Université Lyon 1 Année 2018-2019
Licence Mathématiques 2ème année
Introduction à l'Analyse Numérique

TP1 : Interpolation

1 Interpolation

Pour définir une fonction, on le fait en général dans un fichier externe. Pour cela, on ouvre le menu "Fichier" et on créé un nouveau fichier. Une fonction Python se déclare comme suit :

```
def nom_de_fonction(input1, input2, ...):
   instructions...
   return output1, output2, ...
```

où les `input` sont les arguments d'entrée de la fonction et les output les argument de sortie. Ni les uns ni les autres ne sont obligatoires. Il faut ensuite enregistrer votre fichier dans le répertoire de travail avec une extension `.py`.

Attention ! Les indentations sont très importantes en Python. C'est ce qui définie si les instructions font partie de la fonction ou du script. En effet, il n'y a rien qui indique la fin de la fonction à part les indentations !

La fonction `lagrange` suivante construit le polynôme d'interpolation de Lagrange de degré `n` interpolant une fonction `f` définie avec la commande `deff`, en $n+1$ nœuds équi-distribués dans l'intervalle $[a, b]$

```
import numpy as np

def lagrange(f, a, b, n):
    x = np.linspace(a, b, n+1)
    X = np.poly1d([1, 0])
    P = 0
    for i in range(n+1):
        L = 1
        for j in range(0, i):
            L = L * (X - x[j]) / (x[i] - x[j])
        for j in range(i+1, n+1):
            L = L * (X - x[j]) / (x[i] - x[j])

        P = P + L * f(x[i])

    return P
```

1.1 Interpolation de Lagrange

On considère la fonction $f : [0, 3\pi] \to \mathbb{R}$, $x \mapsto \sin(x)$ sur l'intervalle $[0, 3\pi]$.

1. **Polynome de Lagrange**
   (a) Écrire la fonction `lagrange` dans un fichier intitulé `interpolation.py`.
   (b) Définir la fonction f dans un fichier :
   `polynome_lagrange.py`
   qui prend un tableau x en input et retourne le tableau `np.sin(x)`. Penser à importer Numpy au début du fichier. Attention aux tabulations pour la fonction !

(c) Toujours dans le même fichier, définir une distribution uniforme de 100 points dans l'intervalle $[0, 3\pi]$ dans un vecteur `xp`.

(d) Utiliser la fonction `lagrange` pour calculer les polynômes d'interpolations de Lagrange $P_n^{\sin}$ sur $[0, 3\pi]$ pour $n \in [\![1, 5]\!]$. Pour utiliser la fonction `lagrange`, il faut importer le fichier `interpolation.py` en ajoutant la commande suivante au début du fichier : `import interpolation as inter`

On pourra faire une boucle `for`, comme dans le code `lagrange` ou bien utiliser une *comprehension list* (plus dans l'esprit Python) :

```
nmax = 5
Psin = [inter.lagrange(f, 0, 3*np.pi, n) for n in range(1, nmax+1)]
```

(e) Tracer sur une même figure, la fonction f et les 5 polynômes obtenus. Pour cela, on utilisera la fonction de Numpy `polyval` (voir l'aide) qui permet d'évaluer le polynôme $P_n^{\sin}$ obtenu en un point ou un tableau de points `xp`. N'oubliez pas d'importer matplotlib ! Qu'observez-vous ?

RÉPONSE :

(f) Mettre un titre et une légende à la figure.

2. **Erreur d'interpolation**

(a) L'erreur d'interpolation $E_n(\sin)$ est définie par

$$E_n(\sin) = \max_{x \in [0,3\pi]} |\sin(x) - P_n^{\sin}(x)|.$$

Tracer sur un graphe quelques valeurs de $E_n(\sin)$ en fonction de n et commenter le résultat.

RÉPONSE :

(b) En observant que, si $|b - a| \geq 2\pi$, alors, pour tout $n \in \mathbb{N}$,

$$\max_{x \in [a,b]} |\sin^{(n)}(x)| \ = \ 1$$

comparer les valeurs de l'erreur obtenues à la question précédente avec celles fournies par la majoration théorique

$$E_n(f) \leq \frac{1}{4(n+1)} \left(\frac{b-a}{n}\right)^{n+1} \max_{x \in [a,b]} \left|f^{(n+1)}(x)\right|$$

en les traçant sur une même figure.

RÉPONSE :

1.2 Phénomène de Runge

On considère, sur l'intervalle $[-5, 5]$, la fonction

$$f : [-5, 5] \to \mathbb{R}, \quad x \mapsto \frac{1}{1 + x^2}\,.$$

1. **Points équirépartis**

   (a) En s'inspirant de l'exercice précédent, définir dans un fichier `runge.py` la fonction f et un vecteur `xp` contenant une distribution uniforme de 100 points dans l'intervalle $[-5, 5]$.

   (b) Utiliser la fonction `lagrange` pour construire le polynôme d'interpolation de Lagrange P_n^f de degré n, avec $n = 2, 4, 8$ et 12, de f en des nœuds équi-répartis sur $[-5, 5]$ et comparer graphiquement les polynômes obtenus avec la fonction donnée en les traçant sur la même figure. Qu'observe-t-on ?

   RÉPONSE :

   (c) L'erreur d'interpolation $E_n(f)$ est définie par

   $$E_n(f) = \max_{x \in [-5,5]} |f(x) - P_n^f(x)|.$$

   Tracer sur un graphe quelques valeurs $E_n(f)$ en fonction de n et commenter le résultat.

   RÉPONSE :

   Cela s'appelle le *phénomène de Runge*.

2. **Points de Tchebytchev** - Interpoler une fonction en des nœuds équi-distribués sur un intervalle $[a, b]$ n'est pas forcément le meilleur choix, comme le montre l'absence de convergence des interpolations de Lagrange constatée à la question précédente. Pour une interpolation de degré n, on peut montrer que l'erreur sera minimale si les nœuds d'interpolation $(x_k)_{0 \le k \le n}$ sont, à une transformation affine près, les racines du polynôme de Tchebychev $T_{n+1}(x)$, c'est-à-dire si

   $$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2k+1}{2(n+1)}\,\pi\right),\ k \in [\![0, n]\!].$$

   (a) Écrire une fonction `lagrangeTcheb` dans le fichier `interpolation.py` sur le modèle de la fonction `lagrange` pour que l'interpolation se fasse aux points de Tchebychev définis ci-dessus.

   (b) Reprendre les questions 1- et 2- en utilisant la fonction `lagrangeTcheb`.

   RÉPONSE :