Préparation aux oraux – informatique

1. *(Centrale)* On définit (f_n) sur $[0,1]$ par

$$f_0(x) = 1 \quad \text{et} \quad f_{n+1}(x) = 2\int_0^x \sqrt{f_n(t)}\mathrm{d}t.$$

Calculer f_1 et f_2 puis montrer que $f_n(x)$ est de la forme $\alpha_n x^{\beta_n}$. Déterminer les relations de récurrence entre les termes des suites (α_n) et (β_n). Calculer β_n et donner la limite de la suite β_n.

Écrire un programme Python qui retourne les 30 premiers termes de la suite (α_n) ; conjecturer la limite.

Montrer que $\ln \alpha_n = -\sum_{k=1}^{n} 2^{-k+1} \ln\left(1 - 2^{-n-1+k}\right)$ et en déduire la limite de α_n.

En déduire que (f_n) converge uniformément sur $[0,1]$.

2. *(Centrale)* On pose $P_0 = 1$, $P_1 = 2X$ et $P_{n+1}(X) = 2XP_n(X) - P_{n-1}(X)$.

Avec Python, calculer P_n pour $n \in [\![2,8]\!]$. Conjecturer la parité, le degré et le coefficient dominant de P_n ; démontrer ces conjectures.

Montrer que $(P,Q) \mapsto (P|Q) = \displaystyle\int_{-1}^{1} \sqrt{1-t^2} P(t) Q(t)\,\mathrm{d}t$ définit un produit scalaire sur $E = \mathbb{R}_8[X]$.

Avec Python, calculer $(P_i|P_j)$ pour $0 \leq i,j \leq 8$. Qu'en déduire pour (P_n) ?

Exprimer la matrice de $\phi : P \mapsto 3XP' - P''$ dans la base des P_n.

3. *(Centrale)* À une liste $L = [a_0, \ldots, a_{n-1}]$ on associe la matrice $M = (m_{i,j})_{1 \leqslant i,j \leqslant n}$ vérifiant

$$m_{i,j} = 1 \quad \text{si } i = j+1, \quad m_{i,n} = -a_{i-1} \quad \text{et} \quad m_{i,j} = 0 \quad \text{sinon.}$$

Créer un programme Python `Matrice(L)` qui renvoie la matrice M liée à la liste $L = [a_0, \ldots, a_{n-1}]$.

Calculer le polynôme caractéristique de M si L est une liste aléatoire de 8 entiers de $[1, 10[$ *(on pourra utiliser la fonction* `poly` *du module* `numpy`*)*. Que peut-on conjecturer ?

Calculer le polynôme caractéristique de M dans le cas général. Montrer que si P est un polynôme de degré strictement inférieur à n, alors $P(M) \neq 0$. En déduire une condition nécessaire et suffisante pour que M soit diagonalisable.

4. *(Centrale)* On note $\sigma(n)$ le cardinal de $H_n = \left\{(p,q) \in \mathbb{N}^2 \ / \ 2p + 3q = n\right\}$. Justifier l'existence de $\sigma(n)$ pour tout entier naturel n. Expliciter $\sigma(0)$, $\sigma(1)$ et $\sigma(2)$; montrer que, pour $n \geq 3$, $\sigma(n) \geq 1$.

Écrire une fonction Python qui calcule $\sigma(n)$ et donner les valeurs pour $n \in [\![0,25]\!]$.

Calculer le rayon de convergence de $\sum \sigma(n) x^n$ et montrer que, en cas de convergence, on a

$$\sum_{n=0}^{\infty} \sigma(n) x^n = \frac{1}{(1-x^2)(1-x^3)}.$$

Écrire une fonction Python basée sur cette série entière permettant de retrouver les valeurs précédentes.

5. *(ENSAM)* Tracer le graphe de $f : x \mapsto \dfrac{1}{1+x^2}$ sur $[-5,5]$.

Donner la définition des polynômes de Lagrange. Écrire une fonction `coef(h,a)` renvoyant la liste des coefficients du polynôme d'interpolation de la fonction h associé à la **liste** a des points d'interpolation *(on pourra déterminer numériquement les coefficients à l'aide de la commande* `solve` *du module* `numpy.linalg`*)*.

Écrire une fonction `evalP(L,x)` qui évalue en x le polynôme dont les coefficients sont donnés dans la **liste** L.

Écrire une fonction `affiche(j)` qui affiche les graphes sur $[-5,5]$ de f et du polynôme de Lagrange associé pour une liste de $j+1$ valeurs équiréparties dans $[-5,5]$, puis tester cette fonction pour $j \in \{5, 10, 15, 20\}$. Commenter.

Tester avec la fonction $f : x \mapsto |x|$. Commenter.

6. *(ENSAM)* Une balle se déplace dans une série de $n+1$ cases ; au début de l'expérience, elle se trouve dans la première et, à chaque étape, avance d'une case ou reste immobile avec équiprobabilité.

Écrire une fonction `tirer(n)` qui renvoie la position finale de la balle (on pourra utiliser `random`, du module `random`). On réitère N fois l'expérience ; donner une fonction Python qui prend en arguments n, N et qui renvoie la liste indexée par k de la proportion de balles étant à la case k en position finale.

Tracer la courbe donnant cette proportion en fonction de k. On prendra $n = 10$ et $N = 100$.

Tracer la courbe théorique, après avoir écrit une fonction renvoyant $\binom{n}{k}$.

La probabilité que la balle avance d'une case est désormais p ; modifier le programme et faire quelques essais.

7. *(ENSAM)* On veut générer la liste des nombres premiers.

Créer une liste `L30` de 31 `True`.

Écrire l'algorithme `modifier(L, p)` tel que, si $p = 0$, le premier terme de `L` est mis à `False`, si $p = 1$ le deuxième terme de `L` est mis à `False` et sinon le k-ième terme de `L` est mis à `False` dès que $k > p$ est un multiple de p. Le tester sur `L30` pour quelques valeurs de p.

Générer la liste des nombres premiers jusqu'à n, en écrivant une fonction `premiers(n)`.

Dire si 823 417 est premier.

8. *(ENSAM)* On dispose d'un fichier texte `Pi.txt` contenant les 10 000 premières décimales de π.

Lire le fichier et le convertir en chaîne de caractères. Afficher les 10 premières et les 10 dernières décimales.

Écrire un programme permettant de vérifier si une chaîne de caractères `motif` est incluse dans une chaîne `source` et qui retourne l'indice du premier caractère si c'est le cas.

Voir si les chaînes 000, 0000, 123, 1234, 314, appartiennent aux 10 000 premières décimales de π.

9. *(ENSAM)* Écrire une fonction `chiffres` qui retourne la liste de chacun des chiffres d'un entier n, ordonnée selon l'ordre des chiffres du nombre. Un nombre de p chiffres est dit *narcissique* si la somme de ses chiffres élevée à la puissance p vaut le nombre lui-même ; par exemple 81 est narcissique.

Écrire une fonction d'argument n qui répond en booléen à la question : n est-il narcissique ?

Retourner tous les nombres narcissiques entre 0 et 10 000.

10. *(Centrale)* On pose $u_n = \left(e - \sum_{k=0}^{n-1} \frac{1}{k!}\right)^{1/n}$ pour $n \in \mathbb{N}^*$.

Avec Python, donner une valeur approchée de nu_n, pour $10 \leq n \leq 30$. Que constate-t-on ?

Pour améliorer la précision des calculs, on peut utiliser le module `mpmath` : après son chargement par `from mpmath import mp` et l'instruction `mp.dps = 50`, les calculs sont faits avec 50 chiffres significatifs, à condition d'utiliser les versions *multiprécision* des fonctions usuelles : `mp.exp`, `mp.factorial`, etc.

Montrer que :

$$\forall n \in \mathbb{N}^* \quad \exists c_n \in [0,1] \quad e - \sum_{k=0}^{n-1} \frac{1}{k!} = \frac{e^{c_n}}{n!}.$$

À l'aide de la formule de Taylor avec reste intégral, déterminer un développement asymptotique à deux termes de c_n. En déduire un développement asymptotique à trois termes de u_n.

11. *(Centrale)* Soit $u_n = \sin\left(\pi\left(2+\sqrt{3}\right)^n\right)$. Avec Python, calculer une valeur approchée des 30 premiers termes de $S_n = \sum_{k=0}^{n} u_k$. Que peut-on conjecturer quant à la série de terme général u_n ?

Étudier cette conjecture en comparant u_n à $v_n = \sin\left(\pi\left(2-\sqrt{3}\right)^n\right)$.

12. *(ENSAM)* On veut tracer la courbe $\mathcal{C}$ d'équation $f(x,y)=0$, où $f(x,y)=x^4+2y^2+2xy-1$.

Écrire une fonction `def f(u)`, où `u` est un vecteur `numpy` de composantes x, y, renvoyant $f(x,y)$.

Écrire de même une fonction `def nablaf(u)` qui renvoie grad $f(x,y)$.

Écrire une fonction `def points(A0, pas, n)` qui renvoie la liste des points A_k construits par récurrence à partir de $A_0 \in \mathcal{C}$ par

$$\forall k \in [\![0, n-1]\!] \quad A_{k+1} = A_k + pas * \vec{v}$$

où $\vec{v}$ est approximativement un vecteur tangent à $\mathcal{C}$ en A_k, lui-même approximativement sur $\mathcal{C}$.

En déduire un tracé approximatif de $\mathcal{C}$.

Comment améliorer la précision en utilisant le développement limité à l'ordre 1 de f au voisinage de A_{k+1} ?

13. *(Centrale)* La probabilité d'obtenir Pile en lançant une pièce est $p \in]0,1[$; on note E_n l'événement : « ne jamais obtenir deux Pile d'affilée au cours des n premiers lancers » et p_n sa probabilité.

Écrire un programme qui prend n et p comme paramètres et renvoie `True` ou `False` selon que E_n est réalisé ou pas.

Montrer que : $p_{n+2} = (1-p)\,p_{n+1} + p\,(1-p)\,p_n$ et en déduire que l'événement « obtenir deux Pile d'affilée » sur une infinité de lancers est presque sûr.

Programmer le calcul par récurrence de p_n et comparer le résultat à celui d'une simulation.

On note T la variable aléatoire qui donne le plus petit $n \geq 2$ tel que l'on obtienne Pile aux lancers de rangs $n-1$ et n.

Donner la fonction génératrice de T. Montrer que T admet une espérance, la calculer et vérifier grâce à une simulation.

14. <u>*Dés de Sicherman*</u> : on lance deux dés cubiques équilibrés, dont les faces portent respectivement les valeurs $(1,2,2,3,3,4)$ et $(1,3,4,5,6,8)$. On note S la variable aléatoire prenant pour valeur la somme des deux résultats fournis par lesdits dés *(on pourra utiliser le module* `Polynomial`*).*

a) Déterminer la loi de probabilité de S. Commenter.

b) Existe-t-il d'autres valeurs dans $\mathbb{N}^*$, pour les faces des deux dés, conduisant à la même loi pour la somme ?

On remarquera que

$$\sum_{k=1}^{6} x^k = x\,(x+1)\left(x^2+x+1\right)\left(x^2-x+1\right)$$

et l'on caractérisera les polynômes correspondant à la fonction génératrice du résultat donné par un dé équilibré dont les faces portent un entier naturel non nul.