Assignment 5: Bezier Kurver

Marie Elkjær Rødsgaard - dck495
Department of Computer Science

November 29, 2022

Contents

1 Introduktion

Denne opgave omhandler at forstå, forklare og implementere algoritmer for at visualisere Bezier kurver. I rapporten vil vi komme ind på Hermite kurver som giver os en geometri matrix G og en basis matrix M som vi kan bruge til at finde geomatri matrix og basis matrix til vores Bezier kurver. Vi kommer også ind på sampling, forward differencing, sub division og flatness som vi bruger til at tegne Bezier kurver.

2 Teori

Inden vi kigger på hvad en Bezier kurve er, kigger vi først på Hermite kurver. Hermite og Bezier kurver er to forskellige måder at beskrive et 3. gradspolynomium . En Hermite Kurve kan blive udtrykt som en Bezier kurver og omvendt.

3. grads polynomium den laveste grad af et polynomium, som kan være et ikke-plan. Et 3. grads polynomiel kan skrives op på følgene måde

$$f(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} a_x t^3 + b_x t^2 + c_x t + d_x \\ a_y t^3 + b_y t^2 + c_y t + d_y \\ a_z t^3 + b_z t^2 + c_z t + d_z \end{pmatrix}, \quad 0 \leq t \leq 1$$

Vi kan skrive dette om til

$$f(t) = \underbrace{\begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}}_{f(t)} = \underbrace{\begin{pmatrix} a_x + b_x + c_x + d_x \\ a_y + b_y + c_y + d_y \\ a_z + b_z + c_z + d_z \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}}_{t}, \quad 0 \leq t \leq 1$$

Vi har nu fået en koefficient matrix C og en parameter vektor t. Hvis vi nu vil ændre på den kurve vi har, så skal vi bruge vores C matrix. Det er dog ikke nemt at ændre dette da der er 12 værdier, så vi vil skrive vores C om så det er nemmere at gøre. Vi deler vores C matrix op i to matricer G og M. G kalder vi vores geometri matrix som bestemmer hvordan formen af kurven ser ud. M matrixen kalder vi basis matrix som bestemmer hvilken type kurve det er, eksempelvis Hermite eller Bezier. G matricen har størrelsen 3x4 og

M matricen har størrelsen 4x4.
$f(t) = Ct = GMt$

$$G = \begin{pmatrix} g_{1x} & g_{2x} & g_{3x} & g_{4x} \\ g_{1y} & g_{2y} & g_{3y} & g_{4y} \\ g_{1z} & g_{2z} & g_{3z} & g_{4z} \end{pmatrix}, \quad M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{41} \\ m_{21} & m_{22} & m_{23} & m_{42} \\ m_{31} & m_{32} & m_{33} & m_{43} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix}, \quad t = \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

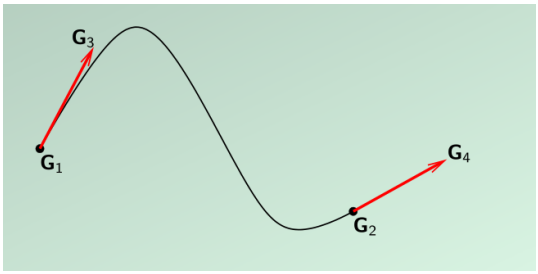

Figure 1: Eksempel på Hermite kurve med punkterne G_1, G_2, G_3 og G_4

Den ene type kurve vi har er Hermite kurver. Hermite kurver er defineret ved at have et startpunkt G_1 og et slut punkt $G2$, det er de punkter som er indgange til vores geometri matrix. Trejde indgang G_3 er tangentens start punkt. Fjerde indgang er G_4 tangentens slutpunkt. På figure 1 ses hvordan dette ser ud. Vi kan nu skrive vores geometri matrix op for Hermite kurven:

$$G_H = \begin{pmatrix} G_1 & G_2 & G_3 & G_4 \end{pmatrix}$$

Nu har vi så fundet vores geometri matrix G, vi vil nu også gerne finde vores basis matrix M_H. Vi kan skrive vores funktion f op igen

$$f_H(t) = G_H M_H t$$

Vi ved ikke hvad vores M_H er men vi ved noget om vores funktion: hvis vi sætter $t = 0$ som vores startpunkt og $t = 1$ som vores slutpunkt får vi

$$f_H(0) = G_1 = G_H M_H \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$f_H(1) = G_2 = G_H M_H \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

For nu at finde tangenterne af kurven skal vi bruge $f'(t)$ som fås af

$$f'_H(t) = G_H M_H \begin{pmatrix} 3t^2 \\ 2t \\ 1 \\ 0 \end{pmatrix}$$

Nu finder vi tangenternes startpunkt ved at sige $t = 0$ og slutpunkt $t = 1$

$$f'_H(0) = G_3 = G_H M_H \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$f'_H(1) = G_4 = G_H M_H \begin{pmatrix} 3 \\ 2 \\ 1 \\ 0 \end{pmatrix}$$

Hver ligning for startpunkterne og slutpunkterne svare til vores geometri matrix G_H, de fire ligninger kan vi derfor bruge til at udtrykke G_H

$$\begin{pmatrix} G_1 & G_2 & G_3 & G_4 \end{pmatrix} = G_H = G_H M_H \begin{pmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Dette kan kun være sandt hvis vores M_H er lig den inverse matrix af matricen på sidste led.

$$M_H = \begin{pmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix}$$

Nu har vi fundet både vores geometri matrix og basis matrix. Hvis vi nu ville ændre på kurven kunne vi vælge en af startpunkterne eller slutpunkterne og

ændre dem.

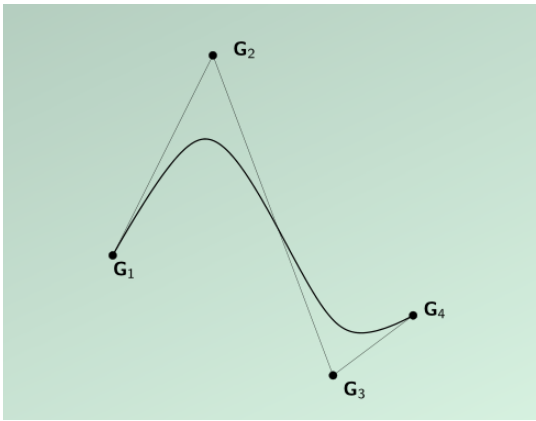

Figure 2: Bezier kurve med start og slutpunkt og dens kontrol punkter

Bezier Kurver er også 3. gradspolynomium . Som nævnt tidligere så kan Hermite kurver også laves som Beizer kurver og omvendt. I Bezier kurver har vi startpunktet G_1 og slutpunktet G_4. Punkterne G_2 og G_3 kalder vi for kontrolpunkter. Kontrolpunkterne ligger ikke nødvendigvis på kurven som også ses på figure 2 ovenover. Vores geometri matrix G_B skrives som

$$G_B = \begin{pmatrix} G_1 & G_2 & G_3 & G_4 \end{pmatrix}$$

Ligesom med Hermite kurve har vi en funktion f som er

$$f(t) = GMt$$

Fordi de to typer kurver kan blive udtrykt som den anden kan vi skrive det op således

$$f(t) = GMt = G_H M_H t = G_B M_B t$$

Vi kender vores to geometri matricer og vores basis matrix M_H. Der er en relationen mellem vores geometri matrix G_H og G_B. Hvis vi tager den vektor der går fra G_1 til G_2 på vores Bezier så skal den være lig med 3 ganget med tangenten vektorens startpunkt fra vores Hermite kurve. Det samme gælder linje fra G_3 til G_4 på Bezier med G_4 tangent vektorens slutpunkt på Hermite. Vi vælger tallet 3 i denne relation fordi i en bestemt Bezier kurver, hvor alle

kontrol punkter ligger på en linje med samme afstand så når man kører med sine parameter 0 til 1 så bevæger punktet sig med konstant hastighed på kurven, det betyder at vores start tangent og slut tangent er 3 ganget med afstanden mellem de punkter. Den relation ses i liningen nedefor.

$$G_H = \begin{pmatrix} G_1 & G_2 & G_3 & G_4 \end{pmatrix} = \begin{pmatrix} G_1 & G_4 & 3(G_2 - G_1) & 3(G_4 - G_3) \end{pmatrix}_B$$

Det kan vi så nu skrive om på matrix form

$$G_H = \begin{pmatrix} G_1 & G_2 & G_3 & G_4 \end{pmatrix}_B \begin{pmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{pmatrix}_{H \leftarrow B} = \begin{pmatrix} G_1 & G_4 & 3(G_2 - G_1) & 3(G_4 - G_3) \end{pmatrix}$$

Nu vil vi så gerne finde vores basis matrix for vores Bezier. Vi har nu en matrix der tager os fra Bezier til Hermite.

$$M_{H \leftarrow B} = \begin{pmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{pmatrix}_{H \leftarrow B}$$

Det betyder at

$$G_H = G_B M_{H \leftarrow B}$$

som giver os

$$f(t) = G_H M_H t = (G_B M_{H \leftarrow B}) M_H t = G_B (M_{H \leftarrow B} M_H) t$$

Jeg ganger nu de to matrixer sammen for at få Bezier matrixen

$$M_B = M_{H \leftarrow B} M_H = \begin{pmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{pmatrix}_{H \leftarrow B} \begin{pmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix}_H = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Nu har vi fundet G_B og M_B.

Nu hvor vi har gennemgået hvad Bezier kurver, geometri og basis matricen er kan vi kigge på hvordan vi kan tegne vores Bezier kurver. De fire

måder vi kigger på er 'sampling', 'forward differences', 'sub division' og 'flatness'. For at tegne en generel kurve med sampling tager man et punkt t_0 på kurven også beregner man videre $t_0 + \delta$ ind til $t_0 + n\delta$. Man tegner så kurven ved at tegne streger mellem punkterne, jo tættere punkterne er desto pænere en kurve. Det er dog dyrt at sample da man skal ind og beregne 3.gradspolynomium hver gang.

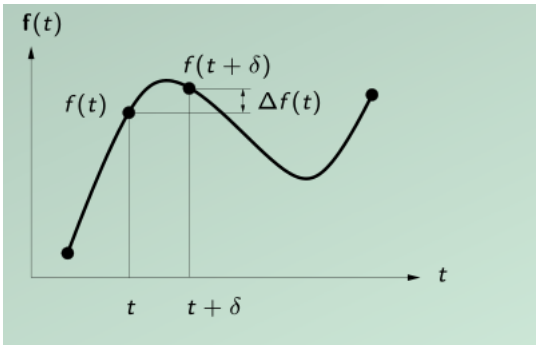

Figure 3: Forward differences

Forward differences virker godt til at tegne polynomier så som 3. gradspolynomium som Hermite og Bezier kurver. Hvis vi ganger vores G og M matrix ud så får vi vores koefficienter a, b og c.

$$f(t) = at^3 + b^t 2 + c^t + d$$

Forward differences gør at vi kan smide vores multiplikationer væk og nøjes kun med additioner for at finde vores punkter. Figure 4 over viser punkter vi kommer til at kigge på. Vi starter i punktet $f(t)$ og beregner det. For at finde næste punkt beregner vi ikke $f(t + \delta)$ men i stedet for finder vi $\Delta f(t)$, som er hvor meget vores funktionsværdi ændre sig. Det ender så med at vi kan addere vores delta værdier hver gang for at finde det næste punkt. Før vi dog kan gøre dette skal vi følgene

$$\Delta f(t) = f(t + \delta) - f(t)$$

$$f(t + \delta) = f(t) + \Delta f(t)$$

Vores f fandt vi ud af før var på formen af et 3. gradspolynomium

$$f(t) = at^3 + bt^2 + ct + d$$

så kan vi finde $\Delta f(t)$

$$\Delta f(t) = 3a\delta t^2 + (3a\delta^2 + 2b\delta)t + a\delta^3 + b\delta^2 + c\delta$$

Vi kan nu se at vi er gået fra et 3. gradspolynomium til et 2. grads. Vi kan nu fortsætte og finde $\Delta^2 f(t)$ osv indtil vi har en konstant. Det gør vi på følgene måde

$$\Delta^2 f(t) = \Delta(\Delta f(t)) = \Delta f(t+\delta) - \Delta f(t)$$

$$\Delta f(t+\delta) = \Delta f(t) + \Delta^2 f(t)$$
$$\Delta^2 f(t) = 6a\delta^2 t + 6a\delta^3 + 2b\delta^2$$

$\Delta^2 f(t)$ kan vi nu se er et første grads polynomium. Vi køre en gang til for at ende med en konstant.

$$\Delta^3 f(t) = \Delta(\Delta^2 f(t)) = \Delta^2 f(t+\delta) - \Delta^2 f(t)$$

$$\Delta^2 f(t+\delta) = \Delta f(t) + \Delta^3 f(t)$$
$$\Delta^2 f(t) = 6a\delta^3$$

Nu har vi $\Delta^3 f(t)$ som er en konstant. Jeg kan nu tegne kurven $f(t) = at^3 + bt^2 + ct + d;\ 0 \leq t \leq 1$. Ved at bruge forward differences med n antal punkter $\delta = \frac{1}{n}$ siger vi

$$\begin{pmatrix} f_0 \\ \Delta f_0 \\ \Delta^2 f_0 \\ \Delta^3 f_0 \end{pmatrix} = \begin{pmatrix} f(0) \\ \Delta f(0) \\ \Delta^2 f(0) \\ \Delta^3 f(0) \end{pmatrix} = \begin{pmatrix} d \\ a\delta^3 + b\delta^2 + c\delta \\ 6a\delta^3 + 2b\delta^2 \\ 6a\delta^3 \end{pmatrix}$$

Den nye f værdi får vi ved at ligge rækkerne i og i + 1 sammen. Vi kan også skrive det op som

$$\begin{pmatrix} f_{i+1} \\ \Delta f_{i+1} \\ \Delta^2 f_{i+1} \\ \Delta^3 f_{i+} \end{pmatrix} = \begin{pmatrix} f_i \\ \Delta f_i \\ \Delta^2 f_i \\ \Delta^3 f_i \end{pmatrix} + \begin{pmatrix} \Delta f_i \\ \Delta^2 f_i \\ \Delta^3 f_0 \\ 0 \end{pmatrix} = \begin{pmatrix} f_i + \Delta f_i \\ \Delta f_i + \Delta^2 f_i \\ \Delta^2 f_i + \Delta^3 f_i \\ \Delta^3 f_0 \end{pmatrix}$$

Det kan ses her at det altså kun tager 3 additioner at finde det næste punkt $f_i + 1$ fra vores punkt f_i. Sådan virker Forward Differences.

Når man laver sub division har man en kurve hvor man laver en linje mellem startpunktet og slutpunktet. Hvis den kurve man har er tilstrækkelig flad så siger man at den linje er en god nok tilnærmelse. Hvis ikke kurven er tilstrækkelig flad laver man et dele punkt, det kunne være ca i midten. Så kigger man så på den nye kortere del af linje og ser om det er en god tilnærmelse. Dette bliver man ved med indtil de punkter man har er tilstrækkelig tæt på kurvens afstand.

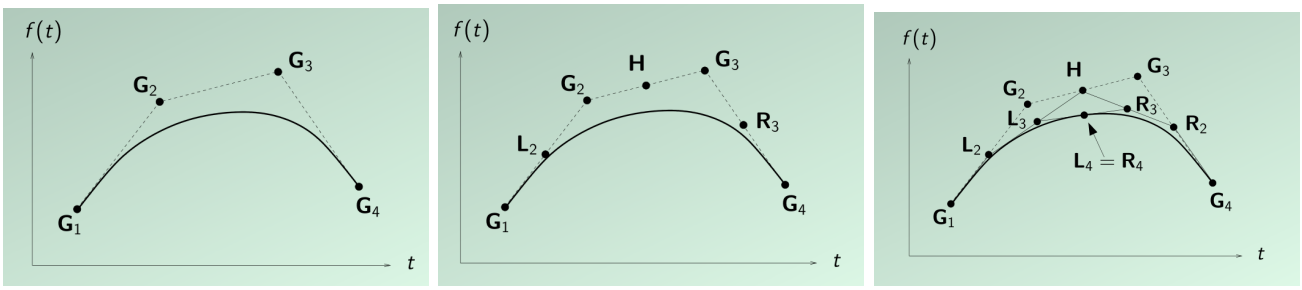

Figure 4: Sub division på Bezier kurve

På Bezier kurver har vi vores fire kontrol punkter G_1, G_2, G_3, G_4. Når vi så laver sub division tager vi og deler vores kontrol punkter på midten de tre steder så vi får L_2, H, R_3, se figure 4. Nu tager vi igen og deler det sidste stykke på midten så får vi punktet $L_4 = R_4$ se figure 4. Det sidste punkt vi finder det ligger på vores Bezier kurve. Vores L og R punkter er nye kontrolpunkter for halvdelen af kurven dvs $G_L^B = [L_1, L_2, L_3, L_4]$ og $G_R^B = [R_1, R_2, R_3, R_4]$. Man kan dele venstre- og højresiden op i flere kontrolpunkter igen hvis det ikke passer efter første gang. De nye punkter kan findes ved kun at udtrykke den med G'er.
$L1 = G1$
$L2 = G1/2 + G2/2$
$L3 = G1/4 + G2/2 + G3/4$
$L4 = G1/8 + 3G2/8 + 3G3/8 + G4/8$
$R1 = G4$
$R2 = G3/2 + G4/2$
$R3 = G2/4 + G3/2 + G4/4$
$R4 = G1/8 + 3G2/8 + 3G3/8 + G4/8$
Vi kan nu få delingsmatricen for venstre og højre side ud ved at sige

$L1 = (8G1)/8$
$L2 = (4G1 + 4G2)/8$
$L3 = (2G1 + 4G2 + 2G3)/8$
$L4 = (G1 + 3G2 + 3G3 + G4)/8$
$R1 = (8G4)/8$
$R2 = (4G3 + 4G4)/8$
$R3 = (2G2 + 4G3 + 2G4)/8$
$R4 = (G1 + 3G2 + 3G3 + G4)/8$
Vi har nu fået alle 8 tallene ud fra parenteserne og kan finde vores matrice DLB og DRB. Vores venstre matrice er nu lig med

$$DLB = \begin{pmatrix} 8 & 4 & 2 & 1 \\ 0 & 4 & 4 & 3 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Vores højre matrix er

$$DRB = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 \\ 3 & 4 & 4 & 0 \\ 1 & 2 & 4 & 8 \end{pmatrix}$$

Med de to matricer kan vi nu finde vores kontrolpunkter rekursivt. Vi finder så de nye kontrol punkter for venstre side ved at sige $G_L^B = \begin{pmatrix} L_1 & L_2 & L_3 & L_4 \end{pmatrix} = \begin{pmatrix} G_1 & G_2 & G_3 & G_4 \end{pmatrix} DLB$ og på samme måde med vores højre side. På den måde bruges matrixerne til at finde de nye kontrolpunkter.

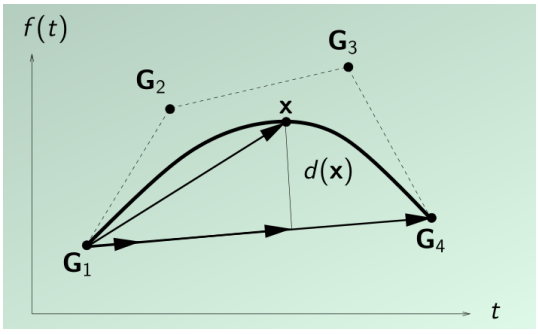

Figure 5: Flatness test

Nå man har Bezier kurven laver man så en flatness test for at sørge for at linjerne er tæt nok. En flatness test består af 7 ting vi skal tjekke ikke er sande. Først skal vi tage et punkt på kurven x, derefter vil vi så finde afstanden $d(x)$, se figure 5. For at finde afstanden tager jeg vektoren fra G_1 op til x. Så laver vi en enhedsvektor af den vektor fra G_1 op til G_4. Så tager vi de to vektorer og prikker dem. Så får vi længden op til d(x) på vektoren fra G_1 op til G_4. Hvis vi så trækker den nye vektor fra vores G_1 op til x vektor så får vi vektoren d(x) og så kan vi udregne længden på den. Hvis den længde er mindre end vores ϵ så betyder det at vores Bezier kurve er flad nok. Der er flere ting vi skal tjekke i vores flatness test. Hvis alle tingene er falske så er kurven flad nok ellers deler vi med et nyt punkt ved brug af sub division. De ting der skal tjekkes er
$||\text{G1G4}|| = 0$
$\text{G1G2} \cdot u < 0$
$\text{G4G3} \cdot u > 0$
$\text{G1G2} \cdot u > ||\text{G1G4}||$
$\text{G4G3} \cdot u > ||\text{G1G4}||$
$d_1 = ||\text{G1G2}(\text{G1G2} \cdot u)u|| > \epsilon$
$d_2 = ||\text{G4G3}(\text{G4G3} \cdot u)u|| > \epsilon$
Hvis alle kravene er falske er kurven flad nok. Dette kalder man også adeptive subdivision.

3 Implementation

For at tegne de forskellige Bezier kurver skal vi implementere metoderne som blev gennemgået i teorien. Vi starter med at implementere sampling for at tegne Bezier kurve. Vi starter med at initialisere vores t værdier, vores delta=$\frac{1}{N}$ værdi og en vektor T. Så pusher vi vores første punkt så det bliver tegnet. Derefter kører vi et loop hvor vi sætter t1 +0 delta, t2 = t1 * t1 og t3 = t2 * t1. Dem tilføjer vi så til vores vektor T og derefter tager vi så og ganger vores geometri matrix og basis matrix med vores parameter vektor T så vi får vores vertex. Til sidst pusher vi vores vertex. Vi beregner på den måde hvergang det nye vertex. Efter vi har været alle punkterne igennem pusher vi også vores slutpunkt G_4.

For at implementere forward differences starter vi med at initialisere vores delta værdier 1-3. Derefter initialisere vi vores Bezier koefficient ved at gange

vores geometri matrix med vores basis matrix. Så initialisere vi vores forward differences række ved at skrive ind hvad hver Δf er. Efter det pusher vi igen vores første punkt G_1. Så kører vi en for loop hvor vi siger

```
F[1] += F[2];
F[2] += F[3];
F[3] += F[4];
```

Vi pusher så vores F[1] værdi hvergang. Igen pusher vi vores slutpunk G_4 til sidst efter loopet.

For at implementere sub division gør vi det rekursivt. Vi har en N værdi som er hvor mange gange vi vil dele kurven hvis den N er 0 betyder det vi ikke vil dele mere og at vi pusher vores start og slutpunkt. Ellers kalder vi `SubDivide` 2 gange.

```
SubDivide(G * DLB, N-1, Vertices);
SubDivide(G * DRB, N-1, Vertices);
```

På denne måde bliver vi ved med at finde vores nye kontrolpunkter indtil det ikke er nødvendigt at dele kurven mere.

Flatness testen er også blevet implementeret. Her laver jeg de 7 tjek vi så i teorien. Hvis ingen af de 7 tjek er sande så returnere jeg true. Så bruger jeg en ny subdivide, der ligner den første ud over at vi nu har bare et ekstra thresshold `Flatness(G, epsilon)`. Nu har vi implementeret alle de ønskede dele.

4 Test

Det implementerede program for at tegne Bezier kurver bruger af sampling, forward differences, sub division og flatness. Da alle metoderne tegner graferne ens med de mindste forskelle antager vi at metoderne er rigtigt implementeret. Alle graferne ligner hinanden med de forskellige metoder. På figure 6 under ses graferne tegnet med de forskellige metoder.

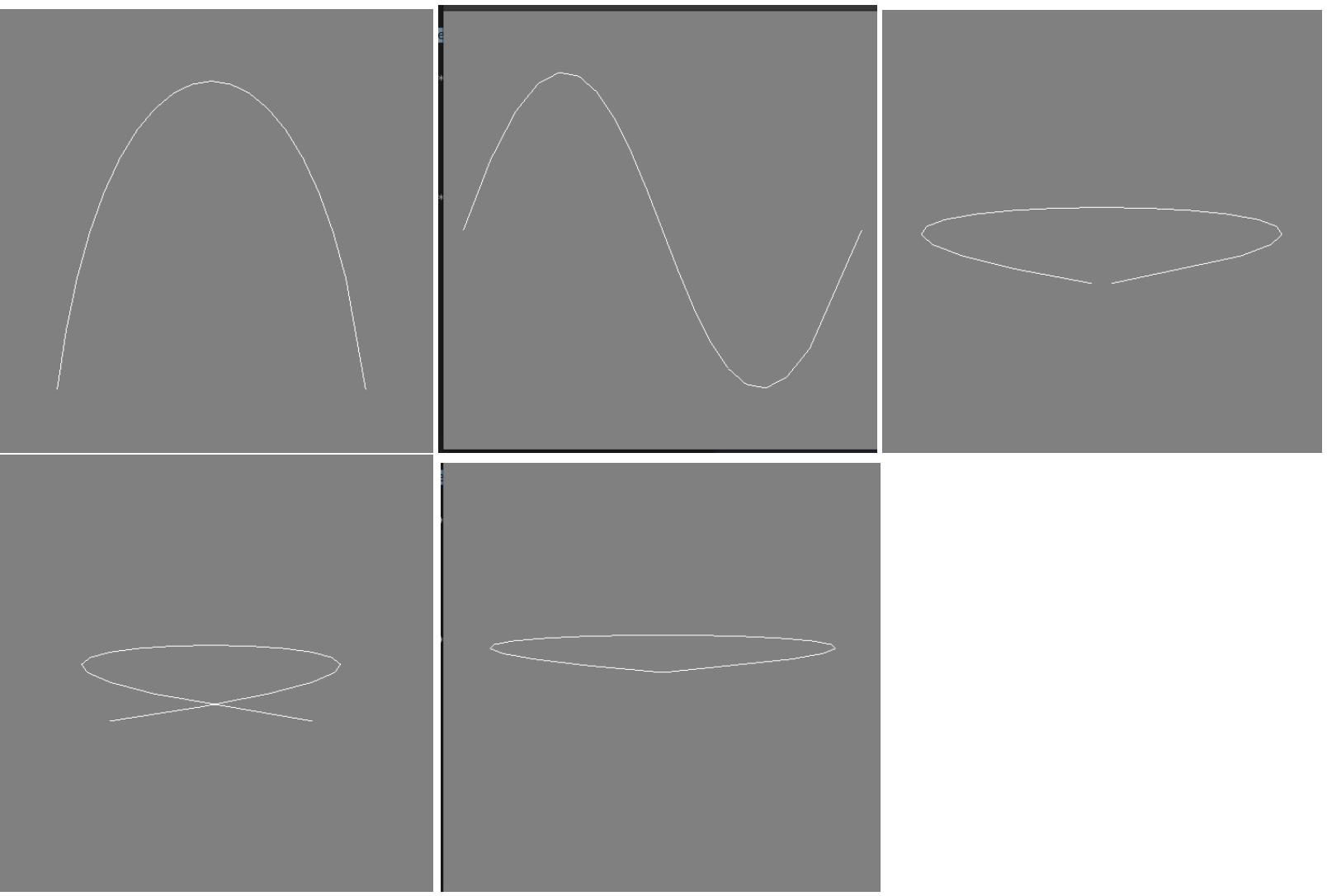

Figure 6: Sampling implementation af Bezier kurve

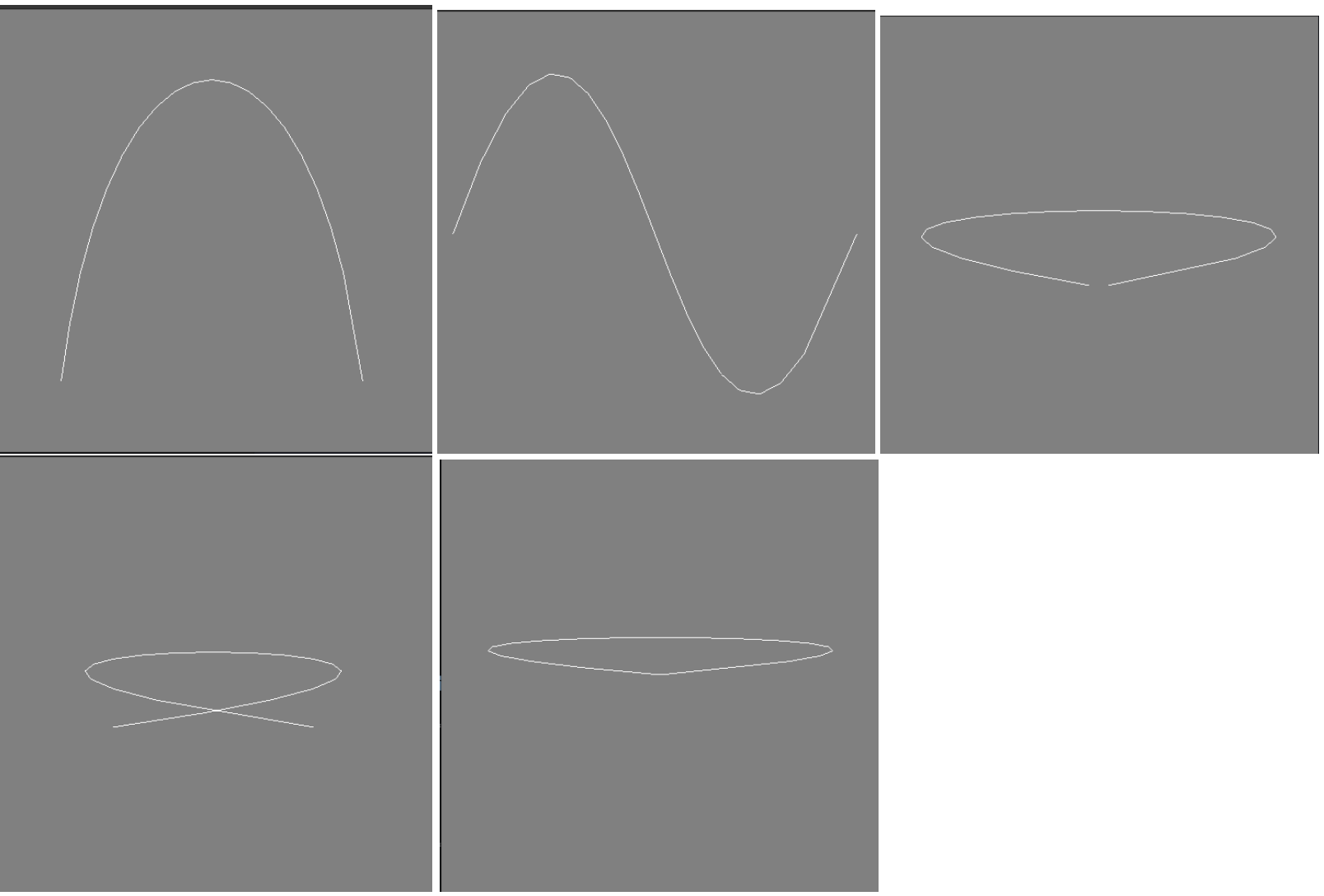

Figure 7: Forward differences implementation af Bezier kurve

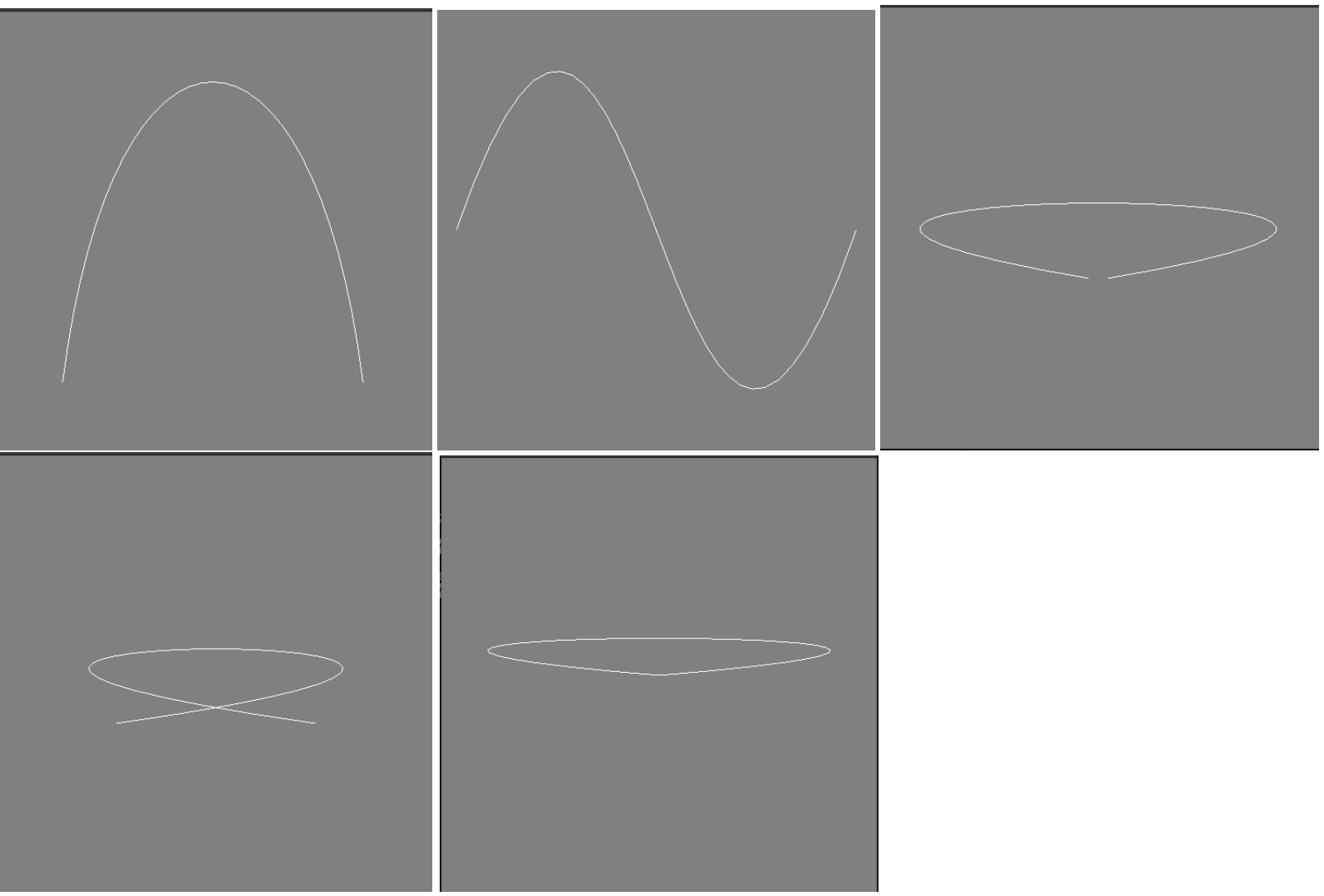

Figure 8: Sub division implementation af Bezier kurve

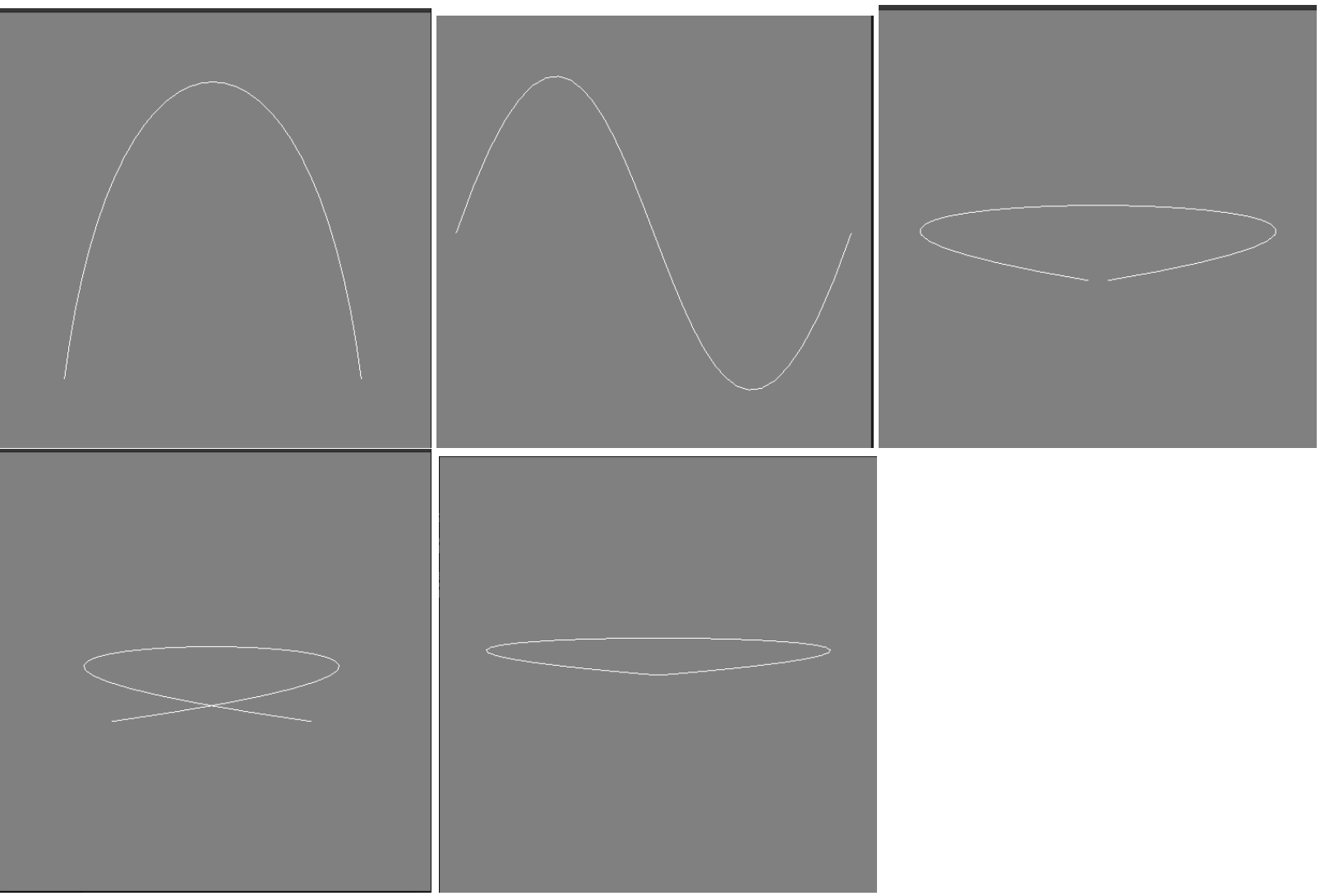

Figure 9: Flatness implementation (adaptive sub division) af Bezier kurve

5 Konklusion

Vi har nu forklaret og implemneteret algoritmerne for at tegne Berzier kurver. Vi har set hvad geometri og basis matrix er og hvordan vi finder dem. Vi har også kigget på de forskellige måder at tegne Bezier kurver på ved brug af sampling, forward differences, sub division og flatness. Tilsidst har vi testet at implementationen af de forskellige metoder virker som ønsket.

References