Agrégation 2023 :

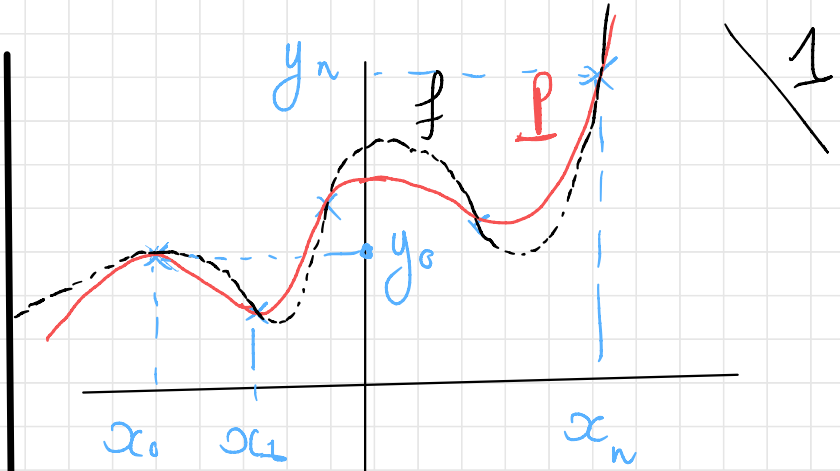
Interpolation de Lagrange

Motivation : à partir de données

* $(n+1)$ abscisses : $x_0 < x_1 < \dots < x_n$
* $(n+1)$ ordonnées $\{y_0, y_1, \dots, y_n\}$

on cherche à construire une fonction polynôme $P : \mathbb{R} \to \mathbb{R}$ telle que

$\forall i \in \{0, \dots, n\}, \quad P(x_i) = y_i$



Questions

→ l'objet existe-t-il ? (↔ P)
" est-il unique ?

→ l'objet représente-t-il correctement la "réalité" ? (↔ f)

→ Peut-on construire "facilement" cet objet ?

1) Existence et unicité du PIL

Théorème : sous les notations précédentes, il existe un unique polynôme P de degré inférieur ou égal à n tel que :

$$\forall i \in \{0, \dots, n\}, \ P(x_i) = y_i$$

On appelle P le polynôme d'interpolation de Lagrange (*) associé aux points $\{(x_0, y_0), \dots (x_n, y_n)\}$

(*) abréviation : PIL.

preuve : 2 démonstrations possibles : 2

→ non constructive : soit l'application

$$\varphi : \begin{pmatrix} \mathbb{R}_n[X] \to \mathbb{R}^{n+1} \\ P \mapsto (P(x_0), P(x_1), \dots P(x_n)) \end{pmatrix}$$

($\mathbb{R}_n[X]$: espace des polynômes de degré $\leq n$)

On montre que φ est :

→ linéaire

→ injective : si $\varphi(P) = 0$, P possède $n+1$ racines ; étant de degré $\leq n$, $P = 0$.

Comme $\dim \mathbb{R}_n[X] = \dim(\mathbb{R}^{n+1}) = n+1$,

φ est donc bijective et le PIL P est égal à :

$$P = \varphi^{-1}((y_0, \dots y_n))$$

→ constructive : on utilise la famille des polynômes de base de Lagrange associée aux abscisses $\{x_0, \dots, x_n\}$:

$$\ell_i(x) = \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)} \quad (\in \mathbb{R}_n[X]) \quad \neq 0$$

(on a $\ell_i(x_j) = \delta_{i,j}$)

On remarque que

$$P(x) = \sum_{i=0}^{n} y_i \ell_i(x) \quad (\in \mathbb{R}_n[X]) \quad (1)$$

convient : en effet

$$P(x_j) = \sum_{i=0}^{n} y_i \delta_{i,j} = y_j$$

et c'est le seul : en effet, la famille $(\ell_i)_{0 \leq i \leq n}$ est une base de $\mathbb{R}_n[X]$ (car libre).

2) Algorithme de construction du PIL

A priori, on dispose déjà d'un algorithme avec la preuve constructive précédente. Cependant, la formule (1) ne permet pas de construire efficacement le PIL, d'une part, en raison d'un coût élevé,

et surtout d'un mauvais conditionnement (*).

(*) comportement de l'algorithme vis à vis des erreurs d'arrondis

* Une autre possibilité pour construire P est de l'exprimer sous forme canonique :

$$P(x) = a_0 + a_1 x + \dots + a_n x^n$$

et de rechercher les coefficients $(a_i)_{0 \le i \le n}$ en résolvant le système :

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$$

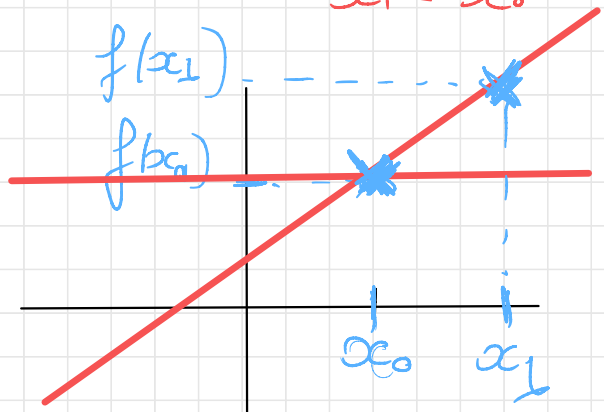
A nouveau, le problème d'un mauvais conditionnement de la matrice de Vandermonde se pose, ainsi que le coût de la résolution. On propose un algorithme, appelé algorithme des différences divisées, permettant de construire rapidement et efficacement le PIL :

Définition : on note $f[x_0, \dots x_n]$ le coefficient de degré n du PIL, associé aux points $((x_0, f(x_0)), \dots (x_n, f(x_n)))$ où $f : \mathbb{R} \to \mathbb{R}$ quelconque.

Par exemple :

$f[x_0] = f(x_0)$

$f[x_0, x_1] = \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$



On montre la proposition suivante :

Proposition : On a

$$f[x_0, \dots, x_j] = \frac{f[x_1, \dots, x_j] - f[x_0, \dots, x_{j-1}]}{x_j - x_0}$$

Le PIL P est alors donné par la relation suivante : 5

Proposition : on a

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + \dots + f[x_0, \dots x_n](x - x_0) \dots (x - x_{n-1})$$

Remarque : il s'agit de l'expression de P dans la base de Newton

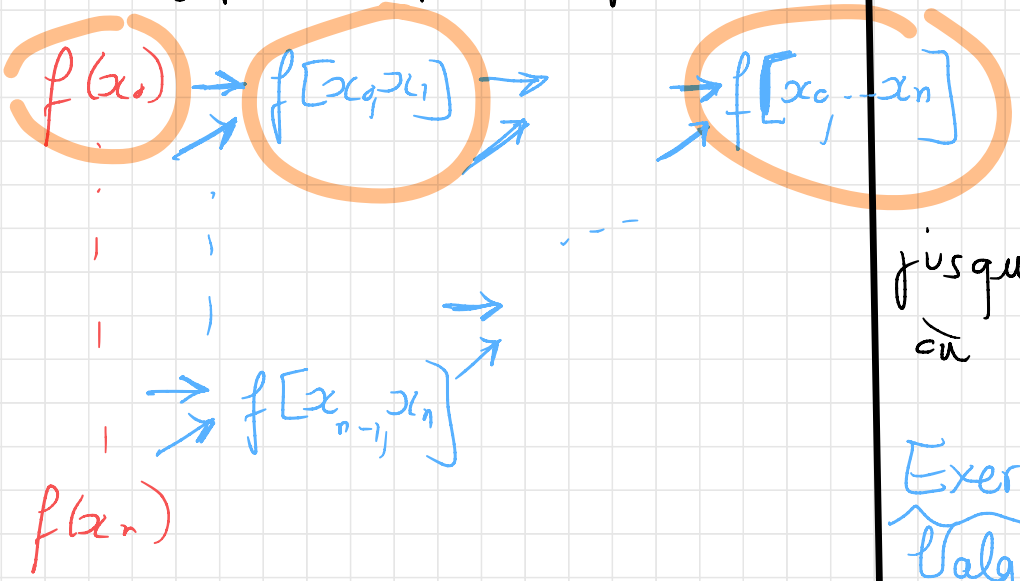
$$\{1, (x - x_0), \dots, (x - x_0) \dots (x - x_{n-1})\}$$

preuves : à voir dans [Dem] ou poly de cours

On dispose à present d'un algorithme de construction du PIL, dit des différences divisées :

→ Etape 1 : calcul des coefficients

$f[x_0], f[x_0,x_1], \dots, f[x_0,\dots,x_n]$



→ Etape 2 : calcul de L

on utilise la méthode d'estimation de Hörner :

$$\begin{cases} u_n = f[x_0,\dots x_n] \\ u_{n-1} = f[x_0,\dots x_{n-1}] + u_n(x-x_{n-1}) \\ \vdots \\ u_k = f[x_0,\dots,x_k] + u_{k+1}(x-x_k) \end{cases}$$

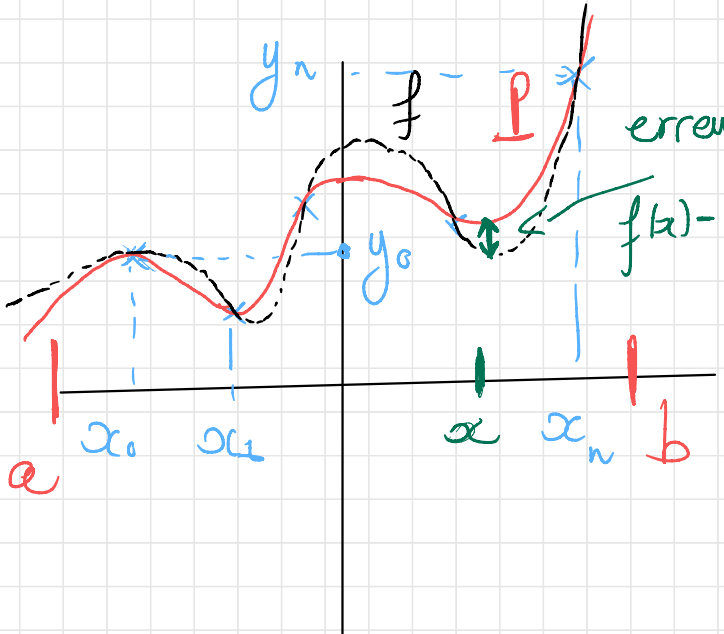
jusqu'à $u_0 = f[x_0] + u_1(x-x_0)$

où $u_0 = P(x)$

Exercice Python : programmer l'algorithme des différences divisées

→ à faire et à envoyer à laurent.dumas@uvsq.fr avant la prochaine séance (Ve 23, 9h)

3) Estimation d'erreur



La question restant en suspens est la convergence du PIL, $\underline{P}$, vers f, lorsque on augmente le nombre de points (si f est continue par exemple).

On dispose du résultat suivant :

Théorème on suppose $f \in C^{n+1}(\mathbb{R},\mathbb{R})$ et que $\underline{P}$ désigne le PIL de f aux points $\{x_0, \dots x_n\}$.

Il existe $\xi \in [x_0, x_n]$, tel que

$$f(x) - P(x) = \frac{\Pi_n(x)}{(n+1)!} f^{(n+1)}(\xi)$$

où $\Pi_n(x) = \prod_{i=0}^{n} (x - x_i)$

$(x \in [x_0, x_n])$

preuve : voir [Dem] au poly de cours

(basée sur l'utilisation d'une fonction auxiliaire et l'application de Rolle).

Remarque :

* lorsque $n \to +\infty$,

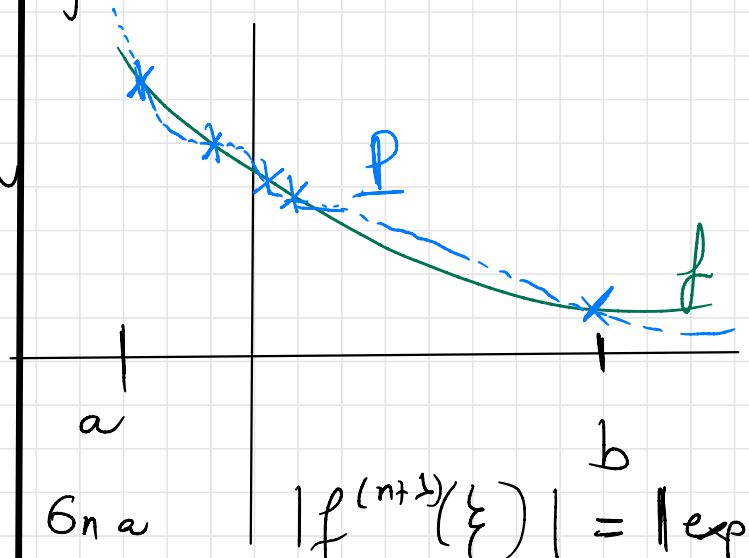
$$\left| \frac{\Pi_n(x)}{(n+1)!} \right| \leq \frac{(b-a)^{n+1}}{(n+1)!} \to 0 \quad (+\infty)$$

mais on peut avoir

$$f^{(n+1)}(\xi) \to +\infty$$

→ un exemple favorable :

$f(x) = e^{-x}$



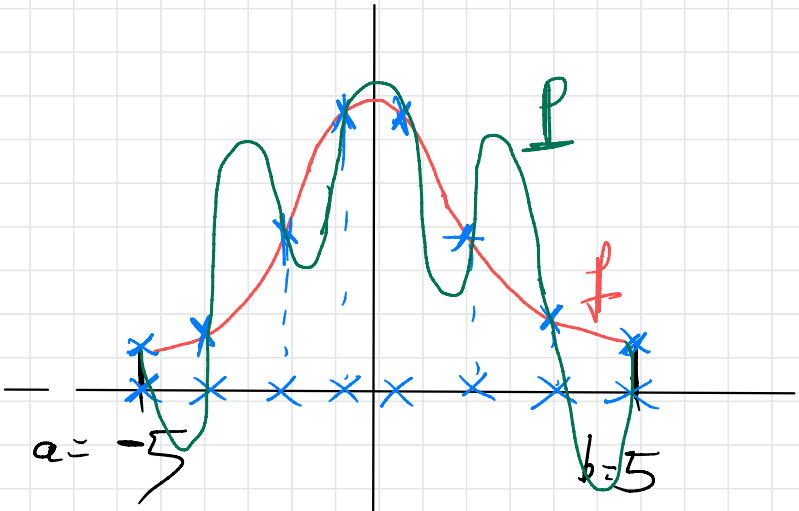
On a $|f^{(n+1)}(\xi)| = |\exp(-\xi)| \leq e^{-a}$

On a $P_n \xrightarrow[n \to +\infty]{\text{unif.}} f$ sur $[a,b]$

quelle que soit la répartition des points

→ un exemple défavorable :

$f(x) = \frac{1}{1+x^2}$ sur $[-5,5]$



Plus n augmente, plus des oscillations fortes apparaissent aux extrémités (phénomène de Runge)

Exercice : test numérique avec Python et le code précédent

Conclusion : la méthode d'interpolation de Lagrange ne permet pas d'obtenir de manière générale une bonne approximation d'une fonction f. Elle est cependant très importante, d'un point de vue théorique, pour d'autres chapitres du programme (intégration numérique).

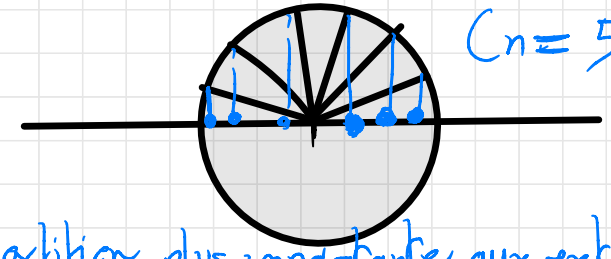
Remarque : il existe une autre estimation d'erreur, globale, basée sur la

sur la constante de Lebesgue (voir poly ou [Dem]).

On peut aussi montrer qu'il existe une répartition "optimale" de points, en l'occurrence les points de Tchebychev :

$$x_i = \cos\left(\frac{2k+1}{2n+2}\pi\right) \quad (0 \le k \le n)$$

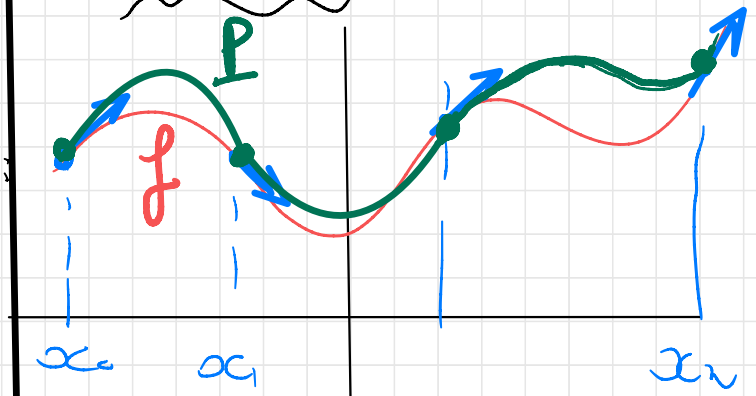
sur l'intervalle $[-1, 1]$.



(répartition plus importante aux extrémités)

4) Autres méthodes

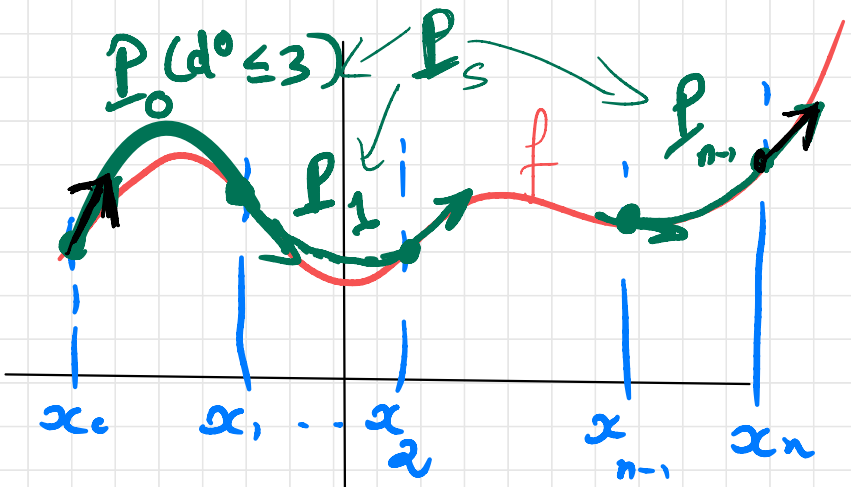
→ extension : interpolation d'Hermite



P interpole f et ses dérivées éventuelles ($P(x_i) = f(x_i)$ et $P'(x_i) = f'(x_i)$)

→ Pb de non convergence maintenu

→ interpolation par splines cubiques :



On construit une interpolation cubique par morceaux telle que

$P_s / [x_i, x_{i+1}]$ est de degré ≤ 3

$P_s(x_i) = f(x_i)$, $0 \leq i \leq n$

$P_s'(x_0) = f'(x_0)$ et $P_s'(x_n) = f'(x_n)$

P_s est C^2 sur $[a,b]$ (recollement C^2)

Il existe un résultat d'existence et d'unicité de P_s, un algorithme de construction (basé sur la résolution d'un système linéaire tridiagonal) et un résultat de convergence (de P_s vers f lorsque $n \to +\infty$)

→ Exemples d'utilisation :

- → Latex
- → CAO
- → Jeux vidéos (graphisme)

Aussi : B-splines, courbes de Bézier, krigeage (voir texte), ...

A faire :

→ revoir leçon (mise en ligne)

→ code Python à envoyer avant jeudi 22

→ exercices (feuilles TD1)