Pendule double

Cette page doit être lue en version MathML.

1. Introduction

Ce document est une introduction à la simulation des systèmes de solides indéformables en liaison. L'exemple traité est un pendule double.

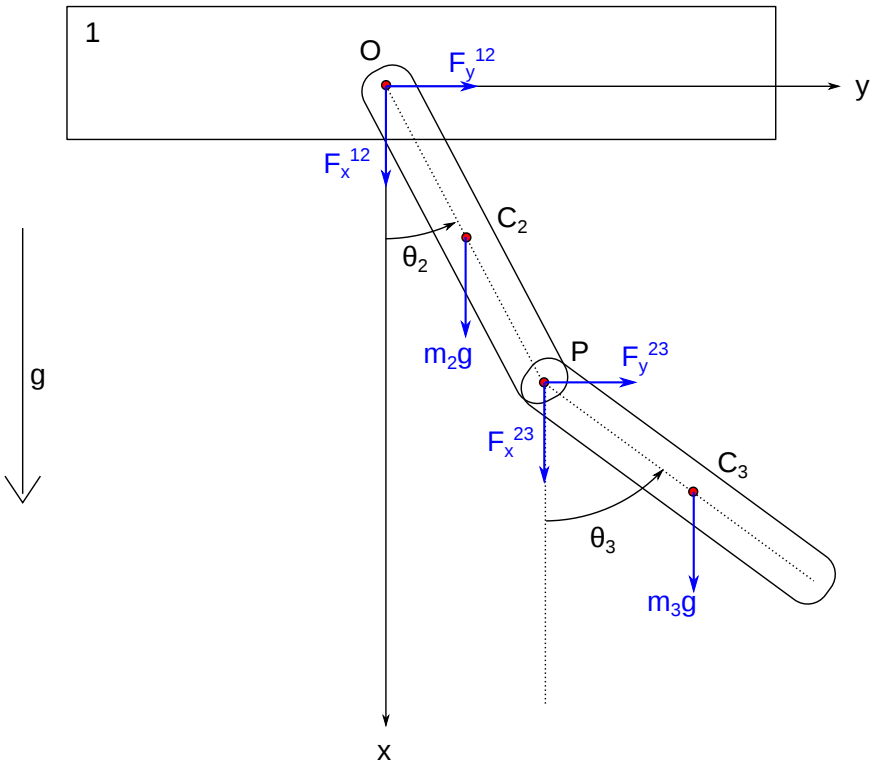
La première méthode de mise en équation consiste à utiliser pour chaque solide tous ses degrés de liberté (3 pour un mouvement plan) et à introduire les forces de liaison. Cette méthode conduit à un nombre d'inconnues beaucoup plus grand que le nombre de degrés de liberté du système. Elle a cependant l'avantage de se prêter à un traitement automatique pour N solides. L'intégration numérique nécessite la résolution d'un système linéaire à chaque pas de temps.

La seconde méthode consiste à utiliser les équations de Lagrange. Les inconnues sont les degrés de liberté du système (2 pour le pendule double). Les équations obtenues sont plus faciles à intégrer, mais la méthode nécessite beaucoup de calculs manuels, et ne se prête pas à un traitement automatique pour N solides.

2. Équations de Newton et d'Euler

2.a. Équations du mouvement

On considère un système en mouvement plan constitué de deux barres. La première barre (notée 2) est en liaison pivot avec le support (noté 1). La seconde barre (notée 3) est en liaison pivot avec la première. Pour simplifier, on assimile les liaisons pivot à des liaisons ponctuelles, ce qui revient à supposer qu'elles sont parfaites. Le support est fixe dans le référentiel galiléen. On note C_2 et C_3 les centres de masse des deux barres. Les longueurs des barres sont notées l_1 et l_2 et on suppose que les centres de masse sont au milieu. On introduit les composantes des forces agissant sur les points de liaison. Par exemple F_x^{12} est la composante sur x de la force exercée par le support 1 sur la barre 2.



Le système a deux degrés de liberté, représentés par les angles θ_2 et θ_3. On commence par exprimer l'accélération des centres de masse en fonction des angles et de leurs dérivées :

$$a_{2x} = \frac{l_2}{2}\left(-\ddot{\theta}_2 \sin\theta_2 - \dot{\theta}_2^2 \cos\theta_2\right) \tag{1}$$

$$a_{2y} = \frac{l_2}{2}\left(\ddot{\theta}_2 \cos\theta_2 - \dot{\theta}_2^2 \sin\theta_2\right) \tag{2}$$

$$a_{3x} = l_2\left(-\ddot{\theta}_2 \sin\theta_2 - \dot{\theta}_2^2 \cos\theta_2\right) + \frac{l_3}{2}\left(-\ddot{\theta}_3 \sin\theta_3 - \dot{\theta}_3^2 \cos\theta_3\right) \tag{3}$$

$$a_{3y} = l_2\left(\ddot{\theta}_2 \cos\theta_2 - \dot{\theta}_2^2 \sin\theta_2\right) + \frac{l_3}{2}\left(\ddot{\theta}_3 \cos\theta_3 - \dot{\theta}_3^2 \sin\theta_3\right) \tag{4}$$

Pour la barre 2, on écrit le théorème de la résultante cinétique et le théorème du moment cinétique au centre de masse (appelés aussi équation de Newton et d'Euler) :

$$m_2 a_{2x} = F_x^{12} - F_x^{23} + m_2 g \tag{5}$$

$$m_2 a_{2y} = F_y^{12} - F_y^{23} \tag{6}$$

$$J_2\ddot{\theta}_2 = \frac{l_2}{2}\left(F_x^{12}\sin\theta_2 - F_y^{12}\cos\theta_2 + F_x^{23}\sin\theta_2 - F_y^{23}\cos\theta_2\right) \tag{7}$$

On procède de même pour la barre 3 :

$$m_3 a_{3x} = F_x^{23} + m_3 g \tag{8}$$
$$m_3 a_{3y} = F_y^{23} \tag{9}$$
$$J_3 \ddot{\theta}_3 = \frac{l_3}{2}\left(F_x^{23}\sin\theta_3 - F_y^{23}\cos\theta_3\right) \tag{10}$$

On obtient ainsi 10 équations pour les 10 inconnues suivantes :

$$X = (a_{2x}, a_{2y}, \ddot{\theta}_2, a_{3x}, a_{3y}, \ddot{\theta}_3, F_x^{12}, F_y^{12}, F_x^{23}, F_y^{23})^T \tag{11}$$

Le système d'équations est linéaire. On peut donc le mettre sous la forme matricielle $AX = B$ avec :

$$A = \begin{pmatrix}
1 & 0 & \frac{l_2}{2}\sin\theta_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -\frac{l_2}{2}\cos\theta_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & l_2\sin\theta_2 & 1 & 0 & \frac{l_3}{2}\sin\theta_3 & 0 & 0 & 0 & 0 \\
0 & 0 & -l_2\cos\theta_2 & 0 & 1 & -\frac{l_3}{2}\cos\theta_3 & 0 & 0 & 0 & 0 \\
m_2 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
0 & m_2 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\
0 & 0 & J_2 & 0 & 0 & 0 & -\frac{l_2}{2}\sin\theta_2 & \frac{l_2}{2}\cos\theta_2 & -\frac{l_2}{2}\sin\theta_2 & \frac{l_2}{2}\cos\theta_2 \\
0 & 0 & 0 & m_3 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & m_3 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & J_3 & 0 & 0 & -\frac{l_3}{2}\sin\theta_3 & \frac{l_3}{2}\cos\theta_3
\end{pmatrix} \tag{12}$$

$$B = \begin{pmatrix}
-\frac{l_2}{2}\dot{\theta}_2^2\cos\theta_2 \\
-\frac{l_2}{2}\dot{\theta}_2^2\sin\theta_2 \\
-l_2\dot{\theta}_2^2\cos\theta_2 - \frac{l_3}{2}\dot{\theta}_3^2\cos\theta_3 \\
-l_2\dot{\theta}_2^2\sin\theta_2 - \frac{l_3}{2}\dot{\theta}_3^2\sin\theta_3 \\
m_2 g \\
0 \\
0 \\
m_3 g \\
0 \\
0
\end{pmatrix} \tag{13}$$

Il est en principe possible de résoudre ce système formellement pour exprimer les accélérations angulaires en fonction des angles. Une approche plus simple et plus générale pour les problèmes de systèmes de solides ([1]) consiste à laisser le système d'équations sous cette forme. Il comporte 8 inconnues de plus que les deux degrés de liberté du système : les accélérations des centres de masse et les forces de liaison. Les équations sont intégrées par une méthode numérique à un pas (méthode d'Euler ou Runge-Kutta). À chaque pas de temps, le système $AX = B$ est résolu numériquement pour obtenir les accélérations angulaires.

2.b. Intégration numérique

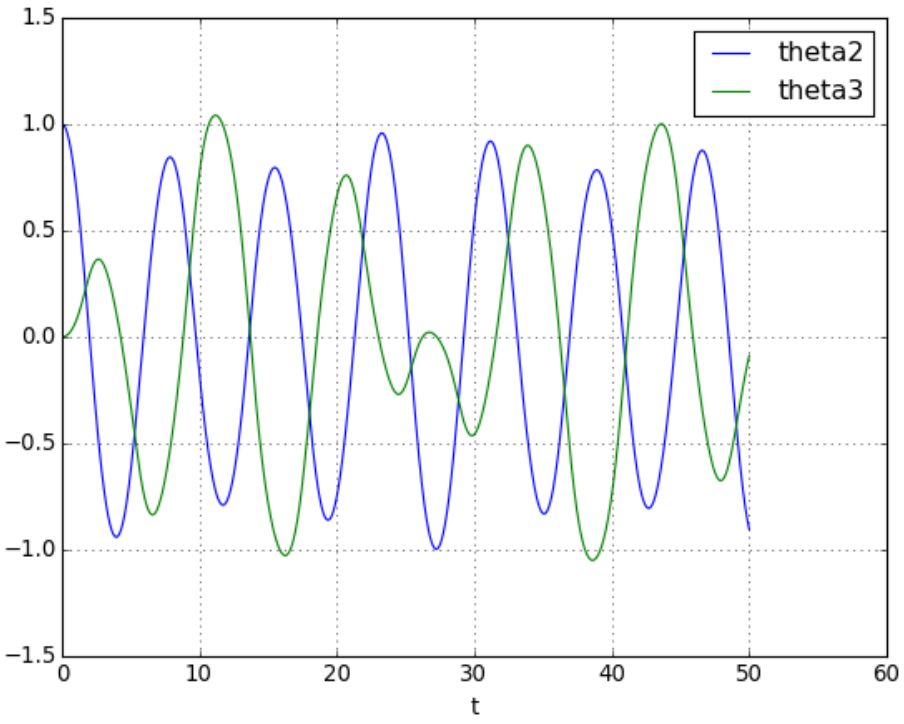
L'évaluation des accélérations angulaires est très coûteuse puisqu'elle nécessite la résolution d'un système linéaire à 10 inconnues. Une methode à un pas et à une seule

évaluation de la dérivée par pas, comme la méthode d'Euler, est donc recommandée. Le système étant conservatif, on a intérêt à utiliser une méthode d'Euler symplectique. À chaque pas de temps, on calcule les nouveaux angles avant de calculer les accélérations angulaires par résolution du système linéaire, puis on calcule les nouvelles vitesses angulaires.

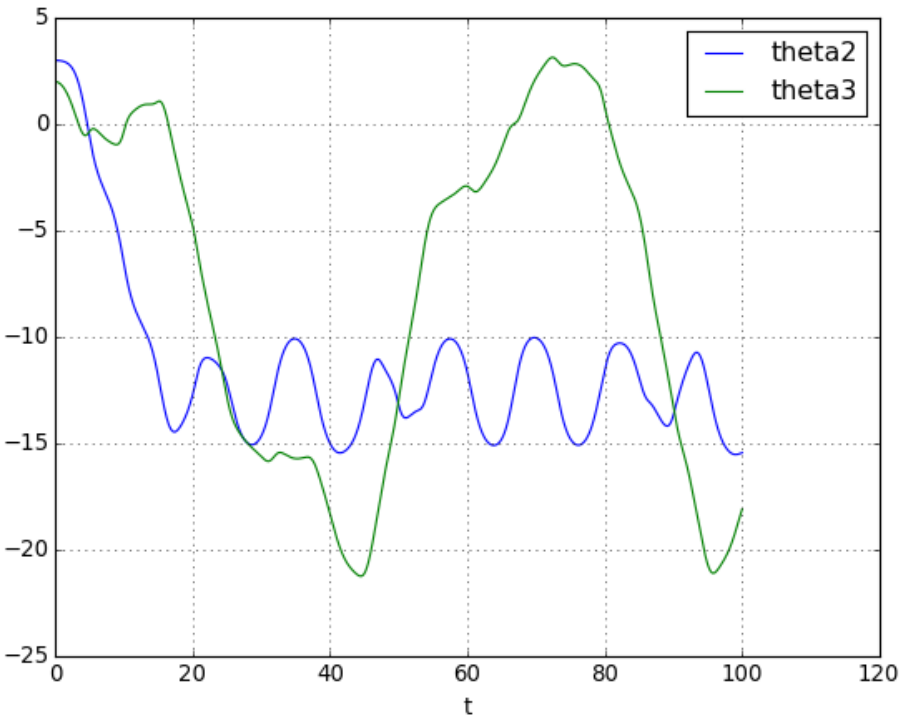
```
from scipy.linalg import solve
import math
import numpy
from matplotlib.pyplot import *

class PenduleDouble:
    def __init__(self,l2,l3,m2,m3,J2,J3):
        self.A = numpy.zeros((10,10))
        self.B = numpy.zeros((10,1))
        self.A[0,0] = 1.0
        self.A[1,1] = 1.0
        self.A[2,3] = 1.0
        self.A[3,4]= 1.0
        self.A[4,0] = m2
        self.A[4,6] = -1.0
        self.A[4,8] = 1.0
        self.A[5,1] = m2
        self.A[5,7] = -1.0
        self.A[5,9] = 1.0
        self.A[6,2] = J2
        self.A[7,3] = m3
        self.A[7,8] = -1.0
        self.A[8,4] = m3
        self.A[8,9] = -1.0
        self.A[9,5] = J3
        self.B[4] = m2
        self.B[7] = m3
        self.l2 = l2
        self.l3 = l3
        self.theta2 = 0.0
        self.theta3 = 0.0
        self.dtheta2 = 0.0
        self.dtheta3 = 0.0

    def pas_euler(self,h):
        self.theta2 += self.dtheta2*h
        self.theta3 += self.dtheta3*h
        sin2 = math.sin(self.theta2)
        cos2 = math.cos(self.theta2)
        sin3 = math.sin(self.theta3)
        cos3 = math.cos(self.theta3)
        l2 = self.l2*0.5
```

```
        l3 = self.l3*0.5
        self.A[0,2] = l2*sin2
        self.A[1,2] = -l2*cos2
        self.A[2,2] = self.l2*sin2
        self.A[2,5] = l3*sin3
        self.A[3,2] = -self.l2*cos2
        self.A[3,5] = -l3*cos3
        self.A[6,6] = -l2*sin2
        self.A[6,7] = l2*cos2
        self.A[6,8] = -l2*sin2
        self.A[6,9] = l2*cos2
        self.A[9,8] = -l3*sin3
        self.A[9,9] = l3*cos3
        dtheta2_2 = self.dtheta2**2
        dtheta3_2 = self.dtheta3**2
        self.B[0] = -l2*dtheta2_2*cos2
        self.B[1] = -l2*dtheta2_2*sin2
        self.B[2] = self.B[0]*2-l3*dtheta3_2*cos3
        self.B[3] = self.B[1]*2-l3*dtheta3_2*sin3
        x = solve(self.A,self.B)
        self.dtheta2 += x[2][0]*h
        self.dtheta3 += x[5][0]*h


    def euler(self,T,Yi,h):
        Y = Yi
        t = 0.0
        liste_y = [Yi]
        liste_t = [t]
        self.theta2 = Yi[0]
        self.dtheta2 = Yi[1]
        self.theta3 = Yi[2]
        self.dtheta3 = Yi[3]
        while t < T:
            self.pas_euler(h)
            t += h
            liste_y.append([self.theta2,self.dtheta2,self.theta3,self.dtheta3])
            liste_t.append(t)
        return (numpy.array(liste_t),numpy.array(liste_y))




pendule = PenduleDouble(1.0,1.0,1.0,1.0,1.0,1.0)
Yi = [1.0,0.0,0.0,0.0]
```

```
(t,y) = pendule.euler(50,Yi,1e-3)

figure()
plot(t,y[:,0],label="theta2")
plot(t,y[:,2],label="theta3")
legend(loc="upper right")
xlabel("t")
grid()
```




```
Yi = [3.0,0.0,2.0,0.0]
(t,y) = pendule.euler(100,Yi,1e-3)
figure()
plot(t,y[:,0],label="theta2")
plot(t,y[:,2],label="theta3")
legend(loc="upper right")
xlabel("t")
grid()
```



3. Équations de Lagrange

3.a. Équations du mouvement

Les équations de Lagrange permettent d'obtenir directement des équations du mouvement ne faisant intervenir que les degrés de liberté, ici les angles θ_2, θ_3. Elle s'applique pour des liaisons dont la puissance totale des forces est nulle, c'est-à-dire pour des liaisons parfaites.

Le lagrangien est la différence entre l'énergie cinétique et l'énergie potentielle :

$$L(\theta_2, \theta_3, \dot{\theta}_2, \dot{\theta}_3) = E_c(\theta_2, \theta_3, \dot{\theta}_2, \dot{\theta}_3) - E_p(\theta_2, \theta_3) \tag{14}$$

Pour ce système à deux degrés de liberté, les deux équations de Lagrange ([2]) sont :

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_2} - \frac{\partial L}{\partial \theta_2} = 0 \tag{15}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_3} - \frac{\partial L}{\partial \theta_3} = 0 \tag{16}$$

Pour exprimer l'énergie cinétique, il faut tout d'abord écrire les vitesses des centres de masse :

$$v_{2x} = -\frac{l_2}{2}\dot{\theta}_2 \sin\theta_2 \tag{17}$$

$$v_{2y} = \frac{l_2}{2}\dot{\theta}_2 \cos\theta_2 \tag{18}$$

$$v_{3x} = -l_2\dot{\theta}_2 \sin\theta_2 - \frac{l_3}{2}\dot{\theta}_3 \sin\theta_3 \tag{19}$$

$$v_{3y} = l_2\dot{\theta}_2 \cos\theta_2 + \frac{l_3}{2}\dot{\theta}_3 \cos\theta_3 \tag{20}$$

Pour chaque solide, l'énergie cinétique est la somme de l'énergie cinétique du centre de masse affecté de la masse du solide et de l'énergie cinétique de rotation autour du centre de masse :

$$E_c = \frac{1}{2}m_2 v_2^2 + \frac{1}{2}m_3 v_3^2 + \frac{1}{2}J_2\dot{\theta}_2^2 + \frac{1}{2}J_3\dot{\theta}_3^2 \tag{21}$$

$$= \frac{1}{2}m_2\left(\frac{l_2}{2}\right)^2\dot{\theta}_2^2 + \frac{1}{2}m_3 l_2^2\dot{\theta}_2^2 + \frac{1}{2}m_3\left(\frac{l_2}{2}\right)^2\dot{\theta}_3^2 + \frac{1}{2}m_3 l_2 l_3\dot{\theta}_2\dot{\theta}_3(\sin\theta_2\sin\theta_3 + \cos\theta_2\cos\theta_3) + \frac{1}{2}J_2\dot{\theta}_2^2 + \frac{1}{2}J_3 \tag{22}$$

L'énergie potentielle, due à la pesanteur, est :

$$E_p = -m_2 g\frac{l_2}{2}\cos\theta_2 - m_3 g(l_2\cos\theta_2 + \frac{l_3}{2}\cos\theta_3) \tag{23}$$

Les deux équations de Lagrange conduisent à :

$$a_{11}\ddot{\theta}_2 + a_{12}\ddot{\theta}_3 = b_1 \tag{24}$$

$$a_{21}\ddot{\theta}_2 + a_{22}\ddot{\theta}_3 = b_2 \tag{25}$$

avec :

$$a_{11} = m_2\frac{l_2^2}{4} + m_3 l_2^2 + J_2 \tag{26}$$

$$a_{22} = m_3\frac{l_3^2}{4} + J_3 \tag{27}$$

$$a_{21} = a_{12} = \frac{1}{2}m_3 l_2 l_3(\sin\theta_2\sin\theta_3 + \cos\theta_2\cos\theta_3) \tag{28}$$

$$b_1 = \frac{1}{2}m_3 l_2 l_3(\cos\theta_2\sin\theta_3 - \sin\theta_2\cos\theta_3)\dot{\theta}_3^2 - m_2 g\frac{l_2}{2}\sin\theta_2 - m_3 g l_2\sin\theta_2 \tag{29}$$

$$b_2 = \frac{1}{2}m_3 l_2 l_3(\sin\theta_2\cos\theta_3 - \cos\theta_2\sin\theta_3)\dot{\theta}_2^2 - m_3 g\frac{l_3}{2}\sin\theta_3 \tag{30}$$

On obtient finalement les accélérations angulaires :

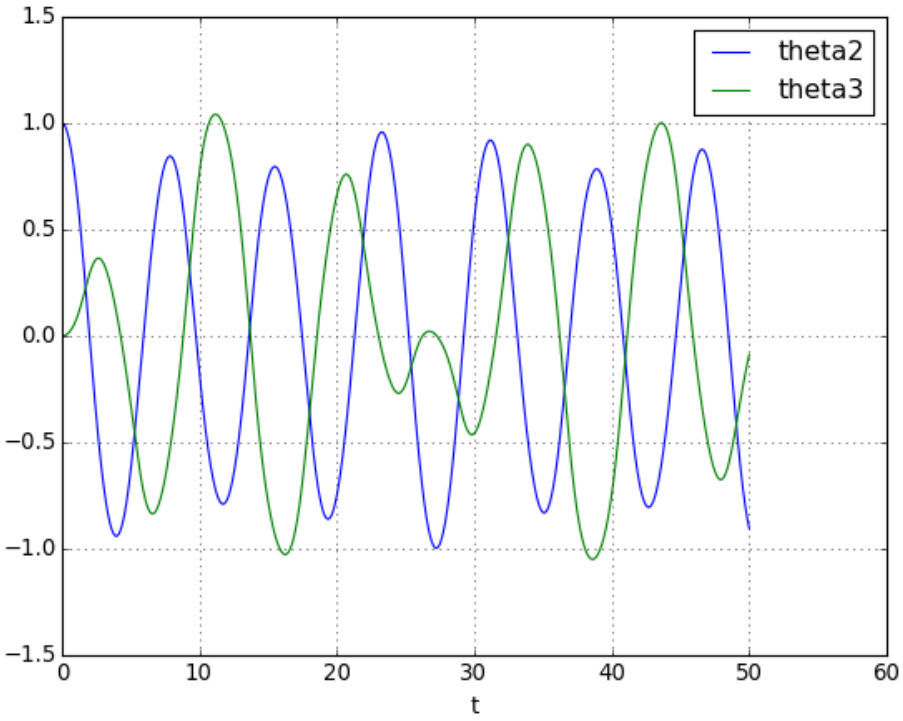
$$\ddot{\theta}_2 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}^2} \tag{31}$$

$$\ddot{\theta}_3 = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{12}^2} \tag{32}$$

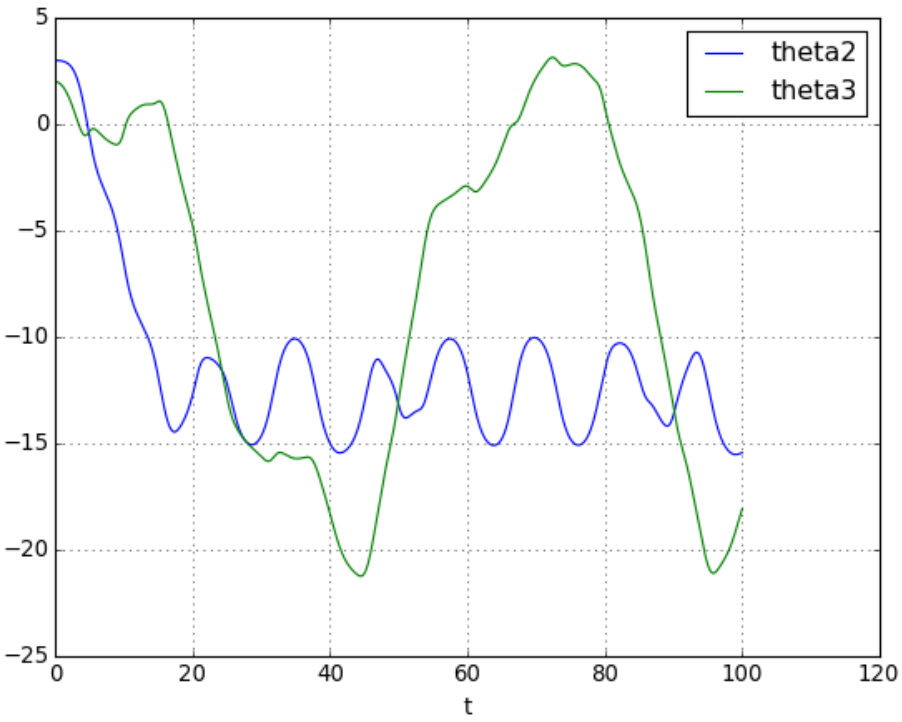
3.b. Intégration numérique

```
class PenduleDouble2:
    def __init__(self,l2,l3,m2,m3,J2,J3):
        self.a11 = m2*l2**2/4+m3*l2**2+J2
        self.a22 = m2*l2**2/4+J3
        self.d = self.a11*self.a22
        self.u = 0.5*m3*l2*l3
        self.v = m2*l2*0.5+m3*l2
        self.w = m3*l3*0.5
        self.theta2 = 0.0
        self.theta3 = 0.0
        self.dtheta2 = 0.0
        self.dtheta3 = 0.0

    def pas_euler(self,h):
        self.theta2 += self.dtheta2*h
        self.theta3 += self.dtheta3*h
        sin2 = math.sin(self.theta2)
        cos2 = math.cos(self.theta2)
        sin3 = math.sin(self.theta3)
        cos3 = math.cos(self.theta3)
        a12 = self.u*(sin2*sin3+cos2*cos3)
        det = self.d-a12**2
        b1 = self.u*(cos2*sin3-sin2*cos3)*self.dtheta3**2-self.v*sin2
        b2 = self.u*(sin2*cos3-cos2*sin3)*self.dtheta2**2-self.w*sin3
        self.dtheta2 += (self.a22*b1-a12*b2)/det*h
        self.dtheta3 += (self.a11*b2-a12*b1)/det*h

    def euler(self,T,Yi,h):
        Y = Yi
        t = 0.0
        liste_y = [Yi]
        liste_t = [t]
        self.theta2 = Yi[0]
        self.dtheta2 = Yi[1]
        self.theta3 = Yi[2]
        self.dtheta3 = Yi[3]
        while t < T:
            self.pas_euler(h)
            t += h
            liste_y.append([self.theta2,self.dtheta2,self.theta3,self.dtheta3])
            liste_t.append(t)
        return (numpy.array(liste_t),numpy.array(liste_y))


pendule = PenduleDouble2(1.0,1.0,1.0,1.0,1.0,1.0)
Yi = [1.0,0.0,0.0,0.0]
(t,y) = pendule.euler(50,Yi,1e-3)
```

```
figure()
plot(t,y[:,0],label="theta2")
plot(t,y[:,2],label="theta3")
legend(loc="upper right")
xlabel("t")
grid()
```




```
Yi = [3.0,0.0,2.0,0.0]
(t,y) = pendule.euler(100,Yi,1e-3)
figure()
plot(t,y[:,0],label="theta2")
plot(t,y[:,2],label="theta3")
legend(loc="upper right")
xlabel("t")
grid()
```



Références

[1] A.A. Shabana, *Computational dynamics*, (John Wiley and Sons, 2001)
[2] H. Goldstein, *Classical mechanics*, (Addison-Wesley, 1980)