Capacités numériques pour les TP

Notions et contenus	Capacités exigibles
1. Outils graphiques	
Représentation graphique d'un nuage de points. Représentation graphique d'une fonction. Courbes planes paramétrées.	Utiliser les fonctions de base de la bibliothèque `matplotlib` pour représenter un nuage de points, la courbe représentative d'une fonction ou une courbe plane paramétrée.
2. Équations algébriques	
Résolution d'une équation algébrique ou d'une équation transcendante : méthode dichotomique. *(Voir poly de cours)*	Déterminer, en s'appuyant sur une représentation graphique, un intervalle adapté à la recherche numérique d'une racine par une méthode dichotomique. Mettre en œuvre une méthode dichotomique afin de résoudre une équation avec une précision donnée. Utiliser la fonction `bisect` de la bibliothèque `scipy.optimize` (sa spécification étant fournie).
3. Intégration – Dérivation	
Calcul approché d'une intégrale sur un segment par la méthode des rectangles.	Mettre en œuvre la méthode des rectangles pour calculer une valeur approchée d'une intégrale sur un segment.
Calcul approché du nombre dérivé d'une fonction en un point.	Utiliser un schéma numérique pour déterminer une valeur approchée du nombre dérivé d'une fonction en un point.
4. Équations différentielles	
Équations différentielles d'ordre 1. *(Voir poly de cours)*	Mettre en œuvre la méthode d'Euler explicite afin de résoudre une équation différentielle d'ordre 1.
Équations différentielles d'ordre supérieur ou égal à 2 *(Voir poly de cours)*	Transformer une équation différentielle d'ordre n en un système différentiel de n équations d'ordre 1. Utiliser la fonction `odeint` de la bibliothèque `scipy.integrate` (sa spécification étant fournie).
5. Probabilité – statistiques	
Variable aléatoire.	Utiliser les fonctions de base des bibliothèques `random` et/ou `numpy` (leurs spécifications étant fournies) pour réaliser des tirages d'une variable aléatoire. Utiliser la fonction `hist` de la bibliothèque `matplotlib.pyplot` (sa spécification étant fournie) pour représenter les résultats d'un ensemble de tirages d'une variable aléatoire. Déterminer la moyenne et l'écart-type d'un ensemble de tirages d'une variable aléatoire.
Régression linéaire.	Utiliser la fonction `polyfit` de la bibliothèque `numpy` (sa spécification étant fournie) pour exploiter des données. Utiliser la fonction `random.normal` de la bibliothèque `numpy` (sa spécification étant fournie) pour simuler un processus aléatoire.

1 Utilisation de numpy

1 Le type array

Le type `array` correspond à un tableau de valeurs, nous utiliserons des nombres de type `float`. On retiendra les fonctions suivantes :

```
import numpy as np              # pour pouvoir utiliser les fonctions np.
A = np.array(liste)             # converti une liste en tableau numpy sous la variable A
N = len(A)                      # mesure la taille de A
B = np.zeros(N)                 # créer un tableau numpy contenant N valeurs 0.
C = np.linspace(ini, fin, N)    # créer un tableau de N valeurs comprises entre ini et fin incluses.
```

On notera qu'on ne peut pas modifier la taille d'un tableau, pas d'ajouter d'éléments, ni de suppression possibles. On peut cependant enregistrer un sous-tableau avec le slicing :

```
A = B[i:j:k]                    # prend un élément sur k de i inclus à j exclu.
B = [i:]                        # commence à i jusqu'à la fin incluse.
```

L'avantage du type `array` par rapport au type `list` est de pouvoir faire des calcul élément par élément avec les opération élémentaires `+`, `-`, `*` et `/` ainsi que les fonctions mathématiques usuelles présentes dans la bibliothèque `numpy` :

```
B = np.exp(A)

C = np.log(A)
```

```
D = np.log10(A)

E = np.sqrt(A)
```

```
F = np.cos(A)
G = np.acos(A)
```

2 Importer une série de mesure

Lorsqu'on prend une série de mesures qu'on enregistre dans une fichier `TPXX.txt` sous la forme :

t	X	Y
15	...	...
30	...	...
45	...	...

Puis dans un fichier `TPXX.py` dans le même dossier :

```
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt("TPXX.txt",skiprows=1,dtype=
    float)
t, X, Y = data[:,0], data[:,1], data[:,2]
```

2 Calculer une incertitude-type

1 Définition

Definition : Résultat d'une mesure

Lorsqu'on procède à une observation expérimentale, on utilise un instrument de mesure et on obtient une valeur numérique associée à une unité (lecture graphique d'une graduation sur une verrerie en chimie, affichage numérique d'un instrument électronique comme un multimètre en électronique, autre protocole plus complexe).

Si on répète cette observation ou qu'on change l'instrument de mesure pour un autre identique on obtient une valeur numérique différente de la précédente. On appelle ce phénomène la **variabilité de la mesure**. Pour tenir compte de ce phénomène on écrit le résultat d'une mesure de la grandeur x, avec $u(x)$ **l'incertitude-type** de la mesure, comme :

$$x = x_{\text{mes}} \text{ unité} \,;\, u(x) = u \text{ unité}$$

Definition : Écart normalisé

On définit l'écart normalisé E entre deux résultats de mesures $x_1 \pm u_1$ et $x_2 \pm u_2$ par la relation : $\boxed{E = \dfrac{|x_1 - x_2|}{\sqrt{u_1^2 + u_2^2}}}$

Par convention, on dit que les deux mesures sont compatibles si $E < 2$.

Remarque : Chiffres significatifs

- On prend toujours deux chiffres significatifs pour $u(x)$;
- On choisi alors le nombre de chiffre significatif pour obtenir la même précision sur x_{mes}.

Exemple : Chiffres significatifs

Soit la mesure d'une résistance $R = 1\,\text{k}\Omega$ on utilise deux méthodes avec chacune une incertitude-type différente :

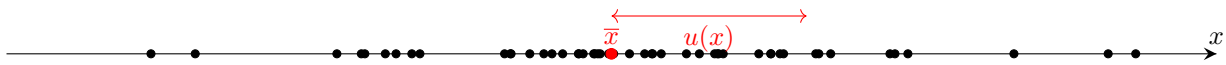
- Dans la première méthode on a $u(R) = 0{,}10\,\text{k}\Omega$, on écrit alors $R_{\text{mes}} = 1{,}00\,\text{k}\Omega$.
- Dans la seconde méthode on a $u(R) = 10\,\Omega$, on écrit alors $R_{\text{mes}} = 1{,}0 \times 10^3\,\Omega$

2 Type A

Definition : incertitude-type A

Dans le cas d'une mesure comportant une série de N (entre 5 et 20) valeurs qui correspondent aux observations x_i :

$$\text{La moyenne : } x_{\text{mes}} = \overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{L'écart-type : } u(x) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2} \qquad \text{et } u(x_{\text{mes}}) = \frac{u(x)}{\sqrt{N}}$$



Pour un tableau `X` contenant une série de mesures (x_1, x_2, x_3, x_4, ...) on retient les fonctions suivantes :

```
import numpy as np
X_mes = np.mean(X)       # Calcule la moyenne de la série de valeurs contenues dans X
u_X = np.std(X, ddof=1) # Calcule l'écart-type de la série de valeurs contenue dans X
```

3 Type B

Definition : incertitude-type B

Lorsqu'on effectue une mesure à partir d'une observation unique on ne peut pas observer la variabilité. Généralement cette observation est comprise entre une valeur minimale X_{min} et une valeur maximale X_{max}, on définit alors le résultat de la mesure par

$$x_{\text{mes}} = \frac{X_{\text{max}} + X_{\text{min}}}{2} \qquad u(x) = \frac{X_{\text{max}} - X_{\text{min}}}{2\sqrt{3}}$$

L'écart-type associé à une distribution de probabilité uniforme sur l'intervalle de largeur 2Δ.

Si on effectue une mesure d'une grandeur x avec un résultat x_{mes} et $u(x)$. On peut simuler via la méthode Monte Carlo la répétition de cette mesure avec une fluctuation aléatoire en utilisant la fonction

```
import numpy as np
X = np.random.normal(X_mes, u_X, N)
```

On obtient un tableau `X` contenant `N` mesures simulées de x tirée aléatoirement selon une loi normale centrée sur `X_mes` et d'écart-type `u_X`.

4 Propagation

Dans cas où on effectue la mesure des grandeurs x et y pour calculer la grandeur $g = g(x, y)$:

$g(x,y)$	$x+y$	$x-y$	$x \times y$	$\frac{x}{y}$
$u(g)$	$\sqrt{u^2(x)+u^2(y)}$	$\sqrt{u^2(x)+u^2(y)}$	$\lvert\overline{g}\rvert\sqrt{\left(\frac{u(x)}{\overline{x}}\right)^2+\left(\frac{u(y)}{\overline{y}}\right)^2}$	$\lvert\overline{g}\rvert\sqrt{\left(\frac{u(x)}{\overline{x}}\right)^2+\left(\frac{u(y)}{\overline{y}}\right)^2}$

avec $\overline{g} = g(\overline{x}, \overline{y})$.

Propriété : Capacité numérique au programme

Simuler, à l'aide de python, un processus aléatoire permettant de caractériser la variabilité de la valeur d'une grandeur composée.

Méthode : Propagation des incertitudes (bis)

Dans le cas où une grandeur g est reliée aux grandeurs x et y par la relation : $g = f(x, y)$ En utilisant les résultats des mesures de x et y :

$$x = x_{\text{mes}}\,;\ u(x) = u_x \text{ et } y = y_{\text{mes}} \qquad \text{et } u(y) = u_y$$

calculer simplement $g = g_{\text{mes}} = f(x_{\text{mes}}, y_{\text{mes}})$ et évaluer $u(g)$ de la manière suivante :

- Soit N valeurs de x notée $x_i \quad \in \quad [x_i - u_x\sqrt{3}; x_i + u_x\sqrt{3}]$
- Soit N valeurs de y notée $y_i \quad \in \quad [y_i - u_y\sqrt{3}; y_i + u_y\sqrt{3}]$
- Soit N valeurs de g notée g_i en appliquant la formule : $g_i = f(x_i, y_i)$
- on calcul l'écart-type des valeurs de g_i ce qui nous donne l'incertitude-type : $\quad u(g) = u_g$
- on peut également vérifier qu'on a bien : $\quad \overline{g} = f(x_{\text{mes}}, y_{\text{mes}})$

Exemple : Propagation des incertitudes produit

Soit $z = xy$ avec $x = x_{\text{mes}}$ et $u(x)$, $y = y_{\text{mes}}$ et $u(y)$.

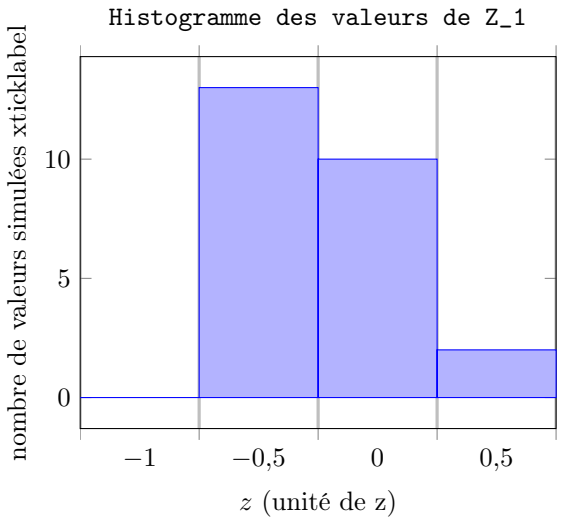
```
import numpy as np
X = np.random.normal(X_mes, u_X, N)
Y = np.random.normal(Y_mes, u_Y, N)
Z = X*Y
Z_mes = np.mean(Z)
u_Z = np.std(Z, ddof=1)
```

3 Graphiques

1 Histogramme

On peut tracer l'histogramme des N valeurs z_i contenues dans un tableau `Z` à partir d'une simulation Monte Carlo :

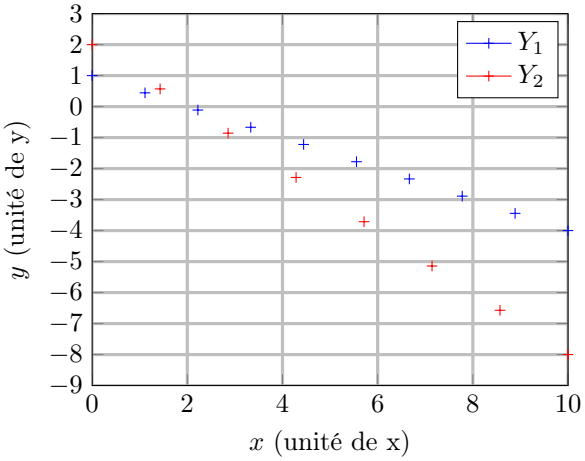
```
import numpy as np
import matplotlib.pyplot as plt
plt.figure()
plt.hist(Z_1, bins='rice')
plt.title("Histogramme des valeurs de Z_1")
plt.xlabel("z (unité de z)")
plt.ylabel("nombre de valeurs simulées")
plt.grid()
plt.show()
```



2 Courbes

Soit deux séries de mesures associées aux tableaux (`X_1`, `Y_1`) de la **même taille** et (`X_2`, `Y_2`) de la **même taille** avec `Y_1` et `Y_2` qui représentent la même grandeur et `X_1` et `X_2` également, on trace les courbes :

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure()
plt.plot(X_1, Y_1, '+', label="Y_1")
plt.plot(X_2, Y_2, '+', label="Y_2")
plt.xlabel("x (unité de x)")
plt.ylabel("y (unité de y)")
plt.grid()
plt.show()
```

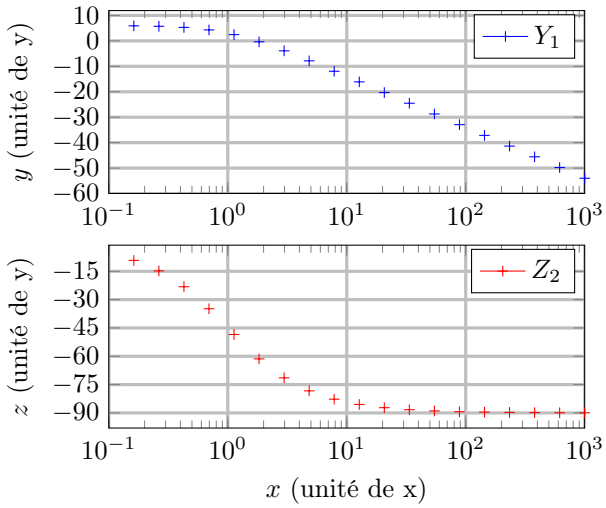


Si on veut tracer l'axe des x à l'échelle log on ajoute `plt.xscale('log')`

3 Plusieurs graphes

Soit deux séries de mesures associées aux tableaux (`X_1`, `Y_1`) de la **même taille** et (`X_2`, `Z_2`) de la **même taille** avec `Y_1` et `Z_2` qui ne représentent pas la même grandeur mais que `X_1` et `X_2` représentent la même grandeur, on trace les courbes dans une même fenêtre l'une en dessous de l'autre :

```
import numpy as np
import matplotlib.pyplot as plt
fig, (ax1, ax2) = plt.subplots(2, 1)
fig.suptitle('Titre')
ax1.plot(X_1, Y_1, '+',label="Y_1")
ax1.set_xscale('log')
ax1.grid(which='both')
ax1.set_ylabel('y (unité de y)')
ax2.plot(X_2, Z_2, '+', label="Y_2")
ax2.set_xscale('log')
ax2.grid(which='both')
ax2.set_ylabel('z (unité de z)')
plt.xlabel('x (unité de x)')
plt.show()
```



4 Régression linéaire

1 Calcul des coefficients

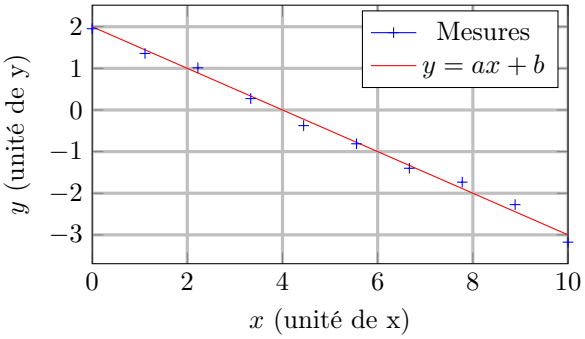
On cherche à faire une régression linéaire, c'est-à-dire l'expression de la droite de régression $y = ax + b$ qui minimise l'écart des mesures de x_i et y_i contenues dans les tableaux `X` et `Y` de même taille à la droite :

```
import numpy as np
a,b=np.polyfit(X,Y,1) # regression lineaire de la forme Y=a*X+b
```

2 Ajouter la courbe de régression sur un graphique

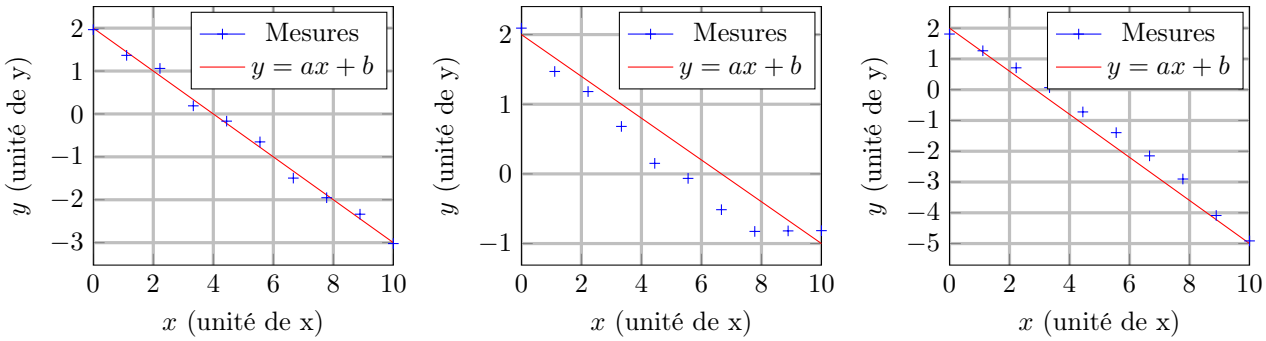
Soit la série de mesures associée aux tableaux (`X`, `Y`) de la **même taille**, on a effectué une régression linéaire de la forme $y = ax + b$ et on trace les courbes :

```
import numpy as np
import matplotlib.pyplot as plt
a,b=np.polyfit(X,Y,1)
plt.figure()
plt.plot(X, Y, '+', label="Mesures")
plt.plot(X, a*X+b, label="y=aX+b")
plt.xlabel("x (unité de x)")
plt.ylabel("y (unité de y)")
plt.grid()
plt.show()
```



3 Validation visuelle

Dans un premier temps, on regarde la courbe pour voir si la tendance linéaire se vérifie :



4 Prise en compte des incertitudes

Propriété : Capacité numérique

Simuler, à l'aide de python, un processus aléatoire de variation des valeurs expérimentales de l'une des grandeurs (simulation Monte-Carlo) pour évaluer l'incertitude sur les paramètres du modèle.

Méthode : Régression linéaire

Dans le cas où on a effectué k mesures de x et y avec les incertitude-types associées notée :

$$x_k = x_{k,\text{mes}}\,;\ u(x_k) = u_k, x \text{ et } y_k = y_{k,\text{mes}}\,;\ u(y_k) = u_k, y$$

On souhaite calculer les coefficients a et b et leurs incertitudes associées $u(a)$ et $u(b)$ qui optimisent la relation : $y = ax+b$
Pour cela on va utiliser python pour utiliser la méthode suivante :

- on calcul N valeurs pour chaque x_k notée $x_{k,i}$ avec une probabilité uniforme pour que :

$$x_{k,i} \in [x_{k,i} - u_{k,x}\sqrt{3}; x_{k,i} + u_{k,x}\sqrt{3}]$$

- on calcul N valeurs pour chaque y_k notée $y_{k,i}$ avec une probabilité uniforme pour que :

$$y_{k,i} \in [y_{k,i} - u_{k,y}\sqrt{3}; y_{k,i} + u_{k,y}\sqrt{3}]$$

- on effectue N régression linéaires en prenant pour chaque valeur de i les k valeurs de $x_{k,i}$ et $y_{k,i}$ obtenir la droite d'équation :

$$y = a_i x + b_i$$

  On possède alors N valeurs de a notées a_i et N valeurs de b notées b_i.

- on calcul l'écart-type des valeurs de a_i puis de b_i ce qui nous donne les incertitudes-types :

$$u(a) = u_a \quad \text{et} \quad u(b) = u_b$$

- on peut également vérifier qu'on a bien : $y = \overline{a}x + \overline{b}$
  en faisant une régression linéaire sur nos k valeurs de x et y.
- On note alors le résultat de notre régression linéaire : $y = ax + b$ avec $a = \overline{a}$; $u(a) = u_a$ et $b = \overline{b}$; $u(b) = u_b$

5 Calcul numérique

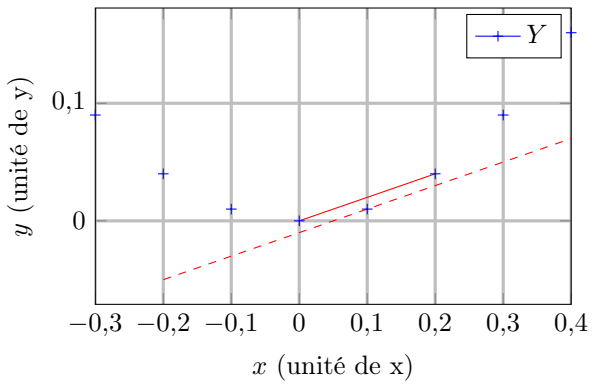
1 Dérivation

On considère une série de mesure contenue dans deux tableau `X` et `Y` de même taille contenant les N mesures (x_i, y_i), on utilise alors la définition du taux d'accroissement :

$$\frac{\mathrm{d}y}{\mathrm{d}x}(x_i) \simeq \frac{y_{i+1} - y_{i+1}}{x_{i+1} - x_{i-1}}$$

qui permet d'avoir une valeur approchée de la dérivée centrée sur la ^{e} mesure.

```
import numpy as np
import matplotlib.pyplot as plt
N = len(X)
dY = np.zeros(N-2)
X_2 = X[1:N-1]
for i in range(N-2):
    dY[i] = (Y[i+2]-Y[i])/(X[i+2]-X[i])
```



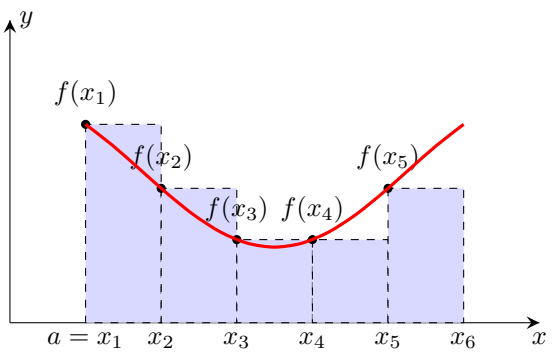
Le tableau `X_2` est nécessaire pour extraire les mesures contenues dans `X` sans les deux points extrêmes où l'on ne peut pas calculer la dérivée.

2 Intégration

Definition : Intégration numérique par la méthode des rectangles

À l'aide de python, on peut calculer une valeur approchée d'une intégrale d'une fonction f en séparant un intervalle $[a; b]$ en N intervalles $[x_i; x_{i+1}]$ tel que $x_{i+1} - x_i = (b - a)/N$ et où la fonction f est considérée comme constante sur l'intervalle $[x_i; x_{i+1}]$. On choisi alors $f(x) \simeq f(x_i)$ sur l'intervalle $[x_i; x_{i+1}]$ qu'on appelle méthode des rectangles à gauche ce qui donne :

$$I = \int_a^b f(x)\mathrm{d}x \simeq \left(\frac{b-a}{N}\right) \sum_{i=1}^{i=N} f\left(\frac{x_{i+1} + x_i}{2}\right)$$



On considère qu'on a une série de mesures contenue dans deux tableaux `X` et `Y` de même taille, on cherche à calculer la valeur approchée de l'intégrale :

$$I = \int_a^b y(x)\mathrm{d}x$$

```
import numpy as np
N = len(X)
I = 0
for i in range(N-1):
    I = I + (X[i+1]-X[i]) * Y[i]
print(I)
```

3 Résolution d'une équation

Voir poly de cours pour la méthode dichotomique. On peut utiliser la fonction `bisect(f, a, b)`. Qui renvoie la valeur approchée à 10^{-12} près de $x_0 \in [a, b]$ tel que $f(x_0) = 0$ avec `f` une fonction à définir.

4 Résolution d'équations différentielles

Voir poly de cours pour la méthode d'Euler. On peut utiliser la fonction `odeint(F,Y_0,t)`.Avec `F(Y,t)` qui correspond à la fonction $\frac{\mathrm{d}Y}{\mathrm{d}t} = F(Y, t)$, `Y_0` qui correspond à $Y(t = 0)$.