DS4 d'informatique (durée : 2h)

Le sujet vous demande d'écrire des fonctions ou morceaux de programmes en Python, attention à indenter correctement votre code, vous pouvez marquer les différents niveaux d'indentation par des lignes verticales.

On pourra à tout moment utiliser une fonction demandée à une question précédente, même si la question n'a pas été traitée.

Si la syntaxe Python vous gène pour répondre à certaines questions, vous pouvez écrire la partie de programme qui pose problème en pseudo-code, c'est-à-dire en langue française, par exemple :

Si a est entier, alors ..., sinon ...

Pour i allant de 1 à 100 faire...

Mesures de Houle

On s'intéresse à des mesures de niveau de la surface libre de la mer effectuées par une bouée (représentée sur la figure 1) [1]. Cette bouée contient un ensemble de capteurs incluant un accéléromètre vertical qui fournit, après un traitement approprié, des mesures à étudier [2].

Figure 1 – Bouée de mesure de houle.

Partie I. Stockage interne des données

Une campagne de mesures a été effectuée. Les caractéristiques de cette campagne sont les suivantes :

- durée de la campagne : 15 jours ;
- durée d'enregistrement : 20 min toutes les demi-heures ;
- fréquence d'échantillonnage : 2 Hz.

Les relevés de la campagne de mesure sont écrits dans un fichier texte dont le contenu est défini comme suit :

- Les informations relatives à la campagne sont rassemblées sur la première ligne du fichier, séparées par des points-virgules (" ; "). On y indique différentes informations importantes comme le numéro de la campagne, le nom du site, le type du capteur, la latitude et la longitude de la bouée, la date et l'heure de la séquence.
- Les lignes suivantes contiennent les mesures du déplacement vertical (en mètre). Chaque ligne comporte 8 caractères (dont le caractère de fin de ligne). Par exemple, on trouvera dans le fichier texte les trois lignes suivantes :

```
+0.4256
+0.3174
-0.0825
...
```

1. Cette étude utilise des résultats extraits de la base de données du Centre d'Archivage National des Données de Houle In Situ. Les acquisitions ont été effectuées par le Centre d'Etudes Techniques Maritimes et Fluviales.
2. L'ensemble des paramètres des états de mer présent dans la base CANDHIS est calculé par les logiciels :
   - Houle5 (CETMEF) : analyse vague par vague (temporelle) ;
   - PADINES (EDF/LNHE) : analyse spectrale et directionnelle (fréquentielle).

1. On suppose que chaque caractère est codé sur 8 bits. En ne tenant pas compte de la première ligne, déterminer le nombre d'octets correspondant à 20 minutes d'enregistrement à la fréquence d'échantillonnage de 2 Hz.
2. En déduire le nombre approximatif (un ordre de grandeur suffira) d'octets contenus dans le fichier correspondant à la campagne de mesures définie précédemment. Une carte mémoire de 1 Go est-elle suffisante ?
3. Si, dans un souci de réduction de la taille du fichier, on souhaitait ôter un chiffre significatif dans les mesures, quel gain relatif d'espace mémoire obtiendrait-on ?
4. Les données se trouvent dans le répertoire de travail sous forme d'un fichier `donnees.txt`. Proposer une suite d'instructions permettant de créer à partir de ce fichier une liste de flottants `liste_niveaux` contenant les valeurs du niveau de la mer. On prendra garde à ne pas insérer dans la liste la première ligne du fichier.

Partie II. Analyse « vague par vague »

On considère ici que la mesure de houle est représentée par un signal $\eta(t) \in \mathbb{R}$, $t \in [0, T]$.
On appelle niveau moyen m la moyenne de $\eta(t)$ sur $[0, T]$.
On définit Z_1, Z_2, ..., Z_n l'ensemble (supposé fini) des Passages par le Niveau moyen en Descente (PND, voir figure 2). À chaque PND, le signal traverse la valeur m en descente.

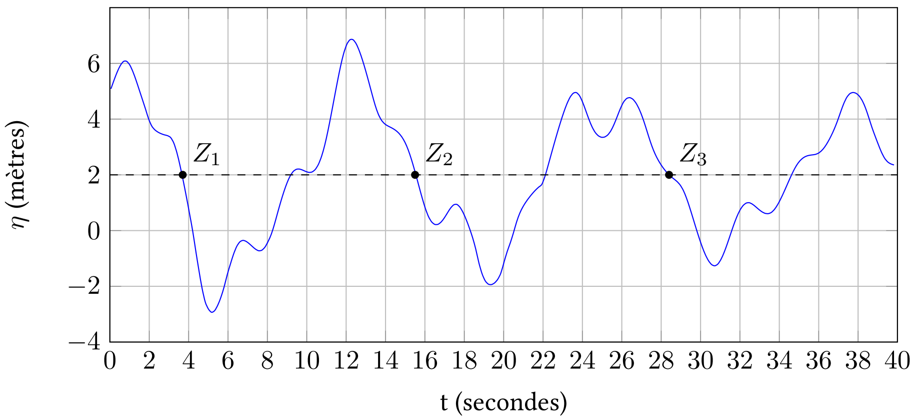

FIGURE 2 – Passage par le Niveau moyen en Descente (PND). Ici la moyenne m vaut 2 m.

Les hauteurs des vagues H_i sont définies par les différences

$$H_i = \max_{t\in[Z_i, Z_{i+1}]} \eta(t) - \min_{t\in[Z_i, Z_{i+1}]} \eta(t) \quad \text{pour} \quad 1 \leqslant i < n \tag{1}$$

On définit les *périodes de vagues* par $T_i = Z_{i+1} - Z_i$.

5. Pour le signal représenté sur la figure 2, que valent approximativement H_1 et H_2 ? Que valent approximativement T_1 et T_2 ?

On adopte désormais une représentation en temps discret du signal. On appelle *horodate* un ensemble (fini) des mesures réalisées sur une période de 20 minutes à une fréquence d'échantillonnage de 2 Hz. Les informations de niveau de surface libre d'un horodate sont stockées dans une liste de flottants `liste_niveaux`. On suppose qu'aucun des éléments de cette liste n'est égal à la moyenne.

6. Proposer une fonction `moyenne(liste_niveaux)` prenant en argument une liste non vide `liste_niveaux` et renvoyant sa moyenne.
7. Pour améliorer l'estimation de la moyenne, on va considérer sa valeur moyenne $\langle\eta\rangle$ définie par

$$\langle\eta\rangle = \frac{1}{T}\int_0^T \eta(t)\mathrm{d}t. \tag{2}$$

On se propose de calculer cette intégrale par la méthode des trapèzes. Compléter la fonction suivante qui calcule l'intégrale par la méthode des trapèzes :

```python
def integrale(liste_niveaux):
  S=...
  h=...
  for i in range(...):
    S += h*(liste_niveaux[i]+liste_niveaux[i+1])/2
  return ...
```

8. Déduire de l'équation (2) une fonction `moyenne_precise(liste_niveaux)`, utilisant la fonction `integrale`, prenant en argument une liste non vide `liste_niveaux` et renvoyant une estimation de la moyenne de η sur la période de 20 minutes de `liste_niveaux`.
9. Proposer une fonction `ind_premier_pnd(liste_niveaux)` renvoyant, s'il existe, l'indice du premier élément de la liste tel que cet élément soit supérieur à la moyenne et l'élément suivant soit inférieur à la moyenne. Cette fonction devra renvoyer -1 si aucun élément vérifiant cette condition n'existe.
10. Proposer une fonction `ind_dernier_pnd(liste_niveaux)` renvoyant, s'il existe, l'indice i du *dernier* élément de la liste tel que cet élément soit supérieur à la moyenne et l'élément suivant soit inférieur à la moyenne. Cette fonction devra retourner -2 si aucun élément vérifiant cette condition n'existe.
11. Quelle est la complexité de cette fonction, dans le meilleur des cas, en fonction du nombre n d'échantillons de la liste `liste_niveaux` ? Comment pourrait-on écrire une fonction avec une complexité $O(1)$ dans le meilleur des cas.

On souhaite stocker dans une liste `successeurs` les indices des points succédant (strictement) aux PND (voir figure 3).

12. On propose la fonction `construction_successeurs` ci-dessous. Elle renvoie la liste `successeurs`. Compléter (sur la copie) les lignes 6 et 7.

```
1 def construction_successeurs(liste_niveaux):
2     n = len(liste_niveaux)
3     successeurs = []
4     m = moyenne(liste_niveaux)
5     for i in range(n-1):
6         if                  # À compléter
7                             # À compléter
8     return successeurs
```

13. En utilisant la fonction précédente, proposer une fonction `decompose_vagues(liste_niveaux)` qui permet de décomposer une liste de niveaux en liste de vagues. On omettra les données précédant le premier PND et celles succédant au dernier PND.
Ainsi `decompose_vagues([1,-1,-2,2,-2,-1,6,4,-2,-5])` (noter que cette liste est de moyenne nulle) renverra `[[-1,-2,2],[-2,-1,6,4]]`.

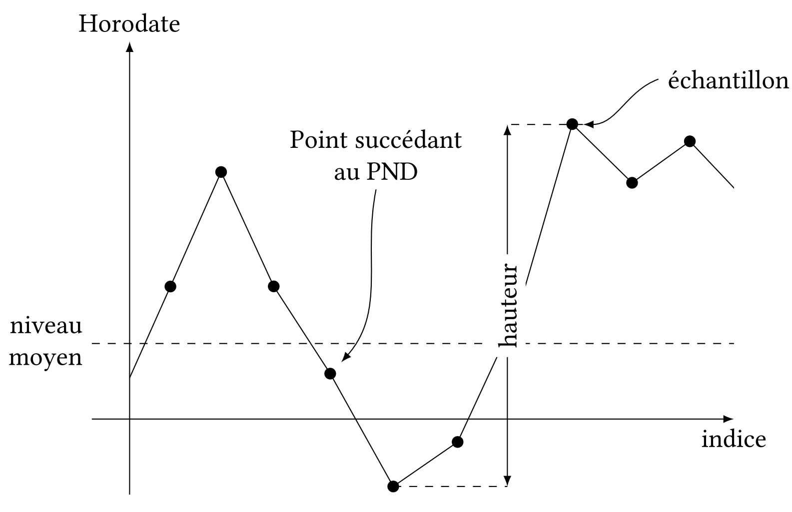

FIGURE 3 – Propriétés d'une vague

On désire maintenant caractériser les vagues.

Ainsi, on cherche à concevoir une fonction `proprietes(liste_niveaux)` renvoyant une liste de listes à deux éléments `[Hi,Ti]` permettant de caractériser *chacune des vagues* i par ses attributs :

— `Hi`, sa hauteur en mètres (m) (voir figure 3) ;
— `Ti`, sa période en secondes (s).

Par exemple `proprietes([1,-1,-2,2,-2,-1,6,4,-2,-5])` renvoie `[[3, 1.5],[8, 2]]`.

14. Pour calculer la hauteur d'une vague, on aura besoin de déterminer le plus grand et le plus petit élément d'une liste. Proposer deux fonctions `plusGrand(liste)` et `plusPetit(liste)` qui renvoient respectivement le plus grand et le plus petit élément de la liste `liste`.
15. Proposer une fonction `proprietes(liste_niveaux)` qui renvoie les propriétés des vagues de la liste `liste_niveaux`.

Partie III. Contrôle des données

Plusieurs indicateurs sont couramment considérés pour définir l'état de la mer. Parmi eux, on note :

— H_{max} : la hauteur de la plus grande vague observée sur l'intervalle d'enregistrement $[0, T]$;

— $H_{1/3}$: la valeur moyenne des hauteurs du tiers supérieur des plus grandes vagues observées sur $[0, T]$;

— $T_{H_{1/3}}$: la valeur moyenne des périodes du tiers supérieur des plus grandes vagues observées sur $[0, T]$.

16. Proposer une fonction `hauteur_max(liste_niveaux)` prenant en argument la liste `liste_niveaux` et renvoyant H_{max}.

Afin de déterminer $H_{1/3}$ et $T_{H_{1/3}}$, il est nécessaire de trier la liste des propriétés des vagues. On suppose que l'on dispose d'une fonction `tri(proprietes)` qui prend en paramètre une liste de propriétés de vagues et qui renvoie la liste triée par hauteur de vague décroissante.

17. Proposer une fonction `hauteur_13(liste_niveaux)` prenant en argument la liste `liste_niveaux` et renvoyant $H_{1/3}$.
18. Proposer une fonction `periode_13(liste_niveaux)` prenant en paramètre la liste `liste_niveaux` et renvoyant $T_{H_{1/3}}$.

La distribution des hauteurs de vague lors de l'analyse *vague par vague* est réputée être gaussienne. On peut contrôler ceci par des tests de *skewness* (variable désignée par S) et de *kurtosis* (variable désignée par K) définis ci-après. Ces deux tests permettent de quantifier respectivement l'asymétrie et l'aplatissement de la distribution.

On appelle $\overline{H}$ et σ^2 les estimateurs non biaisés de l'espérance et de la variance, n le nombre d'éléments $H_1, H_2, \dots, H_n$. On définit alors

$$S = \frac{n}{(n-1)(n-2)} \times \frac{1}{\sigma^3} \times \sum_{i=1}^{n} (H_i - \overline{H})^3$$

$$K = \frac{n}{(n-1)(n-2)(n-3)} \times \frac{1}{\sigma^4} \times \sum_{i=1}^{n} (H_i - \overline{H})^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

Le test suivant est appliqué :

— si la valeur absolue de S est supérieure à 0,3 alors l'horodate est déclaré non valide ;

— si la valeur de K est supérieure à 5 alors l'horodate est déclaré non valide.

On utilise la fonction `moyenne` pour estimer la valeur de $\overline{H}$, et on suppose disposer de la fonction `ecartType` qui permet de renvoyer la valeur de l'écart-type non biaisé σ.

19. Un codage de la fonction `skewness` pour une liste ayant au moins trois éléments est donné ci-dessous. Le temps d'exécution est anormalement long. Proposer une modification simple de la fonction pour diminuer le temps d'exécution (sans remettre en cause l'implémentation des fonctions `ecartType` et `moyenne`).

```python
def skewness(liste_hauteurs):
    n = len(liste_hauteurs)
    et3 = (ecartType(liste_hauteurs))**3
    S = 0
    for i in range(n):
        S += (liste_hauteurs[i] - moyenne(liste_hauteurs))**3
    S = n/(n-1)/(n-2) * S/et3
    return S
```

20. Doit-on s'attendre à une différence de type de la complexité entre une fonction évaluant S et une fonction évaluant K ? La réponse devra être justifiée.