Кривата на Безие и нейната първа поляра

курсов проект по Компютърно геометрично моделиране (CAGD)

Лектор: доц. Красимира Влъчкова

Цветомир Стайков - 1MI0800469

2 курс, Компютърни науки

12 февруари 2025 година

Съдържание

1. Въведение

Настоящият проект представя двуизмерна крива на Безие и нейната първа поляра. За генерирането на кривите се използва алгоритъмът на de Casteljau. Реализирани са стандартни функции за създаване на криви. Програмата работи в уеб браузър и е тествана на настолни компютри.

2. Структура и технологии

Проектът представлява уеб приложение, разработено с HTML, CSS и JavaScript, като използва панел за рисуване, известен като Canvas, за визуализация на кривите. Не се използват външни библиотеки, което позволява проекта да работи без интернет връзка. Изборът на тези технологии позволява лесна адаптация и използване на програмата на различни устройства.

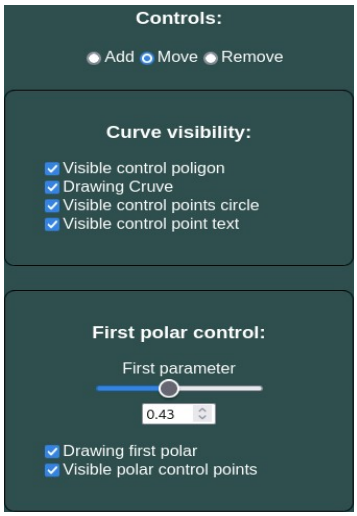
Компресирана структура на проекта

- *CAGD-1MI0800469.html* – Целият проект, събран в един файл.

Оригинална структура на проекта

- *index.html* – Начална страница на сайта.
- *styles.css* – Дизайн на интерфейса.
- *Interface.js* – Функции за създаване, управление и изтриване на контролните точки, както и визуализация на крайната крива.
- *Algorithms.js* – Алгоритми за изчисление на кривата и нейната първа поляра.
- *Debug.js* – Помощни функции за извеждане на информация и грешки.

3. Интерфейс



Потребителският интерфейс се състои от две основни части:

- Ляв панел за управление - Панелът има настройки за създаване на крива. Позволява добавяне, преместване и премахване на контролни точки. Разполага с отметки за промяна на видимостта на контролния полигон, точките, кривата, надписите, както на полярата и неините точки. Предвиден е слайдер за управление на първия параметър на

поларята. До него има текстово поле, което приема числови стойности за по-прецизен контрол на параметъра с до два знака след десетичната запетая. Въведено е ограничение в интервала [0,1].

За управление на контролните точки в горната част на панел има три режима – Add, Move и Remove. При стартиране на програмата по подразбиране е активиран режимът Add.

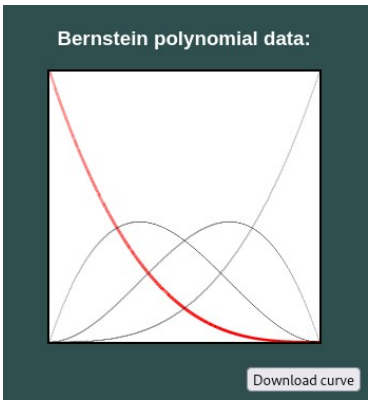
- Десен панел за рисуване (Canvas) – След избиране на една от настройките за контролните точки чрез натискане на десен бутон на мишката може да се манипулират точките върху този панел. Когато курсорът е върху контролна точка в режим Move или Remove, тя се огражда в червено, сигнализирайки, че може да бъде избрана.

Основни функции върху контролните точки

- Add – Позволява добавяне на нови точки, като ново добавената става следващата в реда. Точка се добавя чрез натискане на левия бутон върху панела за рисуване.
- Move – Като курсорът се намира върху контролна точка, тя се огражда с червен кръг. При натискане и задържане на лявия бутон този кръг изчезва, а контролната точка ще следва движението на мишката.
- Remove – Работи аналогично на Move, но вместо да премества контролната точка, я изтрива при натискане на лявия бутон.

Допълнително

- Програмата позволява изтеглянето на нарисуваната крива без фон, чрез бутон „Download curve“, най-долу в контролния панел.
- В малък панел в долната част на контролния панел са представени полиномите на Бернщайн. Когато курсорът е върху контролна точка в режим Move или Remove, съответният полином се маркира в червено.



4. Алгоритъм

За пресмятане на кривата е използван алгоритъм на de Casteljau. След добавянето на контролни точки, програмата автоматично генерира кривата. Използва се цикъл с малка стъпка на t, за да осигури гладкост.

Програмата използва две функции за генериране на кривата и нейната поляра и три помощни подфункции. Главните функции имат за цел да нарисуват кривата със стъпка 0.01 и проверяват дали има достатъчно контролни точки.

Помощните подфункции изчислят междинните точки, като резултата е нужната точка за чертаене на кривата. Приемат като аргумент t и контролни точки, след което се използва алгоритъма на de Casteljau.

```
function calculateAtPos(t,LocalControlPoints){
    let count = LocalControlPoints.length;

    let intermediatePoints = [LocalControlPoints]

    for(let i = 0;i<count-1;++i){
        let prev = intermediatePoints[i];
        let newPoints = new Array(prev.length-1);

        for(let j =0;j<prev.length-1;++j){
            let first = prev[j];
            let second = prev[j+1];
            newPoints[j] = {
                x: (1-t)* first.x + t * second.x,
                y: (1-t)* first.y + t * second.y
            }
        }
        intermediatePoints.push(newPoints);

    }

    return intermediatePoints[intermediatePoints.length-1][0];
}
```

```
function drawCurve(){
    if(controlPoints.length <=1){
        return;
    }
    ctx.lineWidth = 2;
    ctx.strokeStyle = "grey";

    for(let i =0;i<=1;i+=0.001){
        pos =calculateAtPos(i,controlPoints);

        ctx.beginPath();
        ctx.moveTo(pos.x, pos.y);

        pos =calculateAtPos(i+0.001,controlPoints);

        ctx.lineTo(pos.x, pos.y);
        ctx.stroke();
    }

    ctx.strokeStyle = "black";
}
```

Функцията за поляра използва същия метод, но тя допълнително приема променливата t_1, подадена от интерфейса. Тази променлива се подава на подфункцията, която изчислява полярната крива, като замества t при първоначалното изчисляване на алгоритъма.

```
function calculateAtPosFirstPolar(t0,t,LocalControlPoints){
    let count = LocalControlPoints.length;

    let intermediatePoints = [LocalControlPoints]

    for(let i = 0;i<count-1;++i){
        let prev = intermediatePoints[i];
        let newPoints = new Array(prev.length-1);
        for(let j =0;j<prev.length-1;++j){

            let first = prev[j];
            let second = prev[j+1];

            if(i == 0){
                newPoints[j] = {
                    x: (1-t0)* first.x + t0 * second.x,
                    y: (1-t0)* first.y + t0 * second.y
                }
            }else{
                newPoints[j] = {
                    x: (1-t)* first.x + t * second.x,
                    y: (1-t)* first.y + t * second.y
                }
            }
        }
        intermediatePoints.push(newPoints);
    }
    return intermediatePoints[intermediatePoints.length-1][0];
}
```

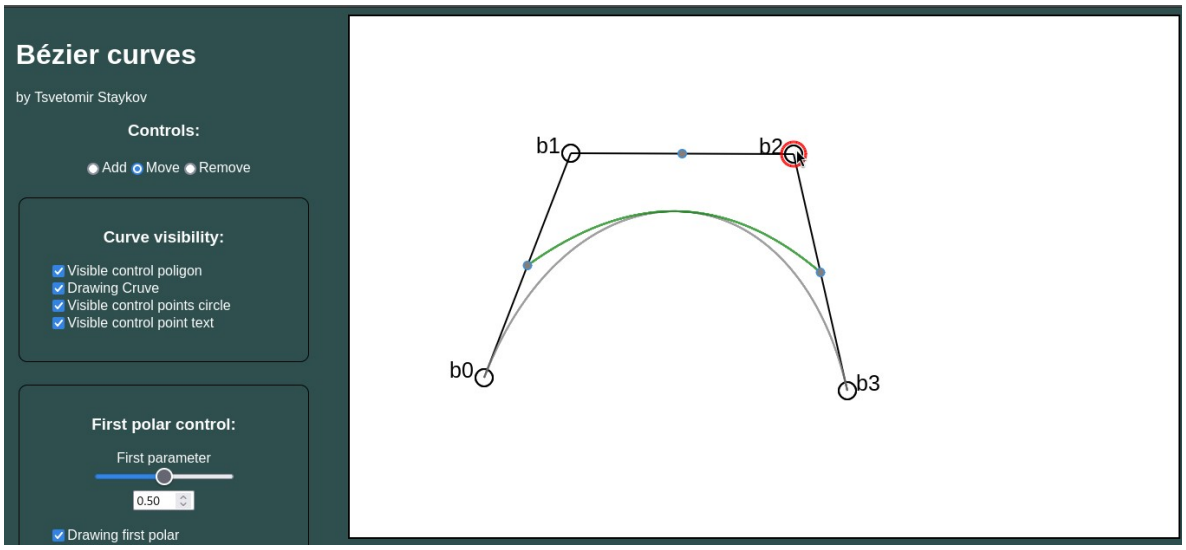
```
function PolarControlPoints(t0,LocalControlPoints){

    let count = LocalControlPoints.length;
    let PolarControlPoints = new Array(count-1);

    for(let j =0;j<count-1;++j){

        let first = LocalControlPoints[j];
        let second = LocalControlPoints[j+1];

        PolarControlPoints[j] = {
            x: (1-t0)* first.x + t0 * second.x,
            y: (1-t0)* first.y + t0 * second.y
        }

    }

    return PolarControlPoints;
}
```

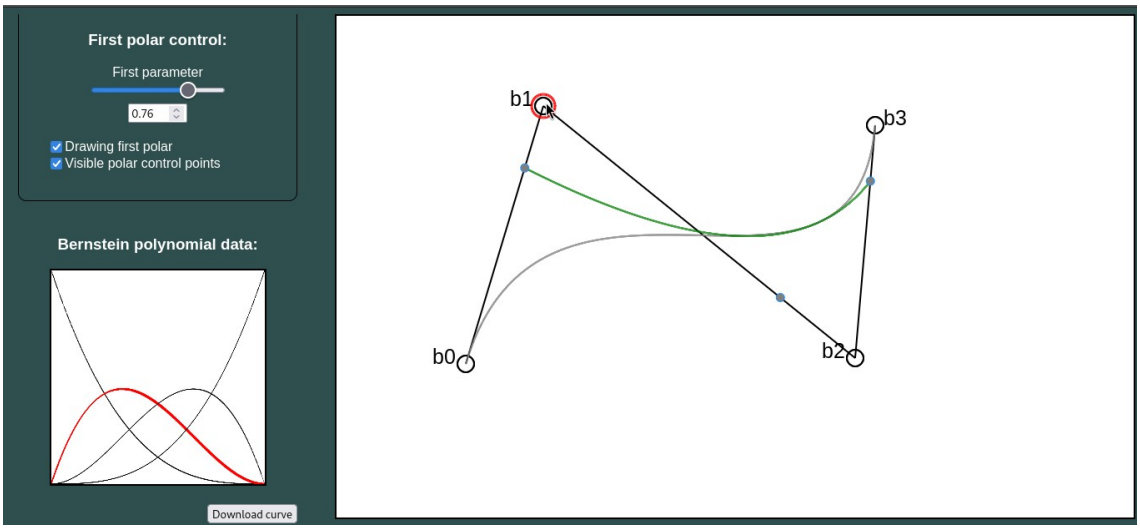
Има отделна функция, която изчисля веднъж алгоритъма на de Casteljau и връща точките { $b_0^1(t_1)$, ... $b_{n-1}^1(t_1)$ }, които се изобразяват на контролния полигон. Те могат да се разгледат като контролни точки на полярната крива.

```
if(document.getElementById("DrawPolarControlPointsCheckbox").checked){
  let polarControlPoints = PolarControlPoints(t0,controlPoints);
  for(let i = 0; i<polarControlPoints.length; ++i){

    ctx.beginPath();
    ctx.arc(polarControlPoints[i].x,
            polarControlPoints[i].y,
             5, 0, 2 * Math.PI);
    ctx.fillStyle = "gray";
    ctx.fill();
    ctx.lineWidth = 1;
    ctx.strokeStyle = "DodgerBlue";
    ctx.stroke();
  }
```

```
function drawFirstPolar(t0, drawControlPointOfPolar){

    if(t0 < 0 || t0 > 1){
        DebugError("Polar t0 is out of [0,1] bound");
        return;
    }

    if(controlPoints.length <=2){
        return;
    }

    for(let i =0;i<=1;i+=0.001){
        let pos =calculateAtPosFirstPolar(t0,i,controlPoints, drawControlPointOfPolar,5,i)

        ctx.beginPath();
        ctx.moveTo(pos.x, pos.y);

        pos =calculateAtPosFirstPolar(t0,i+0.001,controlPoints,false);

        ctx.lineTo(pos.x, pos.y);
        ctx.lineWidth = 2;
        ctx.strokeStyle = "green";
        ctx.stroke();
    }
```

5. Снимки на проекта





6. Краен продукт

Крайният продукт е един HTML файл, който съдържа всичко необходимо за функционирането на проекта без нуждата от интернет и може да бъде пуснат през стандартните браузъри, които поддържат функционалността Canvas.

Работеща версия на проекта е качена на сайта: eguardian.dev/CAGD

Кода е също качен в Github.

7. Използвани ресурси

- Записки по курса Компютърно геометрично моделиране (CAGD) на лектор доц. Красимира Влъчкова, София, 2024 г.
- Mozilla’s Javascript canvas tutorial/documentation - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial

Допълнителни връзки

- https://www.eguardian.dev/CAGD/
- GitHub https://github.com/eGuardianDev/University-Work/tree/main/(%202%20)%20second%20Year/(%203%20)%20%20Third%20semester/CAGD/