TP 13 - Polynômes 2

Un polynôme est caractérisé par la liste de ses coefficients. Aucun module n'est strictement au programme, voici les deux plus courants :

➢ Le module `numpy.polynomial` propose un type `Polynomial` défini à partir de la liste $[a_0, a_1, \dots, a_n]$

```
>>> import numpy.polynomial as pol
>>> P=pol.Polynomial([1,2,3,4])
>>> P(0)
1.0
```

Les fonctions spécifiques à ce module vous seront données, si besoin, dans l'énoncé ou les annexes du sujet.

➢ Le module numpy propose un type `poly1d` défini à partir de la liste $[a_n, a_{n-1}, \dots, a_1, a_0]$

```
>>> import numpy as np
>>> P=np.poly1d([1,2,3,4])
>>> print(P)
   3     2
1 x + 2 x + 3 x + 4
>>> P(0)
4
```

Nous allons travailler avec ce type dans ce TP. Les fonctions suivantes, du **module numpy**, existent aussi sous la forme d'attributs :

Notation	Explication
`poly1d(L)`	Polynôme dont les coefficients sont donnés dans la liste `L` dans l'ordre des puissances décroissantes
`P[i]`	Coefficient de X^i
`P(a)`	Evaluation de `P` en a
`P.coeffs`	Liste des coefficients
`len(P)`	Dégré, voir `P.order`
`polyder(P)`	Polynôme dérivé, voir `P.deriv()`
`polyint(P)`	Primitive de `P` nulle en 0 voir `P.integ()`
`roots(P)`	Racines complexes de `P`
`poly1d(L,True)`	Polynôme dont les racines sont donnés dans la liste `L`
`q,r=A/B`	Quotient et reste de la division euclidienne de `A` par `B` voir aussi `q,r=polydiv(A,B)`

Arithmétique sur les polynômes

Exercice 1 Division euclidienne

1. Compléter l'algorithme de la division euclidienne des polynômes A par B :

Entrée : A, B deux polynômes

$Q \leftarrow \dots$
$R \leftarrow \dots$
Tant que … faire
  $T \leftarrow \dots$
  $Q \leftarrow \dots$
  $R \leftarrow \dots$
FinTantque

Sortie : Q, R tel que $A = QB + R$ avec $\deg(R) < \deg(B)$

2. Donner une fonction `DE(A,B)` qui effectue la division euclidienne du polynôme `A` par le polynôme `B` (sans utiliser l'instruction / ou la fonction `polydiv`).

Exemple : $X^4 + X^2 + X - 1 = (X^2 - 1)(X^2 + 2) + X + 1$

Exercice 2 *Euclide étendu* **Prog**

1. Formaliser l'algorithme d'Euclide étendu sur le couple de polynômes $(A;B)$.
2. Écrire une fonction `bezout(A,B)` qui réalise l'algorithme d'Euclide étendu.

Algorithme de Sturm

Objectif : Rechercher les racines d'un polynôme sur un intervalle

Considérons un polynôme $P \in \mathbb{R}[X]$ a racines simples.
On définit la suite de Sturm, comme la suite des polynômes (P_k) telle que :

$$P_0 = P \quad P_1 = P'$$

$\forall k \geq 2$, $P_k = -R$ où R est le reste de la DE de P_{k-2} par P_{k-1})
P_n est le dernier polynôme non nul de la suite.

On introduit $S(\alpha)$, la fonction donnant le nombre de changements de signes (stricts) dans la suite $(P_k(\alpha))_{k \in [\![0,n]\!]}$.

Le théorème de Sturm dit que si $\alpha < \beta$ tel que $P(\alpha) \neq 0$ et $P(\beta) \neq 0$ alors $S(\alpha) - S(\beta)$ est le nombre de racines réelles de P dans l'intervalle $[\alpha, \beta]$

Exercice 3

1. Écrire une fonction `S(L)` qui dénombre les changements de signes (stricts) des termes de la liste `L` sachant que le premier élément est non nul.
2. Proposer un moyen de construire un polynôme à racines simples ayant les mêmes racines que P.

Compléter la fonction `simple(P)` qui retourne un tel polynôme.

```
def simple(P):
    A,B=...
    while len(B)>0:
        Q,R=np.polydiv(A,B)
        A,B=...
    ...
    return(...)
```

3. Écrire une fonction `sturm(P)` qui retourne le suite de Strum du polynôme `P` (jusqu'au dernier polynôme non nul).
4. Écrire une fonction `nbzeros(L,a,b)` qui renvoie le nombres de racines réelles du polynôme `P` sur `[a,b]` où `P` est supposé à racines simples ne s'annulant ni en a, ni en b et `L` est sa suite de Sturm associée.
5. Écrire une fonction `racines(P,a,b,h,L=[])` qui retourne la liste des racines réelles de P, à une précision h, sur l'intervalle $[a,b]$ en utilisant le principe de dichotomie.

Interpolation de Lagrange

Considérant $\big((x_i,y_i)\big)_{i\in[\![0,n-1]\!]} \in (\mathbb{R}^2)^n$.
On définit pour $i \in [\![0,n-1]\!]$

$$L_i = \prod_{\substack{j=0\\ j\neq i}}^{n-1} \frac{X-x_j}{x_i-x_j}$$

L'unique polynôme $P \in \mathbb{R}_{n-1}[X]$ tel que pour tout $i \in [\![0,n-1]\!]$, $P(x_i) = y_i$ est

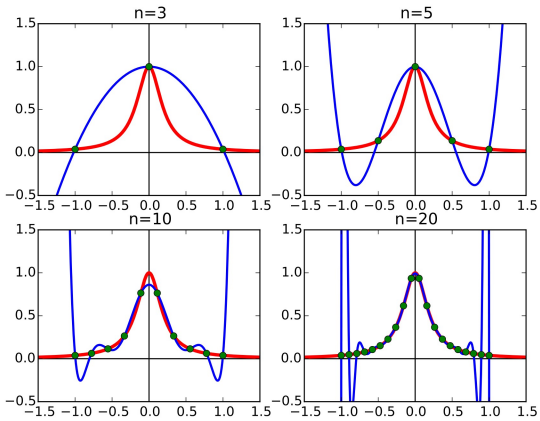
$$P = \sum_{i=0}^{n} y_i L_i$$

Exercice 4

1. Écrire une fonction `lagrange(X,Y,t)` qui prend deux listes `X` et `Y` et retourne l'expression de $P(t)$ où P est le polynôme d'interpolation de Lagrange des couples associés aux deux familles de nombres X et Y.
Remarque : la l'instruction `enumerate` est recommandée.
2. Écrire une fonction qui affiche la courbe de la fonction f, les points d'interpolations et la courbe du polynôme d'interpolation. Les points x_i sont choisis uniformément répartis entre a et b :

$$\forall i \in [\![0,n-1]\!], \quad x_i = a + i\frac{b-a}{n-1}$$

Les courbes sont tracés sur la fenêtre $[x_m,x_M]\times[y_m,y_M]$.
L'intitulé de la fonction est :
`interpolation(f,a,b,n,xm,xM,ym,yM)`
3. Tester la fonction précédente avec $g : x \mapsto \dfrac{1}{1+25x^2}$, sur $[-1,1]$ et $f : x \mapsto \dfrac{2t+3}{3}$



➜ Dès que le nombre de points d'interpolation est grand (>60), la fonction obtenue ne correspond plus au polynôme de Lagrange. On observe des oscillations de plus en plus grandes aux extrémités de l'intervalle et elle on tendance à se propager progressivement à l'intérieur : ceci est le connu sous le nom du **phénomène de Runge**.
Remarque : Nous avons travaillé avec un répartition uniforme de points d'interpolation. Intéressons nous maintenant à une autre répartition !

Exercice 5 *Subdivision de Chebychev* On rappelle le polynôme de Chebychev de première espèce d'ordre n :

$$\forall x \in \mathbb{R}, \quad T_n(\cos(x)) = \cos(nx)$$

1. Identifier les racines de T_n qui sont dans $[-1,1]$.
Répartir les racines sur $[a,b]$ à l'aide d'une transformation affine.
2. Écrire une fonction effectuant l'interpolation d'un fonction f sur $[a,b]$ en les points définis ci-avant :
`interpolation_T(f,a,b,n,xm,xM,ym,yM)`
3. Comparer les résultats en fonction des deux répartitions.