Experiment No. 8

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

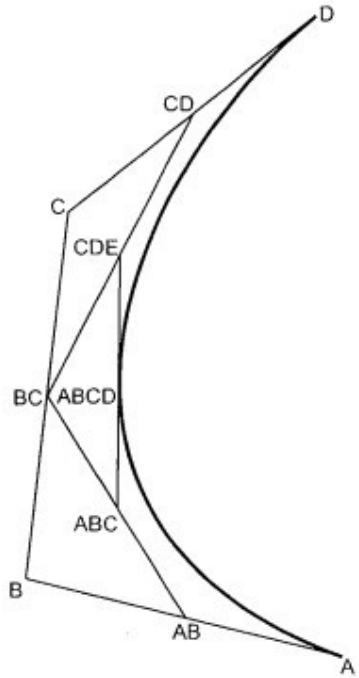
Objective:

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.

xab = (xa + xb) / 2

yab = (ya + yb) / 2 xbc =

(xb + xc) / 2 ybc = (yb + yc) /

2 xcd = (xc + xd) / 2

ycd = (yc + yd) / 2 xabc =

(xab + xbc) / 2 yabc =

(yab + ybc) / 2 xbcd =

(xbc + xcd) / 2 ybcd =

(ybc + ycd) / 2 xabcd =

(xabc + xbcd) / 2 yabcd =

(yabc + ybcd) / 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:

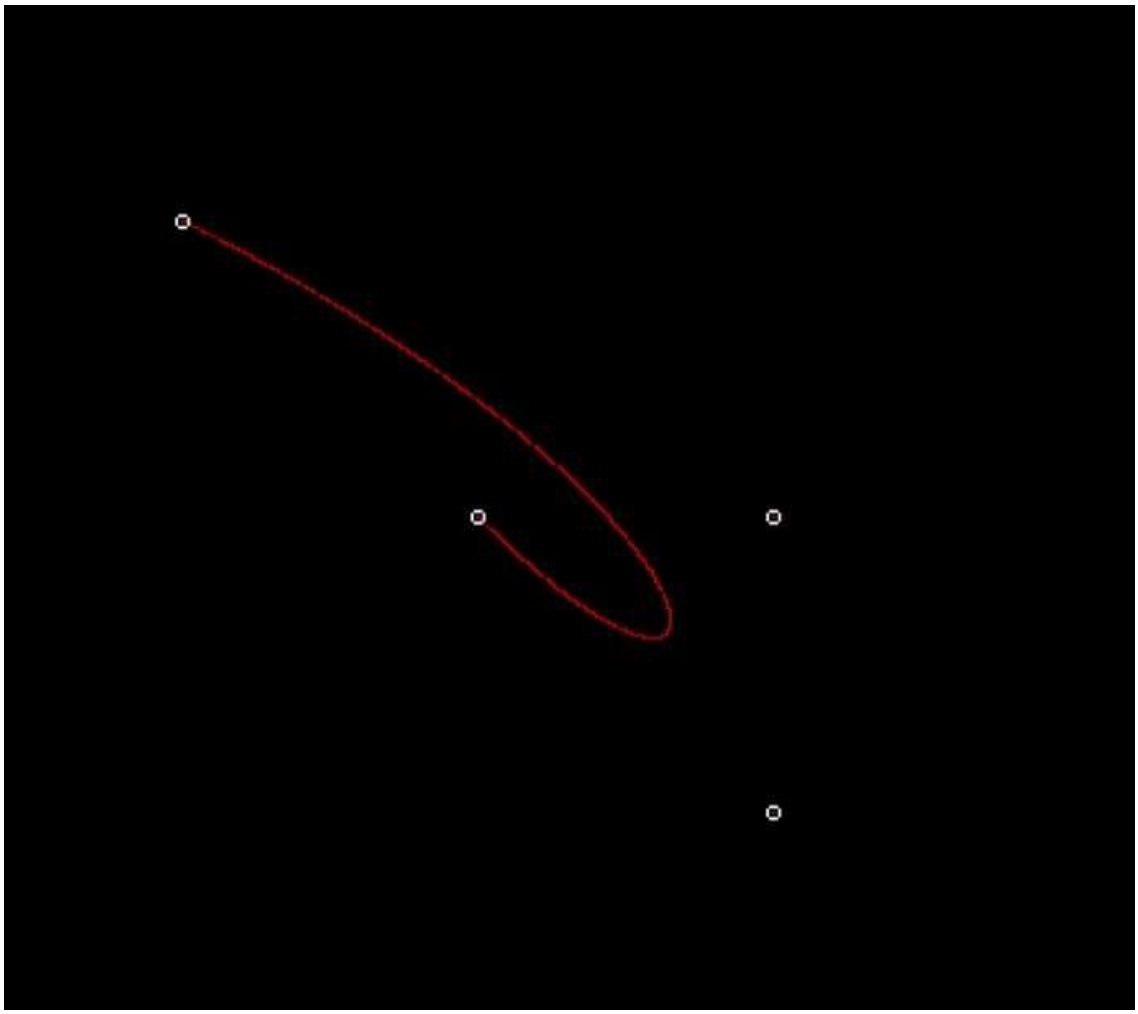
```
#include<graphics.h>

#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

Line:

- A line is a straight path connecting two points, extending infinitely in both directions if not limited by endpoints.
- Defined by two endpoints.

Arc:

- An arc is a curved segment of a circle or other curve, connecting two points along th e curve.

- Defined by endpoints and a center or radius, representing part of a circle's circumfer ence.

2. Importance of control point

Control points play a crucial role in defining and manipulating curves and shapes:

- Shape Control: They determine the curve's direction, curvature, and overall shape. B y adjusting control points, designers can fine-
  tune the shape to achieve the desired outcome.
- Complex Curves: Essential for creating complex shapes like Bézier curves and splin es, where multiple control points dictate the curve's flow and flexibility.
- Precision: Allow precise adjustments, making them invaluable in graphic design, ani mation, and CAD applications.

3. Applications

- Graphic Design: Used to create and manipulate vector graphics and intricate designs.
- Animation: Vital in creating smooth, fluid motions for characters and objects.
- CAD Software: Essential in designing and modeling precise mechanical parts and ar chitectural plans.
- Game Development: Helps in pathfinding, terrain modeling, and character animatio n.