



Experiment No. 8

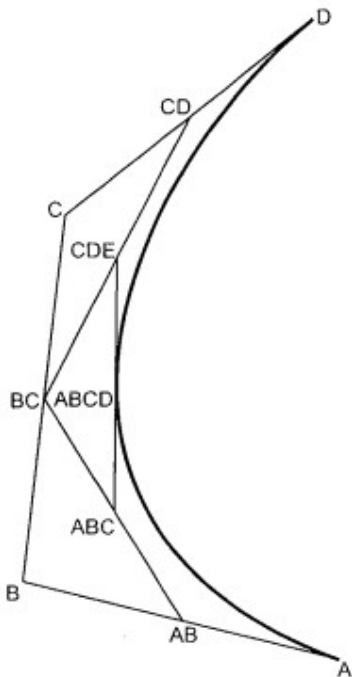
Aim: To implement Bezier curve for n control points. (Midpoint approach)

Objective:

Draw a Bezier curves and surfaces written in Bernstein basis form. The goal of interpolation is to create a smooth curve that passes through an ordered group of points. When used in this fashion, these points are called the control points.

Theory:

In midpoint approach Bezier curve can be constructed simply by taking the midpoints. In this approach midpoints of the line connecting four control points (A, B, C, D) are determined (AB, BC, CD, DA). These midpoints are connected by line segment and their midpoints are ABC and BCD are determined. Finally, these midpoints are connected by line segments and its midpoint ABCD is determined as shown in the figure –





The point ABCD on the Bezier curve divides the original curve in two sections. The original curve gets divided in four different curves. This process can be repeated to split the curve into smaller sections until we have sections so short that they can be replaced by straight lines.

Algorithm:

- 1) Get four control points say A(xa, ya), B(xb, yb), C(xc, yc), D(xd, yd).
- 2) Divide the curve represented by points A, B, C, and D in two sections.

$$x_{ab} = (x_a + x_b) / 2$$

$$y_{ab} = (y_a + y_b) / 2$$

$$x_{bc} = (x_b + x_c) / 2$$

$$y_{bc} = (y_b + y_c) / 2$$

$$x_{cd} = (x_c + x_d) / 2$$

$$y_{cd} = (y_c + y_d) / 2$$

$$x_{abc} = (x_{ab} + x_{bc}) / 2$$

$$y_{abc} = (y_{ab} + y_{bc}) / 2$$

$$x_{bcd} = (x_{bc} + x_{cd}) / 2$$

$$y_{bcd} = (y_{bc} + y_{cd}) / 2$$

$$x_{abcd} = (x_{abc} + x_{bcd}) / 2$$

$$y_{abcd} = (y_{abc} + y_{bcd}) / 2$$

- 3) Repeat the step 2 for section A, AB, ABC, ABCD and section ABCD, BCD, CD, D.
- 4) Repeat step 3 until we have sections so that they can be replaced by straight lines.
- 5) Repeat small sections by straight lines.
- 6) Stop

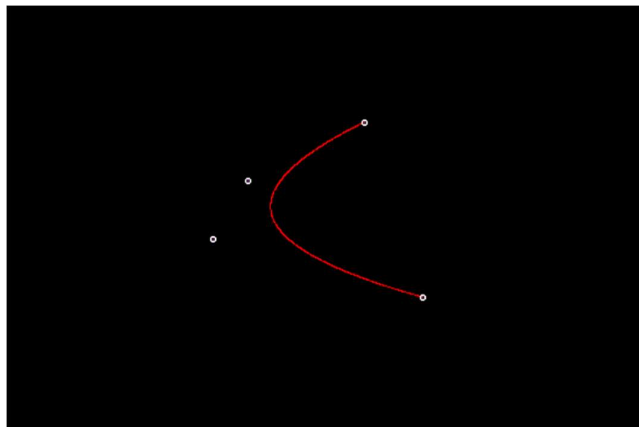
Program:

```
#include<graphics.h>
#include<math.h>
int x[4],y[4];
void bezier(int x[4],int y[4])
```



```
{
int gd=DETECT,gm,i;
double t,xt,yt;
initgraph(&gd,&gm," ");
for(t=0.0;t<1.0;t+=0.0005)
{
xt=pow((1.0-t),3)*x[0]+3*t*pow((1.0-t),2)*x[1]+3*pow(t,2)*(1.0-t)*x[2]+pow(t,3)*x[3];
yt=pow((1.0-t),3)*y[0]+3*t*pow((1.0-t),2)*y[1]+3*pow(t,2)*(1.0-t)*y[2]+pow(t,3)*y[3];
putpixel(xt,yt,4);
delay(5);
}
for(i=0;i<4;i++)
{
putpixel(x[i],y[i],5);
circle(x[i],y[i],2);
delay(2);
}
getch();
closegraph();
}
int main()
{
{
int i,x[4],y[4];
printf("Enter the four control points : ");
for(i=0;i<4;i++)
{
scanf("%d %d",&x[i],&y[i]);
}
}
bezier(x,y);
}
```

Output:





Conclusion:

Comment on –

1. Difference from arc and line- Bezier curves are parametric curves defined by control points and offer more flexibility in defining shapes compared to arcs. Arcs typically represent portions of circles or ellipses and have simpler geometric definitions. Bezier curves can create a wide range of curves with various shapes.

2. Importance of control point- Control points in the Bezier curve determine the shape and behavior of the curve. They provide a high degree of control and allow for precise manipulation of the curve's path. The number and position of control points influence the shape, direction, and curvature of the curve. Control points are essential for defining custom shapes and paths.

3. Applications-

- Graphic Design: Bezier curves are fundamental in graphic design software for creating and editing shapes, paths, and curves.
- Font Design: They are used to define and manipulate the shapes of characters and letters in font design.
- Animation: Bezier curves are employed to create smooth and natural motion paths for animation.
- Engineering and CAD: They are used in computer-aided design (CAD) for modeling curves and surfaces in engineering and architecture.
- Robotics: Bezier curves help define the paths that robots follow for tasks such as welding or assembly.
- Simulation: Bezier curves are used in simulations, such as flight simulations, to define the trajectory of objects.