(a) Il ne s'agit pas d'un système différentiel mais de deux équations différentielles indépendantes l'une de l'autre. On les résout donc séparément.

❧ On les traite comme des équations du second ordre en les écrivant sous forme résoluble (ce qui nous donne l'expression des fonctions `f` et `g` à définir pour appliquer `odeint`).

$$\frac{d}{dt}\begin{pmatrix} x(t) \\ x'(t) \end{pmatrix} = \begin{pmatrix} x'(t) \\ -\alpha x'(t) \end{pmatrix} \qquad \frac{d}{dt}\begin{pmatrix} y(t) \\ y'(t) \end{pmatrix} = \begin{pmatrix} y'(t) \\ -\alpha y'(t) - 1 \end{pmatrix}$$

REMARQUE.— Bien entendu, si on devait mener les calculs à la main, on traiterait ces équations comme des équations du *premier ordre* d'inconnues $x'(t)$ et $y'(t)$.

❧ Pour une résolution numérique, tous les paramètres doivent être choisis : pour le moment, la constante de temps α, la durée d'étude T_{max} et la vitesse initiale v sont fixés arbitrairement.

```
import numpy as np
from scipy.integrate import odeint

a = 0.5
T_max, v = 15, 3.5

def f(X, t):
   x, x_prime = X[0], X[1]
   x_seconde = -a*x_prime
   return np.array([x_prime, x_seconde])

def g(Y,t):
   y, y_prime = Y[0], Y[1]
   y_seconde = -a*y_prime - 1
   return np.array([y_prime, y_seconde])

T = np.arange(0, T_max, 0.01)
CI = np.array([0, v])

X = odeint(f, CI, T)
Y = odeint(g, CI, T)
```

❧ On rappelle la manière dont est défini le tableau retourné par la fonction `odeint` :

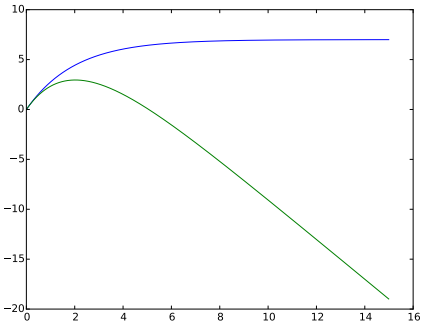
- la valeur `T[0]` est l'instant initial t_0 ;
- le tableau `CI` contient $x(t_0)$ et $x'(t_0)$;
- le résultat `X` est un tableau de n lignes et 2 colonnes où n est égal à `T` ;
- pour tout $0 \leqslant k < n$, les flottants `X[k,0]` et `X[k,1]` sont des valeurs approchées de $x(t_k)$ et $x'(t_k)$ où t_k est égal à `T[k]`.

❧ Dans un premier temps, on peut superposer les graphes de x et de y en fonction du temps.

```
import matplotlib.pyplot as plt

plt.plot(T, X[:,0])
plt.plot(T, Y[:,0])
```

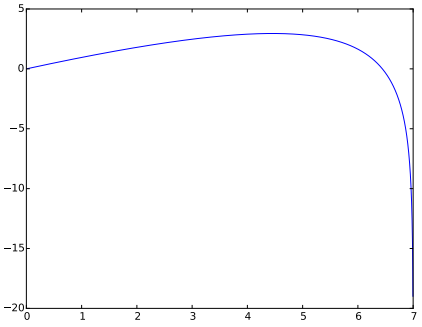
On constate que la fonction x tend vers une limite finie, tandis que la fonction y admet une asymptote oblique.



❧ On peut aussi tracer le support de l'arc paramétré $\big(x(t), y(t)\big)_{t\in T}$.

```
plt.figure()
plt.plot(X[:,0], Y[:,0])
```

On retrouve, sous une forme différente, les propriétés précédentes : la trajectoire admet une asymptote verticale (limite finie pour $x(t)$, limite infinie pour $y(t)$) mais on perd une information : on ne voit plus que $y(t) = \mathcal{O}(t)$ lorsque t tend vers $+\infty$.



❧ Pour tracer les trajectoires qui correspondent à plusieurs conditions initiales, il faut répéter les opérations effectuées précédemment au sein d'une boucle *for*.

```
plt.figure()
for v in np.arange(0.2, 5.2, 0.2):
   CI = np.array([0, v])
   X = odeint(f, CI, T)
   Y = odeint(g, CI, T)
   plt.plot(X[:,0], Y[:,0])
```