Numerical Algorithm Assignment 1

Zhiwei Cao, SCSE, G2100614H

Abstract

This paper discusses the methodology as well as the implementation results for assignment 1. The paper first derive the linear system for cubic B-spline interpolation. Following that, this paper provides the technical details for the linear system solver which is an implementation of the triagonal matrix algorithm. After that, we present the cubic-interpolation result for two examples to validate the correctness of the implementation. Finally, some discussion is provided for the cases when: a) the number of input data points is smaller than 4; b) there are two identical data points in the input data point list.

I. Linear System Estabilishment

In this task, the set contains all input data points $\mathbf{d}_i \in \mathbb{R}^m, i = 0, 1, ..., N-1$ are given and we are required to give the linear equation which determines the control points corresponding the input data points. The input data points in this task are all two dimensional ($m = 2$) data points as shown in the following:

$$\begin{matrix} 0 & 0 \\ 0 & 2 \\ 2 & 2 \\ 2 & 0 \\ 4 & 0 \end{matrix}$$

The first step to determine the linear system is to parametrize the input data points so that each data point corresponds to a unique parameter t in the parametric cubic B-spline curve. In this regard, the Chord length parameterization technique is utilized to parametrize the input data points. To be specific, the parameter t_i for the i-th data point $_i$ is given as:

$$t_i = \begin{cases} 0, & i = 0 \\ \frac{\sum_{k=1}^{i} ||\mathbf{d}_k - \mathbf{d}_{k-1}||_2}{\sum_{k=1}^{N} ||\mathbf{d}_k - \mathbf{d}_{k-1}||_2}, & i = 1, .., N-1 \\ 1, & i = N \end{cases} \tag{1}$$

where $||\cdot||_2$ denotes the Euclidean norm. Plugging the input data points into Eq (1), the corresponding parametrization result is:

$$\mathbf{t} = [0, 0.25, 0.50, 0.75, 1.0]$$

After input data point parametrization, the idea for cubic B-spline interpolation is that we want to find a curve given by $r(t) = \sum_{i=1}^{N} N_i^3(t)\mathbf{p}_i$ so that $r(t_i) = \mathbf{d}_i \ \forall i = 1, 2, ..., N$, where $N_i^3(t)$ is the i-th basis function for cubic B-spline curve and $\mathbf{p}_i \in \mathbb{R}^2$ is the i-th control point. To achieve this goal, we need to specify the knot vector and the control points for this cubic B-spline curve. In terms of the knot vector, as the curve must interpolate the first and the last data point and the degree of the curve is 3, we can immediately write down the knot vector $\mathbf{u}$ as:

$$\mathbf{u} = [0.0, 0.0, 0.0, 0.0, 0.25, 0.50, 0.75, 1.0, 1.0, 1.0, 1.0]$$

Noted that $t_0 = 0.0$ as well as $t_4 = 1.0$ are repeated 4 times to ensure the curve interpolates $\mathbf{d}_0$ and $\mathbf{d}_4$.

After we obtain the knot vector $\mathbf{u}$, the only thing we need to do is to find out the $N+3$ control points $\mathbf{p}_i, \ i = 0, 1, ..., N+2$ so that the resulting curve will interpolate the input data points. To achieve this, we should establish $N+3$ equations since we have $N+3$ control points to be determined. From the interpolation requirements, we can immediately obtain $N+1$ equations

$$r(t_k) = \sum_{i=0}^{N+2} N_i^3(t_k)\mathbf{p}_i = \mathbf{d}_i, \quad i = 0, 1, 2, ..., N \tag{2}$$

Furthermore, according to the vanishing property of the cubic B-spline basis function, Eq. (2) can be simplified as

$$r(t_k) = N_i^3(t_k)\mathbf{p}_k + N_i^3(t_{k+1})\mathbf{p}_{k+1} + N_i^3(t_{k+2})\mathbf{p}_{k+2} = \mathbf{d}_k, \quad i = 0, 1, 2, ..., N \tag{3}$$

For another two equation, we use the C^2 continuity condition gives by $r''(t) = 0$ at the start and the end point:

$$r''(t_0) = \frac{d^2 N_0^3(t_0)}{du^2}\mathbf{p}_0 + \frac{d^2 N_1^3(t_0)}{du^2}\mathbf{p}_1 + \frac{d^2 N_2^3(t_0)}{du^2}\mathbf{p}_2 = 0 \tag{4}$$

$$r''(t_N) = \frac{d^2 N_N^3(t_0)}{du^2}\mathbf{p}_N + \frac{d^2 N_{N+1}^3(t_0)}{du^2}\mathbf{p}_{N+1} + \frac{d^2 N_{N+2}^3(t_0)}{du^2}\mathbf{p}_{N+2} = 0 \tag{5}$$

By combining Eq. (3), Eq. (4) and Eq. (5), we finally establish $N+3$ equations for $N+3$ control points so that we can solve the linear system to find the $N+3$ control points which gives a cubic B-sline curve that interpolates all input data points. By running the implemented program for this case, we can obtain the corresponding coefficient matrix for the linear system as:

$$\mathbf{A} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \\ 96.0 & -144.0 & 48.0 & 0.0 & 0.0 & 0.0 & 0.0 & \\ 0.0 & 0.25 & 0.583 & 0.167 & 0.0 & 0.0 & 0.0 & \\ 0.0 & 0.0 & 0.167 & 0.667 & 0.167 & 0.0 & 0.0 & \\ 0.0 & 0.0 & 0.0 & 0.167 & 0.583 & 0.25 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 48.0 & -144.0 & 96.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & \end{bmatrix}. \tag{6}$$

Noted that as the input data points are two-dimensional, we must solve the x and y coordinates of the control points separately, and the corresponding constant vector for the two linear system are:

$$\mathbf{b}_x = [0, d_{0,x}, d_{1,x}, d_{2,x}, d_{3,x}, d_{4,x}, 0] = [0, 0, 0, 2, 2, 4, 0]^T \tag{7}$$

$$\mathbf{b}_y = [0, d_{0,y}, d_{1,y}, d_{2,y}, d_{3,y}, d_{4,y}, 0] = [0, 0, 2, 2, 0, 0, 0]^T \tag{8}$$

Subsequently, we solve the following two linear systems to obtain the x and y coordinates of the control points:

$$\mathbf{A}\mathbf{p}_x = \mathbf{b}_x \tag{9}$$

$$\mathbf{A}\mathbf{p}_y = \mathbf{b}_y \tag{10}$$

where $\mathbf{p}_x$ and $\mathbf{p}_y$ are the vectors for the x and y coordinates of the control points. By solving the equations, we can obtain the control points as:

$$\begin{array}{ll} 0.000 & 0.000 \\ -0.238 & 0.786 \\ -0.714 & 2.357 \\ 2.857 & 2.571 \\ 1.286 & -0.643 \\ 3.095 & -0.214 \\ 4.000 & 0.000 \end{array}$$

II. Tridiagonal Matrix Algorithm for Solving Tridiagonal Linear System

In this section, we demonstrate the tridiagonal matrix algorithm that we applied in this paper to solve the linear system (9) and (10). As we can see from Eq. (6), the linear system that we deal with is a tridiagonal linear system. A tridiagonal linear system with n unknown variables can be written in a general form as:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \quad i = 1, 2, .., n \tag{11}$$

where $a_1 = c_n = 0$. The matrix form can be written as:

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{12}$$

Such a tridiagonal system can be solved efficiently in $\mathcal{O}(n)$ instead of $\mathcal{O}(n^3)$ with Gauss Elimination. Typically, this algorithm is a special case for Gauss Elimination. Let us write down the n equations to be solved explicitly as:

$$\begin{aligned} 0 + b_1 x_1 + c_1 x_2 &= d_1 \\ a_2 x_1 + b_2 x_2 + c_2 x_3 &= d_2 \\ &\dots \\ a_i x_{i-1} + b_i x_i + c_i x_{i+1} &= d_i \\ &\dots \\ a_n x_{n-1} + b_n x_n + 0 &= d_n \end{aligned}$$

Similar to Gauss Elimination, we first try to eliminate the first variable x_1 by equation2 $\cdot\, b_1 -$ equation1 $\cdot\, a_2$, which yields:

$$(b_2 b_1 - c_1 a_2) \cdot x_2 + c_2 b_1 \cdot x_3 = d_2 b_1 - d_1 a_2$$

Similarly, we can apply the same technique to the third and the forth equation, yielding:

$$[b_3(b_2 b_1 - c_1 a_2) - c_2 a_3] \cdot x_3 + c_3 \cdot (b_2 b_1 - c_1 a_2) \cdot x_4 = d_3 \cdot (b_2 b_1 - c_1 a_2) - (d_2 b_1 - d_1 a_2) \cdot a_3$$

At this time, the second variable x_2 is eliminated. If we apply such a technique iteratively until the $n-1$-th variable has been eliminated, we can obtain one equation which contains only one variable x_n. Solving this equation will yield the solution for x_n, and then back substitution is implemented to obtain the solution for the remaining $n-1$ variables.

Now we provide the formulation for the tridiagonal matrix algorithm. By examining the aforementioned variable elimination procedure, we define the modified coefficients (variables in tilde) recursively:

$$\begin{aligned} \tilde{a}_1 &= 0, \\ \tilde{b}_1 &= b_1, \\ \tilde{b}_i &= (b_i \tilde{b}_{i-1} - \tilde{c}_{i-1} a_i), \quad i = 2, 3, ..., n \\ \tilde{c}_1 &= c_1, \\ \tilde{c}_i &= c_i \tilde{b}_{i-1}, \quad i = 2, 3, ..., n \\ \tilde{d}_1 &= d_1, \\ \tilde{d}_i &= d_i \tilde{b}_{i-1} - \tilde{d}_{i-1} a_i, \quad i = 2, 3, ..., n \end{aligned}$$

By using these modified coefficients, we can obtain an equivalent linear system where there are two unknown variables in any equation:

$$\tilde{b}_i \cdot x_i + \tilde{c}_i \cdot x_{i+1} = d_i \tilde{b}_{i-1} - \tilde{d}_{i-1} a_i$$

Furthermore, if we ignore the risk of dividing by zero for $\tilde{b}_i$, we can define another set of modified coefficients which make our formulation even simpler:

$$\begin{aligned} a'_1 &= 0, \\ b'_1 &= 1, \\ c'_1 &= \frac{c_1}{b_1}, \\ c'_i &= \frac{c_1}{b_i - c'_{i-1} a_i}, \\ d'_1 &= \frac{d_1}{b_1}, \\ d'_i &= \frac{d_i - d'_{i-1} a_i}{b_i - c'_{i-1} a_i} \end{aligned}$$

Now we obtain a simpler system with the first $n-1$ equations have exact two variables and the last equation has only one variable x_n as:

$$\begin{aligned} x_i + c'_i x_{i+1} &= d'_i, \quad i = 1, 2, ..., n-1 \\ x_n &= d'_n \end{aligned}$$

We first let x_n be d'_n, and then plug in the solution of x_n into the penultimate equation, which gives the solution for x_{n-1}. Likewise, we can perform this procedure iteratively to obtain the solution of all unknown n variables.

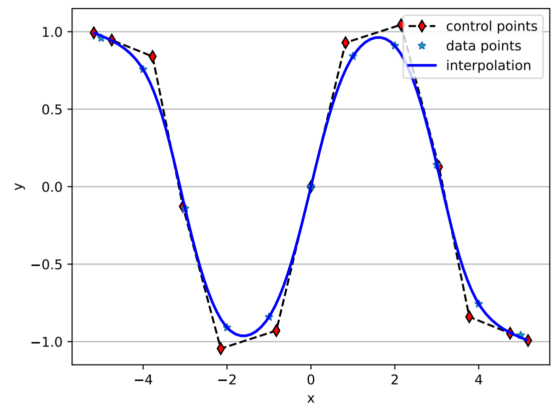

Fig. 1: Interpolation results for example 1. The green solid curve is the interpolation result. The circle points are the input data points. The red points are the corresponding control points.

III. Curve Visualization

In this section, the interpolation results will be illustrated for two different input data point sets. The first input data point set contains the following 11 data points:

$$\begin{array}{ll} -5 & 0.95892427 \\ -4 & 0.7568025 \\ -3 & -0.14112 \\ -2 & -0.9092974 \\ -1 & -0.841471 \\ 0 & 0 \\ 1 & 0.84147098 \\ 2 & 0.90929743 \\ 3 & 0.14112001 \\ 4 & -0.7568025 \\ 5 & -0.9589243 \end{array}$$

The interpolation result is visualized in Fig. 1. The second input data point set contains the following 11 data points:

$$\begin{array}{ll} -5 & 0.00673795 \\ -4 & 0.01831564 \\ -3 & 0.04978707 \\ -2 & 0.13533528 \\ -1 & 0.36787944 \\ 0 & 1 \\ 1 & 2.71828183 \\ 2 & 7.3890561 \\ 3 & 20.0855369 \\ 4 & 54.59815 \\ 5 & 148.413159 \end{array}$$

The interpolation result is visualized in Fig. 2.

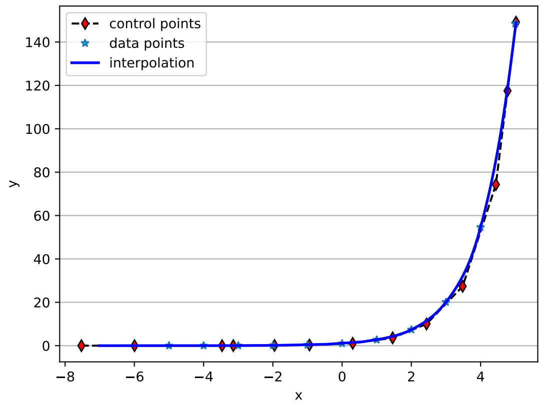

Fig. 2: Interpolation results for example 1. The green solid curve is the interpolation result. The circle points are the input data points. The red points are the corresponding control points.

IV. Task III——Discussion

In this section, we will discuss the following two questions: 1) what if the number of the input data points is less than 4; and (2) what if there are two points in the input data point set have the same coordinates.

A. *Insufficient Input Data Points*

In section, we discuss what will happen if the number of input data points is less than 4. In particular, we will illustrate the interpolated curve when the number of input data points is 2 and 3 respectively. To start with, we first investigate what will happen if the number of input data points is 3, and the coordinates of input data points are demonstrated in the following:

$$\begin{array}{cc} 0 & 0 \\ 0 & 2 \\ 2 & 2 \end{array}$$

Using these data points, the interpolated curve is illustrated in Fig. (**??**). It can be seen from Fig. (3) that all control points

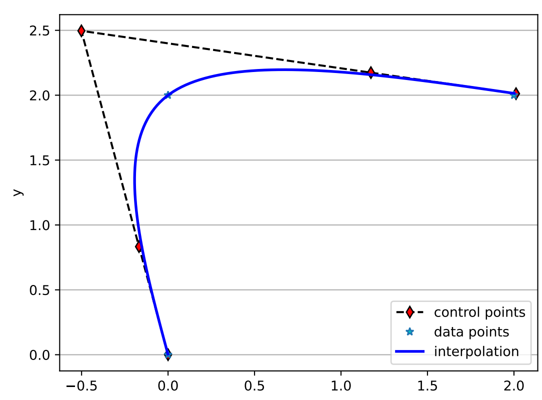

Fig. 3: Cubic B-spline interpolation with 3 input data points.

form the convex hall of the cubic B-spline interpolation curve. At such a situation, the cubic B-spline curve degrades to the Beizer curve and the change of any input data point will affect all control points simultaneously.

We further investigate what will happen if the number of input data points is 2, and the coordinates of two input data points are shown as follows:

$$\begin{matrix} 0 & 0 \\ 0 & 2 \end{matrix}$$

The cubic B-spline curve the two input data points are illustrated in Fig. (4). As we can see, the cubic B-spline curve now degrades to a line which interpolates the given two data points.

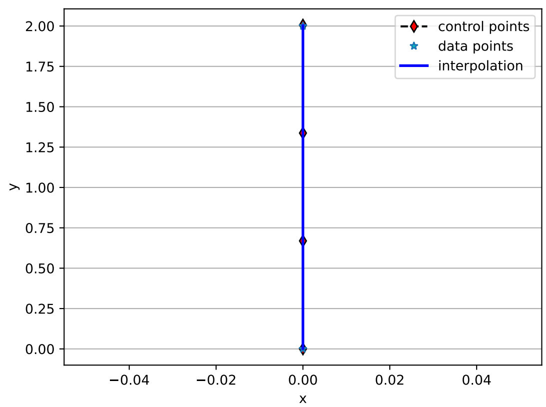

Fig. 4: Cubic B-spline interpolation with 2 input data points.

B. Repeated Input Data Points

In this section, we provide some discussion on the situation where there are two data points have the same coordinates in the input data points. We split this situation into two sub-cases. The first case is that there are two consecutive data points having the same coordinates, and the second case is that there are two data points have the same coordinates but they are not neighbors.

For the first case, we use the input data points as shown in the following:

$$\begin{matrix} -5 & 0.00673795 \\ -4 & 0.01831564 \\ -3 & 0.04978707 \\ -2 & 0.13533528 \\ -2 & 0.13533528 \\ 0 & 1 \\ 1 & 2.71828183 \\ 2 & 7.3890561 \\ 3 & 20.0855369 \\ 4 & 54.59815 \\ 5 & 148.413159 \end{matrix}$$

The cubic B-spline curve is illustrated in Fig. (5). It can be observed that the smoothness of the curve degrades at the vicinity around $(-2, 0.13)$, which is the coordinate of the 4-th and the 5-th input data point. It is because the same coordinates of two consecutive input data points will lead to the multiplicity in the knot vector, which will decrease the continuity of the cubic B-spline curve.

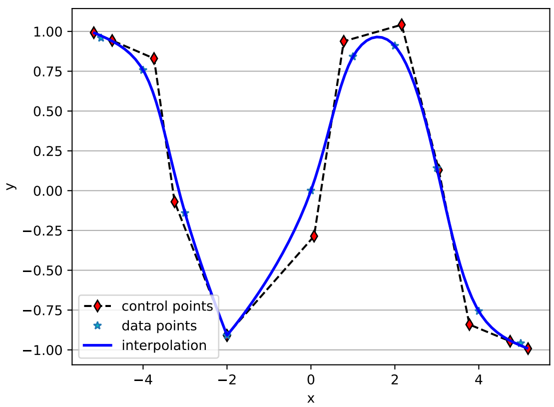

Fig. 5: Cubic B-spline interpolation with two consecutive input data points having the same coordinates.

For the second case, we use the following input data points:

-5	0.00673795
-4	0.01831564
-3	0.04978707
-2	0.13533528
-1	0.36787944
0	1
-2	0.13533528
2	7.3890561
3	20.0855369
4	54.59815
5	148.413159

where the 4-th and the 7-th data point have the same coordinate. In this case, the cubic B-spline curve is shown in Fig. (6). From Fig. (6), it can be observed that even two data points having the same coordinates, the continuity of the curve is

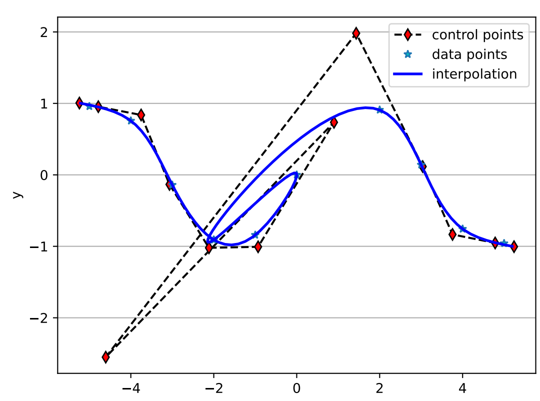

Fig. 6: Cubic B-spline interpolation with two nonconsecutive input data points having the same coordinates.

well-preserved because there is no multiplicity in the knot vector in such a situation. It is also important to note that even though the curve is very twisted, it still follows the basic property of the cubic B-spline curve: any point on the curve still lies in the convex hall of adjacent 4 control points.