Софтуер за рисуване на поляра на крива на Безие

Описание на проекта

Проект представлява програма на Python, в която чрез използване на биболитеките matplotlib и NumPy се рисува поляра на крива на Безие.

Функционалости

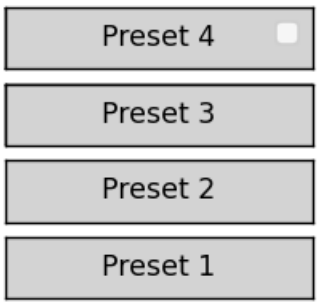
1. Класът DraggablePoint се грижи за местенето на точките и рисуването и прерисуването на кривана на Безие и полярата след преместването ѝ.
2. Функцията de_casteljau е реализация на алгоритъма на De Casteljau за оценка на точка по крива, дефинирана от контролни точки, при дадено стойност на параметъра t. Ето как работи функцията:
    a. Приема два аргумента: списък от контролни точки control_points и параметър t.
    b. Проверява дали има само една контролна точка (т.е. n = 0). Ако това е така, връща тази една контролна точка.
    c. В противен случай, функцията създава нов списък intermediate_points, в който събира новите промежуточни точки, които се изчисляват чрез линейна интерполация между всеки две съседни контролни точки.
    d. Рекурсивно извиква себе си, подавайки intermediate_points като нов списък от контролни точки.
    e. Процесът продължава, докато се достигне случаят с една контролна точка.
    f. В крайния случай (когато n = 0) функцията връща последната оценена контролна точка.
3. Функцията bezier_curve е по-високо ниво от de_casteljau. Тя използва функцията de_casteljau за да оцени точките на кривата Безие при дадените стойности на параметъра t. Ето как работи:
    a. Приема два аргумента: списък от контролни точки control_points и списък от стойности на параметъра t_values.t_values са floating points numbers в интервала между 0 и 1 (дефинирани са на ред 190)
    b. Създава празен списък curve_points, в който ще се запишат оценките на кривата за всяка стойност на параметъра t.
    c. Итерира през всеки елемент от списъка t_values.
    d. За всяка стойност на t извиква функцията de_casteljau, за да оцени точката на кривата при тази стойност на параметъра t.
    e. Добавя оценената точка към списъка curve_points.
    f. Връща списъкът curve_points като NumPy масив, който съдържа оценките на кривата за всички стойности на параметъра t.
4. Функцията displace_control_points , приема списък от оригинални контролни точки и параметър t, който представлява разстоянието, с което ще бъдат изместени контролните точки.
   Ето как работи:
    a. Приема два аргумента: списък от оригинални контролни точки original_points и параметър t.
    b. Създава празен списък displaced_points, в който ще се запишат изместените контролни точки.

c. Итерира през всеки индекс i в интервала от 0 до дължината на списъка original_points минус 1.
d. За всеки индекс изчислява посока между текущата контролна точка original_points[i] и следващата контролна точка original_points[i+1].
e. Изчислява изместената точка като скалира посоката с параметъра t и я добавя към оригиналната контролна точка original_points[i]
f. Добавя изместената точка към списъка displaced_points.
g. Връща списъка displaced_points като NumPy масив, съдържащ изместените контролни точки.

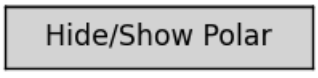
5. Функцията redraw прерисува наново кривата на безие и полярата, когато е необходимо.
6. Функцията update е се грижи за промяна на t при настъпили такива от Slider-a.
7. Функцията add_point добавя точка при клик да обособените от програмата рамки.
8. Функцията remove_point премахва последната добавена точка, докато има поне 2.
9. Функцията toggle_visibility променя видимостта та полярата.
10. Функцията set_preset пренарисува точките с избрани предварително подготвени.
11. Функцията toggle_add_point() променя стойността на променливата която отоговаря за това дали може да се добавят точки.
12. От ред 175 – 241 и се дефинират бутоните, слайдер-а и се рисуват на екрана накрая. Също така всички основни променливи като points, drawn_points, t, всички preset-ове, add_point_enabled, всички променливи свързани с рисуване – curve,control_polygon,polar,polar_pts,control_polygon и др.

Интерфейс

1. Бутони от Preset1,2,3 и 4 рисуват кривата на Безие с полярата с предварително подготвени точки, като всяка е с различен брой точки – от 2 до 5.

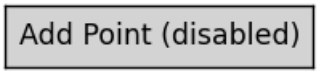


2. Бутонът Hide/Show Polar скрива или показва полярата на настоящата крива.



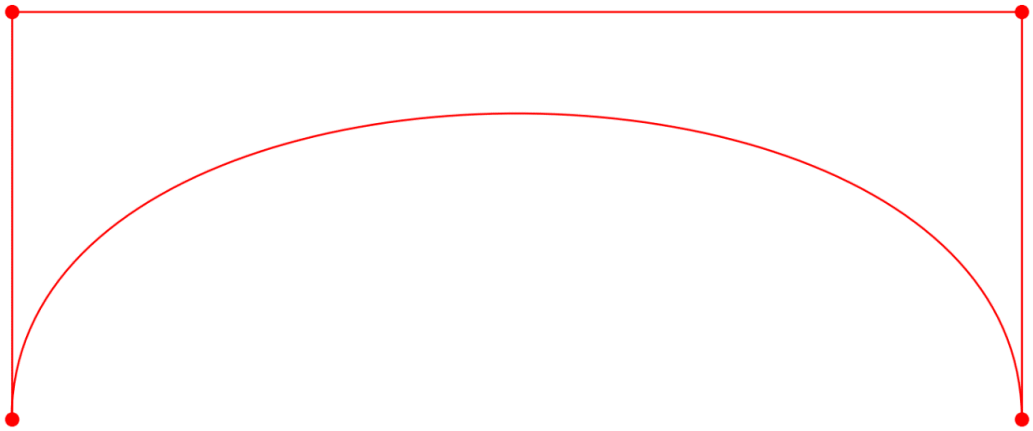
3. Бутон Remove Point премахва последната добавена точка.

4. Бутонът Add Point(disabled/enabled) позволява на потребителя да добавя точки. Когато е enabled е позволено да се добавят точки в рамките на осите, а съответно когато е disabled не може.



5. Слайдърет се грижи за промените на стойност на t в интервала между 0 и 1.

6. Самата крива на безие, конролните точки и контролния полигон.(без полярата)

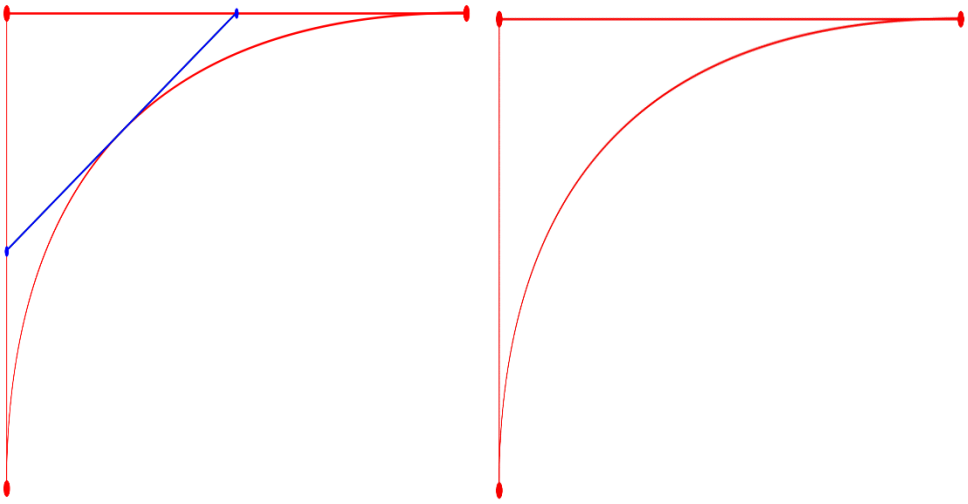


7. Допълнително всяка точка може да бъде влачена, за да промени кривата на безие и съответната и поляра.

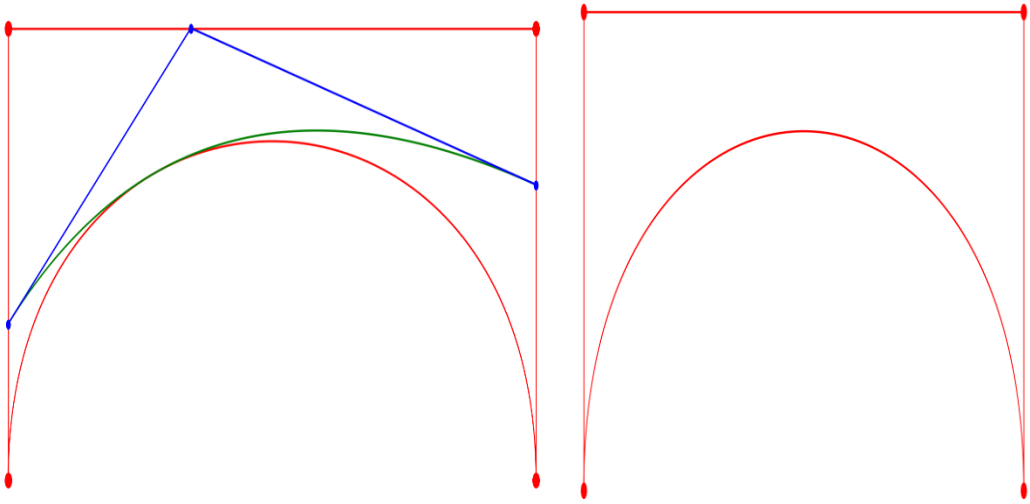
Примери

1. Пример с 2 точки

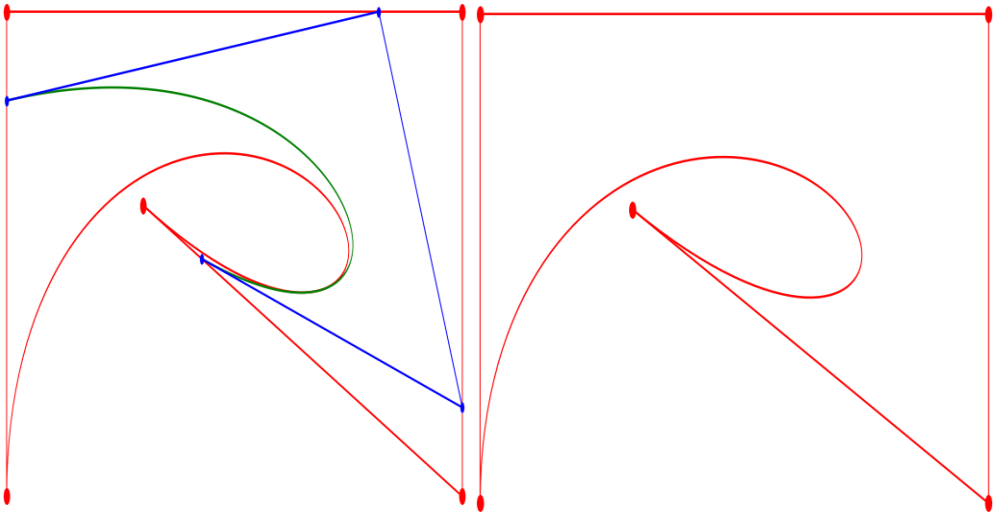
2. Пример с 3 точки с и без показана поляра



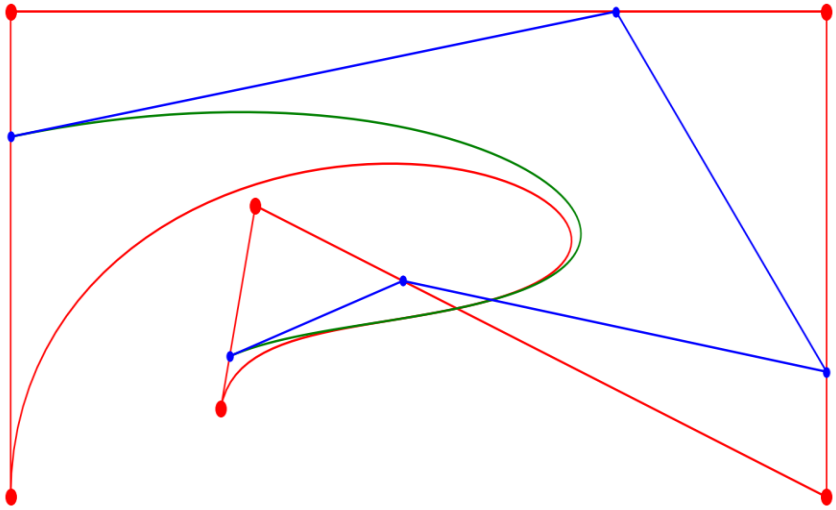
3. Пример 4 точки с и без поляра

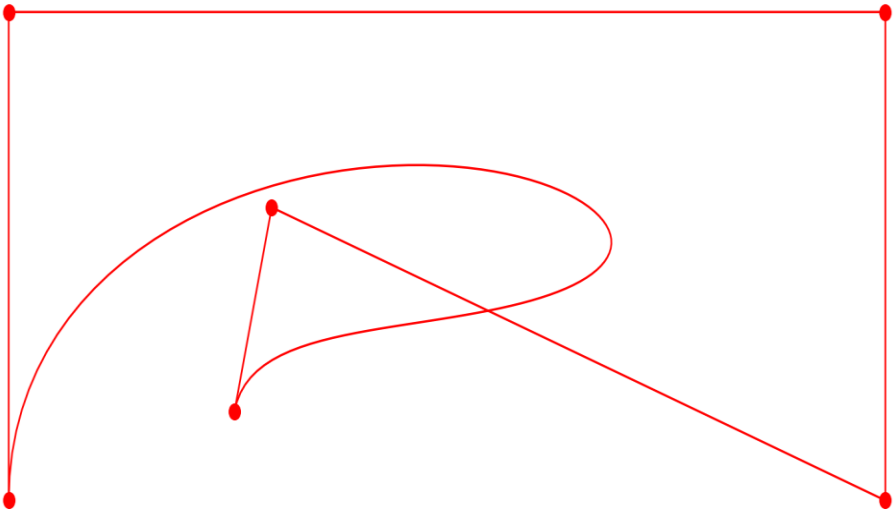


4. Пример с 5 точки с и без поляра

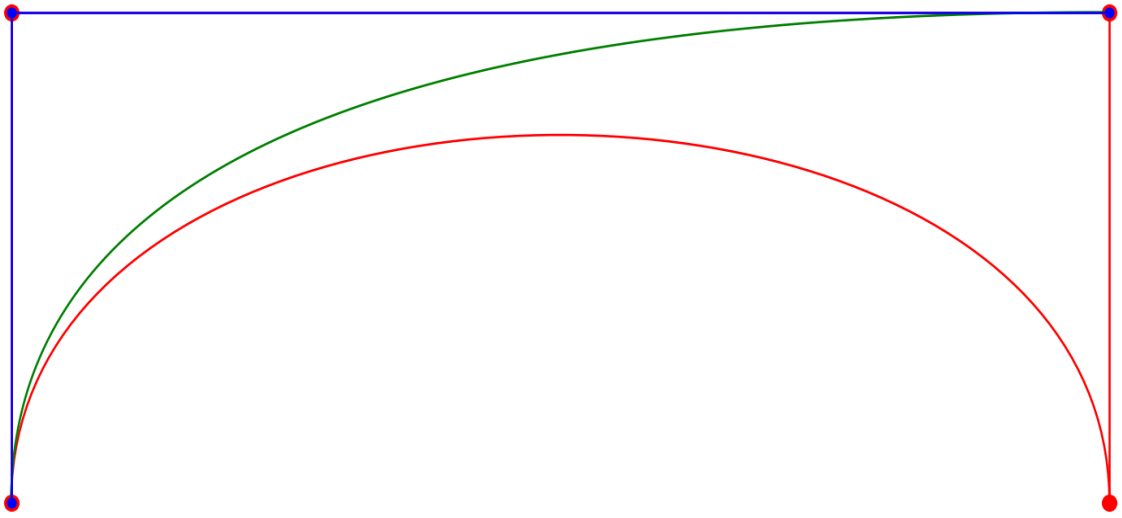


5. Пример с 6 точки с и без поляра

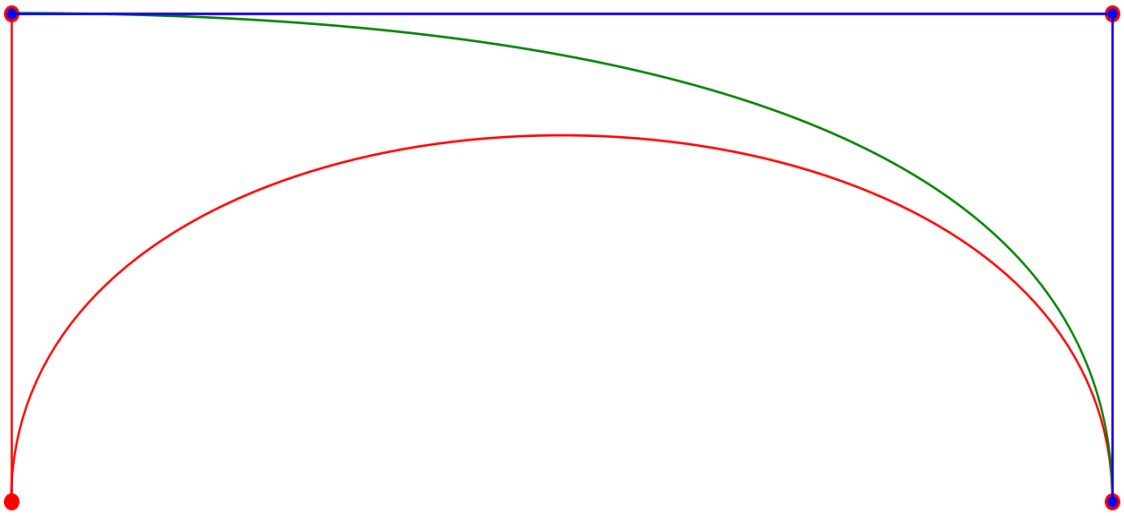




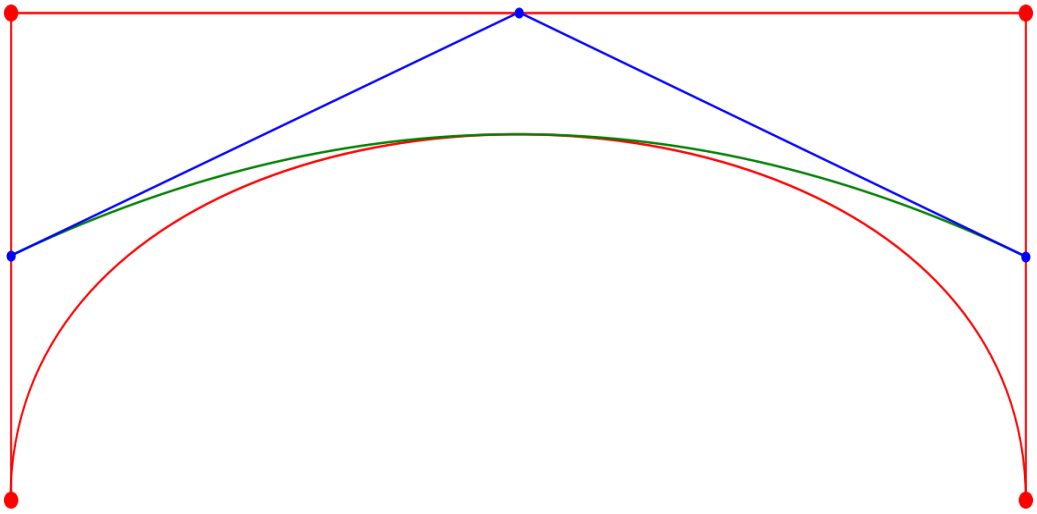
6. Пример с поляра със стойнст на t=0



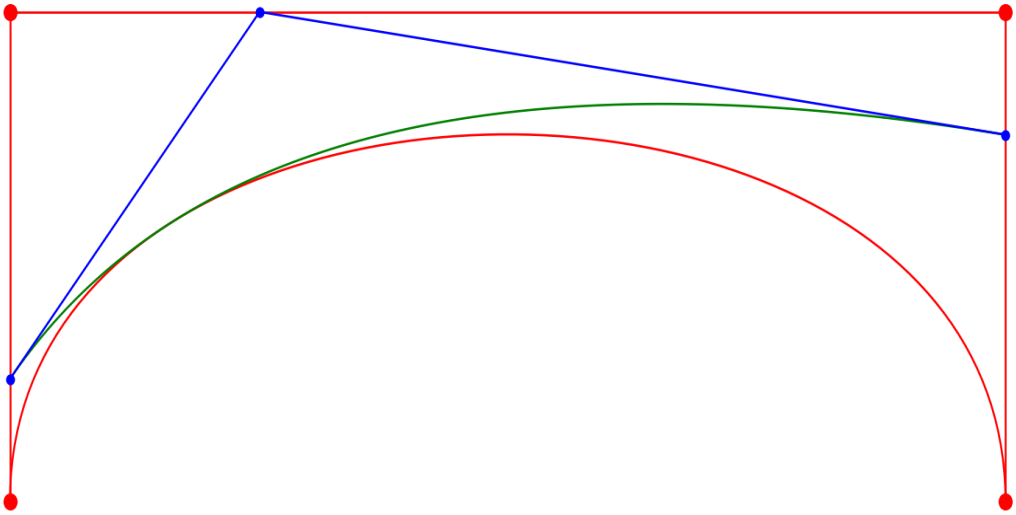
7. Пример с поляра със стойнсто t=1



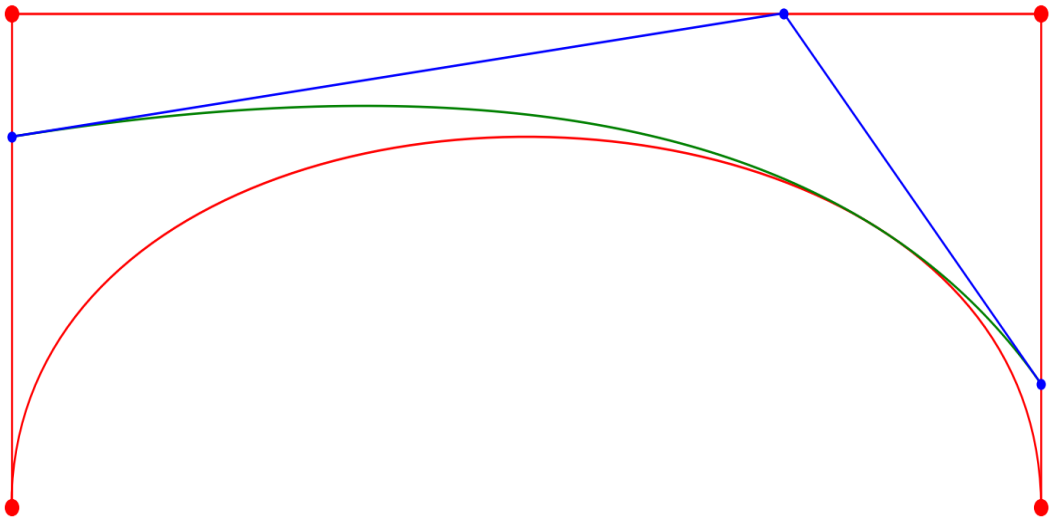
8. Пример с поляра със стойнсто t=0.5



9. Пример с поляра със стойност t=0.25



10. Пример с поляра със стойност t=0.75



Използвани математически алгоритми

- Алгоритъм на Де Кастелжо

Нека $n \in \mathbb{N}$, $\mathbf{b}_0$, $\mathbf{b}_1$, $\ldots$, $\mathbf{b}_n$ са $n+1$ различни точки в $\mathbb{R}^3$ и $t \in [0,1]$. Алгоритъмът на de Casteljau използва последователни линейни интерполации и след n стъпки построява точка $\mathbf{b}_0^n(t)$ върху полиномиална крива $\mathcal{B}$ от степен n. Кривата $\mathcal{B}$ се нарича **крива на Bézier**. Точките $\mathbf{b}_i$, $i = 0, \ldots, n$, се наричат **контролни точки** или **точки на Bézier**. Полигонът с върхове $\mathbf{b}_0, \ldots, \mathbf{b}_n$ се нарича **контролен полигон** или **полигон на Bézier** на кривата $\mathcal{B}$.

Алгоритъмът е следния:

Algorithm 1 *de Casteljau*

Вход: $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^3$, $t \in \mathbb{R}$

$$\begin{aligned} &\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t), \\ &r = 1, \ldots, n; \;\; i = 0, \ldots, n-r \\ &\mathbf{b}_i^0 = \mathbf{b}_i \end{aligned} \tag{9}$$

Изход: $\mathbf{b}_0^n(t)$ е точка от кривата $\mathcal{B}$, съответстваща на параметъра t.

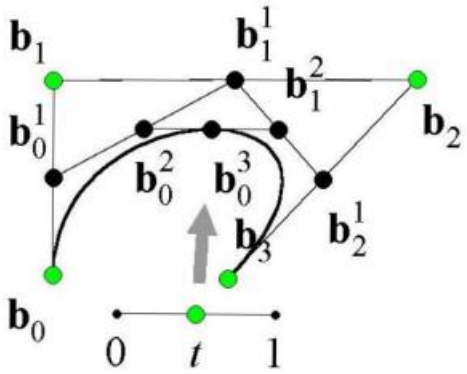
- Схема на алгоритъма

$(0,0)$

$(0,2)$ $(0,1)$

$(8,2)$ $(4,2)$ $(2,3/2)$

$(4,0)$ $(6,1)$ $(5,3/2)$ $(7/2,3/2)$

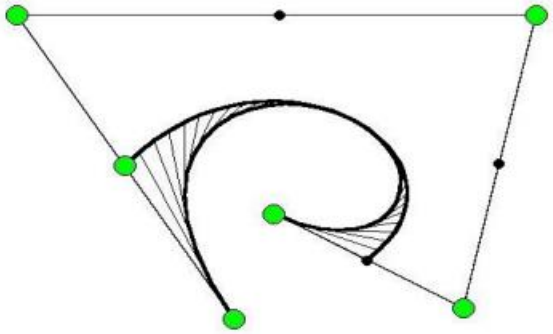


Фигура 18: Алгоритъм на de Casteljau за $n = 3$.

- Поляра на крива на Безие

  Нека $\mathbf{b}_0, \ldots, \mathbf{b}_n$ са контролните точки за кривата на Bézier $\mathbf{b}(t)$ от степен n. Прилагаме един път алгоритъма на de Casteljau за стойност на параметъра t_1. Получаваме точките $\mathbf{b}_0^1(t_1), \ldots, \mathbf{b}_{n-1}^1(t_1)$, които може да разглеждаме като контролен полигон на крива $p_1(t)$ от степен $n-1 \Rightarrow$

$$\mathbf{p}_1(t) = \mathbf{b}(t_1, t^{<n-1>}). \tag{24}$$



Фигура 25: Крива на Bézier за $n = 4$ и първата ѝ поляра за $t_1 = 1/2$

Окончателно получихме

$$\mathbf{p}_1(t) = \mathbf{b}(t) + \frac{t_1 - t}{n}\frac{d}{dt}\mathbf{b}(t).$$

Полиномиалната крива $\mathbf{p}_1(t)$ от степен $n - 1$ се нарича **първа поляра** на кривата $\mathbf{b}(t)$ относно параметъра t_1.

Използвани технологии

- Python
- Numpy
- Matplotlib

Източници

https://learn.fmi.uni-sofia.bg/pluginfile.php/458081/mod_resource/content/5/lecture_notes_CAGD.pdf