Un algorithme dynamique pour les polynômes d'interpolation de Lagrange

Julien Devevey

2018-2019

Ref : Dumas - Modélisation à l'oral de l'Agrégation p.37

Remarque 1. Les polynômes d'interpolation de Lagrange sont très utiles en maths car ils permettent d'approcher des fonctions par des polynômes : Si on a $n+1$ points distincts, on peut trouver un unique polynôme de degré n qui a les mêmes valeurs que f en ces $n+1$ points. Malheureusement, la formule vue classiquement en cours de mathématiques n'est pas récursive : il est impossible de rajouter un $n+2$-ème point sans reprendre les calculs du polynôme à 0. On propose donc ici une méthode en programmation dynamique pour calculer puis évaluer en certains points le polynôme d'interpolation de Lagrange d'une fonction f donnée, et on étudie la correction et la complexité de l'algorithme proposé.

Définition 2 (Différence divisée). On appelle différence divisée d'ordre n de f aux points $(x_i)_{0\leq i\leq n}$ le coefficient (même s'il est nul) de X^n dans le polynôme d'interpolation de f en ces mêmes points. On la note alors $f[x_0, \dots x_n]$.

Remarque 3. La différence divisée d'ordre n a besoin de $n+1$ points. Par exemple, $f[x_0] = f(x_0)$ car dans ce cas, le polynôme d'interpolation est la constante égale à $f(x_0)$.

Lemme 4. On a la relation de récurrence suivante :

$$\forall i, j \in [\![0, n]\!], i < j, f[x_i, \dots, x_j] = \frac{f[x_{i+1}, \dots, x_j] - f[x_i, \dots, x_{j-1}]}{x_j - x_i}$$

Démonstration.

Soit P et Q les polynômes d'interpolation de Lagrange de f en $(x_{i+1}, \dots, x_j)$ et $(x_i, \dots, x_{j-1})$ respectivement. On pose alors $R(X) = \frac{(X-x_i)P(X)-(X-x_j)Q(X)}{x_j-x_i}$. On montre alors que R est le polynôme de Lagrange associé aux points $(x_k)_{i\leq k\leq j}$. On a en effet :

- $R(x_i) = Q(x_i) = f(x_i)$
- $R(x_k) = \frac{(x_k-x_i)P(x_k)-(x_k-x_j)Q(x_k)}{x_j-x_i} = \frac{(x_k-x_i-x_k+x_j)f(x_k)}{x_j-x_i} = f(x_k) \forall k \in [\![i+1, j-1]\!]$

— $R(x_j) = P(x_j) = f(x_j)$

Or, on a que le coefficient de X^{j-i} de P et Q est respectivement $f[x_{i+1}, \ldots, x_j]$ et $f[x_i, \ldots, x_{j-1}]$, par hypothèse. Alors par construction de R, on a que son coefficient de X^{j-i+1} est $\frac{f[x_{i+1},\ldots,x_j]-f[x_i,\ldots,x_{j-1}]}{x_j-x_i}$. □

Théorème 5. Le polynôme interpolateur L de f aux points $(x_i)_{0\leq i\leq n}$ peut s'écrire de la manière suivante :

$$L(X) = f[x_0] + \sum_{j=1}^{n} f[x_0, \ldots, x_j](X - x_0)\ldots(X - x_{j-1})$$

Démonstration.

On procède par récurrence sur n. Pour $n = 0$ on a bien, comme dit dans la remarque 3, que $f(x_0) = f[x_0]$.

Soit alors $n \in \mathbb{N}^*$ et supposons que la formule soit vraie pour $n + 1$ points. Soit x_{n+1} un $n + 2$-ème point qu'on souhaite rajouter. On note L_n le polynôme d'interpolation de f en $(x_i)_{0\leq i\leq n}$ et L_{n+1} celui en $(x_i)_{0\leq i\leq n+1}$. Alors on a $\forall i \in [\![0, n]\!], (L_{n+1} - L_n)(x_i) = 0$ et donc comme $\deg(L_{n+1} - L_n) \leq n + 1$ on a $\exists\lambda \in \mathbb{R}, L_{n+1} - L_n = \lambda(X - x_0)\ldots(X - x_n)$. On en a en outre, par définition des différences divisées et du fait que $\deg(L_n) < \deg(L_{n+1})$ que $\lambda = f[x_0, \ldots, x_{n+1}]$. On applique alors l'hypothèse de récurrence pour réécrire L_n et conclure. □

Algo 6. Algorithme d'interpolation :

Algorithm 1 Calcul des coefficients

Entrée: $f, x_0, \ldots, x_n$
Sortie: T

```
Créer une matrice T de taille (n+1) × (n+1)
pour i ← 0 à n faire
    T[i][i] ← f(x_i)
fin pour
pour k ← 1 à n faire
    pour i ← 0 à n − k faire
        T[i][i+k] ← (T[i+1][i+k] − T[i][i+k−1]) / (x_{i+k} − x_i)
    fin pour
fin pour
```

Algorithm 2 algorithme de Hörner adapté

Entrée: $T, x_0, x_1, \ldots, x_n, x$, le tableau des différences divisées en les $x_0, x_1, \ldots, x_n$ et un point où évaluer

Sortie: res, l'évaluation du polynôme d'interpolation en x

```
res ← 0
pour i ← n à 0 faire
    res ← (x − x_i) × res + T[0][i]
fin pour
```

Théorème 7. L'algorithme précédent est correct, et le calcul des coefficients prend un temps en $O(n^2)$. Le coût d'une évaluation est $O(n)$.

Démonstration.

L'algorithme 1 construit la matrice T telle que $T[i][j] = f[x_i, \ldots, x_j]$. En effet, l'algorithme est construit en se basant sur la relation de récurrence trouvée dans le lemme 4. L'algorithme 2 réutilise la relation de récurrence du théorème 5 et utilise l'algorithme de Hörner pour évaluer le polynôme d'interpolation de $(x_0, x_1, \ldots, x_n)$ en x. Les deux algorithmes sont donc corrects de par leur construction elle même.

La complexité se déduit de la taille des boucles et le fait que chaque itération prenne un temps constant.

Remarque 8. On peut améliorer la complexité spatiale de l'algorithme en ne conservant qu'un vecteur colonne à la place du tableau, ce qui rend l'algorithme encore plus efficace !