Physique : DM11

On s'intéresse à la résolution d'équations du mouvement dans une approche classique de la mécanique afin d'étudier le mouvement d'une plateforme en mer. Le modèle envisagé est un système à un degré de liberté considéré comme oscillateur harmonique : une masse est reliée à un ressort, avec ou sans amortissement, et peut être soumise à une excitation externe.

La résolution est tout d'abord abordée de façon analytique puis de façon numérique, avant enfin de comparer les résultats obtenus.

Les résolutions analytiques et numériques sont largement indépendantes.

Dans la suite de l'énoncé, toutes les grandeurs vectorielles sont indiquées en gras. Un aide-mémoire sur `numpy` est donné en Annexe.

On considère le mouvement d'une plateforme en mer soumise à un courant marin. Sa partie supérieure de masse m = 110 tonnes est considérée comme rigide et le mouvement principal de la plateforme a lieu suivant x (figure 1a).

Afin d'étudier le mouvement de cette plateforme, on la représente par une masse m, liée à un ressort de constante de raideur k et à un amortisseur de constante d'amortissement γ, pouvant subir une excitation externe de force $\overrightarrow{F_{exc}}$, et se déplaçant sur un support (figure 1b). Le ressort représente la rigidité de l'ensemble du support de la plateforme. L'amortisseur permet de prendre en compte l'effet de l'eau environnante et la force d'excitation externe celui des vagues qui frappent périodiquement la plateforme. La masse est supposée se déplacer selon une seule direction parallèle à l'axe Ox en fonction du temps t.

Les projections sur l'axe Ox de la position, de la vitesse et de l'accélération de la masse en fonction du temps sont notées respectivement $x(t)$, $\dot{x}(t)$ et $\ddot{x}(t)$. La force totale $\overrightarrow{F_{tot}}$ agissant sur la masse correspond à la réaction normale $\overrightarrow{R_N}$ de la base horizontale, à la force de frottement $\overrightarrow{F_d}$, à la force de rappel $\overrightarrow{F_k}$ du ressort, au poids $\overrightarrow{P}$ de la masse et à la force $\overrightarrow{F_{exc}}$ d'excitation externe. La position d'équilibre de la masse sera choisie à $x = 0$. En l'absence d'action de l'amortisseur, la masse se déplace sur la base horizontale sans frottements.

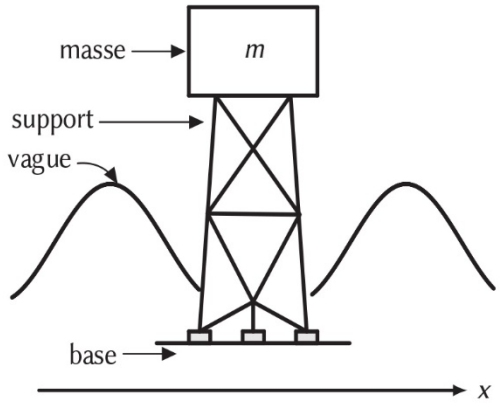
0.1 Résolution analytique et détermination des paramètres pour la modélisation

Q1 — En effectuant une projection sur l'axe Ox, montrer que $\overrightarrow{P}$ et $\overrightarrow{R_N}$ n'interviennent pas dans le bilan des forces.

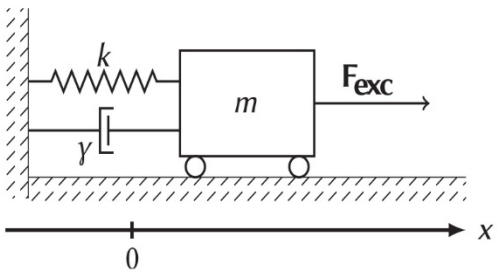
Solution : On applique le PFD dans le référentiel terrestre, galiléen. Le bilan des forces est donné par l'énoncé, on obtient :

$$m\overrightarrow{a} = \sum \overrightarrow{F} = \overrightarrow{R_N} + \overrightarrow{F_d} + \overrightarrow{F_k} + \overrightarrow{P} + \overrightarrow{F_{exc}}$$

Le mouvement est suivant $\overrightarrow{u_x}$ uniquement, la projection sur Oz la verticale indique que le poids et $\overrightarrow{R_N}$



(a) Plateforme en mer soumise aux vagues marines



(b) système masse (m), ressort (k), amortisseur (γ) et excitation externe ($\overrightarrow{F_{exc}}$)

s'annulent, et suivant l'horizontale on obtient l'équation du mouvement :

$$\boxed{m\ddot{x} = 0 + F_d + F_k + 0 + F_{exc}}$$

Le poids et la réaction normale n'interviennent pas dans cette équation du mouvement, mais font cependant bel et bien partie du bilan des forces...

0.1.1 Ressort sans amortissement et sans excitation

Q2 —Démontrer que l'équation du mouvement de la masse correspond à l'équation différentielle du second ordre suivante :

$$m\ddot{x} + kx = 0 \tag{1}$$

Solution : $F_d = 0 = F_{exc}$ traduit les conditions de travail de cette partie, et la force élastique est une force de rappel vers la position d'équilibre. Dans la limite d'élasticité de la structure, $F_k = -kx$ puisque la position d'équilibre est choisie comme origine de l'axe horizontal. On a donc bien finalement $\boxed{m\ddot{x} + kx = 0}$

Q3 — La solution de cette équation prend la forme générale suivante

$$x(t) = A_0 \sin(\omega_0 t) + B_0 \cos(\omega_0 t) \tag{2}$$

avec A_0 et B_0 deux coefficients réels. Exprimer ω_0 en fonction des grandeurs caractéristiques du système et donner sa signification physique. De plus, en remarquant qu'à $t = 0$: $x(t) = x_0$ et $\dot{x}(t) = \dot{x}_0$, déterminer les expressions de A_0 et de B_0 en fonction de x_0, $\dot{x}_0$ et de ω_0.

Solution : L'équation différentielle obtenue est celle d'un oscillateur harmonique de pulsation propre $\boxed{\omega_0 = \sqrt{\frac{k}{m}}}$, cette pulsation est celle des oscillations libres de ce système. La condition initiale sur la position donne $x(t=0) = \boxed{x_0 = B_0}$ et celle sur la vitesse $\omega_0 A_0 = \dot{x}_0$, soit $\boxed{A_0 = \frac{\dot{x}_0}{\omega_0}}$ et donc $\boxed{x(t) = x_0 \cos\omega_0 t + \frac{\dot{x}_0}{\omega_0}\sin\omega_0 t}$

Q4 —On cherche à reformuler l'équation précédente sous une forme plus compacte du type :

$$x(t) = R_0 \cos(\omega_0 t - \varphi_0) \tag{3}$$

Donner les expressions de R_0 et de φ_0 en fonction de x_0, $\dot{x}_0$ et de ω_0.

Solution :

$$x(t) = R_0 \cos(\omega_0 t - \varphi_0) = R_0(\cos(\omega_0 t)\cos(\varphi_0) + \sin(\omega_0 t)\sin(\varphi_0)$$

Par identification avec $x(t) = A_0 \sin(\omega_0 t) + B_0 \cos(\omega_0 t)$, il vient :

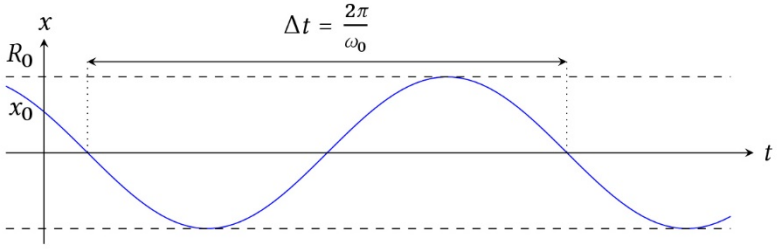
$$\begin{cases} A_0 = R_0 \sin(\varphi_0) \\ B_0 = R_0 \cos(\varphi_0) \end{cases}$$

Ce qui permet d'obtenir R_0 et φ_0 en réinjectant les résultats de la question précédente :

$$\begin{cases} R_0 = \sqrt{A_0^2 + B_0^2} = \sqrt{x_0^2 + \left(\frac{\dot{x}_0}{\omega_0}\right)^2} \\ \tan(\varphi_0) = \frac{A_0}{B_0} = \frac{\dot{x}_0}{\omega_0 x_0} \end{cases}$$

Q5 —Représenter qualitativement $x(t)$ en fonction de t et indiquer sur le tracé R_0, x_0 et $2\pi/\omega_0$.

Solution : R_0 est l'amplitude de la sinusoïde, x_0 sa valeur à $t = 0$ et $2\pi/\omega_0$ la période :



Q6 —En utilisant les expressions des énergies cinétique $K(t)$ et potentielle $U(t)$ du système, montrer que l'énergie totale $E(t)$ du système est alors :

$$E(t) = \frac{kR_0^2}{2} \tag{4}$$

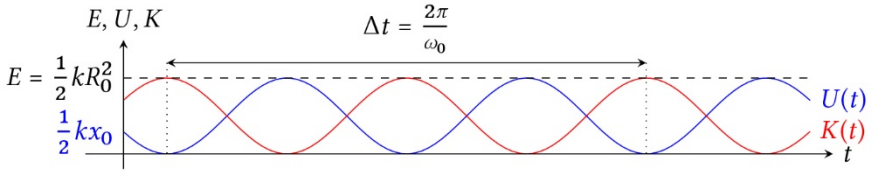
Justifier le résultat obtenu.

Solution : Par définition de l'énergie cinétique, $K(t) = \frac{1}{2}mv^2 = \frac{1}{2}m\left(-R_0\omega_0\sin(\omega_0 t - \varphi_0)\right)^2 = \frac{1}{2}mR_0^2\omega_0^2\sin^2(\omega_0 t - \varphi_0)$. Pour un système élastique de constante de raideur k, l'énergie potentielle élastique s'exprime sous la forme $U = \frac{1}{2}kx^2$, soit $U(t) = \frac{1}{2}k^2R_0^2\cos^2(\omega_0 t - \varphi_0)$. Avec $\omega_0^2 = k/m$, on en déduit $E(t) = K(t) + U(t) = \frac{1}{2}kR_0^2\sin^2(\omega_0 t - \varphi_0) + \frac{1}{2}k^2R_0^2\cos^2(\omega_0 t - \varphi_0) = \boxed{\frac{1}{2}kR_0^2}$.

E est une constante car il n'y a que des forces conservative. Lorsque l'énergie cinétique est nulle, le déplacement esst maximal, vaut $\pm R_0$ et toute l'énergie est sous forme potentielle élastique, d'où cette expression de cette constante.

Q7 —Représenter qualitativement $E(t)$, $K(t)$ et $U(t)$ en fonction de t.

Solution :



0.1.2 Ressort avec amortissement et sans excitation

Q8 —La force de frottement que l'amortisseur exerce sur la masse est considérée comme linéaire, c'est-à-dire proportionnelle au vecteur vitesse $\vec{v}$ de celle-ci : $\vec{F_d} = -\gamma\vec{v}$, avec γ la constante d'amortissement, positive. En considérant une projection sur l'axe Ox, démontrer que la position de la masse en fonction du temps suit l'équation du mouvement ci-après

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0 \tag{5}$$

avec ω_0 défini en question 3 et ζ à exprimer en fonction de γ, k et m.

Solution : On repart de la question 1 avec $F_d = -\gamma\dot{x}$ et toujours $F_{exc} = 0$, on obtient $\boxed{\ddot{x} + \frac{\gamma}{m} + \frac{k}{m}x = 0}$. Par identification avec la forme de l'équation **??**, on obtient bien $\omega_0^2 = k/m$ comme précédemment, et $2\zeta\omega_0 = \frac{\gamma}{m}$, ce qui donne $\boxed{\zeta = \frac{\gamma}{2\sqrt{mk}}}$

Q9 —Dans le cas où $\zeta < 1$, $x(t)$ prend la forme suivante :

$$x(t) = e^{-\zeta\omega_0 t}\left(A_d\cos(\omega_d t) + B_d\sin(\omega_d t)\right) \tag{6}$$

Déterminer les deux coefficients réels A_d et B_d en fonction de x_0, $\dot{x}_0$, ζ, ω_0 et $\omega_d = \omega_0\cdot\sqrt{1-\zeta^2}$. On utilisera pour cela les mêmes conditions initiales que celles utilisées en question 3.

Solution : À $t = 0$, $x(t=0) = x_0 = A_d e^0$, donc $\boxed{A_d = x_0}$. On calcule ensuite $\dot{x}$ pour pouvoir utiliser la seconde condition initiale :

$$\dot{x}(t) = e^{-\zeta\omega_0 t}\left((-\zeta\omega_0 x_0 + B_d\omega_d)\cos(\omega_d t) + (-\zeta\omega_0 B_d - x_0\omega_d)\sin(\omega_d t)\right)$$

d'où $\dot{x}(t=0) = -\zeta\omega_0 x_0 + B_d\omega_d = \dot{x}_0$ et donc $\boxed{B_d = \frac{\dot{x}_0 + \zeta\omega_0 x_0}{\omega_d}}$

Q10 —Montrer alors que l'on peut obtenir une forme du type

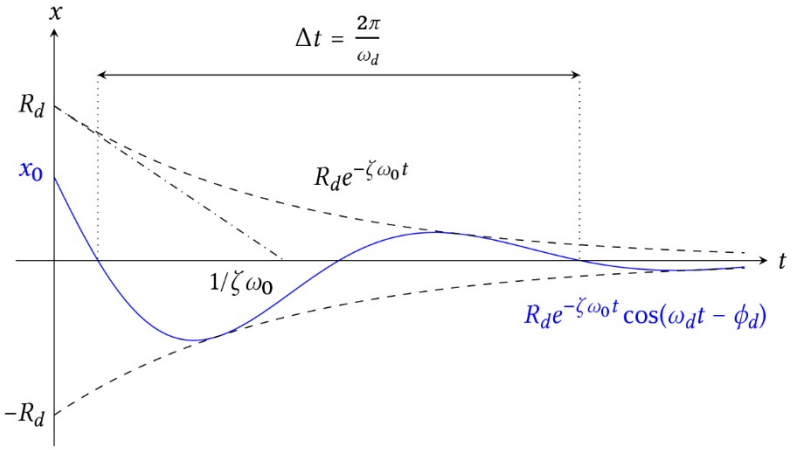
$$x(t) = R_d e^{-\zeta\omega_0 t}\cos(\omega_d t - \phi_d) \tag{7}$$

avec R_d et ϕ_d à préciser.

Solution : On développe $\cos(\omega_d t - \phi_d) = \cos(\omega_d t)\cos(\phi_d) + \sin(\omega_d t)\sin(\phi_d)$, ce qui permet d'identifier les termes entre les deux formes $\begin{cases} A_d = R_d\cos(\phi_d) \\ B_d = R_d\sin(\phi_d) \end{cases}$ ce qui donne $\boxed{\begin{cases} R_d = \sqrt{A_d^2 + B_d^2} \\ \tan(\phi_d) = B_d/A_d \end{cases}}$ On pourrait réinjecter les valeurs de A_d et B_d en fonction des conditions initiales, mais ce n'est pas explicitement demandé et sans intérêt majeur.

Q11 —Représenter qualitativement $x(t)$ en fonction de t et indiquer sur le tracé $R_d e^{-\zeta\omega_0 t}$, x_0 et $2\pi/\omega_d$.

Solution :



Q12 —Donner l'expression de $E(t)$ et commenter les cas où $\zeta = 0$ et $\zeta = 1$.

Solution : Pour $\zeta = 0$, l'énergie E est constante : c'est le cas étudié précédemment. Pour $\zeta = 1$, $\omega_d = 0$ les expressions obtenues précédemment ne sont plus valables car cela correspond au régime critique et donc plus au régime pseudo-périodique sur lequel les expressions fournies par l'énoncé sont valables. On a alors un retour à l'équilibre particulièrement rapide. Pour $1 > \zeta > 0$, on peut essayer de se laisser guider par le calcul précédent :

$$\begin{aligned}
E(t) &= K(t) + U(t) = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2 = \frac{1}{2}k\frac{\dot{x}^2}{\omega_0^2} + \frac{1}{2}kx^2 \\
&= \frac{1}{2}kR_0^2\left(\frac{\mathrm{d}\left(e^{-\zeta\omega_0 t}\cos(\omega_d t-\phi_d)\right)}{\omega_0\,\mathrm{d}\,t}\right)^2 + \frac{1}{2}k^2R_0^2e^{-2\zeta\omega_0 t}\cos^2(\omega_d t-\phi_d) \\
&= \frac{1}{2}kR_0^2\left[e^{-2\zeta\omega_0 t}\left(-\zeta\cos(\omega_d t-\phi_d) - \frac{\omega_d}{\omega_0}\sin(\omega_d t-\phi_d)\right)^2 + e^{-2\zeta\omega_0 t}\cos^2(\omega_d t-\phi_d)\right] \\
&= \frac{1}{2}kR_0^2e^{-2\zeta\omega_0 t}\left(\zeta^2\cos^2(\omega_d t-\phi_d) + (1-\zeta^2)\sin^2(\omega_d t-\phi_d) + \ldots\right. \\
&\qquad \left.\ldots + 2\zeta\sqrt{1-\zeta^2}\cos(\omega_d t-\phi_d)\sin(\omega_d t-\phi_d) + \cos^2(\omega_d t-\phi_d)\right) \\
&= \frac{1}{2}kR_0^2e^{-2\zeta\omega_0 t}\left(1+\zeta^2\left(\cos^2(\omega_d t-\phi_d) - \sin^2(\omega_d t-\phi_d)\right) + 2\zeta\sqrt{1-\zeta^2}\cos(\omega_d t-\phi_d)\sin(\omega_d t-\phi_d)\right) \\
&= \frac{1}{2}kR_0^2e^{-2\zeta\omega_0 t}\left(1+\zeta^2\cos(2\omega_d t-2\phi_d) + \zeta\sqrt{1-\zeta^2}\sin(2\omega_d t-2\phi_d)\right)
\end{aligned}$$

Pas certain de l'intérêt de ce calcul. On retrouve bien la valeur constante pour $\zeta = 0$. Pour $\zeta \longrightarrow 1$ dans cette expression, le terme en sinus disparaît, et le cosinus est constant le temps de la décroissance de l'exponentielle, puisque $\omega_d = 0$. On a alors principalement une décroissance exponentielle de l'énergie selon $E(t) = E_0 e^{-2\omega_0 t}$.

Q13 —Montrer de façon simple que E est une fonction décroissante de t. À quoi cela est-il dû ?

Solution : Le théorème de l'énergie mécanique écrit sur les puissances nous indique que la dérivée temporelle de E est égale à la puissance des forces non conservatives, c'est-à-dire ici la puissance de la force de frottement :

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,t} = -\gamma\vec{v}\cdot\vec{v} = -\gamma\dot{x}^2$$

γ étant une constante positive, cette expresssion est nécessairement toujours négative et l'énergie totale E est bien une fonction décroissante du temps. C'est dû à la caractéristique principale de toute force de frottement, qui s'oppose toujours a déplacement (ce qui ce traduit dans les équations en imposant $\gamma > 0$).

Q14 — On envisage deux temps successifs t_1 et t_2 pour lesquels les déplacements sont x_1 et x_2, tels que $s_2 > t_1$ et $t_2 - t_1 = \tau_d$, avec τ_d : période des oscillations amorties. En utilisant l'équation 7 et en considérant que $\zeta \ll 1$, montrer que :

$$\ln(x_1/x_2) \approx 2\pi\zeta. \tag{8}$$

Solution : Les deux instants étant séparés d'une période de la fonction sinusoïdale, les cosinus ont la même valeur et se simplifient en faisant le rapport, tout comme l'amplitude R_d. Dès lors il ne reste dans ce rapport que les termes exponentiels :

$$\ln(x_1/x_2) = \ln\left(\frac{e^{-\zeta\omega_0 t_1}}{e^{-\zeta\omega_0 t_2}}\right) = \zeta\omega_0(t_2-t_1) = \zeta\omega_0\frac{2\pi}{\omega_d} = \frac{2\pi\zeta}{\sqrt{1-\zeta^2}}$$

Dans la limite $\zeta \ll 1$, $\sqrt{1-\zeta^2} \approx 1$ et on obtient bien $\boxed{\ln(x_1/x_2) \approx 2\pi\zeta}$

Q15 — Le relevé du déplacement horizontal de la plateforme en fonction du temps est représenté en figure 2.
En utilisantl' les deux points qui sont indiqués sur la figure, déterminer k, ζ et γ. Comment ce tracé serait modifié en fonction de la valeur de ζ?

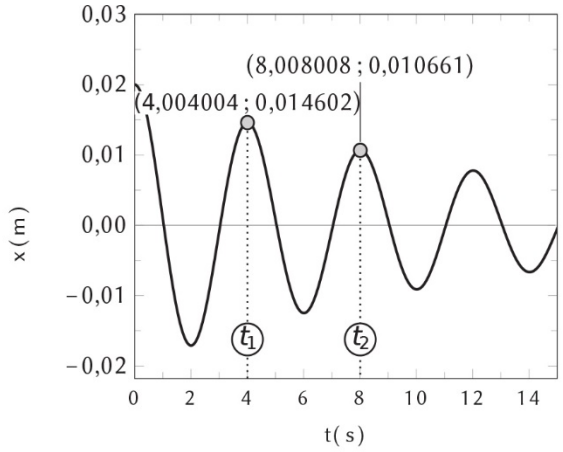


FIGURE 2 : Relevé du déplacement horizontal x (en m) de la plateforme de masse m = 110 tonnes en fonction du temps t (en s). Les deux temps t1 et t2 mentionnés en question 14 sont indiqués.

Solution : La figure nous indique 2 points séparés d'une période de la pseudo-oscillation, on utilise pour ces points le résultat précédent. Avec $x_1 = 1{,}460\,2$ cm et $x_2 = 1{,}066\,1$ cm, $\zeta = \frac{1}{2\pi}\ln(x_1/x_2) = \boxed{5{,}01 \times 10^{-2} = \zeta}$. La condition $\zeta \ll 1$ est raisonnablement vérifiée. En conservant la même approximation, on a $\omega_d \approx \omega_0$, qui peut être évalué via les valeurs de t_1 et t_2 : $\omega_0 \approx \frac{2\pi}{t_2 - t_1}$. Comme $k = m\omega_0^2$, $\boxed{k = m\left(\frac{2\pi}{t_2 - t_1}\right)^2 = 2{,}71 \times 10^5\ \mathrm{kg \cdot s^{-2}}}$.

Enfin, $\boxed{\gamma = 2\zeta\sqrt{mk} = 1{,}73 \times 10^4\ \mathrm{kg \cdot s^{-1}}}$.

Lorsque le coefficient d'amortissement ζ augmente, la décroissance est plus rapide, et lorsqu'on quitte la condition $\zeta \ll 1$, la période des pseudo-oscillation augmente (ω_d décroit avec la croissance de ζ)

0.1.3 Ressort avec amortissement et avec excitation

On envisage enfin le cas où le système est soumis à la fois aux esffets d'amortissement et d'excitation.

On se limite ici à la réponse à une excitation harmonique sinusoïdale de fréquence ω produite par une force extérieure au système

$$\overrightarrow{F_{exc}}(t) = F_0\cos(\omega t)\vec{e_x} \tag{9}$$

avec $\vec{e_x}$ vecteur unitaire sur l'axe Ox et on se place dans le cas traité précédemment pour l'étude de l'amortisseur, c'est-à-dire $\zeta < 1$ (partie 0.1.2).

On admet de plus dans ce qui suit que la réponse du système dans le cas où amortisseur et excitation sont pris en compte peut s'écrire comme somme de la solution donnée par l'équation 6 et de la contribution due à l'excitation :

$$x_{exc}(t) = X\cos(\omega t - \phi). \tag{10}$$

Q16 —Montrer que l'équation différentielle caractérisant le système devient alors :

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = \frac{F_0}{m}\cos(\omega t). \tag{11}$$

Solution : On reprend le PFD établit à la question 1, et on obtient pour la projection de ce PFD sur l'axe Ox $m\ddot{x} = -\gamma\dot{x} - kx + F_0\cos(\omega t)$, ce qui se réécrit :

$$\boxed{\ddot{x} + \frac{\gamma}{m}\dot{x} + \frac{k}{m}x = \frac{F_0}{m}\cos(\omega t)}$$

On a bien l'expression voulue en gardant comme précédemment $\omega_0^2 = \frac{k}{m}$ et $2\zeta\omega_0 = \frac{\gamma}{m}$

Q17 —En utilisant l'équation 10 et en privilégiant une représentation complexe, vérifier que :

$$\begin{cases} X = \dfrac{F_0}{m} \cdot \dfrac{1}{\sqrt{\left(\omega_0^2-\omega^2\right)^2+\left(2\zeta\omega_0\omega\right)^2}} \\ \tan\phi = \dfrac{2\zeta\omega_0\omega}{\omega_0^2-\omega^2} \end{cases} \tag{12}$$

Solution : L'équation différentielle étant linéaire, traiter la solution particulière et la solution de l'équation homogène séparément pour ensuite les additionner et obtenir l'ensemble des solutions ne surprend guère.

On introduit les représentations complexes des grandeurs évoluant sinusoïdalement, la grandeur complexe est soulignée :

$$\square(t) = \square_0 \cos(\omega t + \phi) \leftrightarrow \underline{\square}(t) = \square_0 e^{j\omega t+\phi} = \underline{\square_0} e^{j\omega t}$$

Pour les représentations complexes des grandeurs, l'équation 11 devient, les exponentielles se factorisant puis pouvant être simplifiées (une dérivée temporelle devient une multiplication par $j\omega$) :

$$\underline{X}\left(-\omega^2 + 2\zeta\omega_0 j\omega + \omega_0^2\right) = \frac{F_0}{m}$$

D'où :

$$\underline{X} = \frac{\dfrac{F_0}{m}}{-\omega^2 + 2\zeta\omega_0 j\omega + \omega_0^2}$$

X et ϕ sont respectivement le module et l'argument de $\underline{X}$, on en déduit bien les résultats annoncés :

$$\begin{cases} X = |\underline{X}| = \dfrac{\frac{F_0}{m}}{\sqrt{\left(\omega_0^2-\omega^2\right)^2+\left(2\zeta\omega_0\omega\right)^2}} \\ \tan\phi = -\dfrac{Im(\omega_0^2-\omega^2+j\cdot 2\zeta\omega_0\omega)}{Re(\omega_0^2-\omega^2+j\cdot 2\zeta\omega_0\omega)} = \dfrac{2\zeta\omega_0\omega}{\omega_0^2-\omega^2} \end{cases}$$

Q18 —Exprimer la grandeur $M = \dfrac{X}{F_0/k}$ en fonction de $r = \omega/\omega_0$ et expliciter le sens physique de M.

Solution : $M = \dfrac{\frac{k}{m}}{\sqrt{\left(\omega_0^2-\omega^2\right)^2+\left(2\zeta\omega_0\omega\right)^2}} = \dfrac{1}{\sqrt{\frac{\left(\omega_0^2-\omega^2\right)^2}{\omega_0^4}+\frac{\left(2\zeta\omega_0\omega\right)^2}{\omega_0^4}}} = \dfrac{1}{\sqrt{\left(1-r^2\right)^2+4\zeta^2 r^2}}$

M est un facteur sans dimension qui caractérise la réponse fréquentielle du système, ce qui ramène au cas classique d'une fonction de transfert lorsque l'excitation et la réponse sont mesurées sur la même grandeur physique.

Q19 —Trouver la condition sur r puis sur ω pour laquelle M est maximale.

Solution : $M(r)$ atteindra son maximum lorsque $\left(1-r^2\right)^2 + 4\zeta^2 r^2 = r^4 + (4\zeta^2 - 2)r^2 + 1$ sera minimal. Cette expression est un polynôme du second degré en $R = r^2$, le minimum est atteint en « $-b/2a$ », soit pour $R = 1 - 2\zeta^2$, soit $r = \sqrt{1-2\zeta^2}$ si $\zeta < \dfrac{1}{\sqrt{2}}$. Dans la limite $\zeta \ll 1$ du problème, cette condition est bien vérifiée. On en déduit la pulsation de résonance pour laquelle $M(\omega)$ est maximale, $\omega_r = \omega_0 \cdot r = \omega_0\sqrt{1-2\zeta^2}$

Q20 —Si l'on considère une période moyenne des vagues en mer de 8 s, que peut-on conclure sur le mouvement de la plateforme ?

Solution : La période moyenne des vagues en mer est de 8 s, la valeur du coefficient d'amortissement et de la période propre obtenus en exploitant la figure 2 nous permettent de dire que la fréquence de résonance

sera proche de la fréquence propre, puisque $\omega_r = \omega_0\sqrt{1-2\zeta^2} \approx \omega_0$ pour $\zeta \ll 1$. La plateforme peut entrer en résonance pour une excitation de période proche de 4 s, pour une excitation avec une période de 8 s on est suffisamment loin de la résonance pour ne pas avoir le développement de mouvements de grande amplitude.

0.2 Modélisation : codage

On souhaite maintenant obtenir $x(t)$ et $E(t)$ de façon numérique et comparer les résultats obtenus à ceux fournis par les solutions analytiques précédentes pour $x(t)$. On rappelle que $x(t)$ et $E(t)$ représentent respectivement la position de la masse et l'énergie mécanique totale en fonction du temps.

Pour cela, le temps est discrétisé en N points $t = 0, \Delta t, 2\Delta t, ..., (N-1)\Delta t$ avec un pas de temps constant Δt. Les $N-1$ pas sont effectués pendant la simulation de durée totale t_{max}. On note respectivement x_n, v_n, a_n, E_n et F_n les valeurs de $x(t)$, $\dot{x}(t)$, $\ddot{x}(t)$, $E(t)$ et $\overrightarrow{F_{exc}}(t)$ à $t = n\Delta t$.

À chaque pas, les équations du mouvement reliant x_{n+1} et v_{n+1} à x_n et v_n sont utilisées afin d'obtenir les valeurs de x, v et E. Les conditions initiales x_0 et v_0 sont connues et permettent de démarrer le processus d'intégration numérique. Deux algorithmes distincts (Euler et Leapfrog) vont être utilisés dans la suite.

Pour l'écriture du code, on se place dans le cas le plus général, c'est-à-dire avec amortissement et excitation harmonique externe. Les variables et tableaux choisis sont notamment listés dans la table 1

`N` : nombre N de points sur l'axe des temps utilisés pendant toute la simulation
`t[]` : tableau des temps t (s), de dimension N
`x[]` : tableau des positions x (m), de dimension N
`v[]` : tableau des vitesses v ($m \cdot s^{-1}$), de dimension N
`E[]` : tableau des énergies totales (J), de dimension N
`F[]` : tableau des forces d'excitation (N), de dimension N
`dt` : pas de temps Δt (s)
`tmax` : temps total de la simulation t_{max} (s)
`k` : constante de raideur k du ressort ($N \cdot m^{-1}$)
`m` : masse m du système (kg)
`om0` : ω_0 (s^{-1})
`zeta` : ζ (sans unité)

TABLE 1 : Principaux tableaux et principales variables utilisés pour la résolution numérique

On rappelle qu'un aide-mémoire sur `numpy` est fourni en Annexe, page 7.

Q21 —Écrire les lignes de code permettant de définir l'entier N. On suppose t_{max} et Δt connus et fixés par l'utilisateur en début de code.

Solution : Le nombre de points est le nombre d'intervalles plus un, on divise le temps total par la durée d'un intervalle pour avoir le nombre d'intervalles. Le code étant exécuté pour la rédaction de ce document, les réponses fournies contiennent un peu plus que ce qui est demandé afin d'illustrer les commande sans obtenir de message d'erreur. Ainsi pour cette question, la ligne 5 est le seul code demandé.

```
1 >>> from math import *
2 >>> import numpy as np
3 >>> tmax=5
4 >>> dt=1
5 >>> N=int(tmax/dt + 1)
6 >>> N
7 6
```

Q22 —Écrire alors l'instruction permettant de définir le tableau `t` qui contient toutes les valeurs de t telles que : $0s \leq t \leq t_{max}$. On rappelle que le temps est discrétisé en N points $t = 0, \Delta t, 2\Delta t, ..., (N-1)\Delta t$ avec un pas de temps constant Δt et que $N-1$ pas sont effectués.

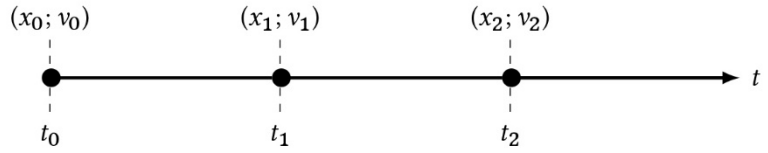


FIGURE 3 : Discrétisation en temps utilisée dans le cas de l'algorithme d'EULER. Les conditions initiales correspondent à $(x_0; v_0)$

Solution :

```
>>> t=np.linspace(0,tmax,N)
>>> t
array([0., 1., 2., 3., 4., 5.])
```

Q23 —En effectuant des développements de TAYLOR de x et v tronqués à l'ordre 1, on obtient l'algorithme d'EULER, où x et v sont évalués au même temps t selon le schéma donné figure 3 :

$$\begin{cases} x_{n+1} \approx & x_n + v_n \cdot \Delta t \\ v_{n+1} \approx & v_n + a_n \cdot \Delta t \end{cases} \tag{13}$$

Donner les expressions de x_{n+1} et v_{n+1} en fonction de x_n, v_n et F_n.

Solution : On remplace dans la formule de récurrence l'accélération a_n par son expression obtenue à partir de l'équation différentielle du système, $a_n = \frac{F_n}{m} - 2\zeta\omega_0 v_n - \omega_0^2 x_n$. On en déduit :

$$\begin{cases} x_{n+1} \approx & x_n + v_n \cdot \Delta t \\ v_{n+1} \approx & v_n + \left(\frac{F_n}{m} - 2\zeta\omega_0 v_n - \omega_0^2 x_n\right) \cdot \Delta t \end{cases}$$

Q24 —Écrire la boucle en `i` permettant d'obtenir toutes les valeurs de `x[i+1]`, `v[i+1]` et `E[i+1]` où `i` correspond à un point sur l'axe des temps. On précisera les variables éventuellement introduites en supposant qu'elles ont été définies dans le code.

Solution :

```
>>> ## les paramètres expérimentaux doivent être rentrés: zeta, w0, la force
>>> zeta=5e-2
>>> k=2.7e5
>>> om0=2*np.pi/4 #T0=4s
>>> om=2*np.pi/8 #Tv=8s pour la période des vagues
>>> F0=1e5

>>> F=np.linspace(-F0,F0,N) #arbitrairement, pour exister et F/m raisonnable
>>> m=110e3 #110 tonnes

>>> # initialisation des tableaux, cf Annexe
>>> x=np.zeros(N)
>>> v=np.zeros(N)
>>> E=np.zeros(N)

>>> # valeurs initiales
>>> x0=0.02
```

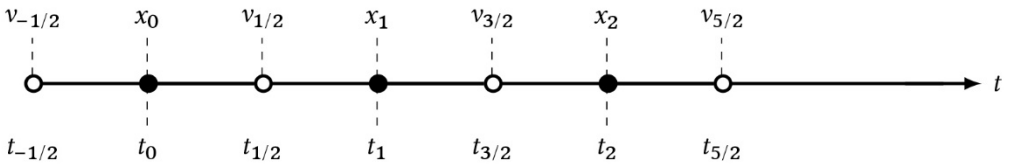


FIGURE 4 : Discrétisation en temps utilisée dans le cas de l'algorithme de Leapfrog. Les conditions initiales correspondent à $(x_0; v_{-1/2})$

```
>>> v0=0
>>> E0=0.5*k*x0**2+0.5*m*v0**2

>>> ## la question commence là:
>>> x[0]=x0
>>> v[0]=v0

>>> for i in range(N-1):
...     x[i+1]=x[i]+v[i]*dt
...     v[i+1]=v[i]+(F[i]/m-2*zeta*om0*v[i]-x[i]*om0**2)*dt
...     E[i+1]=0.5*k*x[i+1]**2+0.5*m*v[i+1]**2
...
>>> x
array([ 0.02      ,  0.02      , -0.93843893, -2.34112919, -1.38979832,
        5.37042079])
>>> v
array([ 0.        , -0.95843893, -1.40269026,  0.95133088,  6.7602191 ,
        9.67297081])
>>> E
array([      0.        ,   50577.28515526,  227104.82831241,
        789696.27126188, 2774288.74267099, 9039741.6627917 ])
```

Q25 —Dans l'algorithme de Leapfrog, les x sont évalués aux temps entiers, c'est à dire à $t = 0, \Delta t, 2\Delta t, ..., (N-1)\Delta t$, alors que les v sont évalués à $t = -\Delta t/2, \Delta t/2, 3\Delta t/2, ..., (N-1)\Delta t - \Delta t/2$ selon le schéma donné figure 4. Ainsi, pour cet algorithme, `x[i]` représente de façon approchée la position à l'instant $i\Delta t$, et `v[i]` représente de façon approchée la vitesse à l'instant $(i-1/2)\Delta t$.

Pour le système considéré, montrer alors que x_{n+1} et $v_{n+1/2}$ prennent les formes suivantes :

$$\begin{cases} x_{n+1} \approx & x_n + v_{n+1/2} \cdot \Delta t \\ v_{n+1/2} \approx & -\frac{\omega_0^2 \Delta t}{1+\zeta\omega_0\Delta t} x_n + \frac{1-\zeta\omega_0\Delta t}{1+\zeta\omega_0\Delta t} v_{n-1/2} + \frac{F_n}{m(1+\zeta\omega_0\Delta t)} \Delta t \end{cases} \tag{14}$$

Solution : Le principe du décalage se comprend en considérant que pour évaluer au mieux la valeur d'une fonction Δt plus tard, il faut connaitre la valeur moyenne de la vitesse sur cet intervalle, et que celle-ci est plus probablement proche de celle du milieu de l'intervalle qu'à une extrémité. On a alors comme équivalent de l'équation 13 :

$$\begin{cases} x_{n+1} \approx & x_n + v_{n+1/2} \cdot \Delta t \\ v_{n+1/2} \approx & v_{n-1/2} + a_n \cdot \Delta t \end{cases}$$

a_n peut être obtenu par l'équation différentielle, mais celle-ci fait intervenir v_n dont on ne dispose pas. On le remplace dans l'équation différentielle par la valeur moyenne entre les valeurs de $v_{m\pm1/2}$, l'équation différentielle permet donc d'écrire :

$$a_n = \frac{F_n}{m} - 2\zeta\omega_0 \frac{v_{n-1/2} + v_{n+1/2}}{2} - \omega_0^2 x_n$$

L'équation de récurrence pour les vitesses devient alors :

$$\left(1+\zeta\omega_0\Delta t\right)v_{n+1/2} \approx \left(1-\zeta\omega_0\Delta t\right)v_{n-1/2} + \left(\frac{F_n}{m} - \omega_0^2 x_n\right)\cdot\Delta t$$

On obtient alors bien :

$$\begin{cases} x_{n+1} \approx & x_n + v_{n+1/2}\cdot\Delta t \\ v_{n+1/2} \approx & -\frac{\omega_0^2\Delta t}{1+\zeta\omega_0\Delta t}x_n + \frac{1-\zeta\omega_0\Delta t}{1+\zeta\omega_0\Delta t}v_{n-1/2} + \frac{F_n}{m(1+\zeta\omega_0\Delta t)}\Delta t \end{cases}$$

Q26 —Quel problème pose l'évaluation de `E[i+1]` ? Dans la suite, on préfèrera ainsi évaluer `E[i]`.

Solution : Pour évaluer `E[i+1]`, il faut la position et la vitesse à l'instant $(i+1)\Delta t$. La vitesse n'étant pas calculée sur les temps entiers, il faut l'évaluer par sa moyenne entre les valeurs décalées de $\Delta t/2$ avant et après, ce qui demande d'avoir accès à la valeur qui sera stockée dans `v[i+2]` et qui ne sera évaluée qu'à l'itération suivante de la boucle. Calculer `E[i]` par contre est réalisable dans la boucle qui puisqu'on aura accès à toutes les valeurs de x et de v utiles.

Q27 —Comment obtenir `E[i]` à partir de `x[i]` et `v[i]` ?

Solution : $E_n = \frac{1}{2}kx_n^2 + \frac{1}{2}mv_n^2 = \frac{1}{2}kx_n^2 + \frac{1}{2}m\left(\frac{v_{n-1/2}+v_{n+1/2}}{2}\right)^2$ ce qui ce code en :

```
E[i]=0.5*k*x[i]**2+0.125*m*(v[i]+v[i+1])**2
```

Q28 — En introduisant deux variables `fac1` et `fac2` correspondant respectivement à $1-\zeta\omega_0\Delta t$ et $\frac{1}{1+\zeta\omega_0\Delta t}$, écrire la boucle en `i` permettant d'obtenir toutes les valeurs de `x[i+1]` et `v[i+1]`.

Solution :

```
>>> ### les facteurs de proportionnalité
>>> fac1=1-zeta*om0*dt
>>> fac2=1/(1+zeta*om0*dt)

>>> ### initialisation: les tableaux existent déjà, je réinitialise juste les
 ↪  valeurs initiales

>>> x[0]=x0
>>> v[0]=v0

>>> ### la boucle demandée:

>>> for i in range(N-1):
...     v[i+1]=-fac2*om0**2*dt*x[i]+fac1*fac2*v[i]+F[i]*fac2*dt/m
...     x[i+1]=x[i]+v[i+1]*dt
...
>>> x
array([ 0.02      , -0.86864492, -0.14638139,  0.63699303,  0.01759004,
       -0.04610936])
>>> v
array([ 0.        , -0.88864492,  0.72226353,  0.78337442, -0.61940299,
       -0.0636994 ])
```

Q29 —Compléter la boucle de la question 28 avec le calcul du terme d'énergie totale.

Solution :

```
>>> for i in range(N-1):
...     v[i+1]=-fac2*om0**2*dt*x[i]+fac1*fac2*v[i]+F[i]*fac2*dt/m
...     x[i+1]=x[i]+v[i+1]*dt
...     E[i]=0.5*k*x[i]**2+0.125*m*(v[i]+v[i+1])**2
...
>>> ### Il reste le dernier élément du tableau E à remplir, pour lequel il faut un
 ↪  calcul de vitesse supplémentaire:

>>> vsup=-fac2*om0**2*dt*x[N-1]+fac1*fac2*v[N-1]+F[N-1]*fac2*dt/m
>>> E[N-1]=0.5*k*x[N-1]**2+0.125*m*(v[N-1]+vsup)**2

>>> E
array([ 10912.23471717, 102244.07814066,  34063.21676746,  55147.30811376,
         6457.91733222,   9765.20164435])
```

Q30 —Écrire une fonction `integration(F)` qui prend en argument le tableau `F[]` des forces d'excitation et renvoie les tableaux `x[]`, `v[]` et `E[]` complétés.

On introduira pour cela une variable `algo` supposée définie en global, permettant d'appliquer l'algorithme d'Euler si `algo==0` ou de Leapfrog si `algo==1`.

On considèrera également le cas $t = 0$.

Solution :

```
>>> def integration(F):
...     ''' À partir d'une excitation donnée par un tableau numpy, retourne les
 ↪  tableaux x, v et E, initialisés à 0.'''
...     global algo # 0 pour Euler, 1 pour Leapfrop
...     ### Initialisation. Le calcul de E[0] est omis car tout est au repos
...     x=np.zeros(len(F))
...     v=np.zeros(len(F))
...     E=np.zeros(len(F))
...     ## Valeurs initiales
...     x[0],v[0]=x0,v0
...     ### Boucles
...     if algo==0:
...         # Euler
...         for i in range(N-1):
...             x[i+1]=x[i]+v[i]*dt
...             v[i+1]=v[i]+(F[i]/m-2*zeta*om0*v[i]-x[i]*om0**2)*dt
...             E[i+1]=0.5*k*x[i+1]**2+0.5*m*v[i+1]**2
...     elif algo==1:
...         # Leapfrog
...         for i in range(N-1):
...             v[i+1]=-fac2*om0**2*dt*x[i]+fac1*fac2*v[i]+F[i]*fac2*dt/m
...             x[i+1]=x[i]+v[i+1]*dt
...             E[i]=0.5*k*x[i]**2+0.125*m*(v[i]+v[i+1])**2
...         # dernier élément du tableau E
...         vsup=-fac2*om0**2*dt*x[N-1]+fac1*fac2*v[N-1]+F[N-1]*fac2*dt/m
...         E[N-1]=0.5*k*x[N-1]**2+0.125*m*(v[N-1]+vsup)**2
...     else:
...         print("Erreur: pas d'algo reconnu!")
```

```
...     return x,v,E
...
>>> algo=1
>>> x,v,E=integration(F)
>>> x
array([ 0.02      , -0.86864492, -0.14638139,  0.63699303,  0.01759004,
       -0.04610936])
>>> v
array([ 0.        , -0.88864492,  0.72226353,  0.78337442, -0.61940299,
       -0.0636994 ])
>>> E
array([ 10912.23471717, 102244.07814066,  34063.21676746,  55147.30811376,
         6457.91733222,   9765.20164435])
```

Q31 — Écrire une fonction `force(f,t,w)` qui prend en argument F_0, t, et ω de l'équation 9 et retourne la valeur de $\overrightarrow{F_{exc}}$ pour un temps `t` donné.

Solution :

```
>>> def force(f,t,w):
...     ''' Calcul et retourne la force suivant x à un instant donné t, en
 ↪  fonction de son amplitude et de sa pulsation.'''
...     return f*np.cos(w*t)
...
>>> force(2,3,2*np.pi/6)
-2.0
```

Q32 —Écrire une fonction `force_exc()` qui complète et retourne le tableau `F[]` des forces d'excitation en fonction du booléen `exc` défini globalement valant `True` si une excitation est appliquée au système, `False` sinon.

Dans le cas où `exc==True`, on appellera la fonction force définie précédemment en question 31. Dans le cas où `exc==False`, on prendra alors : `F[i]=0`, `∀i`.

Solution : C'est étrange de ne pas passer les caractéristiques de la force en argument, mais les arguments sont toujours spécifiés dans ce sujet.Je considère que l'amplitude de la force `F0` et sa pulsation `om` sont définis par ailleurs avant l'appel et que Python ira les chercher en les considérant comme des variables globales.

```
>>> def force_exc():
...     ''' Retourne le tableau décrivant la force d'excitation'''
...     global exc
...     F=np.zeros(N)
...     if exc:
...         for i in range(len(F)):
...             F[i]=force(F0,i*dt,om)
...     return F
...
```

Q33 —Donner alors les lignes de code permettant de réaliser la simulation numérique à partir des fonctions précédentes.

Solution :

```
>>> ### on reprend la définition des paramètres du problème:
>>> zeta = 5e-2
>>> k = 2.7e5
>>> om0 = 2*np.pi/4 #T0=4s
>>> om = 2*np.pi/8 #Tv=8s pour la période des vagues
>>> F0 = 5e3 #N, au pif, mais tel que x soit proche des 2cm en statique d'après la
 ↪  valeur de k et celle de x0...
>>> m = 110e3 #110 tonnes
>>> x0 = 0.02 #m
>>> v0 = 0 #m.s^-1

>>> tmax=10 #s d'après partie C
>>> dt=0.05 #s d'après le tableau fourni

>>> N=int(tmax/dt + 1)

>>> algo=1
>>> exc=0

>>> x,v,E=integration(force_exc())

>>> x[:10]
array([0.02      , 0.01988561, 0.01967415, 0.01938097, 0.01901964,
       0.01860216, 0.01813908, 0.01763971, 0.01711218, 0.0165636 ])
>>> len(E)
201
```

Q34 —On souhaite désormais estimer la qualité des résultats numériques par rapport aux données analytiques de référence. Écrire une fonction `ema(d,dref)` qui calcule l'erreur maximale absolue entre un jeu de données numériques `dn` et analytiques `da`. On supposera que les tableaux `dn` et `da` sont de même dimension n.

Solution :

```
>>> # pour le test
>>> dn=np.array([1.1,1.9,3.2,3.6])
>>> da=np.array([1,2,3,4])

>>> def ema(d,dref):
...     return np.max(abs(d-dref)) # par défaut sur un tableau numpy les
 ↪  opérations se font sur chaque élément.
...
>>> ema(dn,da)
0.3999999999999999
```

Q35 —Pour le problème particulier qui nous intéresse, si l'on souhaite appliquer cette fonction aux tableaux contenant les données numériques et analytiques pour E, quel est l'indice maximal de ces tableaux à considérer ?

Réécrire alors la fonction `ema` pour qu'elle soit applicable aux deux algorithmes considérés.

Solution : À lire cette question, il n'était probablement pas attendu de calculer le dernier élément du tableau des énergies : l'ayant fait, la fonction précédente fonctionne pour les deux algorithmes. Si ce n'était pas le cas, il faudrait dans le cas d'un calcul avec l'algorithme de Leapfrog ne pas tenir compte du dernier point. On peut tester la valeur de la variable globale pour essayer de savoir quel algorithme, ou poser la question en le demandant en paramètre. Pour allier les deux possibilités, on peut ajouter un paramètre optionnel qu'on initialise à la valeur de la variable globale s'il n'est pas renseigné.

	EMA	
Δt (s)	Euler	Leapfrog
0,050	128,0203160	0,0837314
0,010	15,1373529	0,0033484
0,005	7,1159053	0,0008371
0,001	1,3556230	0,0000335

Table 2 : EMA obtenues pour E (en J) par rapport à la valeur analytique, pour différents pas de temps Δt, dans le cas où il n'y a pas d'amortissement et d'excitation externe.

```
>>> def ema2(d,dref,algorithme=None):
...     global algo
...     if algorithme==None:
...         algorithme=algo
...     if algorithme==1:
...         return np.max(abs(d[:-1]-dref[:-1]))
...     else:
...         return np.max(abs(d-dref))
...
>>> print(ema2(da,dn),'en utilisant la variable globable par défaut,',algo)
0.20000000000000018 en utilisant la variable globable par défaut, 1
>>> algo=0
>>> print(ema2(da,dn),'en utilisant la variable globable par défaut,',algo)
0.3999999999999999 en utilisant la variable globable par défaut, 0
>>> print(ema2(da,dn,1),'en précisant utiliser Leapfrog')
0.20000000000000018 en précisant utiliser Leapfrog
>>> print(ema2(da,dn,0),'en précisant utiliser Euler')
0.3999999999999999 en précisant utiliser Euler
```

0.3 Modélisation : analyse des résultats d'un cas simple

Toutes les données numérique suivantes ont été obtenues avec : t_{max} = 10 s, m = 110 tonns, x_0 = 0,02 m et v_0 = 0 m · s^{-1} et la valeur de k obtenue en question 15, dans le cas d'un système sans amortissement et sans excitation externe. Le tableau 2 présente les erreurs maximales absolues (EMA) calculées avec la fonction ema des énergies obtenues numériquement par rapport à celles obtenues analytiquement, pour les deux algorithmes envisagés et divers pas de temps.

Q36 —Justifier l'ordre de grandeur des Δt considérés du tableau 2 pour la discrétisation en temps utilisée.

Solution : Les Δt doivent être petits par rapport aux durées caractéristiques des évolutions pour pouvoir les suivre, donc petits par rapport à la plus petite période caractéristique du système. Ici, c'est T_0 = 4 s, on a bien $\Delta t \ll T_0$ pour toutes les valeurs, tout en gardant un nombre de points facilement gérable pour les calculs à faire ($N \approx 1 \times 10^4$ sur 10 s).

Q37 —En utilisant les données numériques du tableau 2 donner l'ordre approximatif de l'erreur globale sur E des deux algorithmes considérés. Justifier votre réponse.

Solution : On cherche un coefficient n tel que $EMA \approx A(\Delta t)^n$, on peut donc regarder s'il y a un nombre $n = \frac{\ln(EMA(\Delta t_1)/EMA(\Delta t_2)}{ln(\Delta t_1/\Delta t_2)}$ qui se dégage. En prenant Δt_1 = 0,001 s comme référence, on obtient n = 1,16, 1,045, 1,040 et donc n = 1 probablement pour l'ordre de l'erreur pour l'algorithme d'Euler et n = 2 (remarquablement précisément !) pour Leapfrog.

Q38 —Dans le cas simple de l'algorithme d'Euler, comment augmente E lorsqu'on passe de t_n à t_{n+1} ?

Relier alors ce résultat aux données du tableau 2.

Solution : L'algorithme d'Euler est défini par l'équation 13, évaluons $E_{n+1} - E_n$ à partir de là.

$$\begin{aligned}
E_{n+1} &= \frac{1}{2}k\left(x_n + v_n\cdot\Delta t\right)^2 + \frac{1}{2}m\left(v_n + a_n\cdot\Delta t\right)^2\\
E_{n+1} &= \frac{1}{2}k\left(x_n^2 + 2x_nv_n\Delta t + v_n^2(\Delta t)^2\right) + \frac{1}{2}m\left(v_n^2 + 2v_na_n\Delta t + a_n^2(\Delta t)^2\right)\\
E_n &= \frac{1}{2}kx_n^2 + \frac{1}{2}mv_n^2\\
E_{n+1} - E_n &= \frac{1}{2}k\left(2x_nv_n\Delta t + v_n^2(\Delta t)^2\right) + \frac{1}{2}m\left(2v_na_n\Delta t + a_n^2(\Delta t)^2\right)\\
E_{n+1} - E_n &= \left(kx_nv_n + mv_na_n\right)\Delta t + \left(\frac{1}{2}kv_n^2 + \frac{1}{2}ka_n^2\right)(\Delta t)^2
\end{aligned}$$

Le premier terme $P_n = kx_nv_n + mv_na_n$) correspond à ce qu'on attend pour l'évaluation des puissances dans l'algorithme d'Euler, la méthode utilisée ici s'écarte de ce calcul en ajoutant un terme toujours positif $\left(\frac{1}{2}kv_n^2 + \frac{1}{2}ka_n^2\right)(\Delta t)^2$ sur chaque pas de temps.

Si on regarde l'effet sur une intégration sur une durée finie t_{max} divisée en N pas, l'effet cumulatif donne une erreur en $N \times c\left(\frac{t_{max}}{N}\right)^2$, on retrouve une erreur cumulée proportionnelle à $1/N$, le coefficient de proportionnalité c correspondant à la moyenne d'une somme de termes tous positifs n'ayant aucune raison ni de tendre vers 0 ni de diverger.

Une propriété importante que devrait vérifier un algorithme d'intégration est la réversibilité dans le temps : en partant des positions et vitesses d'un temps $t + \Delta t$ et en appliquant un pas de temps $-\Delta t$, un algorithme réversible en temps devrait redonner les positions et vitesses du temps t.

En pratique, en partant d'un couple $(x_n; v_n)$, on applique donc tout d'abord un pas Δt pour déterminer $(x_{n+1}; v_{n+1})$, puis on applique un pas $-\Delta t$ afin d'obtenir $(\bar{x_n}; \bar{v_n})$. Si $(\bar{x_n}; \bar{v_n}) = (x_n; v_n)$, alors l'algorithme est dit réversible en temps.

Q39 —Pourquoi s'agit-il d'une propriété importante à vérifier pour le problème considéré ?

Solution : Les équations de la mécanique sont réversibles, c'est fondamentalement bien que les méthodes numériques d'intégration vérifient cette propriété.

Pour un système dont le mouvement est périodique, on espère se retrouver après intégration sur une période dans le même état. Sur chaque demi-période successive dans le problème étudié, on peut considérer qu'on reproduit le même mouvement en inversant le sens du temps. Si la méthode est réversible, les erreurs accumulées sur une demi-période pourront annuler celles accumulées sur la demi-période suivante.

Q40 —Donner l'expression de $\bar{x_n}$ en fonction de x_n pour les deux algorithmes considérés. Peut-on déjà conclure sur lsilité en stemps de chacun de ces algorithmes ?

Solution : Pour Euler :

$$x_n \xrightarrow{+\Delta t} x_{n+1} = x_n + v_n\Delta t \xrightarrow{-\Delta t} \bar{x_n} = x_{n+1} - v_{n+1}\Delta t = x_n + (v_n - v_{n+1})\Delta t$$

Pour Leapfrog :

$$x_n \xrightarrow{+\Delta t} x_{n+1} = x_n + \frac{v_{n-1/2} + v_{n+1/2}}{2}\Delta t \xrightarrow{-\Delta t} \bar{x_n} = x_{n+1} - \frac{v_{n+1/2} + v_{n-1/2}}{2}\Delta t = x_n$$

L'algorithme d'Euler n'est pas réversible puisque les deux valeurs ne sont pas identiques. Pour l'algorithme de Leapfrog, la vérification nécessaire pour les positions est validée, reste à vérifier que cet algorithme est également réversible pour les évaluations des vitesses (il faut que positions et vitesses soient les mêmes pour se retrouver dans le même état physique, au même point de l'espace des phases.)

Q41 —Donner alors l'expression de $\bar{v_n}$ en fonction de v_n lorsque nécessaire et conclure sur la réversibilité en temps.

Solution : On doit traiter l'algorithme de Leapfrog pour les vitesses :

$$v_{n-1/2} \xrightarrow{+\Delta t} v_{n+1/2} = v_{n+1/2} + a_n \Delta t \xrightarrow{-\Delta t} v_{n-1/2} = v_{n+1/2} - a_n \Delta t = v_{n-1/2}$$

La réversibilité est donc vérifiée pour les vitesses également, il l'est donc pour un état du système : l'algorithme de Leapfrog est réversible.

Q42 —On s'intéresse enfin à une simulation de plus grande durée ($t_{max} = 60$ s). La figure 5 donne l'évolution de l'erreur absolue sur x entre données numériques et analytiques ($|e_x|$) pour les deux algorithmes choisis. Que peut-on mettre en évidence sur cette figure ?

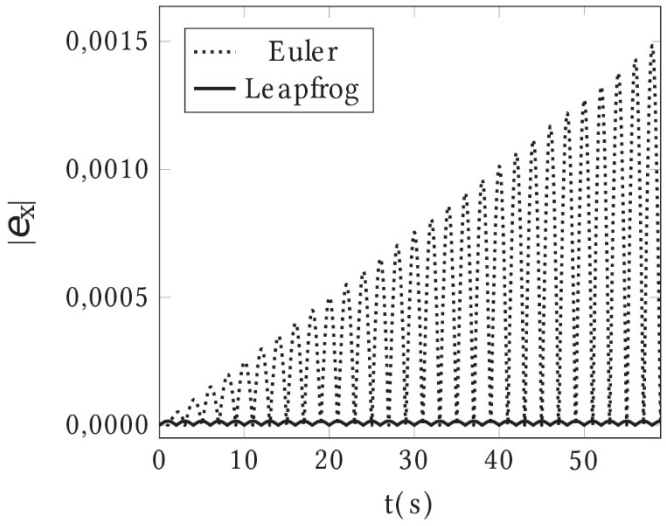


FIGURE 5 : Erreurs absolues calculées sur la position de la masse à $\Delta t = 0{,}001$ s, pour les deux algorithmes considérés.

Solution : Vu la régularité de la figure, on peut penser que le phénomène simuler aboutit à un régime périodique, très probablement sinusoïdal. Pour l'algorithme de Leapfrog, il semble qu'après chaque période, le système se retrouve bien dans son état initial, et qu'en tous cas l'erreur sur une période mesurée sur les différentes périodes est de moyenne nulle : l'erreur reste contenue. Pour l'algorithme d'EULER, sur une oscillation l'énergie calculée augmente systématiquement : à chaque passage au niveau de la position d'équilibre, la vitesse est un peu plus grande qu'attendue. Ces erreurs systématiques s'additionnent période après période, avec une condition initiale de plus en plus dégradée au départ de la période, la simulation est de plus en plus mauvaise.

Q43 —Conclure sur les avantages et les inconvénients de chacun de ces algorithmes et sur leur adéquation pour le traitement numérique de ce problème dans le cas où on envisage une simulation de plusieurs eures.

Solution : L'algorithme d'EULER a pour lui sa simplicité qui le rend très lisible, très simple à implémenter car la démarche transparait dans chacune des relations de l'implémentation.

L'algorithme d'EULER est donc particulièrement pour une approche didactique de l'intégration numérique.

Les algorithmes plus évolués comme celui de Leapfrog peuvent être beaucoup plus précis, au prix de quelques calculs préliminaires et d'une ligne de calcul un peu moins lisible pour la mise en place de la relation de récurrence.

Pour un travail de simulation demandant de la précision et limité par les ressources disponibles, les algorithmes plus évolués sont à préférer.

ANNEXE : Aide-mémoire sur numpy

Les bibliothèques sont importées de la façon suivante :

```
>>> from math import *
>>> import numpy as np
```

La création d'un tableau numpy tab à une dimension possédant n éléments, tous initialisés à 0, est réalisée à l'aide de l'instruction :

```
>>> n=5
>>> tab =np.zeros ( n )
```

Celle d'un tableau numpy tab à une dimension possédant n éléments, uniformément répartis entre deux valeurs debut et fin, se fait avec :

```
>>> debut = 0; fin = 10; n = 5
>>> tab=np.linspace(debut,fin,n)
>>> print(tab)
[ 0.   2.5  5.   7.5 10. ]
```

L'accès à un élément du tableau tab (en lecture ou en écriture) se fait par tab[i], la numérotation des indices se faisant à partir de 0 :

```
>>> tab = np.zeros ( 4 )
>>> tab
array([0., 0., 0., 0.])
>>> tab[1]=2; tab[2]=6
>>> print(tab)
[0. 2. 6. 0.]
```

La sélection de l'ensemble des j premiers éléments du tableau tab est possible avec :

```
>>> j=3; print(tab[:j])
[0. 2. 6.]
```

Le maximum des éléments d'un tableau tab s'obtient avec :

```
>>> np.max ( tab )
6.0
```

FIN