1 Premiers exemples

Cette section constitue le cours d'informatique sur le équations différentielles. Testez les exemples. Si tout ne se passe pas comme prévu, demandez de l'aide et ne passez surtout pas à la question suivante : chaque question est un prérequis pour la suite.

I) **Quelques modules utiles**

Importer les modules suivants :

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

Ces modules seront utiles dans toute la suite.

1.1 Une équation d'ordre 1

Nous allons étudier l'équation différentielle (E_1) :

(E_1) $\qquad y' = 2xy^2$ avec la condition initiale $y(0) = 0.2$ sur le segment $I = [0, 2]$

II) **Création d'une subdivision**

Nous allons créer une subdivision équirépartie du segment $[0, 2]$. Le but est de créer le tableau de type `numpy.ndarray` :

```
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,
        1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ])
```

`numpy` offre deux solutions :

- La fonction `np.linspace` dont la syntaxe est `np.linspace (`*début inclus*`,` *fin incluse*`,` *nombre de points*`)`
  Ici l'appel correspondant est `np.linspace (0, 2., 21)`.

  Attention : il y a 21 points, et pas 20.

  Remarque : si on avait importé le module `numpy` avec l'instruction `from numpy import *` au lieu de l'instruction `import numpy as np`, il n'y aurait pas besoin de préfixer : on écrirait `linspace` au lieu de `np.linspace`
- La fonction `np.arange` dont la syntaxe est `np.arange (`*début inclus*`,` *fin exclue*`,` *pas*`)`
  Ici l'appel correspondant est `np.arange (0., 2.1, 0.1)`

  Remarque : il est possible d'appeler la fonction `arange` avec un seul paramètre. Dans ce cas, ce paramètre est la fin exclue. Le début inclus est fixé à 0 et la pas est fixé à 1.
  Par exemple l'appel `arange (3.5)` renvoie `array([ 0., 1., 2., 3.])`

Affecter à la variable `x` cette subdvision.

Ensuite vérifier que vous avez bien ce que vous voulez :

```
>>> print x
             array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.,
                    1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.])
>>> type (x)
             <type 'numpy.ndarray'>
```

III) **Réécriture de l'équation**

Nous allons réécrire l'équation différentielle sous la forme $y' = f(y, x)$

Créer la fonction f appropriée :

```
def f (y, x) : return (2 * x * y**2)
```

Attention à l'ordre des variables !

IV) **Résolution approchée avec la méthode d'Euler**

Soit `n` tel que `x` soit la subdivision $[\mathtt{x}_0, \dots, \mathtt{x}_n]$ (sur notre exemple `n` vaut 20). Soient `a` et `b` tels que $\mathtt{I} = [\mathtt{a}, \mathtt{b}]$ (sur notre exemple $\mathtt{a} = 0.$ et $\mathtt{b} = 2.$)

Posons $\mathtt{h} = \dfrac{\mathtt{b} - \mathtt{a}}{\mathtt{n}}$ (c'est la distance entre deux valeurs consécutives de la subdivision). Sur notre exemple, $\mathtt{h} = 0.1$

Notons y la solution de (E_1). Nous ne cherchons pas à calculer explicitement y. Il s'agit de créer la liste $[\mathtt{y}_0, \dots, \mathtt{y}_n]$ où pour tout $\mathtt{i} \in [\![0, \mathtt{n}]\!]$, $\mathtt{y}_i$ est une valeur approchée de $y(\mathtt{x}_i)$.

Rappelons le schéma d'Euler (explicite) :

$$\begin{cases} y_0 = 0.2 & \text{(valeur initiale)} \\ \forall k \in [\![1, n]\!], y_k = y_{k-1} + hf(y_{k-1}, x_{k-1}) \end{cases}$$

Programmer le calcul de cette liste et la mémoriser dans une variable nommée `ye` (comme $\mathtt{y}_{\mathtt{Euler}}$). Vous pouvez à votre convenance générer une liste classique ou un objet de type `np.ndarray`

Vérification : $\mathtt{y}_{20}$ doit être à peu près égal à 0.658

V) **Résolution avec la fonction préprogrammée `odeint`**

La fonction `odeint` du module `scipy.integrate` a pour syntaxe :

`scipy.integrate.odeint (`*fonction f*`,` *valeur initiale en a*`,` *subdivision de $[a, b]$*`)`

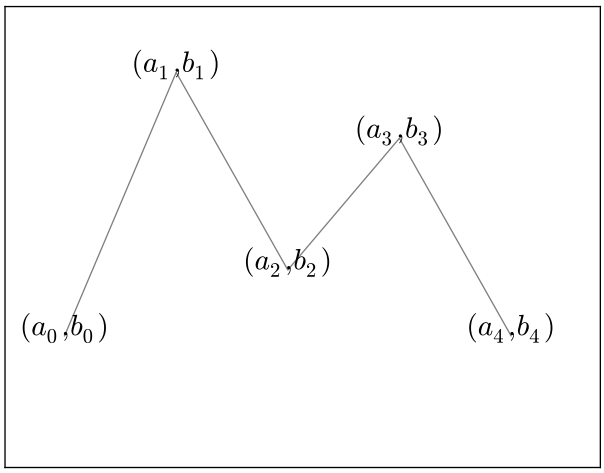
La valeur de retour est une matrice colonne formée des valeurs $\mathtt{y}_0, \dots, \mathtt{y}_n$.

Exécuter la ligne : `ys = scipy.integrate.odeint (f, 0.2, x)`

Afficher ensuite la valeur de `ys` avec une instruction `print`.

VI) **Représentation simultanée**

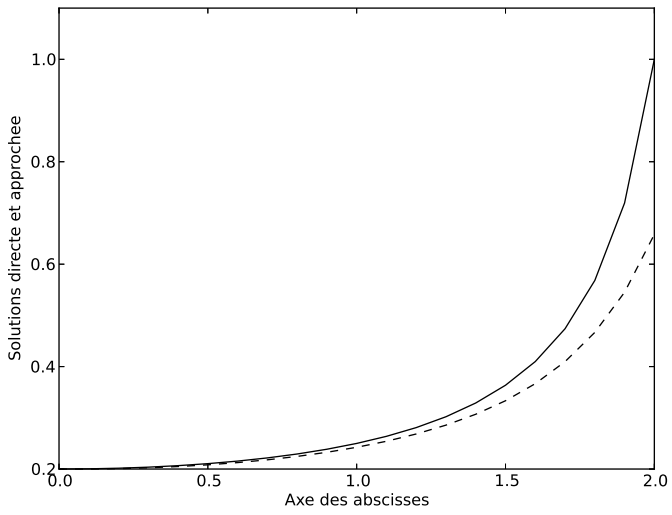
L'appel `plt.plot ([`$a_0, \dots, a_n$`], [`$b_0, \dots, b_n$`])` trace une ligne brisée passant par les points de coordonnées $(a_0, b_0), \dots, (a_n, b_n)$. Cette instruction accepte indifféremment des listes ou des objets de type `numpy.ndarray`



Pour obtenir l'affichage de la fenêtre graphique correspondante il faut après l'instruction `plt.plot` exécuter l'instruction `plt.show ()`

Nous allons tracer simultanément sur le même schéma la courbe donnée par le schéma d'Euler explicite et la courbe donnée par l'usage de la fonction `odeint`. Exécuter les instructions :

```
plt.plot (x, ys)
plt.plot (x, ye, linestyle='--')      # trace en trait discontinu
plt.xlabel ('Axe des abscisses')      # annotation de l'axe des abscisses
plt.ylabel ('Solutions directe et approchee')  # annotation axe des ordonnees
plt.show ()
```



Si on augmente la valeur de n, le schéma d'Euler s'approche de la solution calculée par `odeint`

VII) **Explosion en temps fini**

Reprendre la résolution avec `odeint` mais en résolvant sur le segment $I = [0, 3]$ au lieu de $[0, 2]$.

Il se passe des choses très bizarres : `odeint` envoie un certain nombre de messages ; le tracé de la fonction n'a pas vraiment de sens.

Pour comprendre ce phénomène résolvons explicitement l'équation en supposant que y ne s'annule jamais :

En notations de physique, $\frac{\mathrm{d}y}{\mathrm{d}x} = 2xy^2$ donc $\frac{\mathrm{d}y}{y^2} = 2x\mathrm{d}x$ qui s'intègre en $-\frac{1}{y} = x^2 + Cste$. La condition initiale $y(0) = \frac{1}{5}$ nous donne $Cste = -5$, d'où $y(x) = \frac{1}{5 - x^2}$.

En notations de math on écrirait plutôt $y' = 2xy^2$ nous donne $\frac{y'}{y^2} = 2x$ mais la suite de la résolution est exactement la même.

Bref il est impossible d'intégrer cette équation différentielle sur un segment qui contient $\sqrt{5}$.

Essayez de l'intégrer sur le segment $[0, 2.2]$ car 2.2 est légèrement inférieur à $\sqrt{5}$, et observez les valeurs prises par y.

1.2 Équation d'ordre 2

Nous allons maintenant nous intéresser à une équation d'ordre 2 :

$$(E_2) \qquad y'' + y' - y = x^2 - 4\sin(x) \text{ avec } \begin{cases} y(0) = 1 \\ y'(0) = 2 \end{cases}$$

La fonction `odeint` ne permet de résoudre que des équations d'ordre 1. Le "truc" c'est de transformer l'équation d'ordre 2 en une équation d'ordre 1 en considérant que l'inconnue n'est pas la fonction y mais le couple $z = (y, y')$.
On calculer alors $z' = (y', y'') = (y', -y' + y + x^2 - 4\sin(x)) = f(z)$ où f est une fonction que l'on peut définir :

```
def f (z, x) :
    (y, dy) = z     # on recupere les deux composantes
    return ([dy, -dy + y + x**2 - 4*np.sin(x)])
```

La condition initiale est `z0=[1., 2.]`

VIII) **Résolution**

Définir la fonction comme ci–dessus.

Résoudre ensuite l'équation différentielle sur l'intervalle $[0, 4]$:

```
x = np.linspace (0., 4., 100)
z0 = [1., 2.]
z = scipy.integrate.odeint (f, z0, x)
```

Regarder la valeur de `z` : c'est la matrice $[[\mathtt{y}(\mathtt{x}_0), \mathtt{y}'(\mathtt{x}_0)], \ldots, [\mathtt{y}(\mathtt{x}_{n-1}), \mathtt{y}'(\mathtt{x}_{n-1})]]$ où $[\mathtt{x}_0, \ldots, \mathtt{x}_n]$ est le tableau des abscisses.

IX) **Extraction de la valeur de y et tracé**

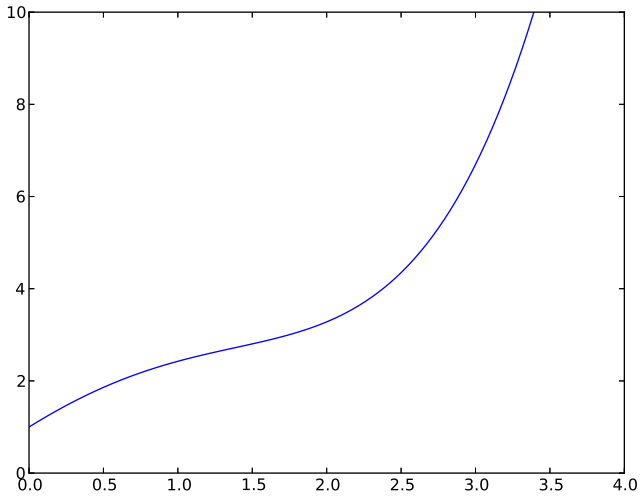
Pour effectuer le tracé il faut extraire la première colonne de la matrice `z`. C'est ici que les matrices `numpy` montrent leur puissance :

```
y = z[:, 0]
```

Effectuer ensuite le tracé :

```
plt.plot (x, y)
plt.show ()
```

Pour ne pas trop écraser la figure, il est possible de fixer l'échelle sur l'axe des ordonnées en insérant l'instruction `plt.ylim (0., 10.)` entre `plt.plot(x,y)` et `plt.show()`. Il existe aussi bien sûr une méthode `xlim` pour l'axe des abscisses.



1.3 Un exemple d'instabilité numérique

Considérons le système différentiel linéaire d'ordre 2 :

$$y'' + y' - 6y = 0 \text{ avec } y(0) = 2 \text{ et } y'(0) = -6$$

Le polynôme associé est $X^2 + X - 6 = (X+3)(X-2)$ donc la solution est de la forme $t \mapsto Ae^{-3t} + Be^{2t}$. Les conditions initiales nous donnent $A + B = 2$ et $-3A + 2B = -6$ ce qui se résout en $A = 2$ et $B = 0$. La solution de notre équation différentielle est donc $t \mapsto 2e^{-3t}$.

X) Exprimer cette équation différentielle sous forme vectorielle.

La résoudre avec `odeint` et la tracer sur l'intervalle $[0, 10]$. Normalement le tracé correspond aux attentes.

XI) Reprendre maintenant la résolution et le tracé sur $[0, 20]$. Que constate–t–on ? Auriez–vous un début d'explication de ce phénomène ?

2 Problème : l'équation de Van der Pol

L'équation de Van der Pol provient du comportement de certains oscillateurs en électronique. Elle dépend de deux paramètres $\varepsilon > 0$ et $\omega_0 > 0$. L'inconnue est une fonction $t \mapsto y(t)$ (le paramètre t désignerait physiquement le temps).

Cette équation est :

$$(V\mathrm{d}P)_{\varepsilon,\omega_0} : \qquad \frac{\mathrm{d}^2y}{\mathrm{d}t^2} - \varepsilon\omega_0\left(1-y^2\right)\frac{\mathrm{d}y}{\mathrm{d}t} + \omega_0^2 y = 0$$

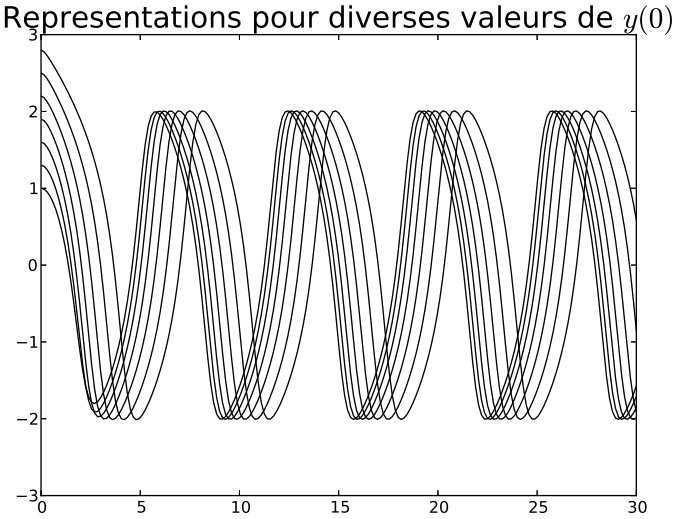
L'étude de cette équation révèle des propriétés très intéressantes. En particulier il y a en général convergence vers un régime périodique dépendant de ε et de ω_0 mais indépendant des conditions initiales (à déphasage près). Par ailleurs ce régime périodique est parfois assez différent d'une sinusoïdale.

XII) **Convergence vers le régime permanent**

Dans cette question on fixe $\varepsilon = 1$, $\omega_0 = 1$ et $\dfrac{\mathrm{d}y}{\mathrm{d}t}(0) = 0$. Seule la condition initiale $y(0)$ n'est pas fixée.

Effectuer sur un même schéma la représentation graphique des différentes solutions sur le segment $[0, 30]$ suivant différentes valeurs de $y(0)$.

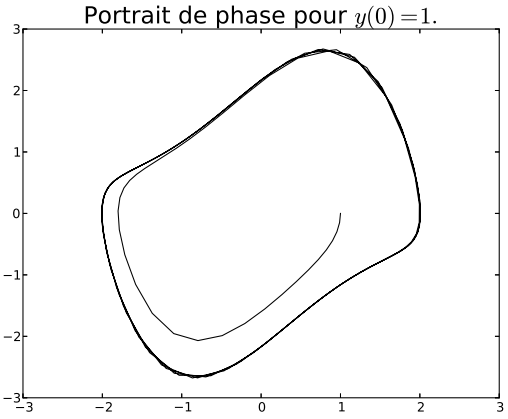
Il n'est permis d'écrire qu'une seule fois `odeint` : on utilisera une boucle et à chaque passage dans la boucle, on fixera la valeur de $y(0)$. La fonction `odeint` sera appelée à l'intérieur de chaque passage dans la boucle.

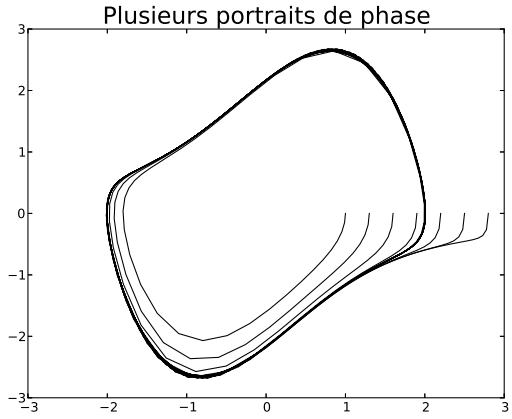


XIII) **Portraits de phase**

Un *portrait de phase* pour une solution y est une représentation avec $y(t)$ en abscisse et $y'(t)$ en ordonnée (au lieu d'avoir t et abscisse et $y(t)$ en ordonnée).

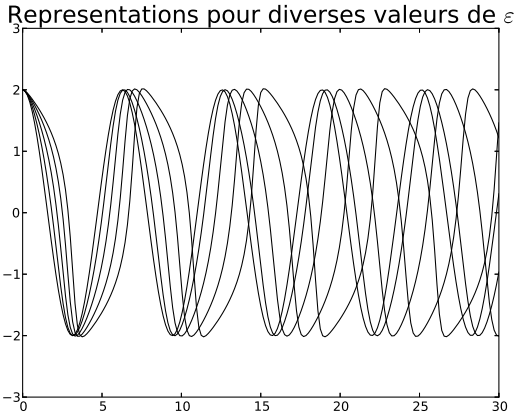
Représenter sur un même schéma les portraits de phase pour différentes valeurs de $y(0)$: on doit voir des trajectoires qui s'enroulent autour d'une même trajectoire cyclique. Cette trajectoire cyclique correspond au régime permanent.





XIV) **Variation de ε**

Désormais on fixe $\omega_0 = 1$, $y(0) = 2.$ et $\dfrac{\mathrm{d}y}{\mathrm{d}t}(0) = 0$, par contre on fait varier ε. Superposer les tracés (grâce à une seuls boucle) pour $\varepsilon = 0.0,\ 0.5,\ 1.0,\ 1.5,\ 2.0$ (on ne demand pas les portraits de phase).



XV) **Détermination de la période du régime permanent**

Reprenons les conditions précédentes : $\omega_0 = 1$, $y(0) = 2.$, $\dfrac{\mathrm{d}y}{\mathrm{d}t}(0) = 0$; seul ε varie.

Il semblerait que la période du régime permanent dépende de la valeur de ε.

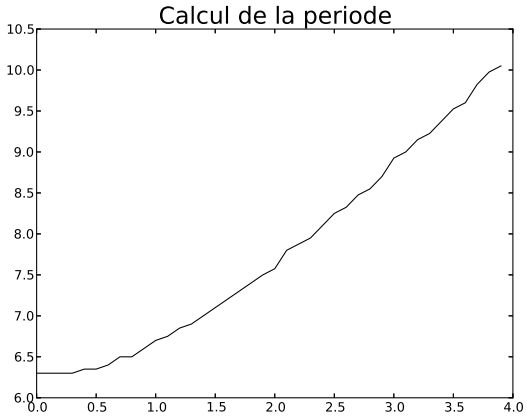
On admettra que la période est la différence des abscisses où sont atteints deux maxima locaux consécutifs.

Écrire une fonction `max_locaux (epsilon)` qui retourne la liste triée par ordre croissant des abscisses où sont atteints les maxima locaux sur le segment $[0, 30]$.

On obtient une liste $[\mathtt{m_0}, \ldots, \mathtt{m_{p-1}}]$. Puisqu'il s'agit nécessairement de valeurs approchées, on considérera que la période est la moyenne des $m_{i+1} - m_i$.

Écrire une fonction `periode (epsilon)` qui renvoie $\dfrac{1}{p-1}\sum_{i=1}^{p-1}(m_i - m_{i-1})$.

Tracer la fonction `periode` sur $[0, 4]$.



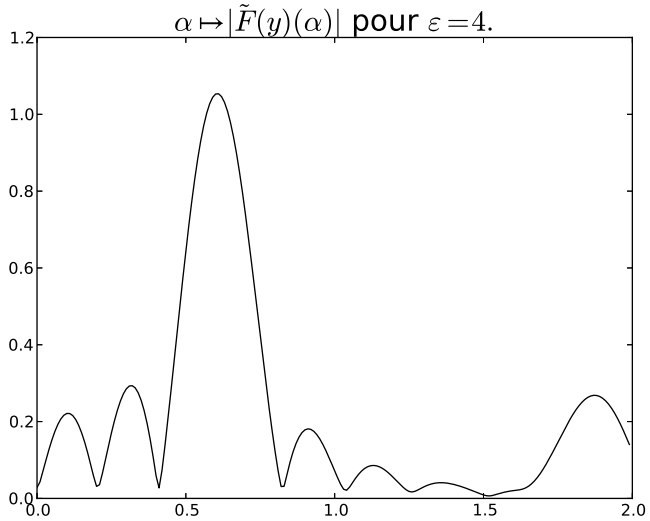
XVI) **Transformation de Fourier de la fonction y**

On considère la fonction $F(y) : \alpha \mapsto \lim\limits_{A\to+\infty} \dfrac{1}{A}\displaystyle\int_0^A y(t)e^{-\alpha t}\mathrm{d}t$.

Puisqu'ici nous avons simplement d'une subdivision équidistribuée $0 = t_0 < t_1 < \cdots < t_{n-1} = 30$ du segment $[0, 30]$ ainsi que les valeurs $y_0, \ldots, y_{n-1}$ correspondantes prises par y en ces points, on prend $A = 30$ et on approxime l'intégrale par une somme de Riemann :

$$\tilde{F}(y)(\alpha) = \frac{1}{n}\sum_{k=0}^{n-1} y_k e^{-\alpha t_k} = \frac{1}{n}\left(\sum_{k=0}^{n-1} y_k \cos(\alpha t_k) - i\sum_{k=0}^{n-1} y_k \sin(\alpha t_k)\right)$$

Enfin, puisqu'il est plus facile de calculer des réels que des complexes, écrire une fonction `fourier (epsilon, alpha)` qui calcule $|\tilde{F}(y)(\alpha)|$. Tracer $\alpha \mapsto |\tilde{F}(y)(\alpha)|$ sur $[0, 2]$ lorsque $\varepsilon = 4$.



Si y est périodique de période T, alors on peut s'attendre à un "pic" à l'abscisse $\alpha = \dfrac{2\pi}{T}$: observez.

XVII) Nous allons chercher à donner un début de justification mathématique à cette dernière affirmation.

Si s est une sinusoïdale, calculer $F(s)(\alpha)$ (il y a 2 cas suivant les valeurs de α).

Si y est périodique de période T et si α n'est pas un multiple de $\dfrac{2\pi}{T}$, montrer que $F(y)(\alpha) = 0$.