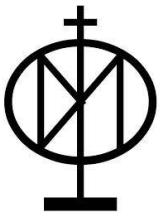
Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика



Курс по CAGD, ФМИ, СУ „Св. Кл. Охридски“,

Лектор: доц. Красимира Влъчкова

Зимен семестър, 2022/2023 год.

Тема: Реализация на Cross plot на крива на Bezier

Курсов проект

Автор: Атанас Иванов Ников

Специалност: Софтуерно инженерство, II курс

Февруари, 2023

София

Съдържание

1. Задача

Да се реализира посредством алгоритъма на de Casteljau програма за изчертаване на криви на Bezier.

Да се имплементира алгоритъм за Cross plot на дадена крива.

Да се моделира подходящ потребителски интерфейс, позволяващ лесно изчертаване на крива и нейния Cross plot.

2. Техническо описание на функции

Програмата в основата си представлява уеб приложение, реализирано с помощта на javascript, HTML5, CSS3, посредством библиотеката CanvasRenderingContext2D.

- getPointWithDeCasteljau

  Функция, реализираща алгоритъма на de Casteljau итеративно. Приема два аргумента – число **t**, за което е изпълнено: t ∈ R, t ∈ [0, 1], и масив **controlPoints** от двойки стойности (точки с атрибути реални числа x и y), който представлява съвкупността от текущите начертани контролни точки. Връщаният резултат е единствена точка (със същите атрибути).

  Алгоритъмът на de Casteljau:

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

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t),$$
$$r = 1, \ldots, n;\ i = 0, \ldots, n-r$$
$$\mathbf{b}_i^0 = \mathbf{b}_i$$

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

В тялото на функцията първоначално се проверява за броя налични контролни точки във вектора, като за да се осъществи калкулация, са необходими поне две точки. След което се пристъпва към итеративната основа на тази функция.

Нека $n \in N$, $b_0, b_1, \ldots, b_n$ са $n + 1$ различни точки в R^2 . Тъй като алгоритъмът на de Casteljau след n стъпки връща като резултат една единствена точка b_0^n, определяме условие за изпълнение на итерации на алгоритъма докато резултатът му не е само една точка. В междинните изчисления броят точки надхвърля 1, като при първата итерация резултантните точки ще бъдат n - 1 на брой, на втората – n – 2, и т.н. Това е така, защото се изчислява междинна точка за всяка двойка последователни контролни точки, като на всяка следваща итерация текущите точки се заместват с междинните им, докато не се стигне до изхода: междинната точка на последните две точки. След като имаме цикъл с дефинирано условие за край, преминаваме към същинската част – пресмятане на междинните точки.

Текущите точки, с които разполагаме, ще назоваваме „текущи точки“, а междинните, които на всяка следваща итерация заемат мястото на текущите, ще наричаме „междинни точки“.

Алокира се нов празен масив, който ще бъде контейнер за новополучените междинни точки. Нека j е поредният номер на стъпката, а i е поредният номер на точката в масива с текущи точки и $i \in N, j \in N$.

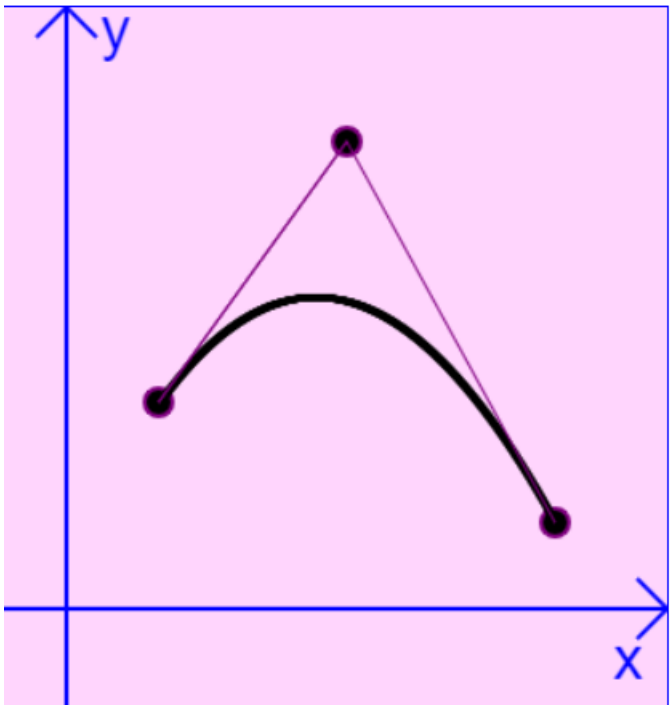
За всяка двойка последователни точки (b_{i-1}^j, b_i^j) изчисляваме координатите на тяхната междинна точка и я добавяме в контейнера за междинни точки. На края на всяка итерация присвояваме контейнера с текущите точки на контейнера с получените междинни. Следва нова итерация.

$$
\begin{array}{cccc}
\mathbf{b}_0 & & & \\
\mathbf{b}_1 & \mathbf{b}_0^1 & & \\
\mathbf{b}_2 & \mathbf{b}_1^1 & \mathbf{b}_0^2 & \\
\mathbf{b}_3 & \mathbf{b}_2^1 & \mathbf{b}_1^2 & \mathbf{b}_0^3
\end{array}
$$

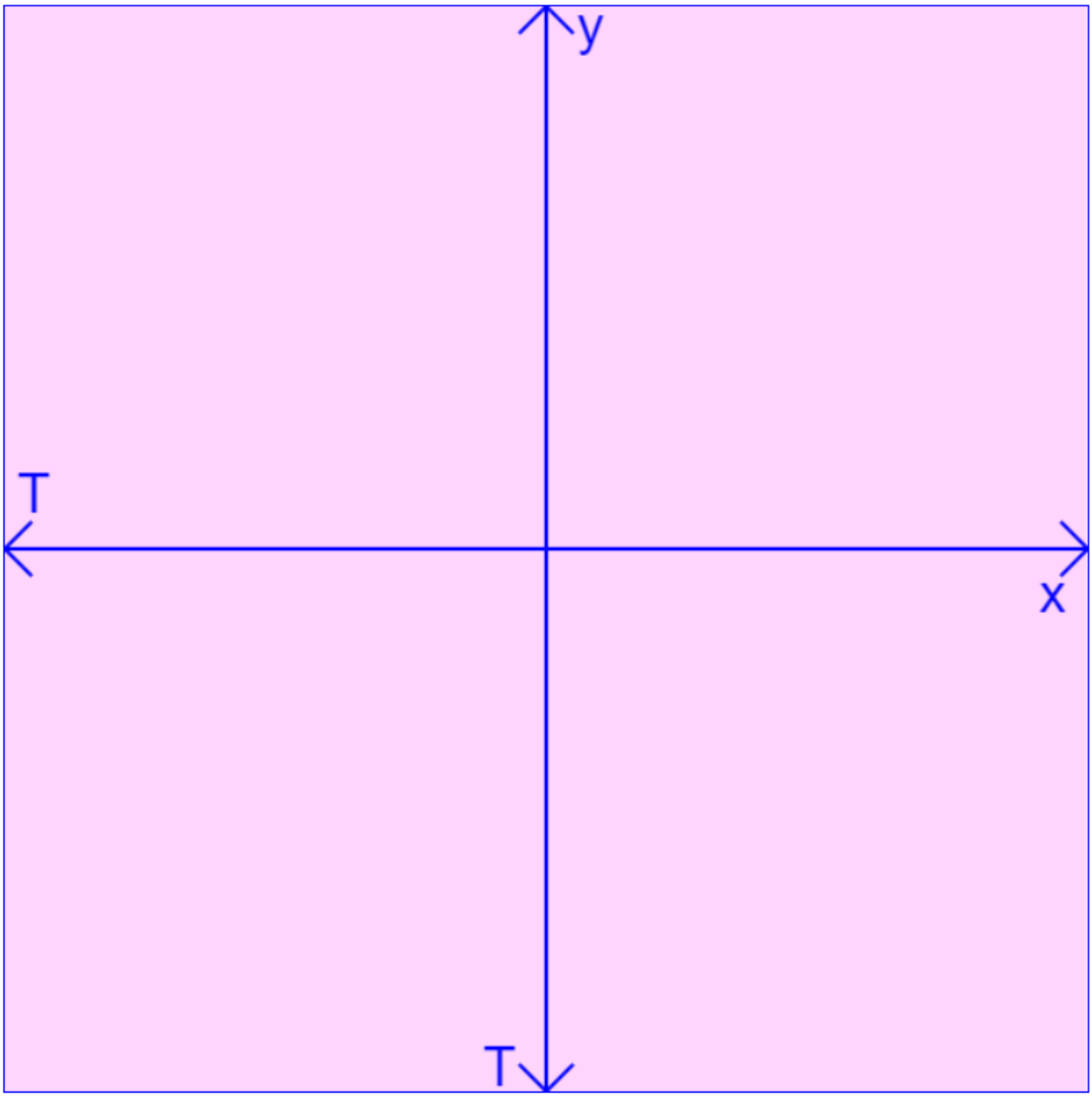
По този начин достигаме до крайния резултат – контролна точка, която ще бъде част от нашата крива на Bezier.

- drawCurveWithParams
  Функция, която изчертава контролните точки и кривата на Bezier. Приема аргументи **from, to, points, pointStroke, color**. Последните два служат за оцветяване съответно на точките и на кривата. Аргументът **points** е масив с двойки стойности (точки с атрибути реални числа x и y). Аргументите **from** и **to** са горната и долната граница на дефиниционната област на дадено число t, за което : $t \in R$, $t \in [0, 1]$. В реализацията на програмата тези граници са тъкмо числата 0 и 1.
  Първоначално се изчертават самите контролни точки (посредством чертане на запълнени кръгове).
  След което се преминава към итеративно изчисляване на контролните точки за всяко t в допустимите граници през интервал от 10^{-3}. Интервалът е малък за по-добра прецизност и съответно за изчертаване на по-гладка крива (тъй като включва изчисляването на повече точки от кривата). На всяка итерация изчисляваме една точка от кривата на Bezier чрез гореспоменатия алгоритъм на de Casteljau и я изчертаваме. Изчертаването на точката в контекста на използваната библиотека за графично представяне представлява изчертаване на правоъгълник, чиято стартова точка е конкретната получена точка, а дължините му са по 1 пиксел.

Резултатът е получената крива и контролните точки, като обвиваща функция изчертава и връзките между тях под формата на отсечки.



- drawCartesianCoordinateSystem
  Процедура, която определя размерите на платното за чертане и спрямо тях начертава осите Декартовата координатна система (двуизмерна). В допълнение се начертават и стрелките.

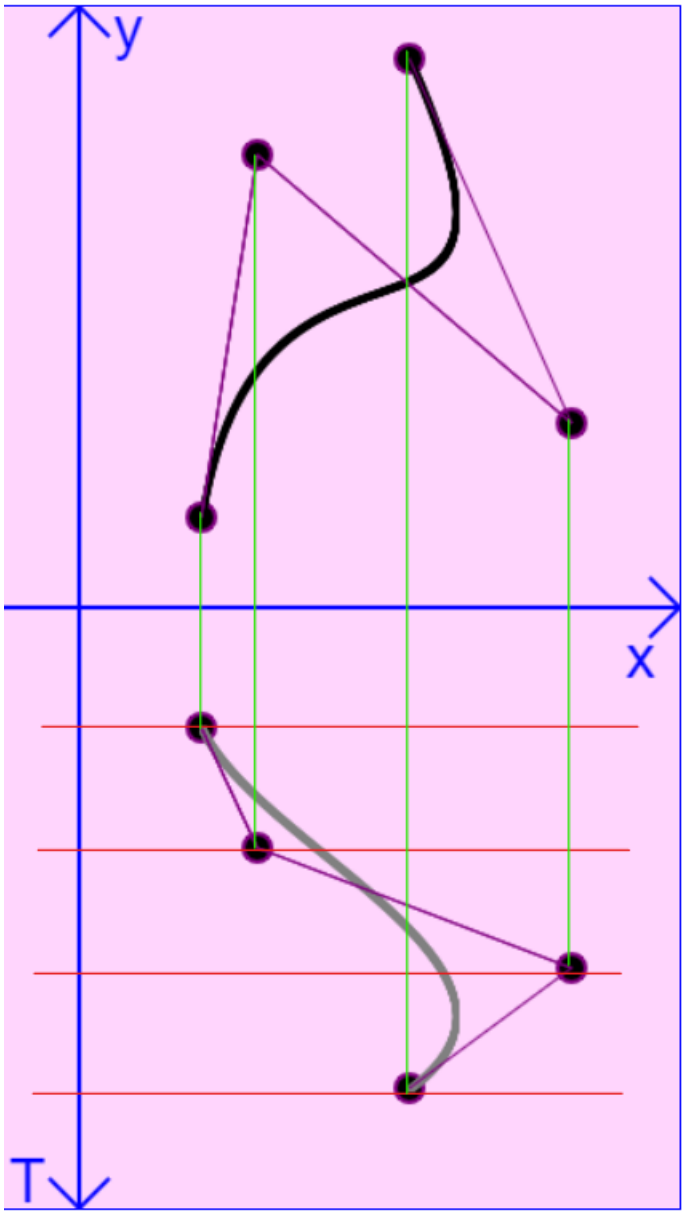


- calculateFourthQuadrantPoints
  Процедура, която изчислява координатите на точките на функционалната крива x(t). Областта на действие е IV квадрант на координатната система. За да се достъпи той, се изчислява най-напред височината, на която се намира правата y = 0 спрямо y на платното за чертане. Съответно, тя е границата между I и IV квадрант. След което, за да имаме мярка какви ще бъдат стойностите за y на точките на функционалната крива x(t), взимаме броя текущи начертани

контролни точки в 1-ви квадрант, добавяйки 1 към числото. Целта е да разделим квадранта (по отношение на y) на броя начертани контролни точки + 1 равни части. След което разделяме дължината на квадранта на гореспоменатия делител и получаваме нужния коефициент, чието свойство е да разделя не само точките на равни разстояния една от друга, а и правата y = 0 от първата точка в IV квадрант, както и последната точка от долната граница на квадранта на платното. По този начин се постига равномерно разпределение на точките спрямо y. На фигурата долу е посочено разпределението на интервалите между точките. (4 точки образуват в квадранта 5 равни интервала). **За всяка точка от IV квадрант x-координатата е равна на x-координатата на съответната по ред контролна точка от I квадрант. Изчисляването на y-координатата на всяка точка става чрез следната формула:**

$$\boldsymbol{y = l + ic,}$$

Където l е дължината на квадранта спрямо оста y, $i \in \mathrm{N}$, i е мултипликатор за коефициента на всяка стъпка, а c е коефициентът на отместване, равен на $\frac{l}{\text{броя контролни точки}+1}$.



- calculateSecondQuadrantPoints
  Напълно аналогично на процедурата, която връща точките в IV квадрант, тази процедура служи за изчисляване на координатите на точките на функционалната крива y(t) от II квадрант. Тук първоначално се изчислява широчината на този квадрант, която съвпада с местоположението (спрямо x) на правата x = 0. Тя е и границата между I и II квадрант в координатната система. След което със същия подход се пресмята броят интервали, на които ще бъде разделен II

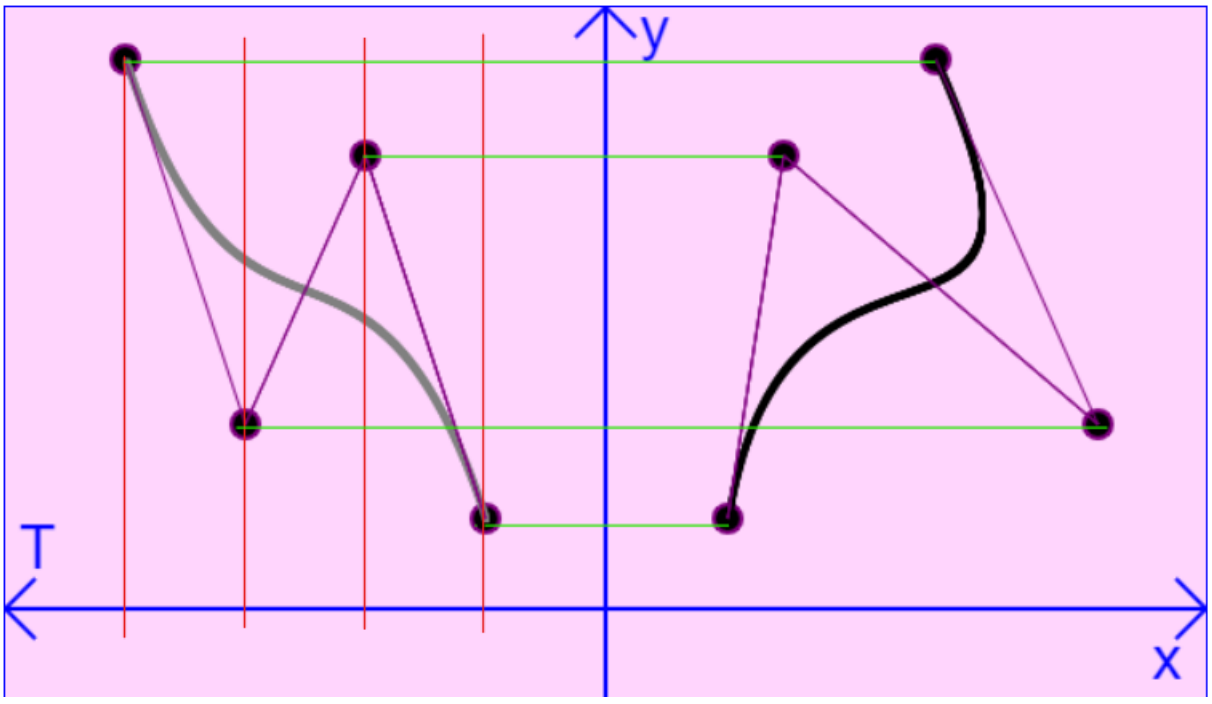
квадрант в широчина, който е равен на броя контролни точки + 1, а след него – коефициентът на отместване, равен на $\frac{\text{широчината на квадранта}}{\text{броя интервали}}$. **За всяка точка от II квадрант y-координатата е равна на y-координатата на съответната по ред контролна точка от I квадрант. За да изчислим стойността на x-координатата на текущата точка от II квадрант, правим следното: нека широчината на квадранта е *l*, а коефициентът на отместване е c.**

Получаваме следната формула:

$$x = l - ic,$$

където $i \in N$, i е мултипликатор за коефициента на всяка стъпка. Формулата е напълно идентична по логика на предната, отчитайки, че знакът пред i зависи от визуалната репрезентация на алгоритъма и използваните библиотеки, ресурси и т.н.

Отново получаваме равномерно разпределени точки, този път във II квадрант, като за n точки има n + 1 интервали, $n \in N$.

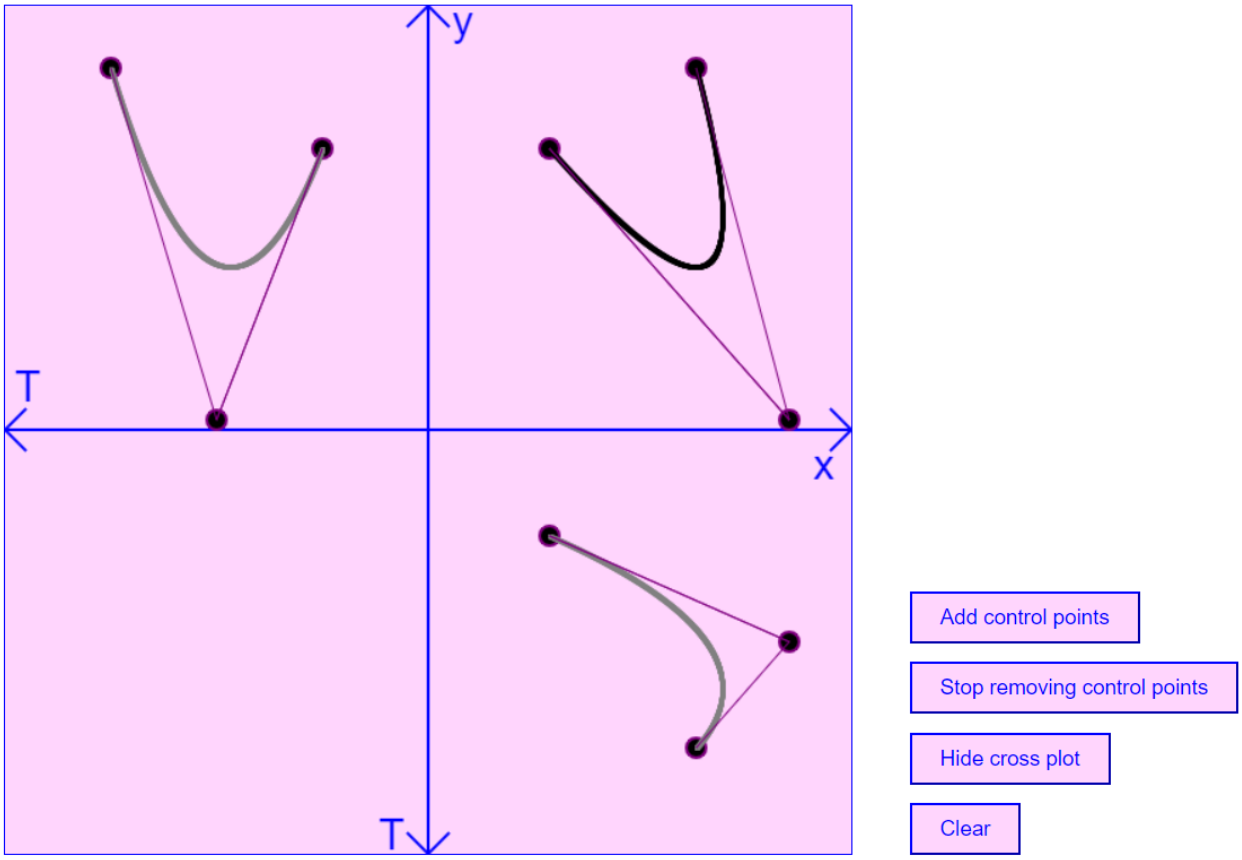


- isPointSelected
  Функция, която проверява дали потребителят е селектирал някоя от съществуващите в първи квадрант контролни точки чрез клик на мишката. Вземат се координатите за всеки клик и се изчислява дали този клик е в радиуса на някой от начертаните кръгове, които репрезентират контролните точки. Съответно параметрите са точката с координати на клика на мишката на потребителя, контролната точка, за която проверяваме, и нейният радиус на обхват.
  Функцията връща булев резултат (‚да‘/‚не‘). Чрез формулата за Евклидови разстояния, където x_1 и y_1 са координатите на едната точка, а x_2 и y_2 – координатите на другата, се пресмята разстоянието, и ако то е по-малко или равно на радиуса на кръга, функцията връща положителен отговор, в противен случай – отрицателен.

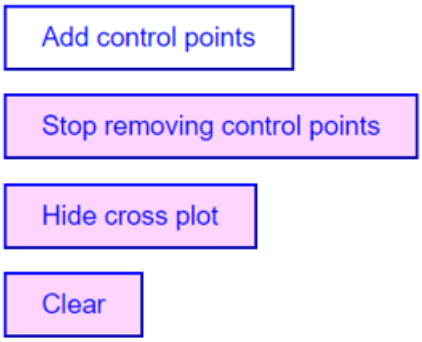
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

  Функцията се използва когато потребителят иска да премести дадена точка, или да я изтрие, понеже и двете операции изискват локализиране на точката с помощта на мишката.
  Забележка: поради тази причина, за удобство на потребителя, за да са лесни за локализиране, точките са с голям радиус (7 пиксела).

3. Потребителски интерфейс

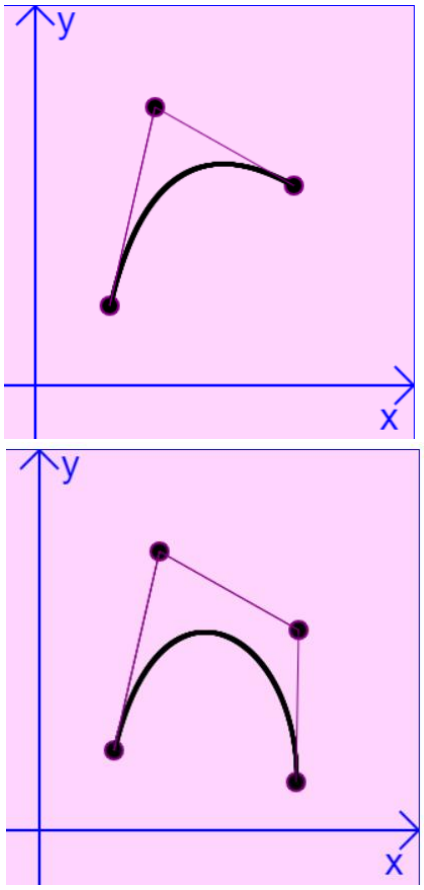


Уеб приложението представлява платно за чертане с четири бутона, с които можем да влияем на изпълнението на програмата.



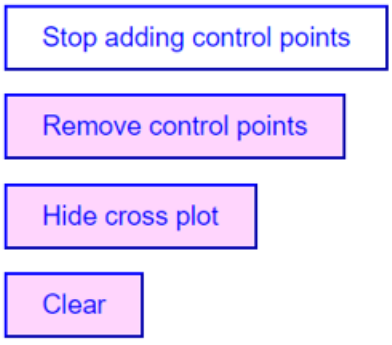
- Бутонът ‚Add control points‘ позволява на потребителя да започне да добавя контролни точки на платното (в I квадрант на координатната система). При единично кликане върху бутона потребителят може да добавя произволен брой точки. Докато потребителят добавя точки, за всяка добавена точка

динамично се преизчислява на база на всички контролни точки нова крива на Bezier, както и отсечките, очертаващи полигона, който образуват контролните точки.

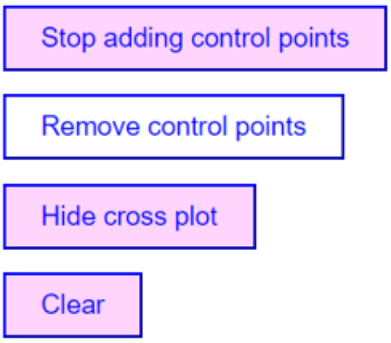


Докато тази функция е активна, означението на бутона се сменя на ‘Stop adding control points’. Съответно, при повторен клик върху бутона, се прекратява състоянието на добавяне на точки и се запазват текущите.

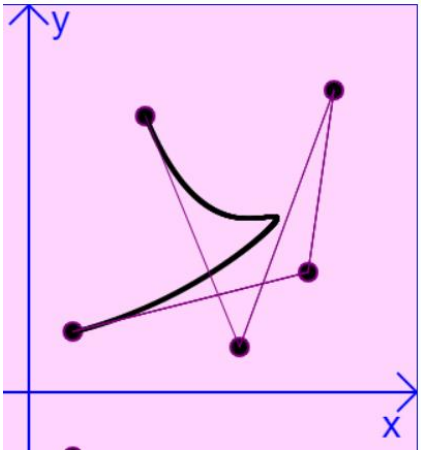
Забележка: Поради спецификите на Cross plot, потребителят може да добавя точки само в I квадрант.

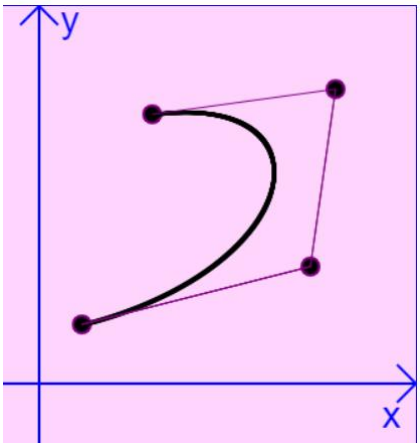


- Следващият бутон е бутонът ‘Remove control points’, който служи за премахване на вече съществуващи контролни точки в I квадрант.

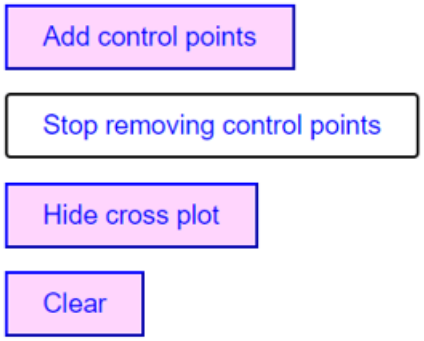


При единично кликане върху бутона се активира състоянието на премахване на контролни точки, като единственото, което потребителят трябва да направи, е да локализира точката, която желае да изтрие, и да кликне върху нея (където и да е върху кръгчето). В резултат на това действие точката се премахва от платното и отново се преизчислява нов полигон и нова крива на Bezier.





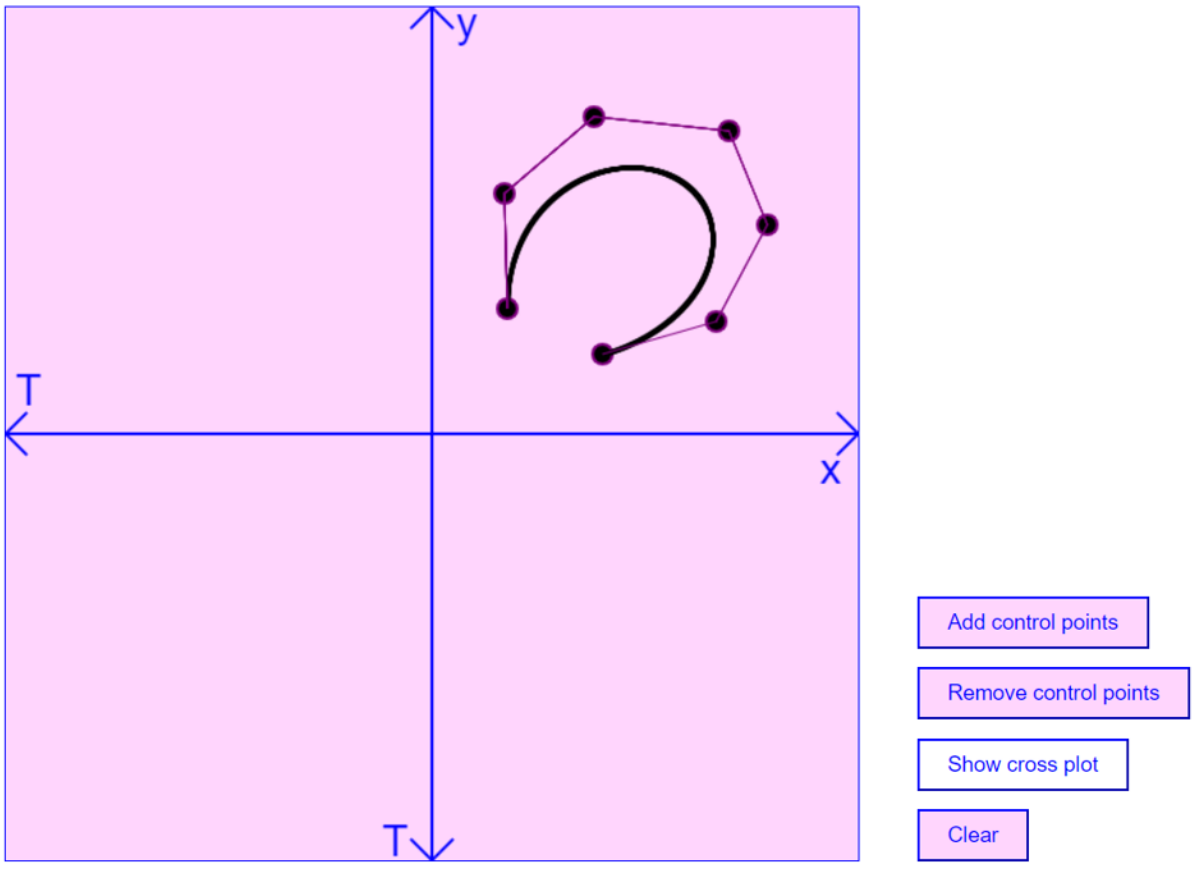
Докато е активно състоянието на премахване на точки, означението на бутона се променя на ‘Stop removing control points’, като съответно при втори клик върху бутона прекратяваме изтриването на точки.



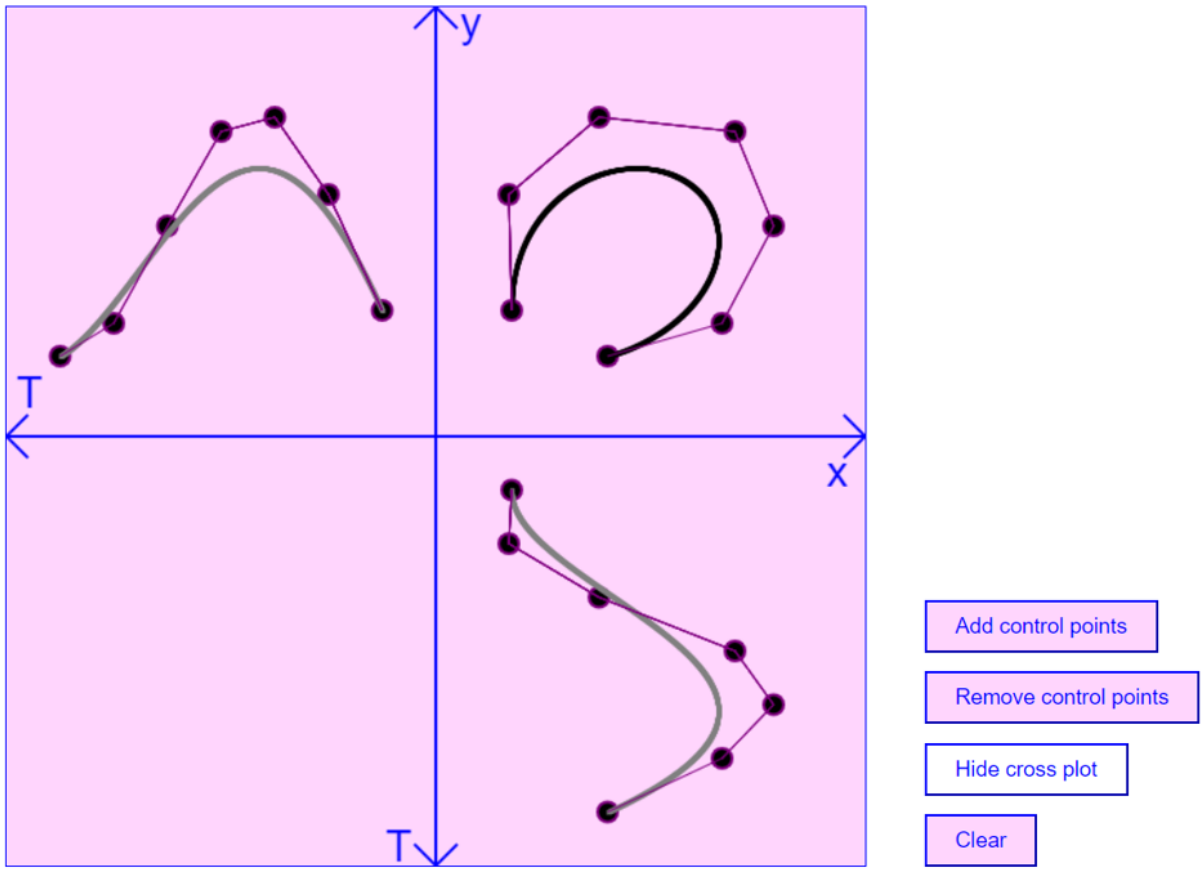
Забележка: ‘Add control points’ и ‘Remove control points’ не могат да работят едновременно.

- Преместване на точки
  Преместването на точка може да се осъществи само когато ‘Add control points’ и ‘Remove control points’ не са активни, тъй като всяко от тези действия изисква клик върху платното и води до различен резултат. Потребителят локализира точката, която желае да премести (отново в рамките на I квадрант) и я придвижва при натиснат ляв бутон на мишката. Когато е достигната желаната локация, левият бутон се освобождава и точката запазва новата си позиция, като отново се преизчисляват полигонът и кривата на Bezier.
- Показване/скриване на x(t) и y(t).

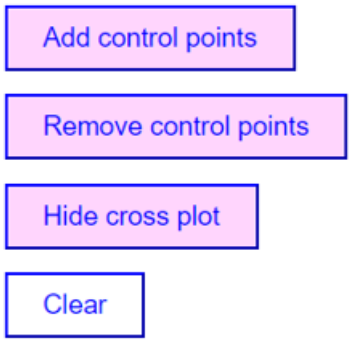
Бутонът ‘Show cross plot’ при единично кликане показва функционалните криви x(t) и y(t). В този момент се изчисляват техните контролни точки, полигони, криви на Bezier, и графиките им се появяват съответно на IV и на II квадрант от координатната система на платното. При добавяне на контролна точка, премахване на контролна точка, или преместване на контролна точка от I квадрант, промените върху графиките се отразяват динамично и върху функционалните криви x(t) и y(t).



След като върху бутона е приложен единичен клик с мишката, неговото означение се променя на ‘Hide cross plot’ и при повторен клик върху бутона графиките от IV и II квадрант отново се премахват от платното.



- Изчистване на платното
  Последният бутон ‘Clear’ служи за премахване на начертаните обекти от платното. Активира се с единичен клик.



4. Източници

Литература: Записки и приложения от лекциите към курса „Компютърно геометрично моделиране”.
Софтуерна документация и среди за разработка:

https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D

https://code.visualstudio.com/