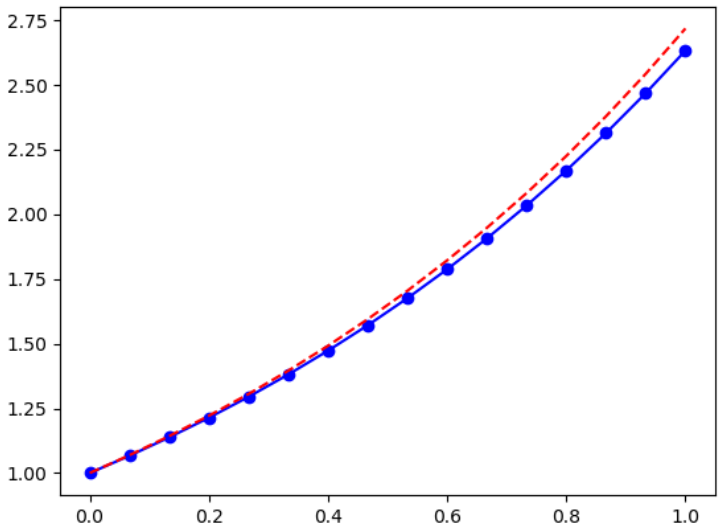
Corrigé du TP "Euler"

Méthode d'Euler

Exercice 3. — **(Pas trop dur pour commencer)**. Résolution du problème de Cauchy :

$$\forall\, t \in [0;\ 1]\,, \quad y'(t) = y(t) \quad \text{et} \quad y\,(0) = 1$$

```
from math import *
import matplotlib.pyplot as plt;
import numpy as np;

print('Choisissez un entier n')
N =int(input())

h =1/N # Définition du pas

y =1 # Initialisation de y (valeur de y_0)
t =0 # Initialisation de t (valeur de a)

ValT =[t] # Initialisation de la liste des abscisses
ValY =[y] # Initialisation de la liste des ordonnées

for k in range(N):
    t =t +h
    y =y +h * y # Approx affine: y(t+h) = y(t) + h * y'(t)
    ValT =ValT +[t]
    ValY =ValY +[y]

# Creation du graphique
plt.clf();
plt.plot(ValT, ValY, 'b-o', ValT, np.exp(ValT), 'r--');
plt.show();
```



Ci-contre, la ligne polygonale (en bleu) approchant la courbe représentative (en rouge) de la solution exacte (qui n'est autre que la fonction exponentielle).

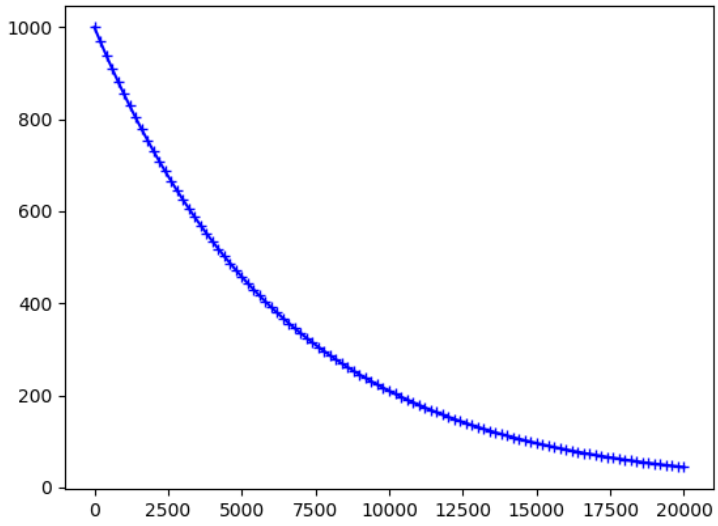
A noter que l'approximation est déjà ici "assez bonne", même pour un petit nombre d'intervalles ($n = 15$ sur cette illustration).

Exercice 4. — **(Décroissance radioactive)**. Il s'agit ici de résoudre le problème de Cauchy :

$$\forall\, t \in [0;\, T]\,, \quad \frac{dN\,(t)}{dt} + \frac{N\,(t)}{\tau} = 0 \quad \text{et} \quad N\,(0) = N_0 \ (\text{avec } N_0 \in \mathbb{R}^*).$$

```
from math import *
import matplotlib.pyplot as plt;
import numpy as np;

print('Choisissez un entier n')
N =int(input())

h =20000/N # Définition du pas

N0 =1000 # Nombre initial d'atomes de carbone 14
tau =6500

y =N0 # Initialisation de y (valeur de y_0)
t =0 # Initialisation de t (valeur de a)

ValT =[t] # Initialisation de la liste des abscisses
ValY =[y] # Initialisation de la liste des ordonnées

for k in range(N):
    t =t +h
    y =y - (h * y) /tau # Approx affine: y(t+h) = y(t) + h * y'(t)
    ValT =ValT +[t]
    ValY =ValY +[y]

# Creation du graphique
plt.clf();
plt.plot(ValT, ValY, 'b-+');
plt.show();
```

Ci-contre, la ligne polygonale (en bleu) approchant la solution exacte du problème de Cauchy (avec $n = 100$).



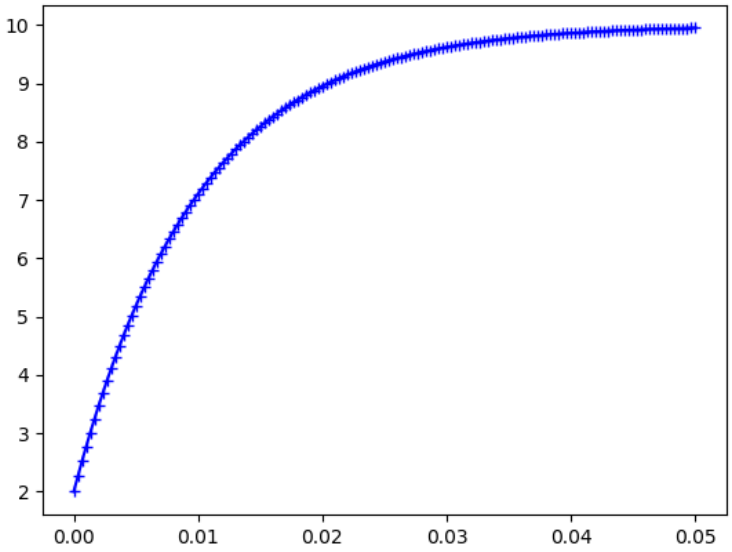
Exercice 5. — **(Evolution d'une tension)**. Adapter le programme de l'exercice précédent à l'étude de l'évolution de la tension $u(t)$ lors de la charge d'un condensateur à travers un résistor, sous une tension $E = 10\ V$. La tension $u(t)$ est dans ce cas donnée par :

$$\forall\, t \in [0;\ T]\,, \quad \frac{du\,(t)}{dt} + \frac{u\,(t)}{RC} = \frac{E}{RC}$$

On prendra : $u\,(0) = 2,0\ V$; $R = 1,0.10^4\Omega$ et $C = 1,0.10^{-6}\ F$.

```
print('Choisissez un entier n')
N =int(input())

h =0.05/N # Définition du pas

U0 =2
R =10000
C =0.000001
E =10

y =U0 # Initialisation de y (valeur de y_0)
t =0 # Initialisation de t (valeur de a)

ValT =[t] # Initialisation de la liste des abscisses
ValY =[y] # Initialisation de la liste des ordonnées

for k in range(N):
    t =t +h
    y =y +h * (E-y)/(R*C) #Approx affine: y(t+h) = y(t) + h * y'(t)
    ValT =ValT +[t]
    ValY =ValY +[y]

# Creation du graphique
plt.clf();
plt.plot(ValT, ValY, 'b-+');
plt.show();
```

Ci-contre, la ligne polygonale (en bleu) approchant la solution exacte du problème de Cauchy (avec $n = 150$).



Exercice 6. — **(Ski de vitesse)**. L'application de la relation fondamentale de la dynamique prenant en compte le poids du skieur, la force de réaction de la piste et le frottement de l'air conduit alors à l'équation différentielle suivante :

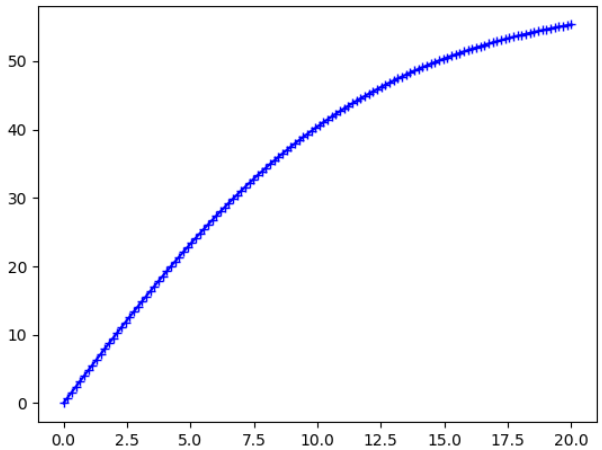
$$\frac{dv\,(t)}{dt} = g\ \sin\alpha - \frac{k}{m}\ v^2(t)$$

avec : $g = 9,81\text{m.s}^{-2}$; $\alpha = 30^0$; $m = 70\text{kg}$; $k = \dfrac{1}{2}\ \rho\ S\ Cx$ où : $\rho = 1,008\text{kg.m}^{-3}$, $S = 0,55\text{m}^2$, $Cx = 0,35$.

```
print('Choisissez un entier n')
N =int(input())

h =20/N # Définition du pas

g =9.81
alpha =pi/6
m =70
k =1.008 * 0.55 * 0.35 / 2

y =0 # Initialisation de y (valeur de y_0)
t =0 # Initialisation de t (valeur de a)

ValT =[t] # Initialisation de la liste des abscisses
ValY =[y] # Initialisation de la liste des ordonnées

for k in range(N):
    t =t +h
    y =y +h * ((9.81*sin(alpha))-(1.008 * 0.55 * 0.35 * 0.5*y*y/m))
    ValT =ValT +[t]
    ValY =ValY +[y]

print('La vitesse du skieur est ',y,'m/s au bout de 20s.')

# Creation du graphique
plt.clf();
plt.plot(ValT, ValY, 'b-+');
plt.show();
```

Ci-contre, la ligne polygonale (en bleu) approchant la solution exacte du problème de Cauchy (avec $n = 120$).

Au bout de 20s, la vitesse du skieur est environ 55m/s (approximativement 200km/h), ce qui est certes très impressionnant, mais cohérent avec les records enregistrés dans cette discipline.



Exercice 7. — (Généralisation).

Coder une fonction **euler(f,a,b,n,y0)** qui reçoit comme paramètres ceux que vous pouvez imaginer facilement d'après les paragraphes précédents, et qui construit la solution approchée de l'équation différentielle : $y'(t) = f(t, y(t))$.

```
def F(t,y):
    return (-y/6500) # Correspond à l'exo "décroissance  radioactive "

def EULER(a, b, N, y0):
    h =(b-a)/N # Définition du pas
    y =y0 #  Initialisation  de y (valeur de y_0)
    t =a #  Initialisation  de t (valeur de a)
    ValT =[t] #  Initialisation  de la  liste  des  abscisses
    ValY =[y] #  Initialisation  de la  liste  des ordonnées
    for k in range(N):
        t =t +h
        y =y +h*F(t,y) # Approx affine: y(t+h) = y(t) + h * y'(t)
        ValT =ValT +[t]
        ValY =ValY +[y]
    plt.clf();
    plt.plot(ValT, ValY, 'b-+');
    plt.show();
```

Troisième Partie — Euler (en dimension 2)

Système de Lotka-Volterra (prédateurs-proies)

Le modèle concerne deux populations dont les effectifs au temps t sont respectivement notés $x(t)$ et $y(t)$, la seconde (les prédateurs) se nourrissant de la première (les proies). On fait les hypothèses suivantes (forcément simplificatrices !) :

- Les proies $x(t)$ disposent de nourriture en quantité illimitée, seuls les prédateurs $y(t)$ s'opposent à leur croissance et en l'absence de prédateurs la population des proies a une croissance exponentielle.
- Le nombre de prédateurs est limité par la quantité de proies dont ils disposent pour se nourrir et en l'absence de proies, la population des prédateurs a une décroissance exponentielle.
- Le nombre de rencontres entre proies et prédateurs et à la fois proportionnel à $x(t)$ et $y(t)$, donc proportionnel au produit $x(t)y(t)$.
- Le taux de disparition des proies ainsi que le taux de croissance des prédateurs dûs à ces rencontres sont l'un et l'autre proportionnels au nombre de rencontres entres les deux populations.

Ce qui conduit au modèle suivant :

$$\textbf{(P)}\ :\quad \forall\, t\in[0;\ T]\,,\quad \begin{cases} x'(t) = \alpha_1 x\,(t) - \beta_1 x(t)y(t) \\ \\ y'(t) = -\alpha_2 y(t) + \beta_2 x(t)y(t) \end{cases} \quad \text{et} \quad \begin{cases} x\,(0) = x_0 \\ \\ y\,(0) = y_0 \end{cases} \quad \text{(populations initiales)}$$

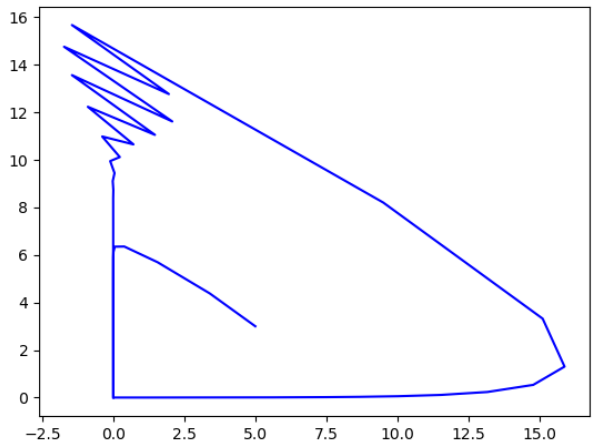
où $\alpha_1 > 0$ est le taux de natalité (naturel) des proies, $\alpha_2 > 0$ le taux de mortalité (naturel) des prédateurs, $\beta_1 > 0$ et $\beta_2 > 0$ des coefficients d'interaction entre les deux populations.

Exercice 8. — Résoudre le problème (P) en prenant comme valeurs numériques :

- $\alpha_1 = 0,8$; $\alpha_2 = 0,2$; $\beta = 0,8$; $\beta_2 = 0,5$;
- T entre 10 et 100 (par exemple) ;
- $x_0 = 5$ et $y_0 = 3$, en ayant présent à l'esprit que ces deux valeurs ne correspondent pas exactement aux populations initiales, mais donnent plutôt le ratio initial entre nombre de proies et nombre de prédateurs. Rien ne vous empêche de prendre 5000 et 3000 à la place de 5 et 3... sauf la capacité de calcul de l'ordinateur, et la taille de l'erreur (incomparablement plus importante dans ce cas).

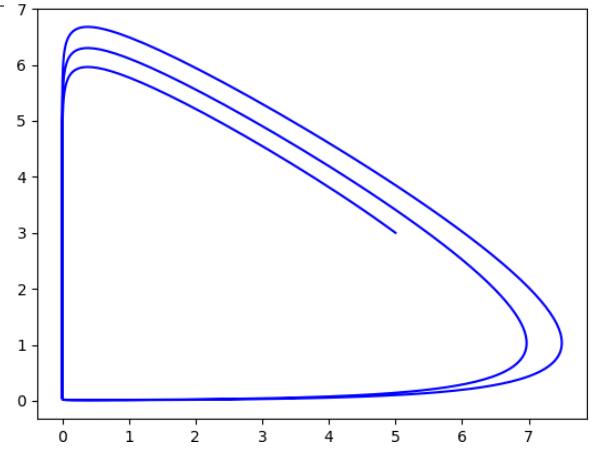
```
A1 =0.8
A2 =0.2
B1 =0.8
B2 =0.5

print('Valeur de N')
N =int(input())

h =(100)/N # Définition du pas
x =5 # Initialisation de x (valeur de x_0)
y =3 # Initialisation de y (valeur de y_0)
t =0 # Initialisation de t (valeur de a)
ValX =[x] # Initialisation de la liste des "proies"
ValY =[y] # Initialisation de la liste des "prédateurs"
for k in range(N):
    t =t +h
    c =x
    x =x +h*(-B1*x*y+A1*x)
    y =y +h*(B2*c*y-A2*y)
    ValY =ValY +[y]
    ValX =ValX +[x]

# Creation du graphique
plt.clf();
plt.plot(ValX, ValY, 'b-');
plt.show()
```

Ci-contre, la ligne polygonale (en bleu) représentant l'évolution du nombre de prédateurs en fonction du nombre de proies ; l'originalité de cet exemple est que l'on ne représente donc plus directement des quantités en fonction du temps.



Lorsque n est "petit", les erreurs d'approximation sont assez importantes, et sont assez éloignées de la réalité ("l'évolution est cyclique"). Ce problème peut être résolu en prenant de plus grandes valeurs de n, et en demandant donc davantage de calculs à la machine ($n = 500$ pour le graphe ci-contre).

Evolution des populations prédateurs/proies sur 100 ans ($n = 5000$).



Evolution des populations prédateurs/proies sur 100 ans ($n = 60000$).