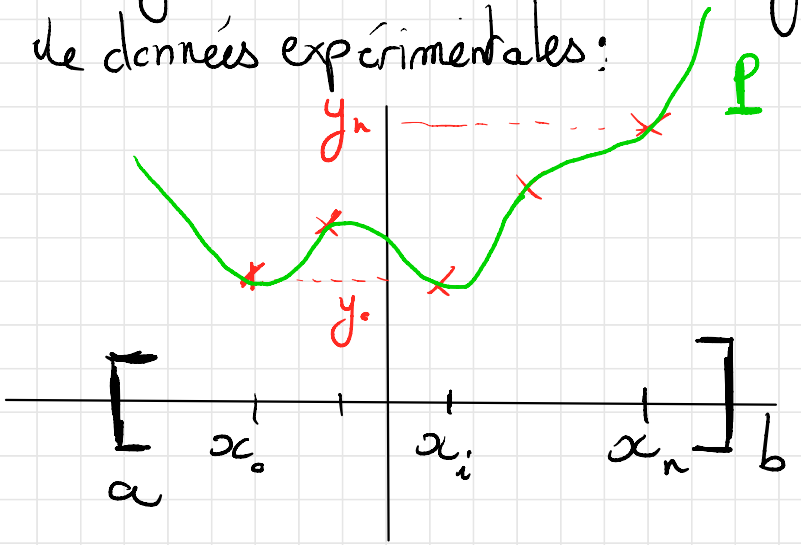
Séance 2 : Interpolation de Lagrange : aspects théoriques et numériques

* Modélisation : on cherche à reconstruire une fonction réelle sur un segment à partir d'un nombre fini de données expérimentales :



* Données expérimentales :
  → famille $x_0 < x_1 < \dots < x_n \in [a,b]$
  → valeurs mesurées $(y_i)_{0 \le i \le n} \in \mathbb{R}^{n+1}$

* Objectif : reconstruction d'une fonction continue $f : \begin{pmatrix} [a,b] \to \mathbb{R} \\ x \mapsto f(x) \end{pmatrix}$ telle que $\forall i \in [\![0,n]\!], f(x_i) = y_i$

Parce que f existe et soit unique, il faut rajouter des contraintes sur f (à définir). On doit ensuite définir un algorithme de construction de ce polynôme, rapide et robuste (vis à

vis des erreurs d'arrondi).

On vérifie enfin que l'objet construit et simulé numériquement, répond de manière satisfaisante au problème initial.

1) Etape 1 : existence et unicité du polynôme d'interpolation de Lagrange

Théorème-def : Les notations sont inchangées. Il existe un unique polynôme $P \in \mathbb{R}_n[X]$ tel que $\forall i \in \{0, \dots, n\}$, $P(x_i) = y_i$

on l'appelle polynôme d'interpolation associé à la famille $\{(x_0, y_0), \dots, (x_n, y_n)\}$ (PIL en abrégé)

Preuve : soit

$$\varphi : \left(\begin{array}{ccc} \mathbb{R}_n[X] & \longrightarrow & \mathbb{R}^{n+1} \\ P & \longmapsto & (P(x_0), \dots, P(x_n)) \end{array} \right)$$

On a :

* φ linéaire (immédiat)
* φ injective : En effet, si $P(x_0) = \dots = P(x_n) = 0$, alors P possède $(n+1)$ racines distinctes. P étant de degré $\leq n$, il est nul.

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

alors φ est bijective.

Ainsi, par toute famille $(y_i)_{0\le i\le n} \in \mathbb{R}^{n+1}$,

il existe un unique polynôme $P \in \mathbb{R}_n[X]$

tq $\varphi(P) = (y_0, \dots, y_n)$, soit

exactement $P(x_i) = y_i$ si $0 \le i \le n$.

Remarque : il existe aussi une démonstration constructive de P :

on définit la famille $(l_i)_{0\le i\le n}$ des polynômes de base de Lagrange par les points $(x_i)_{0\le i\le n}$:

$$l_i(x) = \frac{\prod_{j\neq i} (x - x_j)}{\prod_{j\neq i} (x_i - x_j)}$$

On remarque que

$l_i \in \mathbb{R}_n[X]$ (de degré n)

et $\begin{cases} l_i(x_j) = 0 \text{ si } j \neq i \\ l_i(x_i) = 1 \end{cases}$

On montre ensuite que $(l_i)_{0\le i\le n}$ est une base de $\mathbb{R}_n[X]$.

En effet, si $\sum_{i=0}^{n} \lambda_i l_i = 0$,

en évaluant en x_j, on trouve $\lambda_j = 0$

On a donc (analyse/synthèse) // nécessairement et de manière suffisante :

$$P(a) = \sum_{i=0}^{n} y_i \ell_i(a)$$

Etape 2 : algorithme de construction du PIL

* L'algorithme basé sur la famille $(\ell_i)_{0 \le i \le n}$ présente plusieurs défauts : coût et surtout sa sensibilité aux erreurs d'arrondis (quotients petits)

* Si on exprime P dans la base canonique : $P(X) = a_0 + a_1 X + \dots + a_n X^n$,

on doit avoir :

$$\begin{cases} a_0 + a_1 x_0 + \dots + a_n x_0^n = y_0 \\ \vdots \\ a_0 + a_1 x_n + \dots + a_n x_n^n = y_n \end{cases}$$

soit matriciellement :

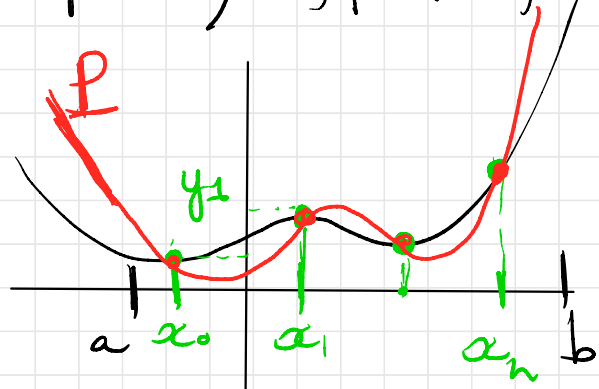
$$\begin{pmatrix} 1 & x_0 & \dots & x_0^n \\ \vdots & x_1 & & x_1^n \\ & \vdots & & \vdots \\ 1 & x_n & & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}$$

On se ramène donc à la résolution d'un système linéaire avec la matrice de Vandermonde. Cet algorithme, au fait satisfaisant, ne convient toujours pas d'un point de vue numérique (la matrice de Vandermonde est mal conditionnée (*) : voir chapitre syst. linéaires) (**) très sensible aux erreurs d'arrondis

On utilise un algorithme, dit des différences divisées, qui remplit tous les critères :

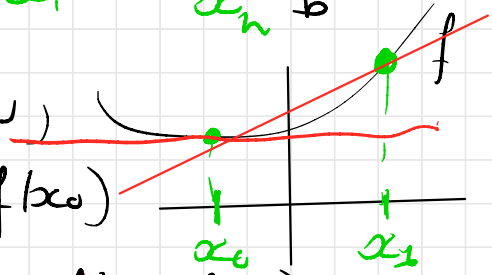
Déf on note, par $f : \mathbb{R} \to \mathbb{R}$ 5

$f[x_0, \dots x_n]$ le coefficient de degré n du PIL associé aux points $\{(x_i, f(x_i)),\ 0 \le i \le n\}$



Par exemple,

$f[x_0] = f(x_0)$



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

On construit alors par récurrence toutes les différences divisées :

Lemme : on a

$$f[x_0, \dots x_j] = \frac{f[x_1, \dots x_j] - f[x_0, \dots x_{j-1}]}{x_j - x_0} \quad (1 \leq j \leq n) \qquad (\bullet)$$

(preuve : voir poly / livre LD, Demailly)

On a le résultat suivant qui exprime le PIL en fonction des coefficients précédents dans une base adaptée :

Proposition : soit P_j le PIL de f associé aux points $(x_0, \dots x_n)$. On a

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

(preuve : voir poly / Demailly)

Par estimer $P(x)$, on dispose à présent d'un algorithme très efficace, appelé algorithme des différences divisées :

Etape 1 : calcul des coefficients $f[x_0], \dots f[x_0, \dots x_n]$:

$f(x_0)$ — (•) — $f[x_0, x_1]$ — (•) — $f[x_0, \dots x_n]$

$f(x_1)$ — $f[x_1, x_2]$

(•) $f[x_{n-1}, x_n]$

$f(x_n)$

Etape 2 : estimation de $P(\alpha)$ par la méthode de Horner à partir de (••) : on construit la suite à indice décroissant :

$$\begin{cases} u_n = f[x_0, \dots x_n] \\ \downarrow \\ u_{n-1} = (\alpha - x_{n-1}) u_n + f[x_0, \dots x_{n-1}] \\ \vdots \\ u_k = (\alpha - x_k) u_{k+1} + f[x_0, \dots x_k] \\ (0 \leq k \leq n-1) \end{cases}$$

et $u_0 = P(\alpha)$

3) Etape 3 : implémentation Python de l'algorithme Dif. Div

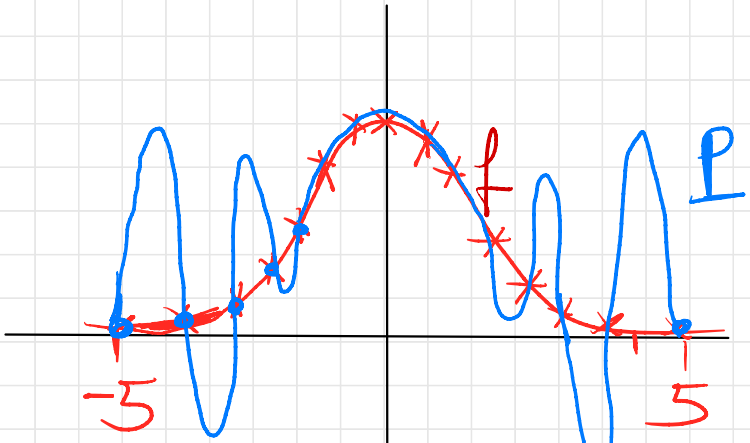
* Par apprendre à utiliser Python, nombreuses ressources en

ligne (doc, tuto, ...) dont :
la chaîne Youtube
"les maths par l'exemple"
(Maëlle Nodet)
etc...

Cette année, les exemples seront réalisés non plus avec Jupyter ("feuilles de calcul") mais sous forme de scripts réalisés (Spyder, ...)

* Script python disponible sur page web :
https://dumas.perso.math.cnrs.fr/Agreg-LM6P.html

* Méthode des différences divisées (basée sur les formules (.) et (..) et la méthode de Horner).
* On observe quand on augmente le nombre de points, une instabilité numérique.
* Une autre instabilité peut apparaître sur certains exemples précis : $f(x) = \frac{1}{1+5x^2}$ sur $[-5,5]$ et $n \geq 10$



On parle du phénomène d'instabilité (mathématique) de Runge.

En conclusion de cette étude numérique, il s'avère que l'interpolation polynomiale de Lagrange n'est pas toujours bien adaptée en modélisation.

En pratique, on utilise plutôt une interpolation par splines cubiques (voir texte "krigeage").

Néanmoins, la méthode d'interpolation de Lagrange est très utile mathématiquement et par d'autres applications (calcul d'intégrales)

4) Compléments mathématiques sur l'interpolation de Lagrange

On peut montrer le résultat

mathématique suivant d'estimation d'erreur entre f et P :

Théorème : soit P le PIL de f aux points $x_0 < \dots < x_n$. Alors il existe $\xi \in [x_0, x_n]$ tel que $\forall x \in [x_0, x_n]$

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

(en supposant f C^{n+1} sur $[x_0, x_n]$)

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

preuve : poly / Demailly

Remarque : on n'a pas toujours $|f(x) - P(x)| \to 0$ quand $n \to +\infty$

(même si $\left| \frac{\Pi_n(x)}{(n+1)!} \right| \to 0$ toujours)

* Suivant le choix de la fonction f, et de la répartition des points d'interpolation, on pourra observer une convergence ou une non-convergence de $|f(x) - P(x)|$ vers 0 lorsque $n \to +\infty$.