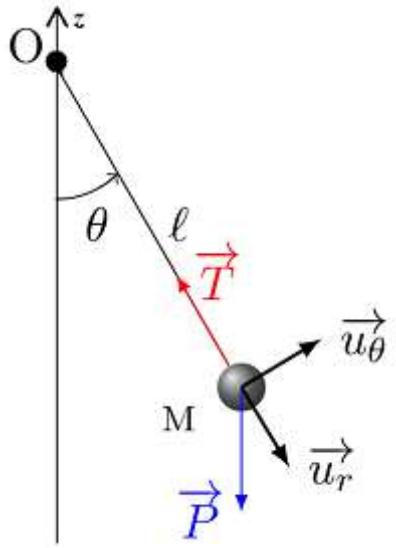
Pendule simple : E.D. non linéaire d'ordre 2 ; calcul d'intégrale.

1) Description et mise en équation



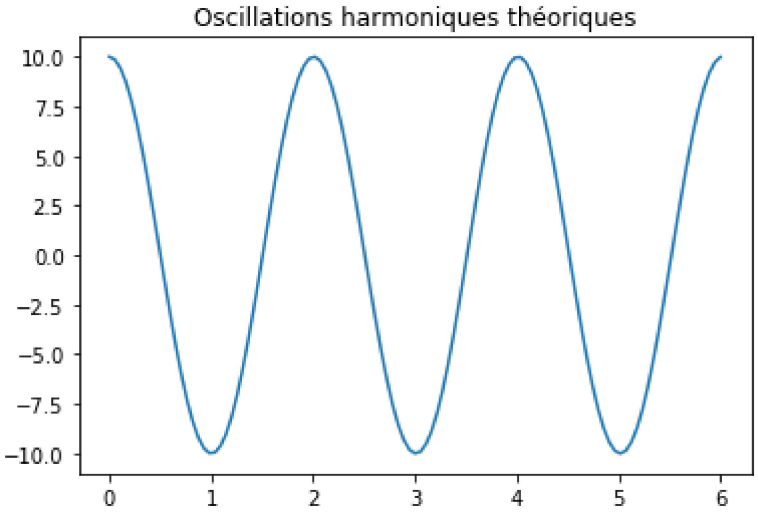
OM est une tige rigide de masse négligeable devant m.
En référentiel terrestre galiléen, on obtient par la R.F.D. dans la base polaire l'équation différentielle :
$\ddot{\theta} + \frac{g}{L}\sin\theta = 0$.
C'est une E.D. du second ordre, *non linéaire* pour une amplitude quelconque.

2) Oscillations de faible amplitude

Dans la mesure où l'on peut faire l'approximation $\sin\theta = \theta$, le pendule effectue des oscillations harmoniques de période propre $T_0 = \frac{2\pi}{\omega_0} = 2\pi\sqrt{\frac{L}{g}}$.
Avec les conditions initiales $\theta(t=0) = \theta_0$ et $\dot{\theta}(t=0) = 0$, on a : $\theta(t) = \theta_0\ \cos(\omega_0\, t)$.
Rq : avec $L = 1$ m et $g \approx 9,8$ m/s², on a $T_0 \approx 2$ s, soit un pendule qui "bat la seconde".

Entrée [177]:

```
import matplotlib.pyplot as plt
import numpy as np
L = 1 ; g = 9.8 ;
omega0 = np.sqrt(g/L) ; T0 = 2*np.pi/omega0
theta0 = 10 # valeur en degrés
dates = np.linspace(0,6,111)
angles1 = theta0*np.cos(omega0*dates)
plt.close()
plt.figure()
plt.plot(dates, angles1)
plt.title('Oscillations harmoniques théoriques')
plt.show()
```

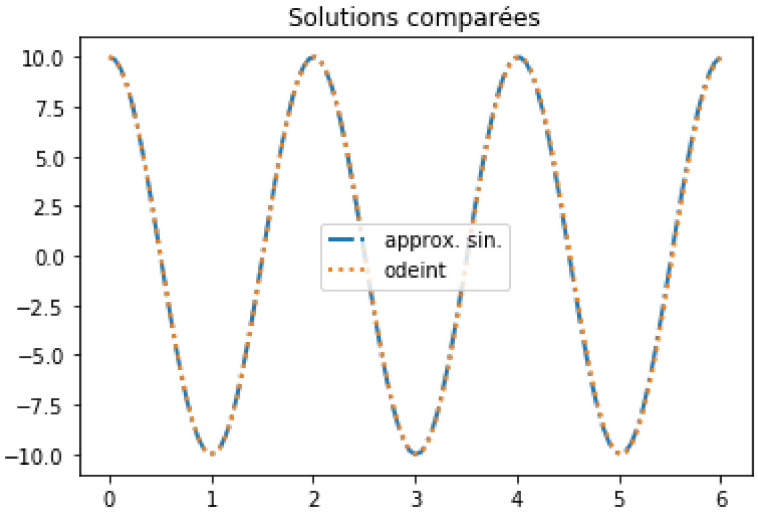


3) Résolution numérique par "odeint"

Pour résoudre une E.D. du 1^{er} ordre, la syntaxe est $scipy.integrate.odeint(func, y0, t)$, où 'func' désigne la fonction calculant y'(t), 'y0' la valeur initiale, et 't' la liste des dates t_k.
On reçoit en sortie un tableau contenant les $y(t_k)$: $[y_0, y(t_1), y(t_2), \ldots]$.
Pour adapter odeint à une E.D. du 2^e ordre de la forme $\ddot{\theta} = f(\theta)$, on *vectorise* celle-ci :
$Y = \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} \Rightarrow \dot{Y} = \begin{pmatrix} \dot{\theta} \\ f(\theta) \end{pmatrix}$, avec ici $f(\theta) = -\omega_0^2 \ \sin\theta$.
La condition initiale 'y0' de odeint sera alors la liste $[\theta_0, \omega_0]$.

Entrée [178]:

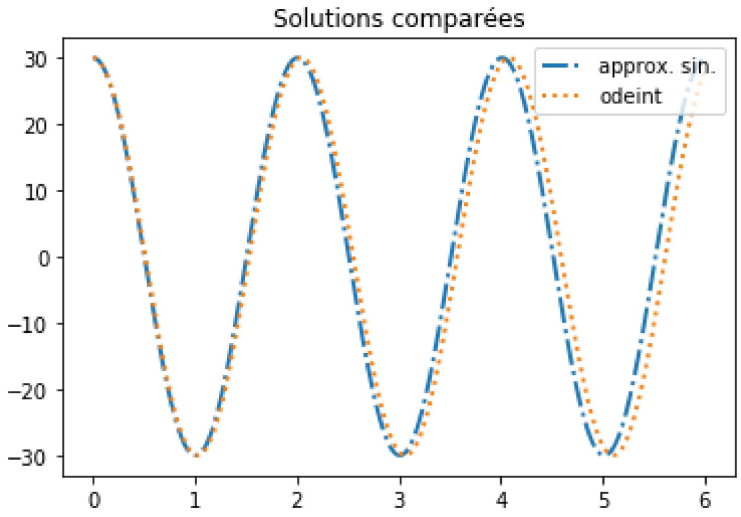
```
import scipy.integrate as integr
def eqdiff(y,t): # la fonction qui renvoie y'
    return y[1], -omega0**2 * np.sin(y[0])
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
angles2 = [ k[0] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, angles1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, angles2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



3) Influence de l'amplitude

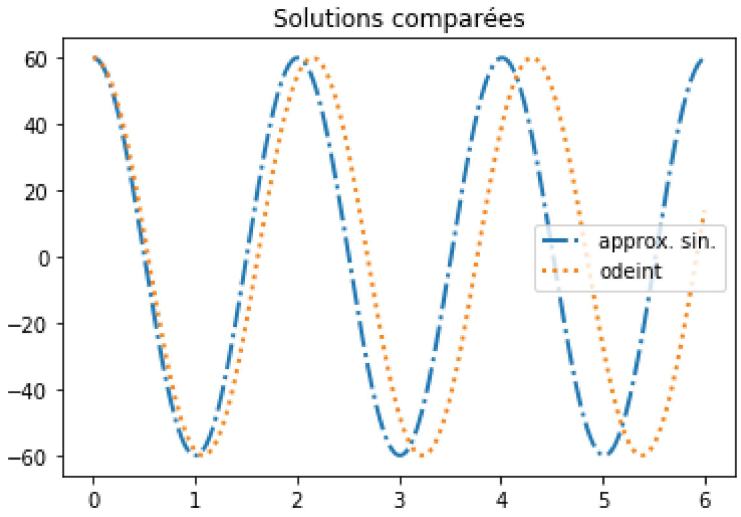
Entrée [32]:

```
theta0 = 30 # valeur en degrés
dates = np.linspace(0,6,1111)
angles1 = theta0 * np.cos(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
angles2 = [ k[0] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, angles1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, angles2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



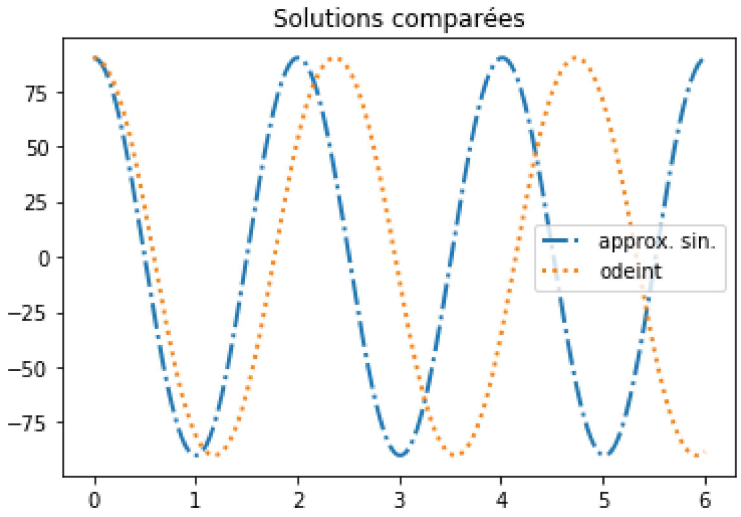
Entrée [33]:

```
theta0 = 60 # valeur en degrés
dates = np.linspace(0,6,1111)
angles1 = theta0 * np.cos(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
angles2 = [ k[0] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, angles1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, angles2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



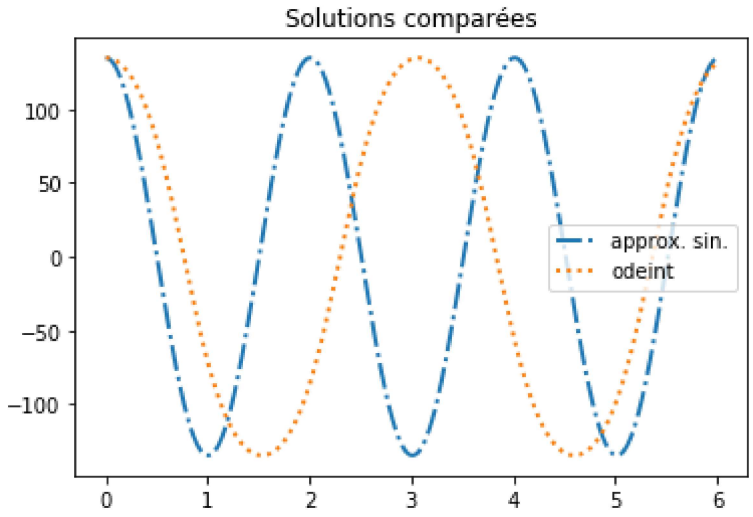
Entrée [35]:

```
theta0 = 90 # valeur en degrés
dates = np.linspace(0,6,1111)
angles1 = theta0 * np.cos(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
angles2 = [ k[0] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, angles1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, angles2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



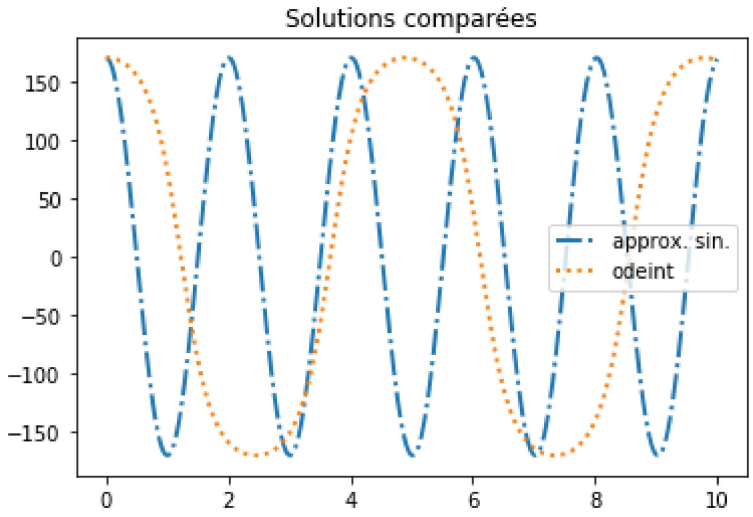
Entrée [36]:

```
theta0 = 135 # valeur en degrés
dates = np.linspace(0,6,1111)
angles1 = theta0 * np.cos(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
angles2 = [ k[0] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, angles1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, angles2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



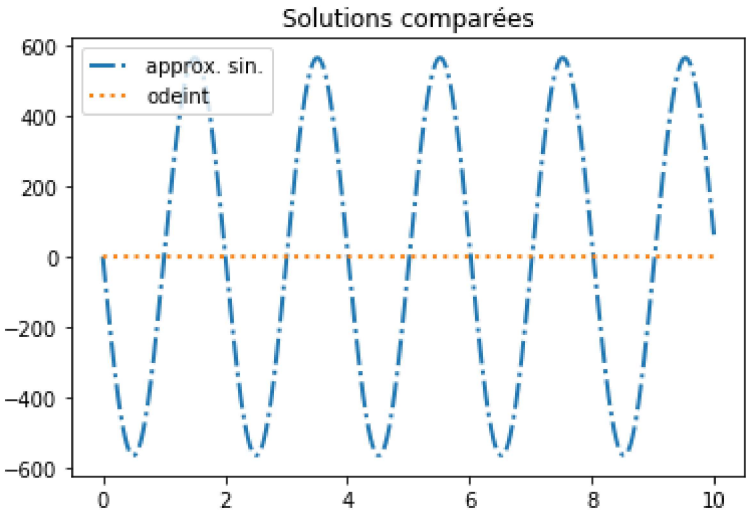
Entrée [38]:

```
theta0 = 170 # valeur en degrés
dates = np.linspace(0,10,1111)
angles1 = theta0 * np.cos(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
angles2 = [ k[0] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, angles1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, angles2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



Entrée [49]:

```
theta0 = 180 # valeur en degrés
dates = np.linspace(0,10,1111)
vitang1 = -theta0 * omega0 * np.sin(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
vitang2 = [ k[1] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, vitang1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, vitang2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```

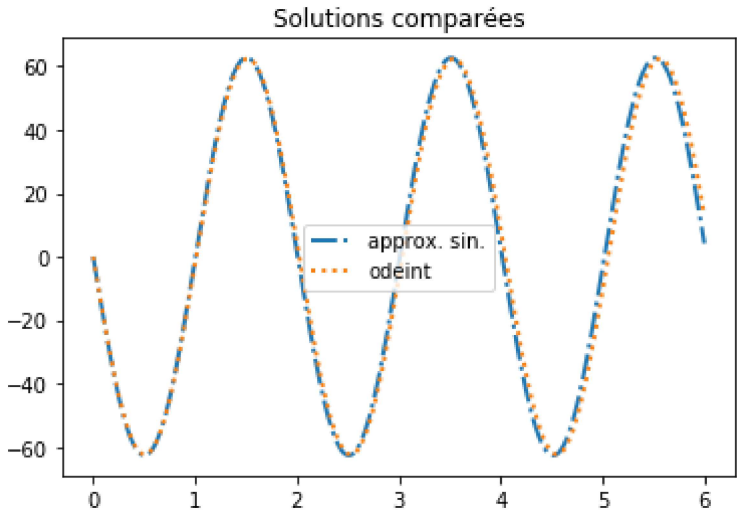


Intéressons nous à la vitesse angulaire $\dot{\theta}$:

- petits angles :

Entrée [44]:

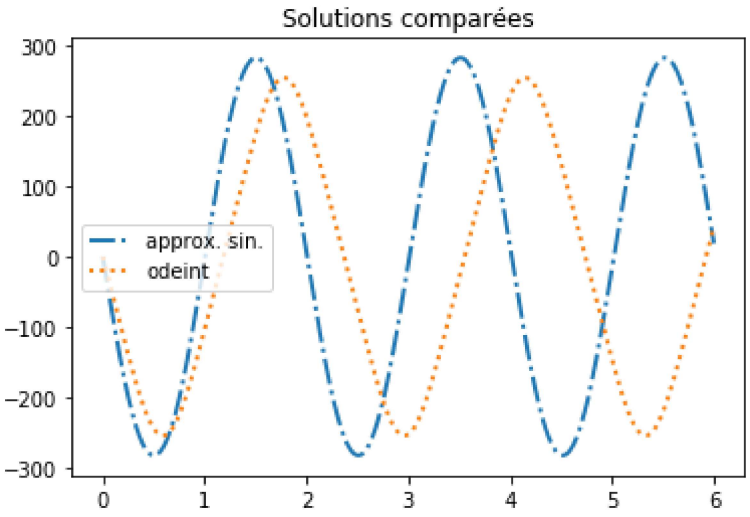
```
theta0 = 20 # valeur en degrés
dates = np.linspace(0,6,1111)
vitang1 = -theta0 * omega0 * np.sin(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
vitang2 = [ k[1] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, vitang1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, vitang2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



- amplitude moyenne :

Entrée [42]:

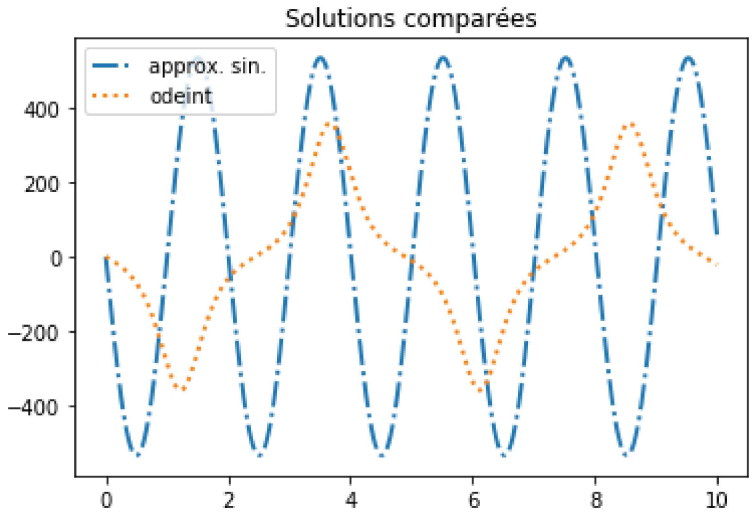
```
theta0 = 90 # valeur en degrés
dates = np.linspace(0,6,1111)
vitang1 = -theta0 * omega0 * np.sin(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
vitang2 = [ k[1] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, vitang1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, vitang2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



- grande amplitude :

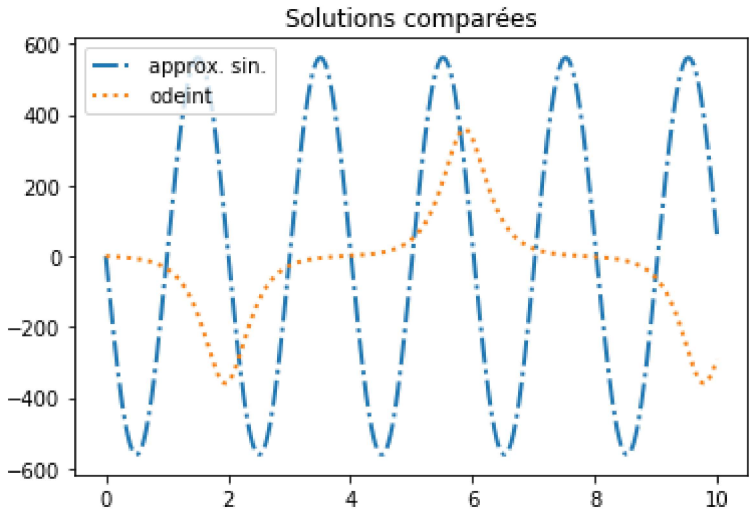
Entrée [45]:

```
theta0 = 160 # valeur en degrés
dates = np.linspace(0,10,1111)
vitang1 = -theta0 * omega0 * np.sin(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
vitang2 = [ k[1] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, vitang1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, vitang2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



Entrée [47]:

```
theta0 = 179 # valeur en degrés
dates = np.linspace(0,10,1111)
vitang1 = -theta0 * omega0 * np.sin(omega0*dates)
thetarad0 = theta0 * np.pi / 180# calcul en radians
sol = integr.odeint(eqdiff, [thetarad0, 0],dates)
#print(sol)
vitang2 = [ k[1] / np.pi * 180 for k in sol]
#print(angles2)
plt.close()
plt.figure()
plt.plot(dates, vitang1, ls='dashdot', lw=2, label='approx. sin.')
plt.plot(dates, vitang2, ':', lw=2, label='odeint')
plt.title('Solutions comparées')
plt.legend()
plt.show()
```



Conclusion : lorsque l'amplitude (donc l'énergie) augmente :

- la période des oscillations augmente ;
- la fonction $\theta(t)$ n'est plus sinusoïdale.
- lorsque θ s'approche de 180° (pendule à la verticale du point de rotation), il tend à s'arrêter au sommet de sa trajectoire, ce qu'il fait lorsque $\theta = 180°$.

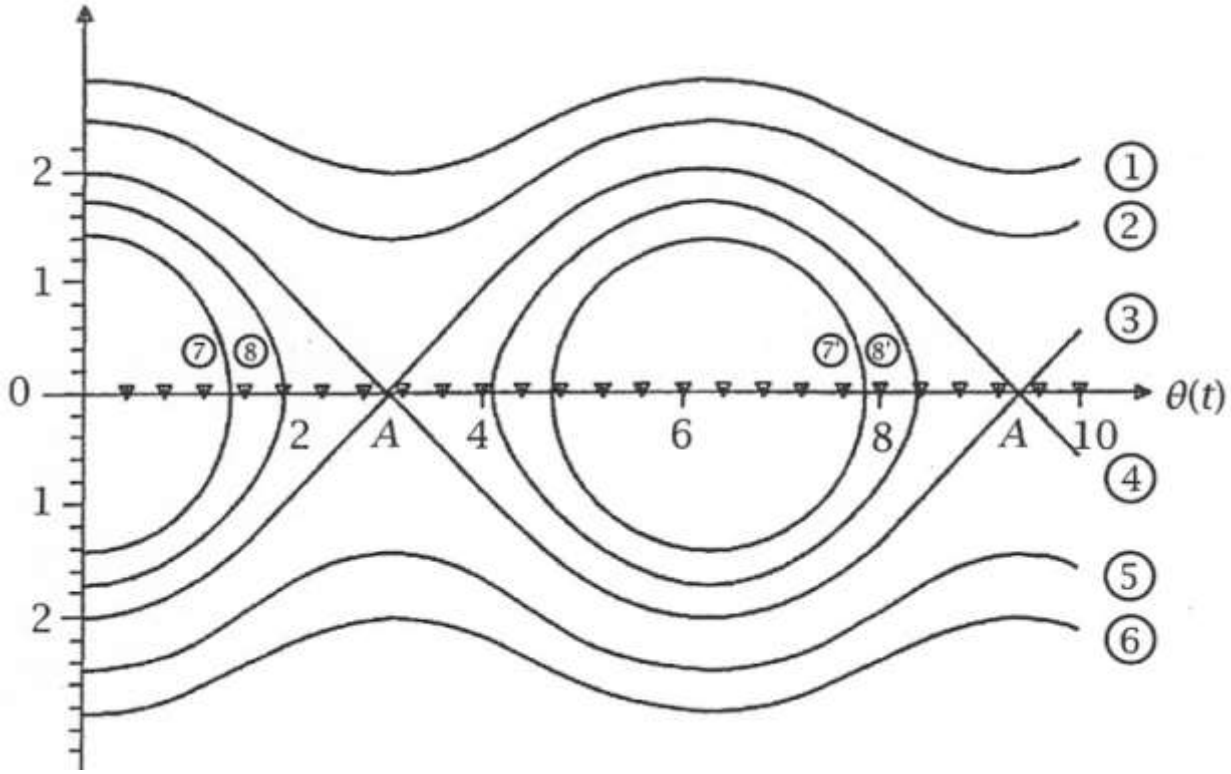
4) Portrait de phase et Bifurcation

Portrait de phase

```
Un **portrait de phase** est une représentation géométrique des trajectoires d'un système
dynamique dans **l'espace des phases**, c'est-à-dire la représentation des évolutions de
$\dot{x}(t)$ en fonction de $x(t)$ (si $x$ est la grandeur d'étude), pour différentes
conditions initiales.\
Pour un oscillateur, on pourra normaliser les grandeurs en représentant $\frac{\dot{x}
(t)}{\dot{x}_{max}}$ en fonction de $\frac{x(t)}{x_{max}}$; la trajectoire de phase
s'inscrira dans un carré de valeurs extrêmes -1 et +1 sur chaque axe.\
Un *mouvement périodique* est naturellement associé à une *trajectoire de phase fermée*.\
**Pour un oscillateur harmonique, le portrait de phase normalisé est circulaire**.\
```

```
En effet : $x(t)=A\,\sin(\omega\,t+\varphi) \Rightarrow \dot{x}
(t)=A\omega\,\cos(\omega\,t+\varphi)$\
donc $\frac{x(t)}{x_{max}}=\sin(\omega\,t+\varphi)$ et $\frac{\dot{x}(t)}
{\dot{x}_{max}}=\cos(\omega\,t+\varphi)$, ce qui correspond à un cercle.\
*A contrario*, un portrait normalisé non circulaire (non elliptique si non normalisé)
révèle des oscillations non harmoniques, c'est-à-dire non décrites par des fonctions
sinusoïdales.\
\
Exemples :
![title](TrajPhase1-red.png)
![title](TrajPhase3-red.png)
![title](TrajPhase2-red.png)
```

Portrait de phase du pendule simple



- (1) et (2) : mouvement révolutif dans le sens de θ croissant ;
- (5) et (6) : mouvement révolutif dans le sens de θ décroissant ;
- (7), (8), (7'), (8') : mouvement périodique (oscillations), pratiquement harmoniques pour (7) et (7'), visiblement non harmoniques pour (8() et (8') ;
- (3) et (4) : trajectoire critique, la vitesse angulaire s'annule pour $\theta = \pi\ [2\pi]$, c'est-à-dire lorsque le pendule est à la verticale, M au-dessus de O : le pendule s'immobilise, mais c'est un équilibre instable et il va retomber... d'un côté ou de l'autre, sans que l'on puisse prévoir lequel !

Dans le cas d'une trajectoire de phase critique, le système "perd la mémoire" au passage par $\theta = \pi\ [2\pi]$, et le portrait de phase présente une **bifurcation** car deux évolutions ultérieures sont possibles.
Ce type de comportement est une voie d'accès aux *phénomènes chaotiques*, dans lesquels le déterminisme habituel de la mécanique classique est battu en brèche.

5) Calcul de la période pour une amplitude quelconque

Expression sous forme d'une intégrale

Reprenons l'équation différentielle $\ddot{\theta} + \frac{g}{L}\sin\theta = 0$. Il vient en multipliant par $\dot{\theta}$: $\dot{\theta}\,\ddot{\theta} + \frac{g}{L}\sin\theta\,\dot{\theta} = 0$.
Ceci s'intègre une fois par rapport au temps pour donner : $\frac{1}{2}\dot{\theta}^2 - \frac{g}{L}\cos\theta = C^{te} = -\frac{g}{L}\cos\theta_0$ avec les C.I. choisies.
Par *séparation des variables* puis intégration sur 1/4 d'oscillation, de $t = 0$ à $t = \frac{T_0}{4}$ (et donc $\dot{\theta} < 0$) :

$$\int_{\theta_0}^{0} \frac{-\mathrm{d}\theta}{\sqrt{2(\cos\theta - \cos\theta_0)}} = \int_0^{T/4} \sqrt{\frac{g}{L}}\mathrm{d}t = \int_0^{T/4} \omega_0\,\mathrm{d}t = \omega_0\frac{T}{4} = \omega_0 T_0 \frac{T}{4T_0} = \frac{\pi}{2}\frac{T}{T_0}$$

et par conséquent :

$$T = T_0 \times \frac{\sqrt{2}}{\pi}\int_0^{\theta_0} \frac{\mathrm{d}\theta}{\sqrt{\cos\theta - \cos\theta_0}}.$$

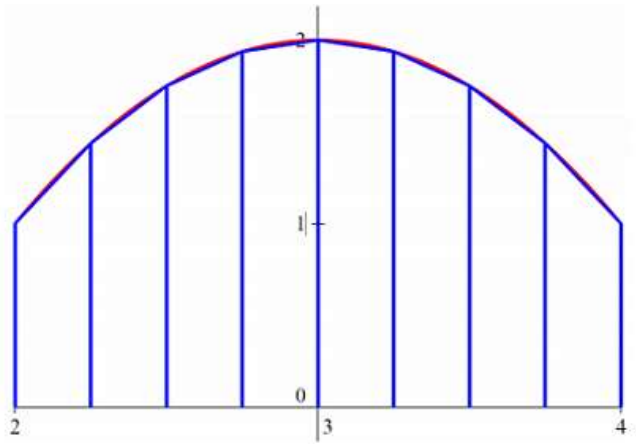
Malheureusement cette intégrale (dite *intégrale elliptique*) ne peut pas s'exprimer à l'aide de fonctions simples !
Dans le cas d'angles "pas trop grands", on obtient par changement de variable et développements limités une expression approchée dite **formule de Borda** (1733-1799) :

$$T(\theta) \approx T_0\left(1 + \frac{\theta_0^2}{16}\right).$$

Cette relation donne une erreur relative inférieure à 1% pour des amplitudes inférieures à environ 70° (et 3% pour 90°).
Pour des amplitudes quelconques, il ne reste qu'à calculer numériquement.

Calcul numérique approché de l'intégrale par découpage

Utilisons la *méthode des trapèzes*, dérivée de la méthode des rectangles mais plus précise :



$$\int_a^b f(x)dx \approx T_n = \sum_{k=0}^{n-1} h \cdot \frac{f(x_k) + f(x_{k+1})}{2};$$

on limite les calculs en réécrivant la somme :

$$T_n = h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{k=1}^{n-1} f(x_k)\right).$$

Entrée [133]:

```
# fonction calculant Tn pour une fonction quelconque 'func' :
def trap(func, a, b, n):
    h = ( b - a ) / n # largeur des tranches
    x = a # valeur initiale de l'abscisse
    s = ( func(a) + func(b) ) / 2 # variable de travail
    for k in range(n-1):
        x = x + h
        s = s + func(x)
    return s * h
```

Calculons maintenant $X = \frac{T}{T_0}$ pour le pendule d'amplitude θ_0 :

Entrée [165]:

```
def per(ampl): # calcul de X pour une amplitude donnée en degrés
    amplrad = ampl / 180 * np.pi
    thetamax = ( 1 - 1e-6 ) * amplrad # pour éviter une division par 0
    def f(theta): # l'intégrande
        return 1 / np.sqrt( np.cos(theta) - np.cos(amplrad) )
    return round(np.sqrt(2) / np.pi * trap(f, 0, thetamax, 100000),2)
print(per(20)) ; print(per(90)) ; print(per(179))
```

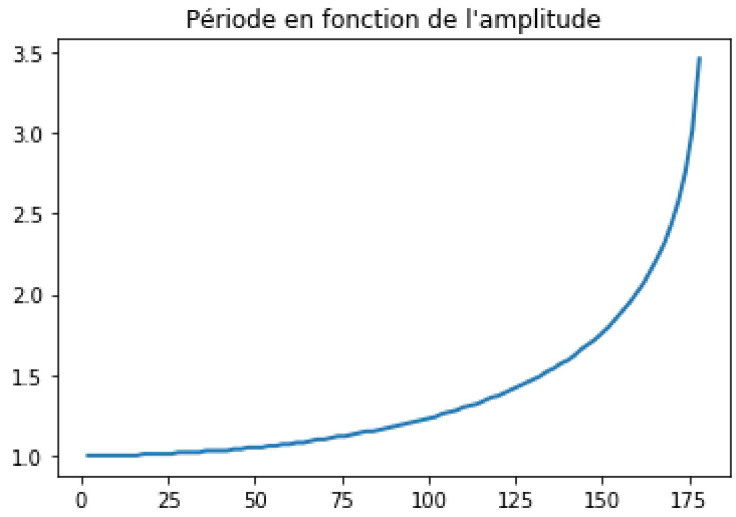
```
1.01
1.18
3.9
```

Rq : on n'augmente pratiquement plus la précision du résultat au-delà de 100 000 tranches.

Augmentation de la période avec l'amplitude

Entrée [159]:

```
listA = [ 2*i for i in range(1,90) ]
listT = [ per(i) for i in listA ]
plt.close()
plt.figure()
plt.plot(listA, listT, lw=2)
plt.title("Période en fonction de l'amplitude")
plt.show()
```



Calcul approché de l'intégrale par une méthode Python

Entrée [175]:

```
import scipy.integrate as integr
def per2(ampl): # calcul de X pour une amplitude donnée en degrés
    amplrad = ampl / 180 * np.pi
    thetamax = ( 1 - 1e-6 ) * amplrad # pour éviter une division par 0
    def f(theta): # l'intégrande
        return 1 / np.sqrt( np.cos(theta) - np.cos(amplrad) )
    return round(np.sqrt(2) / np.pi * integr.quad(f, 0, thetamax)[0],2)
print(per2(20)) ; print(per2(90)) ; print(per2(179))
```

```
1.01
1.18
3.89
```

Les résultats sont obtenus beaucoup plus rapidement !

NB : quad renvoie un doublet (valeur, erreur), d'où la commande $quad(\ldots)[0]$ pour récupérer la valeur.

Entrée [176]:

```
listA = [ 2*i for i in range(1,90) ]
listT = [ per2(i) for i in listA ]
plt.close()
plt.figure()
plt.plot(listA, listT, lw=2)
plt.title("Période en fonction de l'amplitude")
plt.show()
```

```
1.01
```