Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 3

Bezier Curves

Octave Project 3

Description of the problem:

The mathematical problem we are trying to solve with these methods is interpolating 2D or 3D curves using a set of control points. Bezier curves are defined as a linear combination of Bernstein polynomials as follows. At the end we are using those control points to create the Bernstein polynomials that define the curve. The points P_i for i = 0, 1, …, n are the control points of the curve.

$$\gamma(t) = \sum_{i=0}^{n} P_i B_i^n(t), \quad t \in [0,1]$$

Explanation of the method:

Direct Evaluation:

Given points $P_0, P_1, P_2, \dots, P_n \; in \; R^2 \; or \; R^3$ we will compute the Bernstein polynomials $B_1^n(t), B_2^n(t), B_3^n(t), \dots, B_n^n(t)$ where is $n = 0,1,2,3, \dots, n$, we will have a mesh of t values in $[0,1]$. Finally compute the Bezier curve $\gamma(t) = \sum_{i=0}^{n} P_i B_i^n(t)$. Normally we will compute the curve first and then compute for each of the t values. In the code we first compute the values of the Bernstein polynomials by plugging the t values for each one of them.

```
for i = 0:n
  B = [B;nchoosek(n, i)*power((1 - t),n - i).*power(t,i)];
endfor
```

Figure 1

In *Figure 1* we can see the computation of each of the Bernstein polynomials, in the code the t represents an array of values from $[0,1]$.

```
for i = 1:n+1
  yx(1,:) += _PX(i)*B(i,:);
  yy(1,:) += _PY(i)*B(i,:);
  if(_dimension == 3)
    yz(1,:) += _PZ(i)*B(i,:);
  endif
endfor
```

Figure 2

In *Figure 2* we are computing $\sum_{i=0}^{n} P_i B_i^n(t)$.

Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 3

De Casteljau:

Given P_0, P_1 … P_n points in R^2 or R^3 the Bezier curve $\gamma(t) = \sum_{i=0}^{n} P_i B_i^n(t), \ \ t \in [0,1]$ can be computed constructing a regular mesh of m + 1 nodes in [0, 1] and then applying linear interpolation recursively on for each node. The recursion is noted as follows:

$$P_i^k(t_j) \ = \ t_j P_{i+1}^{k-1} + (1 \ - \ t_j) P_i^{k-1}$$

$$\begin{array}{ccccccc}
\boldsymbol{P_0} & & & & & & \\
 & \boldsymbol{P_0^1} & & & & & \\
\boldsymbol{P_1} & & \boldsymbol{P_0^2} & & & & \\
 & \boldsymbol{P_1^1} & & \boldsymbol{P_0^3} & & & \\
\boldsymbol{P_2} & & \vdots & \vdots & \ddots & & \\
 & \boldsymbol{P_2^1} & & \vdots & \cdots & \boldsymbol{P_0^n} & \\
\boldsymbol{P_3} & & \vdots & \vdots & \iddots & & \\
\vdots & \vdots & & \boldsymbol{P_{n-3}^3} & & & \\
\boldsymbol{P_{n-1}} & & \boldsymbol{P_{n-2}^2} & & & & \\
 & \boldsymbol{P_{n-1}^1} & & & & & \\
\boldsymbol{P_n} & & & & & &
\end{array}$$

```
% All the iterations of the algorithm
for j = 1 : n - 1
  for k = 1: n - j
    X(k, j + 1) = t(i) * X(k + 1, j) + (1 - t(i)) * X(k, j);
    Y(k, j + 1) = t(i) * Y(k + 1, j) + (1 - t(i)) * Y(k, j);
    if(dimension == 3)
      Z(k, j + 1) = t(i) * Z(k + 1, j) + (1 - t(i)) * Z(k, j);
    endif
```

Figure 3

In *Figure 3* we can see the computation of each node.

To plot the curve, we only need to store the points given by the last iteration of the method with all the values $t \in [0,1]$.

```
% Check that we are in the last iteration of the method
% which is the actual point from the curve and store it
% in the final curve.
if(n - j == 1)
  curveX(i) = X(k, j + 1);
  curveY(i) = Y(k, j + 1);
  if(dimension == 3)
    curveZ(i) = Z(k, j + 1);
  endif
endif
```

Figure 4

In *Figure 4* we can see how the points created by the last iteration of the method are stored into the curve's points.

Finally, to create the shell we have to store the control points of the current iteration for later plot.

```
    % If we are on the iteration of the desired shell,
    % store the control points for later plot
    if((i - 1) == shellindex)
      controlPointsX = X;
      controlPointsY = Y;
      if(dimension == 3)
        controlPointsZ = Z;
      endif
    endif
  endfor
```

Figure 5

Midpoint:

Given P_0, P_1 … P_n points in R^2 or R^3 the Bezier curve $\gamma(t) = \sum_{i=0}^{n} P_i B_i^n(t),\ \ t \in [0,1]$ can be split into two different curves, $\gamma_1(t)$ starting at P_0 and ending at $\gamma(t_s)$ where $t_s = \frac{1}{2}$, hence the name of midpoint subdivision, and $\gamma_2(t)$ starting at $\gamma(t_s)$ and ending at P_n. Using De Casteljau evaluating at $t = \frac{1}{2}$ we can compute the first midpoint:

$$
\begin{array}{ccccccc}
\boldsymbol{P_0} & & & & & & \\
 & \boldsymbol{P_0^1} & & & & & \\
\boldsymbol{P_1} & & \boldsymbol{P_0^2} & & & & \\
 & \boldsymbol{P_1^1} & & \boldsymbol{P_0^3} & & & \\
\boldsymbol{P_2} & & \vdots & \vdots & \ddots & & \\
 & \boldsymbol{P_2^1} & & \vdots & \cdots & \boldsymbol{P_0^n} & \\
\boldsymbol{P_3} & & \vdots & \vdots & \iddots & & \\
\vdots & \vdots & & \boldsymbol{P_{n-3}^3} & & & \\
\boldsymbol{P_{n-1}} & & \boldsymbol{P_{n-2}^2} & & & & \\
 & \boldsymbol{P_{n-1}^1} & & & & & \\
\boldsymbol{P_n} & & & & & &
\end{array}
$$

```
prev_values = pointcount;
for i=2:pointcount
  count = 0;
  for j=i:pointcount
      midpoints(j, i) = (0.5 * midpoints(j, i - 1)) +  (0.5 * midpoints(j - 1, i - 1));
      count++;
  endfor
  prev_values = count;
endfor
```

From this we can use $\mathrm{P}_0,\ \mathrm{P}_0^1,\ \mathrm{P}_0^2,\ \mathrm{P}_0^3\ \cdots\ \mathrm{P}_0^n$ as the new control points to construct the curve $\gamma_1(t)$, which can be defined by the following formula:

$$\gamma_1(t) = \sum_{i=0}^{n} P_0^i(t_s)B_i^n(t), \qquad t \in [0,1]$$

```
%recursion call to compute the midpoint and subcurves
%towards the left
leftLoop = populate(leftDiv, currIt + 1, maxIT);
```

Similarly, $\gamma_2(t)$ can also be constructed by using $\mathrm{P}_0^n,\ \cdots\ ,\ \mathrm{P}_1^3,\ \mathrm{P}_2^2,\ \mathrm{P}_3^1,\ \mathrm{P}_\mathrm{n}$ as control points, which can be defined by the following formula:

$$\gamma_2(t) = \sum_{i=0}^{n} P_i^{n-i}(t_s)B_i^n(t), \qquad t \in [0,1]$$

```
%appending the points computed towards the right
rightLoop = populate(rightDiv, currIt + 1, maxIT);
```

Here leftDiv and rightDiv will contain the control points for $\gamma_1(t)$ and $\gamma_2(t)$. Furthermore, we can keep subdividing the curves $\gamma_1(t)$ and $\gamma_2(t)$ to approximate our curves with more precision and smoothness. Code implementation wise this algorithm has exponential growth, which is something to avoid, that is why when implemented usually is restricted to a certain number of iterations for the subdivision, in this case the variable maxIT represents that while currIt keeps track of the current iteration in which we are.

Examples:

Direct Evaluation:

Given $P_0(1,\ -1), P_1(2,0), P_2(3,-1)$ and a mesh of 5 nodes in [0,1] $t = [0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1]$, we compute $B_0^2(t), B_1^2(t), B_2^2(t)$.

$$B_0^2(t) = \binom{2}{0}(1-t)^2t^0 = 1 - 2t + t^2 \Rightarrow [1, 0.56, 0.25, 0.06, 0]$$
$$B_1^2(t) = \binom{2}{1}(1-t)^1t^1 = 2t - 2t^2 \Rightarrow [\ 0, 0.37, 0.50, 0.37, 0]$$
$$B_2^2(t) = \binom{2}{2}(1-t)^0t^2 = t^2 \Rightarrow [0\ \ \ 0.06, 0.25, 0.56, 1]$$

Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 3

We check the results with the ones from the implemented code

```
B =

   1.0000    0.5625    0.2500    0.0625         0   B_0^2(t)
        0    0.3750    0.5000    0.3750         0   B_1^2(t)
        0    0.0625    0.2500    0.5625    1.0000   B_2^2(t)
```

We can see that the output is the same. We can now compute the points for the x and y coordinate with $\sum_{i=0}^{n} P_i B_i^n(t)$

$$P_0B_0^2(0) + P_1B_1^2(0) + P_2B_2^2(0) = P_0'(1,-1)$$
$$P_0B_0^2\left(\frac{1}{4}\right) + P_1B_1^2\left(\frac{1}{4}\right) + P_2B_2^2\left(\frac{1}{4}\right) = P_1'(1.5,-0.62)$$
$$P_0B_0^2\left(\frac{2}{4}\right) + P_1B_1^2\left(\frac{2}{4}\right) + P_2B_2^2\left(\frac{2}{4}\right) = P_2'(2,-0.5)$$
$$P_0B_0^2\left(\frac{3}{4}\right) + P_1B_1^2\left(\frac{3}{4}\right) + P_2B_2^2\left(\frac{3}{4}\right) = P_3'(2.5,-0.62)$$
$$P_0B_0^2(1) + P_1B_1^2(1) + P_2B_2^2(1) = P_4'(3,-1)$$

We check the results with the ones from the implemented code.

```
yx =                                        yy =

   1.0000    1.5000    2.0000    2.5000    3.0000   -1.0000   -0.6250   -0.5000   -0.6250   -1.0000
```

We can see that the results are the same (each column represents one P').

De Casteljau:

Given $P_0(1,\ -1), P_1(2,0), P_2(3,-1)\ and\ t\ =\ \{0,\ \frac{1}{2}, 1\}$.

For $t\ = 0$:

$P_0(1,-1)$

$$P_0^1 = 0 * (2,0) + (1-0) * (1,-1) = (1,-1)$$

$P_1(2,0)$

$$P_1^1 = 0 * (3,-1) + (1-0) * (2,0) = (2,0)$$

$$P_0^2 = 0 * (2,0) + (1-0) * (1,-1) = (1,-1)$$

$P_2(3,-1)$

```
X =ug>

   1    1    1
   2    2    0
   3    0    0
```

```
Y =

  -1   -1   -1
   0    0    0
  -1    0    0
```

For $t\ = \frac{1}{2}$:

Daniel Herreros| 540002818
David Miranda | 540001818
Nestor Uriarte| 540000817
MAT300-SP23
Project 3

$P_0(1,-1)$

$P_0^1 = 0.5 * (2,0) + (1-0.5) * (1,-1) = (1.5,-0.5)$

$P_1(2,0)$ $\qquad P_0^2 = 0.5 * (2.5,-0.5) + (1-0.5) * (2,-0.5) = (2,-0.5)$

$P_1^1 = 0.5 * (3,-1) + (1-0.5) * (2,0) = (2.5,-0.5)$

$P_2(3,-1)$

```
X =

   1.0000   1.5000   2.0000
   2.0000   2.5000        0
   3.0000        0        0
```

```
Y =

  -1.0000  -0.5000  -0.5000
         0  -0.5000         0
  -1.0000         0         0
```

For $t = 1$:

$P_0(1,-1)$

$P_0^1 = 1 * (2,0) + (1-1) * (1,-1) = (2,0)$

$P_1(2,0)$ $\qquad P_0^2 = 1 * (3,-1) + (1-1) * (2,0) = (3,-1)$

$P_1^1 = 1 * (3,-1) + (1-1) * (2,0) = (3,-1)$

$P_2(3,-1)$

```
X =

   1   2   3
   2   3   0
   3   0   0
```

```
Y =

  -1   0  -1
   0  -1   0
  -1   0   0
```

Midpoint Subdivision:

Given: $P_0(0,\ -1),\ P_1(2,2),\ P_2(1,3)$ and performing two iterations of the algorithm we get the following results:

1st Iteration:

$P_0\ (0,-1)$

$P_0^1(1,\frac{1}{2})$

$P_1\ (2,2)$ $\qquad P_0^2\ (\frac{5}{4},\frac{3}{2})$

$P_1^1\ (\frac{3}{2},\frac{5}{2})$

$P_2\ (1,3)$

Points in $\gamma(t)$ after the iteration: {(0, -1), $(\frac{5}{4},\frac{3}{2})$, (1, 3) }.

2 new curves can be created: $\gamma_1(t)$ and $\gamma_2(t)$ with control points:

$\gamma_1(t)$: {(0, -1), $(1, \frac{1}{2})$, $(\frac{5}{4}, \frac{3}{2})$}

$\gamma_2(t)$: {$(\frac{5}{4}, \frac{3}{2})$, $(\frac{3}{2}, \frac{5}{2})$, (1, 3)}

2nd Iteration:

For $\gamma_1(t)$ given the control points $P_0(0, -1)$, $P_1(1, \frac{1}{2})$, $P_2(\frac{5}{4}, \frac{3}{2})$:

$$P_0\ (0,-1)$$
$$P_0^1(\frac{1}{2}, \frac{-1}{4})$$
$$P_1\ (1, \frac{1}{2}) \qquad P_0^2\ (\frac{13}{16}, \frac{3}{8})$$
$$P_1^1\ (\frac{9}{8}, 1)$$
$$P_2\ (\frac{5}{4}, \frac{3}{2})$$

For $\gamma_2(t)$ given the control points $P_0(\frac{5}{4}, \frac{3}{2})$, $P_1(\frac{3}{2}, \frac{5}{2})$, $P_2(1, 3)$:

$$P_0\ (\frac{5}{4}, \frac{3}{2})$$
$$P_0^1(\frac{11}{8}, 2)$$
$$P_1\ (\frac{3}{2}, \frac{5}{2}) \qquad P_0^2\ (\frac{21}{16}, \frac{19}{8})$$
$$P_1^1\ (\frac{5}{4}, \frac{11}{4})$$
$$P_2\ (1, 3)$$

Points in $\gamma(t)$ after the iteration: {(0, -1), $(\frac{13}{16}, \frac{3}{8})$, $(\frac{5}{4}, \frac{3}{2})$, $(\frac{21}{16}, \frac{19}{8})$, (1, 3)}.

4 new curves can be created $\gamma_{11}(t)$, $\gamma_{12}(t)$, $\gamma_{21}(t)$, $\gamma_{22}(t)$ with control points:

$\gamma_{11}(t)$: {(0, -1), $(\frac{1}{2}, \frac{-1}{4})$, $(\frac{13}{16}, \frac{3}{8})$}

$\gamma_{12}(t)$: {$(\frac{13}{16}, \frac{3}{8})$, $(\frac{9}{8}, 1)$, $(\frac{5}{4}, \frac{3}{2})$}

$\gamma_{21}(t)$: {$(\frac{5}{4}, \frac{3}{2})$, $(\frac{11}{8}, 2)$, $(\frac{21}{16}, \frac{19}{8})$}

$\gamma_{22}(t)$: {$(\frac{21}{16}, \frac{19}{8})$, $(\frac{5}{4}, \frac{11}{4})$, (1, 3)}

Bibliography:

MAT300 Lecture Notes: lecture11, lecture 12.