TP3 : Résolution approchée d'équations différentielles ordinaires d'ordre 1 et 2 - Méthode d'Euler

I. Équations différentielles d'ordre 1

I.1. Aspect théorique

- On appelle équation différentielle d'ordre 1 toute équation de la forme :

$$y' = f(t,y), \quad \text{c'est-à-dire :} \quad \forall t \in I,\ y'(t) = f\big(t, y(t)\big)$$

  × l'inconnue $y : I \to \mathbb{R}$ est une fonction définie et dérivable sur un intervalle I de $\mathbb{R}$,
  × $f : I \times \mathbb{R} \mapsto \mathbb{R}$ est une fonction définie sur $I \times \mathbb{R}$.
- On appelle problème de Cauchy en $(t_0, y_0) \in I \times \mathbb{R}$ le système :

$$\boxed{\begin{cases} y' &= f(t,y) \\ y(t_0) &= y_0 \end{cases}}$$

  Sous certaines conditions (par exemple le caractère $\mathcal{C}^1$ de f sur I), on peut démontrer que le problème de Cauchy possède une unique solution.

I.2. Aspect pratique

I.2.a) Résolution des équations différentielles linéaires d'ordre 1

Le théorème précédent (dit de Cauchy-Lipschitz) affirme que le problème de Cauchy admet une unique solution. Cependant, cette solution n'admet pas forcément une expression simple.

La forme de la fonction f permet parfois de pouvoir expliciter les solutions.

- On appelle équation différentielle linéaire d'ordre 1 toute équation de la forme :

$$y' + a(t)y \ = \ b(t)$$

  × l'inconnue $y : I \to \mathbb{R}$ est une fonction définie et dérivable sur un intervalle I de $\mathbb{R}$,
  × les fonctions $a : I \to \mathbb{R}$ et $b : I \to \mathbb{R}$ sont définies et continues sur l'intervalle I,
  × lorsque $b = 0$ (la fonction b est identiquement nulle), on dit que l'équation est homogène.
- Le problème de Cauchy $\begin{cases} y' &= b - ay \\ y(t_0) &= y_0 \end{cases}$ admet une unique solution y définie sur I dont l'expression est :

$$\boxed{\forall t \in I,\ y(t) \ = \ y_0 \ \exp\left(-\int_{t_0}^{t} a(s)\ ds\right)}$$

- La solution de l'équation générale (avec second membre) s'obtient, en vertu du principe de superposition des solutions, par l'ajout d'une solution particulière. Cette solution particulière est déterminée :
  × soit car il existe une solution évidente,
  (on est aussi parfois amené à chercher une solution particulière d'une certaine forme)
  × soit à l'aide de la méthode de variation de la constante.
  (on procède alors par analyse-synthèse)

Exemple *(Dynamique d'une population sans compétition pour la nourriture)*
En l'absence de prédateurs et avec une nourriture abondante, l'évolution d'une famille de lapins suit un modèle de Malthus :

$$\begin{cases} y' &= ry \\ y(0) &= 2 \end{cases} \qquad \text{où } r \in \mathbb{R}_+$$

► Résoudre cette équation.

On se place sur $I = [0, +\infty[$.

- Il s'agit d'une équation linéaire homogène d'ordre 1. Il existe donc $C \in \mathbb{R}$ tel que : $y : t \mapsto C\ \mathrm{e}^{rt}$.
- La condition initiale impose que : $y(0) \ = \ 2 \ = \ C$.
- Ce problème de Cauchy admet donc pour unique solution la fonction $y : I \to \mathbb{R}$ définie par : $\forall t \in I,\ y(t) = 2\ \mathrm{e}^{rt}$

► On souhaite maintenant obtenir la représentation graphique de cette fonction.
À l'aide de quelle fonction peut-on obtenir le tracé d'une courbe en **Python** ?
Rappeler brièvement son fonctionnement.

- La courbe représentative d'une fonction peut se tracer à l'aide de la fonction `plot` de la bibliothèque `matplotlib`.
- Le tracé d'une fonction se fait « point par point ». Ainsi, seul un nombre fini de points du graphe est représenté. Cependant, la puissance de calcul permet que ce nombre soit suffisamment grand pour fournir une très bonne approximation de la courbe souhaitée. Le nuage de points obtenu est alors complété en reliant les paires de points successifs, ce qui permet d'obtenir un tracé continu.

► Dans la même fenêtre graphique, représenter la solution du système de Malthus pour $r_1 = \ln(2)$ et $r_2 = 1$. La première courbe devra être représentée en rouge et la seconde en bleu. On effectuera le tracé sur l'intervalle $[0, 10]$.

```
import numpy as np
import matplotlib.pyplot as plt

r1 =         np.log(2)
r2 =         1

X =         np.linspace(0, 10)
Y1 =         2 * np.exp(r1 * X)
Y2 =         2 * np.exp(r2 * X)

plt.clf()
plt.plot(       X, Y1, color = 'red', label = 'Solution pour r = ln(2)')
plt.plot(       X, Y2, color = 'blue', label = 'Solution pour r = 1')

plt.xlabel('Temps')
plt.ylabel('Évolution de la population')
plt.title('Solutions du système de Malthus')
plt.legend()
plt.show()
```

I.2.b) Résolution approchée par la méthode d'Euler

Les équations différentielles n'ont pas toujours de solutions explicites. Cependant, en tout point, on peut connaître la valeur **approchée** de la solution à un problème de Cauchy.
Pour ce faire, on utilise des méthodes numériques telles que la **méthode d'Euler**.

Principe de la méthode d'Euler

On cherche ici à connaître une solution approchée d'un problème de Cauchy $\begin{cases} y' &= f(t,y) \\ y(t_0) &= y_0 \end{cases}$ sur un intervalle $I = [a,b]$ où : $b > a$.

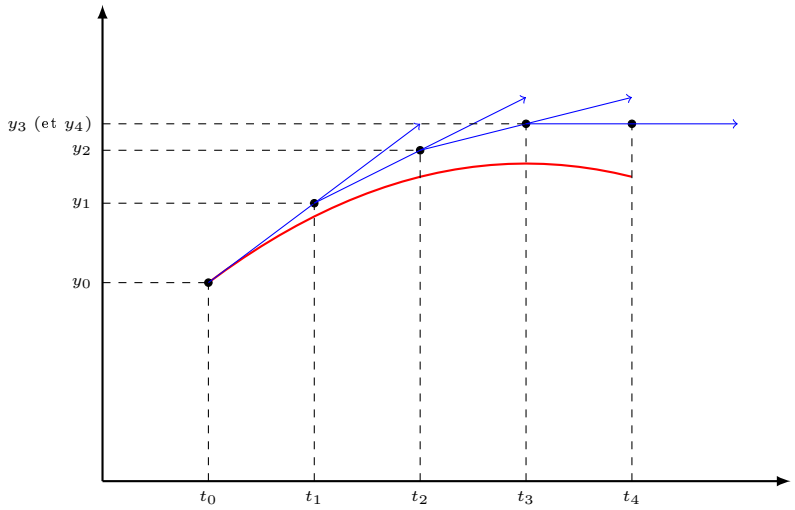
- On commence par découper l'intervalle $[a,b]$ à l'aide de $N+1$ points régulièrement espacés d'un pas de $h = \dfrac{b-a}{N}$.

$$t_0 = a, \quad t_1 = a+h, \quad t_2 = a+2h, \quad \ldots \quad , \quad t_k = a+kh, \quad \ldots \quad , \quad t_N = b$$

- On cherche alors à déterminer, pour tout $k \in [\![0, N]\!]$, une valeur y_k telle que $y_k \simeq y(t_k)$.
  (y_k est une valeur approchée de $y(t_k)$)

  Pour ce faire, on approche la fonction f par une constante sur chaque sous-intervalle de la subdivision.

  Approche naïve de la méthode d'Euler
  En partant de y_k (valeur approchée de $y(t_k)$), l'idée est de « suivre » la tangente au point $\big(t_k, y(t_k)\big)$ à la courbe représentative $\mathcal{C}_y$ de y, jusqu'au point d'abscisse t_{k+1}. Cela revient à considérer que la fonction $f\big(\cdot, y(\cdot)\big)$ est constante égale à $f(t_k, y_k)$ sur l'intervalle $[t_k, t_{k+1}]$. L'ordonnée obtenue est alors une valeur approchée de $y(t_{k+1})$, que l'on notera y_{k+1}.
  On peut illustrer cette idée par la figure ci-dessous.



  Plus précisément, on écrit :

$$\forall t \in [t_k, t_{k+1}], \ f(t, y(t)) \simeq f(t_k, y(t_k)) \simeq f(t_k, y_k)$$

  (cela correspond à l'approximation faite dans la méthode des rectangles)

- Les valeurs de t_k s'obtiennent en remarquant que :

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} y'(t)\, dt = \int_{t_k}^{t_{k+1}} f(t, y(t))\, dt$$

- Avec les approximations mentionnées, on est amené à poser :

$$y_{k+1} - y_k \;=\; \int_{t_k}^{t_{k+1}} f(t_k, y_k)\, dt \;=\; (t_{k+1} - t_k) \times f(t_k, y_k)$$

Autrement dit, on considère :

$$\forall k \in [\![0, N-1]\!], \quad \boxed{y_{k+1} \;=\; y_k \;+\; h \times f(t_k, y_k)}$$

► Écrire en **Python** la fonction `euler(f, y0, a, b, N)` qui renvoie le couple `(T, Y)` où `T` est la liste des réels t_k et `Y` est la liste des réels y_k.

```python
def euler(f, y0, a, b, N) :
    h = (b - a) / N
    # Attention : N intervalles, N+1 points
    T = np.linspace(a, b, N+1)
    Y = [0]*(N+1)
    Y[0] = y0
    for k in range(N) :
        Y[k+1] = Y[k] + h * f(T[k], Y[k])
    return (T, Y)
```

On pouvait aussi décider de mettre à jour la liste `T` au sein de la structure itérative. On obtient la fonction suivante :

```python
def euler(f, y0, a, b, N) :
    h = (b - a) / N
    T = [0]*(N+1)
    Y = [0]*(N+1)
    T[0] = a
    Y[0] = y0
    for k in range(N) :
        T[k+1] = T[k] + h
        Y[k+1] = Y[k] + h * f(T[k], Y[k])
    return (T, Y)
```

On souhaite maintenant tester la fonction `euler` sur le système de Malthus précédent.
Pour ce faire, il suffit essentiellement de coder la fonction $f : (t, y) \mapsto f(t, y)$.

► Écrire en **Python** la fonction f de l'exemple ***I.2.a)***. .

```python
r = np.log(2)

def f(t, y):
    return r * y
```

▶ Tester la fonction `euler` sur l'exemple.
On se placera sur l'intervalle $[0, 10]$ ($a = 0$ et $b = 10$) et on tracera sur le même graphique la solution obtenue avec une subdivision en 10 points, en 100 points et en 1000 points.

```
r = np.log(2)
a =         0
b =         10
N1 =         10
N2 =         100
N3 =         1000

X =         np.linspace(0, 10)
(T1, Y1) =          euler(f, 2, a, b, N1)
(T2, Y2) =          euler(f, 2, a, b, N2)
(T3, Y3) =          euler(f, 2, a, b, N3)

Y =         2 * np.exp(r * X)

plt.clf()
plt.plot(       X, Y, color = 'red', label = 'Solution exacte')
plt.plot(       T1, Y1, color = 'blue', label = 'Solution approchée 1')
plt.plot(       T2, Y2, color = 'blue', label = 'Solution approchée 2')
plt.plot(       T3, Y3, color = 'blue', label = 'Solution approchée 3')

plt.xlabel('Temps')
plt.ylabel('Évolution de la population')
plt.title('Solutions du système de Malthus')
plt.legend()
plt.show()
```

Remarque

- L'exemple précédent a un intérêt pratique : comme l'on sait déterminer de manière explicite la solution exacte, on peut la comparer à la méthode d'Euler pour se convaincre de son bon fonctionnement.
- Évidemment, cette utilisation est artificielle. La méthode d'Euler est utilisée pour trouver des solutions approchées à des équations différentielles plus compliquées.

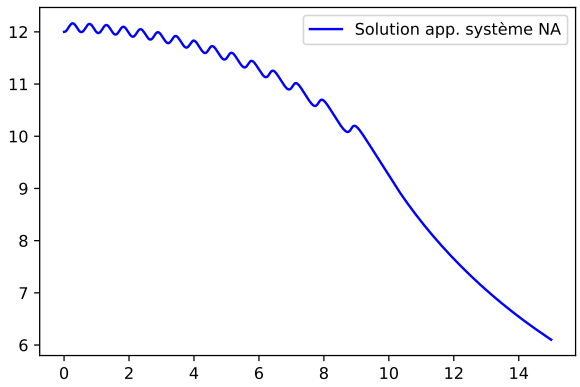
Exemple

L'équation différentielle : $y' = \sin(ty)$ est dite **non autonome**.
En effet, la fonction $f : (t, y) \mapsto f(t, y) = \sin(ty)$ dépend de t.

▶ Tester la fonction `euler` sur ce nouvel exemple.
On se placera sur l'intervalle $[0, 15]$ ($a = 0$ et $b = 15$) et on choisira $y_0 = 12$ et $N = 1000$.

```
def g(t, y) :
    return np.sin(t * y)
a =         0
b =         15
N =         1000
(T, Y) =          euler(g, 12, a, b, N)
plt.clf()
plt.plot(       T, Y, color = 'blue', label = 'Solution app. système NA')
```

On obtient le diagramme suivant :



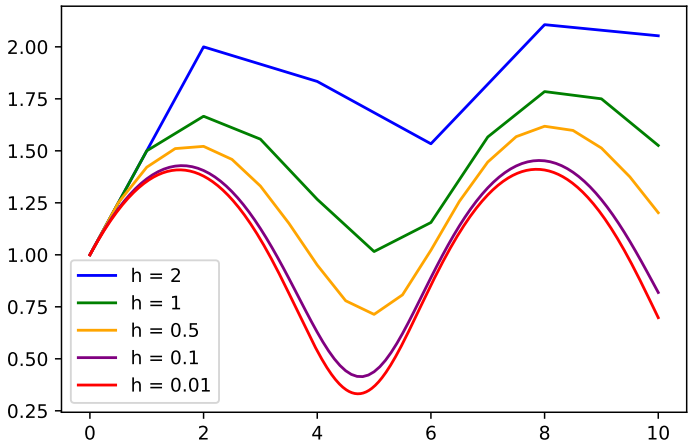
Exercice 1
On considère le problème de Cauchy suivant, définie sur l'intervalle $[0, 10]$:

$$\begin{cases} y' &= \dfrac{\cos(t)}{1+y^2} \\ y(0) &= 1 \end{cases}$$

Le graphique ci-dessous fait apparaitre les courbes obtenues en appliquant la méthode d'Euler, avec les pas suivants :

$$h = 2, \quad h = 1, \quad h = \frac{1}{2}, \quad h = \frac{1}{10}, \quad \text{et} \quad h = \frac{1}{100}$$

On observe bien la convergence de la ligne polygonale vers la courbe représentative de la solution.
Écrire un programme **Python**, utilisant la fonction `euler` et permettant d'obtenir le graphe ci-dessous.



II. Équations différentielles d'ordre 2

II.1. Aspect théorique

- On appelle équation différentielle d'ordre 2 toute équation de la forme :

$$y'' = f(t, y, y'), \quad \text{c'est-à-dire} \quad \forall t \in I,\ y''(t) = f\big(t, y(t), y'(t)\big)$$

  × l'inconnue $y : I \to \mathbb{R}$ est une fonction définie et deux fois dérivable sur un intervalle I de $\mathbb{R}$,
  × $f : I \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ est une fonction définie sur $I \times \mathbb{R} \times \mathbb{R}$.

- On appelle problème de Cauchy en $(t_0, y_0, \dot{y_0}) \in I \times \mathbb{R} \times \mathbb{R}$ le système :

$$\boxed{\begin{cases} y'' &= f(t, y, y') \\ y(t_0) &= y_0 \\ y'(t_0) &= \dot{y_0} \end{cases}}$$

  Sous certaines conditions (par exemple le caractère $\mathcal{C}^2$ de f), on peut démontrer que le problème de Cauchy possède une unique solution.

II.2. Aspect pratique

II.2.a) Résolution des équations différentielles linéaires d'ordre 2 à cœfficients constants

Le théorème précédent (dit de Cauchy-Lipschitz) affirme que le problème de Cauchy admet une unique solution. Cependant, cette solution n'admet pas forcément une expression simple.

La forme de la fonction f permet parfois de pouvoir expliciter les solutions.

- On appelle équation différentielle linéaire d'ordre 2 à cœfficients **constants** toute équation de la forme :

$$y'' + ay' + by \ = \ m$$

  × l'inconnue $y : I \to \mathbb{R}$ est une fonction définie et deux fois dérivable sur un intervalle I de $\mathbb{R}$,
  × a et b sont des réels,
  × $m : t \mapsto m(t)$ est une fonction continue sur I.

- Le problème de Cauchy $\begin{cases} y'' &= -ay' - by \\ y(t_0) &= y_0 \\ y'(t_0) &= \dot{y_0} \end{cases}$ admet une unique solution y sur I dont l'expression est donnée en fonction des solutions λ et μ de l'équation caractéristique $r^2 + ar + b = 0$.
  (cf cours)

- La solution de l'équation générale (avec second membre) s'obtient, en vertu du principe de superposition des solutions, par l'ajout d'une solution particulière. Cette solution particulière est déterminée :
  × soit car il existe une solution évidente,
  (on est aussi parfois amené à chercher une solution particulière d'une certaine forme)
  × soit à l'aide de la méthode de variation des constantes.
  (on procède alors par analyse-synthèse)

II.2.b) Résolution approchée à l'aide de la méthode d'Euler

La méthode précédente peut s'adapter au cas de l'ordre 2.

En effet, en notant $Z : t \mapsto \begin{pmatrix} y(t) \\ y'(t) \end{pmatrix}$ on obtient $Z' : t \mapsto \begin{pmatrix} y'(t) \\ y''(t) \end{pmatrix}$ et l'équation différentielle d'ordre 2 précédente peut s'écrire sous la forme :

$$\forall t \in I,\ \begin{pmatrix} y'(t) \\ y''(t) \end{pmatrix} \ = \ \begin{pmatrix} 0 & 1 \\ -b & -a \end{pmatrix} \times \begin{pmatrix} y(t) \\ y'(t) \end{pmatrix}$$

Autrement dit : $\boxed{Z' = A \times Z}$
On a alors de nouveau affaire à une équation différentielle d'ordre 1 !

L'idée est donc d'adapter la méthode d'Euler scalaire précédente à ce cas matriciel.
Commençons par les opérations sur les tableaux de la bibliothèque `numpy`.

▶ Quel est le résultat renvoyé par l'appel `(1,2) + (4,5)` ? Et `[1,2] + [4,5]` ?
Comment se nomme l'opérateur `+` dans ce cas ?

- On obient `(1,2,4,5)` et `[1,2,4,5]`.
- Cette opération se nomme la concaténation.

▶ Quel est le résultat de l'appel `2 * np.array([0,1]) + np.array([3,0])` ? Commenter.

- On obtient alors `np.array([3,2])`.
- Pour les objets de type `np.array`, il y a **surcharge** de l'opérateur `+` afin qu'il effectue la somme.

▶ Adapter la méthode d'Euler pour résoudre une équation du type $Z' = A \times Z$.
Il s'agit donc d'écrire en **Python** la fonction `euler_ordre2(f, y0, yp0, a, b, N)` qui renvoie le couple `(T, Z)` où `T` est la liste des points t_k et `Z` est la liste des vecteurs $\begin{pmatrix} y_k \\ \dot{y}_k \end{pmatrix}$.

- La fonction `euler_ordre2` est essentiellement la même que la fonction `euler`.
La seule différence réside dans le fait que `Z` contient des objets de type `array` et plus des flottants.

```
def euler_ordre2(f, y0, yp0, a, b, N):
    h =         (b - a) / N
    # Attention : N intervalles, N+1 points
    T =         np.linspace(a, b, N+1)
    # Il est important que les objet dans Z soient de type array
    Z = np.zeros(N+1, dtype = np.ndarray)
    Z[0] =          np.array([y0, yp0])
    for k in range(N):
        Z[k+1] =        Z[k] + h * f(T[k], Z[k])
    return (T, Z)
```

Comme pour la méthode d'Euler d'ordre 1, on peut aussi choisir de mettre `T` à jour au cours de la structure itérative. On obtient :

```
def euler_ordre2(f, y0, yp0, a, b, N):
    h =         (b - a) / N
    T =         [0]*(N+1)
    # Il est important que les objet dans Z soient de type array
    Z = np.zeros(N+1, dtype = np.ndarray)
    Z[0] =         np.array([y0, yp0])
    T[0] =         a
    for k in range(N):
        T[k+1] =         T[k] + h
        Z[k+1] =         Z[k] + h * f(T[k], Z[k])
    return (T, Z)
```

On souhaite maintenant tester cette méthode sur l'équation différentielle $y'' + y = 0$ avec les conditions initiales $y(0) = 0$ et $y'(0) = 1$, dont on sait que la solution est la fonction sinus.

► Quelle est la fonction $f : (t, Z) \mapsto f(t, Z)$ dans ce cas ?

- Ici, on a $y'' = -y$. Cette équation se réécrit :

$$\begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \times \begin{pmatrix} y \\ y' \end{pmatrix} \qquad \left(= \begin{pmatrix} y' \\ -y \end{pmatrix} \right)$$

- Il s'agit donc de coder la fonction $f : I \times \mathbb{R}^2 \to \mathbb{R}^2$ telle que :

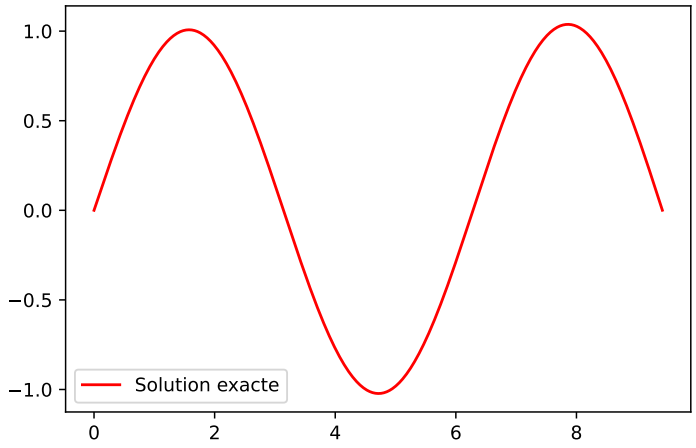
$$\begin{array}{ccccccc} f & : & I & \times & \mathbb{R}^2 & \to & \mathbb{R}^2 \\ & & (t & , & \begin{pmatrix} u \\ v \end{pmatrix} & \mapsto & \begin{pmatrix} v \\ -u \end{pmatrix} \end{array}$$

```
def f(t, Z) :
    return np.array([Z[1], -Z[0]])
```

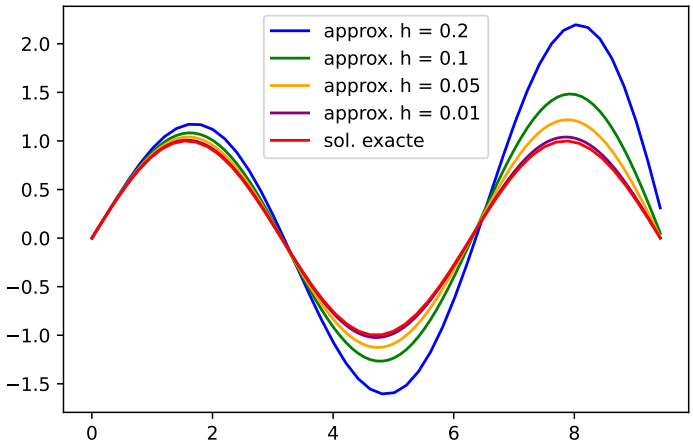
► En déduire l'appel nécessaire pour obtenir la solution approchée au système de Cauchy $\begin{cases} y'' + y = 0 \\ y(0) = 0 \\ y'(0) = 1 \end{cases}$.

On effectuera le tracé sur l'intervalle $[0, 3\pi]$ et on prendra $N = 100$.

```
a = 0
b = 3 * np.pi
N = 1000
y0 = 0
yp0 = 1
(T,Z) = euler_ordre2(f, y0, yp0, a, b, N)
Y = [vect[0] for vect in Z]
plt.plot(T, Y, color = 'red')
```

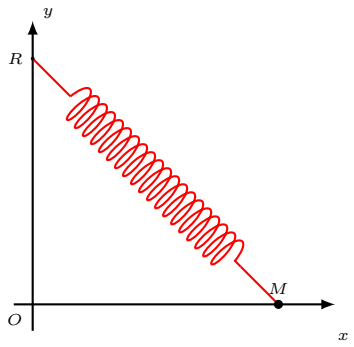


Remarque
Naturellement, plus le pas est petit, plus la solution est précise, comme on peut encore le voir sur la figure ci-dessous.



Exercice 2

On s'intéresse au système mécanique suivant : un point matériel M de masse m est fixé à l'extrémité d'un ressort de longueur à vide ℓ_0 et de constante de raideur k. La masse peut coulisser sans frottement sur une tige. On repère la position de la masse sur cette tige par l'abscisse x dont l'axe est confondu avec la tige, et dont l'origine O est située sur la même verticale que le point d'attache R fixe du ressort. On appelle d la distance OR.



On peut démontrer que le mouvement est régi par l'équation :

$$m\,x'' + k\,x' - \frac{k\,\ell_0\,x}{\sqrt{x^2+d^2}} \;=\; 0 \qquad (*)$$

1. Déterminer une fonction $f : \mathbb{R}^2 \to \mathbb{R}^2$ telle que l'équation $(*)$ se mette sous la forme $Y' = f(t, Y)$, avec $Y : t \mapsto \big(x(t), x'(t)\big)$.

2. Écrire une fonction `Euler`, qui prend en paramètres :

- une fonction `f`,
- des réels `a`, `b`, `x0` et `x0prime`,
- un réel `h` strictement positif,

qui renvoie deux listes `liste_t` et `liste_x`, et qui permet, en utilisant la méthode d'Euler de pas h, de résoudre de manière approchée l'équation $Y' = f(t, Y)$, sur l'intervalle $[a, b]$ avec les conditions initiales :

$$x(a) = x_0 \qquad \text{et} \qquad x'(a) = x'_0$$

3. On prend les valeurs numériques suivantes :

- $m = 300$ g,
- $k = 5$ N·m^{-1},
- $\ell_0 = 20$ cm,
- $d = 10$ cm,
- $x'(0) = 0$ m·s^{-1}.

En choisissant judicieusement différentes valeurs de $x(0)$, montrer qu'avec les valeurs ci-dessus, le système admet :

- un équilibre instable si $x(0) = 0$,
- un équilibre stable si $x(0) = \pm\sqrt{\ell_0^2 - d^2}$.