TP 2

Ce TP est assorti de plus de remarques et d'indications qu'en temps normal. Vous pouvez bien sûr demander des précisions pendant les "chats" qui essaient de remplacer les TPs.
Il y a par ailleurs des questions qui doivent être résolues à la main, la machine ne fera pas tout.

1 Divers

Exercice 1 Trois "indicateurs"
Soient $x_1, \dots, x_n$ n réels vérifiant $x_1 < x_2 < \cdots < x_n$. On définit les trois fonctions suivantes :

$$f_1(x) = \sum_{i=1}^{n} |x - x_i|; \quad f_2(x) = \sum_{i=1}^{n} (x - x_i)^2; \quad f_\infty(x) = \max_{1 \le i \le n} |x - x_i|.$$

Le but de l'exercice est de tracer le graphe de ces fonctions et de déterminer leur minimum.

Faire une fonction ou un programme prenant comme argument $X = (x_1, \dots x_n)$ (non nécessairement ordonnés) et

- créant, en fonction de X, un vecteur $Y = (y_1, \dots y_p)$ représentant les abscisses (idéalement obtenu avec linspace) ;
- calculant $f_1(y), f_2(y), f_\infty(y)$ pour tout $y = y_i = y[i-1]$ de Y
- traçant le graphe de f_1, f_2, f_∞.

On peut limiter le nombre de boucles car certains calculs peuvent être faits vectoriellement.

Remarque : Cet exercice présente trois "indicateurs de position". Ceux qui sont associés à f_1, f_2 sont classiques (médiane et moyenne) et le troisième est absolument inutile (vous verrez quand vous l'aurez trouvé). Mais il est naturel de se poser la question sur l'indicateur donné par la norme infinie. La médiane est plus "stable", par exemple considérez 10 personnes, donc 9 gagnent 1000 euros par mois et la troisième, une somme $s > 1000$. La médiane reste à 1000 quelle que soit la valeur de s car 5 personnes gagnent plus que 1000 euros et 5 gagnent plus. si l'on fait tendre s vers l'infini, on trouve une moyenne qui tend vers l'infini et l'on se fait une idée fausse du niveau de vie de ces dix personnes.

Indications :

- *Y en fonction de X : prendre pour intervalle de tracé un intervalle contenant le minimum et le maximum des x_i, dépasser un peu (min -1, max $+1$ par exemple).*
- *Commencer par tracer une seule fonction et avec X pas trop gros. On peut utiliser l'instruction* sum.

Exercice 2 Urne de Polya
Principe : Soient a, b, c des entiers > 0. On considère une urne contenant a boules noires et b boules blanches. On tire "au hasard", c'est-à-dire avec une probabilité uniforme, une boule de l'urne, puis on la remet dans l'urne, en rajoutant c boules de la même couleur. L'urne contiendra donc $a + c$ boules noires et b blanches si la boule tirée est noire, a noires et $b + c$ blanches sinon.

1. En ligne de commande, attribuer à a, b les valeurs $2, 3$ puis faire calculer $\frac{a}{a+b}$. Remarquez-vous une anomalie ? Si oui, expliquez-la et remédiez-y.
2. Faire une fonction (Polya) qui simule ce procédé, prenant comme données $[a, b, c]$ et renvoyant $[afinal, bfinal, cfinal]$ (avec $cfinal = c$ mais il faut pouvoir laisser le choix de c à l'utilisateur et avoir des résultats de même type que les données).

3. Faire une fonction (Polya_successif) qui simule N étapes successives de ces tirs et remises. Les données doivent être $[a, b, c, N]$ et les résultats, $[afinal, bfinal]$. Une boucle for peut être utile ici.
4. Calculer la proportion finale de boules noires dans l'urne (fonction proportion_finale, données $[a, b, c, N]$ et résultat p).

Indications :

— *L'anomalie, c'est d'obtenir* $\frac{2}{2+3}$ *égale* 0 *au lieu de* 0.4. *Les* a *et* b *étant entiers, Python comprend que* $\frac{a}{a+b}$ *est un calcul dans les entiers et donne le quotient de la division euclidienne :* $2 = 0 \times 5 + 2$. *Le plus simple est de demander* $a * 1.0/(a+b)$. *Le* $*1.0$ *indique que l'on calcule dans les réels, donc qu'on demande une approximation.*

— *Testez votre programme. Il peut tourner mais vous donner des résultats suspects, comme par exemple, toujours seulement des boules noires. Il faut alors le réparer.*

2 Monte-Carlo

Les exercices suivants présentent une manière probabiliste d'approximer des intégrales. Ils reposent sur le théorème central limite et le théorème de Berry Esseen qui donne la vitesse de convergence dans le TCL : *Soit* (X_n) *une suite de variables aléatoires i.i.d de moments d'ordre* 3 *finis, de moyenne* $m = E(X_1)$ *et de variance* $\sigma^2 > 0$. *On pose* $\rho = E(|X_1 - m|^3)$. *Soit*

$$V_n = \frac{X_1 + \cdots + X_n - nE(X_1)}{\sigma\sqrt{n}},$$

et soit G_n *sa fonction de répartition. Si l'on note* Φ *la fonction de répartition d'une loi normale* $\mathcal{N}(0,1)$, *on a :*

$$\sup_{x\in\mathbb{R}} |G_n(x) - \Phi(x)| \leq \frac{C\rho}{\sigma^3\sqrt{n}},$$

où $C > 0$ *ne dépend pas de* n *ni de la loi de* X_1.

Le TCL, que vous connaissez, donne la limite, le théorème de Berry Esseen donne la vitesse de convergence. Une référence possible est le livre de Bercu-Chafaï.

La simulation ne sert pas ici à se faciliter la vie ou à illustrer un résultat théorique bien connu, elle sert à approximer des intégrales qu'une méthode classique ne permet pas de traiter (parce que c'est en dimension > 1, que la fonction n'est pas assez régulière ou qu'elle est définie sur un intervalle non borné).

Exercice 3 Une intégrale sur $[0,1]^2$

1. Ecrire une fonction, d'argument $N \in \mathbb{N}^*$, qui effectue les opérations suivantes
   (a) simule deux échantillons de N v.a. de loi uniforme $\mathcal{U}([0,1])$, $U = (U_1, \ldots, U_N)$ et $V = (V_1, \ldots, V_N)$;
   (b) calcule, sans boucle, $Z = (f(U_1, V_1), \ldots, f(U_N, V_N))$ pour la fonction $f : (x, y) \mapsto y\sin(xy)^2$;
   (c) Détermine
   $$I = \frac{1}{N}\sum_1^N Z_k = \frac{1}{N}\sum_1^N f(U_k, V_k).$$
2. Comparer à la valeur théorique de I, que l'on peut calculer à la main (faites-le).
3. Refaire le travail de programmation avec la fonction $f : (x, y) \mapsto \sin(xy)^2$.

Exercice 4 Monte Carlo : une intégrale sur $[0, \infty[$

1. Ecrire une fonction, d'arguments $N \in \mathbb{N}^*, \lambda > 0$, qui effectue les opérations suivantes
   (a) simule un échantillon de N v.a. de loi exponentielle $\mathcal{E}(\lambda)$, $U = (U_1, \ldots, U_N)$
   (b) calcule, sans boucle, $Z = (f(U_1), \ldots, f(U_N))$ pour la fonction $f : x \mapsto (1 + x^2)^{-1}$;
   (c) Détermine
   $$I = \frac{1}{N} \sum_1^N Z_k = \frac{1}{N} \sum_1^N f(U_k).$$
   Que représente I ?

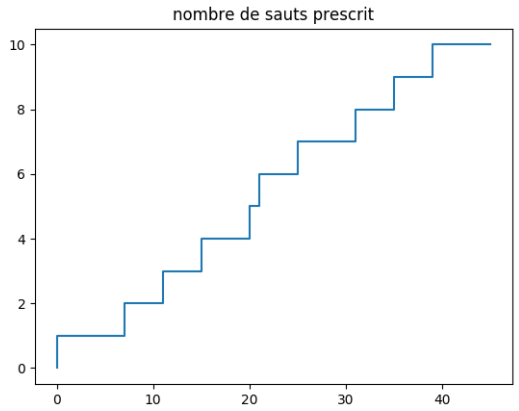
Indications : pas de difficulté spéciale, voir TP 1

3 Processus de Bernoulli

Soit $(Y_n)_{n\geq 1}$ une suite de v.a.i.i.d. de loi $\mathcal{B}(p), 0 < p < 1$. On pose $B_0 = 0$ et $B_n = Y_1 + \cdots + Y_n$. On définit la suite des temps de saut $(S_n)_{n\in\mathbb{N}}$ par $S_0 = 0$ et, sur l'ensemble $(S_n < \infty)$,

$$S_{n+1} = \inf\{k \geq S_n : B_k \neq B_{S_n}\}$$

Cela signifie que $B_{S_n} = B_{(S_n)+1} = \cdots = B_{(S_{n+1})-1}$. Ces temps peuvent valoir $+\infty$. Enfin on pose $E_n = S_n - S_{n-1}$ pour $n \geq 1$ (temps inter-sauts, pendant lesquels rien ne bouge).



On peut démontrer que
— B_n suit une loi binomiale $b(n, p)$;
— B_n est une suite croissante de variables aléatoires (on rajoute 0 ou 1) et $\frac{B_n}{n}$ converge vers $E(Y_1)$ p.s. par la LGN ;
— Les (E_j) constituent une suite de v.a.i.i.d. de loi $\mathcal{G}^*(p)$.

Exercice 5 Étude des instructions

1. Lire l'aide des instructions step, sort.
2. Créer un vecteur X par arange, un vecteur Y de même taille (pas trop gros, pas un million de coordonnées) (Y= X*X pour fixer les idées) et comparer les résultats des instructions plot(X,Y), plot(Y), step(Y), step(X,Y), plot(Y,X), step(Y,X).
   Certaines peuvent renvoyer un message d'erreur. On pourra faire un affichage avec subplot pour avoir un bilan.

3. Créer un vecteur X en utilisant rand et l'ordonner. Noter Z ce nouveau vecteur. Créer les vecteurs Y, W par Y= X*X, W=Z*Z et comparer les résultats des instructions
step(X,Y), step(Y,X), step(Z,W), step(W,Z).
Si les coordonnées du vecteur X ne sont pas rangées dans l'ordre croissant, a-t-on un résultat bien exploitable ?

Indications : les instructions servent à tracer des graphes de fonctions en escalier. Ce ne sont pas vraiment des graphes de fonctions puisque certains points ont une infinité d'images mais l'allure y est.

Exercice 6

Soit $p \in]0,1[$. On note $(X_k)_{1\leq k\leq N}$ une suite de v.a.i.i.d. de loi $\mathcal{B}(p)$ et $S_n = \sum_{k=1}^{n} X_k$.

1. Faire une fonction d'arguments N (taille de l'échantillon), $p \in]0,1[$, $nb \in \mathbb{N}^*$ (nombre de trajectoires voulues) qui simule nb échantillons $(X_1, \dots X_{nb})$, calcule les $(S_1, \dots S_{nb})$ correspondants et les trace (avec un subplot).
2. Représenter une (ou deux) trajectoire(s) de $(S_n)_{1\leq n\leq N}$ modulo 2 et modulo 3. On ne réutilise pas forcément les simulations de la question précédente. Instruction mod

Indications : un subplot peut rentrer dans une boucle, mais il ne prend pas la position 0 *même si les vecteurs et tableaux de Python sont numérotés à partir de* 0.

Exercice 7 Tracé d'une trajectoire sur l'intervalle $[0, N]$.

1. Simuler un échantillon de N v.a.(i.) Y_i suivant une loi de Bernoulli $\mathcal{B}(p)$, calculer les B_i et tracer la trajectoire du processus de Bernoulli correspondante. Elle doit commencer à $(0, 0)$, il faut éventuellement "rajouter" un 0 au vecteur des abscisses ou des ordonnées.
2. Aménager le programme écrit ci-dessus pour déterminer les temps de saut.
Il existe une instruction permettant de *trouver (find)* les coordonnées non nulles d'un vecteur.

Exercice 8 Tracé d'une trajectoire avec n temps de saut.

1. Simuler une famille $(E_1, \dots, E_n)$ de n v.a.i. de loi $\mathcal{G}^*(p)$. (voir un exercice de la feuille précédente).
2. Tracer la trajectoire du processus de Bernoulli dont les temps de saut sont donnés par $S_0 = 0, S_1 = E_1, \dots, S_n = E_1 + \dots + E_n$.
Indication : la fonction à tracer est en escaliers, un saut de 1 se produit à chaque temps de saut, entre deux temps de saut successifs la fonction est constante ...