L'interpolation polynomiale

Le calcul d'un polynôme

Evaluation d'un Polynôme

Considérons un polynôme de degré n défini par ses $n+1$ coefficients a_i

$$A(X) = \sum_{i=0}^{n} a_i X^i$$

```
double evalpol(double x, double *pol, int n)
{
  double y;
  int i;
  y = pol[0];
  for(i=1; i<=n; i++)
    y = y + pol[i]*pow(x, i);
  return y;
}
```

Un peu mieux

Considérons un polynôme de degré n défini par ses $n+1$ coefficients a_i

$$A(X) = \sum_{i=0}^{n} a_i X^i$$

```
double evalpol(double x, double *pol, int n)
{
  double y, aux;
  int i;
  y = pol[0];
  aux = 1;
  for(i=1; i<=n; i++)
    {
      aux *= x;
      y = y + pol[i]*aux;
  return y;
}
```

Schéma de Horner du premier ordre

Considérons un polynôme de degré n défini par ses $n + 1$ coefficients a_i

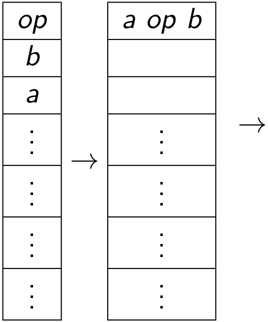
$$A(X) = \sum_{i=0}^{n} a_i X^i$$

Nous avons

$$A(X) = (...(a_n.X + a_{n-1}).X + ... + a_3).X + a_2).X + a_1).X + a_0$$

$$A(X) = (\ldots(a_n.X + a_{n-1}.)X + \ldots\ldots + a_3).X + a_2).X + a_1).X + a_0$$

Une pile



$$\begin{array}{ll}
\leftarrow & a_n \\
\leftarrow & x \\
\leftarrow & "*" \\
\leftarrow & a_{n-1} \\
\leftarrow & "+" \\
\leftarrow & x \\
\leftarrow & "*" \\
\leftarrow & a_{n-2} \\
\leftarrow & "+" \\
\vdots & \\
\leftarrow & a_1 \\
\leftarrow & "+" \\
\leftarrow & x \\
\leftarrow & "*" \\
\leftarrow & a_0 \\
\leftarrow & "+"
\end{array}$$

Le code C

n est le degré du polynôme.

```
double horner(double *a, double x, int n)
{
  double y;
  int i;
  y = a[n];
  for(i=n-1; i>=0; i--)
    y = y*x + a[i];
  return y;
}
```

Exemple

$$A(X) := 2 * X^4 - 3 * X^3 - 5 * X^2 - 4 * X + 1 \text{ en } X = 3$$

- $Y \leftarrow 2 * 3$
- $Y \leftarrow 6 - 3$
- $Y \leftarrow 3 * 3$
- $Y \leftarrow 9 - 5$
- $Y \leftarrow 4 * 3$
- $Y \leftarrow 12 - 4$
- $Y \leftarrow 8 * 3$
- $Y \leftarrow 24 + 1 = 25$

Schéma de Horner du second ordre

On sépare les indices pairs des indices impairs.

$$\begin{aligned} A(X) &= \sum_{i=0}^{n} a_i X^i \\ &= \sum_{i=0}^{\lceil n/2 \rceil} a_{2i} X^{2i} + X. \sum_{i=0}^{\lceil (n-1)/2 \rceil} a_{2i+1} X^{2i} \\ &= A_0(X^2) + X.A_1(X^2) \end{aligned}$$

$\lceil k \rceil$ est la partie entière de k.

Facilite une approche parallélisable

Efficace s'il faut calculer $P(x)$ et $P(-x)$

On veut évaluer

$$a_5.x^5 + a_4.x^4 + a_3.x^3 + a_2.x^2 + a_1.x^1 + a_0$$

On somme les puissances impaires dans y_i et les paires dans y_p.

$$y_i \leftarrow a_5 \Rightarrow y_i \leftarrow y_i.x^2 \Rightarrow y_i \leftarrow y_i + a_3 \Rightarrow y_i \leftarrow y_i.x^2 \Rightarrow y_i \leftarrow y_i + a_1$$
$$y_p \leftarrow a_4 \Rightarrow y_p \leftarrow y_p.x^2 \Rightarrow y_p \leftarrow y_p + a_4 \Rightarrow y_p \leftarrow y_p.x^2 \Rightarrow y_p \leftarrow y_p + a_0$$

Si on veut évaluer

$$a_6.x^6 + a_5.x^5 + a_4.x^4 + a_3.x^3 + a_2.x^2 + a_1.x^1 + a_0$$

On se ramène au cas précédent en posant $a_7 = 0$.

Le code C

```
double horner2(double *a, double x, int n)
{
  double yi, yp, x2;
  int i;
  x2 = x*x;
  if (n%2 != 0) // ou "if (n/2)"
    { yi = a[n]; yp = a[n-1]; i = n-2; }
  else
    { yi = 0; yp = a[n]; i = n-1; };
  for(; i >= 0; i -= 2)
    {
      yi = yi*x2 + a[i];
      yp  = yp*x2 + a[i-1];
    };
  return yp + x*yi;
}
```

Evaluation d'un puissance X^n

$$n = \sum_{i=0}^{k} n_i 2^i = (...(2+n_{k-1})*2+...+n_1)*2+n_0, \; n_i = 0 \text{ ou } 1 \text{ et } n_k = 1$$

$\Rightarrow$

$$X^n = (...(X^2 \times X^{n_{k-1}})^2 \times ... \times X^{n_1})^2 \times X^{n_0}$$

Entrée $X \in \mathbb{R}$ et un entier $n = \sum_{i=0}^{k} n_i 2^i$

Sortie $p = X^n$

Corps $p \leftarrow X$

Pour $i = k - 1$ **à** 0 **faire**

$p \leftarrow p * p$

si $n_i = 1$ **alors** $p \leftarrow p * X$

Stockage d'un polynôme

1. stockage des $n+1$ coefficients dans un tableau de dimension $n+1$.
   On parle de *représentation dense*.

2. On stocke les couples (coefficient, indice)
   On parle de *représentation creuse*.

Exemple : $P(X) = 7.X^{20} - 3.X^{14} + 10.X^{8} + 5.x^{2} - 1$

Stockage 1 : $(7, 0, 0, 0, 0, 0, -3, 0, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 5, 0, -1)$

Stockage 2 : $((7, 20), (-3, 14), (10, 8), (5, 2), (-1, 0))$

Fin

Le polynôme interpolateur de Lagrange

Soit une fonction réelle f sur un intervalle $I = [a, b]$ et $n + 1$ points deux à deux distincts $x_i, i = 0..n$ de I alors

Théorème : Il existe un unique polynôme $P_I(X)$ de degré inférieur ou égal à n tel que

$$\forall\ 0 \leq i \leq n, P_f(x_i) = f(x_i).$$

Il est défini par

$$P_f(X) = \sum_{i=0}^{n} f(x_i).L_i(X)$$

avec

$$L_i(X) = \frac{\prod_{j=0,\neq i}^{n}(X - x_j)}{\prod_{j=0,\neq i}^{n}(x_i - x_j)}$$

C'est le *polynôme interpolateur de Lagrange*.

Démonstration

Remarquons d'abord que $L_i(x_j) = \delta_{ij}$. En effet :

$$L_i(x_j) = \frac{\prod_{k=0,\neq i}^{n}(x_j - x_k)}{\prod_{k=0,\neq i}^{n}(x_i - x_k)} = \delta_{ij}$$

donc

$$P_f(x_i) = \sum_{k=0}^{n} f(x_k).L_k(x_i) = f(x_i)$$

Si $\exists\ Q \in \mathbb{R}[x]$ de degré au plus n tel que
$\forall\ 0 \leq i \leq n, Q(x_i) = f(x_i)$ alors

$$\forall\ 0 \leq i \leq n, (Q - P_f)(x_i) = 0$$

donc $Q - P_f$ est un polynôme de degré inférieur ou égal à n qui amdet $n+1$ racines donc $Q - P_f = 0$ donc P_f est unique.

Calcul de l'erreur

Théorème : Soit f une fonction C^{n+1} sur $[a, b]$ et $n+1$ points $(x_0, x_1, ..., x_n)$ de I. Soit P_f le polynôme interpolateur de Lagrange sur les x_i. Alors

$\forall x \in [a, b], \exists c \in [a, b]$ tel que

$$f(x) - P_f(x) = \frac{f^{(n+1)}(c)}{(n+1)!} . \prod_{i=0}^{n} (x - x_i)$$

Remarque : rien ne garantit que l'erreur tende vers 0 quand n tend vers $+\infty$.

Démonstration

Soit $x_t \in [a, b]$ tel que $\forall\ 0 \leq i \leq n, x_t \neq x_i$ alors $\exists\ A \in \mathbb{R}$ tel que

$$g(x_t) = f(x_t) - P_f(x_t) - \frac{A}{(n+1)!}.\prod_{i=0}^{n}(x_t - x_i) = 0$$

donc $g(x) = f(x) - P_f(x) - \dfrac{A}{(n+1)!}.\displaystyle\prod_{i=0}^{n}(x - x_i)$ admet $x_0, x_1, ..., x_n, x_t$ comme racines soit $n+2$ racines.

En appliquant le théorème de Rolle à $g(x)$ et à ses dérivées successives, $g^{(n+1)}(x)$ admet une racine dans $[a, b]$. Or

$$g^{(n+1)}(x) = f^{(n+1)}(x) - A$$

cqfd.

Calcul du polynôme de Lagrange

Soit $P_f(X) = a_n.x^n + a_{n-1}.X^{n-1} + ... + a_1.X + a_0$,

$$\forall 0 \leq i \leq n, P_f(x_i) = f(x_i) = y_i$$

$$\Leftrightarrow$$

$$\underbrace{\begin{pmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & \dots & x_n^n \end{pmatrix}}_{V} \times \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ \\ a_n \end{pmatrix}}_{P} = \underbrace{\begin{pmatrix} y_0 \\ y_1 \\ \\ y_n \end{pmatrix}}_{Y}$$

V est une matrice de Vandermonde. Sur ordinateur, on résout le système linéaire.

À la main, on peut résoudre le système linéaire ou utiliser la formule

$$P_f(X) = \sum_{i=0}^{n} f(x_i).L_i(X)$$

Fin

Le polynôme des moindres carrés

Le problème d'interpolation

Une fonction f est définie sur un intervalle $I = [a, b]$ et connue en $n + 1$ points $(x_0, x_1, ..., x_n)$ de I. On cherche à approcher f sur I par un polynôme P en minimisant une norme :

$$\underset{x \in [a,b]}{Sup} |f(x) - P(x)|$$

$$\sum_{k=0}^{n} (f(x_k) - P(x_k))^2$$

$$\frac{1}{b-a} \int_a^b (f(x) - P(x))^2 dx$$

On va s'intéresser à $\sum_{k=0}^{n} (f(x_k) - P(x_k))^2$ et on cherche une solution dans $\mathbb{R}_p[X]$, ensemble des polynômes réels de degré inférieur ou égal à p.

Cas où $p < n$

L'objectif est de minimiser $v(P) = \sum_{k=0}^{n}(f(x_k) - P(x_k))^2$ sur $\mathbb{R}_p[X]$.

Remarque : Si $p \geq n$, la solution est le polynôme interpolateur de Lagrange.

Soit $\langle P|Q\rangle = \sum_{i=0}^{n} P(x_i)Q(x_i)$. C'est un produit scalaire sur $\mathbb{R}_n[X]$ (forme bilinéaire symétrique définie positive) :

$$\langle P|Q\rangle = \langle Q|P\rangle$$

$$\langle \lambda_1.P_1 + \lambda_2.P_2|Q\rangle = \lambda_1. \langle P_1|Q\rangle + \lambda_2. \langle P_2|Q\rangle \text{ (idem à droite)}$$

$$\langle P|P\rangle > 0 \text{ si } P \neq 0$$

Si $p < n$ alors $\mathbb{R}_p[X]$ est un sous espace vectoriel de $\mathbb{R}_n[X]$.

Si $P_f(X)$ est le polynôme interpolateur de f sur les x_i alors

$$v(P) = \sum_{k=0}^{n} (f(x_k) - P(x_k))^2 = \sum_{k=0}^{n} (P_f(x_k) - P(x_k))^2 = \langle P_f - P | P_f - P \rangle$$

Si $p < n$ alors $\mathbb{R}_p[X]$ est un sous espace vectoriel de $\mathbb{R}_n[X]$.

Si $P_f(X)$ est le polynôme interpolateur de f sur les x_i alors

$$v(P) = \sum_{k=0}^{n}(f(x_k) - P(x_k))^2 = \sum_{k=0}^{n}(P_f(x_k) - P(x_k))^2 = \langle P_f - P | P_f - P \rangle$$

Théorème : Si $P_f(X)$ est le polynôme interpolateur de Lagrange de f sur les x_i, le polynôme P_m qui minimise $v(P)$, dit *polynôme des moindres carrés*, est la projection orthogonale de P_f sur $\mathbb{R}_p[X]$ relativement au produit scalaire $\langle P|Q \rangle = \sum_{i=0}^{n} P(x_i)Q(x_i)$.

Construction du polynôme des moindres carrés

On commence par construire une base orthonormée Y_i pour le produit scalaire $\langle P|Q \rangle = \sum_{i=0}^{n} P(x_i)Q(x_i)$ à partir de la base canonique X^i (procédé d'orthonormalisation de Gram-Schmidt).

On commence par Y_0 et $Y_0 = \frac{1}{\sqrt{n+1}}$

Y_1 doit vérifier $\langle Y_1|Y_0 \rangle = 0$ et $\langle Y_1|Y_1 \rangle = 1$ donc si $Z_1 = X - \langle X|Y_0 \rangle .Y_0$

$$Y_1 = Z_1/||Z_1||$$

Si $Y_0, Y_1, ..., Y_{k-1}$ sont construits,

Soit $Z_k = X^k - \sum_{i=0}^{k-1} \left\langle Y_i|X^k \right\rangle .Y_i$, alors

$$Y^k = Z_k/||Z_k||$$

Au final

$$P_m = \sum_{i=0}^{p} \langle P_f|Y_i \rangle .Y_i$$

Le code C

On a besoin de

```
float horner(float *pol, float x, int n)
{
  float y;
  ......
  return y;
}

float prodscal(float *p, int np, float *q, int nq, float *x, int n)
{
  float y=0;
  for(i = 0; i <= n; i++)
    y += horner(p, x[i], np) * horner(q, x[i], nq);
  return y;
}

float prodscal2(float *p, int np, int k, float *x, int n)
{
  float y=0;
  for(i = 0; i <= n; i++)
    y += horner(p, x[i], np) * pow(x[i],k);
  return y;
}
```

Le code C : le calcul des Y_k

$$Z_k = X^k - \sum_{i=0}^{k-1} \left\langle Y_i | X^k \right\rangle . Y_i \text{ et } Y^k = Z_k / ||Z_k||$$

Les Y_k sont stockés comme les lignes d'une matrice à p lignes et n colonnes définie comme un vecteur.

```
float *y
*y = 1.0/sqrt(n+1);
for(k = 1; k <= p; k++)
  {
    for(j = 0; j < k; j++)
      *(y + n*k + j) = 0;
    *(y + n*k + k) = 1
    for(i = 0; i <= k; i++)
      {
        aux = prodscal2(y + n*i, i, k, x, n);
        for (j = 0; j < k; j++)
          *(y + n*k + j) -= aux * *(y + n*i + j)
      }
    aux = sqrt(prodscal(y + n*k, k, y + n*k, k, x, n));
    for(j = 0; j <= k; j++)
      *(y + n*k + j) /= aux;
  }
```

Le code C : le calcul de P_m

$$P_m = \sum_{i=0}^{p} \langle P_f | Y_i \rangle . Y_i$$

```
//Calcul de P_f
  .................
for(i = 0; i <= p; i++)
  {
    aux = prodscal(pf, n, *(y + n*i), i, x, n);
    for(j = 0; j <= p; j++
      *(p+j) += aux * *(y + n*i + j);
  }
```

Fin