PT **DEVOIR NON SURVEILLÉ Nº 1** **2024/2025**

à rendre le 20 septembre 2024

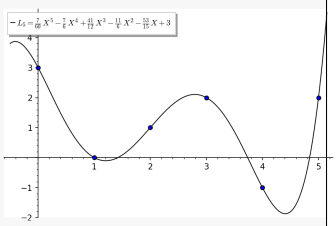
Polynôme d'interpolation de Lagrange

On considère $n \in \mathbb{N}^*$, et n réels $(a_1, \dots, a_n)$ *distincts*, c'est-à-dire que $i \neq j \Longrightarrow a_i \neq a_j$.
On considère également n réels quelconques $(u_1, \dots, u_n)$.
L'objectif de ce devoir est de montrer l'existence et l'unicité d'un polynôme $L_n \in \mathbb{R}_{n-1}[X]$ tel que

$$\forall k \in [\![1, n]\!], \quad L_n(a_k) = u_k.$$

Ce polynôme L_n est appelé « *polynôme d'interpolation de* LAGRANGE » associé à $(a_k)_{1 \leqslant k \leqslant n}$ et $(u_k)_{1 \leqslant k \leqslant n}$.

Du point de vue graphique, trouver L_n revient à trouver une courbe (polynomiale) de degré n dont le graphe passe par tous les points de coordonnées $(a_k, u_k)_{1 \leqslant k \leqslant n}$ (appelés *points de contrôle*).
Si $n = 1$, ceci n'a pas d'intérêt ; si $n = 2$, on retrouve l'unique droite passant par deux points d'abscisses différentes ; si $n = 3$, on détermine l'unique parabole d'axe vertical passant par trois points d'abscisses distinctes, *etc.*



Exemple avec $n = 5$.

1 Existence et unicité : algèbre linéaire

On considère l'application $\Phi : \begin{array}{ccc} \mathbb{R}_{n-1}[X] & \longrightarrow & \mathbb{R}^n \\ P & \longmapsto & \big(P(a_1), \dots, P(a_n)\big) \end{array}$

1 Montrer que Φ est linéaire.

2 Montrer que si $P \in \mathrm{Ker}(\Phi)$, alors $P = 0$. En déduire que Φ est un isomorphisme.

3 En conclure l'existence et l'unicité du polynôme $L_n \in \mathbb{R}_{n-1}[X]$ tel que $\forall k \in [\![1, n]\!], \quad L_n(a_k) = u_k$.

2 Détermination pratique

Les réels $(a_k)_{1 \leqslant k \leqslant n}$ et $(u_k)_{1 \leqslant k \leqslant n}$ étant connus et fixés, on souhaite pouvoir calculer $L_n(x)$ pour $x \in \mathbb{R}$. Il serait même préférable de trouver un mode de calcul cumulatif, c'est-à-dire qu'ajouter un point de contrôle n'oblige pas à reprendre le calcul depuis le début. Pour cela, remarquons que :

4 Soit $L_n \in \mathbb{R}_{n-1}[X]$ le polynôme d'interpolation associé à $(a_1, \dots, a_n)$ et $(u_1, \dots, u_n)$, et $\big(a_{n+1}, u_{n+1}\big)$ un « autre » point (avec $a_{n+1} \notin \{a_1, \dots, a_n\}$) ; λ_{n+1} étant un réel, considérons

$$L_{n+1} = \lambda_{n+1}(X - a_1) \cdots (X - a_n) + L_n.$$

Montrer que L_{n+1} est le polynôme d'interpolation associé à $(a_1, \dots, a_n, a_{n+1})$ et $(u_1, \dots, u_n, u_{n+1})$ si et seulement si :

$$\lambda_{n+1} = \frac{u_{n+1} - L_n(a_{n+1})}{(a_{n+1} - a_1) \cdots (a_{n+1} - a_n)}. \qquad (\star)$$

Ceci permet de construire le polynôme d'interpolation étape par étape :
$L_1 = u_1 = \lambda_1$ (**initialisation**) ;
L_k étant déterminé, on peut calculer $L_k(a_{k+1})$ puis utiliser $(\star)$ pour déterminer λ_{k+1}, puis
$L_{k+1} = L_k + (X - a_1) \cdots (X - a_k)\lambda_{k+1}$ (**hérédité**).
Les premières étapes donnent

$L_1 =$ λ_1 ;
$L_2 =$ $\lambda_1 + (X - a_1)\lambda_2$;
$L_3 = L_2 + (X - a_1)(X - a_2)\lambda_3 =$ $\lambda_1 + (X - a_1)\big(\lambda_2 + (X - a_2)\lambda_3\big)$;
$L_4 = L_3 + (X - a_1)(X - a_2)(X - a_3)\lambda_4 =$ $\lambda_1 + (X - a_1)\big(\lambda_2 + (X - a_2)\big(\lambda_3 + (X - a_3)\lambda_4\big)\big)$;
etc.

On constate alors que, une fois déterminés $(\lambda_1, \dots, \lambda_n)$, on peut calculer $L_n(x)$ pour $x \in \mathbb{R}$ à l'aide de la boucle : $\begin{cases} P_1 = \lambda_n \\ P_{i+1} = \lambda_{n-i} + \big(x - a_{n-i}\big) P_i \quad (1 \leqslant i \leqslant n-1) \end{cases}$; alors $P_n = L_n(x)$ (essayer avec $n = 4$).

Mise en œuvre

5 Calculer à la main $L_3(X)$ si $n = 3$, $(a_1, a_2, a_3) = (0, 1, 2)$ et $(u_1, u_2, u_3) = (0, 1, 3)$ (On précisera les valeurs de $(\lambda_1, \lambda_2, \lambda_3)$).

6 Que renvoient les commandes Python suivantes ?

```
>>>L0=[0,1,1/2]
>>>[L0[k] for k in range(len(L0)-2,-1,-1)]
```

7 On définit les fonctions Python suivantes :

```
>>>def Pval(la,ll,x):
>>>    v=ll[-1]
>>>    for k in range(len(ll)-2,-1,-1):
>>>        v=(x-la[k])*v+ll[k]
>>>    return(v)
```

```
>>>def listeLambda(la,lu):
>>>    L=[lu[0]]
>>>    for k in range(1,len(la)):
>>>        ls=lu[k]-Pval(la[:k],L,la[k])
>>>        for j in range(k):
>>>            ls=ls/(la[k]-la[j])
>>>        L.append(ls)
>>>    return(L)
```

```
>>>def interpol(la,lu,x):
>>>    L0=listeLambda(la,lu)
>>>    return(Pval(la,L0,x))
```

À quoi servent-elles ? Retrouver à l'aide de ces fonctions les valeurs de λ trouvées en question **5**.

3 Un exemple graphique

On considère la fonction $f : x \mapsto \dfrac{1}{1+25x^2}$, sur l'intervalle $[-1, 1]$.

On chargera les librairies `numpy` et `pyplot` à l'aide des commandes `import numpy as np` et `import matplotlib.pyplot as plt`.

8 Étudier l'effet des commandes :

```
>>>def F(x):
>>>    return(1/(1+25*x**2))
>>>Fv=np.vectorize(F)
>>>X1=[-1+k/5 for k in range(11)]
>>>Y1=Fv(X1)
>>>print(Y1)
>>>F(X1)
```

Quel est l'effet de `Fv=np.vectorize(F)` ?

9 Écrire les commandes Python qui permettent de superposer sur le même graphe le tracé de f sur $[-1, 1]$, et celui du polynôme d'interpolation P_3, obtenu avec une suite $(a_k)_{1\leqslant k\leqslant 11}$ arithmétique, telle que $a_1 = -1$ et $a_{11} = 1$, et une suite $(u_k)_{1\leqslant k\leqslant 11}$ définie par $u_k = f(a_k)$.

10 Écrire les commandes Python qui permettent de superposer sur le même graphe les deux tracés précédents, et celui du polynôme d'interpolation Q_3, défini par $a_k = \cos\left(\dfrac{(2k-1)\pi}{22}\right)$ pour $1 \leqslant k \leqslant 11$ et $u_k = f(a_k)$.
Lequel des polynômes P_3 et Q_3 est le plus proche de f (en comparant les graphes) ?