TP 2
Intégration numérique d'équations différentielles

À la fin de la séance: envoyer vos programmes (format .py) et graphiques (format pdf) à l'adresse ~~vincent.ballenegger@univ-fcomte.fr~~ *(spécifier PAN2:TP 2 comme titre de message).*

david.cornu@utinam.cnrs.fr

1. Mouvement d'un projectile

On veut calculer la trajectoire $\vec{r}(t)$ d'un projectile de masse m = 5 kg. Elle obéit à l'équation de Newton

$$m\vec{r}''(t) = \vec{F}(\vec{r}(t), \vec{r}'(t)) \qquad (1)$$

où $\vec{F}(\vec{r}(t), \vec{r}'(t))$ est la force agissant sur le projectile, qui dépend en général de sa position $\vec{r}(t)$ et de sa vitesse $\vec{v}(t) = \vec{r}'(t)$. Le mouvement a lieu dans le plan Oxy:

$$\vec{r}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \qquad \vec{v}(t) = \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix}.$$

Le projectile est lancé depuis l'origine, avec une vitesse initiale $\vec{v}_0 = v_0 \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$ où θ est l'angle entre $\vec{v}_0$ et l'horizontale. Pour intégrer numériquement l'équation de Newton, qui est une équation différentielle ordinaire (EDO) d'ordre 2, il est utile de la réécrire comme un système d'EDO d'ordre 1:

$$\begin{cases} x'(t) = v_x(t) \\ y'(t) = v_y(t) \\ v_x{}'(t) = F_x/m \\ v_y{}'(t) = F_y/m \end{cases} \qquad (2)$$

1) Calculer la trajectoire $\vec{r}(t)$ en intégrant numériquement le système d'équations (2) avec la méthode d'Euler pour $0 \leq t \leq 5$ s. On suppose que le projectile ne ressent que la force de gravitation $m\vec{g}$.

Structure du programme

```
m = 5.0              # masse (kg)
g = 9.81             # champ de pesanteur (m/s^2)
v0 = 30              # en m/s
theta = pi/4
ti, tf = 0, 5        # temps initial et final
dt = 0.1             # pas de temps
t = np.arange(___)   # tableau contenant les différents temps t_i = i Δt, i = 0, …, N
N = ___              # nb de pas d'intégration (doit être cohérent avec la durée tf−fi et avec dt)
r = ___              # tableau formé de N+1 lignes et 2 colonnes, initialisé à 0.
```

Le tableau `r` *sert à stocker les positions* $\vec{r}_i = \vec{r}(t_i)$*. La ligne* `i` *contient les composantes x (colonne 0) et y (colonne 1) du vecteur* $\vec{r}_i$*. L'expression* `r[i]` *retourne le tableau à 1 dimension représentant le vecteur* $\vec{r}_i$*.*

```
v = ___              # idem pour les vitesses

# Condition initiale
r[0] = (0, 0)
v[0] = ___

# Calcul de la trajectoire
for i in range(1,N):
    ______  #Calculer les vecteurs r[i] et v[i] en utilisant la méthode d'Euler
```

Question optionnelle :

Pour améliorer la lisibilité du programme, définir une fonction `F(r, v)` retournant le vecteur $\vec{F}(\vec{r}, \vec{v})$. Dans cette fonction `F`, les vecteurs sont représentés par des tableaux à 1 dimension contenant les composantes selon les axes x et y.

2) Tracer, avec matplotlib, un graphique de la trajectoire calculée au point 1).

ou RK4
(les méthodes sont à coder vous même)

3) Calculer la trajectoire selon l'algorithme de Velocity-Verlet, pour le même pas de temps `dt = 0.1`.
Définir des tableaux `r2` *et* `v2` *pour effectuer ce calcul.*

4) Tracer un graphique montrant les deux trajectoires (Euler et Velocity-Verlet) et la trajectoire exacte.
Zoomer sur le graphique, en utilisant le mode interactif, pour déterminer, à 0.1 m près, les points de chute du projectile prédits par les 2 méthodes (pour `dt = 0.1`).
Position de l'impact: ______________ (selon Euler), ________________ (selon Velocity-Verlet).

5) Tracer un graphique de l'énergie au cours du temps pour les deux méthodes.

6) Recalculer les trajectoires en ajoutant la force de frottement avec l'air $\vec{F}_f = -\frac{1}{2}\rho_{air} C_f A |\vec{v}|\vec{v}$.

Prendre comme paramètres:

```
rho_air = 1.3          # kg/m^3
C_f = 0.45             # coefficient de frottement (m/s^2)
A = pi*(0.15)**2       # surface de l'objet perpendiculairement au mouvement
```

Quel point de chute prédisez-vous maintenant ? _______________

2. Stabilité de différents intégrateurs

On considère le système d'EDO suivant:

$$\begin{cases} x'(t) = -\frac{1}{4}x(t) + y(t) \\ y'(t) = -x(t) - \frac{1}{4}y(t) \end{cases}$$

avec condition initiale $x(0) = 1$ et $y(0) = 0$.

1) Tracer un graphique de la solution (x(t), y(t)) calculée en utilisant la méthode d'Euler pour $0 \leq t \leq 30$ et un pas de temps $dt = 0.1$.

2) Déterminer le pas de temps dt maximal permettant d'obtenir une solution qualitativement correcte.

3) Quelle forme de trajectoire prédit la méthode d'Euler lorsque $dt = 0.47$? Et lorsque $dt = 0.6$?

4) On peut intégrer un système d'ODE de la forme $\vec{y}'(t) = \vec{f}(\vec{y}(t), t)$ en utilisant la fonction `odeint()` du module `scipy.integrate` :

```
from scipy.integrate import odeint

def derivee(vec_y, t):      #Cette fonction retourne le membre de droite f(y(t), t) du système différentiel
    x, y = vec_y
    dydt = [-0.25*x + y, -x - 0.25*y]
    return dydt
vec_y0 = [1, 0]                       #condition initiale
sol = odeint(derivee, vec_y0, t)
```

`sol` est un tableau à 2 colonnes contenant la solution ($x(t)$, $y(t)$) pour la discrétisation du temps spécifiée dans le tableau `t`. Tester la stabilité de l'intégrateur utilisé par `odeint()` en traçant la solution obtenue pour des pas de temps dt de plus en plus grand.