Documentación Curvas de Bezier

Gabriel Gómez

Abril 2020

1 Introducción

En este trabajo se aplicarán los splines de Bezier para graficar la letra "G" mayúscula en un computador utilizando el lenguaje Python2.7. Primero se resumirá la teoría básica que se utilizó, luego se expondrá el código y finalmente se mostrarán los resultados junto con algunas conclusiones.

2 Teoría

Dados algunos puntos en algún espacio, casi siempre es de interés poder conectarlos de una manera continua a través de funciones conocidas. Uno de los instrumentos matemáticos que nos permiten crear curvas a partir de los que se llaman "puntos de control", son los splines de Bezier. Un spline de Bezier, para lo que nos interesa en este trabajo, es una función $\alpha(t)$ en donde $t \in [0, 1]$.

2.1 Construcción del spline

Dados $n+1$ puntos $(P_i)_{0 \leq i \leq n}$ en $\mathbf{R}^2$ se define el spline de Bezier para $t \in [0, 1]$ como:

$$\alpha(t) = \sum_{i=0}^{n} C(n, i) t^i (1-t)^{n-i} Pi \tag{1}$$

en donde $C(n, i) = \frac{n!}{(n-i!)i!}$.

2.2 Uso

Esto se puede utilizar para graficar la letra "G" en un ordenador de la siguiente manera:

Teniendo ya un boceto de la letra, se colocan algunos puntos de control para que el spline "dibuje" lo que deseamos (para mas detalle sobre cómo funcionan estos puntos de control revise las notas de clase). El spline que se definió anteriormente es una función que toma un valor de t y devuelve un punto en $\mathbf{R}^2$ que veremos de la forma (x, y).

Para nuestro objetivo, tomaremos algunos puntos de control en el boceto de la letra "G", construiremos el spline y tomaremos suficientes valores de t para que la grafica se vea continua. Además, para facilidad de la programación, se va a separar el spline en dos, una parte que tenga las componentes en "x" y la otra que tenga las componentes en "y".

3 Código

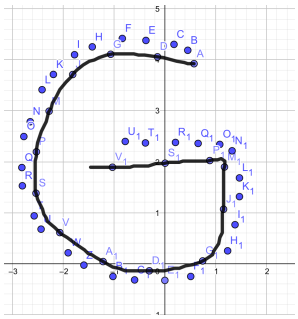
A continuación se presenta el código realizado en Python2.7 junto con explicaciones del funcionamiento en forma de comentarios.

```
#Se importa la librería pyplot que se usa para graficar
from matplotlib import pyplot as pl

#Se crea la función factorial
def factorial(n):
    if n==0:
        return 1
    else:
        return n*factorial(n-1)

#Se crea la función combinatoria
def C(n,i):
    return factorial(n)/(factorial(n-i)*factorial(i))

#Se define una función que toma los puntos de control
#y devuelve una lista con dos listas que tienen
#los coeficientes del spline correspondientes a las
#componentse en "x" y "y" respectivamente.
def Spline(n,puntos):
    coeficientesx = []
    coeficientesy = []
    for i in range(n+1):
        Coef = C(n,i)
        coeficientesx.append(Coef*puntos[i][0])
        coeficientesy.append(Coef*puntos[i][1])
    return [coeficientesx,coeficientesy]

#Se define una función que evalúa el spline.
#Toma los coeficientes y un valor de "t" y devuelve
#el valor correspondiente. El trabajo de separar en
#componentes de "x" y "y" se hace al llamar la función
def B(n,t,coef):
    ans = 0
    for i in range(n+1):
```

```
        ans += coef[i]*((1-t)**(n-i))*(t**i)
    return ans

#Se define la función que toma varios valores de "t"
#y los coeficientes del polinomio. Esta función utiliza
#la librería pyplot para graficar el
#resultado y no devuelve nada
def graficar(n,T,coeficientes):
    x = []
    y = []
    for t in T:
        x.append(B(n,t,coeficientes[0]))
        y.append(B(n,t,coeficientes[1]))
    pl.plot(x,y)
    pl.show()
    return None

#Se crea una lista de 100 valores de "t" ordenados entre 0 y 1
T = []
for i in range(100):
    T.append(i/100.0)

#Se colocan los puntos de control sacados del boceto
puntos = [[0.57,3.92],[0.45,4.19],[0.18,4.3],[-0.14,4.06],[-0.37,4.38],[-0.84,4.42],[-1.06,4

#Se calcula el valor de "n" que depende de la cantidad de puntos
n = len(puntos)-1

#Se calculan los coeficientes del spline
coeficientes = Spline(n,puntos)

#Se grafica el spline con los 100 valores de "t"
graficar(n,T,coeficientes)
```

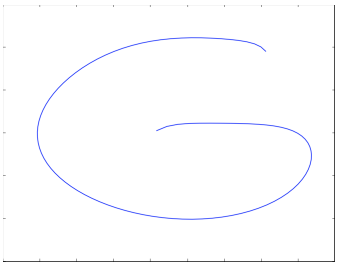
4 Resultados

Para hallar los puntos de control se utilizó el siguiente boceto:



El cual fue realizado en Geogebra. Los puntos de control fueron seleccionados así: cada dos puntos, el punto debe estar sobre el boceto y cada par de puntos que no tocan el boceto deben estaren el exterior de la "G". Esto se realiza así para que la concavidad de la curva sea la correcta y se obtenga un resultado satisfactorio.

La aplicación Geogebra da las coordenadas de cada punto, las cuales fueron colocadas en el código de manera manual. Al correr el código se obtiene el siguiente resultado:



5 Conclusiones

- Se utilizaron varios bocetos y varios puntos de control diferentes y el mejor resultado fue el mostrado.
- La esquina de la "G" no se logra con los splines de Bezier debido a que estas son curvas suaves. Por ende, siempre se tendrá una curvatura suave.
- Los splines de Bezier son muy útiles para dibujar graficas suaves en los computadores de manera sencilla ya que solo hay que dar algunos puntos de control (con solo dar 5 puntos ya se obtenia un resultado aceptable).

- Si se quiere graficar algo que tenga puntas y esquinas de manera correcta, se debería concatenar varios splines de bezier en las partes donde haya un cambio brusco de la pendiente (puntas y esquinas).

6 Referencia

Notas de Clase: Análisis Numérico, Interpolación, Eddy Herrera Daza.