Méthodes numériques
Année : 2018-2019
Formation : L3 Mathématiques

TD 4 : Polynômes de Bernstein et théorème d'approximation uniforme de Weierstrass

1 TD

Exercice 1 : Le but de cet exercice est de démontrer, à l'aide des polynômes de Bernstein, le résultat suivant connu sous le nom de Théorème d'approximation de Weierstrass :

Théorème. *Toute fonction continue sur un intervalle $[a, b]$ est limite uniforme d'une suite de polynômes.*

Ici on se restreint à l'intervalle $[0, 1]$ mais il est facile de généraliser ensuite à un intervalle quelconque $[a, b]$.

On notera pour $0 \leq k \leq n$ entiers,

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

et pour $x \in [0, 1]$,

$$B_k^n(x) = \binom{n}{k} x^k (1-x)^{n-k}.$$

1. Montrer que la fonction B_k^n est positive sur $[0, 1]$.
2. Montrer que pour tout $x \in [0, 1]$,
$$\sum_{k=0}^{n} B_k^n(x) = 1.$$
3. Montrer que si $k \neq 0$ alors $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$. En déduire que pour tout $x \in [0, 1]$,
$$\sum_{k=0}^{n} k B_k^n(x) = nx.$$
4. Montrer que pour tout $x \in [0, 1]$,
$$\sum_{k=0}^{n} k(k-1) B_k^n(x) = n(n-1)x^2.$$
5. En utilisant le fait que $(nx - k)^2 = k(k-1) - (2nx-1)k + n^2x^2$, montrer que pour tout $x \in [0, 1]$,
$$\sum_{k=0}^{n} (nx-k)^2 B_k^n(x) = nx(1-x).$$

On définit le polynôme de Bernstein d'ordre n associé à une fonction $f : [0, 1] \to \mathbb{R}$ par

$$B_n(f) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) B_k^n.$$

6. Montrer que pour tout entier $n \geq 1$ et pour tout réel $\delta > 0$, on a

$$f - B_n(f) = \sum_{|x-k/n| \leq \delta} \left(f - f\left(\frac{k}{n}\right) \right) B_k^n + \sum_{|x-k/n| > \delta} \left(f - f\left(\frac{k}{n}\right) \right) B_k^n.$$

7. En remarquant que toute fonction continue sur $[0,1]$ est uniformément continue (c'est le Théorème de Heine) et en utilisant les questions précédentes, montrer que si f est continue sur $[0,1]$ alors la suite $B_n(f)$ converge uniformément vers f sur $[0,1]$.

2 TD machine

Exercice 1 :

1. Écrire une fonction `function r = binom(k,n)` qui calcule le coefficient binomial $\binom{n}{k}$ en utilisant la formule (1). On fera attention à faire le maximum de simplification pour faire le moins de multiplications. Combien de multiplications sont nécessaires pour calculer $\binom{n}{k}$?

2. Écrire une fonction `function r = binomrec(k,n)` qui calcule récursivement le coefficient binomial $\binom{n}{k}$ en utilisant la formule du triangle de Pascal :

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$$

3. Comparer le temps d'exécution de ces deux fonctions sur des exemples pertinents à l'aide de la commande `timer()`.

4. Écrire une fonction `function r = bernstein(n,f,T)` qui renvoie la liste des valeurs prises par la polynôme $B_n(f)$ sur la liste T. On pourra utiliser la fonction `binomrec`.

5. Tracer sur un même graphe la fonction $f : x \mapsto \cos(10x)$ sur l'intervalle $[0,1]$ ainsi que $B_n(f)$ pour différentes valeurs de n.

6. Cette question porte sur l'optimisation de la fonction `bernstein`. Voici trois pistes :

- Pour calculer $B_n(f)$ il faut calculer $\binom{n}{k}$ pour k variant de 0 à n. Or si on utilise à chaque fois la fonction `binomrec(k,n)` pour calculer $\binom{n}{k}$, on fera plusieurs fois les mêmes calculs. En effet pour calculer $\binom{n}{1}$ et $\binom{n}{2}$ avec la formule de Pascal on a besoin dans les deux cas de calculer $\binom{n-1}{1}$. Il est donc inutile de le calculer deux fois, il suffit de stocker le triangle de Pascal dans un tableau en mémoire (on pourra faire ça de manière itérative plutôt que récursive).
- On peut remarquer que $\binom{n}{k} = \binom{n}{n-k}$.
- Le calcul des $x^k(1-x)^{n-k}$ pour k variant de 0 à n nécessite au plus $3n$ multiplications/divisions (au lieu de n^2 de manière naïve). En effet pour passer de $x^k(1-x)^{n-k}$ à $x^{k+1}(1-x)^{n-k-1}$ il suffit de multiplier par x et de diviser par $1-x$ (si $x \neq 1$).

Écrire une fonction `function r = bernsteinOP(n,f,T)` qui optimise la fonction `bernstein` à l'aide d'une ou plusieurs des trois remarques ci-dessus. On pourra comparer le temps d'exécution de différentes versions de cette fonction.