

Concours Blanc 2021 – MP1

Durée 3h

Calculatrice collègue autorisée

Le sujet comporte deux problèmes indépendants que vous pouvez réaliser dans l'ordre de votre choix.

Le sujet est (trop) long pour 3h mais le barème en tiendra bien évidemment compte.

I. Simulation numérique du transfert thermique dans un mur en régime transitoire.

MESURES DE HOULE

Le sujet comporte des questions de programmation. Le langage à utiliser est **Python**.

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 étudier².

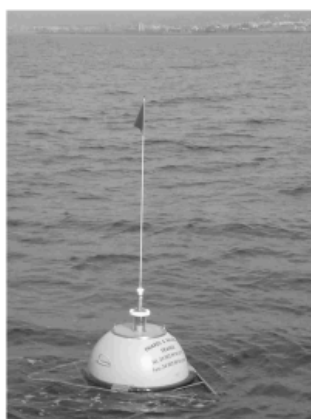


FIGURE 1 : bouée de mesure de houle

Les mesures réalisées à bord de la bouée sont envoyées par liaison radio à une station à terre où elles sont enregistrées, contrôlées et diffusées pour servir à des études scientifiques. Pour plus de sûreté, les mesures sont aussi enregistrées sur une carte mémoire interne à la bouée.

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.

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).

Les lignes suivantes contiennent les mesures du déplacement vertical (m). Chaque ligne comporte 8 caractères dont le caractère de fin de ligne. Par exemple, on trouvera dans le fichier texte les 3 lignes suivantes :

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

□ **Q1** – 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.

□ **Q2** – 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 ?

□ **Q3** – 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 ?

□ **Q4** – 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.

Deux analyses sont effectuées sur les mesures : l’une est appelée “vague par vague”, l’autre est appelée “spectrale”.

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 C^1 .

On appelle niveau moyen m la moyenne de $\eta(t)$ sur $[0, T]$.

On définit Z_1, Z_2, \dots, Z_n l’ensemble (supposé fini) des Passages par le Niveau moyen en Descente (PND) (voir Figure 2). A chaque PND, le signal traverse la valeur m en descente.

On suppose $\eta(0) > m$, et $\frac{d\eta}{dt}(0) > 0$.

On en déduit que $\eta(t) - m \geq 0$ sur $[0, Z_1]$.

Les hauteurs de vagues H_i sont définies par les différences :

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

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

□ **Q5** – 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 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

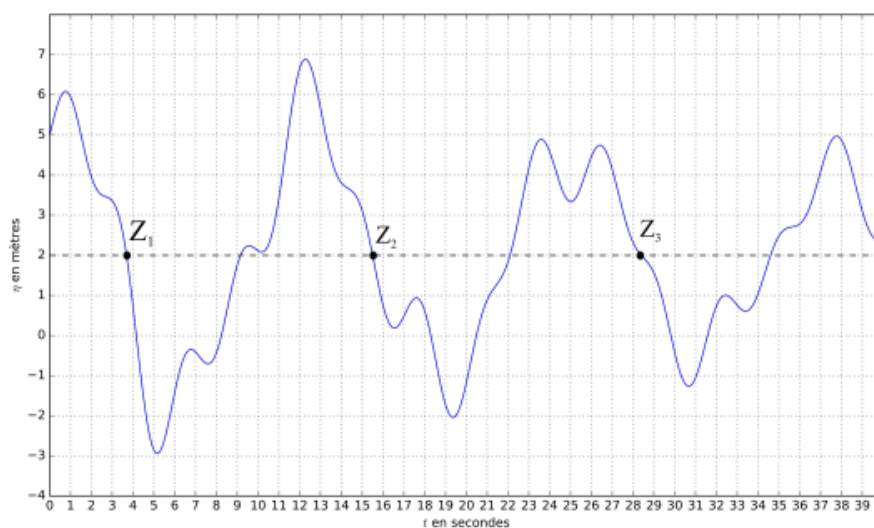


FIGURE 2 : Passages par le Niveau moyen en Descente (PND). Ici la moyenne m vaut 2

liste de flottants `liste_niveaux`. On suppose qu'aucun des éléments de cette liste n'est égal à la moyenne.

❑ **Q6** – Proposer une fonction `moyenne` prenant en argument une liste non vide `liste_niveaux`, et retournant sa valeur moyenne.

❑ **Q7** – Proposer une fonction `integrale_precise` prenant en argument une liste non vide `liste_niveaux`, et retournant la valeur approchée de l'intégrale de η sur une période de 20 minutes. On demande d'utiliser la méthode des trapèzes. En déduire une fonction `moyenne_precise` prenant en argument une liste non vide `liste_niveaux` et retournant une estimation de la moyenne de η sur une période de 20 minutes.

❑ **Q8** – Proposer une fonction `ind_premier_pzd(liste_niveaux)` retournant, 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 retourner -1 si aucun élément vérifiant cette condition n'existe.

❑ **Q9** – Proposer une fonction retournant 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 $O(1)$ dans le meilleur des cas.

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

❑ **Q10** – On propose la fonction `construction_successeurs` en annexe (algorithme 1). Elle retourne la liste `successeurs`. Compléter (sur la copie) les lignes 6 et 7.

❑ **Q11** – 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

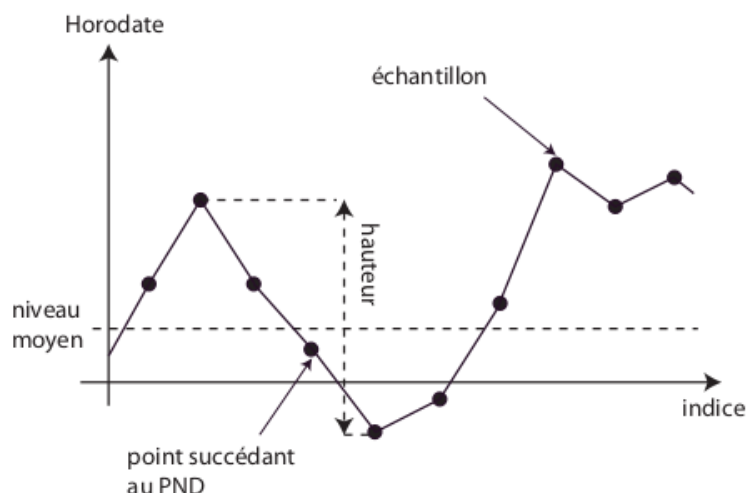


FIGURE 3 : propriétés d'une vague

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) retournera `[[-1, -2, 2], [-2, -1, 6, 4]]`.

On désire maintenant caractériser les vagues.

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

- H_i , sa hauteur en mètres (m) (voir Figure 3),
- T_i , sa période en secondes (s).

□ **Q12** – Proposer une fonction `proprietes(liste_niveaux)` réalisant cet objectif. On pourra utiliser les fonctions de Python `max(L)` et `min(L)` qui retournent le maximum et le minimum d'une liste `L`, respectivement.

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 $[0, T]$;
- $T_{H1/3}$: la valeur moyenne des périodes du tiers supérieur des plus grandes vagues observées sur $[0, T]$.

□ **Q13** – Proposer une fonction prenant en argument la liste `liste_niveaux` de la question 12 et retournant H_{max} .

Afin de déterminer $H_{1/3}$ et $T_{H1/3}$, il est nécessaire de trier la liste des propriétés des vagues. La méthode utilisée ici est un *tri rapide* (*quick sort*). On donne en annexe un algorithme possible pour la fonction `triRapide` (algorithme 2). Trois arguments sont nécessaires : une liste `liste`, et deux indices `g` et `d`.

□ **Q14** – Préciser les valeurs que doivent prendre les arguments `g` et `d` au premier appel de la fonction `triRapide`. Compléter (sur la copie) la ligne 2.

Lorsque le tri rapide est utilisé et que le nombre de données à traiter devient petit dans les sous-listes (de l'ordre de 15), il peut être avantageux d'utiliser un "tri par insertion". On appelle **triInsertion** la fonction qui permet d'effectuer un tri par insertion. Elle admet en argument une liste **liste**, et deux indices **g** et **d**. Ces deux indices permettent de caractériser la sous-partie de la liste à trier (indices de début et de fin inclus).

□ **Q15** – Donner les modifications à apporter à la fonction **triRapide** pour que, lorsque le nombre de données dans une sous-liste devient inférieur ou égal à 15, la fonction **triInsertion** soit appelée pour terminer le tri.

Le code incomplet de la fonction **triInsertion** est donné en annexe : algorithme 3.

□ **Q16** – La fonction **triInsertion** admet trois arguments : une liste de données **liste**, et deux indices **g** et **d**. Elle trie dans l'ordre croissant la partie de la liste comprise entre les indices **g** et **d** inclus. Compléter cette fonction (avec sur la copie le nombre de lignes de votre choix).

La distribution des hauteurs de vague (voir Figure 4) 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.

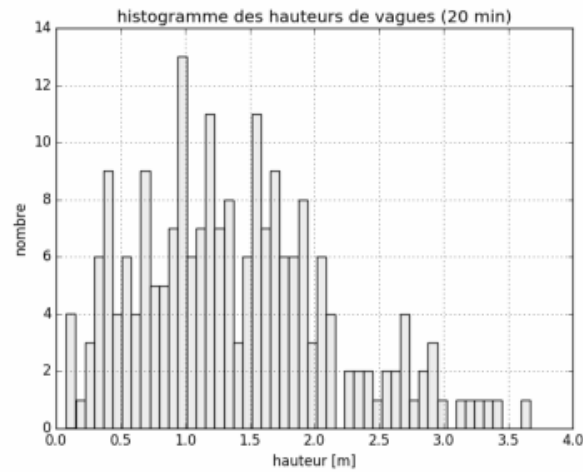


FIGURE 4 : histogramme des hauteurs de vague

On appelle \bar{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

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

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

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 retourner la valeur de l’écart type non biaisé σ .

□ **Q17** – Un codage de la fonction `skewness` pour une liste ayant au moins 3 éléments est donné en annexe (algorithme 4). Le temps d’exécution est anormalement long. Proposer une modification simple de la fonction pour diminuer le temps d’exécution (sans remettre en cause le codage des fonctions `ecartType` et `moyenne`).

□ **Q18** – Doit-on s’attendre à une différence de type de la complexité entre une fonction évaluant S et une fonction évaluant K ?

Partie V. 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.

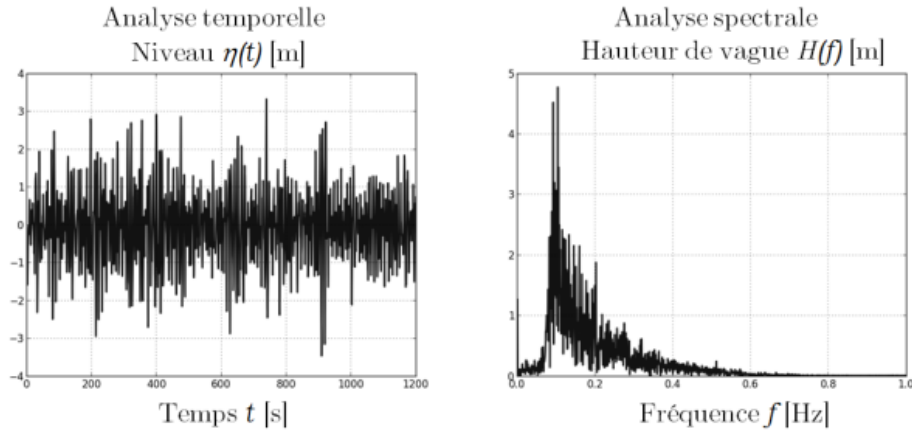


FIGURE 5 : analyses temporelle et spectrale

Des coefficients estimateurs de l’état de la mer issus de l’analyse spectrale ont donc été définis. Parmi eux, on note par exemple H_{m0} la hauteur significative spectrale des vagues ou T_P la période de pic barycentrique.

Pour leur calcul, il est nécessaire d’introduire la Transformation de Fourier Discrète (TFD).

Sa définition pour un signal numérique x de N échantillons, est la suivante :

$$X_k = \sum_{i=0}^{N-1} x_i \times e^{-2\pi j k \frac{i}{N}}, \quad 0 \leq k < N \text{ et } j^2 = -1 \quad (1)$$

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) appelée entrelacement temporel (DIT decimation-in-time).

Dans toute la suite, on suppose que N est une puissance de 2.

On note $w = e^{-\frac{2\pi j}{N}}$ (qui est une racine N -ième de l’unité).

On pose

$$P_k = \sum_{i=0}^{N/2-1} x_{2i} \times e^{-\frac{2\pi j}{N/2} ik}$$

$$I_k = \sum_{i=0}^{N/2-1} x_{2i+1} \times e^{-\frac{2\pi j}{N/2} ik}$$

P_k : TFD des indices pairs, I_k : TFD des indices impairs

On montre alors que pour $0 \leq k < \frac{N}{2}$, $\left| \begin{array}{l} X_k = P_k + w^k I_k \\ X_{k+N/2} = P_k - w^k I_k \end{array} \right|$

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.

□ **Q20** – Quelle est la complexité en temps de cet algorithme en fonction de N ? Justifier en une ou deux lignes.

□ **Q21** – Écrire une fonction *réursive* prenant en argument la liste de données x et retournant la liste X obtenue par transformée de Fourier discrète rapide. La longueur de x est une puissance de 2.

Annexe

Algorithme 1

```

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                                     # A completer
7                                     # A completer
8     return successeurs

```

Algorithme 2

```

1 def triRapide(liste ,g,d):
2     pivot=                                     # A completer
3     i=g
4     j=d
5     while True:
6         while i<=d and liste[i][0]<pivot:
7             i=i+1
8         while j>=g and liste[j][0]>pivot:
9             j=j-1
10        if i>j:
11            break
12        if i<j:
13            liste[i],liste[j]=liste[j],liste[i]
14            i=i+1
15            j=j-1
16        if g<j:
17            triRapide(liste ,g,j)
18        if i<d:
19            triRapide(liste ,i,d)

```

Algorithme 3

```

1 def triInsertion(liste ,g,d):
2     for i in range(g+1,d+1):
3         j=i-1
4         tmp = liste[i]
5         while                                     # A completer
6
7
8
9
10
11
12
13         liste[j+1]=tmp

```

Algorithme 4

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


II. Dynamique gravitationnelle

II Étude de schémas numériques

Soient y une fonction de classe C^2 sur \mathbb{R} et t_{\min} et t_{\max} deux réels tels que $t_{\min} < t_{\max}$. On note I l'intervalle $[t_{\min}, t_{\max}]$. On s'intéresse à une équation différentielle du second ordre de la forme :

$$\forall t \in I \quad y''(t) = f(y(t)) \quad (\text{II.1})$$

où f est une fonction donnée, continue sur \mathbb{R} . De nombreux systèmes physiques peuvent être décrits par une équation de ce type.

On suppose connues les valeurs $y_0 = y(t_{\min})$ et $z_0 = y'(t_{\min})$. On suppose également que le système physique étudié est conservatif. Ce qui entraîne l'existence d'une quantité indépendante du temps (énergie, quantité de mouvement, ...), notée E , qui vérifie l'équation (II.2) où $g' = -f$.

$$\forall t \in I \quad \frac{1}{2}y'(t)^2 + g(y(t)) = E \quad (\text{II.2})$$

II.A – Mise en forme du problème

Pour résoudre numériquement l'équation différentielle (II.1), on introduit la fonction $z : I \rightarrow \mathbb{R}$ définie par $\forall t \in I, z(t) = y'(t)$.

II.A.1) Montrer que l'équation (II.1) peut se mettre sous la forme d'un système différentiel du premier ordre en $z(t)$ et $y(t)$, noté (S).

II.A.2) Soit n un entier strictement supérieur à 1 et $J_n = [0, n-1]$. On pose $h = \frac{t_{\max} - t_{\min}}{n-1}$ et $\forall i \in J_n$, $t_i = t_{\min} + ih$. Montrer que, pour tout entier $i \in [0, n-2]$,

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} z(t) dt \quad \text{et} \quad z(t_{i+1}) = z(t_i) + \int_{t_i}^{t_{i+1}} f(y(t)) dt \quad (\text{II.3})$$

La suite du problème exploite les notations introduites dans cette partie et présente deux méthodes numériques dans lesquelles les intégrales précédentes sont remplacées par une valeur approchée.

II.B – Schéma d'Euler explicite

Dans le schéma d'Euler explicite, chaque terme sous le signe intégrale est remplacé par sa valeur prise en la borne inférieure.

II.B.1) Dans ce schéma, montrer que les équations (II.3) permettent de définir deux suites $(y_i)_{i \in J_n}$ et $(z_i)_{i \in J_n}$, où y_i et z_i sont des valeurs approchées de $y(t_i)$ et $z(t_i)$. Donner les relations de récurrence permettant de déterminer les valeurs de y_i et z_i connaissant y_0 et z_0 .

II.B.2) Écrire une fonction `euler` qui reçoit en argument les paramètres qui vous semblent pertinents et qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i \in J_n}$ et $(z_i)_{i \in J_n}$. Vous justifierez le choix des paramètres transmis à la fonction.

II.B.3) Pour illustrer cette méthode, on considère l'équation différentielle $\forall t \in I, y''(t) = -\omega^2 y(t)$ dans laquelle ω est un nombre réel.

a) Montrer qu'on peut définir une quantité E , indépendante du temps, vérifiant une équation de la forme (II.2).

b) On note E_i la valeur approchée de E à l'instant t_i , $i \in J_n$, calculée en utilisant les valeurs approchées de $y(t_i)$ et $z(t_i)$ obtenues à la question II.B.1. Montrer que $E_{i+1} - E_i = h^2 \omega^2 E_i$.

c) Qu'aurait donné un schéma numérique qui satisfait à la conservation de E ?

d) En portant les valeurs de y_i et z_i sur l'axe des abscisses et l'axe des ordonnées respectivement, quelle serait l'allure du graphe qui respecte la conservation de E ?

e) La mise en œuvre de la méthode d'Euler explicite génère le résultat graphique donné figure 1 à gauche. Dans un système d'unités adapté, les calculs ont été menés en prenant $y_0 = 3$, $z_0 = 0$, $t_{\min} = 0$, $t_{\max} = 3$, $\omega = 2\pi$ et $n = 100$.

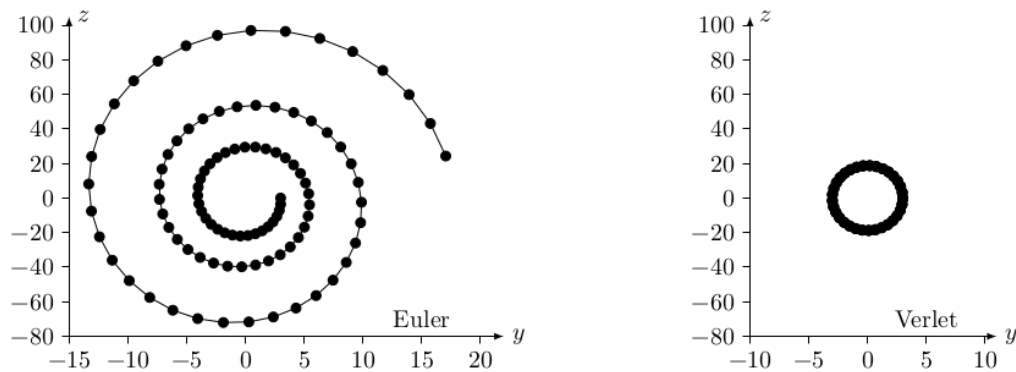


Figure 1

En quoi ce graphe confirme-t-il que le schéma numérique ne conserve pas E ? Pouvez-vous justifier son allure ?

II.C – Schéma de Verlet

Le physicien français Loup Verlet a proposé en 1967 un schéma numérique d'intégration d'une équation de la forme (II.1) dans lequel, en notant $f_i = f(y_i)$ et $f_{i+1} = f(y_{i+1})$, les relations de récurrence s'écrivent

$$y_{i+1} = y_i + h z_i + \frac{h^2}{2} f_i \quad \text{et} \quad z_{i+1} = z_i + \frac{h}{2} (f_i + f_{i+1})$$

II.C.1) Écrire une fonction `verlet` qui reçoit en argument les paramètres qui vous semblent pertinents et qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i \in J_n}$ et $(z_i)_{i \in J_n}$.

II.C.2) On reprend l'exemple de l'oscillateur harmonique (question II.B.3) et on compare les résultats obtenus à l'aide des schémas d'Euler et de Verlet.

a) Montrer que dans le schéma de Verlet, on a $E_{i+1} - E_i = O(h^3)$.

b) La mise en œuvre du schéma de Verlet avec les mêmes paramètres que ceux utilisés au II.B.3e donne le résultat de la figure 1 à droite. Interpréter l'allure de ce graphe.

c) Que peut-on conclure sur le schéma de Verlet ?

III Problème à N corps

On s'intéresse à présent à la dynamique d'un système de N corps massifs en interaction gravitationnelle. Dans la suite, les corps considérés sont assimilés à des points matériels P_j de masses m_j où $j \in \llbracket 0, N-1 \rrbracket$, $N \geq 2$ étant un entier positif donné. Le mouvement de ces points est étudié dans un référentiel galiléen muni d'une base orthonormée. L'interaction entre deux corps j et k est modélisée par la force gravitationnelle. L'action exercée par le corps k sur le corps j est décrite par la force $\vec{F}_{k/j} = G \frac{m_j m_k}{r_{jk}^3} \vec{P}_j \vec{P}_k$ où r_{jk} est la distance séparant les

corps j et k ($r_{jk} = \|\vec{P}_j \vec{P}_k\|$) et $G = 6,67 \times 10^{-11} \text{ N} \cdot \text{m}^2 \cdot \text{kg}^{-2}$ la constante de gravitation universelle.

À tout instant t_i avec $i \in \llbracket 0, n \rrbracket$, chaque corps de masse m_j est repéré par ses coordonnées cartésiennes (x_{ij}, y_{ij}, z_{ij}) et les composantes de son vecteur vitesse $(v_{xij}, v_{yij}, v_{zij})$ dans le référentiel de référence.

Trois listes sont utilisées pour représenter ce système en python

- `masse` conserve les masses de chaque corps : `masse[j] = m_j` ;
- `position` contient les positions successives de chaque corps : `position[i][j] = [x_ij, y_ij, z_ij]` ;
- `vitesse` mémorise les vitesses successives de chaque corps : `vitesse[i][j] = [v_xij, v_yij, v_zij]`.

L'objet de la suite du problème est de construire ces listes en mettant en œuvre l'algorithme de Verlet.

III.A – Position du problème

III.A.1) Exprimer la force \vec{F}_j exercée sur le corps P_j par l'ensemble des autres corps P_k , avec $k \neq j$.

III.A.2) Écrire une fonction python `force2(m1, p1, m2, p2)` qui prend en paramètre les masses (`m1` et `m2` en kilogrammes) et les positions (`p1` et `p2`, sous forme de listes de trois coordonnées cartésiennes en mètres) de deux corps 1 et 2 et qui renvoie la valeur de la force exercée par le corps 2 sur le corps 1, sous la forme d'une liste à trois éléments représentant les composantes de la force dans la base de référence, en newtons.

III.A.3) Écrire une fonction `forceN(j, m, pos)` qui prend en paramètre l'indice j d'un corps, la liste des masses des N corps du système étudié ainsi que la liste de leurs positions et qui renvoie \vec{F}_j , la force exercée par tous les autres corps sur le corps j , sous la forme d'une liste de ses trois composantes cartésiennes.

III.B – Approche numérique

III.B.1) Expliciter la structure et la signification de `position[i]` et `vitesse[i]`.

III.B.2) Écrire une fonction `pos_suiv(m, pos, vit, h)` qui prend en paramètres la liste des masses des N corps du système étudié (en kilogrammes), la liste de leurs positions (en mètres) à l'instant t_i , la liste de leurs vitesses (en mètres par seconde) au même instant et le pas d'intégration h (en secondes) et qui renvoie la liste des positions des N corps à l'instant t_{i+1} calculées en utilisant le schéma de Verlet.

III.B.3) Écrire une fonction `etat_suiv(m, pos, vit, h)` qui prend les mêmes paramètres que la fonction `pos_suiv` et qui renvoie la liste des positions (en mètres) et la liste des vitesses (en m/s) des N corps à l'instant t_{i+1} calculées en utilisant le schéma de Verlet.

III.B.4) En notant τ_N la durée des calculs pour un nombre N de corps, la mise en œuvre de la fonction `etat_suiv` a donné le résultat graphique de la **figure 2** où on a porté $\ln(N)$ en abscisse et $\ln(\tau_N)$ en ordonnée.

a) Quelle relation simple peut-on établir entre $\ln(\tau_N)$ et $\ln(N)$ à partir de la **figure 2** ?

b) Quelle hypothèse peut-on émettre quant à la complexité de l'algorithme étudié ?

III.B.5)

a) Estimer la complexité temporelle de la fonction `etat_suiv` sous la forme $O(N^\alpha)$.

b) Comparer avec le résultat obtenu à la **question III.B.4**.

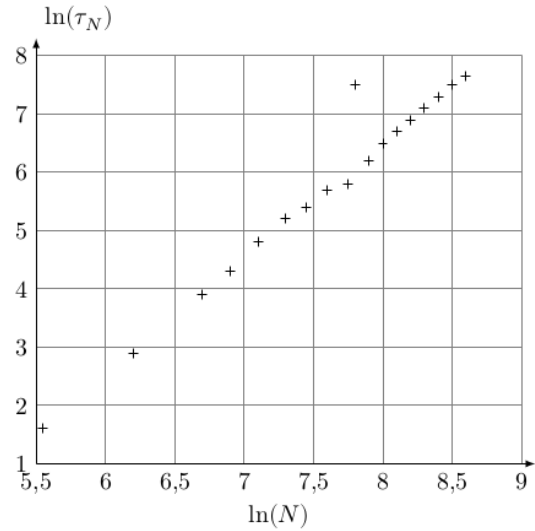


Figure 2