
TP : Courbes de Bézier et algorithme de Casteljau

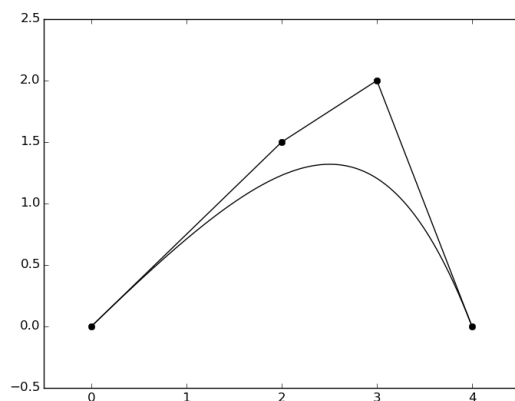


FIGURE 1: Une courbe de Bézier de degré 3

1 Introduction aux courbes de Bézier générales

Si vous avez déjà utilisé un logiciel basique de dessin¹, vous avez sûrement tracé des courbes à l'aide d'un outil pas facile à faire marcher : il faut donner deux points (« l'origine » et la « destination » de la courbe), ainsi que deux « points de contrôle » qui ne sont pas sur la courbe mais qui orientent sa forme. La figure 1 présente une telle courbe : on commence en $(0, 0)$ et on termine en $(4, 0)$, avec points de contrôle $(2, 1.5)$ et $(3, 2)$.

Bien qu'à ma connaissance, ce ne soit pas possible dans les logiciels basiques de dessin, on peut utiliser plus de deux points de contrôle. La figure 2 présente une courbe de Bézier à 6 points (dont 4 points de contrôle). Rien n'empêche les points de former un polygone non convexe, voir aussi la figure 2 : à droite, on a pris les mêmes points que pour la figure 1, en inversant la « destination » et le deuxième point de contrôle.

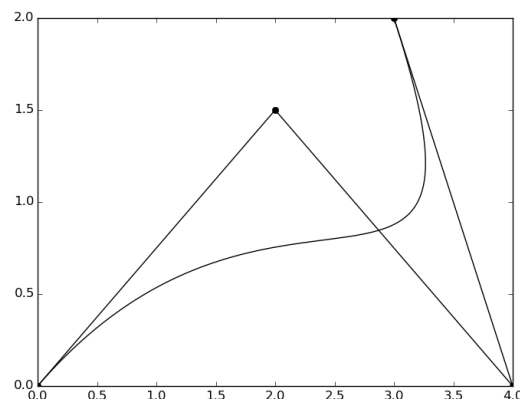
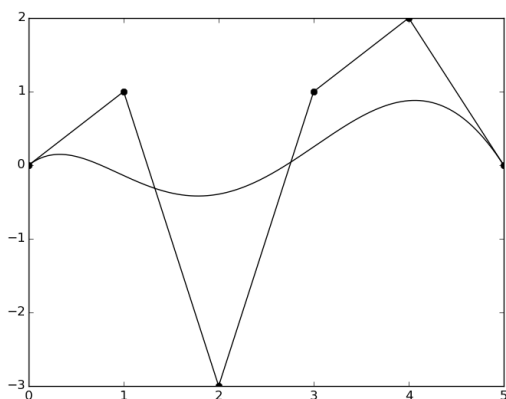


FIGURE 2: Une courbe de Bézier de degré 5, et une de degré 3 formée sur un polygone non convexe

Passons maintenant à la définition des courbes de Bézier : pour n un entier naturel, on définit les polynômes de Bernstein de degré n comme :

1. comme Paint !

$$B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad \text{pour tout } i \text{ entre } 0 \text{ et } n.$$

Ces polynômes sont très courants en mathématiques, vous avez d'ailleurs peut-être vus une démonstration du théorème de Weierstrass² basée sur ces polynômes.

Étant donnés $n+1$ points P_0, \dots, P_n du plan, on définit la courbe de Bézier paramétrée par les (P_i) par

$$M(t) = \sum_{i=0}^n B_{n,i}(t) P_i \quad \text{pour } t \in [0, 1]$$

Par exemple pour $n=3$, on a

$$M(t) = (1-t^3)P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)P_2 + t^3P_3$$

Les propriétés des polynômes de Bernstein impliquent que $M(t)$ est pour tout $t \in [0, 1]$ un barycentre (à poids positif) des points P_i . On remarque aussi que $M(0) = P_0$ et $M(1) = P_n$. De plus la tangente à $M(t)$ suit la direction³ $\overrightarrow{P_0P_1}$ en $t=0$, et la direction $\overrightarrow{P_{n-1}P_n}$ en $t=1$.

Le but de cette partie est de tracer des courbes de Bézier en se donnant un ensemble de points. On commence par importer les modules dont on a besoin :

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
```

Question 1. Écrire une fonction `binome(n,p)` prenant en paramètre deux entiers n et p , et renvoyant $\binom{n}{p}$. On pourra utiliser la fonction `factorial(k)` pour le calcul de $k!$ (elle se trouve normalement dans le module `math`, dont on vient d'importer toutes les fonctions).

Question 2. En déduire une fonction `bernstein(n,i,t)` prenant en paramètres n et i deux entiers, ainsi que $t \in [0, 1]$ et retournant $\binom{n}{i} t^i (1-t)^{n-i}$.

Question 3. Écrire une fonction `bezier(P)` prenant en paramètre une liste de points (une liste de couples, donc), et retournant deux listes X, Y , telles que X et Y soient constituées d'abscisses et d'ordonnées de 1000 points successifs de la courbe de Bézier associée aux points de P . On pourra utiliser `np.linspace(0,1,1000)` pour produire une liste (un tableau Numpy, plutôt) constitué de 1000 flottants régulièrement espacés dans $[0, 1]$.

Question 4. En déduire comment tracer une courbe de Bézier, étant donnée une liste de points. L'appliquer par exemple à $[(0,0), (2,1.5), (3,2), (4,0)]$ pour retrouver le résultat de la figure 1.

Question 5. Améliorer vos graphiques en traçant les points P_i et les segments P_iP_{i+1} . Étant données une liste d'abscisses X et une liste d'ordonnées Y , il suffit d'utiliser `plt.plot(X,Y)` pour relier les points (x_i, y_i) par une ligne brisée. `plt.plot(X,Y, 'o')` « met des petits ronds » sur les points. `plt.plot(X,Y, '-o')` fait les deux. Vous pouvez mettre par exemple une couleur verte en rajoutant `color="green"`.

2 Algorithme de Casteljau pour le tracé des courbes de Bézier de degré 3

On décrit maintenant un algorithme capable de calculer très facilement des points d'une courbe de Bézier, sans avoir à prendre la valeur des polynômes de Bernstein en de multiples réels : la construction est très géométrique ! Cette technique mène à un algorithme récursif pour le tracé d'approximations de courbes de Bézier, en calculant seulement des milieux de segments et en traçant des lignes brisées. L'algorithme est basé sur la propriété suivante, détaillée en figure 3

Soient donc P_0, P_1, P_2 et P_3 un ensemble de 4 points de \mathbb{R}^2 . On considère la courbe de Bézier (de degré 3) définie par ses 4 points. Notons :

- M le milieu du segment $[P_1, P_2]$;
- A_1 le milieu du segment $[P_0, P_1]$;

2. une fonction continue sur un segment est limite uniforme d'une suite de fonctions polynomiales. Pour f continue sur $[0, 1]$, on montre que la suite des $t \mapsto \sum_{i=0}^n f(\frac{i}{n}) B_{n,i}(t)$ converge uniformément vers f lorsque n tend vers l'infini.

3. ce qui est cohérent avec les figures !

- A_2 le milieu du segment $[A_1, M]$;
- B_2 le milieu du segment $[P_2, P_3]$;
- B_1 le milieu du segment $[M, B_2]$;
- $A_3 = B_0$ le milieu du segment $[A_2, B_1]$.

Alors la courbe de Bézier définie par les points P_0, P_1, P_2 et P_3 est exactement la réunion des courbes de Bézier passant par $A_0 = P_0, A_1, A_2$ et A_3 et celle passant par $A_3 = B_0, B_1, B_2$ et $B_3 = P_3$

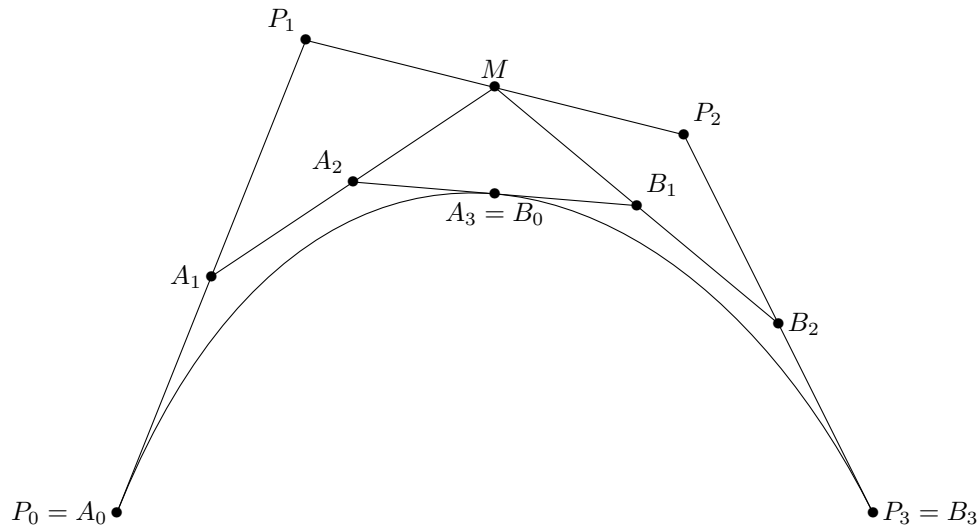


FIGURE 3: Une étape de l'algorithme de Casteljau

Cette construction est l'algorithme de Casteljau. Remarquez que le point $A_3 = B_0$ appartient à la courbe (il correspond au point $M(\frac{1}{2})$), et que la ligne brisée formée des deux suites de segments $[A_0, A_1, A_2, A_3]$ et $[B_0, B_1, B_2, B_3]$ est une approximation bien plus précise de la courbe que n'est la ligne brisée formée par les points $[P_0, P_1, P_2, P_3]$. On peut donc construire récursivement une approximation de la courbe de Bézier : tant que les segments de la ligne brisée sont de longueur supérieure à une certaine borne, on applique l'algorithme de Casteljau.

Question 6. Si ce n'est pas déjà fait, écrivez une fonction `trace_ligne_brisee(P)` traçant les segments $[P_i, P_{i+1}]$ de la liste P .

Question 7. Écrire une fonction `milieu(p,q)` prenant en entrée deux points (représentés par des couples de flottant) et renvoyant le couple associé au milieu du segment $[p, q]$.

Question 8. En déduire une fonction `etape_casteljau(P)` prenant en entrée une liste de 4 points du plan (représentés par des couples de flottants) et retournant deux listes de la forme $[A_0, A_1, A_2, A_3]$ et $[B_0, B_1, B_2, B_3]$ comme détaillé dans l'algorithme de Casteljau.

Question 9. En déduire une fonction (récursive) `bezier_casteljau(P)` prenant en entrée une liste de 4 points du plan (représentés par des couples de flottants) et traçant une approximation de la courbe de Bézier contrôlée par les points de P . Une condition d'arrêt sera par exemple la suivante : la distance entre deux points successifs de P est inférieure à une certaine borne (comme 0.1). Dans ce cas on trace simplement la ligne brisée constituée des points de P .

Question 10. Démontrez que la courbe de Bézier initiale est bien l'union de celles contrôlées par $[A_0, A_1, A_2, A_3]$ et $[B_0, B_1, B_2, B_3]$.

Remarquez que les courbes de Bézier sont vraiment utilisées : toutes les lettres de ce texte sont en fait formées de courbes de Bézier ! Un intérêt est le fait que zoomer sur une lettre dans un fichier pdf ne produit pas de résultat tout « pixellisé » : les courbes de Bézier sont à la base du dessin dit « vectoriel ».

3 Courbes interpolantes de classe \mathcal{C}^1

La problématique de cette section est un peu différente : on se donne une liste de points $(Q_i)_{0 \leq i \leq n}$, et le but est de construire une courbe paramétrée passant par les points Q_i . On ne peut utiliser des courbes de Bézier telles quelles :

- Une courbe de Bézier de degré n construite avec la définition de la section 1 passe par Q_0 et Q_n mais *a priori* pas par les autres points.
- Prendre les points par paquets de 4 (le dernier point d'un paquet étant le premier du suivant) et construire des courbes de Bézier de degré 3 avec l'algorithme de Casteljau mène à une courbe paramétrée qui est seulement \mathcal{C}^1 par morceaux.

On va donc rajouter des points ! Plus précisément, on va construire pour tout $0 \leq i < n$ une courbe de Bézier de degré 3, dont les extrémités seront les points Q_i et Q_{i+1} . Il reste à choisir les deux points de contrôle qu'on notera R_i et S_i .

Question 11. Donner une condition nécessaire et suffisante sur S_i et R_{i+1} pour assurer que la courbe paramétrée obtenue au soit de classe \mathcal{C}^1 au voisinage du point Q_{i+1} (il faut que les tangentes des deux courbes de Bézier en Q_{i+1} soient les mêmes).

Question 12. En notant $M_i(t)$ la courbe de Bézier construite sur les points $[Q_i, R_i, S_i, Q_{i+1}]$, on décide d'imposer $M'_i(1) = \frac{\overrightarrow{Q_i Q_{i+2}}}{\alpha}$, où α est un réel fixé. On décide de plus d'imposer $M'_0(0) = \frac{2\overrightarrow{Q_0 Q_1}}{\alpha}$ et $M'_{n-1}(1) = \frac{2\overrightarrow{Q_{n-1} Q_n}}{\alpha}$. Donner une formulation permettant d'obtenir tous les points R_i, S_i .

Question 13. Écrire une fonction `bezier_c1(Q,alpha)` prenant en paramètres une liste de points Q et un réel α , et traçant avec l'algorithme de Casteljau toutes les courbes de Bézier en question. Observez le résultat obtenu pour plusieurs valeurs de α . (On essaiera par exemple $\alpha = 0.1$, $\alpha = 1$ et $\alpha = 10$).

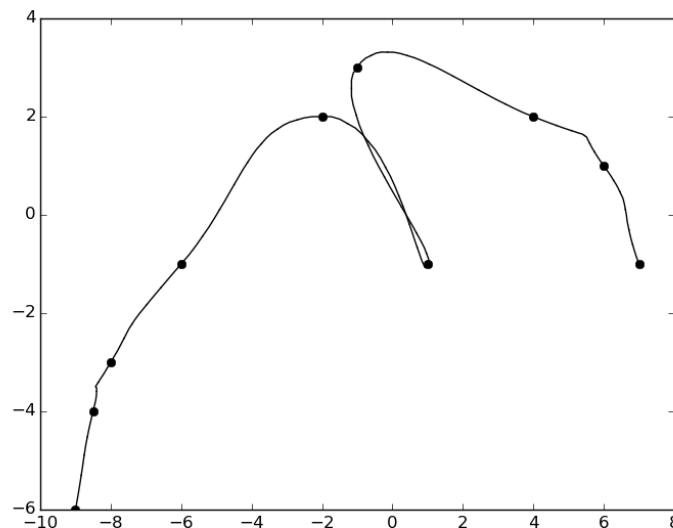


FIGURE 4: Une courbe de classe \mathcal{C}^1 obtenue par raccordement de courbes de Bézier de degré 3. La liste des points est $[(-9, -6), (-8.5, -4), (-8, -3), (-6, -1), (-2, 2), (1, -1), (-1, 3), (4, 2), (6, 1), (7, -1)]$. On a pris $\alpha = 1.1$.

Ici on a imposé simplement à la courbe paramétrée d'être de classe \mathcal{C}^1 , en imposant une valeur à la dérivée en les points Q_i . On aurait pu aussi imposer à la courbe d'être de classe \mathcal{C}^2 .