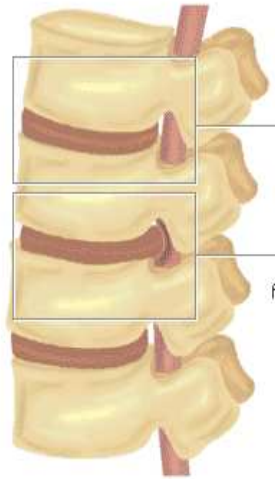
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Interpolation

Prof. Alfio Quarteroni

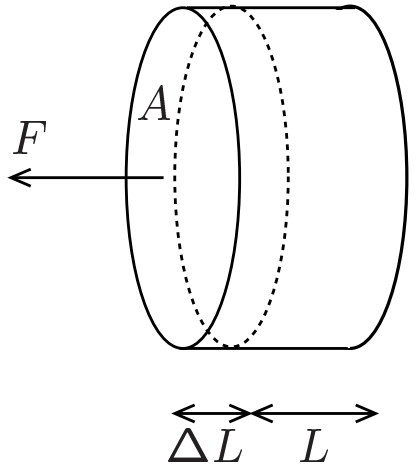
Exemples et motivations

Exemple 1. On considère un test mécanique pour établir le lien entre contrainte ($MPa = 100N/cm^2$) et déformations relatives (cm/cm) d'un échantillon de tissu biologique (disque intervertébral, selon P. Komarek, Ch. 2 de *Biomechanics of Clinical Aspects of Biomedicine*, 1993, J. Valenta ed., Elsevier).



disques intervertébraux

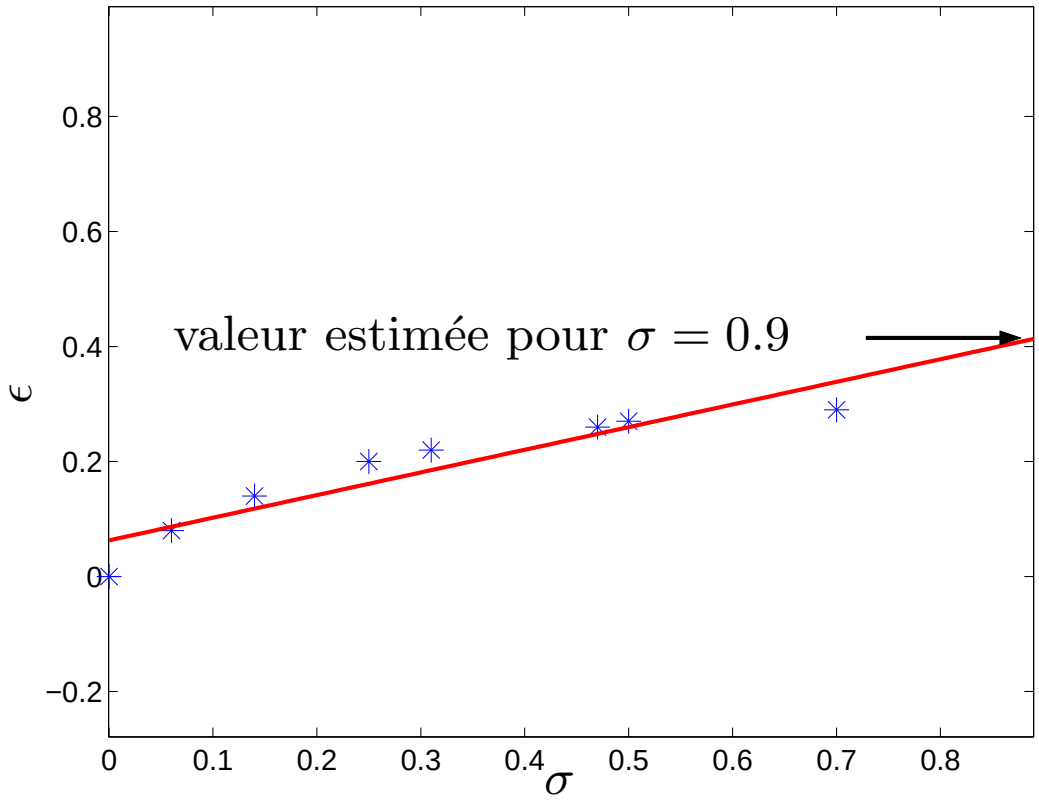
test i	contrainte σ	déformation ϵ
1	0.00	0.00
2	0.06	0.08
3	0.14	0.14
4	0.25	0.20
5	0.31	0.23
6	0.47	0.25
7	0.60	0.28
8	0.70	0.29



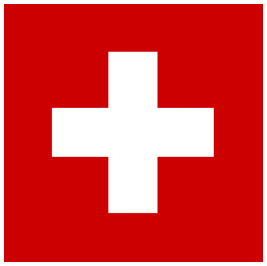
$$\sigma = F/A$$
$$\epsilon = \Delta L/L$$

A partir des ces données, on veut estimer la déformation correspondante à un effort $\sigma = 0.9$ MPa.

A travers la méthode des moindres carrés on obtient que la droite qui approche mieux ces données est $p(x) = 0.3938x - 0.0629$. On peut utiliser cette droite (dite *de régression*) pour estimer ϵ lorsque $\sigma = 0.9$ MPa: on trouve $p(0.9) \simeq 0.4$.



Exemple 2. La population de la Suisse dans les années 1900 à 2000 est recensée comme il suit (en milliers),

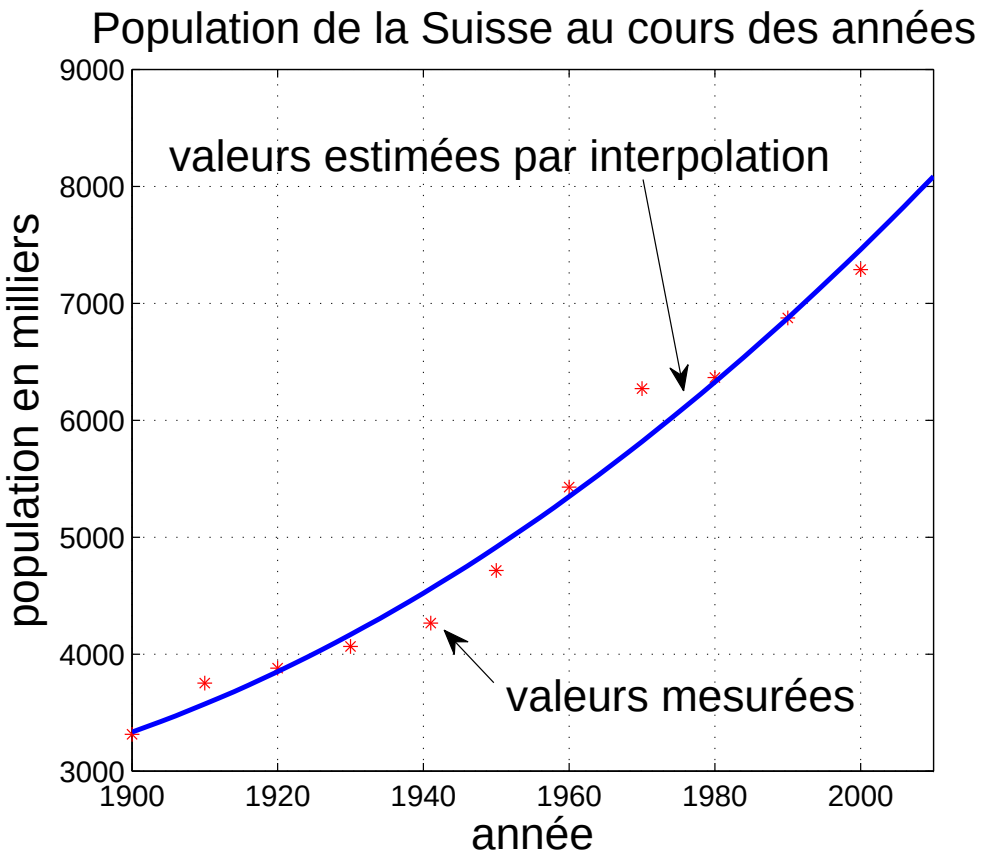


année	1900	1910	1920	1930	1941	1950
population	3315	3753	3880	4066	4266	4715

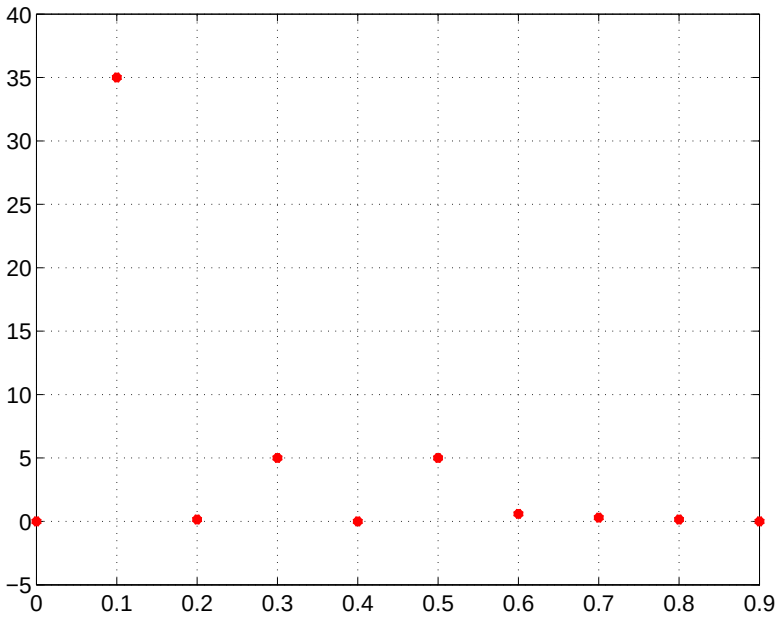
année	1960	1970	1980	1990	2000
population	5429	6270	6366	6874	7288

- Peut-on estimer le nombre d'habitants de la Suisse pendant les années où il n'y a pas eu de recensement, par exemple en 1945 et en 1975?
- Peut-on envisager un modèle pour prédire la population de la Suisse en 2010?

Le polynôme de degré deux (parabole) qui approche ces données au sens des moindres carrés est $p(x) = 0.19x^2 - 710.29x + 657218.92$.



Exemple 3. Les points dans la figure suivante représentent les mesures du débit du sang dans une section de l'artère carotide pendant un battement cardiaque. La fréquence d'acquisition de données est constante et égale à 10/T où T=1 sec. est la période du battement.
On veut déduire de ce signal discret un signal continu représenté par une combinaison linéaire de fonctions connues (par exemple de fonctions trigonométriques, cette représentation est adaptée a notre problème parce que la fonction obtenue va bien approximer un signal périodique).



Soit $f(t)$ le signal dont on connaît $N = 10$ échantillons $[f(t_0), \ldots, f(t_{N-1})]$, avec $t_j = jT/N$. On cherche $\{c_k\} \in \mathbb{C}$, $k \in [0, N-1] \subset \mathbb{N}$ tels que:

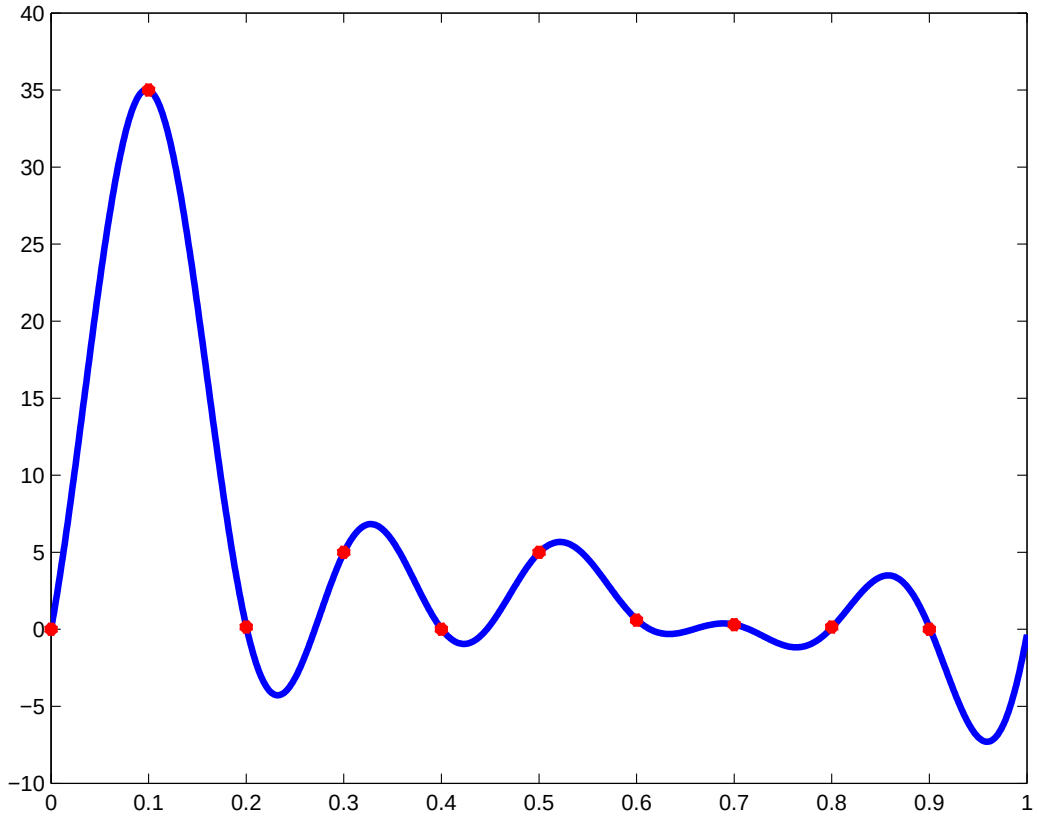
$$f(t_j) = \frac{1}{N} \sum_{k=0}^{N-1} c_k \omega_N^{-kj}, \quad j = 0, \ldots, N-1, \tag{1}$$

où

$$\omega_N = e^{(-2\pi i)/N} = \cos(2\pi/N) - i \sin(2\pi/N),$$

étant i l'unité imaginaire. On peut calculer le vecteur des coefficients $\mathbf{c} = [c_1, \ldots, c_{10}]^T$ par l'algorithme F.F.T. (Fast Fourier Transform, Cooley et Tuckey, 1965), implémenté dans Matlab/Octave avec la commande `fft`.

A partir de l'expression (1), il y a plusieurs techniques pour reconstruire la valeur de f dans chaque point $t \in [0, T]$. Une fois que l'on a reconstruit la fonction, on peut la représenter sur un graphe comme ceci



Position du problème

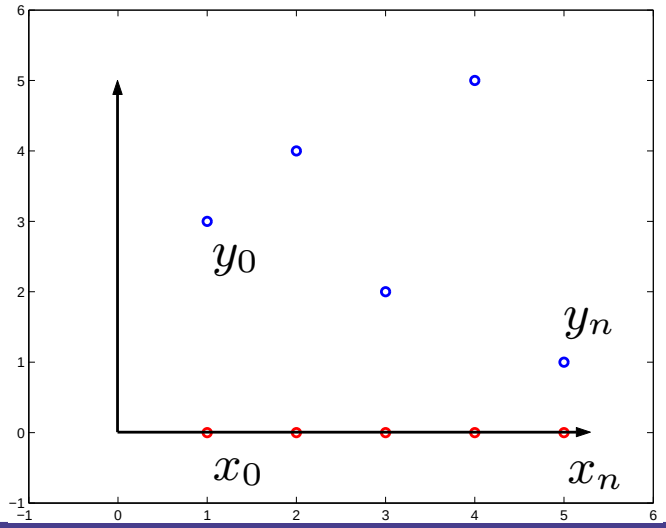
(Sec. 3.1 du livre)

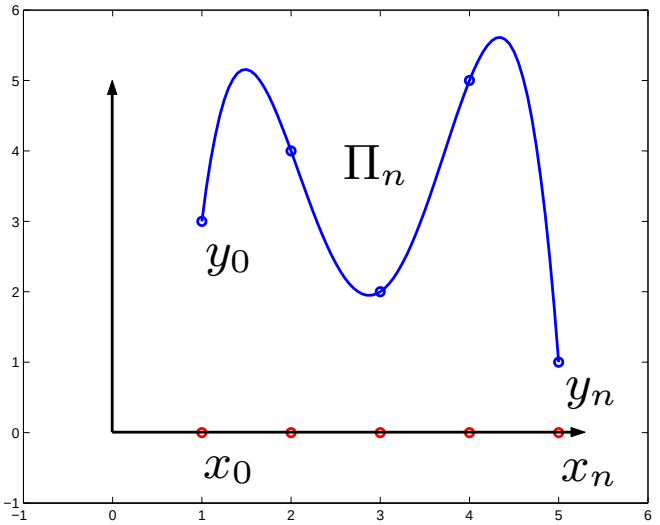
Soit $n \geq 0$ un nombre entier. Etant donnés $n+1$ points distincts $x_0, x_1, \ldots x_n$ et $n+1$ valeurs $y_0, y_1, \ldots y_n$, on cherche un polynôme p de degré n, tel que

$$p(x_j) = y_j \qquad \text{pour } 0 \leq j \leq n. \tag{2}$$

Dans le cas affirmatif, on note $p = \Pi_n$ et on appelle Π_n le polynôme d'interpolation aux points x_j, $j = 0, \ldots, n$.

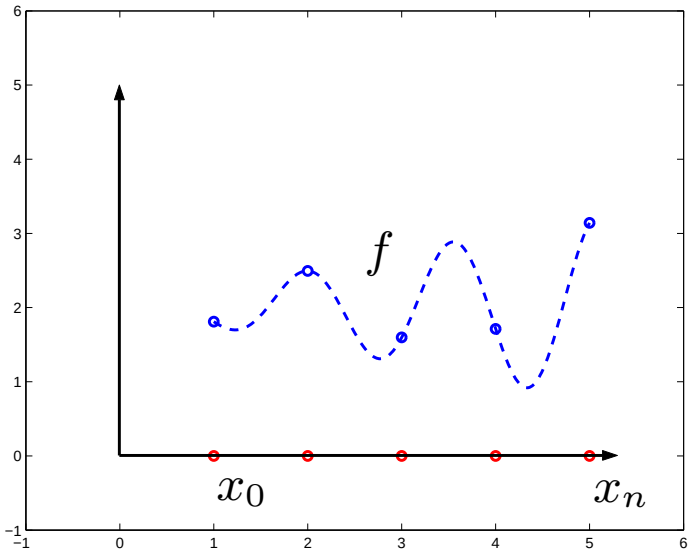
$(n = 4)$

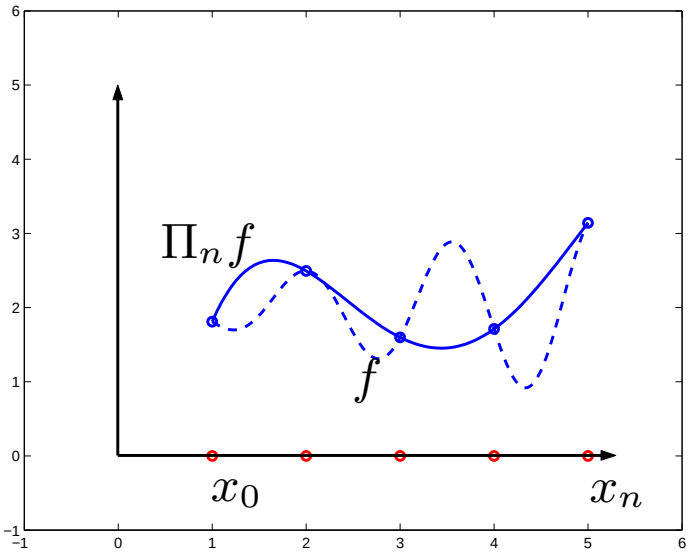




Soit $f \in C^0(I)$ et $x_0, \ldots, x_n \in I$. Si comme valeurs y_j on prend $y_j = f(x_j)$, $0 \leq j \leq n$, alors le polynôme d'interpolation $\Pi_n(x)$est noté $\Pi_n f(x)$ et est appelé l'interpolant de f aux points x_0,... x_n.

$(n = 4)$





Base de Lagrange

(Sec. 3.1.1 du livre)

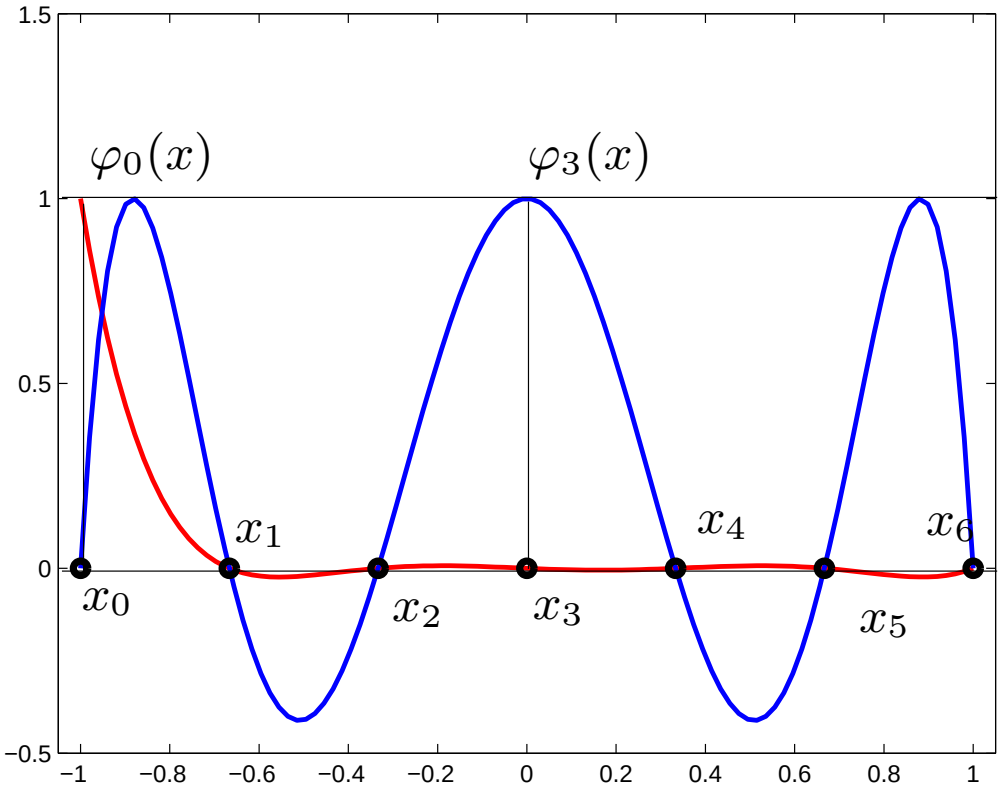
On considère les polynômes φ_k, $k = 0, \ldots, n$ de degré n tels que

$$\varphi_k(x_j) = \delta_{jk}, \qquad k, j = 0, \ldots, n,$$

où $\delta_{jk} = 1$ si $j = k$ et $\delta_{jk} = 0$ si $j \neq k$. Explicitement, on a

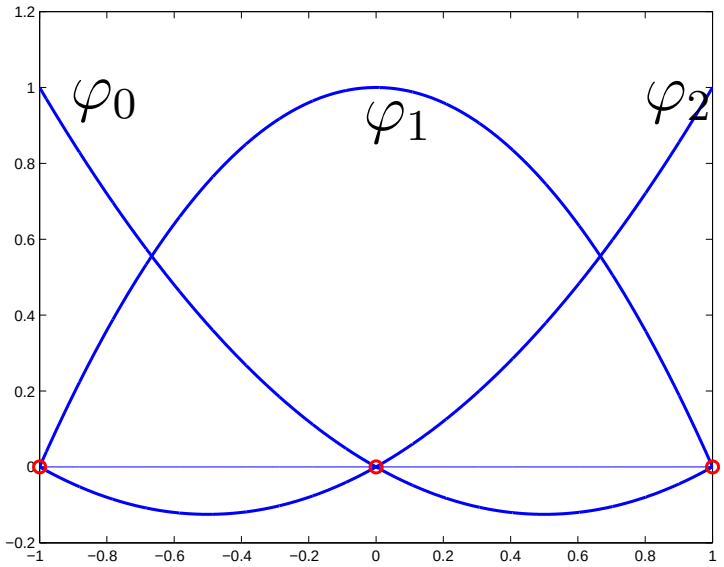
$$\varphi_k(x) = \prod_{j=0, j\neq k}^{n} \frac{(x - x_j)}{(x_k - x_j)}.$$

La figure qui suit montre deux polynômes de Lagrange de degré $n = 6$ relatifs aux points d'interpolations $x_0 = -1$, $x_1 = -2/3, \ldots, x_5 = 2/3$, et $x_6 = 1$.



Exemple 4. Pour $n = 2$, $x_0 = -1$, $x_1 = 0$, $x_2 = 1$ les polynômes de la base de Lagrange sont

$$\begin{aligned}
\varphi_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{1}{2}x(x-1), \\
\varphi_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = -(x+1)(x-1), \\
\varphi_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{1}{2}x(x+1).
\end{aligned}$$



Le polynôme d'interpolation Π_n des valeurs y_j aux points x_j, $j = 0, \ldots, n$, s'écrit

$$\Pi_n(x) = \sum_{k=0}^{n} y_k \varphi_k(x), \tag{3}$$

car il vérifie $\Pi_n(x_j) = \sum_{k=0}^{n} y_k \varphi_k(x_j) = y_j$.

Par conséquent on aura

$$\Pi_n f(x) = \sum_{k=0}^{n} f(x_k) \varphi_k(x).$$

On montre maintenant que le polynôme Π_n en (3) est le seul polynôme de degré n interpolant les données y_i aux nœuds x_i.

Soit, en effet, $Q_n(x)$ un autre polynôme d'interpolation. Alors on a

$$Q_n(x_j) - \Pi_n(x_j) = 0, \qquad j = 0, \ldots, n.$$

Donc, $Q_n(x) - \Pi_n(x)$ est un polynôme de degré n qui s'annule en $n+1$ points distincts; il en suit que $Q_n = \Pi_n$, d'où l'unicité du polynôme interpolant.

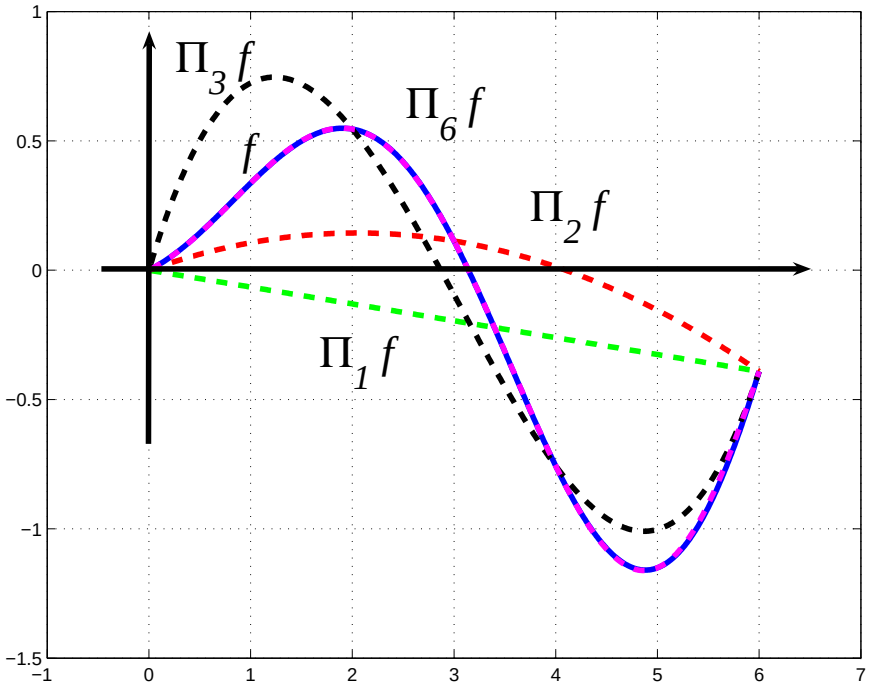
Interpolation d'une fonction regulière

Théorème (Erreur d'interpolation) (voir prop. 3.2 du livre)
Soient $x_0, x_1, \ldots, x_n$, $n+1$ nœuds *équirépartis* dans $I = [a, b]$ et soit $f \in C^{n+1}(I)$. Alors, on a

$$E_n(f) = \max_{x \in I} |f(x) - \Pi_n f(x)| \leq \frac{1}{4(n+1)} \left(\frac{b-a}{n} \right)^{n+1} \max_{x \in I} |f^{(n+1)}(x)|. \tag{4}$$

On remarque que l'erreur d'interpolation dépend de la dérivée $n+1$-ième de f.

Exemple 5. Polynômes d'interpolation $\Pi_i f$ pour $i = 1, 2, 3, 6$ et $f(x) = \dfrac{x+1}{5}\sin(x)$, noeuds équirépartis sur $[0, 6]$.



Interp. numérique avec Matlab/Octave

En Matlab/Octave on peut calculer les polynômes d'interpolation en utilisant les commandes `polyfit` et `polyval`. Voyons plus en détaille comment peut-on utiliser ces commandes.

`p = polyfit(x,y,n)` calcule les coefficients du polynôme de degré `n` qui interpole les valeurs `y` aux points `x`.

Exemple A. On veut interpoler les valeurs $\mathbf{y} = [3.38, 3.86, 3.85, 3.59, 3.49]$ aux points $\mathbf{x} = [0, 0.25, 0.5, 0.75, 1]$ par un polynôme de degré 4. Il suffit d'utiliser les commandes Matlab/Octave suivantes:

```
>> x=[0:0.25:1];   % vecteur des points d'interpolation
>> y=[3.38 3.86 3.85 3.59 3.49]; % vecteur des valeurs
>> p1=polyfit(x,y,4)
p1 =
    1.8133   -0.1600   -4.5933    3.0500    3.3800
```

`p1` est le vecteur des coefficients du polynôme interpolant:
$\Pi_4(x) = 1.8133x^4 - 0.16x^3 - 4.5933x^2 + 3.05x + 3.38.$

Pour calculer le polynôme de degré n qui interpole une fonction f donnée dans $n+1$ points, il faut d'abord construire le vecteur $\mathbf{y}$ en évaluant f dans les nœuds $\mathbf{x}$.

Exemple B. Considérons le cas suivant:

```
>> f='cos(x)';
>> x=[0:0.25:1];
>> y=eval(f);
>> p=polyfit(x,y,4)
p =
    0.0362    0.0063   -0.5025    0.0003    1.0000
```

Remarque 1. *Si la dimension* $m+1$ *de* **x** *et* **y** *est* $m+1 > n+1$ *(*n *étant le degré du polynôme d'interpolation), la commande* `polyfit(x,y,n)` *retourne le polynôme interpolant de degré* n *au sens des moindres carrés (voir sec. 3.4 du livre). Dans le cas où* $m+1 = n+1$*, alors on trouve le polynôme d'interpolation "standard", puisque dans ce cas les deux coïncident.*

`y = polyval(p,x)` calcule les valeurs `y` d'un polynôme de degré `n`, dont les `n+1` coefficients sont memorisés dans le vecteur `p`, aux points `x`, c'est-à-dire: `y` = `p(1)*x.^n` + ... + `p(n)*x + p(n+1)`.

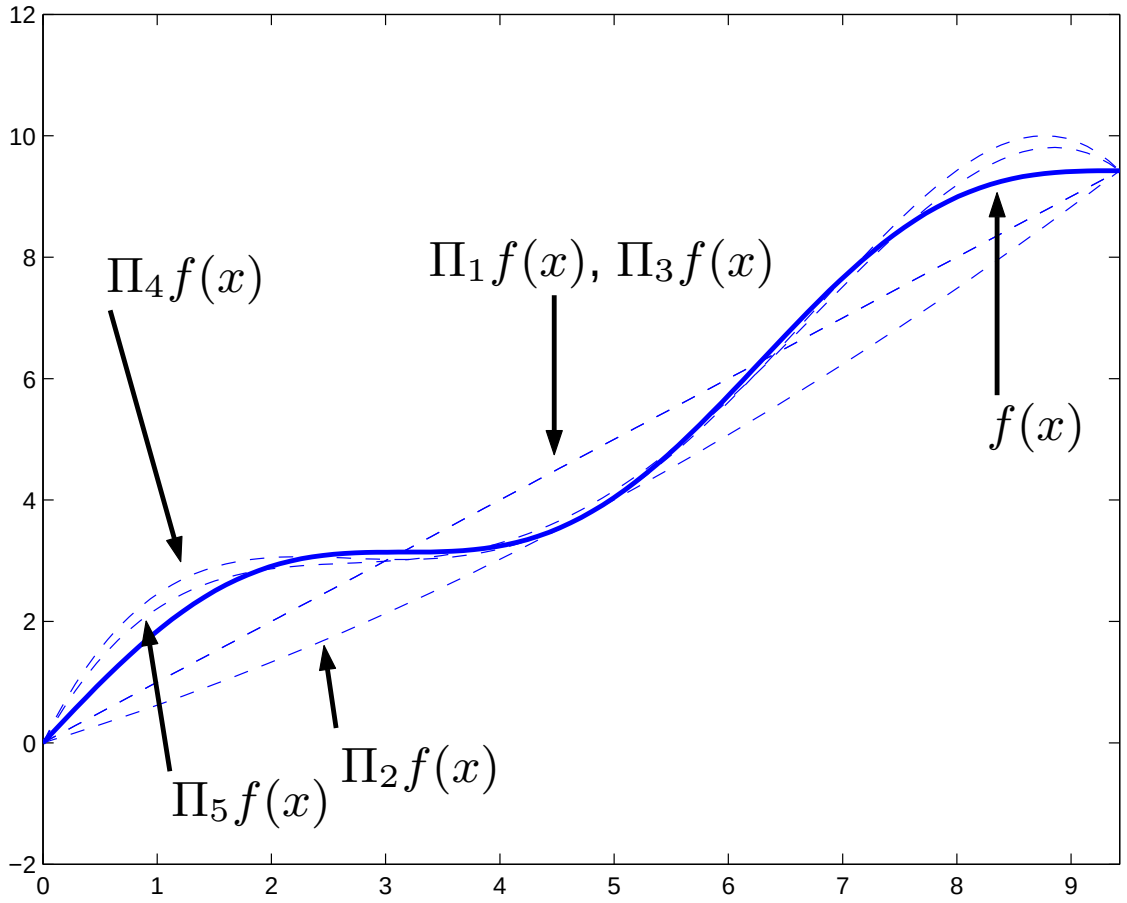
Exemple C. On veut évaluer le polynôme trouvé dans l'Exemple A dans le point $x = 0.4$ et, ensuite, on veut tracer son graphe. On peut utiliser les commandes suivantes:

```
>> x=0.4;
>> y=polyval(p1,x)
y =
    3.9012

>> x=linspace(0,1,100);
>> y=polyval(p1,x);
>> plot(x,y)
```

Exemple 6. On veut interpoler la fonction $\sin(x) + x$ sur $n = 2, 3, 4, 5, 6$ nœuds. En Matlab/Octave on peut utiliser la commande `polyfit` pour calculer les coefficients du polynôme interpolant et `polyval` pour évaluer un polynôme dont on connaît les coefficients dans une suite de points. Voilà les commandes Matlab/Octave :

```
>> f = 'sin(x) + x';
>> x=[0:3*pi/100:3*pi];  x_sample=x;
>> plot(x,eval(f),'b'); hold on
>> for i=2:6
    x=linspace(0,3*pi,i); y=eval(f);
    c=polyfit(x,y,i-1);
    plot(x_sample,polyval(c,x_sample),'b--')
   end
```

La figure suivante montre les 5 polynômes obtenus.



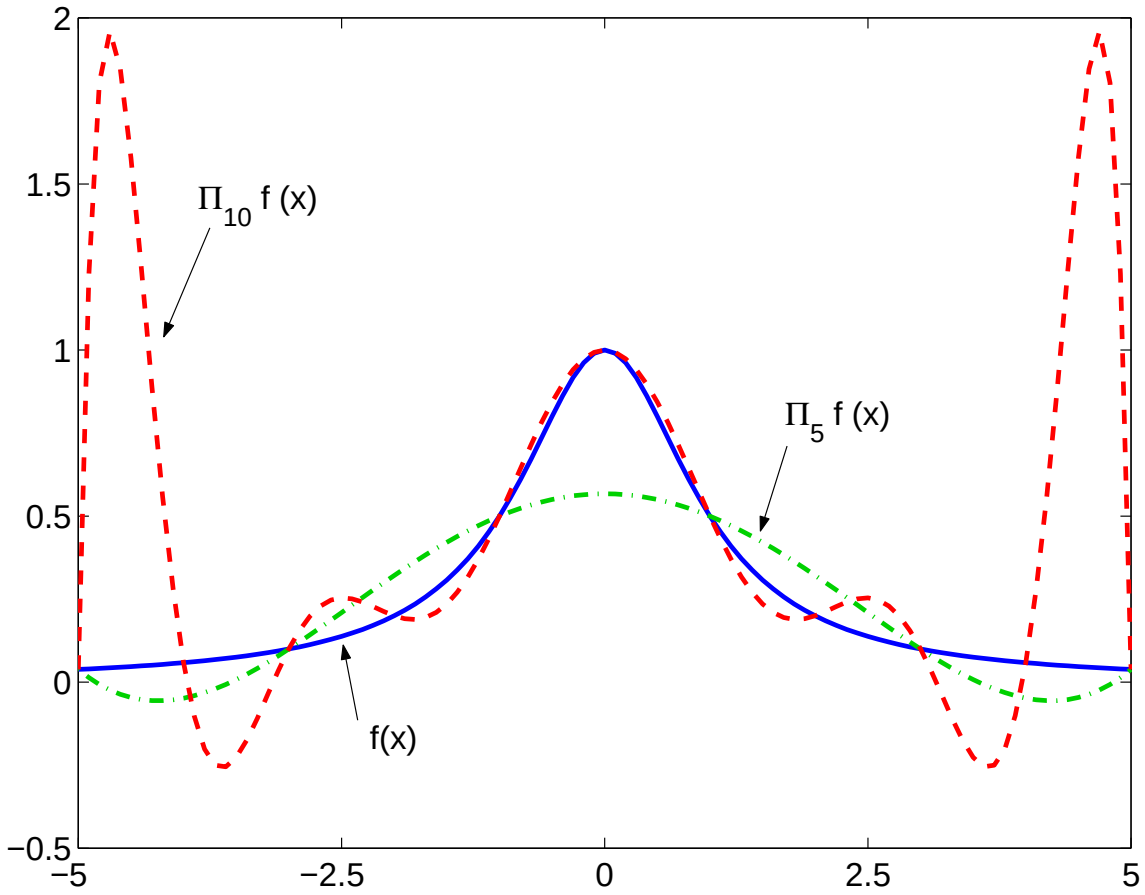
Remarque 2. *Seul le fait que*

$$\lim_{n\to\infty} \frac{1}{4(n+1)} \left(\frac{b-a}{n}\right)^{n+1} = 0$$

n'implique pas que $E_n(f)$ *tend vers zéro quand* $n \to \infty$.

Exemple 7. (Runge) Soit $f(x) = \frac{1}{1+x^2}$, $x \in [-5, 5]$. Si on l'interpole dans des points équirépartis, au voisinage des extrémités de l'intervalle, l'interpolant présente des oscillations, comme on peut le voir sur la figure.

Fonction de Runge et oscillations des polynômes interpolants dans des points équirépartis.



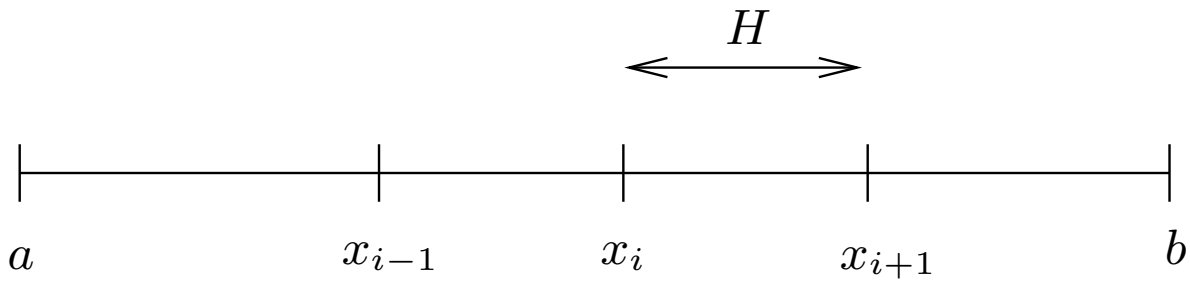
Remède : Interpolation par intervalles.

Interpolation par intervalles

(Sec. 3.2 du livre)

Soient $x_0 = a < x_1 < \cdots < x_N = b$ des points qui divisent l'intervalle $I = [a, b]$ dans une réunion d'intervalles $I_i = [x_i, x_{i+1}]$ de longueur H où

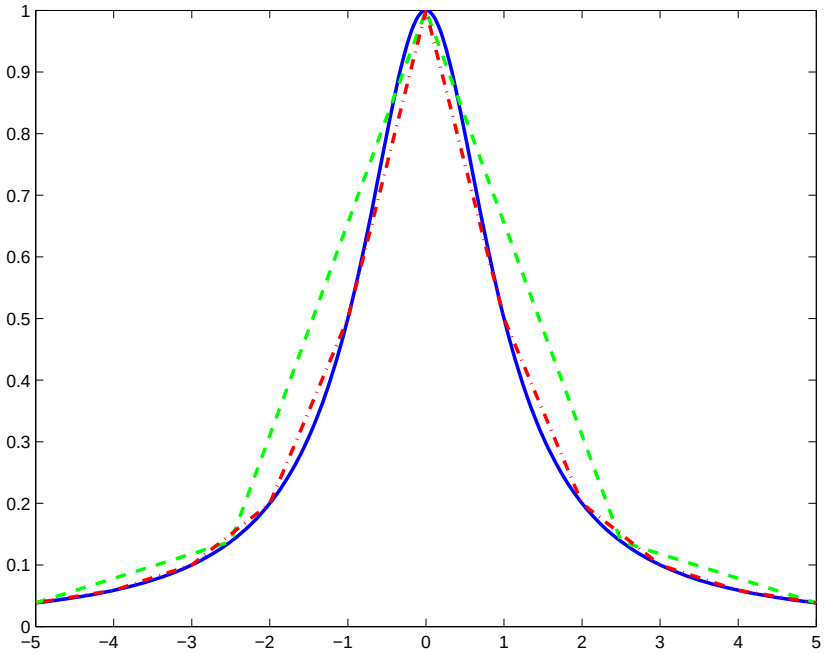
$$H = \frac{b-a}{N}.$$



Sur chaque sous-intervalle I_i on interpole $f_{|I_i}$ par un polynôme de degré 1. Le polynôme par morceaux qu'on obtient est noté $\Pi_1^H f(x)$, et on a:

$$\Pi_1^H f(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \quad \text{pour } x \in I_i.$$

Exemple. 7 (suite) On considère les polynômes par morceaux de degré $n = 1$ interpolants la fonction de Runge, pour 5 et 10 sous-intervalles de $[-5, 5]$.



La figure montre les polynômes $\Pi_1^{H_1} f$ et $\Pi_1^{H_2} f$ pour $H_1 = 2.5$ et $H_2 = 1.0$.

Les commandes Matlab/Octave utilisées sont:

```
>> f = inline('1./(1+x.^2)','x');
>> H1 = 2.5; H2=1.0;
>> x1 = [-5:H1:5]; x2 = [-5:H2:5];
>> y1 = feval(f, x1); y2 = feval(f, x2);
>> x_plot = [-5:.1:5];
>> y1_plot = interp1(x1,y1,x_plot);
>> y2_plot = interp1(x2,y2,x_plot);
>> fplot(f, [-5 5]); hold on;
>> plot(x_plot, y1_plot); plot(x_plot, y2_plot);
```

Théorème 1 (Prop. 3.3 du livre)
Si $f \in C^2(I)$, $(I = [x_0, x_N])$ alors telle que

$$E_1^H(f) = \max_{x \in I} \mid f(x) - \Pi_1^H f(x) \mid \leq \frac{H^2}{8} \max_{x \in I} |f''(x)|.$$

Démonstration. D'après la formule (4), sur chaque intervalle I_i on a

$$\max_{x \in [x_i, x_{i+1}]} \mid f(x) - \Pi_1^H f(x) \mid \leq \frac{H^2}{4(1+1)} \max_{x \in I_i} \mid f''(x) \mid .$$

Remarque 3. *On peut montrer que si l'on utilise un polynôme de degré n (≥ 1) dans chaque sous-intervalle I_i on trouve*

$$E_n^H(f) \leq \frac{H^{n+1}}{4(n+1)} \max_{x \in I} |f^{(n+1)}(x)| .$$

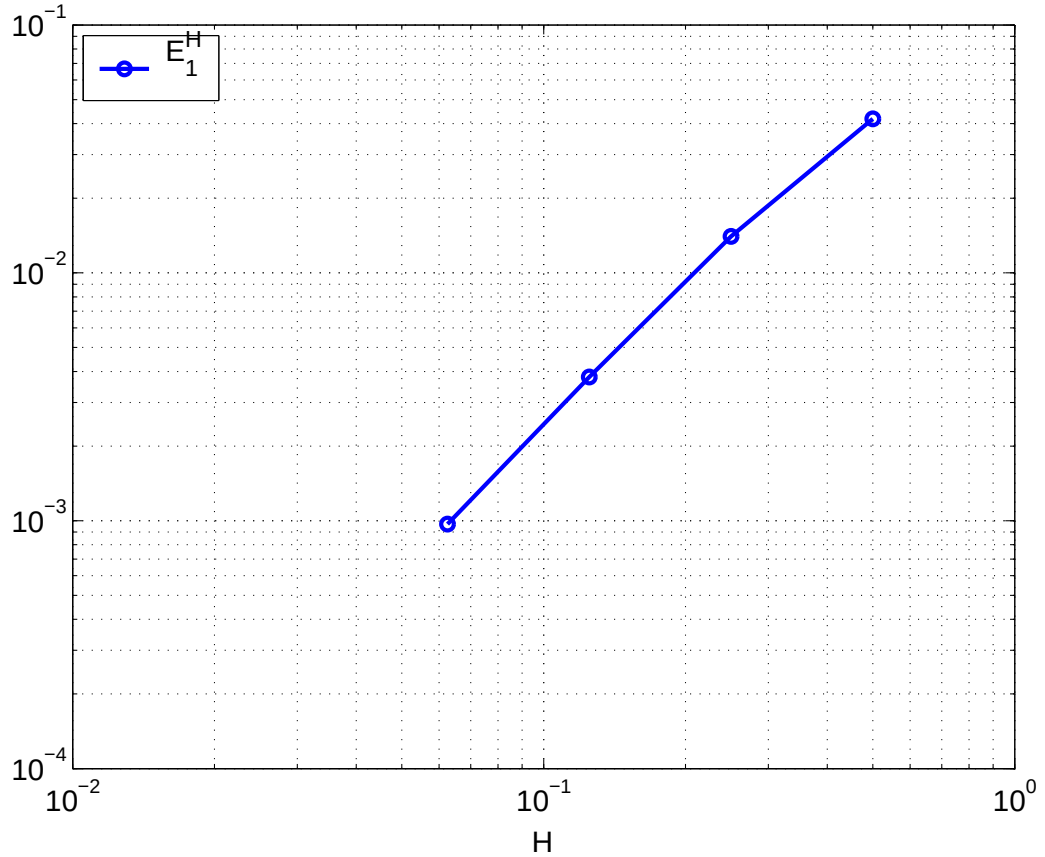
Exemple. 7 (suite) On considère la fonction de Runge $f(x)$ sur $[-5, 5]$ et on prend un nombre K croissant de sous-intervalles $K = 20, 40, 80, 160$ et on estime l'erreur d'interpolation commise en évaluant la différence $|f(x) - \Pi_1^H f(x)|$ sur une grille très fine :

```
>> f=inline('1./(1+x.^2)','x');
>> K=[20 40 80 160]; H=10./K;
>> x_fine = [-5:0.001:5];
>> f_fine = feval(f,x_fine);
>> for i=1:4
    x = [-5:H(i):5]; y = feval(f, x);
    y_fine = interp1(x,y,x_fine);
    err1(i) = max(abs(f_fine - y_fine));
   end
>> loglog(H,err1);
```

La figure qui suit montre (en échelle logarithmique) l'erreur commise en fonction de la taille $H = 10/K$ des sous-intervalles. On voit que l'erreur $E_1^H f$ pour l'interpolation linéaire par morceaux se comporte comme CH^2: ce résultat est en accord avec le théorème 1. En plus, si on calcule les rapports E_1^H / H^2 on peut estimer les constantes C :

```
>> err1./H.^2
ans =
  0.16734   0.22465   0.24330   0.24829
```

Erreur d'interpolation de la fonction de Runge par le polynôme composite Π_1^H en fonction de H.



Approx. au sens des moindres carrés

(Sec. 3.4 du livre)

Supposons que l'on dispose de $n+1$ points $x_0, x_1, \ldots, x_n$ et $n+1$ valeurs $y_0, y_1, \ldots, y_n$. On a vu que, si le nombre de données est grand, le polynôme interpolant peut présenter des oscillations importantes.
Pour avoir une meilleure représentation des données, on peut chercher un polynôme de degré $m < n$ qui approche "au mieux" les données.

Définition 1. *On appelle polynôme aux moindres carrés de degré m* $\tilde{f}_m(x)$ *le polynôme de degré m tel que*

$$\sum_{i=0}^{n} |y_i - \tilde{f}_m(x_i)|^2 \leq \sum_{i=0}^{n} |y_i - p_m(x_i)|^2 \qquad \forall p_m(x) \in \mathbb{P}_m$$

Remarque 4. *Lorsque* $y_i = f(x_i)$ *(*f *étant une fonction continue) alors* $\tilde{f}_m$ *est dit l'approximation de* f *au sens des moindres carrés.*

Autrement dit, le polynôme aux moindres carrés est le polynôme de degré m qui, parmi tous les polynômes de degré m, minimise la distance des données. Si on note $\tilde{f}_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$, et on définit la fonction

$$\Phi(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} \left| y_i - \left(a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_m x_i^m\right)\right|^2$$

alors les coefficients du polynôme aux moindres carrés peuvent être déterminés par les relations

$$\frac{\partial \Phi}{\partial a_k} = 0, \qquad 0 \leq k \leq m \tag{5}$$

ce qui nous donne $m + 1$ relations linéaires entre les a_k.

Utilisons cette méthode dans un cas simple. Considérons les points $x_0 = 1$, $x_1 = 3$, $x_2 = 4$ et les valeurs $y_0 = 0$, $y_1 = 2$, $y_2 = 7$ et calculons le polynôme interpolant de degré 1 au sens des moindres carrés (*ligne de régression*).

Le polynôme recherché a la forme $\tilde{f}_1(x) = a_0 + a_1 x$. On définit:
$\Phi(a_0, a_1) = \sum_{i=0}^{2} [y_i - (a_0 + a_1 x_i)]^2$ et on impose $\frac{\partial \Phi}{\partial a_0} = 0$ et $\frac{\partial \Phi}{\partial a_1} = 0$:

$$\begin{aligned}
\frac{\partial \Phi}{\partial a_0} &= -2 \sum_{i=0}^{2} [y_i - (a_0 + a_1 x_i)] = -2 \left(\sum_{i=0}^{2} y_i - 3a_0 - a_1 \sum_{i=0}^{2} x_i \right) \\
&= -2(9 - 3a_0 - 8a_1) \\
\frac{\partial \Phi}{\partial a_1} &= -2 \sum_{i=0}^{2} x_i [y_i - (a_0 + a_1 x_i)] = -2 \left(\sum_{i=0}^{2} x_i y_i - a_0 \sum_{i=0}^{2} x_i - a_1 \sum_{i=0}^{2} x_i^2 \right) \\
&= -2(34 - 8a_0 - 26a_1)
\end{aligned}$$

Donc les coefficients a_0 et a_1 du polynôme sont les solutions du système linéaire:

$$\begin{cases} 3a_0 + 8a_1 = 9 \\ 8a_0 + 26a_1 = 34 \end{cases}$$

En général, on observe que si pour calculer le polynôme interpolant aux moindres carrés $\tilde{f}_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$ on impose les conditions d'interpolation $\tilde{f}_m(x_i) = y_i$ pour $i = 0, \ldots, n$, alors on trouve le système linéaire $B\mathbf{a} = \tilde{\mathbf{y}}$, où B est la matrice de dimension $(n+1) \times (m+1)$

$$B = \begin{pmatrix} 1 & x_0 & \ldots & x_1^m \\ 1 & x_1 & \ldots & x_2^m \\ \vdots & & & \vdots \\ 1 & x_n & \ldots & x_n^m \end{pmatrix}$$

Puisque $m < n$ le système est surdéterminé, c'est-à-dire que le nombre de lignes est plus grand que le nombre de colonnes. Donc on ne peut pas résoudre ce système de façon classique, mais on doit le résoudre au sens des moindres carrés, en considérant:

$$\boxed{B^T B\mathbf{a} = B^T \tilde{\mathbf{y}}.}$$

De cette façon on trouve le système linéaire $A\mathbf{a} = \mathbf{y}$ (avec $A = B^T B$ et $\mathbf{y} = B^T \tilde{\mathbf{y}}$), dit *système d'équations normales*. On peut montrer que les équations normales sont équivalentes au système (5).

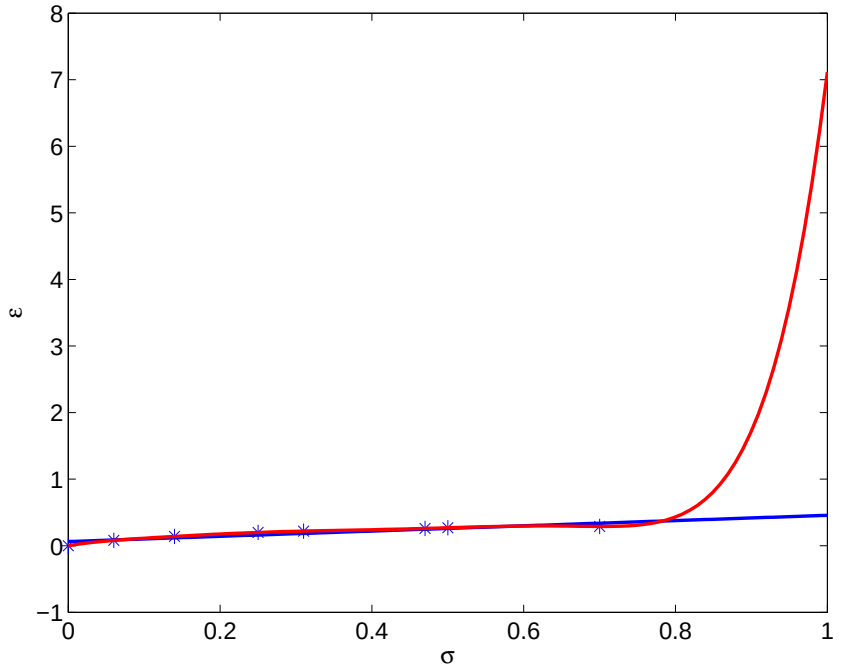
Exemple. 1 (suite) On reprend l'exemple 1 proposé au début du chapitre: on va présenter l'interpolation des données avec Matlab/Octave. D'abord, il faut définir les données expérimentales (8 couples de valeurs σ-ϵ):

```
>> sigma = [0.00 0.06 0.14 0.25 0.31 0.47 0.50 0.70];
>> epsilon = [0.00 0.08 0.14 0.20 0.22 0.26 0.27 0.29];
```

On veut extrapoler la valeur de ϵ pour $\sigma = 0.4$. On considère le polynôme Π_7 et le polynôme aux moindres carrés de degré 1 (la ligne de régression).
On peut utiliser les commandes suivantes:

```
>>  plot(sigma, epsilon, '*'); hold on;  % valeurs connues
>>  sigma_sample = linspace(0,1.0,100);
>>  p7 = polyfit(sigma, epsilon, 7);
>>  pol = polyval(p7, sigma_sample);     % polynome interpolant
>>  plot(sigma_sample,pol, 'r');
>>  p1 = polyfit(sigma,epsilon,1);
>>  pol_mc = polyval(p1, sigma_sample);  % moindres carres
>>  plot(sigma_sample, pol_mc, 'g'); hold off;
```

La figure montre le polynôme interpolant Π_7 (en rouge) et le polynôme d aux moindres carrés de degré 1 (ligne de régression, en bleu).



On remarque que pour $\sigma > 0.7$ MPa les comportements des fonctions qui approchent les données sont tout à fait différents.

En particulier, pour $\sigma = 0.9$ les valeurs de $\epsilon(\sigma)$ extrapolées avec le polynôme interpolant Π_7 et le polynôme d aux moindres carrés de degré 1, sont obtenus par:

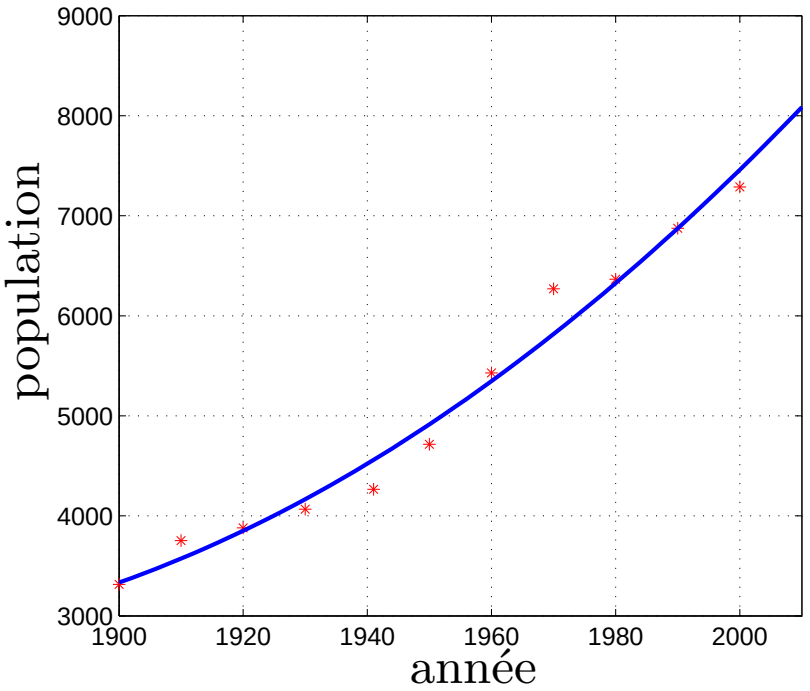
```
>> polyval(p7, 0.9)
ans =
   1.7221

>> polyval(p1, 0.9)
ans =
    0.4173
```

- La déformation obtenue par Π_7 (172.21%) est sûrement trop grande pour être considérée physiologique.
- Par contre, la valeur obtenue grâce à la ligne de régression peut être utilisée pour estimer la déformation pour $\sigma = 0.9$ Mpa.

On revient à l'exemple sur la population suisse.

Exemple. 2 (suite) On veut estimer la population de la Suisse en 2010 en utilisant les valeurs connues en 1900-2000. A ce propos, on utilise un polynôme au moindres carrés $p_2(x)$ de degré 2.

```
>> annee = [1900, 1910, 1920, 1930, 1941, 1950,...
            1960, 1970, 1980, 1990, 2000];
>> population = [3315, 3753, 3880, 4066, 4266, 4715,...
            5429, 6270, 6366, 6874, 7288];
>> p2 = polyfit(annee, population, 2);
>> annee_sample = [1900:2:2010];
>> vp2 = polyval(p2, annee_sample);
>> plot(annee, population, '*r', ...
        annee_sample, vp2, 'b');
```



La valeur éstimées par la méthode pour l'année 2010 est

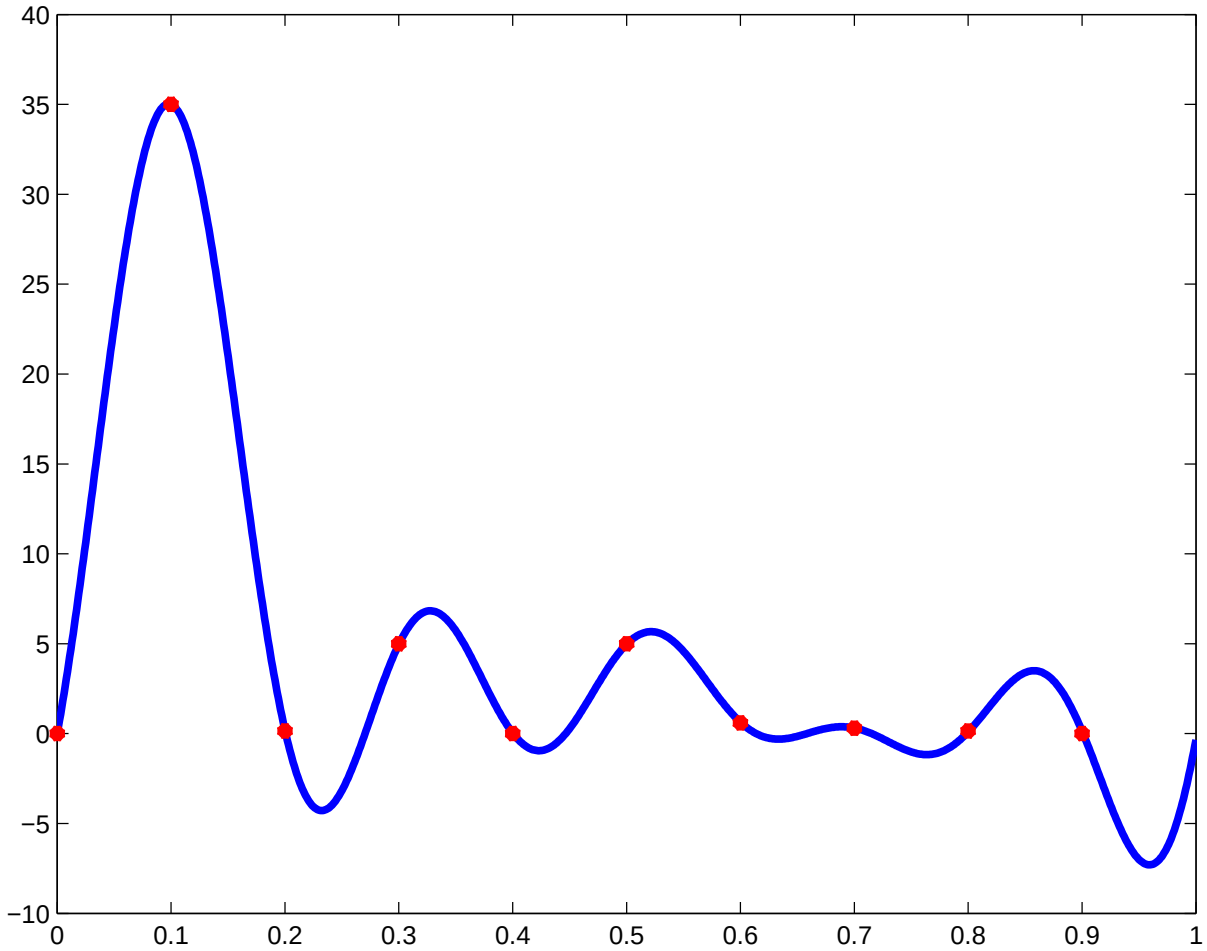
```
>> polyval(p2, 2010)
ans = 8084
```

Finalement, on montre comment il est possible d'interpoler des données périodiques à travers la fonction `interpft`.

Exemple. 3 (suite) On définit les 10 couples de données (temps, débit) par les vecteurs `t` et `deb`; puis, à l'aide de la commande `interpft`, on interpole les données dans 1000 nœuds équirépartis dans l'intervalle de périodicité [0,1]:

```
>> t = [0 .1 .2 .3 .4 .5 .6 .7 .8 .9];
>> deb = [0 35 .15 5 0 5 .6 .3 .15 0];
>> f = interpft(deb, 1000);
>> plot(linspace(0, 1, 1000), f); hold on;
>> plot(t, deb, 'r*')
```

Le résultat est montré par la figure qui suit.


40
35
30
25
20
15
10
5
0
−5
−10
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Traitement des polynômes

Dans Matlab/Octave il y a des commandes spécifiques pour faire des calculs avec polynômes. Soit `x` un vecteur de abscisses, `y` un vecteur d'ordonnées et `p` (respectivement $\texttt{p}_i$) le vecteur des coefficients d'un polynôme $P(x)$ (respectivement P_i); alors on a les commandes suivantes:

commande	action
`y=polyval(p,x)`	`y` $=$ valeurs de $P(x)$
`p=polyfit(x,y,n)`	`p` $=$ coefficients du polynôme d'interpolation Π_n
`z=roots(p)`	`z` $=$ zéros de P tels que $P(\texttt{z}) = 0$
`p=conv(`$\texttt{p}_1$`,`$\texttt{p}_2$`)`	`p` $=$ coefficients du polynôme $P_1 P_2$
`[q,r]=deconv(`$\texttt{p}_1$`,`$\texttt{p}_2$`)`	`q` $=$ coefficients de Q, `r` $=$ coefficients de R tels que $P_1 = QP_2 + R$
`y=polyderiv(p)`	`y` $=$ coefficients de $P'(x)$
`y=polyinteg(p)`	`y` $=$ coefficients de $\int P(x)\, dx$

Interpolation par fonctions splines

(Sec. 3.3 du livre)

Soient $a = x_0 < x_1 < \cdots < x_n = b$ des points qui divisent l'intervalle $I = [a, b]$ dans une réunion d'intervalles $I_i = [x_i, x_{i+1}]$.

Définition 2. *On appelle spline cubique interpolant f une fonction s_3 qui satisfait*

1. $s_{3|I_i} \in \mathbb{P}_3$ *pour tout* $i = 0, \ldots n-1$, $\mathbb{P}_3$ *étant l'ensemble de polynômes de degré* 3,
2. $s_3(x_i) = f(x_i)$ *pour tout* $i = 0, \ldots n$,
3. $s_3 \in C^2([a, b])$.

Cela revient à vérifier les conditions suivantes (on indique par $s_3(x_i^-)$ la limite à gauche de s_3 au point x_i et par $s_3(x_i^+)$ la limite à droite) :

$$
\begin{aligned}
s_3(x_i^-) &= f(x_i) && \text{pour tout } 1 \leq i \leq n-1,\\
s_3(x_i^+) &= f(x_i) && \text{pour tout } 1 \leq i \leq n-1,\\
s_3(x_0) &= f(x_0),\\
s_3(x_n) &= f(x_n),\\
s_3'(x_i^-) &= s_3'(x_i^+) && \text{pour tout } 1 \leq i \leq n-1,\\
s_3''(x_i^-) &= s_3''(x_i^+) && \text{pour tout } 1 \leq i \leq n-1,
\end{aligned}
$$

c'est-à-dire $2(n-1)+2+2(n-1) = 4n-2$ conditions.

Il faut trouver $4n$ inconnues (qui sont les 4 coefficients de chacune des n restrictions $s_{3|I_i}$, $i = 0, \ldots n-1$) et on dispose de $4n-2$ relations.

On rajoute alors 2 conditions supplémentaires à vérifier.

- Si l'on impose

$$s_3''(x_0^+) = 0 \quad \text{et} \quad s_3''(x_n^-) = 0, \tag{6}$$

  alors la spline s_3 est complètement déterminée et s'appelle $spline\ naturelle$.

- Une autre possibilité est d'imposer la continuité des dérivées troisièmes dans le nœuds x_2 et x_{n-1}, c'est à dire:

$$s_3'''(x_2^-) = s_3'''(x_2^+) \quad \text{et} \quad s_3'''(x_{n-1}^-) = s_3'''(x_{n-1}^+). \tag{7}$$

  Les conditions (7) sont dites *not-a-knot*.

Spline cubique naturelle interpolant la fonction
$f(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04} - 6$, (nœuds équirépartis $x_j = -1 + j/4$, $j = 0, \ldots, 16$)