TP11 – Pendule non-linéaire

Amar AHMANE
MP2I

15 décembre 2021

Le but de ce TP est de résoudre analytiquement une équation différentielle non-linéaire décrivant un système physique : il s'agira ici du pendule simple qui ne présente pas de solution analytique simple.

Étude théorique et équations utilisées

Le schéma ci-dessous représente le système physique étudié

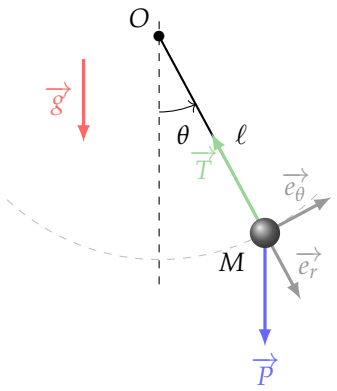

FIGURE 1 – *Pendule simple et forces considérées pour son étude*

On assimile l'objet de masse m à son centre de gravité M. Ainsi, le système étudié est $\{M\}$. Le référentiel choisi est le référentiel terrestre, considéré galiléen pour la durée de l'étude. L'étude cinématique nous donne

$$\overrightarrow{OM} = \ell\vec{e_r} \qquad \vec{v} = \ell\dot{\theta}\vec{e_\theta} \qquad \vec{a} = \ell\ddot{\theta}\vec{e_\theta} - \ell\dot{\theta}^2\vec{e_r}$$

L'application du P.F.D nous donne

$$\ddot{\theta} + \frac{g}{\ell}\sin\theta = 0 \tag{1}$$

Compte rendu

Échauffement

1. C.f `tp11_pendule.py`
2. Il nous suffit de lancer le programme, de déterminer le temps correspondant à 5 périodes en s'aidant du curseur, puis de diviser par 5. On obtient

   $$\boxed{T_0 \simeq 2s}$$

   On aurait pu récupérer cette information directement à l'aide d'un calcul au sein du programme, on s'occupera de cette automatisation plus tard, quand on y trouvera plus d'intérêt.
3. `V0[0]` correspond à θ_0, `V0[1]` à ω_0.
4. On remarque que θ_0 n'a aucune influence sur l'évolution de la période du pendule. C'était bien prévisible, en effet, notre étude théorique nous fournit entre autres l'expression de la période :

   $$\boxed{T = 2\pi\sqrt{\frac{\ell}{g}}}$$

   qui ne dépend donc pas de θ_0. Ceci reste vraisemblablement valide que dans le cadre de l'approximation des petits angles.

Isochronisme ?

5. En prenant $x = \theta$ et $y = \dot{\theta}$, l'équation (1) se réécrit $\dot{y} + \frac{g}{\ell} \sin x = 0$, donc $\boxed{\dot{y} = -\frac{g}{\ell} \sin x}$. D'autre part $\boxed{\dot{x} = y}$.
6. On complète avec le code suivante

```
def pendule(V,t):
    """
    Fonction associee au pendule simple
    """
    x, y = V
    return [y, -(g/l)*np.sin(x)]
```

7. Le résultat obtenu est représenté ci-dessous

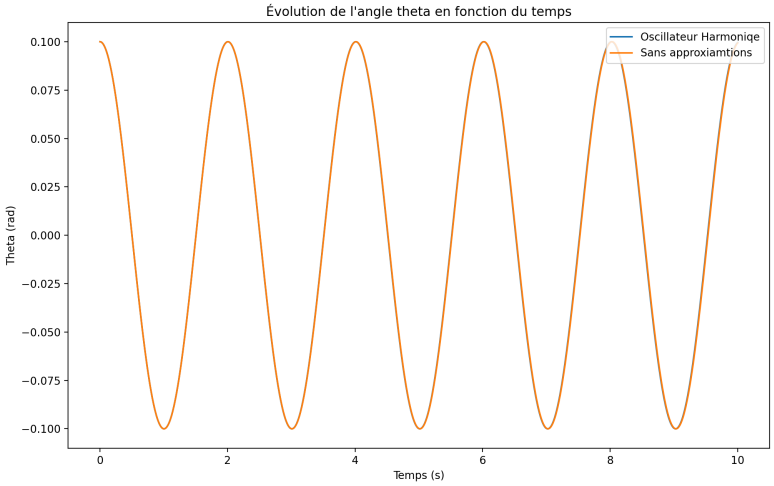

FIGURE 2 – *Deux simulations différentes pour les mêmes conditions initiales* $\theta_0 = 0.1$ *et* $\omega_0 = 0$.

8. Pour des amplitudes plus importantes, les deux simulations ne se superposent plus. Ces observations étaient attendues : en effet, pour des valeurs de θ_0 très petites devant 1, l'approximation des petits angles donnent des résultats satisfaisants ; mais plus l'angle de départ est important, plus les erreurs dues à l'approximation sont importantes : la simulation n'est plus aussi réaliste qu'on ne le voudrait.
9. On représente les résultats numériques dans une même figure :

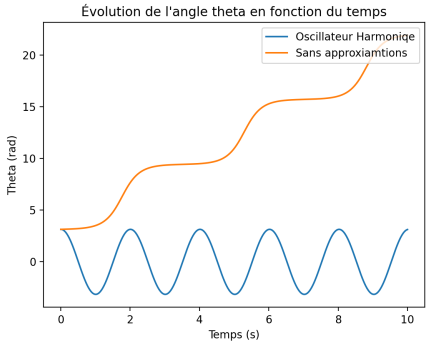

FIGURE 3 – *Deux simulations différentes pour les mêmes conditions initiales* $\theta_0 = \pi$ *et* $\omega_0 = 0.1$.

L'évolution de l'oscillateur harmonique ne diffère pas énormément de ce que l'on a pu observer jusqu'ici avec nos simulations : la période reste la même, l'évolution temporelle de l'angle θ représente une sinusoïde. La résolution numérique de l'équation non-approximée donne des résultats drastiquement différents : non seulement la période est différente, mais l'allure de

la courbe n'est plus du tout sinusoïdale ; physiquement, les données traduisent une rotation indéfinie du pendule autour de l'origine, avec un ralentissement lorsque la masse est au sommet (prévisible, vue la présence du poids).

10. On écrit la fonction suivante qui renverra la valeur de la période pour une valeur de `theta_0` donnée et pour un tableau `t` de temps donné :

```
def periode(theta_0, t):
    V0 = [theta_0, 0]
    V  = odeint(pendule, V0, t)
    v  = V[:,0]
    first = v[0]
    t1 = 0
    for i, e in enumerate(v):
        if e*first <= 0:
            t1 = t[i]
            break
    return 4*t1
```

On complète la section *Simulations numériques* du programme avec les variables et les tableaux dont on aura besoin pour représenter l'évolution de T en fonction de θ_0. En lançant le programme, on obtient

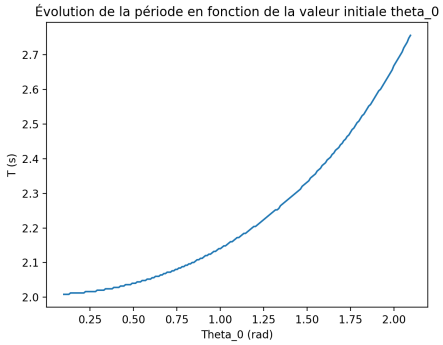

FIGURE 4 – *Représentation de $T(\theta_0)$.*

11. Ci-dessous, sur un même graphe, les mesures des périodes et les périodes calculées grâce à la fonction `borda` :

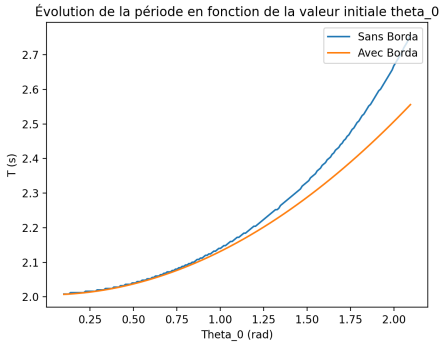

FIGURE 5 – *Représentations de $T(\theta_0)$.*

La fonction `borda` :

```
def borda(theta_0):
    return 2*np.pi*np.sqrt(l/g)(1+(theta_0**2)/16)
```

Pour des valeurs assez grandes de θ_0, on remarque que la formule de Borda ne prédit plus très bien la valeur de T pour des θ_0 trop grands.

12. On représente, en plus des deux autres *versions*, le graphe de $T(\theta_0)$ obtenu à l'aide de la fonction `periode_int` fournie :

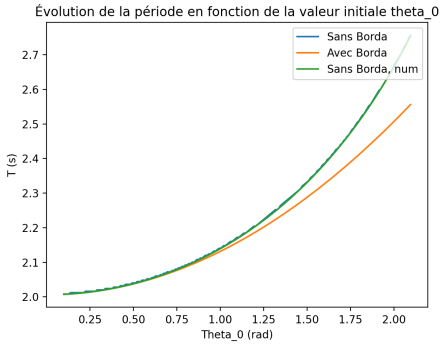

FIGURE 6 – *Représentations de $T(\theta_0)$.*

Le résultat obtenu se superpose quasiment avec le graphe que l'on obtient avec la fonction `periode`.

Énergie mécanique

13. On a directement $\boxed{\mathcal{E}_c = \frac{1}{2}ml^2\omega^2}$.

14. On a que $\mathrm{d}\mathcal{E}_p = -\delta W(\vec{P})$; on calcule :

$$\begin{aligned}\mathrm{d}\mathcal{E}_p &= -\delta W(\vec{P}) \\ &= -\vec{P}\cdot \mathrm{d}\overrightarrow{OM} \\ &= -\vec{P}\cdot(l\mathrm{d}\theta\vec{e_\theta}) \\ &= mgl\sin\theta\mathrm{d}\theta\end{aligned}$$

Ainsi, on a $\mathcal{E}_p = -mgl\cos\theta + k$, $k \in \mathbb{R}$. On choisit k tel que $\mathcal{E}_p = 0$ lorsque $\theta = 0$; d'où $\boxed{\mathcal{E}_p = mgl(1-\cos\theta)}$.

15. Les fonctions sont simples et utilisent les formules obtenues analytiquement :

```
1  def energie_cinetique(omega):
2      return (1/2)*m*(l**2)*(omega**2)
3
4
5  def energie_potentielle(theta):
6      return m*g*l*(1-np.cos(theta))
7
8
9  def energie_mecanique(theta, omega):
10     return energie_cinetique(omega) + ↘
           energie_potentielle(theta)
```

On représente l'évolution temporelle de chacune des grandeurs sur un même graphe :

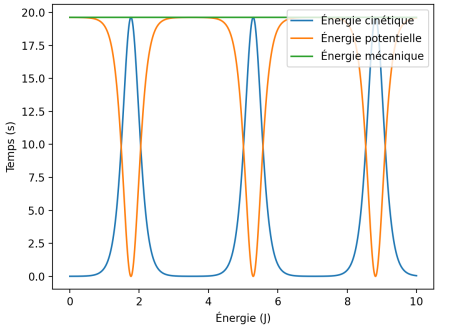

FIGURE 7 – *Évolutions temporelles de $\mathcal{E}_c$, $\mathcal{E}_p$ et $\mathcal{E}_m$.*

16. On résout à nouveau l'équation différentielle en s'aidant de la méthode d'Euler. On représente ensuite les énergies cinétique, potentielle et mécanique sur un même graphe. L'allule globale étant quasiment la même, on décide de zoomer sur une portion du graphe pour mettre en valeur ce qui semble être une anomalie

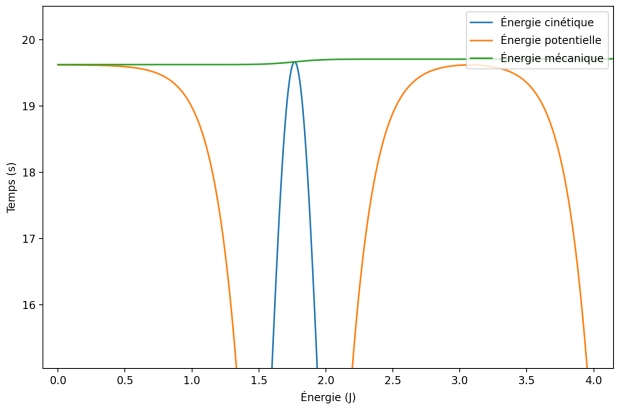

FIGURE 8 – *L'anomalie.*

On le remarque directement : l'énergie mécanique ne semble pas être constante ; n'y a-t-il plus conservation de l'énergie mécanique ? Cela ne peut être, toutes les forces agissant sur le système sont conservatives. L'anomalie vient ici de la méthode utilisée, dont la solution donnée semble contenir trop d'erreurs.

17. En rajoutant des frottements, on introduit un terme en $k\ell\dot{\theta}$ dans l'équation (1). On modifie la fonction `pendule` en conséquence :

```
def pendule(V,t):
    """
    Fonction associee au pendule simple
    """
    x, y = V
    return [y, -(g/l)*np.sin(x)-k*l*y]
```

On représente enfin les énergies cinétique, potentielle et mécanique :

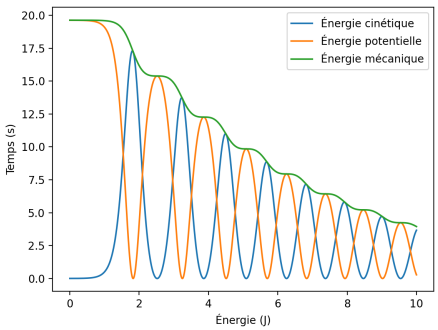

FIGURE 9 – *Évolutions temporelles de $\mathcal{E}_c$, $\mathcal{E}_p$ et $\mathcal{E}_m$ (avec frottements).*

On remarque qu'il n'y a plus conservation de l'énergie mécanique : cela est du aux frottements, qui constituent une force non-conservative.