Chapitre 5

Interpolation polynômiale et extrapolation

Sommaire

5.1 Interpolation de Lagrange 2
5.1.1 Formulation barycentrique 4
5.1.2 Formule de Newton 5
5.1.3 estimation d'erreur 7
5.1.4 Convergence de p_n vers f 10
5.2 Interpolation d'Hermite 11
5.3 Interpolation par morceaux 12
5.3.1 Interpolation affine 12
5.3.2 Interpolation par fonctions splines 13

Deux problèmes classiques

1) Interpolation : on considère une aile d'avion, qu'on soumet à des vents de 10, 50, 100, 200 km/h, et dont on calcule les déformations pour ces valeurs. On veut savoir comment elle résistera à un vent de 150km/h.

2) Extrapolation : on connaît la population française de 1800 à 2010 et on veut en déduire une estimation de la population française dans les 10 prochaines années.

Une solution est de déterminer un polynôme dont la courbe s'approche le plus possible (ou passe par) ces points, et de prendre sa valeur aux nouveaux points. C'est le but de ce chapitre.

Le problème mathématique est le suivant : on se donne $n+1$ mesures $f_0, \cdots, f_n$ en $n+1$ points distincts $x_0, \cdots, x_n$ et on cherche à calculer un polynôme q de degré

inférieur ou égal à m, avec $m \leq n$, qui "approche" les mesures $f_0, \cdots, f_n$. La première approche est quand $m = n$: c'est le polynôme d'interpolation.

5.1 Interpolation de Lagrange

Théorème 5.1 *1) Il existe un unique polynôme $p_n \in \mathbf{P}_n$ (espace vectoriel des polynômes de degré inférieur ou égal à n) tel que*

$$\forall i,\ 0 \leq i \leq n, \quad p_n(x_i) = f_i. \tag{5.1}$$

2) Il s'écrit sous la forme

$$p_n(x) = \sum_{i=0}^{n} f_i \ell_i(x), \quad \textit{avec} \quad \ell_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}. \tag{5.2}$$

Les ℓ_i sont les polynômes d'interpolation de Lagrange. p_n est le polynôme d'interpolation aux points x_i pour les mesures f_i.

Démonstration 1)Notons $p_n(x) = \sum_{k=0}^{n} a_k x^k,\ x \in \mathbb{R}$. Résoudre (5.1) est équivalent à résoudre un système linéaire dont les inconnues sont les coefficients a_k :

$$Ay = b \text{ avec}$$

$$A = \begin{pmatrix} 1 & x_0 & \cdots & x_0^n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{pmatrix}, \quad y = \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix}, \quad b = \begin{pmatrix} f_0 \\ \vdots \\ f_n \end{pmatrix},$$

A est une matrice de Vandermonde. Elle est inversible ce qui conclut la partie 1).

2) ℓ_i est un polynôme de $\mathbf{P}_n$, et vérifie $\ell_i(x_j) = \delta_{ij}$. On vérifie que ce polynôme convient. ■

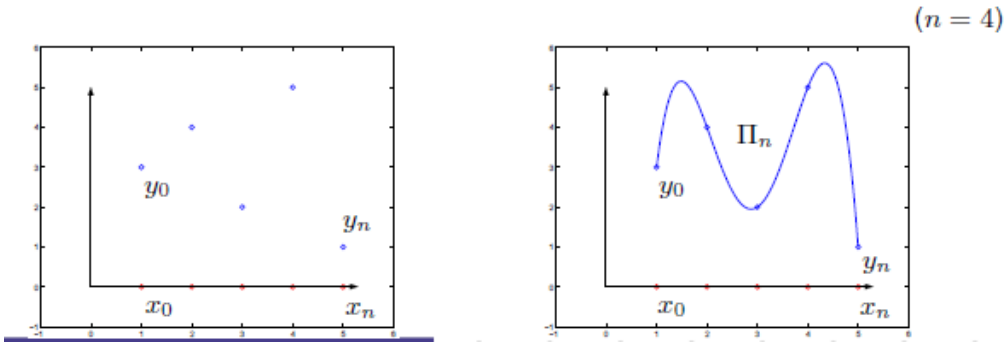

FIGURE 5.1 – polynôme d'interpolation

Lorsque les fi sont les valeurs d'une certaine fonction f aux points x_i, on parle de p_n comme de l'interpolant de f et on la note $\Pi_n f$.

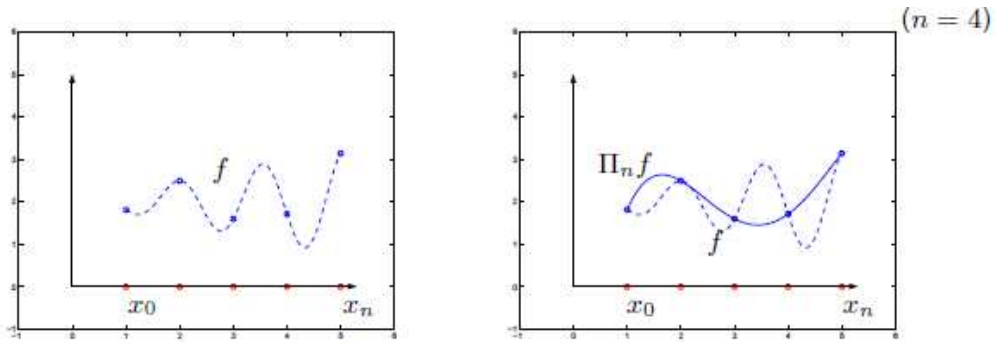

FIGURE 5.2 – interpolant

En principe il suffit de résoudre le système linéaire pour calculer les a_i, puis de calculer en chaque nouveau point x

$$p_n(x) = (a_0, a_1, \ldots, a_n) * \begin{pmatrix} 1 \\ x \\ \vdots \\ x^n \end{pmatrix}$$

Mais le système est très mal conditionné. Il vaut mieux programmer directement (5.2).

```
function [yy] = lagint(x, y, xx)
% LAGINT uses the points (x_i, y_i) for the Lagrange Form of the
% interpolating polynomial and interpolates the values
% yy_i = p_n(xx_i)
n = length(x); nn = length(xx);
for i = 1:nn,
    yy(i) = 0;
    for k =1:n
        yy(i) = yy(i)+y(k)*prod((xx(i) - x([1:k-1,k+1:n])))...
        /prod((x(k) - x([1:k-1,k+1:n])));
    end;
end;
```

5.1.1 Formulation barycentrique

Utiliser la formulation (5.2) mène à $\mathcal{O}(n^2)$ opérations pour chaque x. Nous définissons les coefficients

$$\lambda_i = \frac{1}{\prod_{j \neq i}(x_i - x_j).}$$

et nous réécrivons

$$p_n(x) = \sum_{i=0}^{n} \lambda_i \left(\prod_{j \neq i} (x - x_j) f_i \right) = \prod_j (x - x_j) \left(\sum_{i=0}^{n} \frac{\lambda_i}{x - x_i} f_i \right)$$

Puisque la formule est exacte pour les polynômes de degré 0, on peut écrire pour $f \equiv 1$:

$$1 = \prod_j (x - x_j) \left(\sum_{i=0}^{n} \frac{\lambda_i}{x - x_i} \right)$$

et donc

$$\prod_j (x - x_j) = \frac{1}{\sum_{i=0}^{n} \frac{\lambda_i}{x - x_i}}$$

ce qui nous donne la formule barycentrique

$$p_n(x) = \frac{\sum_{i=0}^{n} \frac{\lambda_i}{x - x_i} f_i}{\sum_{i=0}^{n} \frac{\lambda_i}{x - x_i}}$$

$$p_n(x) = \frac{\displaystyle\sum_{i=0}^{n} \frac{\lambda_i}{x - x_i} f_i}{\displaystyle\sum_{i=0}^{n} \frac{\lambda_i}{x - x_i}}$$

Pour l'utiliser nous calculons d'abord les λ_i en $\mathcal{O}(n^2)$ opérations,

```
function [lambda] = coeffbary(x)
% COEFFBARY computes the coefficients for the barycentric
% representation of the interpolating polynomial through
% the points (x_i, y_i)
n = length(x); x=x(:);
for k = 1:n,
lambda(k) = 1 / prod(x(k) - x([1:k-1,k+1:n]));
end;
```

Puis pour chaque x nous calculons les poids $\mu_i = \dfrac{\lambda_i}{x - x_i}$ et $p_n(x) = \dfrac{\sum_{i=0}^{n} \mu_i f_i}{\sum_{i=0}^{n} \mu_i}$ en seulement $\mathcal{O}(n)$ opérations.

```
function [yy] = intbary(x, y,lambda, xx)
%% INTBARY evaluates the interpolating polynomial
%% through (x_i, y_i) for the values xx: yy = P_n(xx)
x=x(:); y=y(:); xx = xx(:);
nn = length(xx);
for i = 1:nn,
z = (xx(i)-x) + 1e-30; % prevents a division by zero
mue=lambda'./z;
yy(i)=mue'*y/sum(mue);
end;
```

5.1.2 Formule de Newton

On se donne les $n+1$ points $x_0, \cdots, x_n$. Pour tout k plus petit que n, on note p_k le polynôme d'interpolation de f aux points $x_0, \cdots, x_k$. On a

$$p_k - p_{k-1} = C(x - x_0) \cdots (x - x_{k-1})$$

Définition 5.1 *Pour $k+1$ points $y_0, \cdots, y_k$, on note $f[y_0, \cdots, y_k]$ le coefficient de degré k du polynôme d'interpolation de f aux points $y_0, \cdots, y_k$.*

Lemme 5.1

$$p_k - p_{k-1} = f[x_0, \cdots, x_k](x - x_0) \cdots (x - x_{k-1})$$

Démonstration

■

Théorème 5.2 (Formule de Newton)

$$p_n(x) = f(x_0) + \sum_{k=0}^{n} f[x_0, \cdots, x_k](x - x_0) \cdots (x - x_{k-1}) \tag{5.3}$$

Démonstration Il suffit de sommer la formule de récurrence précédente. ■

Lemme 5.2 (Formule des différences divisées)

$$\forall k \geq 1, f[x_0, \cdots, x_k] = \frac{f[x_1, \cdots, x_k] - f[x_0, \cdots, x_{k-1}]}{x_k - x_0} \tag{5.4}$$

Démonstration
Soit $q_{k-1} \in \mathbf{P}_{k-1}$ le polynôme d'interpolation de f aux points $x_1, \ldots, x_k$. Posons

$$\tilde{p}_k = \frac{(x - x_0)q_{k-1} - (x - x_k)p_{k-1}}{x_k - x_0}$$

Alors $\tilde{p}_k = p_k$. En effet

et il ne reste plus qu'à égaler les coefficients directeurs dans la formule de $\tilde{p}_k$. ■

Table de calcul

$$\begin{array}{llllll}
x_0 & f(x_0) = f[x_0] & & & & \\
x_1 & f(x_1) = f[x_1] & f[x_0, x_1] & & & \\
x_2 & f(x_2) = f[x_2] & f[x_1, x_2] & f[x_0, x_1, x_2] & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \\
x_n & f(x_n) = f[x_n] & f[x_{n-1}, x_n] & f[x_{n-2}, x_{n-1}, x_n] & \cdots & f[x_0, \ldots, x_n]
\end{array}$$

FIGURE 5.3 – table des différences divisées

Voici l'algorithme matlab

```
function [d,D] = coeffnewton(x, y)
% COEFFNEWTON computes the divided differences needed for
% constructing the interpolating polynomial through (x_i,y_i)
n = length(x)-1; % degree of interpolating polynomial
The Interpolation Polynomial 335
% divided differences
for i=1:n+1
D(i,1) = y(i);
for j = 1:i-1
D(i,j+1) = (D(i,j)-D(i-1,j))/(x(i)-x(i-j));
end
end
d = diag(D);
```

Une fois les d_i calculés, pour les utiliser nous couplons avec l'algorithme de Hörner, en réécrivant le polynôme p_n sous la forme

$$p_n(x) = d_0 + (x - x_0)(d_1 + (x - x_1)(d_2 + \cdots + (x - x_{n-2})(d_{n-1} + (x - x_{n-1})d_n)))$$

```
function y = intnewton(x,d,z)
% INTNEWTON evaluates the Newton interpolating polynomial
% at the new points z: y = P_n(z) using the Horner form
% and the diagonal d of the divided difference scheme.
n = length(x)-1;
y = d(n+1)
for i= n:-1:1
y = y.*(z-x(i))+d(i);
end;
```

En Matlab, on utilise la fonction *polyfit* pour l'interpolation polynomiale. Cette fonction utilise une interpolation au sens des moindres carrés discrets (voir partie 3).

5.1.3 estimation d'erreur

Théorème 5.3 *si* $f \in \mathcal{C}^{n+1}([a,b])$, $\forall x \in [a,b]$, $\exists \zeta_x$ *appartenant au plus petit intervalle ouvert contenant* $x, x_0, \cdots, x_n$, *tel que*

$$f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\zeta_x) \Pi_{n+1}(x) \tag{5.5}$$

où $\Pi_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$.

Démonstration On remarque d'abord que l'égalité est vraie si x est l'un des x_i. On suppose ensuite que x est fixé, non égal à l'un des x_i. On applique le théorème de Rolle $n+1$ fois à la fonction définie pour x fixé par

$$F(t) = f(t) - p_n(t) - C\Pi_n(t)$$

où C est défini par $F(x) = 0$. ■

On déduit de ce théorème d'abord que les différences divisées sont des approximations des dérivées : il existe un ξ dans l'intervalle $(inf(x_i), sup(x_i))$ tel que

$$f[x_0, \cdots, x_n] = \frac{f^{(n)}(\zeta)}{(n)!}$$

En effet écrivons d'après le lemme 5.1

$$p_n(x_n) - p_{n-1}(x_n) = f[x_0, \cdots, x_n](x_n - x_0) \cdots (x_n - x_{n-1})$$

et d'après l'estimation d'erreur aux points $x_0, \ldots, x_{n-1}$,

$$f(x_n) - p_{n-1}(x_n) = \frac{1}{(n)!} f^{(n)}(\zeta)(x_n - x_0) \cdots (x_n - x_{n-1}).$$

Egalons ces deux expressions pour conclure.

Nous en déduisons aussi une estimation d'erreur grossière. On note pour une fonction φ, $\|\varphi\|_\infty = \sup_{x\in[a,b]} |\varphi(x)|$, et on a

$$\|f - p_n\|_\infty \leq \frac{1}{(n+1)!} \|F^{n+1}\|_\infty \|\Pi_{n+1}\|_\infty \tag{5.6}$$

Pour une fonction f donnée, on minimise l'erreur en choisissant bien les points d'interpolation :

Théorème 5.4 *Sur un intervalle* $[a, b]$, $\|\Pi_{n+1}\|_\infty$ *est minimale pour le choix des points*

$$x_i^T = \frac{a+b}{2} + \frac{b-a}{2} y_i^{n+1}, \quad 0 \leq i \leq n$$

Les $y_i^{n+1} = \cos(\frac{2i+1}{2(n+1)}\pi)$ *sont les zéros du polynôme de Chebyshev* T_{n+1}.

Polynômes de Chebyshev

Pour tout k on définit sur $[-1,1]$ la fonction

$$T_k(y) = \cos(k\mathrm{Arc}\cos y). \tag{5.7}$$

T_k est en fait un polynôme de degré k. Pour le voir, on établit la formule de récurrence

$$T_{k+1}(y) = 2y\,T_k - T_{k-1}, \quad T_0 = 1, \quad T_1 = y, \tag{5.8}$$

ce qui permet de les définir sur tout $\mathbb{R}$. Le coefficient dominant de T_k est $2^{k-1}y^k$, ses zéros sont les $y_i^k = \cos\left(\frac{2i+1}{2k}\pi\right)$ pour $0 \le i \le k-1$, et les extrema valent $(-1)^i$, atteints aux points $\tilde{y}_i^k = \cos\left(\frac{i}{k}\pi\right)$ pour $0 \le i \le k$.

Lemme 5.3 *Pour tout $p \in \mathbf{P}_k$ unitaire (i.e. $p = y^k + \cdots$), on a*

$$\|p\|_\infty \ge \|\frac{T_k}{2^{k-1}}\|_\infty = \frac{1}{2^{k-1}}$$

Démonstration Pour n'importe quel choix des points d'interpolation $x_0, \cdots, x_n$, faisons un changement de variable

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}y_i,\ 0 \le i \le n, \quad x = \frac{a+b}{2} + \frac{b-a}{2}y$$

Lorsque x varie dans l'intervalle $[a,b]$, y varie dans l'intervalle $[-1,1]$ et

$$\Pi_{n+1}(x) = \left(\frac{b-a}{2}\right)^{n+1} \prod_{i=0}^{n}(y - y_i)$$

Minimiser l'erreur est donc minimiser la norme infinie d'un polynôme unitaire sur $(-1,1]$, et

$$\inf_{\{x_i\}} \|\Pi_{n+1}\|_\infty = \|\prod(x - x_i^T)\|_\infty = 2\left(\frac{b-a}{4}\right)^{n+1}$$

Admis, cf Demailly, *analyse numérique et équations différentielles.* ■

Remarque 5.1 *Pour une division en points équidistants, i.e. $x_j = a + \frac{b-a}{n}j$, on montre que*

$$\|\Pi_{n+1}\|_\infty \sim (b-a)^{n+1}\frac{e^{-n}}{\sqrt{n}\ln n}$$

Pour $(a,b) = (-1,1)$, la figure suivante montre le logarithme des erreurs en fonction de n dans les deux cas

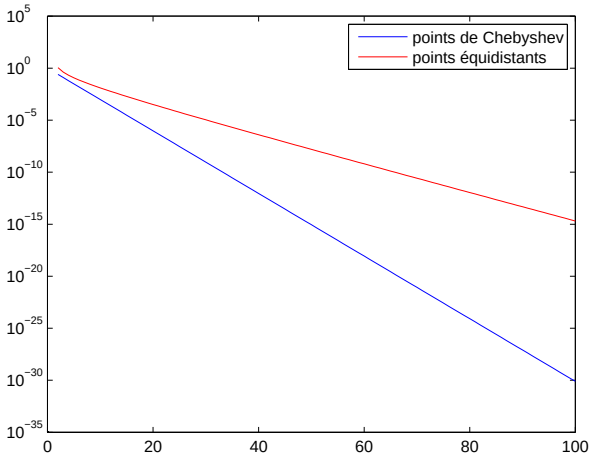

FIGURE 5.4 – Comparaison de $\|\Pi_{n+1}\|_\infty$ pour deux ensembles de points

5.1.4 Convergence de p_n vers f

Considérons la fonction $f(x) = \frac{x+1}{5}\sin(x)$ sur [0,6] (ex de Quarteroni).

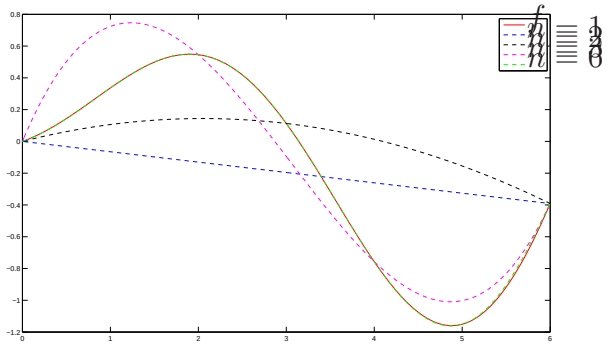

FIGURE 5.5 – interpolant

On constate dans ce cas que la suite $\Pi_n(f)$ converge vers f. Ce n'est pas vrai en général. Le contre-exemple classique est celui de la fonction de Runge $f(x) = \dfrac{1}{1+15x^2}$:

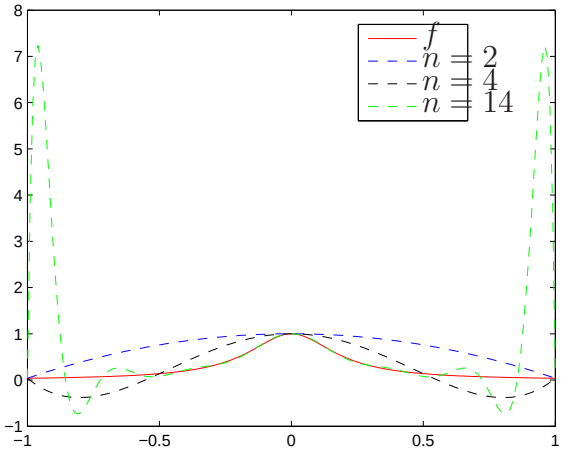

FIGURE 5.6 – interpolant

Bien que la fonction soit tout à fait régulière, on voit que l'erreur en 1 tend vers l'infini.

5.2 Interpolation d'Hermite

f est toujours une fonction suffisamment régulière sur le segment $[a, b]$. On se donne $k+1$ points $x_0, \cdots, x_k$ dans $[a, b]$.

Théorème 5.5 *Posons $n = 2k+1$. Il existe un et un seul polynôme $p_n \in \mathbf{P}_n$ tel que*

$$\forall j,\ 0 \leq j \leq k, \quad p_n(x_j) = f(x_j) \text{ et } p_n'(x_j) = f'(x_j).$$

Théorème 5.6 *si $f \in \mathcal{C}^{n+1}([a, b])$, $\forall x \in [a, b]$, $\exists \zeta_x$ appartenant au plus petit intervalle ouvert contenant $x, x_0, \cdots, x_k$, tel que*

$$f(x) - p_n(x) = \frac{1}{(n+1)!} F^{n+1}(\zeta_x) \Pi_{n+1}(x) \tag{5.9}$$

où $\Pi_{n+1}(x) = \prod_{i=0}^{k} (x - x_i)^2$.

p_n dépend de $2k+2$ coefficients, nous allons l'exprimer sous la forme

$$p_n(x) = \sum_{i=0}^{k} f(x_i) q_i(x) + \sum_{i=0}^{k} f'(x_i) r_i(x) \tag{5.10}$$

où les polynômes q_i et r_i sont définis par

$$\left| \begin{array}{l} q_i(x_j) = \delta_{ij}, \\ q_i'(x_j) = 0, \end{array} \right. \quad \left| \begin{array}{l} r_i(x_j) = 0, \\ r_i'(x_j) = \delta_{ij}. \end{array} \right. \tag{5.11}$$

On peut les déterminer en fonction des polynômes d'interpolation de Lagrange ℓ_i :

$$q_i(x) = (1 + 2(x_i - x)\ell_i'(x_i))\ell_i^2(x), \quad r_i(x) = (x - x_i)\ell_i^2(x).$$

5.3 Interpolation par morceaux

Soient $a \equiv a_0 < a_1 < \cdots < a_N \equiv b$ des points qui divisent l'intervalle $I = [a, b]$ en sous-intervalles $I_j = [a_j, a_{j+1}]$ de longueur $H = \frac{b-a}{N}$, soit $a_j = a + jH$.

5.3.1 Interpolation affine

Sur chaque intervalle I_j, on interpole f par un polynôme de degré inférieur ou égal à 1. On obtient un polynôme par morceaux, noté $\Pi_1^H f$. Il s'écrit

$$\Pi_1^H f(x) = f(a_j) + f[a_j, a_{j+1}](x - a_j), \quad x \in I_j$$

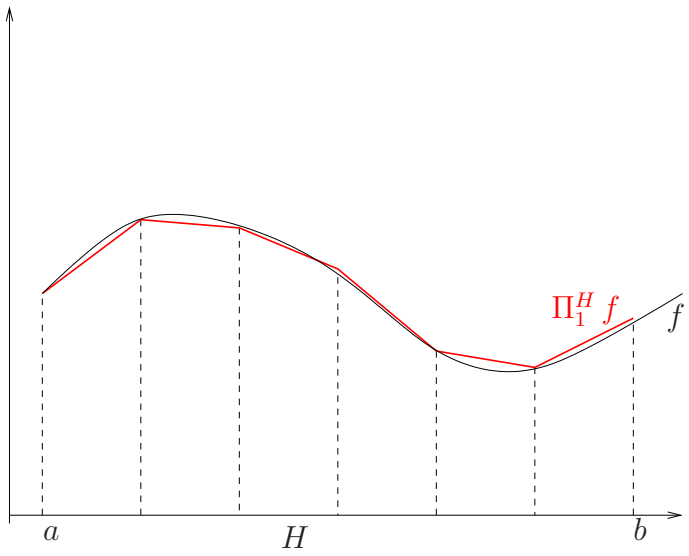

FIGURE 5.7 – interpolation affine par morceaux

Théorème 5.7 *Si $f \in \mathcal{C}^2(I)$, alors*

$$\sup_{x \in I} |f(x) - \Pi_1^H f(x)| \leq \frac{H^2}{8} \sup_{x \in I} |f"(x)|. \tag{5.12}$$

Démonstration Il suffit d'appliquer l'estimation d'erreur (5.6). ■

Remarque 5.2 *Si $f \in \mathcal{C}^{n+1}(I)$, on peut faire de même une interpolation par des polynômes de degré inférieur ou égal à n dans chaque sous-domaine et on obtient l'estimation d'erreur*

$$\sup_{x\in I} |f(x) - \Pi_n^H f(x)| \leq \frac{H^{n+1}}{4(n+1)} \sup_{x\in I} |f^{(n+1)}(x)|. \tag{5.13}$$

5.3.2 Interpolation par fonctions splines

L'inconvénient de la démarche précédente est que l'approximation de f manque de régularité. Ici nous nous donnons $y_i = f(a_i)$ et aussi des valeurs y_i' que nous choisirons ensuite. Dans chaque sous-intervalle, nous interpolons la fonction f par un polynôme $p_i \in \mathbf{P}_3$ tel que

$$p_i(a_i) = y_i,\ p_i(a_{i+1}) = y_{i+1}, \quad p_i'(a_i) = y_i',\ p_i'(a_{i+1}) = y_{i+1}',$$

Faisons le changement de variable $y = (x - a_i)/H$, et posons $p(x) = P(y)$. On doit donc avoir

$$P_i(0) = y_i, \quad P_i(1) = y_{i+1}; P_i'(0) = Hy_i', \quad P_i'(1) = Hy_{i+1}';$$

Nous utilisons les formules données pour les polynômes d'Hermite.

$$P_i = y_i q_0 + y_{i+1} q_1 + y_i' r_0 + y_{i+1}' r_1$$

Les polynôme de Lagrange aux points 0 et 1 sont $\ell_0 = 1-y$, $\ell_1 = y$, et les polynômes q_i et r_i sont donnés par

$$q_0(y) = (2y-1)(1-y)^2,\ q_0(y) = (2y-1)y^2, \quad r_0(y) = y(1-y)^2,\ r_i(y) = (1-y)y^2.$$

Comment maintenant calculer la valeur de p_i en un point x ?

1. Déterminer l'intervalle $[a_i, a_{i+1}]$ où se trouve x.
2. Calculer la variable locale $y = (x - a_i)/H$.
3. Évaluer $P_i(y)$, de préférence par l'algorithme de Hörner.

Pour déterminer l'intervalle où se trouve x, on utiliser un algorithme de recherche binaire si les intervalles ne sont pas de même taille. Sinon bien sûr on prend la partie entière de x/H.

Les y_i' doivent approcher les dérivées $f'(a_i)$ qui ne sont pas données en général. On peut alors approcher par exemple $f'(a_i)$ par des différences divisées $y_i' = f[a_{i-1}, a_{i+1}]$ pour $1 \leq i \leq N-1$. Aux deux extrémités on peut prendre des dérivées décentrées $y_0' = f[a_0, a_1]$ et $y_N' = f[a_{N-1}, a_N]$.

Peut-on déterminer les y_i' de façon à être encore plus régulier ? Par exemple que les dérivées secondes soient aussi continues ? La réponse est oui, ce sont les vrais splines cubiques historiques.