TP2 - 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 Matlab.

1 Courbes de Bézier

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

Dans toute la suite, les points de contrôles seront passés sous forme d'un tableau :

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_{n+1}; \\ y_1 & y_2 & \cdots & y_{n+1} \end{bmatrix}$$

Exercice 1 (Premières courbes)**.**

1. Ecrire une fonction :

   ```
   function [] = plotBezier(P)
   ```

   appelant la fonction `DeCasteljau.m` 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 (`'or'`).

2. Tracez la courbe correspondant à :

   ```
   P = [1 2 3 4; 2 4 2 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. Donc dans la notation Matlab commençant à 1, l'influence du $i$ème point est maximale en $i-1/n$. Modifiez votre fonction pour tracer d'une couleur différente chaque portion de courbe correspondant aux paramètres $t \in [\frac{i-1}{n}, \frac{i}{n}[$.

   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 `rand(1,3)` 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 :

```
im=imread('exemple_courbe.png') ;
imshow(im) ;
hold on
```

(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 `picking.m` permettant de faire un picking très basique et retournant la liste des points au bon format ... Il vous suffit de le lancer sur l'image, vous pourrez taper `n` pour ajouter un point, `q` pour quitter et un entier pour supprimer l'un des points si vous vous êtes trompés.

i. Utilisez ce script pour créer approximativement vos points de contrôle. Vous créerez un tableau cellulaire dont chaque case contiendra (une référence sur) les points de contrôle d'une des courbres de Bézier.
L'allocation et l'accès se font de la manière suivante :

```
model = cell(1,10);
model{1} = [1,2] ;
model{2} = [1,2;3,4] ;
```

ii. Si la courbe est constituée par N courbes de Bézier f^i (et donc N tableaux de points de contrôle `cell{1} ... cell{N}`), on définit la fonction paramétrique globale :

$$f : [0, N] \to \mathbb{R} \qquad f(t) = f^i(t - i) \text{ pour } t \in [i - 1, i[$$

Ecrire une fonction :

```
function [y] = bezier_morceaux(model, t)
```

calculant la valeur de cette fonction f pour le paramètre t.

iii. Corrigez les points de contrôle pour garantir que la courbe globale soit bien continue partout. Que faut-il fait s'il l'on souhaite qu'elle soit $\mathcal{G}^1$ à certains raccords ? Qu'est-ce que cela implique sur les points ?

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.

1: $x \leftarrow x_0$
2: $\delta \leftarrow 1$
3: **tant que** abs$(\delta) > \varepsilon$ **faire**
4: $\quad \delta \leftarrow g(t)/g'(t)$
5: $\quad x \leftarrow x - \delta$
6: **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 :

```
function [t] = projeté(P,X)
```

   calculant le 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 ?