Evaluation multi-points, interpolation, calculs modulaires

Christophe Ritzenthaler

1 Interpolation : algorithmes naïfs

Soient $(x_1, y_1), \ldots, (x_n, y_n) \in k^2$ avec les x_i distincts [1]. Il existe un unique polynôme $P \in k[X]$ de degré $< n$ tel que $P(x_i) = y_i$. Ce polynôme, dit *d'interpolation* peut se construire de plusieurs manières.

Remarque 1. *L'évaluation multi-points, i.e. l'évaluation d'un polynôme de degré au plus n en n points de manière naïve se fait en n évaluations qui coûte chacune $O(n)$ opérations dans k. On a donc un coût total quadratique.*

1.0.1 L'interpolation basique

Elle se fait selon la base canonique X^i. Cela revient donc à résoudre un système linéaire avec une matrice de Vandermonde, ce qui est coûteux.

1.0.2 Interpolation de Lagrange

Soit

$$b_i = \frac{\prod_{j=1}^{n}(X - x_j)}{X - x_i}, \quad i = 0, \ldots, n.$$

Ces polynômes forment une base des polynômes de degré $< n$ et on a facilement

$$P(X) = \sum_{i=1}^{n} \frac{y_i}{b_i(x_i)} b_i(X).$$

Il est donc très facile de faire varier les valeurs des y_i. Par contre, la modification d'un seul des x_i ou l'ajout d'un autre point entraîne le recalcul global.

Remarque 2. *On peut être tenté de faire de l'interpolation avec des polynômes en plusieurs variables. Je vous renvoie à l'article (très simple) "A Simple Expression for Multivariate Lagrange Interpolation" pour les formules.*

Le calcul tel quel de l'interpolation de Newton donne un algorithme en $O(n^3)$ opérations dans k. On peut faire mieux (voir Schost Chap.5, Sec.3).

1.0.3 Interpolation de Newton

On cherche un polynôme P de degré $n-1$ vérifiant $P(x_i) = y_i$ pour $i = 1, \ldots, n$ sous la forme

$$P(X) = \lambda_0 + \lambda_1(X - x_1) + \lambda_2(X - x_1)(X - x_2) + \ldots + \lambda_{n-1}(X - x_1) \cdots (X - x_{n-1}).$$

1. Pour des anneaux plus généraux qu'un corps k, voir la partie 2

On appelle ce polynôme le *polynôme d'interpolation de Newton.*

Remarquons que $P(x_1) = \lambda_0 = y_1$. Pour déterminer les autres coefficients, introduisons

$$P_1(X) = \lambda_1 + \lambda_2(X - x_2) + \ldots + \lambda_{n-1}(X - x_2)\cdots(X - x_{n-1}).$$

Puisque $P_1(X)(X - x_1) + \lambda_0 = P(X)$ on a

$$P_1(x_i) = \frac{y_i - y_1}{x_i - x_1}.$$

On en déduit que P_1 est un polynôme d'interpolation de Newton pour des $x'_1 = x_2, \ldots, x'_{n-1} = x_n$ avec $y'_i = \frac{y_{i+1} - y_1}{x'_i - x_1}$. Cela permet d'envisager un calcul récursif sous la forme Le coût de l'al-

Algorithm 1: Newton

input : $x_1, \ldots, x_n \in \mathbb{R}$ distincts et $y_0, \ldots, y_n \in \mathbb{R}$
output: Le polynôme interpolateur P
if $n = 1$ **then**
  return y_1
else
  $x'_1 = x_2, \ldots, x'_{n-1} = x_n$;
  $y'_i = \frac{y_{i+1} - y_1}{x'_i - x_1}$
  return $y_1 + (X - x_1) \cdot \text{Newton}(x'_i, y'_i)$

gorithme en termes d'opérations dans k est le suivant. À chaque étape j, on doit faire $2j$ multiplications pour calculer les y'_i et j pour le produit avec $(X - x_1)$ soit $3j$ avec $j = 1, \ldots, n-1$ donc en tout $O(n^2)$.

Remarque 3. *Supposons que les $y_i = f(x_i)$ pour f une fonction $\mathcal{C}^\infty$. La limite du polynôme de Newton lorsque tous les noeuds coïncident est le polynôme de Taylor car les différences divisées deviennent des dérivées :*

$$\lim_{(x_0,\ldots,x_n)\to(z,\ldots,z)} F(x) = f(z) + f'(z)(x - z) + \ldots + \frac{f^{(n)}(z)}{n!}(x - z)^n.$$

Remarquons que le calcul des différences divisées devient très simple lorsque les points sont équi-répartis. De plus cette méthode est préférée à la méthode de Lagrange en pratique car elle est plus souple. On peut rajouter un point sans recalculer les coefficients précédents et si on modifie une valeur, on a seulement besoin de recalculer les valeurs après celui-ci.

1.1 Spline

Soient $(x_0, y_0), \ldots, (x_n, y_n) \in \mathbb{R}^2$ avec les x_i croissants distincts. Sur chacun des intervalles $[x_i, x_{i+1}]$, on considère un polynôme (habituellement de degré 3), $(P_i)_{i=0,\ldots,n-1}$ qui vérifient les propriétés suivantes :

$$P_i(x_i) = y_i, \quad P_i(x_{i+1}) = y_{i+1}, \quad P'_i(x_{i+1}) = P'_{i+1}(x_{i+1}), \quad P''_i(x_{i+1}) = P''_{i+1}(x_{i+1}).$$

Ces conditions sont des conditions naturelles de recollement. On suppose parfois que la dérivée seconde est nulle aux extrémités de l'intervalle pour permettre de prolonger par une fonction affine en dehors de l'intervalle.
Ces fonctions sont utilisées dans les modélisations de courbes planes.

Remarque 4. *Les points où la fonction change ne sont pas nécessairement les points de contrôle. C'est ici une simplification.*

1.2 Courbes de Bézier

voir texte.

2 Interpolation rapide

Voir le chapitre 5 du poly Algorithmes efficaces en calcul formel.

Remarque 5. *Nous n'avons traité que le cas de l'évaluation multi-points et de l'interpolation. Comme il est mentionné dans le poly, les techniques peuvent être généralisées aux calculs modulaires, pour les polynômes ou les entiers. Sans entrer dans les détails, rappelons qu'il est de bon ton de savoir comment fonctionne le théorème des restes chinois (qui est une évaluation/interpolation généralisée). Sous sa forme abstraite, celui-ci dit que dans un anneau A, disons euclidien comme $\mathbb{Z}$ ou $k[X]$, si on se donne deux éléments premiers entre eux m et n alors on a un isomorphisme*

$$\phi : A/(mn) \to A/m \times A/n.$$

Cette isomorphisme est simplement $\phi(x) = (x \pmod m, x \pmod n)$ et son calcul peut donc se faire au coût de deux divisions euclidiennes dans A. Son inverse peut également être explicité. Par Bézout, il existe u, v tels que $um + vn = 1$. Alors il est facile de voir que $\phi^{-1}((a, b)) = bum + aun$.

Références

[AF] J.M. Arnaudiès, H. Fraysse : cours de mathématiques-1 Algèbre, Dunod Université.

[Gou] Gourdon

[Goz] I. Gozard, Théorie de Galois.

[Knu] Knuth tome 2.

[Fra] Francinou.

[MS] M. Mignotte, D. Stefanescu, *Polynomials, an algorithmic approach*, Springer.