CE7453 Numerical Algorithm Assignment 1

Zhang Jiehuang G1842648F

1) Implement cubic B-spline interpolation

To perform the cubic B-spline interpolation in python, we use the spyder environment with packages numpy, matplotlib, math and scipy. The input data is stored as a txt file in the environment, and we simply need to call it. As such the ease of manipulating the data points is easier especially for beginners who are not familiar with programming.

n is the order of the curve and number of de boor points, in this case n = 4

k is the degree of the curve, in this case 3

n_knots is the number of knots, we derive it n_knots = n+1+2*k

2) Parameterisation

We choose chord length parameterisation in order to avoid the Runge phenomenon. V vectors consisting of n+1 parameters with values t within [0,1]. After appending k numbers of 0 and 1 at the beginning and ending of knot vector respectively, we obtain the following knot vector:

[0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1]

3) Linear system for input data

We initiate a matrix N of length (n+1)*(n+3), hence in this case, our initial matrix N is 5*7. We assign the values of N[i][i+2] for i in range (2) using the cubic B-spline basis function:

- Cubic B-spline basis

$$N_i^3(u) = \begin{cases} \dfrac{(u-u_i)^3}{(u_{i+1}-u_i)(u_{i+2}-u_i)(u_{i+3}-u_i)}, & u \in [u_i, u_{i+1}) \\ \dfrac{(u-u_i)^2(u_{i+2}-u)}{(u_{i+2}-u_{i+1})(u_{i+3}-u_i)(u_{i+2}-u_i)} + \dfrac{(u_{i+3}-u)(u-u_i)(u-u_{i+1})}{(u_{i+2}-u_{i+1})(u_{i+3}-u_{i+1})(u_{i+3}-u_i)} + \dfrac{(u_{i+4}-u)(u-u_{i+1})^2}{(u_{i+2}-u_{i+1})(u_{i+4}-u_{i+1})(u_{i+3}-u_{i+1})}, & u \in [u_{i+1}, u_{i+2}) \\ \dfrac{(u-u_i)(u_{i+3}-u)^2}{(u_{i+3}-u_{i+2})(u_{i+3}-u_{i+1})(u_{i+3}-u_i)} + \dfrac{(u_{i+4}-u)(u_{i+3}-u)(u-u_{i+1})}{(u_{i+3}-u_{i+2})(u_{i+4}-u_{i+1})(u_{i+3}-u_{i+1})} + \dfrac{(u_{i+4}-u)^2(u-u_{i+2})}{(u_{i+3}-u_{i+2})(u_{i+4}-u_{i+2})(u_{i+4}-u_{i+1})}, & u \in [u_{i+2}, u_{i+3}) \\ \dfrac{(u_{i+4}-u)^3}{(u_{i+4}-u_{i+3})(u_{i+4}-u_{i+2})(u_{i+4}-u_{i+1})}, & u \in [u_{i+3}, u_{i+4}) \end{cases}$$

The matrix N becomes:

[[0. 0. 0. 0. 0. 0. 0.]

[0. 0.25 0.58333333 0.16666667 0. 0. 0.]

[0. 0. 0.16666667 0.66666667 0.16666667 0. 0.]

[0. 0. 0. 0.16666667 0.58333333 0.25 0.]

[0. 0. 0. 0. 0. 0. 0.]]

For the first and last rows of matrix, we apply natural conditions: r”(u)=0 at t0 =(u3) and tn =u(n+3). This gives us the following equations:

$$\frac{d^2 N_0^3(t_0)}{du^2} P_0 + \frac{d^2 N_1^3(t_0)}{du^2} P_1 + \frac{d^2 N_2^3(t_0)}{du^2} P_2 = 0$$

$$\frac{d^2 N_n^3(t_n)}{du^2} P_n + \frac{d^2 N_{n+1}^3(t_n)}{du^2} P_{n+1} + \frac{d^2 N_{n+2}^3(t_n)}{du^2} P_{n+2} = 0$$

We apply these equations in order to updates matrix N. Until now, we are still missing the first and last row of the matrix N, which is the representation of the first and last data point interpolations. It is also easy to verify that

$$N_0^3(t_0) = N_{n+2}^3(t_n) = 1, N_1^3(t_0) = N_2^3(t_0) = N_n^3(t_n) = N_{n+1}^3(t_n) = 0.$$

Hence, we insert [1,0,0,..] and [0,0, … ,1] in the the first and last rows, and the linear system N matrix becomes:

[1. 0. 0. 0. 0. 0. 0]

[96. -144. 48. 0. 0. 0. 0.]

[0. 0.25 0.58333333 0.16666667 0. 0. 0.]

[0. 0. 0.16666667 0.66666667 0.16666667 0. 0.]

[0. 0. 0. 0.16666667 0.58333333 0.25 0.]

[0. 0. 0. 0. 48. -144. 96.]

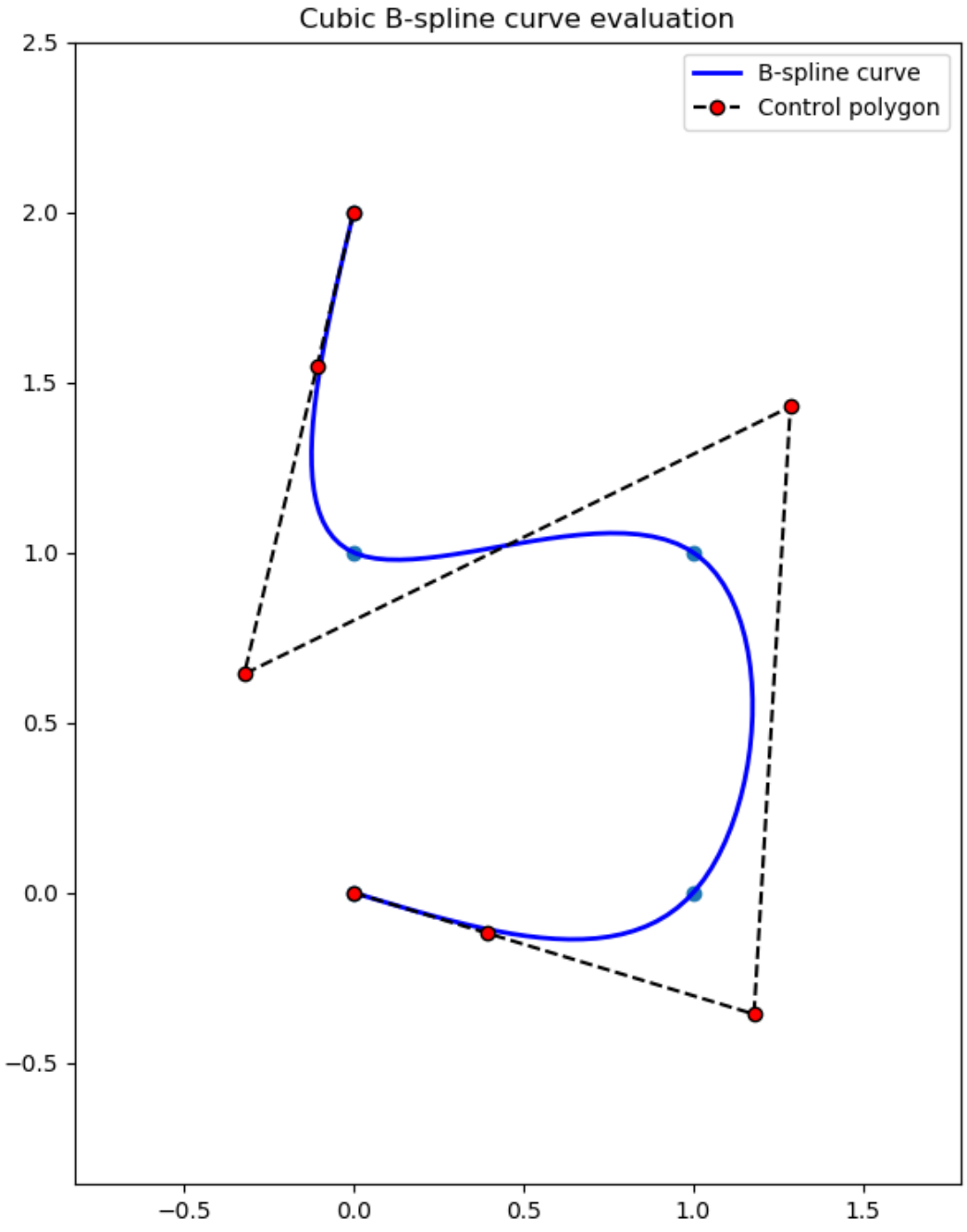
[0. 0. 0. 0. 0. 0. 1]

We then solve the linear system by taking the input points: ([(0,0),(1,0),(1,1),(0,1),(0,2)]).

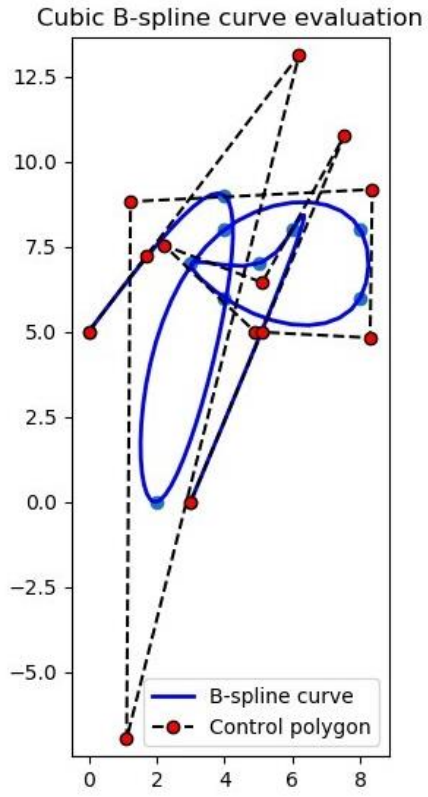
The output control points derived were: ([[0. , -0.],[0.39, -0.12],[1.18, -0.36],[1.29, 1.43],[-0.32, 0.64],[-0.11, 1.55],[-0. , 2.]])

Finally, we define the formula for r(u), then proceed to draw the curve using matlibplot.

$$r(u) = \sum_{i=0}^{n} P_i N_i^k(u), \qquad u \in [u_k, u_{n+1}]$$



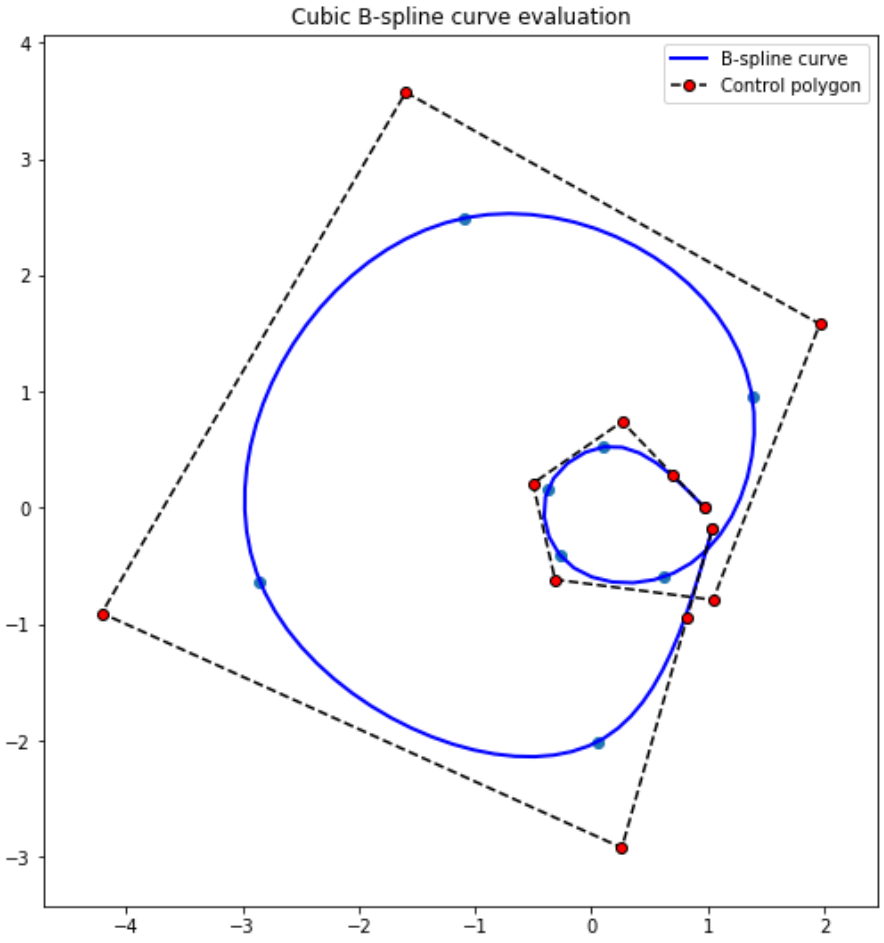
4) Construction of 2 examples with >10 data points



For the first curve, we attempt to write the Chinese character 伊 (my wife's surname), this turns out to be difficult due to the many strokes that were need. However, with some careful investigation, one may be able to make out the Chinese character. Below is the output

```
3
15
[0.         0.         0.         0.         0.01491152 0.07558429
 0.18805403 0.37135727 0.53530867 0.61483678 0.65460084 0.73412895
 0.76224639 0.80201044 0.83012788 1.         1.         1.
 1.        ]
[[ 1.          7.5       ]
 [ 1.29938195  7.53191453]
 [ 2.81690472  7.69368454]
 [-2.8777064   1.9757669 ]
 [ 8.27755179 15.43577872]
 [ 0.51304287 -7.63257361]
 [ 1.30515874  8.90066188]
 [ 8.34279072  9.17170353]
 [ 8.29232615  4.8253791 ]
 [ 4.89161137  4.9988834 ]
 [ 2.22957298  7.53225634]
 [ 5.11267811  6.44472255]
 [ 7.53276455 10.76571839]
 [ 5.0931518   4.97142143]
 [ 3.          0.        ]]
```

For the next curve, we refer to the assignment 3 in which we are required to perform B spline interpolation on a parametric curve. Please see the curve below. Since we are already required to interpolate the parametric curve, I thought might as well kill 2 birds with 1 stone.



Output:

```
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]]
```