Konkuk University Dept. of Physics

Numerical Analysis Assignment

2019 Spring Semester

Youngwan Kim

May 6, 2019

Assignment. *Implement a* Matlab *code in order to plot a Bezier curve using the second method introduced during the class.*

In total there are 3 Matlab script files. The main routine for the code is implemented in `bezier_hw.m`, and the other Matlab scripts are implementations of functions including `Mid(P,n)` and `Bzrt(n)`. Before introducing the main routine, let us go through the functions first.

Listing 1: 'Mid.m'

```
function Mid=Mid(P)

Mid = [  P(1,:);
        (P(1,:)+P(2,:))/2 ;
        (P(1,:)+2*P(2,:)+P(3,:))/4 ;
        (P(1,:)+3*P(2,:)+3*P(3,:)+P(4,:))/8 ;
        (P(1,:)+3*P(2,:)+3*P(3,:)+P(4,:))/8 ;
        (P(2,:)+2*P(3,:)+P(4,:))/4 ;
        (P(3,:)+P(4,:))/2 ;
         P(4,:) ];
```

The above `Mid(P)` function gets a 4×2 matrix as an input, where each of the `P(i,:)` of `P` are a single control point. Thus `Mid(P)` gets 4 control points of a Bezier curve as an input, and it gives us an output of a `8×2` matrix, or 8 points. These 8 points will act as new control points for the half-segments of the initial Bezier curve given. To elaborate, `Mid(P)(1:4,1:2)` will be the control points for $B(\frac{t}{2})$ and `Mid(P)(5:8,1:2)` for $B(\frac{1+t}{2})$ where $B(t)$ is the Bezier curve defined by `P` and for $t \in [0, 1]$.

Here I deliberately duplicated the midpoint in order to ease indexing problems later. (This might have caused some unnecessary computations)

Listing 2: 'Bzrt.m'

```
function Bzrt(P,n)

for i=1:n
 for j=1:2^(i-1)
  Q(8*j-7:8*j,1:2) = Mid(P(4*j-3:4*j,1:2));
 end
P=Q;
end

plot(P(:,1),P(:,2),'o-')
```

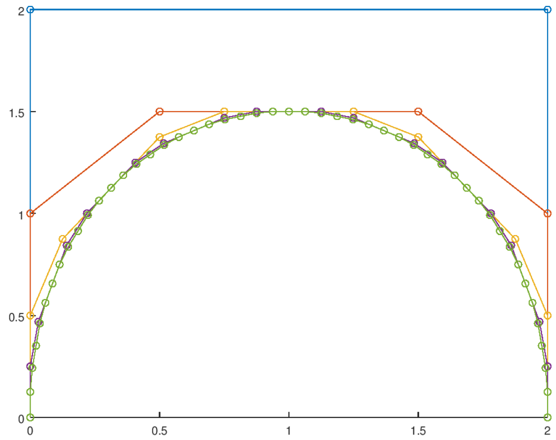
The above function `Bzrt(P,n)` has 2 inputs, where `P` again is the control points of the Bezier curve. Also `n` is an input parameter which will act as the desired accuracy for approximation, or the desired number of iterations. The relation used in the code is a generalization of a recursion I discovered while writing this code :

```
i=1 1
P(1:8,1:2)=Mid(P(1:4,1:2))

i=2 2 2^1
P(1:8,1:2)=Mid(P(1:4,1:2))
P(9:16,1:2)=Mid(P(5:8,1:2))

i=3 4 2^2
P(1:8,1:2)=Mid(P(1:4,1:2))
P(9:16,1:2)=Mid(P(5:8,1:2))
P(17:24,1:2)=Mid(P(9:12,1:2))
P(25:32,1:2)=Mid(P(13:16,1:2))
```

and so on. After getting the points the code plots the final points. The code itself requires a double for loop in order to obtain $2^{(n+2)}$ points to approximate the Bezier curve. The following is a result of a test script to see if the code above works well :



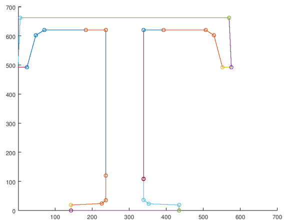
The control points used here for testing the code was $(0, 0)$, $(0, 2)$, $(2, 2)$ and $(2, 0)$. Also starting from the outside the control parameter was set as $n = 0$ to $n = 4$. We can obviously see that as n gets bigger the more closer it gets to the desired Bezier curve.

Listing 3: 'bezier_hw.m'

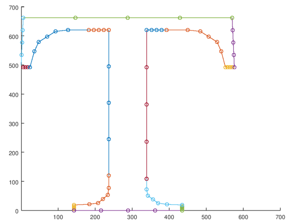
```
1 function bezier_hw(n)
2 figure; hold on
3 axis([1 700 0 700])
4 load T.txt
5
6 for i=1:size(T,1)
7  for j=1:4
8   P(j,:)=T(i,:)(2*j-1:2*j);
9  end
10 Bzrt(P,n);
11 end
```

Now we take a look at the main routine code `bezier_hw.m`, which is also a function as it gets an input `n`, which is a parameter for approximation accuracy. This code loads a `16`×`8` matrix by loading `T.txt`. Each `T(i,:)` has 8 coordinate information of the 4 control points for a single Bezier curve, in total having 16 Bezier curves to express T. The first routine goes through these 16 Bezier curves. The inner routine (lines 7 to 9) is a simple code converting a `1`×`8` matrix into a `4`×`2` matrix in order to fit it into `Mid(P)` and `Bzrt(P,n)`. After reshaping `T(i,:)` into `P`, for each `T(i,:)` it evaluates and plots the $2^{(n+2)}$ points of approximation using the `Bzrt(P,n)` function. The `figure; hold on` code lets the 16 set of line segments plotted in a single figure.

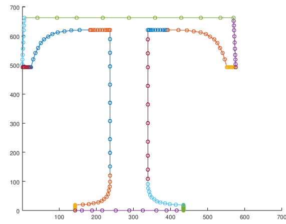
Here is the final result of approximating the Bezier curve data of `T.txt` into line segments :



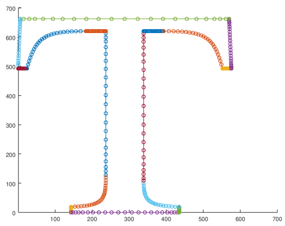
(a) $n = 0$



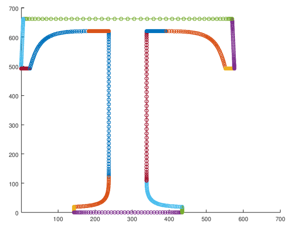
(b) $n = 1$



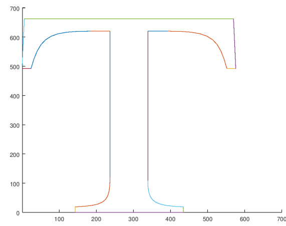
(c) $n = 2$



(d) $n = 3$



(e) $n = 4$



(f) real Bezier curve