Méthode d'Euler

La méthode d'Euler permet de déterminer une solution approchée d'un problème de Cauchy, *i.e.* d'une ou plusieurs équations différentielles munies de conditions initiales. Cette méthode est utile car il est souvent impossible de résoudre analytiquement l'équation différentielle modélisant un phénomène physique.

Vous avez déjà mise cette méthode en pratique en physique-chimie (évolution des concentrations des constituants d'un mélange en chimie, évolution de la tension ou de l'intensité dans un circuit électrique, diffusion thermique).

1 Ordre 1

1.1 Principe mathématique

On considère un problème de Cauchy $(\star)\begin{cases}y'(t)=f(t,y)\\y(t_0)=y_0\end{cases}$ où $t_0\in\mathbb{R}$ et $y_0\in\mathbb{R}$ définissent la condition initiale. On cherche une fonction solution $y\colon [t_0\,;T]\to\mathbb{R}$.

Numériquement, cela consiste à déterminer un certain nombre de points de la courbe représentant y sur cet intervalle. On va donc choisir des abscisses t_0, t_1, t_2, ... dans l'intervalle $[t_0\,;T]$, espacées d'un pas constant h, et chercher des valeurs approchées de $y(t_0)$, $y(t_1)$, $y(t_2)$, ...

En intégrant l'équation différentielle de $(\star)$ sur un intervalle de la forme $[t_i\,;t_{i+1}]$, il vient

$$\begin{aligned}\int_{t_i}^{t_{i+1}} y'(t)\,\mathrm{d}t=\int_{t_i}^{t_{i+1}} f(t,y)\,\mathrm{d}t &\iff y(t_{i+1})-y(t_i)=\int_{t_i}^{t_{i+1}} f(t,y)\,\mathrm{d}t\\ &\iff y(t_{i+1})=y(t_i)+\int_{t_i}^{t_{i+1}} f(t,y)\,\mathrm{d}t.\end{aligned}$$

Ainsi, en partant de la valeur $y(t_0)$ connue comme condition initiale, si on connaît une valeur approchée de cette intégrale sur $[t_0\,;t_1]$, on détermine une valeur approchée y_1 de $y(t_1)$. En procédant de même sur $[t_1\,;t_2]$, à partir de y_1, on détermine une valeur approchée y_2 de $y(t_2)$ et ainsi de suite de proche en proche.

L'idée de la méthode d'Euler est d'approximer la fonction à intégrer $f(t,y)$ sur $[t_i\,;t_{i+1}]$ par sa valeur en $t=t_i$, *i.e.* par la constante $f(t_i,y_i)$ où y_i est l'approximation précédente obtenue pour $y(t_i)$ (au départ $y_0=y(t_0)$ d'après la condition initiale donnée dans $(\star)$). De cette façon l'intégrale vaut

$$\int_{t_i}^{t_{i+1}} f(t,y)\,\mathrm{d}t=f(t_i,y_i)\int_{t_i}^{t_{i+1}}\mathrm{d}t=(t_{i+1}-t_i)f(t_i,y_i)=hf(t_i,y_i),$$

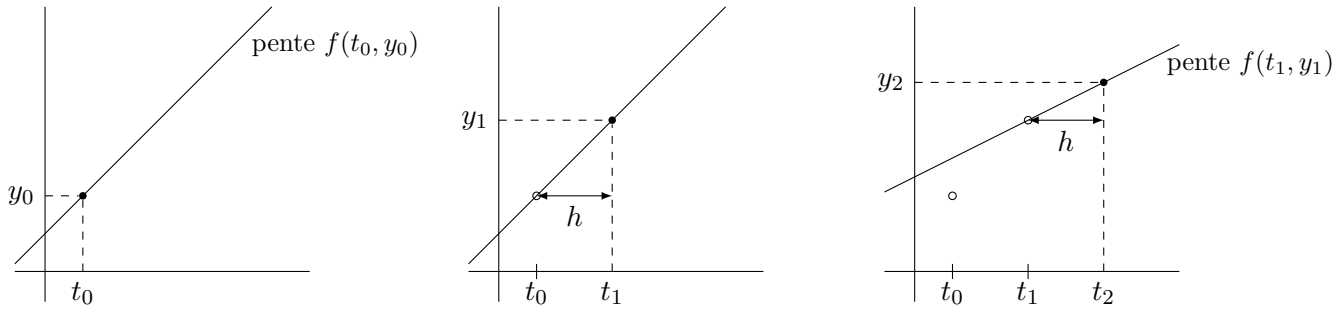
où h est le pas de temps ($h=t_{i+1}-t_i$ pour tout i).

Bilan : Soit $h\in\mathbb{R}_+^*$ le pas de temps. La solution numérique est donnée par les points de coordonnées (t_i,y_i) où t_0 et y_0 sont donnés par la condition initiale puis $\forall i\in\mathbb{N}$, $\begin{cases}t_{i+1}=t_i+h,\\y_{i+1}=y_i+hf(t_i,y_i).\end{cases}$

1.2 Interprétation géométrique

L'idée de la méthode d'Euler peut être reformulée géométriquement : on approxime la courbe de la fonction solution du problème de Cauchy par ses tangentes successives sur des intervalles de longueur h, le pas de temps.

Plus précisément, d'après $(\star)$, la tangente au point d'abscisse $t=t_0$ a pour coefficient directeur $y'(t_0)=f(t_0,y_0)$. On approxime donc notre solution par la droite passant par (t_0,y_0) et de pente $f(t_0,y_0)$, et ce sur un intervalle de longueur h. Ceci nous permet de définir un point (t_1,y_1) à partir duquel on répète le procédé comme on peut le voir ci-dessous :



La même chose en vidéo : `https://youtu.be/-d7qrNkPDtQ`

1.3 Algorithme en python

Il s'agit essentiellement de créer les listes des termes des suites (t_i) et (y_i) données par les relations obtenues à la fin du paragraphe 1.1. On donne ci-dessous le code correspondant à la méthode d'Euler proprement dite, en pratique il faut de plus définir la fonction f décrivant l'équation différentielle $y' = f(t, y)$.

```
def euler(f: callable, t0: float, y0: float, tmax:float, h: float)
                                                    -> ([float], [float]):
    """
    Renvoie deux listes contenant respectivement les abscisses
    et les ordonnées des points de la solution numérique du
    problème de Cauchy y'(t) = f(t, y) avec y(t0) = y0
    sur l'intervalle [t0, tmax] avec un pas de temps h.
    """
    t = t0
    y = y0
    abscisses = [t0]
    ordonnees = [y0]
    while t < tmax:
        y = y + h*f(t, y)
        ordonnees.append(y)
        t = t + h
        abscisses.append(t)
    return abscisses, ordonnees
```

On peut imaginer facilement des variantes concernant l'implémentation comme :

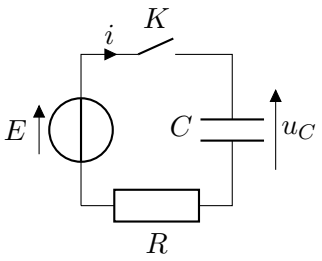
— donner le nombre de points souhaités au lieu du pas ;

— initialiser des listes de 0 pour les abscisses et les ordonnées et les modifier au fur et à mesure ;

— passer en argument la liste des valeurs de t pour lesquelles on souhaite calculer un point.

1.4 Un exemple

On considère le circuit RC ci-contre où l'interrupteur K est ouvert pour $t < 0$ et le condensateur est initialement déchargé ($u_C(0) = 0$). À $t = 0$ on ferme l'interrupteur et on cherche à déterminer l'évolution de la tension $u_C(t)$ aux bornes du condensateur.



On montre classiquement en physique que u_C vérifie l'équation différentielle

$$\frac{\mathrm{d}u_C}{\mathrm{d}t} + \frac{1}{\tau}u_C = \frac{E}{\tau} \quad \text{où } \tau = RC.$$

On peut alors utiliser la méthode d'Euler pour déterminer une solution approchée. Pour cela, on met l'équation différentielle sous la forme $\dfrac{\mathrm{d}u_C}{\mathrm{d}t} = \dfrac{E}{\tau} - \dfrac{1}{\tau}u_C$, *i.e.* $y' = f(t, y)$ avec $f(t, y) = \dfrac{E - y}{\tau}$.

Voici le code créant les listes des t_i et des y_i dans le cas où $E = 3$ V et $\tau = 5$ s pour une solution sur l'intervalle $[0\,;20]$ construite avec un pas de 1.

```
E = 3
tau = 5

def f(t, y):
    return (E - y) / tau

abs, ord = euler(f, 0, 0, 20, 1)
```

On peut alors tracer le résultat obtenu (résultat en figure 1, ce code n'est pas à connaître précisément, il faut juste comprendre le principe, une aide concernant ces fonctions pour effectuer un tracé vous sera toujours donné) :

```
import matplotlib.pyplot as plt
# On trace un nuage de points grâce aux listes des t_i (abs) et y_i (ord).
plt.scatter(abs, ord, color='r', marker='x')
plt.xlabel("temps (en s)")  # Légender les axes
plt.ylabel("tension u_c (en V)")
plt.show()  # Afficher le graphique
```

On aurait bien sûr aussi pu résoudre analytiquement cette équation différentielle [1] pour trouver la solution exacte qui est

$$\forall t \geqslant 0,\ u_C(t) = E\left(1 - \mathrm{e}^{-t/\tau}\right) \text{ avec } \tau = RC.$$

On peut alors également la tracer, voir figure 2, pour comparer avec la solution obtenue via la méthode d'Euler.

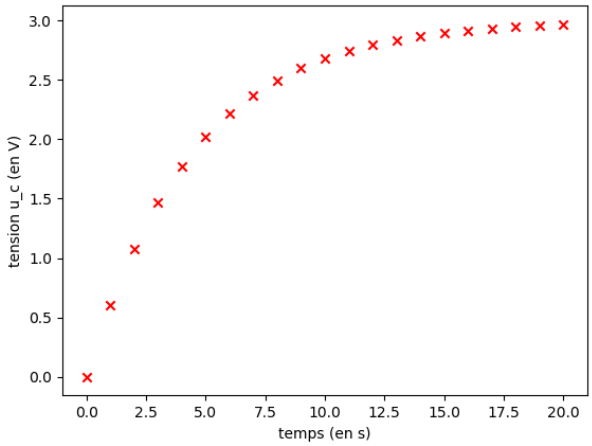


FIGURE 1 – Solution numérique

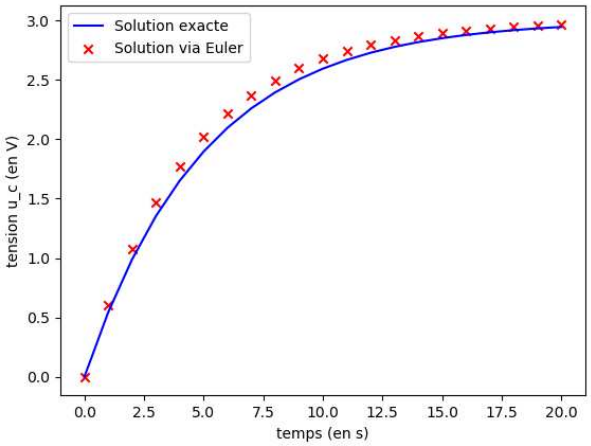


FIGURE 2 – Comparaison des solutions théorique et numérique

1.5 Influence du pas

Lorsqu'on utilise la méthode d'Euler pour résoudre une équation différentielle, se pose la question de la pertinence de la solution numérique obtenue. Le problème principal est celui du choix du pas.

On le voudrait petit pour améliorer la précision (voir figure 3) mais ceci allonge le temps de calcul. Plus précisément, si on veut n points, la complexité de l'algorithme précédent est en $O(n)$, *i.e.* linéaire en n.

Par ailleurs, on peut montrer que la précision évolue aussi de façon linéaire avec le pas choisi [2], cela signifie que pour améliorer la précision d'un facteur 10, il faut calculer 10 fois plus de points, ce qui prend 10 fois plus de temps. Il faut donc opter pour un compromis acceptable entre temps de calcul et précision souhaitée.

1. Méthode usuelle : solution homogène sous forme d'une exponentielle puis solution particulière constante et enfin utilisation de la condition initiale $u_C(0) = 0$.
2. On dit que la méthode d'Euler est d'ordre 1.

Remarque : si on choisit un pas trop grand, la solution numérique peut être très éloignée de la solution analytique. Il peut même y avoir une propagation d'erreurs qui donne un résultat aberrant qui n'a plus de sens physique comme illustré figure 4.

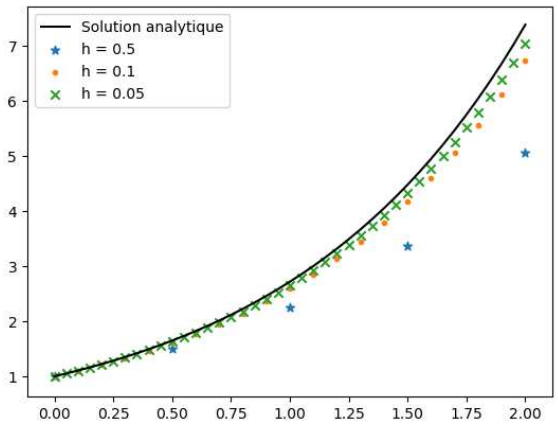


FIGURE 3 – Influence du pas sur la précision de la solution de l'équation $y' = y$ avec $y(0) = 1$.

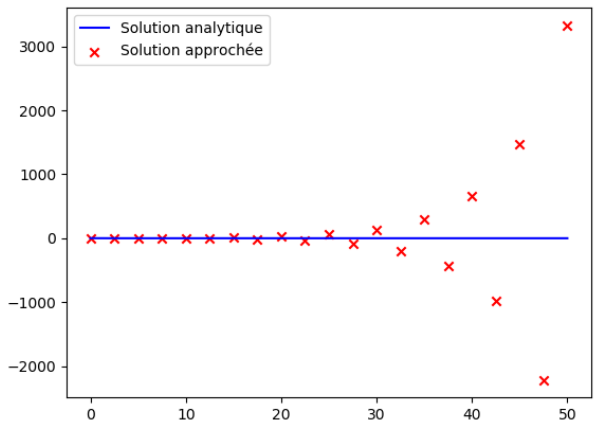


FIGURE 4 – Solution numérique aberrante pour l'équation $y' = -y$ avec $y(0) = 1$ à cause d'un pas trop grand.

2 Adaptation pour l'ordre 2

2.1 Le principe

Si on a affaire à une équation différentielle d'ordre 2, on se ramène à un système différentiel de deux équations d'ordre 1. Plus précisément, considérons le problème de Cauchy

$$(\diamond)\ y'' = f(t, y, y') \text{ avec } y(t_0) = y_0 \text{ et } y'(t_0) = z_0.$$

En posant $z = y'$, on a $z' = y'' = f(t, y, z)$ et ainsi

$$(\diamond) \iff \begin{cases} y' = z \\ z' = f(t, y, z) \end{cases} \quad \text{avec} \quad \begin{cases} y(t_0) = y_0, \\ z(t_0) = z_0. \end{cases}$$

Par un procédé analogue à celui utilisé pour l'ordre 1, on obtient une solution approchée en calculant des points (t_i, y_i), avec en plus une suite auxiliaire (z_i), où t_0, y_0 et z_0 sont donnés par les conditions initiales

puis $\forall i \in \mathbb{N}$, $\begin{cases} t_{i+1} = t_i + h \\ y_{i+1} = y_i + hz_i \\ z_{i+1} = z_i + hf(t_i, y_i, z_i) \end{cases}$ où h est le pas de temps.

2.2 L'algorithme en python

Comme pour l'ordre 1, il suffit de calculer les termes successifs des suites (t_i), (y_i) et (z_i) à partir des conditions initiales et de la fonction f définissant l'équation différentielle. Voici une version où l'on choisit le nombre de points souhaités et non le pas :

```
def euler2(f: callable, t0: float, y0: float, z0: float, n: int,
                            tmax: float) -> ([float], [float]):
    """
    Renvoie deux listes contenant respectivement les abscisses
    et les ordonnées des n points de la solution numérique du
    problème de Cauchy y''(t) = f(t, y, y') avec y(t0) = y0 et
    y'(t0) = z0 sur l'intervalle [t0, tmax].
    """
    T = [t0]
```

```
    Y = [y0]
    Z = [z0]
    h = (tmax - t0) / (n - 1)  # Le pas de temps
    for i in range(n):
        T.append(T[i] + h)
        Y.append(Y[i] + h*Z[i])
        Z.append(Z[i] + h*f(T[i], Y[i], Z[i]))
    return T, Y
```

2.3 Un exemple

On considère un pendule simple de longueur ℓ subissant des frottements de constante k. On le lâche initialement d'un angle θ_0 avec une vitesse nulle. On peut montrer que l'angle avec la verticale vérifie

$$\ddot{\theta} + k\dot{\theta} + \omega_0^2 \sin\theta = 0 \quad \text{avec } \theta(0) = \theta_0 \text{ et } \dot{\theta}(0) = 0 \text{ et où } \omega_0 = \sqrt{\frac{g}{\ell}}.$$

Cette équation étant trop compliquée à résoudre analytiquement (sauf si on utilise l'approximation des petits angles), on utilise la méthode d'Euler en écrivant $\ddot{\theta} = f(t, \theta, \dot{\theta})$ où $f(t, y, z) = -kz - \omega_0^2 \sin(y)$. Ci-dessous le code et la figure obtenue pour déterminer une solution approchée. Pour simplifier, on a pris $\omega_0 = 1$.

```
import matplotlib.pyplot as plt
from math import sin, pi

theta0 = pi/4
k = 0.1

def f(t, y, z):
    return -sin(y) - k*z

abs, ord = euler2(f, 0, theta0, 0, 1000, 50)
plt.scatter(abs, ord, color='r', marker='x', s=0.1)
plt.xlabel("temps (en s)")
plt.ylabel("angle (en rad)")
plt.show()
```

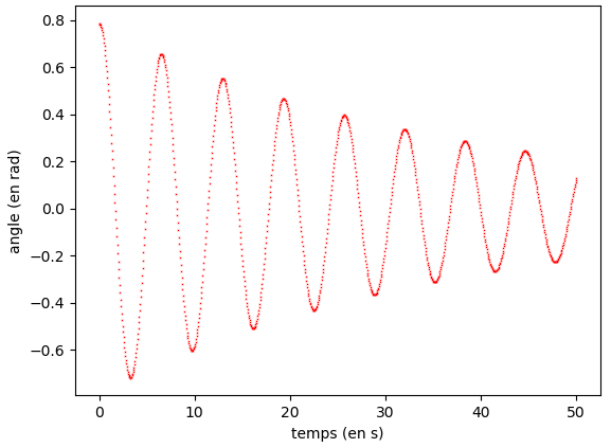


FIGURE 5 – Solution numérique pour le pendule amorti ($k = 0{,}1$).

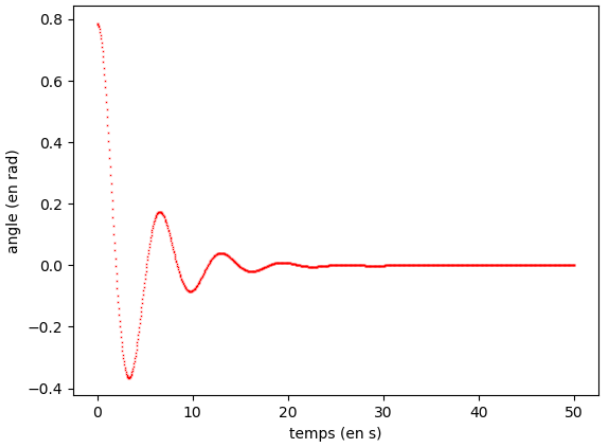


FIGURE 6 – Idem avec $k = 0{,}5$.

3 Utilisation d'une bibliothèque

En première année en physique, vous avez utilisé la fonction `odeint` de la bibliothèque `scipy.integrate` pour résoudre des équations différentielles [3]. Elle ne repose pas sur la méthode d'Euler mais sur une méthode plus élaborée. Cependant, le principe est globalement le même, la mise en œuvre également à quelques petites différences [4] près :

- les arguments de la fonction sont dans l'ordre inverse : c'est $f(y,t)$ (mais il y a une option permettant de les remettre dans l'ordre usuel) ;
- il faut donner en argument la fonction f, la condition initiale ainsi qu'une liste contentant tous les temps pour lesquels on veut calculer un point ;
- la fonction f a toujours seulement deux arguments, ainsi pour des équations d'ordre 2, y doit être donné sous forme d'un vecteur, *i.e.* une liste à deux éléments, la condition initiale également ;
- le résultat `sol` est donné sous la forme d'un tableau `numpy` dont il faut extraire la première colonne via la commande `sol[:,0]`.

Reprenons l'exemple du pendule :

```
1  import matplotlib.pyplot as plt
2  from scipy.integrate import odeint
3  from math import sin, pi
4
5  theta0 = pi/4
6  k = 0.1
7  y0 = [theta0, 0]  # Conditions initiales
8  temps = [i/10 for i in range(501)]
9
10 def F(Y, t):
11     return [Y[0], -sin(Y[0]) - k*Y[1]]
12
13 sol = odeint(F, y0, temps)
14
15 plt.scatter(temps, sol[:, 0])
16 plt.show()
```

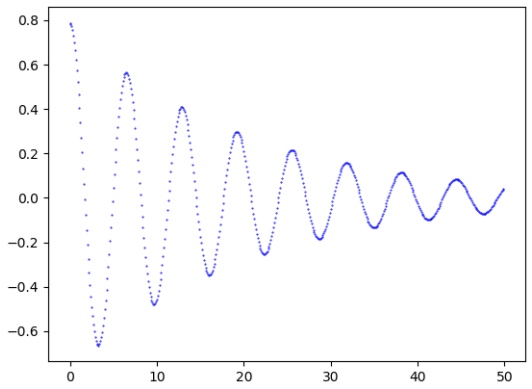


FIGURE 7 – Solution obtenue avec `odeint`.

3. Il existe aussi la fonction `solve_ivp` dans cette même bibliothèque qui est une version modernisée de `odeint`
4. À nouveau, il ne s'agit pas de les connaître, on vous donnera toujours la documentation nécessaire qu'il faudra lire attentivement pour vous adapter à la situation.

TD n° 8 : méthode d'Euler

Exercice 1. *Une autre écriture de la méthode d'Euler à l'ordre 1*
On souhaite déterminer une solution approchée d'une équation différentielle d'ordre 1 de la forme

$$y'(t) = f(t, y) \text{ vérifiant la condition initiale } y(t_0) = y_0.$$

Pour cela, écrire une fonction `euler` prenant en arguments :

- une fonction `f` définissant l'équation différentielle,
- un tuple `cond_init` composé de deux flottants représentant la condition initiale (t_0, y_0),
- un flottant `h` représentant le pas,
- un entier `n` représentant le nombre de points souhaités,

et qui renvoie deux listes de flottants correspondant respectivement aux abscisses et aux ordonnées des points de la solution numérique.

Exercice 2. *Chute avec frottements fluides*
On considère la chute verticale d'un objet de masse m lâché sans vitesse initiale. Il est soumis au champ de pesanteur (on prendra $g = 9{,}81\,\mathrm{m}\cdot\mathrm{s}^{-2}$) et subit également une force de frottements exercée par l'air proportionnelle à la vitesse d'un facteur k. D'après le principe fondamental de la dynamique et après avoir projeté sur l'axe vertical, on obtient que la vitesse v de l'objet vérifie l'équation différentielle

$$mg - kv = m\frac{\mathrm{d}v}{\mathrm{d}t}.$$

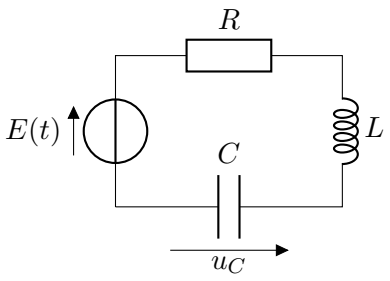
1. Mettre cette équation sous la forme $v' = f(t, v)$ en précisant la fonction f.
2. Définir en python cette fonction f. On définira m et k comme des variables globales dont vous choisirez des valeurs.
3. Écrire la commande permettant de récupérer dans deux variables `temps` et `vitesse` les abscisses et les ordonnées des points d'une solution numérique calculés grâce à la fonction de l'exercice précédent.
4. En reprenant les exemples du cours, écrire les commandes permettant de tracer le nuage de points formant cette solution numérique.

On peut également résoudre cette équation à la main : la solution est donnée par

$$\forall t \geqslant 0,\ v(t) = \frac{mg}{k}\left(1 - \mathrm{e}^{-\frac{k}{m}t}\right).$$

5. Définir une fonction `sol_exacte` correspondant à cette solution. On pourra importer la fonction `exp` du monde `math`.
6. En utilisant la fonction `plot` du module `matplotlib.pyplot`, qui s'utilise comme `scatter`, donner les commandes permettant de tracer cette solution exacte.

Exercice 3. *Circuit RLC série en régime sinusoïdal forcé*
On considère le circuit RLC série ci-contre avec $R = 100\,\Omega$, $L = 1\,\mathrm{H}$, $C = 1\,\mathrm{mF}$. On suppose que l'on a $E(t) = 10\sin(4t)$ pour $t \geqslant 0$.
On peut montrer que la tension u_C aux bornes du condensateur vérifie l'équation différentielle



$$LC\frac{\mathrm{d}^2u_C}{\mathrm{d}t^2} + RC\frac{\mathrm{d}u_C}{\mathrm{d}t} + u_C = E(t).$$

On suppose de plus que $u_C(0) = 0$ et $\dfrac{\mathrm{d}u_C}{\mathrm{d}t}(0) = 0$.

1. On pose $Y(t) = \begin{pmatrix} u_C(t) \\ \dfrac{\mathrm{d}u_C}{\mathrm{d}t}(t) \end{pmatrix}$. Exprimer $Y'(t)$ en fonction de $E(t)$ et des deux coordonnées de $Y(t)$.
2. En déduire une fonction python `f(t, Y)`, où `Y` est une liste à deux éléments, qui renvoie une liste à deux éléments de façon à correspondre à l'équation différentielle mise sous la forme $Y' = f(t, Y)$.
3. Adapter la fonction `euler2` définie dans le paragraphe 2.2 du cours afin qu'elle ait comme entête

```
def euler2(cond_init: list, h: float, n: int) -> (list, list):
```

où `cond_init` est une liste de deux flottants correspondants aux valeurs initiales $u_C(0)$ et $\dfrac{\mathrm{d}u_C}{\mathrm{d}t}(0)$, `h` est le pas et `n` le nombre de points souhaités. Elle utilisera bien sûr la fonction `f` définie précédemment et renverra deux listes de flottants donnant respectivement les abscisses (t) et les ordonnées (u_C) de la solution numérique.

Exercice 4. ★ *Problème de conservation d'énergie et méthode de Verlet (Centrale 2015)*
On considère un pendule non amorti, avec l'approximation des petits angles. L'angle θ avec la verticale vérifie l'équation différentielle $\ddot{\theta} = -\omega^2\theta$. Pour la suite, on notera plutôt y que θ et on s'intéressera donc à l'équation différentielle $(F) : y'' = -\omega^2 y$.

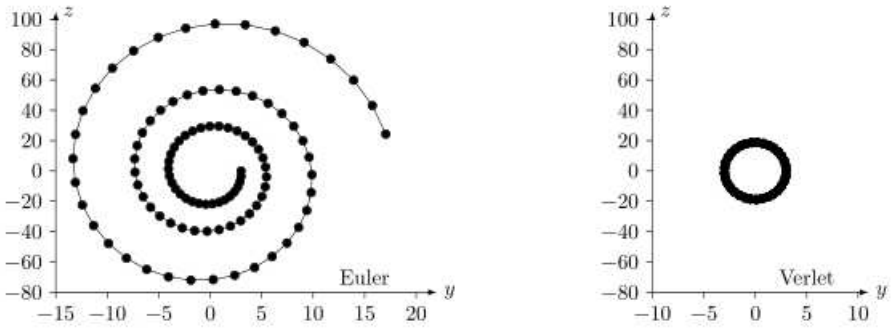
1. On pose $z = y'$. Exprimer y' et z' en fonction de y, z et ω.
2. Montrer que la quantité $E(t) = \dfrac{1}{2}\left(y'(t)\right)^2 + \dfrac{1}{2}\omega^2\left(y(t)\right)^2$ est en fait indépendante de t.

Comme dans le paragraphe 2.1 du cours, le schéma d'Euler mène au système $\forall i \in \mathbb{N}$, $\begin{cases} t_{i+1} = t_i + h \\ y_{i+1} = y_i + hz_i \\ z_{i+1} = z_i - h\omega^2 y_i \end{cases}$
où h est le pas de temps. Par ailleurs, pour tout $i \in \mathbb{N}$, on note E_i la valeur approchée de $E(t_i)$, *i.e.* la quantité définie par $E_i = \dfrac{1}{2}z_i^2 + \dfrac{1}{2}\omega^2 y_i^2$.

3. Montrer que, pour tout $i \in \mathbb{N}$, $E_{i+1} - E_i = h^2\omega^2 E_i$.
4. Qu'aurait donné un schéma numérique satisfaisant à la conservation de la quantité E ?

En 1967, le physicien Loup Verlet a proposé un schéma numérique d'intégration de certaines équations différentielles dont (F). En notant pour tout $i \in \mathbb{N}$, $f_i = -\omega^2 y_i$ (c'est $f(t_i, y_i)$ avec les notations du cours), ce schéma consiste à calculer pour tout $i \in \mathbb{N}$, $\begin{cases} t_{i+1} = t_i + h \\ y_{i+1} = y_i + hz_i + \frac{h^2}{2}f_i \\ z_{i+1} = z_i + \frac{h}{2}(f_i + f_{i+1}) \end{cases}$ où h est le pas de temps.

5. En s'inspirant de la fonction `euler2` du paragraphe 2.2, écrire une fonction `verlet` prenant en argument le pas h, des conditions initiales t_0, y_0 et z_0 et un entier n représentant le nombre de points souhaités. Cette fonction renverra les termes des suites (t_i), (y_i) et (z_i) calculés avec ce schéma de Verlet.
6. On a tracé ci-dessous les portraits de phase (c'est-à-dire l'ensemble des points de coordonnées (y_i, z_i)) pour chacun des deux schémas numériques étudiés avec les valeurs $t_0 = 0$, $y_0 = 3$, $z_0 = 0$, $\omega = 2\pi$ et $n = 100$.



Interpréter ces figures à l'aune de ce qui a été vu dans les questions précédentes.

7. On définit la quantité E_i comme pour le schéma d'Euler. Montrer qu'avec le schéma de Verlet, pour tout $i \in \mathbb{N}$, $E_{i+1} - E_i = o(h^3)$. Comparer ce résultat au portrait de phase tracé ci-dessus.