Capacité numérique :

	Capacité numérique : à l'aide d'un langage de programmation, résoudre numériquement une équation différentielle du deuxième ordre non-linéaire et faire apparaître l'effet des termes non-linéaires.

```
import matplotlib.pyplot as pl
import numpy as np
from scipy.integrate import odeint
```

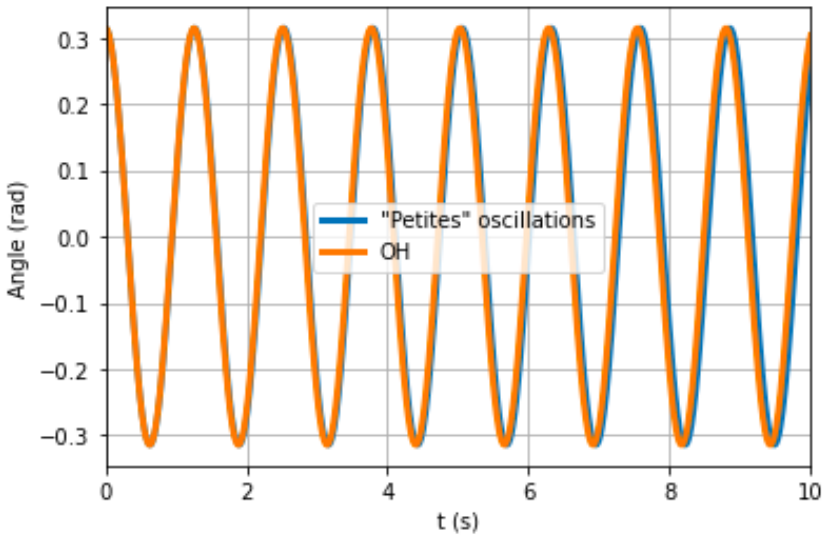
2 Limites

2.2 Non-isochronisme des "grandes" oscillations

Capacité numérique : Utiliser le Jupiter Notebook pour étudier le non-isochronisme des "grandes" oscillations.

```
#on définit la fonction pour résoudre l'équation générale du pendule simple
def pendule(Y,t):
    return np.array([Y[1],-omega**2*np.sin(Y[0])])

#on définit  la fonction pour résoudre l'équation de l'OH de même pulsation propre
def OH(Y,t):
    return ([Y[1],-omega**2*Y[0]])

a,b=0,10 #a instant initial, b instant final
N=1000 #nombre de points utilisés
omega=5 #pulsation propre en rad/s
#on définit l'intervalle de temps sur lequel on résout l'équation.
T=np.linspace(a,b,N)
```

```
#Dans la limite des "petites" oscillations:
Y0=[np.pi/10,0]

#Résolution par la fonction odeint
Thetap=odeint(pendule,Y0,T)
ThetaOH=odeint(OH,Y0,T)

#puis affichage des courbes
pl.figure()
pl.plot(T,Thetap[:,0], linewidth=3,label='"Petites" oscillations')
pl.plot(T,ThetaOH[:,0], linewidth=3,label='OH')
pl.grid()
pl.legend()
#axe des abscisses
pl.xlim(a,b), pl.xlabel("t (s)")
#axe des ordonnées
pl.ylabel("Angle (rad)")
```



```
#on définit la fonction pour résoudre l'équation générale du pendule simple
def pendule(Y,t):
    return np.array([Y[1],-omega**2*np.sin(Y[0])])

#on définit  la fonction pour résoudre l'équation de l'OH de même pulsation propre
def OH(Y,t):
    return ([Y[1],-omega**2*Y[0]])

a,b=0,10 #a instant initial, b instant final
N=1000 #nombre de points utilisés
omega=5 #pulsation propre en rad/s
#on définit l'intervalle de temps sur lequel on résout l'équation.
T=np.linspace(a,b,N)
```

```
#Dans la limite des "grandes" oscillations:
Y1=[np.pi/2,0]

#Résolution par la fonction odeint
Thetap1=odeint(pendule,Y1,T)
ThetaOH1=odeint(OH,Y1,T)

#puis affichage des courbes
pl.figure()
pl.plot(T,Thetap1[:,0], linewidth=3,label='"Grandes" oscillations')
pl.plot(T,ThetaOH1[:,0], linewidth=3,label='OH')
pl.grid()
pl.legend()
#axe des abscisses
pl.xlim(a,b), pl.xlabel("t (s)")
#axe des ordonnées
pl.ylabel("Angle (rad)")

pl.show()
```