Herramienta Interactiva en Python
Curvas de Bezier
Métodos Numéricos

Dan Heli Muñiz — Lic. en Computación Matemática
Jafet Castañeda — 6 de octubre de 2023

---

1. Introducción

Para este proyecto decidimos trabajar sobre el tema de *Ajuste de Curvas e Interpolación* , específicamente sobre la familia de Curvas de *Bézier*. Lo anterior se debe a que representan un método numérico interesante para calcular y suavizar curvas dados ciertos puntos de control.

La denominación de la curva es en honor al ingeniero francés Pierre Bézier. En primera instancia, las curvas tenían como funcionalidad el trazado de dibujos técnicos, en el diseño aeronáutico y en la industria automotriz. Estas curvas, en esencia, son una combinación lineal de "Polinomios de Bernstein", los cuales se dieron a conocer en 1912. En la actualidad, estas curvas son específicamente usadas en el diseño gráfico, donde el diseñador puede ajustar los puntos de control y observar cómo se afecta la curva en tiempo real.

2. Teoría

Existen varios métodos con los cuales podemos lograr la creación, parametrización y control de una curva de *Bézier*. El utilizado para este proyecto fue la implementación y uso de los **Polinomios de Bernstein**. Sea $\mathbf{B}(t)$ una curva de *Bézier* con los puntos $\mathbf{P}_i$, esta se puede representar

$$\mathbf{B}(t) = \sum_{i=0}^{n} \mathbf{P}_i \mathbf{b}_{i,n}(t), \quad t \in [0,1] \tag{2.1}$$

Donde

$$\mathbf{b}_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, \dots n$$

Estos últimos son los **Polinomios de Bernstein**, donde n es el numero de puntos de control, y t es un parámetro que vive en el intervalo $[0,1]$.

Propiedades Útiles

1) Las curva esta delimitada por la envoltura convexa *"Convex Hull"* de los puntos de los puntos de control.

2) Las curvas son simétricas: revertir el Orden de los puntos de control , produce la misma curva pero con una parametrización inversa.

3) Cualquier operación de traslación,escalado,rotación aplicada a los puntos de control es aplicada a la curva en si.

4) El orden de la curva es igual al numero de puntos menos uno.

3. Aplicación en Python

En esta sección, se discutirá sobre la herramienta desarrollada en el lenguaje de Python, en específico sobre su funcionamiento y la interacción entre aplicación y usuario.

El funcionamiento de nuestra herramienta de edición se basa en dos librerías implementadas por los integrantes del equipo. La primer librería cuyo nombre es **bezier_calculate.py** se consta de dos funciones para calcular los coeficientes de los polinomios de *Bernstein* y de una clase que genera los puntos de la curva Bézier.

Para poder visualizar los puntos de forma numérica, tenemos como salida un archivo de texto en el que se guardan los puntos para posteriormente graficarlos usando la librería de **matplotlib.pyplot**. Ahora, la segunda librería que desarrollamos contiene las funciones para graficar la curva a nuestro parecer.

Ahora, lo interesante de esta herramienta es que hicimos uso de una clase para mover los puntos de control en un gráfica de dispersión. Modificamos la clase **DragabbleScatter** para que en cuanto el usuario mueve un punto de control, la curva se modifique correctamente.

3.1. Interfaz

Realizando el siguiente comando en la terminal es posible obtener de manera visual nuestra herramienta,

```
python main.py
```

Al ejecutar el programa obtenemos la siguiente interfaz.

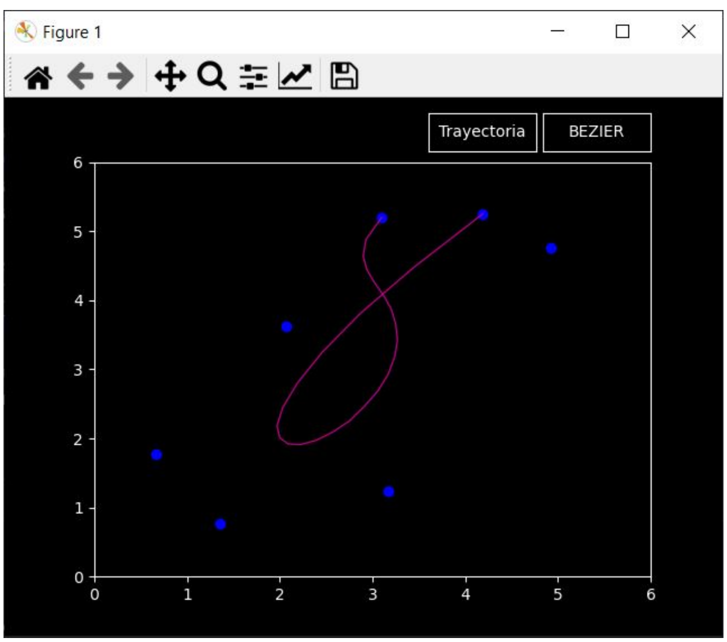


Figura 3.1: Interfaz de Herramienta

Notemos que tenemos nuestros *puntos de control* en color azul y nuestra curva de Bézier es rojo/rosado. Si el usuario selecciona alguno de los puntos (con click izquierdo) y lo arrastra, la curva se modifica con respecto al punto.

Además, contamos con dos botones, **Trayectoria** y **BEZIER**. Si seleccionamos **Trayectoria**, obtenemos el orden de los puntos de control para generar dicha curva. A continuación una visualización

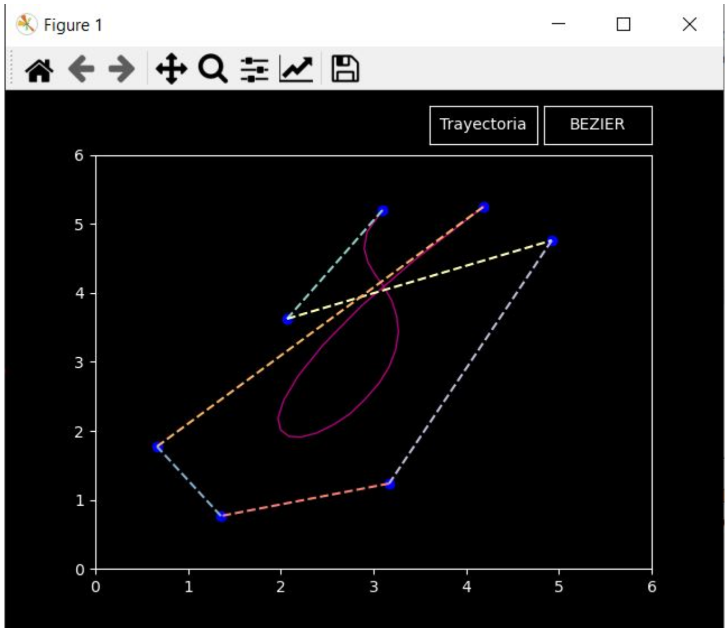


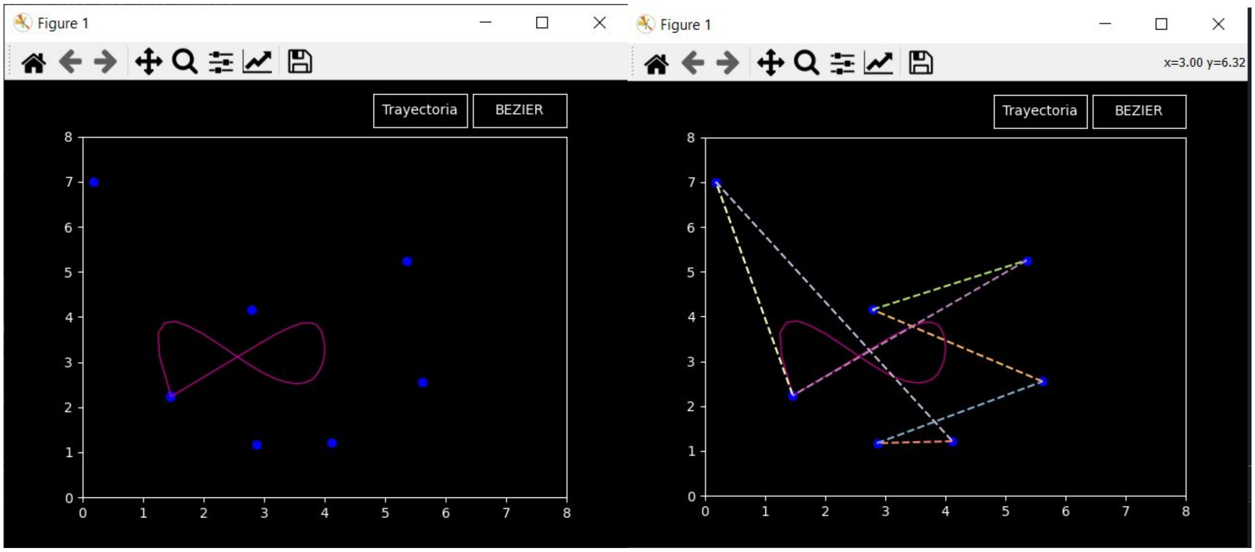
Figura 3.2: Interfaz de Herramienta

Al presionar el botón **BEZIER** obtenemos nuevamente la curva Bézier.

NOTA: Una vez que se presiona algún botón, es necesario quitar el ratón del botón para visualizar el resultado. Además si quiere agregar más puntos al graficador, se tienen que añadir o cambiar manualmente en el archivo de texto de la carpeta del proyecto.

4. Resultados

A continuación se presentan curvas editadas en dicha herramienta.



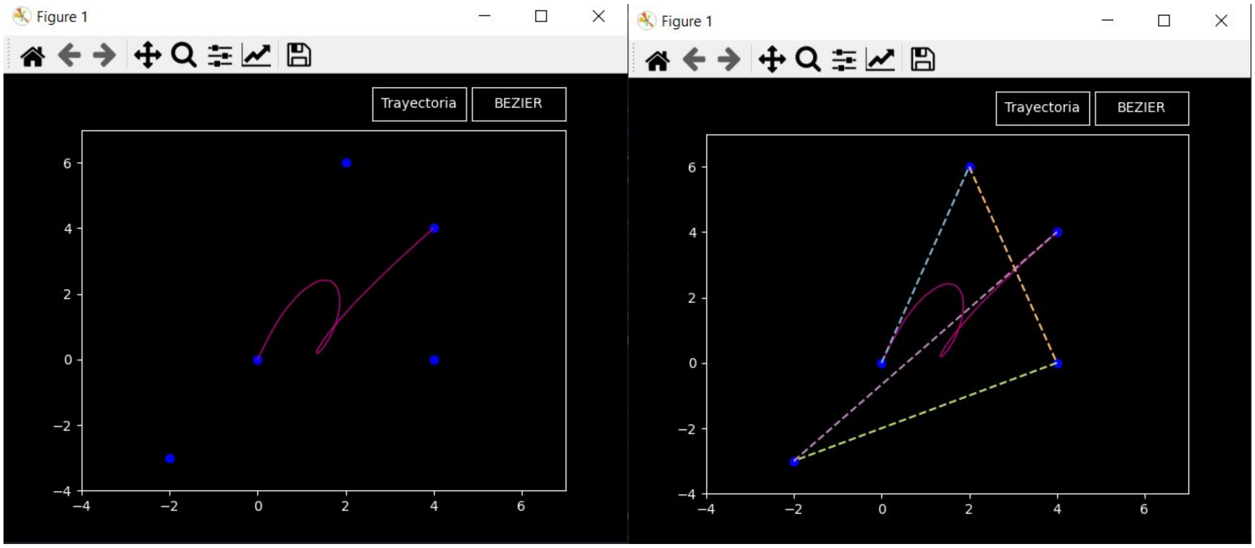
(a) Símbolo Infinito.

(b) Trayectoria.

Figura 4.1: Ejemplo 1

Se obtuvo dicha curva con los siguientes puntos de control

```
1.4566637484005653, 2.2581219403649317
0.18145161290322576,7.0
4.118279569892474,  1.2169618664993795
2.865591397849463,  1.1742502102214336
5.618560172402181,  2.549419427924102
2.78494623655914,   4.153388235608174
5.3518755471748936, 5.252012784723066
1.4566637484005653, 2.233847149735001
```

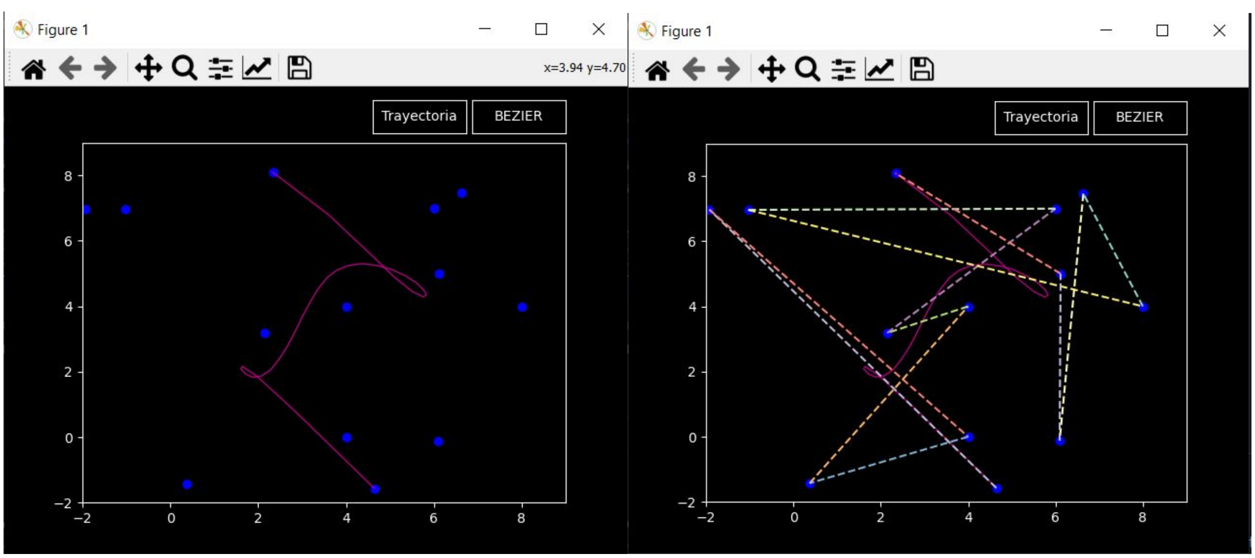


(a) (b)

Figura 4.2: Ejemplo 2

Puntos de Control (Figura 4.2) :

```
0.0 , 0.0
2,6 , 0.0
4.0 , 0.0
-2.0, -3.0
4.0 , 4.0
```



(a) (b)

Figura 4.3: Ejemplo 3

Bibliografía

- Burden, Richard; Douglas Faires. Análisis numérico. Editorial Iberoamérica, México, 1985.
- Bezier Curves - Maple Help. (n.d.). Recuperado el 20 de noviembre, 2022, de https://fr.maplesoft.com/support/help/maple/view.aspx?path=MathApps%2FBezierCurves
- Wikimedia Foundation. (2022, February 23). Curva de Bézier. Wikipedia. Recuperado el 20 de noviembre, 2022, de https://es.wikipedia.org/wiki/Curva_de_B%C3%A9zier#Curvas_c%C3%BAbicas_de_B%C3%A9zier
- ChrisChris1. (1965, 1). 'pathcollection' not iterable - creating a draggable scatter plot. Stack Overflow. Recuperado el 20 de noviembre, 2022, de https://stackoverflow.com/questions/52840767/pathcollection-not-iterable-creating-a-draggable-scatter-plot