TP n^{o}3 : approximation de fonctions

On s'intéresse dans ce TP à l'approximation de fonctions par des polynômes. On introduit les polynômes de Lagrange, de Newton et de Bernstein. ***Les codes sont à rendre pour le dimanche 17 janvier 2010, 23h59.***

Le principe est de se donner une fonction f à interpoler et $n+1$ points $a_0 < a_1 < \ldots < a_n$ dans un intervalle I auxquels on relève les valeurs de la fonction. On construit alors un polynôme P qui prend les mêmes valeurs que la fonction f aux points décrits ci-dessus.

1 Interpolation polynomiale

Le but de ce paragraphe est d'implémenter l'évaluation d'un polynôme P en N points à partir de la donnée de $n+1$ points $a_0 < \ldots < a_n$ et des valeurs de P en ces points. On propose pour cela deux algorithmes utilisant deux bases distinctes de $\mathbb{R}_n[X]$.

L'**interpolation de Newton** fait intervenir les différences divisées. En effet, on vérifie que tout polynôme P peut s'écrire sous la forme :

$$P = P(a_0) + \sum_{k=1}^{n} P[a_0, \ldots, a_k] \prod_{j=0}^{k-1} (X - a_j). \tag{1}$$

On rappelle que le calcul des différences divisées se fait par récurrence :

- $P[a_j] = P(a_j)$ pour tout j,
- $P[a_i, \ldots, a_j] = \dfrac{P[a_{i+1}, \ldots, a_j] - P[a_i, \ldots, a_{j-1}]}{a_i - a_j}$ pour tous $i < j$.

L'**interpolation de Lagrange** consiste quant à elle à décomposer le polynôme P sur la base de Lagrange. Les coordonnées dans cette base sont justement les valeurs de P aux points considérés :

$$P = \sum_{k=0}^{n} P(a_k)\ell_k, \qquad \text{où} \qquad \ell_k = \prod_{\substack{0 \leqslant j \leqslant n \\ j \neq k}} \frac{X - a_j}{a_k - a_j}. \tag{2}$$

Dans le cadre de l'interpolation polynomiale, on construit un polynôme de degré n tel que $P(a_i) = f(a_i)$ pour les $n+1$ points a_i donnés distincts. On montre classiquement que ce polynôme est unique et on peut estimer l'erreur. Pour pouvoir tracer la courbe interpolée, on est amené à calculer les valeurs du polynôme en un nombre important de points, d'où l'intérêt d'optimiser ces évaluations.

1. Ecrire les algorithmes correspondant à ces deux méthodes.
2. Etant donnés les $n+1$ points a_i et les $n+1$ valeurs correspondantes $P(a_i)$, comparer le coût de calcul de ces deux algorithmes adaptés pour l'évaluation d'un polynôme en N points distincts.
3. Programmer ces deux méthodes sous MATLAB. Tracer les courbes de fonctions et de leurs interpolées.

2 Courbes de Bézier

2.1 Polynômes de Bernstein

Les polynômes de Bernstein de degré $n \in \mathbb{N}$ sont définis pour $k \in \{0, \dots, n\}$ par :

$$B_{n,k} = \binom{n}{k} X^k (1-X)^{n-k}.$$

1. Prouver que la famille $(B_{n,k})_{0 \leqslant k \leqslant n}$ forme une base de $\mathbb{R}_n[X]$. Que vaut $B_{n,0}$? $B_{n,n}$? $\displaystyle\sum_{k=0}^{n} B_{n,k}$?
2. Prouver l'égalité pour $n \in \mathbb{N}^*$ et $k \in \{1, \dots, n\}$: $B'_{n,k} = n\,(B_{n-1,k-1} - B_{n-1,k})$.
3. Représenter sur $[0,1]$ les polynômes $B_{5,k}$.

2.2 Algorithme de Casteljau

On se donne $n+1$ réels $a_0, \dots, a_n$. On admet le théorème suivant :

Théorème 1 *On construit la suite double de fonctions $\alpha_{i,j}$ définies sur $[0,1]$ par :*

- $\forall\, j \in \{0, \dots, n\}$, $\alpha_{0,j}(t) = a_j$;
- $\forall\, j \in \{1, \dots, n\}$, $\forall\, i \in \{j, \dots, n\}$, $\alpha_{i,j}(t) = t\alpha_{i-1,j}(t) + (1-t)\alpha_{i-1,j-1}(t)$.

Alors $\alpha_{n,n}(t) = \displaystyle\sum_{k=0}^{n} a_k B_{n,k}(t)$.

2.3 Courbes

Définition 1 *Etant donnés $n+1$ points du plan A_0, A_1, …, A_n, on appelle **courbe de Bézier** $\mathcal{C}_{0\to n}$ la courbe de représentation paramétrique :*

$$M_{0\to n}(t) = \sum_{k=0}^{n} A_k B_{n,k}(t), \qquad t \in [0,1].$$

*Les A_k sont appelés **points de contrôle** de la courbe.*

On se donne une ligne polygonale à $n+1$ points $\mathcal{A} = \overline{A_0 A_1 \dots A_n}$.

1. Quel est le point de départ de la courbe de $\mathcal{C}_{0\to n}$? son point d'arrivée ?
2. Justifier que la courbe $\mathcal{C}_{0\to n}$ admet (A_0A_1) et $(A_{n-1}A_n)$ pour tangentes respectives aux points A_0 et A_n.
3. Montrer que les coordonnées du point $M_{0\to n}(t)$ peut se calculer par un produit matriciel du type : ${}^tTQA$, où $T \in \mathbb{R}^{n+1}$ et $A \in \mathcal{M}_{n+1,2}(\mathbb{R})$ ont pour composantes $T_i = t^{i-1}$, $A_{i,1} = (A_{i-1})_x$ et $A_{i,2} = (A_{i-1})_y$, Q étant à déterminer.
4. En appliquant l'algorithme de Casteljau à chaque composante de $M_{0\to n}(t)$, implémenter le traçage de la courbe de Bézier associée à la ligne $\overline{A_0 \dots A_4}$ avec :

   $$A_0(-1,0),\ A_1(0,1),\ A_2(1,1),\ A_3(1,0),\ A_4(1,-1).$$

   Que se passe-t-il en échangeant l'ordre des points ?
5. Commenter l'efficacité des méthodes de calcul proposées dans les deux questions précédentes.
6. Tracer la courbe de Bézier associée aux points $\big(a_i, f(a_i)\big)$ pour une fonction et des points de votre choix.