I Equations différentielles de degré 1

Le théorème de Cauchy-Lipschitz assure sous des conditions raisonnables sur F qu'il existe une unique application y de classe $\mathscr{C}^1$ sur $[a;b]$ telle que : $\left\{ \begin{array}{rcl} y(a) & = & y_0 \\ y'(t) & = & F(t,y(t)) \end{array} \right.$

Nous utiliserons F de classe au moins $\mathscr{C}^1$ sur $[a;b]$ donc y sera au moins $\mathscr{C}^2$ sur $[a;b]$.

La théorie nous dit qu'une solution existe, mais pas comment la trouver !
L'objectif des schémas numériques est d'obtenir une approximation de cette solution, par la construction d'un certain nombre de points sur l'intervalle $[a;b]$.

Au programme, le **schéma d'Euler explicite** :

— On choisit le nombre de points à construire : n

— Sur l'intervalle $[a;b]$, on construit donc n nouveaux points à partir du point $x=a$ (déjà connu). Ils seront donc espacés d'un pas fixe : $h = \dfrac{b-a}{n}$.
On aura donc $t_k = a + k \times h$, où k varie entre 0 et n.

— On va calculer pas à pas une approximation y_k de $y(t_k)$.

Pour cela on utilise :
$\displaystyle\int_{t_k}^{t_{k+1}} y'(t)dt =$

Si l'intervalle $[t_k;t_{k+1}]$ est suffisamment petit, on peut considérer $F(t,y(t))$ comme constante, égale à $F(t_k,y(t_k))$, approximée par $F(t_k,y_k)$.

Donc les approximations sont calculées de proche en proche par la formule de récurrence :

$$y_{k+1} = y_k + h \times F(t_k,y_k)$$

(et $y_0 = a$, seule valeur exacte).

Ecrivons ensemble la fonction `Eulerexplicite(F,a,b,y0,n)` :

Commençons par quelques exemples très simples

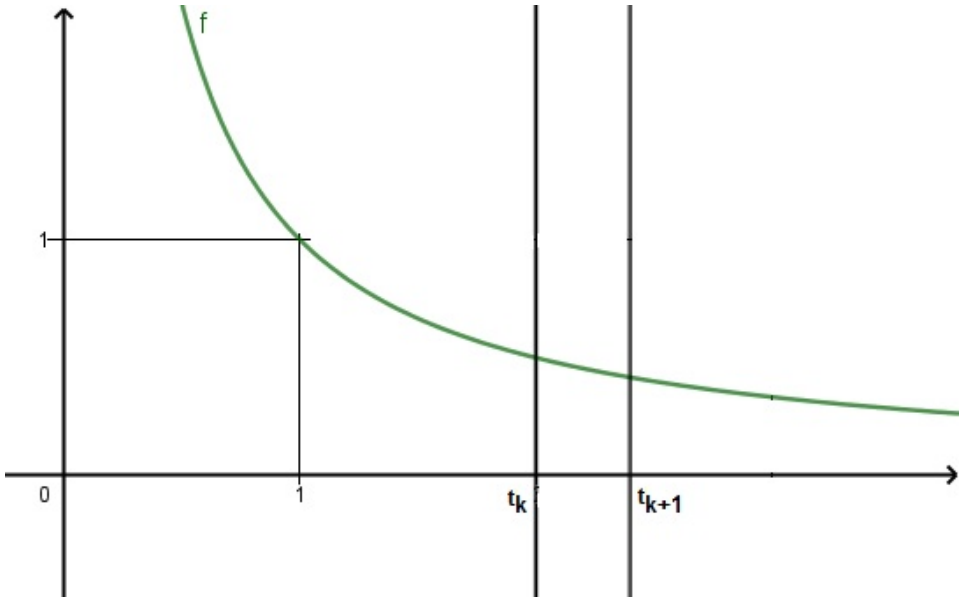
exemple 0 : la fonction logarithme népérien a été définie en terminale comme la fonction qui vaut 0 en 1 et dont la dérivée est $x \to \frac{1}{x}$ donc : $\begin{cases} y(...) &= ... \\ y'(t) &= ... \end{cases}$ On a donc ici $F(t,y) = ...$

```
import matplotlib.pyplot as pl
...

def F(t,y):
    return ...

T,Y =  Eulerexplicite(F,1,5,0,10) #calcul d'une solution sur [1;5]
pl.plot(T,Y )
pl.plot(T,np.log(T))
pl.grid()  #cree un quadrillage

pl.axhline(color='black') #affichage de l'axe des x
pl.axvline(color='black') #affichage de l'axe des y
pl.xlim(-1, 6)
pl.show()
```



Sur la représentation graphique de la fonction $x \to \frac{1}{x}$ ci dessus, placer h (le pas), $y(t_k)$, $y(t_{k+1})$ et y_{k+1}.
La méthode d'Euler correspond à la méthode dite « des rectangles »d'approximation d'une intégrale.

quantification de l'erreur :
On appelle note : $e_k = y(t_{k+1}) - y_{k+1}$ l'erreur entre la vraie valeur en t_{k+1} et l'estimation (oui, c'est décalé).
L'erreur de consistance pour un pas de h, est $e(h) = \sum_{k=0}^{n-1} |e_k|$: la somme de toutes les erreurs cumulées.

Or si y est de classe $\mathscr{C}^2$, $e_k = y(t_k + h) - (y_k + hy'(t_k)) = \frac{1}{2}h^2 y''(t_k) + o(h^2)$ (formule de Taylor).
Puisque y'' est continue sur un fermé $[a;b]$, $|y''(t_k)|$ est majoré par une valeur M.
(Dans notre exemple, $|y''(t)| = |-\frac{1}{t^2}| \leqslant 1$ sur $[1;5]$ donc $M = 1$.)
Donc $e(h) \leqslant (\frac{1}{2}Mh^2 + o(h^2)) \times n$.
Or $h = \frac{b-a}{n}$ donc $e(h) \leqslant \frac{1}{2}Mh^2 \times \frac{b-a}{h} = O(h^1)$.
La méthode d'Euler est donc une méthode dite d'ordre 1 : La somme des erreurs cumulées reste contrôlée par h (le pas).

remarque : dans le cas général, où F(y,t) dépend effectivement de y et pas seulement de t comme dans cet exemple, l'erreur de consistance sous estime l'erreur réellement produite. En effet, dans notre estimation y_{k+1}, on n'utilise pas la vraie valeur $y'(t_k) = F(y(t_k),t_k)$ mais son approximation $F(y_k,t_k)$.

exemple 0 bis : la fonction exponentielle a été définie en terminale comme la fonction qui vaut 1 en 0 et qui est sa propre dérivée donc : $\begin{cases} y(0) &= 1 \\ y'(t) &= y(t) \end{cases}$ On a donc ici $F(t,y) = y$.

Construisons sa courbe sur $[0;1]$

```
import matplotlib.pyplot as pl
...
def F1(t,y):
    return y

T,Y = Eulerexplicite(F1,0,1,1,4)
pl.plot(T,Y,color='red')
T,Y = Eulerexplicite(F1,0,1,1,10)
pl.plot(T,Y)
#pl.plot(T,np.exp(T))
pl.grid() #cree un quadrillage
pl.axhline(color='black') #affichage de l'axe des x
pl.axvline(color='black') #affichage de l'axe des y
pl.xlim(-0.2,1.2)
pl.show()
```

remarque : On a donc $h = \dfrac{1}{n}$.
L'approximation pour $e = exp(1)$ est donc $(1+\frac{1}{n})^n$.
La véritable erreur est donc $E(\frac{1}{n}) = e - (1+\frac{1}{n})^n = \dfrac{e}{2n} + o(\frac{1}{n})$.
Vous pouvez le démontrer en utilisant les développements limités...

exemple 1 : Le modèle de Verhulst de modélisation de la dynamique des populations. $y(t)$ désigne la taille de la population à l'instant t.

Elle est solution de l'équation différentielle : $\dfrac{dy}{dt} = y \times (n(y) - m(y))$ où $n(y)$ désigne le taux de natalité décroissant en y et $m(y)$ désigne le taux de mortalité, croissant en y (effet de surpopulation).

En supposant que ce sont des fonctions affines, on obtient :
$\dfrac{dy}{dt} = y \times ((a - by))$ a et b étant des réels positifs.
$\dfrac{dy}{dt} = ay \times (1 - \dfrac{y}{K})$ avec $K = \dfrac{a}{b}$ capacité d'accueil (car si $y > K$ alors y est décroissante et si $y < K$ alors y est croissante) et a le taux de croissance intrinsèque de la population.
Cette modélisation s'applique aux évolutions auto-catalytiques, comme les cellules tumorales dont l'accroissement est proportionnel au nombres de cellules touchées et au nombre de cellules saines.

Prenons l'exemple du Parc Kruger : L'éléphant africain de la savane se comptait par millions dans la savane africaine avant qu'il ne soit décimé, durant des siècles, par des chasseurs. En 1905, il ne restait que 10 individus en Afrique du Sud, et il fut décidé la création d'un parc naturel : le parc Kruger. Au $20^{\text{ième}}$ siècle, grâce à des mesures de protections strictes des animaux et de leur milieu, la population d'éléphants a cru de manière naturelle : lentement jusque dans les années 30 (on compte alors 403 individus), puis rapidement jusqu'aux années 60. C'est alors qu'on observa un ralentissement du taux de croissance et, en même temps, un début de dégradation par les éléphants d'autres espèces de l'écosystème (comme les baobabs par exemple). On dénombrait alors 7500 éléphants.

Le modèle de Verhulst modélisant cette évolution est : $y_0 = 10$ et $\dfrac{dy}{dt} = 0{,}15y \times \left(1 - \dfrac{y}{7500}\right)$

```
def F(t,y):
    return 0.15*y*(1-y/7500)

T,Y = Eulerexplicite(F,0,100,10,100)
pl.plot(T,Y)
pl.grid() #cree un quadrillage
#pl.ylim(0,1.5)
pl.axhline(color='black') #affichage de l'axe des x
pl.axvline(color='black') #affichage de l'axe des y
pl.show()
```

TD méthode d'Euler

exemple 2 : chute libre avec frottements proportionnels au carré de la vitesse

Nous supposerons, qu'exaspérés, vous décidez de jeter un de vos professeur par la fenêtre afin d'en étudier la chute libre.

On émet l'hypothèse raisonnable que ce corps est soumis à son poids et à une force de frottement (résistance de l'air) proportionnelle au carré de la vitesse $\overrightarrow{F} = -k||\overrightarrow{v}||\overrightarrow{v}$.

On considérera l'axe des z dirigé vers le haut.

On distinguera une phase ascendante (vous étiez particulièrement énervé, cela a décuplé votre force, vous avez réussi lancer votre prof vers le haut...) puis une phase descendante au mouvement.

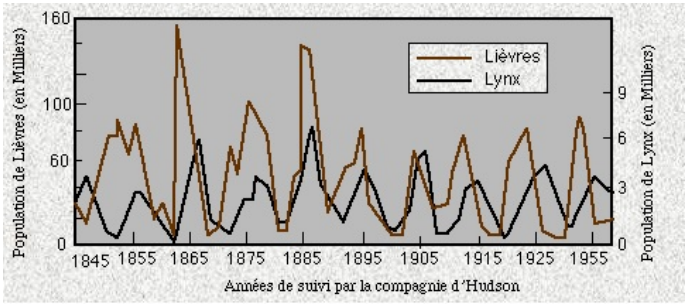
On notera $v(t)$ la vitesse algébrique.

1. A l'aide du principe fondamental de la dynamique, déterminer la fonction f telle que la vitesse algébrique vérifie
$$\frac{dv}{dt} = f(v)$$
La fonction f n'étant **pas linéaire**, la méthode du cours de mathématiques de résolution d'équations différentielles linéaires ne peut s'appliquer. Nous allons donc résoudre numériquement cette équation avec la méthode d'Euler.
2. Afin de préciser la valeur des paramètres, vous observez que le corps d'une masse de 80 kg, atteint une vitesse limite de 69.5 mètres par secondes. Vous savez que $g = 9.8m.s^{-2}$. justifier que $k = 0{,}16kg.m^{-1}$.
3. En Python, écrire une fonction $f(v)$ qui renvoie l'expression de la fonction f du problème considéré (v pouvant être positive ou négative)
4. En Python, écrire une fonction $Euler(f,v0,T,N)$ qui prend en argument la fonction f, la vitesse initiale $v0$, la durée de l'expérience T (nombre entier de seconde) et le nombre N de points souhaités. Elle renvoie les N valeurs de t et les N valeurs de v obtenues grâce à la méthode d'Euler (Vous pouvez inclure les valeurs initiales).
5. Représenter à l'aide de la méthode d'Euler la courbe de la vitesse pendant les 10 premières secondes. On prendra pour vitesse initiale zéro, puis $4m.s^{-1}$.
6. Améliorer votre fonction $Euler$ en une fonction $Euler(f,v0,z0,T,N)$ qui prend en plus en argument la position initiale $z0$ et renvoie en plus les N valeurs de z.
7. Tracer sur le même graphe les fonctions $v(t)$ et $z(t)$ pour vitesse initiale $4m.s^{-1}$ et une position initiale $z0 = 0$ (on pourra se limiter a $T = 2s$).
8. Un théoricien a montré que l'altitude maximale (pour $v0 > 0$ et $z0 = 0$) atteinte vérifie l'expression :
$$z_{max} = \frac{m}{2k}\ln(1 + \frac{v_0^2}{v_l^2})$$
Vérifier ce résultat à l'aide d'une fonction maxi(v0) que vous aurez écrite donnant le max de $z(t)$.

II généralisation à des systèmes différentiels de degré 1 (s'il ne reste qu'une heure, passez directement au III)

Citons Wikipedia :

En mathématiques, les équations de Lotka-Volterra, que l'on désigne aussi sous le terme de « modèle proie-prédateur », sont un couple d'équations différentielles non linéaires du premier ordre, et sont couramment utilisées pour décrire la dynamique de systèmes biologiques dans lesquels un prédateur et sa proie interagissent. Elles ont été proposées indépendamment par Alfred James Lotka en 1925 et Vito Volterra en 1926. Ce système d'équations est classiquement utilisé comme modèle pour la dynamique du lynx et du lièvre des neiges, pour laquelle de nombreuses données de terrain ont été collectées sur les populations des deux espèces par la Compagnie de la baie d'Hudson au XIXe siècle.



En notant $x(t)$ et $y(t)$ le nombre respectif de lynx et de lièvres, on peut modéliser la situation de la manière suivante :

$$\begin{cases} \dfrac{x'(t)}{x(t)} &= -M + Ny(t) \\ \dfrac{y'(t)}{y(t)} &= N' - M'x(t) \end{cases} \text{ donc } \begin{cases} x'(t) &= -x(t)\left(M - Ny(t)\right) \\ y'(t) &= y(t)\left(N' - M'x(t)\right) \end{cases}$$

M est le taux de mortalité des lynx (uniquement du à la vieillesse) alors que leur taux de natalité est proportionnel au nombre de lièvres (source de nourriture).
N' est le taux de natalité des lièvres alors que leur taux de mortalité est proportionnel au nombre de lynx.

L'observation a permis sur le terrain a permis d'obtenir : M = 0,3 et N = 0,002 pour le lynx
M' = 0,01 et N' = 0,5 pour le lièvre

$$\text{donc } \begin{cases} x'(t) &= -x(t)\left(0{,}3 - 0{,}002y(t)\right) \\ y'(t) &= y(t)\left(0{,}5 - 0{,}01x(t)\right) \end{cases}$$

1. Représenter graphiquement l'évolution de ces populations sur 100 ans avec comme populations initiales 50 lynx et 200 lièvres.
2. Si la chasse aux lynx était ré-autorisée et que leur mortalité passait à M = 0,6 , comment cela impacterait-il la dynamique des populations ? et si le réchauffement climatique faisait augmenter le taux de natalité des lièvres à N' = 0,8 ? ou au contraire si leur mortalité (à cause de la réintroduction de loups) passait à M' = 0,025 ?

III généralisation à des équations différentielles de degré supérieur

L'étude du pendule simple non amorti conduit à une équation différentielle d'ordre 2 de la forme $\dfrac{d^2\theta}{dt^2} + \omega^2 \sin\theta = 0$
On posera ici $\omega = 1$ et on supposera que $\theta(0) = 0$ et $\theta'(0) = 1$

1. En posant $x(t) = \theta(t)$ et $y(t) = \theta'(t)$, montrer que l'équation peut s'écrire sous la forme d'un système $\begin{cases} x'(t) &= y(t) \\ y'(t) &= -\sin(x(t)) \end{cases}$
2. Résoudre numériquement ce problème sur $[0, 6\pi]$.
3. Tracer l'évolution de $\theta(t)$ au cours du temps.
4. Tracer le portrait de phase, c'est-à-dire l'ensemble des points de coordonnées $(\theta(t), \theta'(t))$ pour $t \in [0, 4\pi]$.

remarque : En fait, le module scipy de Python propose la fonction odeint() qui vous permet de résoudre numériquement des équations différentielles. La syntaxe est

```
from scipy.integrate import odeint
v=odeint(f,v0,T)
```

où f est une fonction (éventuellement vectorielle) dépendant de v et t
où $v0$ est la condition initiale (éventuellement vectorielle)
où T est un tableau des valeurs de t considérées.

En fait, nous aurions pu écrire en début de TD une fonction $Eulerexplicitevectorielle(F,a,b,X0,n)$ prenant en paramètres une fonction vectorielle F et un tableau de conditions initiales $X0$ qu'on se serait contenté d'appeler, y compris dans les parties II et III, avec les fonctions F adaptées.
Vous pouvez donc refaire le TP, mais ça sera cette fois plus rapide... (pensez à importer numpy !)
Maintenant, vous avez vraiment envie de jeter un prof par la fenêtre ...