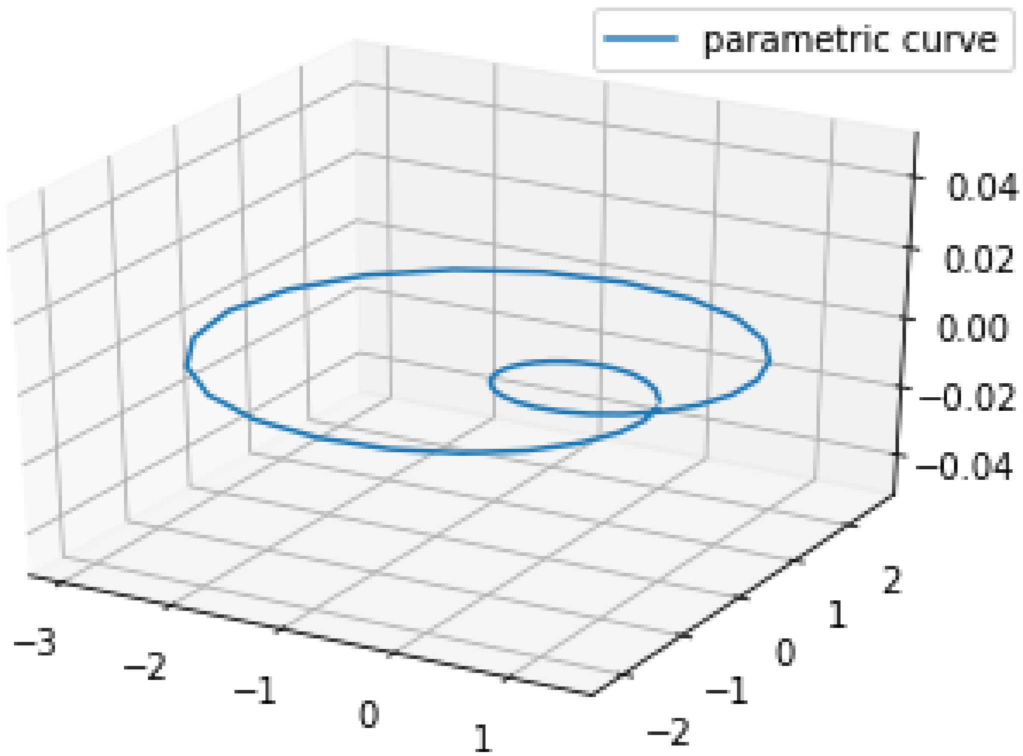


Fig 1: plot of the parametric curve

T2:

We use np.linspace to obtain the 5 data points stated below:

Value
[0.97156754 -0.40860968 1.202058   -2.97416603 1.04090858]
[0.   0.01700497 -0.10022485 0.37304845 -0.17479229]

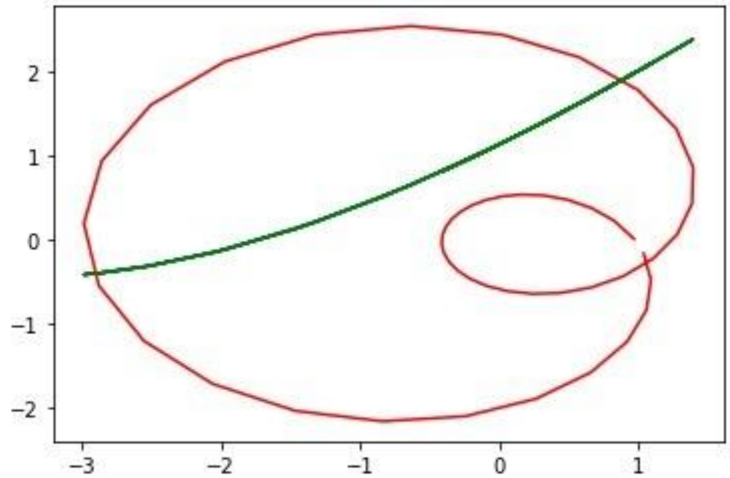
Using np.polyfit, we obtain a least square interpolation of the 5 data points above:

Polynom coeffs [1.12282894 0.81166229 0.07369737 -0.0081886]

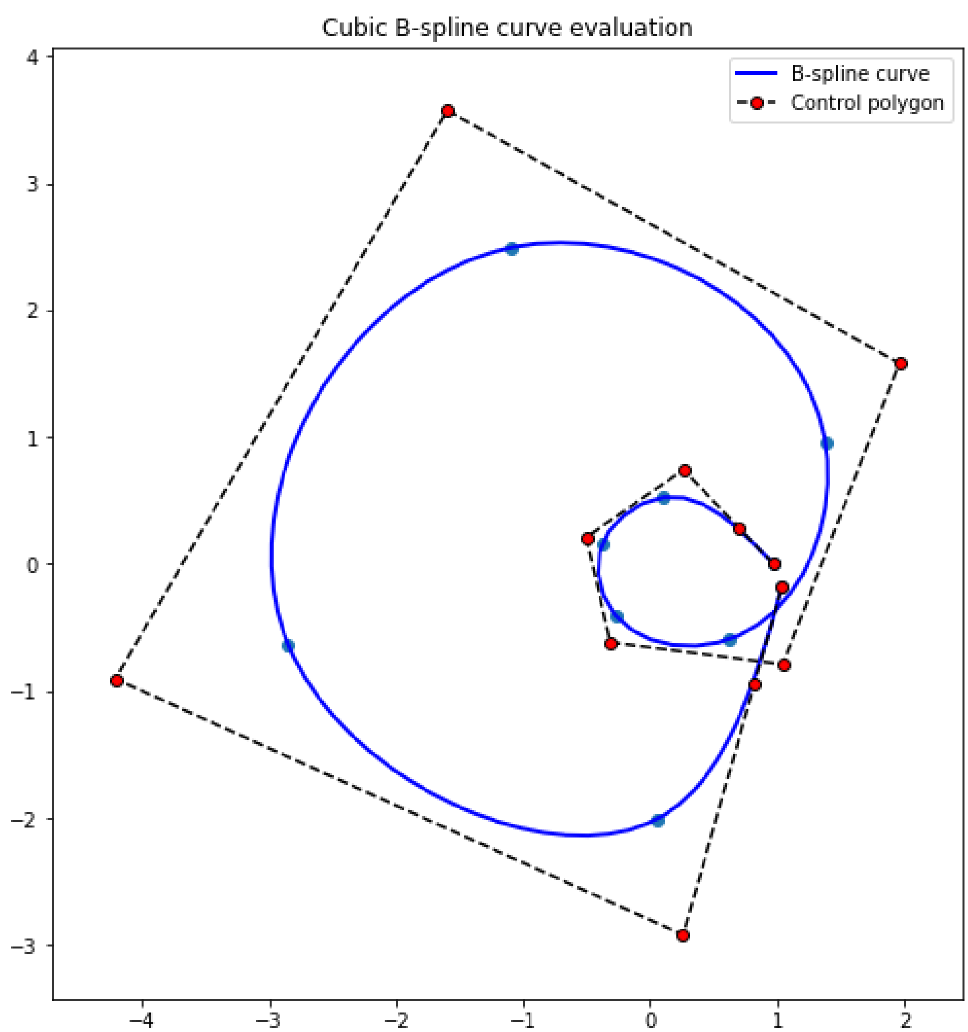
Thus the least square curve is as follows:

Y(x) = 1.1228x^3 + 0.8117x^2 + 0.0737x – 0.0082

We obtained the following plot of the least square curve using matlibplot as show below: the green curve is the least square curve, while the red curve is the parametric curve.



T3



We perform a B spine interpolation of the parametric curve using 10 data points evenly spaced out according to the parametric equation, from 0 to 1. The same technique used in assignment 1 was performed here and the above cubic B spine curve was generated. We also obtained an output file in which the following format of degree, knot vector and control points were displayed:

```
3
12
[0.         0.         0.         0.         0.0610127  0.09742073
 0.13199479 0.18627717 0.29052528 0.46551806 0.68132987 0.87456041
 1.         1.         1.         1.        ]
[[ 9.71567537e-01 -5.77065139e-19]
 [ 6.98880462e-01  2.85034698e-01]
 [ 2.63473169e-01  7.40157794e-01]
 [-5.03760753e-01  2.08845803e-01]
 [-3.12874385e-01 -6.17820576e-01]
 [ 1.04518976e+00 -7.92076708e-01]
 [ 1.96248804e+00  1.57966902e+00]
 [-1.59692865e+00  3.57172975e+00]
 [-4.21106722e+00 -9.06480601e-01]
 [ 2.61020696e-01 -2.92357173e+00]
 [ 8.20627842e-01 -9.51189985e-01]
 [ 1.04090858e+00 -1.74792286e-01]]
```

3 is the degree of the curve, 12 indicates there are a total of 12 control points.

The first array is the knot vectors while the second array shows the 12 control points.

T4:

In order to produce a trigonometric interpolation of this parametric curve, we follow the method proposed in the lecture notes:

Trigonometric interpolation (cont)

(4) For real x_j, we have $a_{n-j} = a_j, b_{n-j} = -b_j$. Thus

$$P_n(\theta) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{\frac{n}{2}-1} [a_k \cos k\theta - b_k \sin k\theta] + \frac{a_{\frac{n}{2}}}{\sqrt{n}} \cos \frac{n}{2}\theta$$

(5) Similar development for odd n.

(6) By reparameterization, we obtain general trigonometric interpolation scheme. That is, given an interval $[c, d]$ and positive integer n, let $t_j = c + j(d-c)/n$ for $j = 0, \cdots, n-1$. The vector $x = (x_0, \cdots, x_{n-1})$ consists of n *real* numbers. Let $\{a_j + ib_j\}$ denote the DFT $F_n x$ of x. Then an *oder* n *trigonometric interpolating function* is

$$\boxed{P_n(t) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{\frac{n}{2}-1} \left[a_k \cos \frac{2\pi(t-c)}{d-c}k - b_k \sin \frac{2\pi(t-c)}{d-c}k\right] + \frac{a_{\frac{n}{2}}}{\sqrt{n}} \cos \frac{\pi(t-c)}{d-c}n}$$

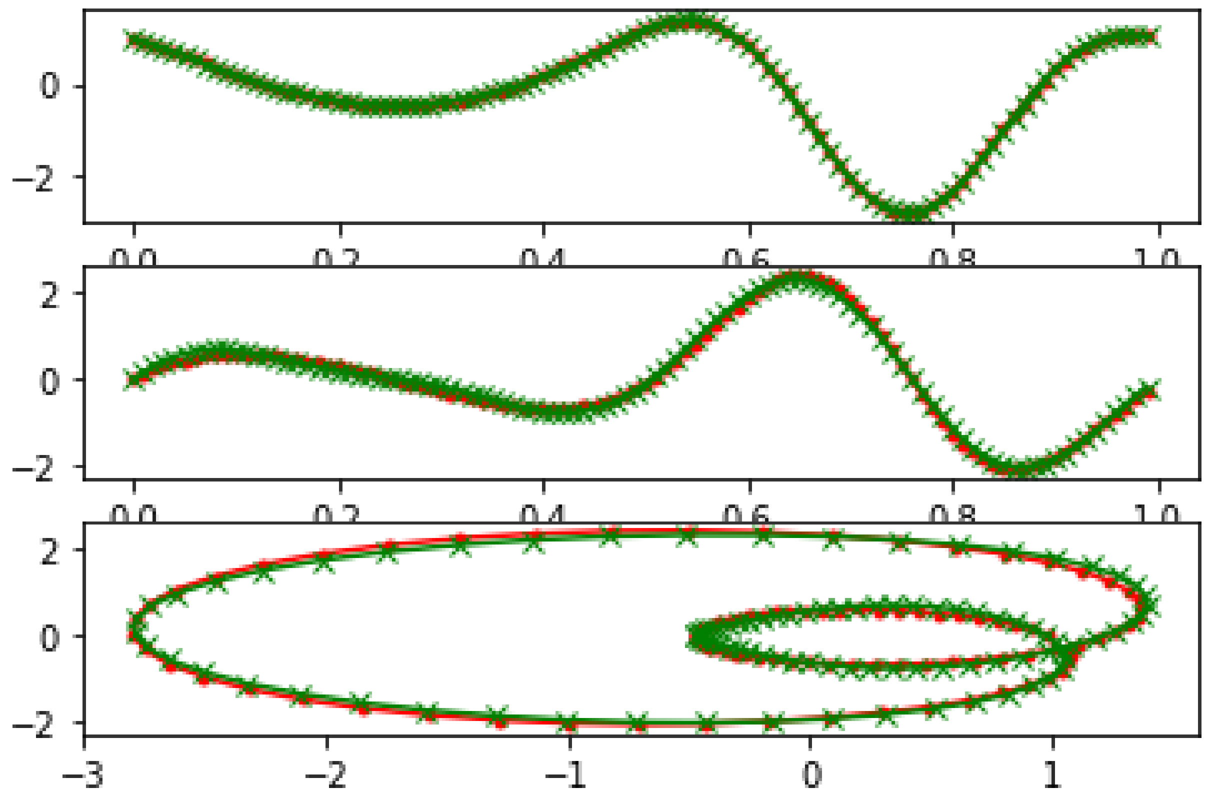
for $t \in [c, d]$.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

-23-

In this case, n = 8, c=0 and d = 1. We then choose 8 points from the original parametric curve and plug it into the equation above. Hence, we obtain 8 equations and can then solve for the 8 unknowns.

After coding the solution, we obtain this plot of the 3 curves: 1) x against u, 2) y against u, 3) parametric vs interpolated curve:



The interpolation results look promising and I believe that we have achieved a satisfactory interpolation. If needed, we can sample even more data points to make the interpolation a better one. For now I believe this results are good enough.

T5:

We use the quad and integrate command from scipy, applied it to the least square fit result from T2:

-0.0081886*x**3+ 0.0736974*x**2+0.811662*x+1.12283

and obtained the following result:

(1.5511796500000001, 1.722155362759992e-14)

The first value is the integration value, while the second is the estimated error.