TP3 - Courbes et surfaces paramétriques (Béziers)

Alexandra Bac

Modélisation géométrique
Polytech Marseille - IRM 4A

Ce TP porte sur les courbes et surfaces paramétriques (de Béziers) et pour simplifier sa mise en oeuvre, on codera cette fois en Python.

1 Courbes de Bézier

Récupérez le matériel de TP. Il contient différentes fonction dont une fonction `DeCasteljau.py` calculant la valeur d'une courbe de Bézier pour un paramètre `t` donné.

Dans toute la suite, les points de contrôles seront passés en ligne, c'est-à-dire sous la forme d'un tableau :

$$\begin{bmatrix}[x_0, y_0], \\ [x_1, y_1], \\ \vdots \\ [x_n, y_n]\end{bmatrix}$$

Exercice 1 (Premières courbes)**.**

1. Ecrire une fonction :

   ```
   plotBezier(P)
   ```

   appelant la fonction `DeCasteljau` pour tracer la fonction de Bézier de points de contrôles stockés dans `P`. Vous tracerez la courbe en bleu et les points de contrôle comme des ronds rouges.

2. Tracez la courbe correspondant à :

   ```
   P = [[1, 2], [2, 4], [3, 2], [4, 4]]
   ```

3. Etant donnés des points de contrôles $P_0, \ldots, P_n$, on sait que l'influence du point P_i est maximale en i/n. Modifiez votre fonction pour tracer d'une couleur différente chaque portion de courbe correspondant aux paramètres $t \in [\frac{i}{n}, \frac{i+1}{n}[$.

   Pour cela, vous pourrez utiliser les fonction `linspace` (permettant de découper de manière régulières des segments) et n'oubliez pas qu'une couleur est un simple triplets de nombres dans $[0, 1]$, donc appeler 3 fois `random.random()` vous donne une couleur aléatoire ...

Exercice 2 (Modéliser une silhouette)**.**

Vous trouverez dans le matériel de TP une image `exemple_courbe.png`. Le but de cette partie est de la modéliser.

Vous pouvez facilement la charger et l'afficher :

```
from PIL import Image
import matplotlib.pyplot as plt

im = Image.open(filename)
fig, ax = plt.subplots()
plt.imshow(im)
```

(a) Est-il possible de modéliser cette forme par une seule courbe de Bézier ?

(b) Combien en faut-il et de quel degré pour chacune ?

(c) Je vous fournis un petit script `script_picking.m` permettant de faire un picking très basique et retournant un `array` numpy des points au bon format ... Il vous suffit de le lancer sur l'image, en cliquant ajoutez des points, en tapant `d` puis un indice de point dans la console, vous pouvez supprimer un point, et enfin `q` pour quitter.

i. Utilisez ce script pour créer approximativement vos points de contrôle. Vous créerez une liste de tableaux numpy contenant les points de contrôle de chaque courbe de Bézier.

ii. Si la courbe est constituée par N courbes de Bézier f^i (et donc N tableaux numpy de points de contrôle), écrire un script traçant l'ensemble des courbes associées sur l'image et faisant également apparaître les points de contrôle. Que pensez-vous de la continuité $\mathcal{G}^1$ de la courbe obtenue ?

Algorithme 1 Algorithme de Newton.

Entrées: Fonction $g : \mathbb{R} \to \mathbb{R}$, t_0 point proche du zéro, ε précision.
Sorties: t^* zéro de g à ε près.

$x \leftarrow x_0$
$\delta \leftarrow 1$
tant que $\text{abs}(\delta) > \varepsilon$ **faire**
    $\delta \leftarrow g(t)/g'(t)$
    $x \leftarrow x - \delta$
fin tant que

Exercice 3 (Projection sur une courbe de Bézier)**.**

1. En utilisant la formule vue en cours :

$$f'(t) = n \sum_{i=0}^{n-1} \varphi_{n-1,i}(t) \cdot (P_{i+1} - P_i)$$

Qui donne le calcul de f' comme une courbe de Bézier pour certains points de contrôle :

(a) Montrez que f'' est donnée par :

$$f''(t) = n(n-1)\sum_{i=0}^{n-2} \varphi_{n-2,i}(t) \cdot (P_{i+2} - 2P_{i+1} + P_i)$$

(b) Modifier la fonction `DeCasteljau` pour créer une fonction de calcul de f' puis de f''

2. Etant donné un point X de l'espace, on cherche le projeté de X sur la courbe, c'est-à-dire le point de la courbe le plus proche de X.

   On peut montrer que ce projeté se situe soit aux extrémités de la courbe, soit au point de paramètre t^* correspondant à :

$$\min_{t\in[0,1]} g(t) \quad \text{avec } g(t) = \|f(t) - X\|^2$$

   Or, si t^* est un minimum de g, alors $g'(t) = 0$.

   Vous connaissez plusieurs algorithmes permettant de trouver un zéro de fonction réelle, dont Newton rappelé ci-dessus. Sachant que :

$$g'(t) = 2\langle f(t) - X, f'(t)\rangle$$
$$g''(t) = 2\left(\|f'(t)\|^2 + \langle f(t) - X, f''(t)\rangle\right)$$

   Ecrire une fonction :

   `projete(P,X)`

   calculant le paramètre `t` du projeté de X sur la courbe de points de contrôles P.

Exercice 4 (Subdivision de la courbe, augmentation de degré)**.**

Comme vu en cours, il l'algorithme de De Casteljau calcule en interne des points qui permettent de subdiviser la courbe et les propriétés des polynômes de Bernstein fournissent un algorithme d'élévation de degré. Coder ces deux outils (très efficaces pour la manipulation des courbes en CAO).

Et si vous devez passer en 3D, à des courbes gauches, qu'est-ce qui change ?

2 Surfaces de Béziers

Le but de cette partie est, en partant de vos connaissances sur les courbes de Béziers, d'explorer les surfaces de Béziers.

Pour rappel, étant donné une grille de points de contôle (appelée *polygone de contrôle*) $\{P_{i,j}\}_{i=0\ldots n, j=0\ldots m}$, un carreau de Béziers de degré (n, m) est obtenu par produit tensoriel de la manière suivante :

$$\begin{array}{rccl} f\ :\ [0,1]^2 & \to & \mathbb{R}^3 \\ (u,v) & \mapsto & \sum_{i=0}^{n}\sum_{j=0}^{m} \phi_{n,i}(u)\phi_{m,j}(v)P_{i,j} \end{array}$$

La figure suivante illustre un tel polygone de contrôle :

Un tel ensemble de points sera codé par une matrice tri-dimensionnelle de taille $3 \times n \times m$ telle que représentée à la figure 2 :

Attention Lorsque vous récupérez des blocs de matrice 3D, ces blocs restent 3D (même quand l'une de leurs dimension n'est que de 1). Il faut utiliser la commande `reshape` pour actualiser leur dimension. Donc par exemple, pour récupérer le point (i, j) (illustré en orange sur le figure 2, il faudra exécuter :

```
P = reshape(M(:,i,j),3,1) ;
```

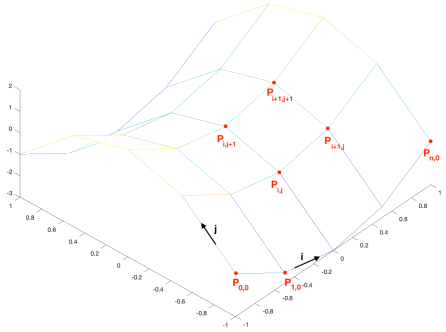

Figure 1: Exemple de polygone de contrôle

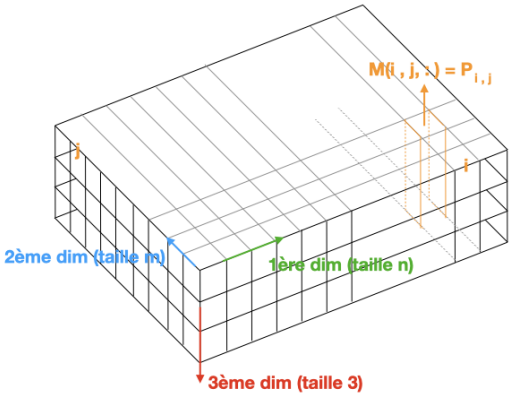

Figure 2: Matrice tri-dimensionnelle de taille $n \times m \times 3$

1. Pour des paramètres (u, v) donnés, en utilisant la factorisation suivante :

$$f(u,v) = \sum_{i=1}^{n} \left(\underbrace{\sum_{j=1}^{m} P_{i,j}\phi_{m,j}(v)}_{Q_i} \right) \phi_{n,i}(u)$$

   (a) A quelle courbe de Bézier f_i les points Q_i appartiennent-ils (pour quels points de contrôle et quel degré) ? Pour quel paramètre t a-t-on $Q_i = f_i(t)$?

   (b) Par quel algorithme (dont vous avez déjà le code) pouvez-vous calculer ces points Q_i ?

   (c) A partir de ces points, à quelle courbe de Béziers appartient le point $f(u, v)$ que l'on souhaitait calculer ?

   (d) En déduire une explication de l'algorithme `DeCasteljau_surf` fourni. Que calcule-t-il ?

2. La fonction `plot_poly_ctrl` permet de tracer un polygone de contrôle et le script `test_DeCasteljau_surf.py` l'appelle sur un fichier contenant un polygône de contrôle (`poly_control1.npy` ou `poly_control2.npy`).

   (a) Ecrivez une fonction `compute_Beziers_surf` calculant un maillage quadratique de la surface de Béziers (donc renvoyant un tableau numpy `M` de taille $N \times N \times 3$ où N est le pas de discrétisation

choisi pour le segment $[0, 1]$ et où `M[i,j,:]` contient les coordonnées du point d'indice `(i,j)` du maillage).

(b) Puis écrivez une fonction `plot_Beziers_surf` affichant ce maillage (utiliser `ax.plot_surface(X,Y,Z)` pour tracer la surface où `X`, `Y` et `Z` sont les grilles 2D des coordonnées des points).

(c) Enfin, tracez le polygone de contrôle ainsi que la surface de Béziers associée. Vous enregistrerez une capture d'écran de votre résultat.

3. Considérons quelques courbes de Béziers particulières :

(a) Adaptez votre code de TP traçant des courbes de Béziers pour qu'il trace des courbes en 3D. Très peu de choses sont à changer ...

(b) Vous tracerez ensuite les 2 courbes de Béziers données par $\{P_{1,j}\}_{j=1...m}$ et $\{P_{i,1}\}_{i=1...n}$. Vous ferez une capture d'écran.

(c) Sont-elles sur la surface ? Expliquez le résultat en fonction des propriétés des Béziers.

(d) Tracez maintenant la courbe donnée par $\{P_{2,j}\}_{j=1...m}$. Est-elle sur la surface ? Pourquoi ?

(e) Si vous devez raccorder deux carreaux de Béziers, quelle est par conséquent la condition pour que le raccordement soit $\mathcal{C}^0$?

(Bonus /2) Chargez maintenant le polygone contenu dans `polygon_2.mat` fournissant un polygone M_2 :

i. Tracez, en plus du carreau de Béziers précédent, celui obtenu pour M_2. Ces deux carreaux de surfaces forment-ils une surface continue ?

ii. Proposez une modification simple de M_2 permettant de raccorder les deux carreaux de surface de manière $\mathcal{C}^0$.

4. Si vous deviez calculer la courbure de la surface au point $P = f(u, v)$ expliquez les étapes de calcul qu'il faudrait suivre.