CB

7 mai 2024

- Si, au cours de l'épreuve, vous repérez ce qui vous semble être une erreur d'énoncé, d'une part vous le signalez au surveillant, d'autre part vous le signalez sur votre copie et vous poursuivez la composition en indiquant les raisons des initiatives que vous avez été amené à prendre.
- L'usage de calculatrice, téléphone portable, ordinateur est interdit
- Séparer chaque question de façon claire
- Utiliser les carreaux pour faire une indentation claire et lisible (1 indentation = 2 grands carreaux).
- La **présentation**, la lisibilité, l'orthographe, la qualité de la **rédaction**, **la clarté** et **la syntaxe du code proposé** entreront pour une **part importante** dans **l'appréciation des copies**.

Ne prenez pas la vague sur ce problème houleux !

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].

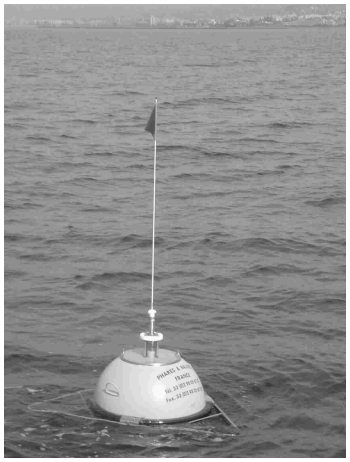

FIGURE 1 – Bouée de mesure de houle.

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 ainsi :

- 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. On suppose que chaque caractère est codé sur 8 bits donc 1 octet.
   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 (*soit 2 mesures par secondes*).
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 SD de 1 Go suffit-elle ?
3. Si, dans un souci de réduction de la taille du fichier, on souhaitait ôter un chiffre significatif dans les mesures, quel gain relatif (en %) 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.

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'Études Techniques Maritimes et Fluviales.
2. L'ensemble des paramètres des états de mer présent dans la base CANDHIS est calculé par le logiciel Houle5 (CETMEF) : analyse vague par vague (temporelle).

La documentation donne des éléments de manipulation de fichiers textuels ci-après.

- `fichier = open(nom_fichier, mode)` ouvre le fichier, en lecture si mode est `"r"`.
- `ligne = fichier.readline()` récupère la ligne suivante de `fichier` ouvert en lecture avec `open`. (*La lecture du fichier reprend alors au début de la ligne suivante.*)
- `lignes = fichier.readlines()` donne la liste des lignes suivantes.
- `fichier.close()` ferme `fichier`, ouvert avec `open`, après son utilisation.
- `ligne.split(sep)` découpe la chaine de caractères `ligne` selon le séparateur `sep` : si `ligne` vaut `"42,43,44"`, alors `ligne.split(",")` renvoie la liste `["42","43","44"]`.

On a donc à partir de maintenant une liste de flottants :

```
liste_niveaux=[0.4256,0.3174,-0.0825,...]
```

Une analyse appelée «vague par vague» est effectuée sur les mesures stockées dans cette liste.

Partie II. Analyse «vague par vague»

On considère ici que la mesure de houle est représentée par un signal $\eta(t) \in \mathbb{R}$, $t \in [0, T]$, avec η une fonction de classe $\mathcal{C}^1$. 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. On suppose $\eta(0) > m$ et $\eta'(0) > 0$, donc $\eta(t) - m \geqslant 0$ sur $[0, Z_1]$ et il y a un maximum en premier.

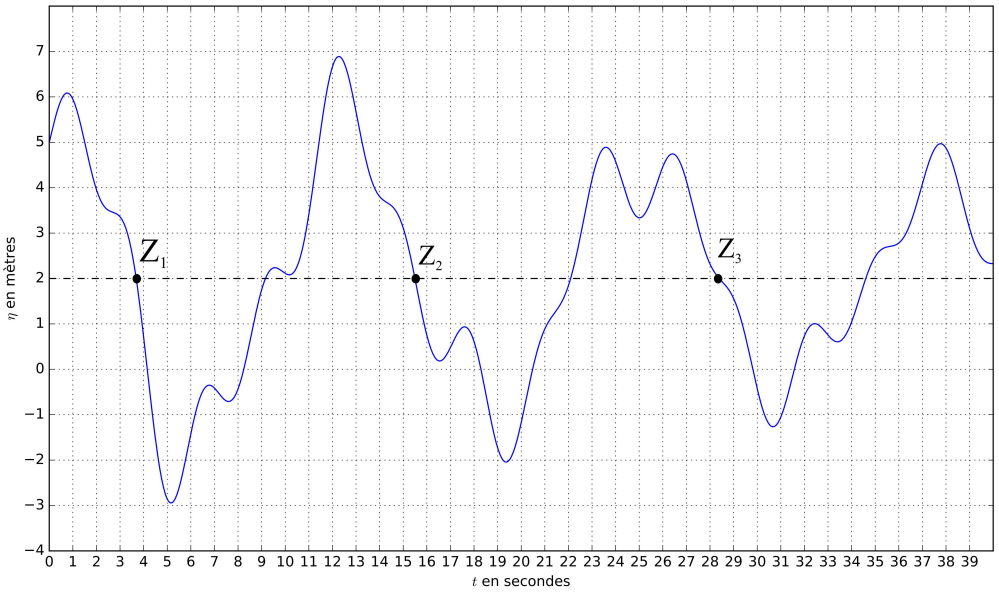

FIGURE 2 – Passage par le Niveau moyen en Descente (PND). Ici la moyenne m vaut 2.

Les hauteurs des vagues H_i sont définies par les différences :

$$\begin{cases} H_1 = \max\limits_{t\in[0,Z_1]} \eta(t) - \min\limits_{t\in[Z_1,Z_2]} \eta(t) \\ H_i = \max\limits_{t\in[Z_{i-1},Z_i]} \eta(t) - \min\limits_{t\in[Z_i,Z_{i+1}]} \eta(t) \quad \text{pour } 2 \leqslant i < n \end{cases}$$

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, H_2 et H_3 ? Que valent approximativement T_1 et T_2 ?

On adopte désormais une représentation en temps discret du signal. On dispose d'un ensemble fini des mesures réalisées sur une période de 20 minutes à une fréquence d'échantillonnage de 2 Hz, ces informations de niveau de surface sont stockées dans la 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` prenant en argument une liste non vide `liste_niveaux` et renvoyant sa valeur moyenne.

7. Écrire les fonctions `max(L)` et `min(L)` qui renvoient respectivement le maximum et le minimum d'une liste non vide de nombres `L`.
8. L'amplitude de la houle est égale la moitié de la différence entre la crête et le creux d'une vague. Une première approche simpliste est donc de prendre la valeur du niveau maximal de la liste (pour estimer la crête) puis celle du niveau minimal (pour estimer le creux).
   Proposer une fonction `amplitude_houle` prenant en argument une liste non vide `liste_niveaux` et renvoyant la valeur de l'amplitude de la houle.

Néanmoins cette approche n'est pas satisfaisante dans la mesure où les valeurs de la crête et du creux retenus dans cette fonction ne sont pas ceux d'une même vague... On va donc affiner.

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 renvoyant 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. On cherchera à proposer une fonction de complexité $\mathcal{O}(1)$ dans le meilleur des cas.

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

11. Recopier et compléter la fonction `construction_successeurs` ci-contre (les lignes 6 et 7) afin qu'elle renvoie la liste `successeurs` des indices des points succédant (strictement) aux PND.

```
def construction_successeurs(liste_niveaux):
    n = len(liste_niveaux)
    successeurs = []
    m = moyenne(liste_niveaux)
    for i in range(n-1):
        if ... #À compléter
            ...#À compléter
    return successeurs
```

12. 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]]`.

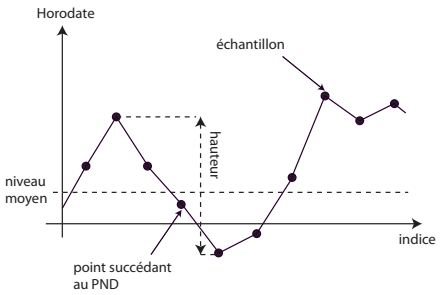

FIGURE 3 – Propriétés d'une vague

On désire maintenant caractériser les vagues. Ainsi, on cherche à concevoir une fonction 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) (figure 3) ;
- `Ti`, sa période en secondes (s).

13. Écrire une fonction `proprietes(liste_niveaux)` réalisant cet objectif.

Partie III. Étude de deux algorithmes de tri

On cherche à trier les valeurs numériques de liste de flottants `liste_niveaux`, on va donc s'intéresser à deux algorithmes de tri que l'on pourra utiliser.

14. La première fonction ci-dessous prend en argument une liste `L` de nombres flottants et en effectue le tri :

```
def tri(L):
    n=len(L)
    for i in range(n):
        for j in range(n-1,i,-1):#j prend les valeurs n-1, n-2, n-3,...,i+1
            if L[j] < L[j-1]:
                L[j],L[j-1] = L[j-1],L[j]# échange d'éléments
```

(a) Cet algorithme trie-t-il les éléments par ordre croissant ou décroissant ?

(b) Quel est le nom de cet algorithme de tri ?

(c) Généralement, quelle est sa complexité ? Justifier.

(d) Lors de l'appel `tri(L)` où `L=[5,2,3,1,4]`, donner le contenu de la liste `L` à la fin de chaque itération de la boucle `for i in range(n):`.

15. Soit la fonction `tri_fusion` suivante :

```
def tri_fusion(L):
    """ Fonction qui prend en argument une liste L de nombres flottants
    et qui trie cette liste"""
    n = len(L)
    if n <= 1:
        return (L)
    else :
        m = n//2
        return fusion(tri_fusion(L[:m]),tri_fusion(L[m:]))
```

(a) Quels sont les deux principes d'algorithmique qu'utilise cet algorithme ?

(b) Écrire une fonction fusion qui prend en arguments deux listes triées `L1` et `L2` et qui renvoie une seule liste triée contenant les éléments de `L1` et `L2`.
Par exemple, l'appel `fusion([2,4,7],[3,5,6,9,11])` renverra la liste `[2,3,4,5,6,7,9,11]`.
On n'utilisera pas la fonction pop et on modifiera pas les listes passées en paramètres.

(c) Dans la pratique, il est constaté que trier une liste avec 15 éléments ou moins prend moins de temps avec la fonction `tri` qu'avec la fonction `tri_fusion`. Écrire donc une nouvelle fonction `tri_fusion2` qui trie la liste par fusion sauf si cette liste à 15 éléments ou moins.

Partie IV. Analyse «spectrale»

L'analyse spectrale (fréquentielle) du niveau permet elle aussi de caractériser l'état de la mer qui peut, en première approximation, être modélisé par une superposition linéaire d'ondes sinusoïdales indépendantes. Pour faire cette analyse, il est possible d'introduire la Transformation de Fourier Discrète (TFD). Sa définition pour un signal numérique x de N échantillons, est la suivante :

$$X_k = \sum_{i=0}^{N-1} x_i \times \mathrm{e}^{-2\pi \mathrm{j} k \frac{i}{N}}, \qquad 0 \leqslant k < N \quad \text{et} \quad \mathrm{j}^2 = -1 \tag{1}$$

Il existe plusieurs méthodes dites de «transformée de Fourier rapide». On étudie dans la suite l'algorithme de Cooley–Tukey adapté de celui de Gauss. On propose ici une réécriture de (1) appelé entrelacement temporel (DIT decimation-in-time).Dans toute la suite, on suppose que N **est une puissance de 2**. On note $w = \mathrm{e}^{-2\pi \mathrm{j}/N}$ (qui est une racine N-ième de l'unité). On pose P_k (TFD des indices pairs) et I_k (TFD des indices impairs) :

$$P_k = \sum_{i=0}^{N/2-1} x_{2i} \times \mathrm{e}^{-2\pi \mathrm{j} k \frac{i}{N/2}} \qquad \text{et} \qquad I_k = \sum_{i=0}^{N/2-1} x_{2i+1} \times \mathrm{e}^{-2\pi \mathrm{j} k \frac{i}{N/2}}$$

On montre alors que pour $0 \leqslant k < \dfrac{N}{2}$,

$$X_k = P_k + w^k I_k \qquad \text{et} \qquad X_{k+N/2} = P_k - w^k I_k$$

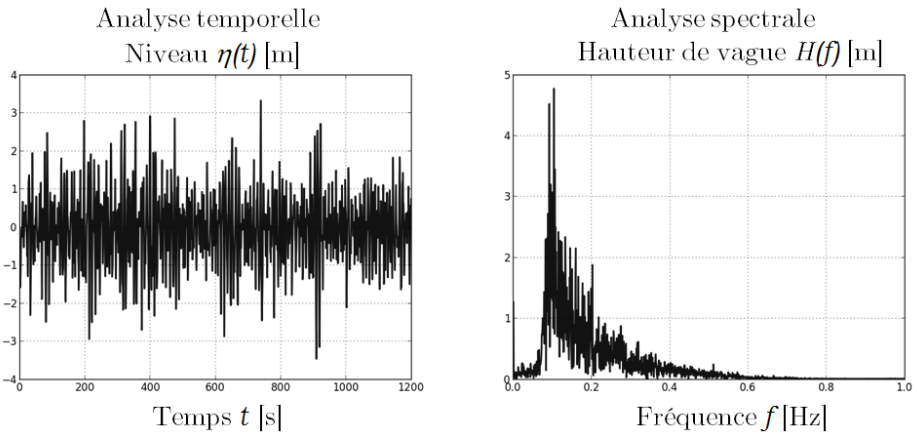

FIGURE 4 – Propriétés d'une vague

L'algorithme est de type «diviser pour régner» : le calcul d'une TFD pour N éléments se fait à l'aide de deux TFD de $N/2$ éléments.

16. Écrire une fonction *récursive* prenant en argument la liste de données `x` et renvoyant la liste `X` obtenue par transformée de Fourier discrète rapide. La longueur de `x` est une puissance de 2.

Rappels concernant le langage `Python`.

Si `l`, `l1`, `l2` désignent des listes en `Python` :

- `len(l)` renvoie la longueur de la liste `l`, c'est-à -dire le nombre d'éléments qu'elle contient. Complexité en $\mathcal{O}(1)$.
- `l1 == l2` teste l'égalité des listes `l1` et `l2`. Complexité en $\mathcal{O}(n)$ avec n le minimum de `len(l1)` et `len(l2)`.
- `l[i]` désigne le `i`-ème élément de la liste `l`, où l'indice `i` est compris entre `0` et `len(l)-1`. Complexité en $\mathcal{O}(1)$.
- `l[i:j]` construit la sous-liste `[l[i], ..., l[j-1]]`. Complexité en $\mathcal{O}(j-i)$. L'usage des variantes `l[i:]` à la place de `l[i:len(l)]`, et de `l[:j]` à la place de `l[0:j]` est aussi autorisé.
- `l.append(e)` modifie la liste `l` en lui ajoutant l'élément `e` en dernière position. Complexité en $\mathcal{O}(1)$.
- `l.pop()` renvoie le dernier élément de la liste `l` (supposée non vide) et supprime l'occurrence de cet élément en dernière position dans la liste. Complexité en $\mathcal{O}(1)$.
- `for i in range(n):` permet de répéter une instruction de i=0 (inclus) à i=n (exclus) en incrémentant de 1 à chaque itération.
- `for i in range(3,n):` permet de répéter une instruction de i=3 (inclus) à i=n (exclus) en incrémentant de 1 à chaque itération.
- `for i in range(3,n,2):` permet de répéter une instruction de i=3 (inclus) à i=n (exclus) en incrémentant de 2 à chaque itération.
- `for i in range(n,3,-1):` permet de répéter une instruction de i=n (inclus) à i=3 (exclus) en décrémentant de 1.
- Manipulation d'un nombre complexe en python : `2+3j` représente 2 + 3i, attention il n'y a pas de * entre 3 et j.
- La fonction exp est utilisable dans le paquet numpy, elle est aussi définie sur les nombres complexes.