Algorithme de Casteljau

L'**algorithme de Casteljau** est un algorithme récursif trouvé par Paul de Casteljau pour approximer efficacement les polynômes écrits dans la base de Bernstein.

Cet algorithme peut être utilisé pour dessiner des courbes et des surfaces de Bézier. L'idée principale dans ce cas repose sur le fait qu'une restriction d'une courbe de Bézier est aussi une courbe de Bézier. L'algorithme calcule de manière efficace le point de paramètre $t = T$ et les points de contrôle des courbes de la restriction à $t \leqslant T$ et à $t \geqslant T$. On applique alors de nouveau l'algorithme sur les deux restrictions jusqu'à réaliser un critère donné (celui-ci peut être, par exemple, que la précision soit inférieure au pixel).

Cet algorithme semble ne plus être le plus efficace [réf. nécessaire] car il ne permettrait pas d'utiliser l'antialiasing étant donné qu'il travaille pixel par pixel et ne donne pas d'information sur la tangente.

Sommaire

Historique

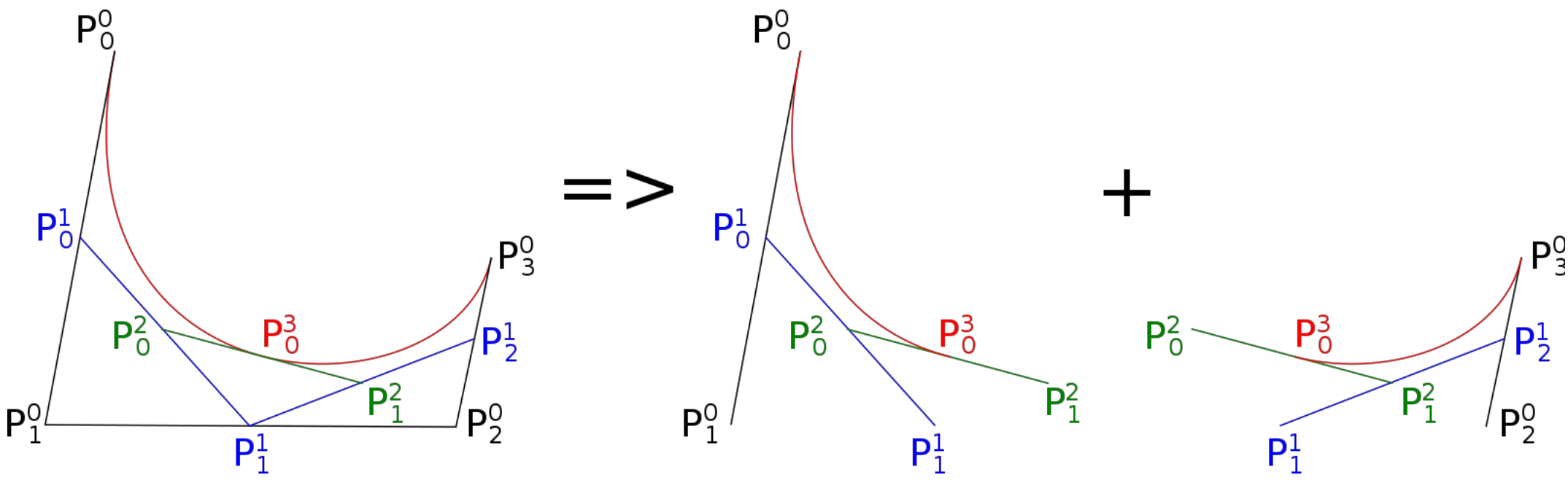
Historiquement, c'est avec cet algorithme que les travaux de M. de Casteljau commençaient en 1959 chez Citroën. Ils étaient publiés comme des rapports techniques, tenus très au secret par Citroën.

Ces travaux restèrent inconnus jusqu'en 1975 quand W. Böhm en a pris connaissance et les a rendu public. Cet algorithme a été très utile pour l'informatique qui utilise les courbes de Bézier dans de nombreux cas (logiciels de dessin, de modélisation…), et sans lequel le développement de l'utilisation des courbes de Pierre Bézier n'aurait pas pu se faire.

L'algorithme de calcul d'un point

Principe

Considérons une courbe de Bézier définie par les points de contrôles $P_0, \ldots, P_N$, où les P_i sont des points de $\mathbb{R}^m, m \geqslant 2$. Ici on souhaite simplement calculer le point de paramètre $t \in [0, 1]$.



Principe de l'algorithme de Casteljau

Comme on peut le voir sur l'image, en calculant les barycentres de paramètres $\{t, 1-t\}$ des points de contrôle consécutifs de la courbe, puis les barycentres de même paramètres de ces barycentres et ainsi de suite itérativement, on définit de cette manière une suite de listes de points que l'on va indexer P_i^j, où P_i^j est le barycentre de $\{P_i^{j-1}, P_{i+1}^{j-1}\}$. La dernière liste ne contient en fait qu'un point, qui est le point de la courbe de paramètre t.

Algorithme

En pseudo-code, ceci donne:

```
// Calcul des points intermédiaires
Pour j de 1 à N faire
|
| Pour i de 0 à N-j faire
| |
| | T[i][j] = t*T[i+1][j-1] + (1-t)*T[i][j-1]
| |
|
Afficher T[0][N] // Afficher (ou stocker) le point
```

Éléments de preuve de l'algorithme

Pour démontrer l'algorithme, il faut prouver par récurrence limitée que

$$\forall i \in [0,n], \forall t \in [0,1], B(t) = \sum_{k=0}^{n} P_k^0 B_k^n(t) = \sum_{k=0}^{n-i} P_k^i B_k^{n-i}(t)$$

Et d'appliquer la formule en $i = n$, ce qui nous donne le résultat directement.

La récurrence se démontre facilement en utilisant la propriété de construction des points $P_i^j = (1-t)P_i^{j-1} + tP_{i+1}^{j-1}$ pour séparer la somme en deux, puis en faisant un changement d'indice et en utilisant une propriété des polynômes de Bernstein $B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i+1}^{n-1}(t)$ pour regrouper les deux sommes. Pour réintégrer les cas particuliers, il faut utiliser deux autres propriétés des polynômes de Bernstein: $B_n^n(t) = t^n = tB_{n-1}^{n-1}$ et $B_0^n(t) = (1-t)^n = (1-t)B_0^{n-1}$

Complexité

L'exécution d'une étape de l'algorithme est quadratique en nombre de points de contrôle de la courbe (la boucle imbriquée donne lieu à $\mathcal{O}(N^2)$ opérations).

L'algorithme dans le cas de courbes de Bézier

Considérons encore une courbe de Bézier définie par les points de contrôles $P_0, \ldots, P_N$, où les P_i sont des points de $\mathbb{R}^m, m >= 2$.

Principe

On va ici appliquer l'algorithme précédent pour trouver le point de paramètre $1/2$, et conserver les barycentres intermédiaires. En effet, la courbe de Bézier de points de contrôle $P_0^i, i \in [0, N]$ est égale à la restriction de la courbe originale à $t \leqslant 1/2$, et la courbe de Bézier de points de contrôle $P_i^{N-i}, i \in [0, N]$ est égale à la restriction de la courbe originale à $t \geqslant 1/2$.

On affiche ou mémorise le point de paramètre $1/2$ (qui est P_0^N) et applique récursivement l'algorithme sur les deux courbes (de même nombre de points de contrôle que l'originale). On s'arrête quand un critère à choisir est vérifié (typiquement la distance entre les points inférieure à une limite).

Plutôt que le paramètre $1/2$, on pourrait prendre un paramètre quelconque et l'algorithme fonctionnerait encore, mais le paramètre $1/2$ est celui qui converge *en moyenne* le plus rapidement. Le paramètre $1/2$ est aussi celui pour lequel les calculs sont les plus rapides quand l'on travaille en coordonnées entières : le calcul de chaque barycentre se fait par une addition et un décalage à droite pour chaque coordonnée, c'est-à-dire sans multiplication ni division.

Algorithme

- **Initialisation**: affecter le tableau des points de contrôle dans les P_i^0
- **Voir que le critère d'arrêt n'est pas vérifié**: il y a plusieurs possibilités dont :
  - Si on fait tous les calculs avec des nombres entiers, un choix peut être de s'arrêter lorsque tous les points sont confondus. (c'est-à-dire que la courbe n'est représentée que par un seul pixel).
  - Si le calcul n'est pas fait sur des nombres entiers, on peut s'arrêter quand les points sont distants d'une distance inférieure à une valeur choisie.
  - On effectue *M* itérations puis on relie les points obtenus (c'est-à-dire que l'on trace le polygone de Bézier), *M* étant déterminé empiriquement.
- **Calcul des points intermédiaires de l'algorithme** : il y a deux méthodes possibles qui donnent finalement les mêmes points :
  - Méthode constructive *(en construisant une suite de milieux)*: On définit itérativement les points $P_0^1, \ldots, P_{N-1}^1, P_0^2, \ldots, P_{N-2}^2, \ldots, P_0^{N-1}, P_1^{N-1}, P_0^N$ tels que P_i^j soit le milieu de $[P_i^{j-1} P_{i+1}^{j-1}]$
  - Méthode matricielle *(en construisant directement les points comme barycentre, les coefficients étant donnés par les matrices de Casteljau)* :

$$\begin{pmatrix} P_0^0 \\ \vdots \\ P_0^N \end{pmatrix} = D_0^N * \begin{pmatrix} P_0^0 \\ \vdots \\ P_N^0 \end{pmatrix} \text{ et } \begin{pmatrix} P_0^N \\ \vdots \\ P_N^0 \end{pmatrix} = D_1^N * \begin{pmatrix} P_0^0 \\ \vdots \\ P_N^0 \end{pmatrix}$$

- **Mémorisation**: on mémorise le point P_0^N qui est sur la courbe.
- **Appel récursif**: on appelle l'algorithme sur les deux courbes de Bézier intermédiaires définies par les points $P_0^i, i \in [0, n]$ d'une part, $P_i^{n-i}, i \in [0, n]$ d'autre part.

Voici un exemple d'implémentation de l'algorithme en pseudo-code avec pour critère d'arrêt l'égalité des points (on travaille donc sur des entiers) et la construction *constructives* pour calculer les points intermédiaires :

```
Entrée : tableau T[0][0…N] des coordonnées des points de contrôle.
```

```
Si T[0][0] = T[0][1] = … = T[0][N]     //si le critère d'arrêt est vérifié on s'arrête
alors
|
| fin
|
Sinon  //pour dessiner
|
| // Calcul des points intermédiaires
| Pour i de 1 à N faire
| |
| | Pour j de 0 à N-i faire
| | |
| | | T[i][j] = milieu de T[i-1][j] T[i-1][j+1]
|
| Afficher T[N][0] // Afficher (ou stocker) le point milieu
|
| // Construction des courbes restreintes
| Pour i de 0 à N faire
| |
| | T'[i] = T[i][0]
| | T"[i] = T[N-i][i]
|
| // Appel récursif
| de_Casteljau(T')
```

```
| de_Casteljau(T")
```

Complexité

Si on prend comme critère d'arrêt un nombre d'appels récursifs constant, comme le nombre de points de contrôle est constant pendant les appels récursifs reste constant et qu'à chaque étape de récursion on double le nombre de courbes étudiées, la complexité de l'algorithme est en $\mathcal{O}(N^2 * 2^M)$ avec M le nombre de récursions (mais il est linéaire en nombre de points calculés car pour M itérations il y a 2^M points calculés).

L'algorithme dans le cas de surfaces de Bézier

Une surface de Bézier est définie par une double somme de points: $P(t_1, t_2) = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{i,j} B_m^i(t_1) B_n^j(t_2)$

Le principe de l'algorithme est de réécrire la formule sous la forme:

$$P(t_1, t_2) = \sum_{i=1}^{m} B_m^i(t_1) \left(\sum_{j=1}^{n} P_{i,j} B_n^j(t_2) \right)$$

et en renommant $Q_i(t_2) = \sum_{j=1}^{n} P_{i,j} B_n^j(t_2)$, on obtient

$$P(t_1, t_2) = \sum_{i=1}^{m} Q_i(t_2) B_m^i(t_1)$$

En remarquant que les $Q_i(t_2)$ sont des points de courbes de Bézier, le principe de l'algorithme arrive. À chaque itération de l'algorithme

- calculer les points $Q_i(1/2)$ et les restrictions des courbes de Bézier pour chaque i par l'algorithme de Casteljau sur les courbes
- calculer puis afficher/stocker les points de la courbe de Bézier de points de contrôle les $Q_i(1/2)$ par l'algorithme de Casteljau sur les courbes
- appliquer récursivement l'algorithme sur les deux surfaces obtenues en regroupant les restrictions pour $t_2 \leqslant 1/2$ et $t_2 \geqslant 1/2$

Voir aussi

Bibliographie

- Claude Brezinski, *Méthodes numériques de base*, Techniques de l'ingénieur (lire en ligne (https://books.google.fr/books?id=KDYyDAsLeFgC&pg=PA7)), p. 7
- Jean-Louis Merrien, *Analyse numérique avec MATLAB*, Dunod, coll. « Sciences Sup / Exercices et problèmes », 2007 (lire en ligne (https://books.google.fr/books?id=7Q04iRIYvtEC&pg=PA89)), p. 89-90
- Alain Yger et Jacques-Arthur Weil, *Mathématiques appliquées L3 : Cours complet avec 500 tests et exercices corrigés*, Pearson Education France, 2009 (lire en ligne (https://books.google.fr/books?id=fqOu6xKdmhcC&pg=PA115)), p. 115

Sur les autres projets Wikimedia :

Courbe de Bézier en C, sur Wikibooks