TD Informatique TSI2

Lagrange polynomials. Part I and Part II

PRELIMINARY

Comment faire des opérations sur les polynômes avec Python

On peut rentrer un polynôme comme une fonction.
On peut aussi utiliser la fonction `Polynomial` du sous-module `numpy.polynomial` et un certain nombre d'attributs associés. Comme ce module peut être utilisé lors de l'oral de Math de CCS ou de CCINP, autant commencer à le connaître. On commence donc par taper :

```
In [1]: from numpy.polynomial import Polynomial
In [2]: import numpy as np
```

- ☞ Pour ***créer un polynôme***, on liste ses coefficients par ordre de degré croissant.
  Ainsi pour $p = 1 - 2X + 5X^2 + 2X^3$, on tape : `p = Polynomial([1.,-2.,5.,2.])`
  L'attribut `coef` donne accès aux coefficients ordonnés par degré croissant.
  Ainsi, si l'on tape : `p.coef`, on obtient `array([1.,-2.,5.,2.])`
  On suppose dans la suite que p ***a été défini avec*** `Polynomial`.
- ☞ Pour ***calculer la valeur*** $p(a)$, on tape simplement : $p(a)$ et pour obtenir $[p(x1), ..., p(xn)]$, soit on tape tout simplement $[p(x1), ..., p(xn)]$), soit on tape $P(np.array([x1, ..., xn]))$
- ☞ `p.coef[i]` fournit le ***coefficient devant*** X^i.
- ☞ `p.degree( )` donne le ***degré*** et `p.roots( )` les ***racines*** de p.
- ☞ Pour ***dériver le polynôme*** p, on tape : `p.deriv( )` qui renvoie un nouveau polynôme. Cet attribut a un argument optionnel n qui indique le nombre de dérivations à effectuer. Si l'on tape par exemple `p.deriv(2).coef`, l'on obtient les coefficients du polynôme dérivée seconde de p.
- ☞ Pour ***intégrer le polynôme*** p, on tape : `p.integ( )` qui renvoie un nouveau polynôme. Cet attribut a un premier argument optionnel donnant le nombre d'intégrations à effectuer et un second argument optionnel qui donne la constante d'intégration (ou la liste de constantes si l'on intègre plusieurs fois) à utiliser.
- ☞ Pour faire des ***opérations*** sur des polynômes, on utilise $+$, $-$ et $*$ pour additionner, soustraire et multiplier des polynômes. Par exemple, l'opération `(p**3).coef` permet d'obtenir la liste des coefficients de p^3.
- ☞ Pour faire la ***division euclidienne*** du polynôme p par le polynôme q, la commande `p // q` donne le polynôme quotient et `p` % `q` donne le reste.

Exemple

Soit le polynôme $P = X^3+2X-3$, *écrire les commandes pour calculer successivement* $P(0)$, $[P(1), P(2), P(3)]$, deg P, *les racines de* P, *puis les polynômes* P', P'' *(à vérifier à la main) puis le quotient et le reste de la division euclidienne de* P *par* $B = X^2 + 1$ *(à vérifier à la main) puis le produit de* P *par* B *(à vérifier à la main) et enfin les coefficients de* P^n *avec* $n \in [\![0, 8]\!]$.

Construction des polynômes de Lagrange

1. Créer une fonction `BASE_LAGRANGE` d'argument la liste `Xabs` , qui renvoie la liste des polynômes de la base d'interpolation de Lagrange associée aux valeurs de `Xabs`, en utilisant l'algorithme écrit dans le cours.

Indications : *On utilisera la fonction* `Polynomial` *de* `numpy.polynomial` *pour construire la liste des polynômes interpolateurs* $L_0, ..., L_n$ *de Lagrange. Au départ, on initialise avec* `L=[]` *et on prend* `n=len(Xabs)-1`.
Puis on construit les polynômes $L_0, L_1, ..., Ln$ *dans une boucle double en calquant l'algorithme du cours.*
Plus précisément, on initialise avec `Pol=Polynomial([1])` *qui est* $Pol = 1$ *juste après le premier* `for` *et après le second* `for`, *on met une instruction conditionnelle avec le code* `if j!=i` *et on fait le produit de* `Pol` *par un certain polynôme de degré* 1
On rappelle que le code `Polynomial([a,b])` *est le polynôme* $a + bX$.
Après être sorti de la boucle intérieure, on met `Pol` *dans* `L` *avec* `append`
Enfin, après la boucle `for` *extérieure, on retourne* `L` *qui sera une liste de listes polynomiales*

T.S.V.P →

Appliquer au cas où `Xabs= [0,1,2]` et retrouver le résultat de l'exercice 01 du cours.

2. En utilisant `BASE_LAGRANGE` comme sous-procédure et la fonction `sum`, créer une fonction `POL_LAG` d'arguments `Xabs` et `Xord` qui renvoie le polynôme d'interpolation de Lagrange passant par les points dont les abscisses sont les valeurs de `Xabs` et les ordonnées sont les valeurs de `Xord`

Appliquer au cas où `Xabs=[0,1,2]` et `Xord=[1,-3,2]` et retrouver le résultat de l'exercice 05 du cours.

Une utilisation des polynômes de Lagrange

L'idée est de pouvoir faire une approximation du graphe d'une fonction en remplaçant ce graphe par celui d'un polynôme d'interpolation qui passe par un certain nombre de points de ce graphe. Il est clair que plus le nombre de points est grand, plus les deux graphes vont se coller (hors éventuellement des effets de bord que l'on laissera de côté dans ce TP, pour ceux que cela intéresse, on utilise pour éviter des effets de bords comme `Xabs` les racines de polynômes de Tchebychev).

1. Modifier la procédure `POL_LAG` en remplaçant l'argument `Xord` par `f`, où `f` est une fonction donnée et la procèdure (qui s'appelle maintenant `NEW_POL_LAG`) renvoie le polynôme d'interpolation de Lagrange qui passe par les points $(x_0, f(x_0)), ..., (x_n, f(x_n))$. Ainsi `Xord[k]` est remplaçé par `f(Xabs[k])`

Appliquer `NEW_POL_LAG` avec $Xabs = [0, 1, 2]$ et la fonction $x \mapsto x^2$. Que remarque t-on ?

```
# RAPPEL. Code pour rentrer la fonction f
In [17]: def f(x): return x**2
```

2. Appliquer `NEW_POL_LAG` avec `Xabs = [-1,0,1,2,3]` et $f = \arctan$. On appellera dans la suite P_1 le dernier polynôme trouvé.

Indication *: on tapera* `import numpy as np` *et donc la fonction* arctan *est* `np.arctan`

Comparer $P(1.5)$ et $\arctan(1.5)$.

3. On veut tracer ensemble P_1 et la fonction arctan sur $[-1, 3]$.

On tapera : (***Warning, les quatre instructions de la ligne 27 sont à mettre sur la même ligne, séparés par des ; et de plus, entre les deux - - et entre un " et un - dans*** `linestyle`***, il ne doit pas y avoir d'espace.***)

```
In [25]: import matplotlib.pyplot as plt
In [26]: XX = np.linspace(-1,3,500)
In [27]: plt.plot(XX,np.arctan(XX),linestyle="- -");
plt.plot(XX,P1(XX),color='0'); plt.axis('equal');plt.show()
```

4. On cherche des valeurs approchées de $I = \displaystyle\int_{-1}^{3} \arctan x \, dx$.

4-a Méthode 1. Trouver une valeur approchée de I en important `scipy.integrate` avec l'alias `integr` et la fonction `integr.quad`

4-b Méthode 2. Trouver une valeur approchée de I en utilisant `P2 =P1.integ()`

4-c Méthode 3. En remarquant qu'une primitive de $\arctan x$ est $x \arctan x - \dfrac{1}{2}\ln(x^2+1)$, calculer I.