

Animations en Python

Table des matières

1	Introduction	5
2	Représentation d'un système	5
2.1	Principe	5
2.2	Les classes PenSim , PenDbl et PenCha	5
2.2.1	Création d'une instance	5
2.2.2	Les attributs	5
2.2.3	Les méthodes d'instance	6
2.3	La classe MethNumInt	6
2.4	La classe AnimObj	6
3	L'animation	7
4	Cas particulier du pendule sur un chariot mobile	7

1 Introduction

Un environnement de programmation en Python est proposé afin de réaliser facilement des animations sur des systèmes mécaniques.

Les exemples proposés sont les suivants :

- Pendule simple
- Pendule double
- Pendule sur chariot avec commande d'asservissement (voir 4 pour les spécificités)

2 Représentation d'un système

2.1 Principe

Dans le fichier **objets.py**, on définit trois classes « objet » (**PenSim**, **PenDb1**, **PenCha**) associées chacune à la description d'un système mécanique, comme ceux mentionnés en introduction.

Elles héritent des classes **MethNumInt** (méthodes d'intégration numérique d'équation différentielle) et **AnimObj** (méthodes pour animer le graphique) qui définissent des méthodes communes aux trois classes mentionnées.

2.2 Les classes **PenSim**, **PenDb1** et **PenCha**

Chaque classe « objet » (**PenSim**, **PenDb1**, **PenCha**) associée à la description d'un système est organisée de la même façon. Pour décrire cette organisation, l'exemple du pendule simple sera utilisé.

2.2.1 Création d'une instance

Le nom de la classe est par exemple **PenSim** pour le pendule simple. La création d'une instance (un pendule avec ses propres paramètres) se fera ainsi :

```
import objets
pensim = objets.PenSim(l,g)
```

Ici, **l** est la longueur du pendule et **g** l'accélération de la pesanteur.

2.2.2 Les attributs

Dans l'instance créée, on pourra ainsi utiliser les paramètres propres au système (ici, longueur **self.l** du pendule et accélération de la pesanteur **self.g**). Les attributs suivants sont ceux associés à l'animation du graphique.

- **self.T** : durée exacte de l'animation en s

- **self.Tdeb** : délai approximatif en s après lequel l'animation commence (doit être différent de 0)
- **self.interval** : intervalle de temps approximatif en ms entre deux mises à jour du graphique (mais non constant en réalité et plus long)
- **self.dt** : pas de calcul constant en s pour la résolution numérique de l'équation différentielle régissant le mouvement.
- **self.y0** : état initial du système

2.2.3 Les méthodes d'instance

2.2.3.1 méthode `__init__()`

On initialise notamment les attributs **self.l** et **self.g** quand l'instance est créée. Les valeurs par défaut de tous les attributs aboutissent à un fonctionnement correct.

2.2.3.2 méthode `F()`

L'équation dynamique du système est mise sous la forme :

$$\frac{dY}{dt} = F(t, Y)$$

La méthode `F()` prend en paramètre l'instant **t** et le vecteur d'état **Y** (pour le pendule simple, l'angle avec la verticale et la vitesse angulaire) et retourne **Yp** (ce que vaut la dérivée du vecteur d'état à l'instant considéré). Cette méthode sera utilisée pour la résolution numérique de l'équation différentielle régissant le mouvement.

2.2.3.3 méthode `init_graph()`

La méthode configure la structure générale du graphique, via l'initialisation de l'attribut **self.fig**. L'attribut **self.listegraph** doit rassembler dans une liste les éléments particuliers qui seront modifiés durant l'animation.

2.2.3.4 méthode `graph()`

La méthode prend en paramètre l'instant **t** et le vecteur d'état **Y**, et sera lancée autant de fois que nécessaire pour la mise à jour du graphique, via la modification des éléments rassemblés dans la liste **self.listegraph**.

2.3 La classe `MethNumInt`

Cette classe rassemble les méthodes numériques d'intégration pour les équations différentielles (`rk2()` ou `rk4()`). L'attribut **self.methode_int** désignera la méthode choisie dans chaque classe (`PenSim`, `PenDb1`, `PenCha`) qui en hérite. On y trouve notamment la méthode de Runge-Kutta d'ordre 4, qui sera celle choisie par défaut (**self.methode_int = getattr(self, meth)** avec **meth="rk4"**).

2.4 La classe `AnimObj`

Cette classe gère la mise en œuvre de l'animation. La méthode **animate()** est appelée à chaque itération, le paramètre **n** désignant le numéro de l'itération.

Cette méthode **animate()** est lancée via une instance de la classe **FuncAnimation** et met ainsi à jour le graphique, le paramètre **n** étant incrémenté d'une unité à chaque mise à jour. Pour remédier au fait que l'intervalle de temps **self.interval** (voir 2.2.2) ne correspond pas au temps qui s'est réellement écoulé entre deux mises à jour consécutives du graphique, on va mesurer ce temps réellement écoulé (fonction **time()**) pour faire avancer la simulation numérique du nombre variable de pas correspondant.

Ainsi, le temps affiché dans le graphique sera bien le temps réel qui s'écoule, et l'état final de la simulation sera toujours le même, quelle que soit la cadence aléatoire de rafraîchissement du graphique.

La méthode **anim()** lance l'animation complète.

3 L'animation

Avec les classes ainsi définies, le lancement de l'animation est très simple pour chaque objet. Les fichiers mentionnés ci-dessous sont directement exécutables. On y crée une instance de la classe choisie, puis on lance l'animation associée à cette instance.

Fichier **pendule_simple.py** :

```
import objets
pensim = objets.PenSim()
pensim.anim()
```

Fichier **pendule_double.py** :

```
import objets
pendbl = objets.PenDb1()
pendbl.anim()
```

Fichier **pendule_chariot.py** :

```
import objets
pencha = objets.PenCha()
pencha.anim()
```

4 Cas particulier du pendule sur un chariot mobile

Pour la classe **PenCha** dans le fichier **objets.py**, on remarquera que d'autres méthodes d'instance ont été ajoutées à celles déjà décrites (voir 2.2.3). Elles introduisent une commande qui est une force horizontale s'appliquant sur le chariot, permettant le maintien du pendule dans sa position d'équilibre instable. On peut interrompre l'application de cette commande à un instant donné (paramètre **tcut**) pour observer ensuite l'évolution du système libre. L'élaboration de cette commande n'est donnée qu'à titre d'exemple et ne fonctionne qu'avec les paramètres par défaut.