De nombreux systèmes oscillants peuvent, en première approximation être modélisés par un oscillateur harmonique. L'équation différentielle associée à un oscillateur harmonique est de la forme :

$$\ddot{x} + \omega_0^2 x(t) = 0$$

L'avantage de cette modélisation est qu'elle conduit à une équation linéaire dont on connait la solution pour des conditions initiales données.

Dans la suite du TP on considère un oscillateur harmonique de fréquence d'oscillation $f = 1\,\mathrm{Hz}$ défini pour $t \in [0, t_{max} = 10\,\mathrm{s}]$ et évoluant à partir des conditions initiales $x(t=0) = x_0 = 1\,\mathrm{ua}$ et $\dot{x}(t=0) = 0$

Étude analytique

1. Rappeler la solution de cette équation différentielle pour les conditions initiales précisées ci-dessus.
2. Pour un oscillateur harmonique de fréquence f, définir une fonction **oscillateur** en langage Python, qui renvoie la valeur $x(t)$ pour une date `t` donnée en entrée.

Résolution approchée de l'équation différentielle par méthode d'Euler

La méthode d'Euler est un algorithme permettant de résoudre un problème de Cauchy du premier ordre, c'est à dire, une équation différentielle d'ordre 1, de la forme $y'(t) = F(y(t), t)$, avec une condition initiale $y(t=0) = y_0$

3. Rappeler l'implémentation en Python de l'algorithme d'Euler vu en cours.
4. Montrer que le problème défini par :
$$\begin{cases} \ddot{x} + \omega_0^2 x = 0 \\ x(t=0) = x_0 \\ \dot{x}(t=0) = 0 \end{cases}$$
peut se mettre sous la forme d'un problème de Cauchy du premier ordre.
5. Définir une fonction **pbnum** qui traduit ce problème de Cauchy du premier ordre.
6. À l'aide de la fonction **euler**, calculer 100 valeurs de $x(t)$ régulièrement espacées dans un intervalle de $[0, t_{max}]$ puis proposer une procédure permettant de récupérer ces données dans une variable `x` de type `'list'`.
7. Représenter graphiquement l'évolution de `x` en fonction du temps et comparer à la solution analytique.
8. Comment faire pour améliorer la qualité de la résolution numérique ? Quel est alors l'inconvénient ? (justifier en précisant la complexité de cet algorithme)

Le module `time` contient des fonctionnalités qui permettent de mesurer une durée de calcul. Une structure syntaxique simple peut être la suivante :

```python
from time import perf_counter as clock
ti=clock()
#Des calculs
tf=clock()
Dt=tf-ti
```

La variable `Dt` correspond à la durée (en secondes) mise pour effectuer les calculs entre `ti` et `tf`.

9. Proposer une procédure permettant de déterminer le temps mis par l'ordinateur pour résoudre numériquement l'équation différentielle avec l'algorithme d'Euler pour différentes valeurs du pas de calcul. Pour chaque valeur du pas testée, stoker le nombre de valeurs calculées (par l'algorithme d'Euler) dans une liste `nombre` et la durée de calcul dans une liste `durees`.
10. Tracer l'évolution de la durée de calcul en fonction du nombre de valeurs calculées par l'algorithme d'Euler et commenter l'allure de la courbe obtenue.
11. Reprendre la même démarche avec la fonction odeint et discuter de sa performance par rapport à la fonction **euler** définie en début de TP.