CS340/MATH321 – Geometrical modelling and numerical analysis Group project 2

Huda Feroz Ahmed Aiman Khan Muhammad Shahrom Ali

Fall 2019

1. **(50 points)** The following question asks you to explore and implement the equation

$$\vec{p}(s,t) = \sum N_i(s,t)\vec{p}_i$$

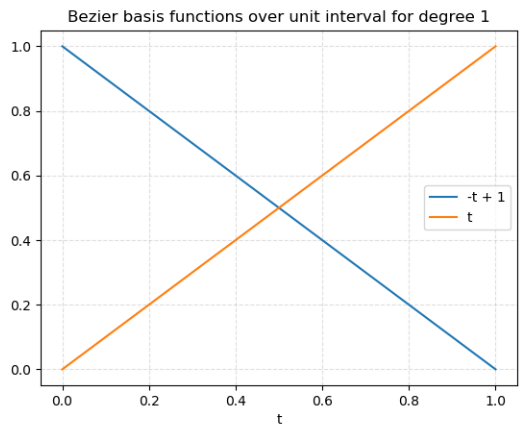
in two- and three-dimensions

a. **(3 points)** Draw all the linear and quadratic Bézier basis functions over a unit interval and a unit square

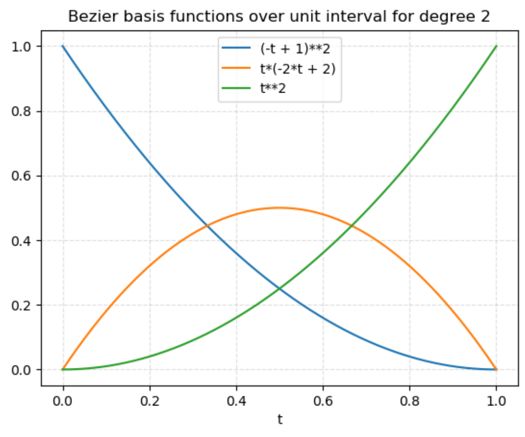
Solution

Figure 1 represents the linear and quadratic Bézier basis functions over a unit interval. The following code has been used to generate the figures 1 and 2. Use `type="curve"` to generate basis functions over a unit interval, and use `type="surface"` to generate basis functions over a unit square.

```python
plotBezierBasisFunctions(type='curve',degree=1)
plotBezierBasisFunctions(type='curve',degree=2)
plotBezierBasisFunctions(type='surface',degree=1)
plotBezierBasisFunctions(type='surface',degree=2)
```



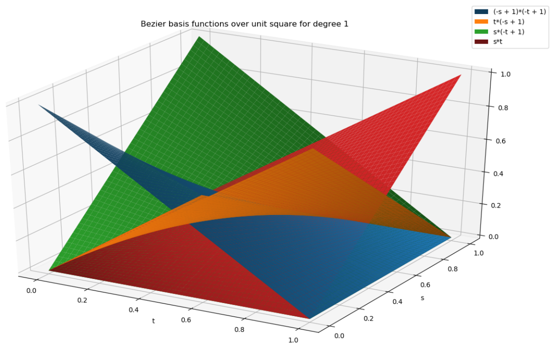
(a) Linear Bezier basis



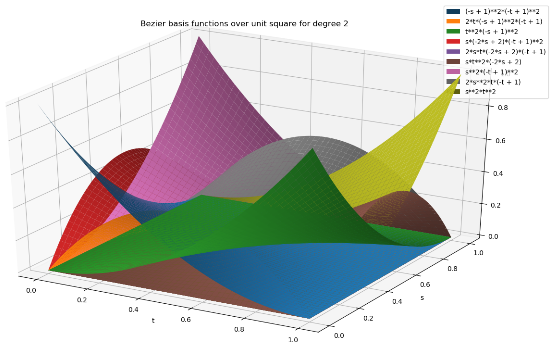
(b) Quadratic Bezier basis

Figure 1: Bezier Basis over Unit Interval

Figure 2 represents the linear and quadratic Bézier basis functions over a unit square:



(a) Linear Bezier basis



(b) Quadratic Bezier basis

Figure 2: Bezier Basis over Unit Square

b. **(3 points)** Draw a linear, quadratic, cubic, quartic b-splines basis functions in 1D over a uniform knot vector

Solution

Figure 3 represents the b-spline basis functions for uniform knot vector $[0, 1, 2, 3, 4, 5, 6]$ and the following code has been used to generate it.

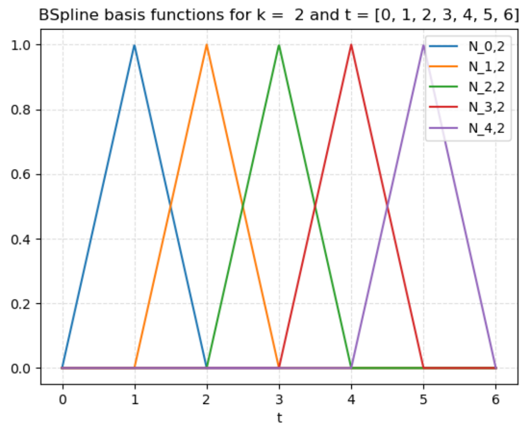
```
knot_vector = [0,1,2,3,4,5,6]
plotBsplineBasisFunctions(knot_vector,2) #linear
plotBsplineBasisFunctions(knot_vector,3) #Quadratic
plotBsplineBasisFunctions(knot_vector,4) #Cubic
plotBsplineBasisFunctions(knot_vector,5) #Quartic
```

c. **(3 points)** Draw quadratic, cubic, and quartic Bézier curves by arbitrary defining the appropriate control points

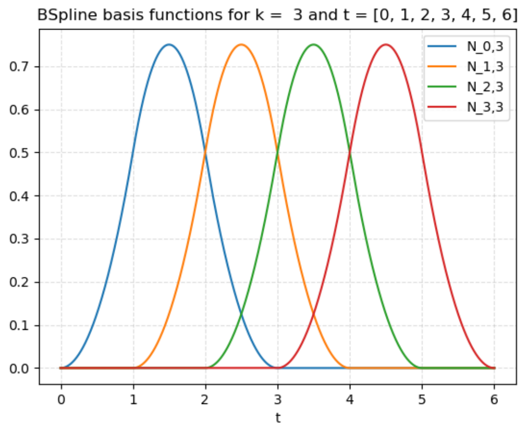
Solution

Figure 4 represents the different bezier curves formed and has been generated through the following code.

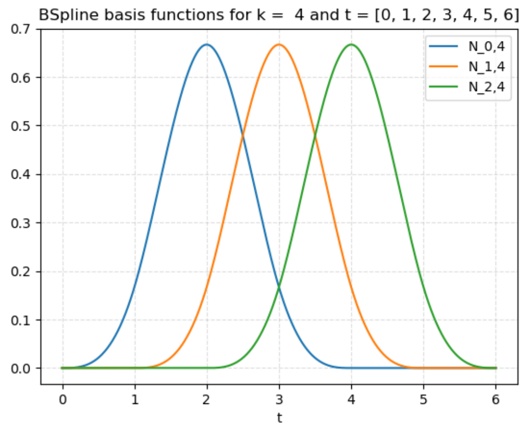
```
curveCP = {1:[[0,0],[0.5,1]],
           2: [[1,2],[0,0],[2,0]],
           3:[[0,0],[0.5,1],[0.5,0],[1,1]],
           4:[[0,0],[1,0],[1,1],[0,1],[0.5,2]]}

plotBezierCurves(curveCP[2],2) #Quadratic
plotBezierCurves(curveCP[3],3) #Cubic
plotBezierCurves(curveCP[4],4) #Quartic
```



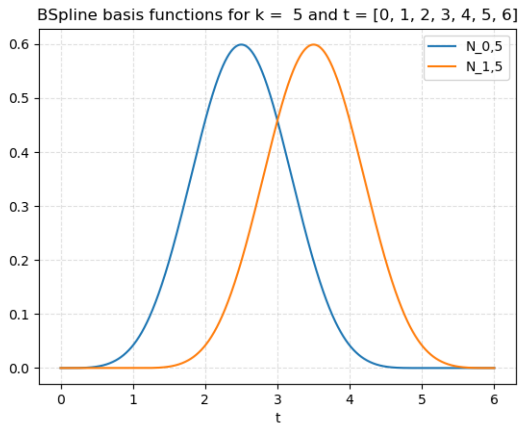
(a) Linear b-spline basis



(b) Quadratic b-spline basis

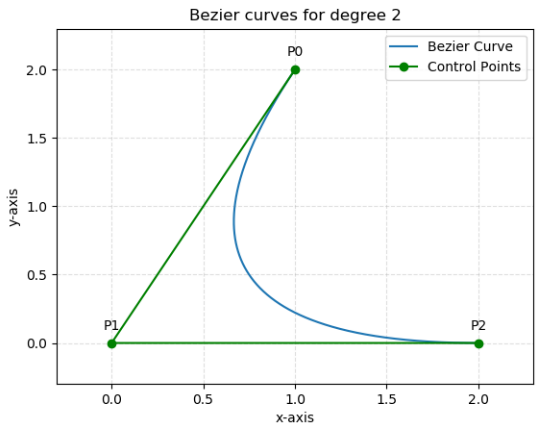


(c) Cubic b-spline basis

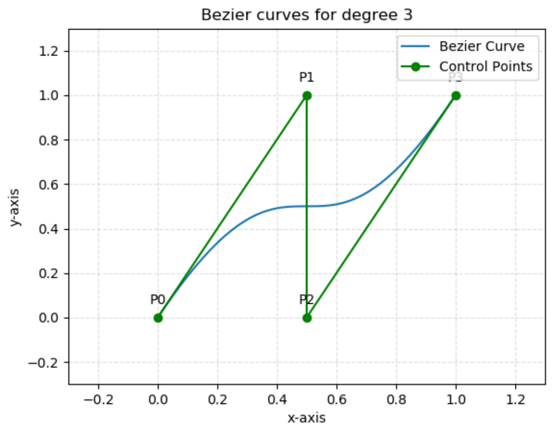


(d) Quartic b-spline basis

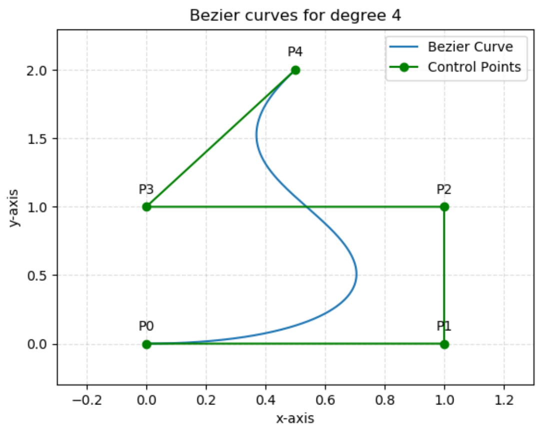
Figure 3: B-Spline Basis for knot vector $[0, 1, 2, 3, 4, 5, 6]$



(a) Quadratic Bezier Curve



(b) Cubic Bezier Curve



(c) Quartic Bezier Curve

Figure 4: Bezier Curves of different degrees

d. **(6 points)** The curve generated using Bézier basis functions do not usually pass through the control points. Derive new quadratic and cubic polynomial basis functions $N_i(t)$ which when linearly combined with the control points $\vec{p_i}$ will generate a curve which passes through the control points. Draw them and comment on potential advantages and disadvantages of these basis functions. You can define the control points arbitrarily.

Solution

The curve generated using Bézier basis functions only passes through the first and last control points because the first and last basis function is equal to 1 at some value of parametric coordinate t. For quadratic bezier curves $(1-t)^2 = 1$ at $t = 0$ and $t^2 = 1$ at $t = 1$. Similarly, for cubic bezier curves $(1-t)^3 = 1$ at $t = 0$ and $t^3 = 1$ at $t = 1$. Other basis functions, for quadratic and cubic both do not equal to 1 at any value of $t \in [0, 1]$. In order to make bezier curve pass through control points, we need to derive new basis which equal to 1 at some t.

New Quadratic Basis

For quadratic basis, lets say we want the basis function equal to 1 at $t = 0$, $t = 0.5$ and $t = 1$. We can have three terms t, $(1-t)$ and $(0.5-t)$.
We want the first basis function to be equal to 1 at $t = 0$, for this we have to choose $(1-t)(0.5-t)$. Similarly, we want the second basis function to be equal to 1 at $t = 0.5$, for this we have to choose $(1-t)(t)$. And similarly, we want the third basis function to be equal 1 at $t = 1$, for this we have to choose $(0.5-t)t$. Hence the new basis functions are:

$$(1-t)(0.5-t); (1-t)(t); (0.5-t)t \tag{1}$$

These basis functions are all quadratic in nature just like original basis functions for quadratic curves but they do not equal to 1 at desired t values. To make them equal to 1 we need to multiply them with some constants. Hence the new basis functions for quadratic bezier curves which pass through control points are given as:

$$2(1-t)(0.5-t); \quad 4(1-t)(t); \quad -2(0.5-t)t \tag{2}$$

New Cubic Basis

For cubic basis, we will do the exact same steps as we did for quadratic, but since now there are 4 terms, we want the basis function to be equal to 1 at $t = 0$, $t = \frac{1}{3}$, $t = \frac{2}{3}$ and $t = 1$. We can have 4 terms t, $(1-t)$, $(\dfrac{1}{3} - t)$ and $(\frac{2}{3} - t)$.
We want the first basis function to be equal to 1 at $t = 0$, for this we have to choose $(1-t)(\frac{1}{3}-t)(\frac{2}{3}-t)$. Similarly, we want the second basis function to be equal to 1 at $t = \frac{1}{3}$, for this we have to choose $(1-t)(t)(\frac{2}{3}-t)$. For third basis function to be equal 1 at $t = \frac{2}{3}$, we have to choose $(\frac{1}{3}-t)(t)(1-t)$. Finally, for last the basis function to be equal to 1 at $t = 1$, we have to choose $(t)(\frac{1}{3}-t)(\frac{2}{3}-t)$. These functions still do not equal to 1 at specified t values, hence we will add some constants. Hence the new basis functions are:

$$\frac{9}{2}(1-t)(\frac{1}{3}-t)(\frac{2}{3}-t); \quad \frac{27}{2}(1-t)(t)(\frac{2}{3}-t); \quad \frac{-27}{2}(1-t)(t)((\frac{1}{3}-t); \quad \frac{9}{2}(t)(\frac{1}{3}-t)(\frac{2}{3}-t) \tag{3}$$

Advantages

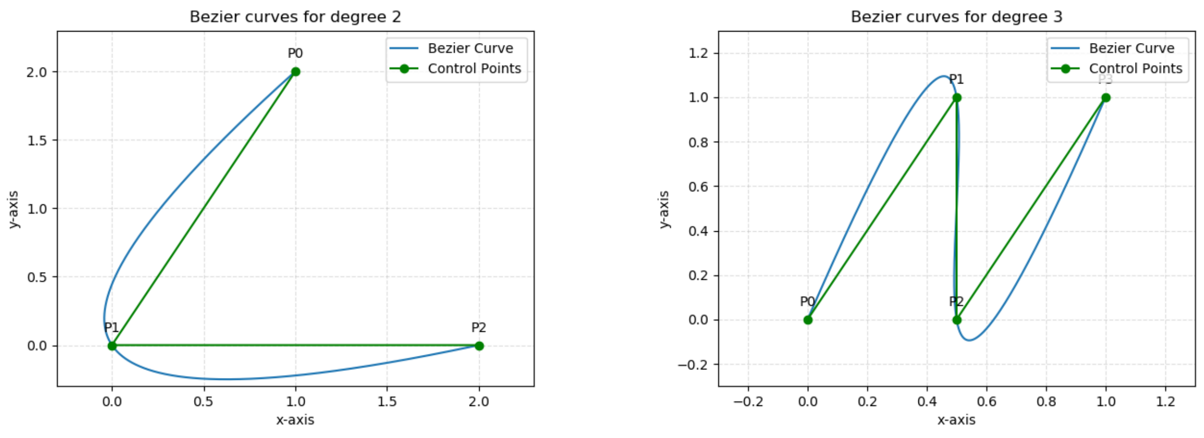
An advantage of such a basis is that we can still reach a certain control point by substituting different values of t. For example, for new quadratic basis, at $t = 0$, we are at point P_0. Similarly at $t = 0.5$, we are at point P_1. And same goes for cubic basis.

Disadvantages

A major disadvantage of these new basis functions is that that they are not an affine combination anymore. As we can see in the graphs that the curved spanned by these control points now goes outside the control polygon.

Figure 5 represents the bezier curves passing through control points. and has been generated using the following code.

```
curveCP = {1:[[0,0],[0.5,1]],
           2:  [[1,2],[0,0],[2,0]],
           3:[[0,0],[0.5,1],[0.5,0],[1,1]],
           4:[[0,0],[1,0],[1,1],[0,1],[0.5,2]]}

newBezierBasisToPassThroughCP(curveCP[2],2) #Quadratic
newBezierBasisToPassThroughCP(curveCP[3],3) #Cubic
```



(a) Quadratic Bezier Curve

(b) Cubic Bezier Curve

Figure 5: Bezier Curves passing through control points.

e. **(6 points)** Now imagine you have a cubic Bézier curves $\vec{P}(t)$ and $\vec{Q}(t)$, derive the conditions on the control points such that the curves $\vec{P}(t)$ join the curve $\vec{Q}(t)$ with C^1 and with C^2 continuity. What is the continuity of the cubic Bézier curve $\vec{P}(t)$ by itself.

Solution

The equations for cubic bezier curves $P(t)$ and $Q(t)$ can be written as:

$$P(t) = (t-1)^3 P_0 + 3(t-1)^2 t P_1 + 3(t-1)t^2 P_2 + t^3 P_3 \tag{4}$$

$$Q(t) = (t-1)^3 Q_0 + 3(t-1)^2 t Q_1 + 3(t-1)t^2 Q_2 + t^3 Q_3 \tag{5}$$

To ensure, C^0 continuity, equation 6 should satisfy

$$P(1) = Q(0) \tag{6}$$

Following from equation 6 we have:

$$P_3 = Q_0 \tag{7}$$

To ensure, C^1 continuity, equation 8 should satisfy

$$P'(1) = Q'(0) \tag{8}$$

Following from equation 8 we have:

$$-3P_2 + 3P_3 = -3Q_0 + 3Q_1 \tag{9}$$

By simplifying equation 9 and substituting the value of Q_0 from equation 7 we get the conditions on the control points such that the curve $\vec{P}(t)$ joins the curve $\vec{Q}(t)$ with C^1 continuity:

$$\boxed{2P_3 - P_2 = Q_1} \tag{10}$$

To ensure, C^2 continuity, equation 11 should satisfy

$$P''(1) = Q''(0) \tag{11}$$

Following from equation 11 we have:

$$6P_1 - 12P_2 + 6P_3 = 6Q_0 - 12Q_1 + 6Q_2 \tag{12}$$

By simplifying equation 12 and substituting the value of Q_0 from equation 7 we get the conditions on the control points such that the curve $\vec{P}(t)$ joins the curve $\vec{Q}(t)$ with C^2 continuity:

$$P_1 - 2P_2 = -2Q_1 + Q_2 \tag{13}$$

Using the constraints for C^1 continuity, we can substitute the value of Q_1 from equation 10, to obtain the simplified condition for C^2 continuity:

$$\boxed{P_1 - 4P_2 + 4P_3 = Q_2} \tag{14}$$

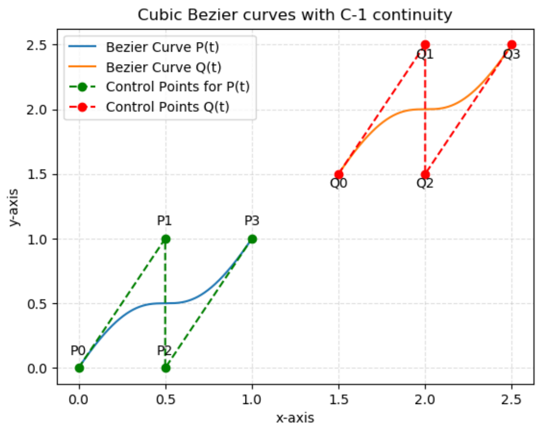
The continuity of the cubic Bézier curve $\vec{P}(t)$ by itself is C^n $\forall n$ because any bezier curve is infinitely differentiable within itself, and is therefore continuous to any degree (C^n-continuous, $\forall n$) [1].

f. **(9 points)** For part e), draw a figure which demonstrates that the curve $\vec{P}(t)$ joins the curve $\vec{Q}(t)$ with C^1 continuity. Draw a similar figure(s) which demonstrates that the curves join with C^2 continuity. The figure must be generated by some algorithm/function.

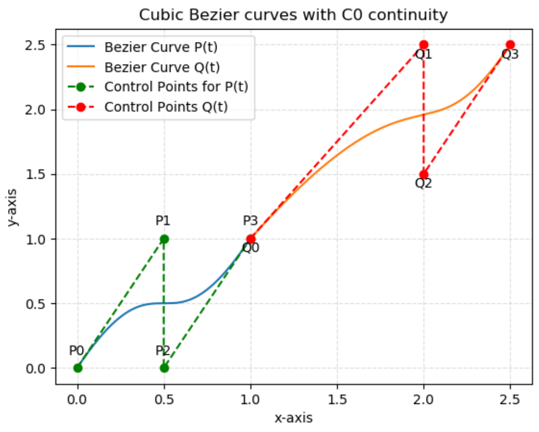
Solution

Using the conditions obtained in part e, we can draw the curves which join with desired continuity. Figure 6 represents the curves $P(t)$ and $Q(t)$ joined by different continuities and has been generated using the following code.

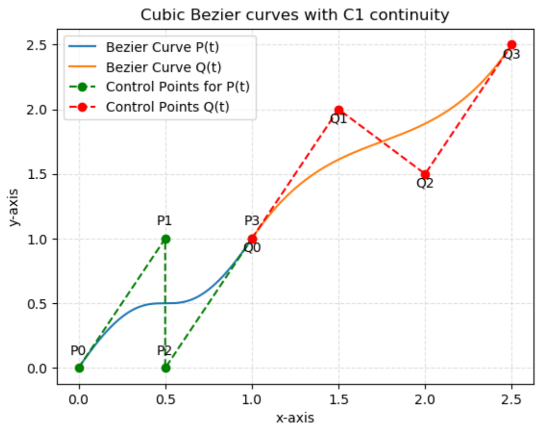
```
cpP = [[0,0],[0.5,1],[0.5,0],[1,1] #control points for P
cpQ = [[1.5,1.5],[2,2.5],[2,1.5],[2.5,2.5]] #control points for Q
cubicBezierCurvesJoinedWithContinuity(continuity='C-1',cpP=cpP,cpQ=cpQ)
cubicBezierCurvesJoinedWithContinuity(continuity='C0',cpP=cpP,cpQ=cpQ)
cubicBezierCurvesJoinedWithContinuity(continuity='C1',cpP=cpP,cpQ=cpQ)
cubicBezierCurvesJoinedWithContinuity(continuity='C2',cpP=cpP,cpQ=cpQ)
```



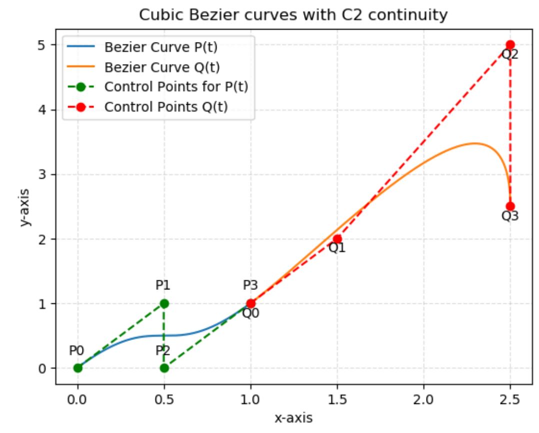
(a) Curves joined by C^{-1} continuity



(b) Curves joined by C^0 continuity



(c) Curves joined by C^1 continuity



(d) Curves joined by C^2 continuity

Figure 6: Cubic Bezier Curves $P(t)$ and $Q(t)$ joined by different continuities.

g. **(9 points)** Repeat parts c-d for a single patch with tensor product structure

Solution

Figures 7, 8,9 represents the different bezier patches formed and has been generated through the following code.

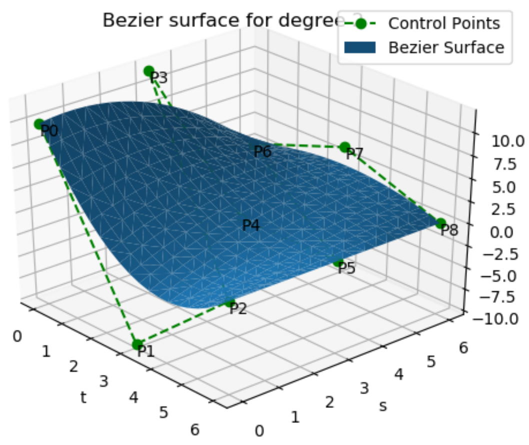
```
patchCP = {2 :  [[0,0,10],[3,0,-10],[6,0,0],[0,3,12],
                  [3,3,0],[6,3,0],[0,6,0],[3,6,4],[6,6,0]],
           3 :  [[0,0,0],  [1,0,2],  [2,0,-1],[3,0,0],
                 [0,1,0],  [1,1,4],  [2,1,4],  [3,1,0],
                 [0,2,0],  [1,2,8],  [2,2,8],   [3,2,0],
                 [0,3,0],  [1,3,0],  [2,3,0],  [3,3,0]],
           4 :  [[0,0,0],  [1,0,2],  [2,0,-1],[3,0,0],
                 [0,1,0],  [1,1,4],  [2,1,4],  [3,1,4],  [3,2,4],  [3,1,0],
                 [0,2,0],  [1,2,8],  [2,2,8],  [2,4,8],  [4,2,8],   [3,2,0],
                 [0,3,0],  [1,3,0],  [2,3,0],  [3,3,0],  [3,2,1],
                 [0,3,1],  [1,3,1],  [2,3,1],  [3,3,1]
           ]}
plotBezierSurfaces(patchCP[2],2) #Quadratic
newBezierSurfacesThroughCP(patchCP[2],2) #Quadratic Through CP
plotBezierSurfaces(patchCP[3],3) #Cubic
newBezierSurfacesThroughCP(patchCP[3],3) #Cubic Through CP
plotBezierSurfaces(patchCP[4],4) #Quartic
```

The new basis for quadratic and cubic bezier patches are given in code file, and has been derived by just replacing the following in the original basis for both s and t. For Quadratic, replace:

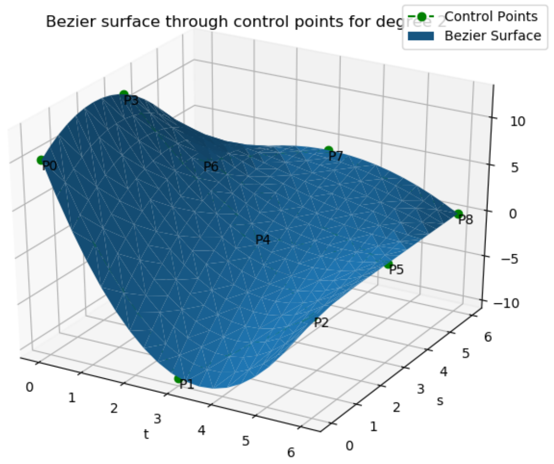
$$\begin{aligned}(1-t)^2 &\longrightarrow 2(1-t)(0.5-t)\\ 2t(1-t) &\longrightarrow 4(1-t)(t)\\ t^2 &\longrightarrow -2(0.5-t)t\end{aligned}$$

For cubic, replace (for both s and t):

$$\begin{aligned}(1-t)^3 &\longrightarrow \frac{9}{2}(1-t)(\frac{1}{3}-t)(\frac{2}{3}-t)\\ 3t(1-t)^2 &\longrightarrow \frac{27}{2}(1-t)(t)(\frac{2}{3}-t)\\ 3t^2(1-t) &\longrightarrow \frac{-27}{2}(1-t)(t)((\frac{1}{3}-t)\\ t^3 &\longrightarrow \frac{9}{2}(t)(\frac{1}{3}-t)(\frac{2}{3}-t)\end{aligned}$$

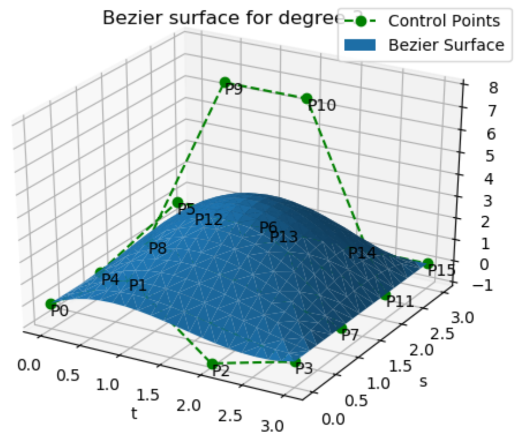


(a) Quadratic Bezier Patch

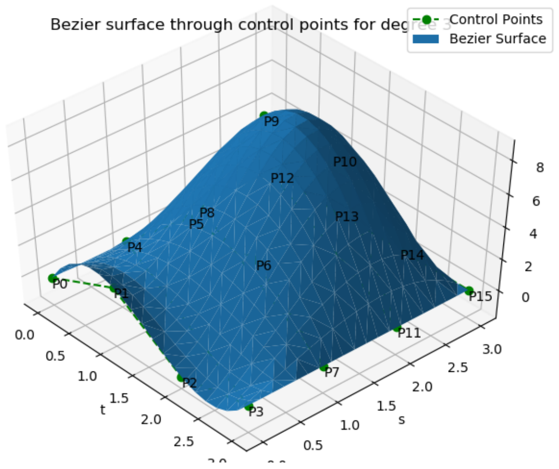


(b) Quadratic Bezier patch passing through control points

Figure 7: Quadratic Bezier Patches



(a) Cubic Bezier Patch



(b) Cubic Bezier patch passing through control points

Figure 8: Cubic Bezier Patches

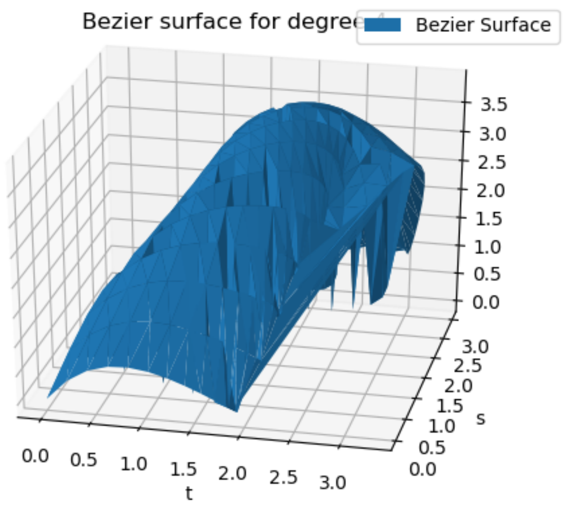


Figure 9: Quartic Bezier Path

h. **(2 points)** Consider two arbitrary tensor product quadratic Bézier patches $\vec{P}(s,t)$ and $\vec{Q}(s,t)$, describe (qualitatively or mathematically) the conditions for C^1 continuity along the edge at which they meet?

Solution

Assumption: Let we are limiting our problem such that the patches meet at one boundary.

We have taken the points in the following order, assuming t on the vertical axis, and s on the horizontal axis.

$$\begin{gathered} P_{2,0}, P_{2,1}, P_{2,2} \\ P_{1,0}, P_{1,1}, P_{1,2} \\ P_{0,0}, P_{0,1}, P_{0,2} \end{gathered}$$

The same alignment is assumed for control points of patch $Q(s,t)$.

The equations for quadratic bezier patches $P(s,t)$ and $Q(s,t)$ can be written as:

$$\begin{aligned} P(s,t) = (1-s)^2[(1-t)^2P_{0,0} + 2t(1-t)P_{0,1} + t^2P_{0,2}] + \\ 2s(1-s)[(1-t)^2P_{1,0} + 2t(1-t)P_{1,1} + t^2P_{1,2}] + \\ s^2[(1-t)^2P_{2,0} + 2t(1-t)P_{2,1} + t^2P_{2,2}] \end{aligned} \tag{15}$$

$$\begin{aligned} Q(s,t) = (1-s)^2[(1-t)^2Q_{0,0} + 2t(1-t)Q_{0,1} + t^2Q_{0,2}] + \\ 2s(1-s)[(1-t)^2Q_{1,0} + 2t(1-t)Q_{1,1} + t^2Q_{1,2}] + \\ s^2[(1-t)^2Q_{2,0} + 2t(1-t)Q_{2,1} + t^2Q_{2,2}] \end{aligned} \tag{16}$$

To ensure C^0 continuity, both the curves should meet at boundary, they should share control points at the edge. Boundary of a bezier patch is obtained when either of the parametric coordinate equals to 1 or 0, while the other coordinate ranges from 0 to 1. Since we are dealing with just one boundary, we can fix either of the parametric coordinates to a specific value (lets say $t = 1$ for $P(s, t)$ and $t = 0$ for $Q(s, t)$). Hence for C^0 continuity, equation 17 should satisfy.
$\forall s$ where $s \in \{0, 1\}$

$$P(s, 1) = Q(s, 0) \tag{17}$$

Following from equation 17 we have:

$$\begin{aligned} P_{2,0} &= Q_{0,0} \\ P_{2,1} &= Q_{0,1} \\ P_{2,2} &= Q_{0,2} \end{aligned} \tag{18}$$

To ensure C^1 continuity, equation 19 should satisfy $\forall s$ where $s \in \{0, 1\}$. This is because we have fixef t for both the curves, now we just need to ensure tangential continuity along s axis.

$$P'(s, 1) = Q'(s, 0) \tag{19}$$

By computing the derivatives of $P'(s, 1)$ and $Q'(s, 0)$, we can write 19 as:

$$-2(1-s)P_{0,2} + 2(1-2s)P_{1,2} + 2sP_{2,2} = -2(1-s)Q_{0,0} + 2(1-2s)Q_{1,0} + 2sQ_{2,0} \tag{20}$$

For C^1 continuity, we need to ensure that these derivatives for $P(s, t)$ at $s = 1$ match with derivatives for $Q(s, t)$ at $s = 0$. Substituting the value of s accordingly in equation 20, we get:

$$-2P_{1,2} + 2P_{2,2} = -2Q_{0,0} + 2Q_{1,0} \tag{21}$$

Using the value of $Q_{0,0}$ from equation 18, we get:

$$-2P_{1,2} + 2P_{2,2} + 2P_{2,0} = 2Q_{1,0} \tag{22}$$

Dividing equation 22 by 2, we get the final conditions for C^1 continuity.

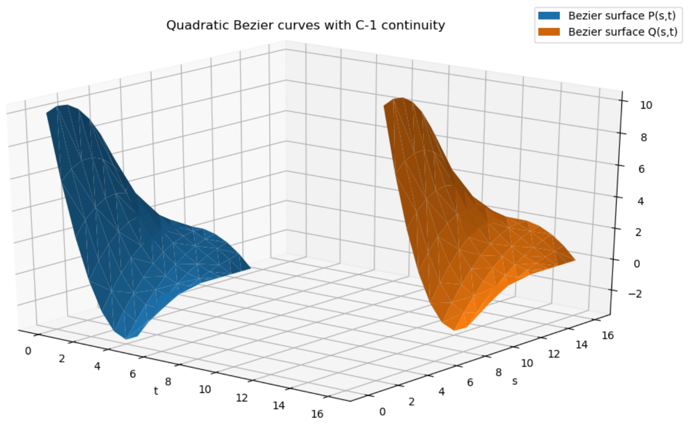
$$\boxed{Q_{1,0} = P_{2,2} + P_{2,0} - P_{1,2}} \tag{23}$$

i. **(9 points)** Define arbitrary control points for patches $\vec{P}(s, t)$ and $\vec{Q}(s, t)$ and generate a figure which demonstrates whether the patches connects with C^1 continuity. The figure must be generated by some algorithm/function.

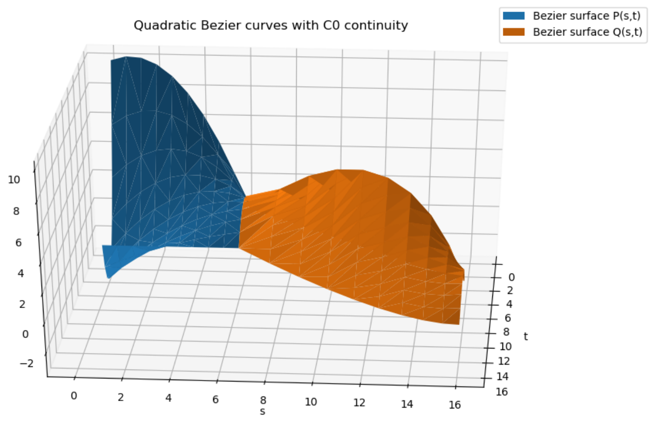
Solution

Using the conditions derived in part h, we can draw the bezier patches joined with C^1 continuity. 10(c) shows that tangents in the combined curve are continuous. The figures have been generated using the following code:

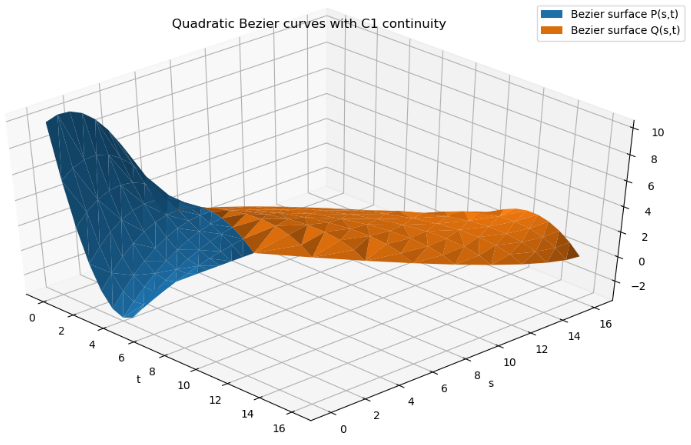
```
cpP  =  [[0,0,10],[3,0,-10],[6,0,0],[0,3,12],
        [3,3,0],[6,3,0],[0,6,0],[3,6,4],[6,6,0]] #control points for P
cpQ  =  [[10,10,10],[13,10,-10],[16,10,0],[10,13,12],[13,13,0],
        [16,13,0],[10,16,0],[13,16,4],[16,16,0]] #control points for Q
quadraticBezierPatchesJoinedWithContinuity(continuity='C-1',cpP=cpP,cpQ=cpQ)
quadraticBezierPatchesJoinedWithContinuity(continuity='C0',cpP=cpP,cpQ=cpQ)
quadraticBezierPatchesJoinedWithContinuity(continuity='C1',cpP=cpP,cpQ=cpQ)
```



(a) Patches joined by C^{-1} continuity



(b) Patches joined by C^{0} continuity



(c) Patches joined by C^{1} continuity

Figure 10: Quadratic Bezier Patches $P(s,t)$ and $Q(s,t)$ joined by different continuities.

2. **(10 points)** This question asks you to generate a curve defined by the control points using the quadratic Bézier basis functions. You are given a circle.txt file. It has 16 vertices and 8 faces (line elements). Describe succinctly what you have done

   Each face (element) of the mesh comprises of three vertices. For example circle.txt file shows that the second face comprises of vertices {2, 3, 4}. Using the parametrisation show in figure .
   Solution

   The first line of the text file represents the number of vertices and faces. Based on that we load up the vertices as `Point` objects into a dictionary and then faces into a dictionary, as `Face` objects which contain those `Point` objects. The curve to be generated is to be defined using the quadratic basis therefore the Face class contains those basis functions as $b0$, $b1$, and $b2$.

   Each face contains 3 points, these three points are used to draw a quadratic bezier curve using the quadratic bezier basis function.

   After loading up all faces, for each face we calculate its underlying curve $F_i(t)$ where i represents the curve corresponding to face F_i. $\boldsymbol{F_{ij}}$ represents the j^{th} vertex of face F_i. The parametric coordinate t is interpolated to produce a smooth curve. The interpolation is done by increasing t from 0 to 1 in steps of 0.0001. We calculate the Bezier curve for each face by,

$$F_i(t) = (1-t)^2\boldsymbol{F_{i0}} + 2(1-t)(t)\boldsymbol{F_{i1}} + t^2\boldsymbol{F_{i2}} \tag{24}$$

   The entire curve as shown in figure 11 is generated by combining the individual curves for all faces.

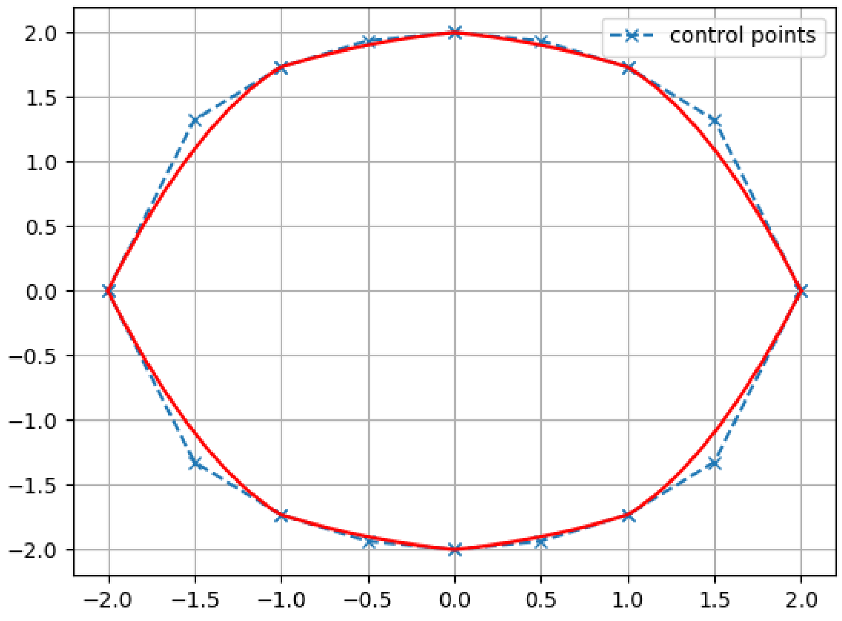


Figure 11: Figure Generated using the control points and quadratic Bezier basis functions

3. **(20 points)** You are now asked to generalise your algorithm in question 2 from curves to surfaces. Given the mesh for the one-eighth of a sphere in sphere.txt. The screenshot of the mesh is shown

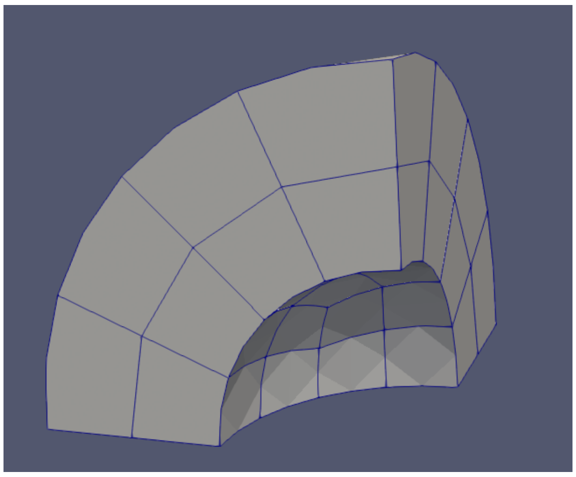


Figure 12: Mesh for the one-eighth of a sphere

in Figure 12 The 'sphere.txt' file has 224 vertices and 60 quadrilateral faces/elements. Each quadrilateral face/element comprises of 9 points. The quadrilateral in the sphere.txt numbers the vertices. Generate the surface represented by the mesh using the quadratic Bézier basis functions.

Solution

Each face contains 9 points, these nine points are used to draw a quadratic bezier patch using the quadratic bezier basis function for surfaces.

After loading up all faces, for each face we calculate its underlying curve $F_i(s,t)$ where i represents the patch corresponding to face F_i. We calculate the Bezier patch for each face by,

$$\begin{aligned} F_i(s,t) = (1-s)^2[(1-t)^2F_{0,0} + 2t(1-t)F_{0,1} + t^2F_{0,2}]+ \\ 2s(1-s)[(1-t)^2F_{1,0} + 2t(1-t)F_{1,1} + t^2F_{1,2}]+ \\ s^2[(1-t)^2F_{2,0} + 2t(1-t)F_{2,1} + t^2F_{2,2}] \end{aligned} \tag{25}$$

The entire patch as shown in figure 13 is generated by combining the individual patches for all faces.

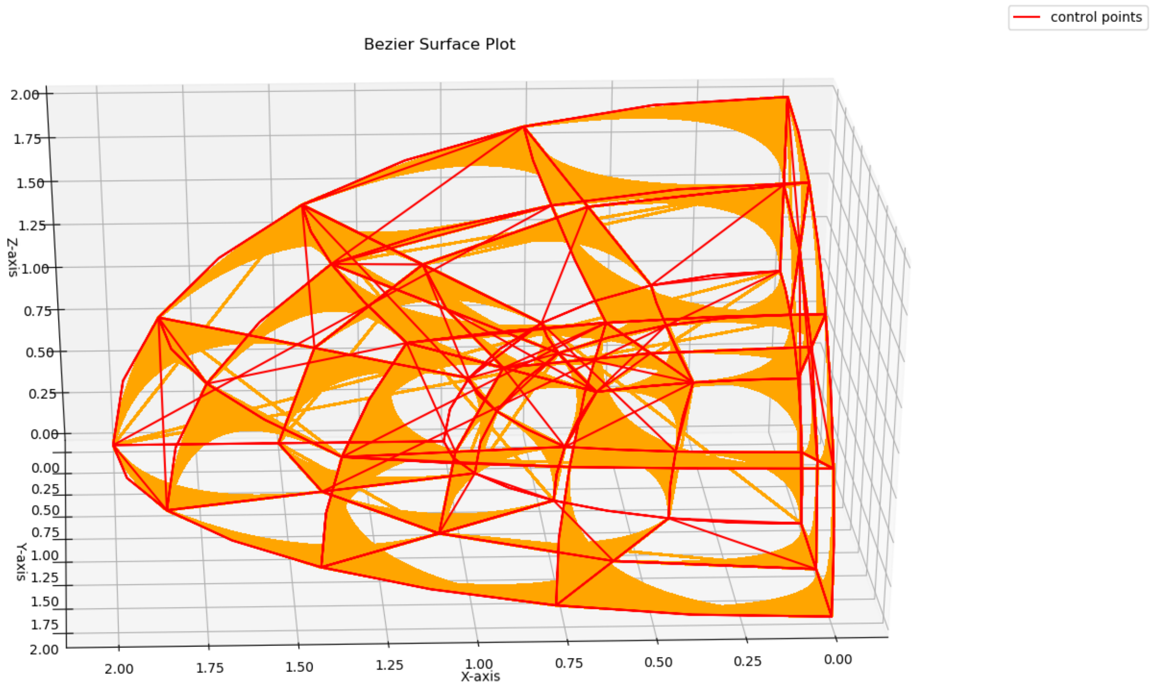


Figure 13: $\frac{1}{8}$ Sphere generated using quadratic bezier patches

References

[1] “Bezier curves.” [Online].
Available: https://www.cl.cam.ac.uk/teaching/2000/AGraphHCI/SMEG/node3.html. [Accessed: 25-Oct-2019].