éduSCOL
Informer et accompagner les professionnels de l'éducation

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'ÉDUCATION NATIONALE ET DE LA JEUNESSE

VOIE GÉNÉRALE

2DE | 1RE | TLE

Physique-chimie

ENSEIGNEMENT SPÉCIALITÉ

PROGRAMMER EN PYTHON
FICHE POUR ALLER PLUS LOIN N°2 : SIMULATION D'UNE DÉCROISSANCE RADIOACTIVE

L'étude d'un processus microscopique aléatoire tel que la fission radioactive permet de révéler l'importance du nombre de particules envisagées dans une modélisation.

Capacité numérique mise en œuvre
Utiliser un langage de programmation pour rendre compte de la décroissance radioactive.

On utilise ici un module qui permet de générer des phénomènes aléatoires : **random.** On commence par l'appeler en même temps que les modules classiques de tracé et d'outils numériques.

```
import numpy as np
import matplotlib.pyplot as plt
import random
```

On définit ensuite une fonction **radioactivité** qui va rendre compte du « fonctionnement » aléatoire de la fission. Cette fonction a pour argument le nombre total d'atomes non désintégrés à un instant (à un pas de calcul). Pour chaque atome dans cette liste, la fonction va effectuer un tirage au sort, où le résultat peut être 1, avec une chance sur 50 (c'est le **random.randint**(1,50) qui génère un entier aléatoire de 1 à 50 et qui a donc une chance sur 50 de générer). Si le résultat de ce tirage est 1, on admet que l'atome correspondant est désintégré et on fait diminuer le nombre d'atomes restants.

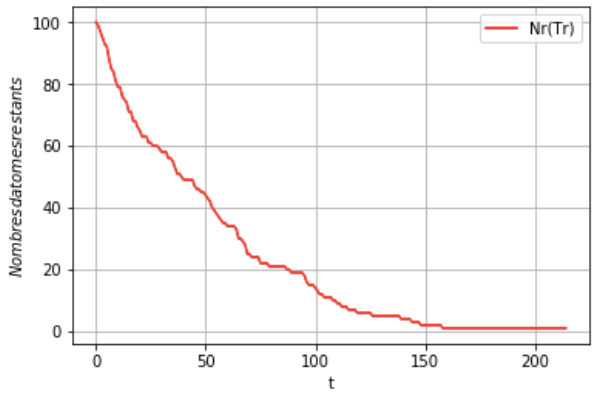
```
def radioactivité(NA):
    Nrestants=NA
    for i in range(0,NA):
        if random.randint(1,50)==1:
            Nrestants=Nrestants-1
    return Nrestants
```

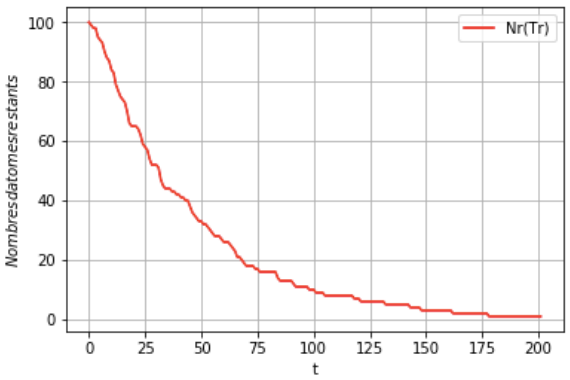
Après avoir défini les conditions initiales, on crée grâce à la fonction **append** une liste de temps et une liste de nombre d'atomes restants.

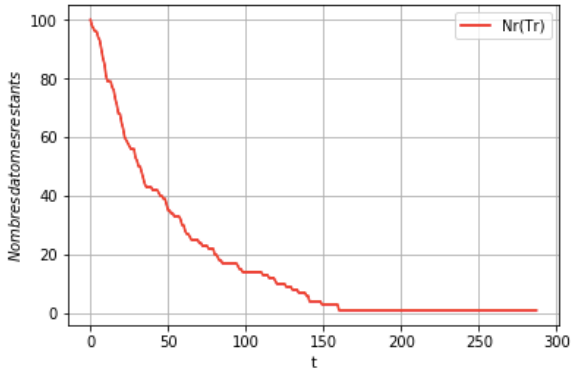
```
t=0
nb=100
n0=100
Nr=[]
Tr=[]
while nb>0:
    Tr.append(t)
    Nr.append(nb)
    nb=radioactivité(nb)
    t=t+1
```

On peut alors tracer l'évolution du nombre d'atomes restants en fonction du temps avec les démarches classiques. Il est intéressant de relancer le calcul plusieurs pour constater que l'allure de la courbe varie : on illustre ici le caractère aléatoire du processus microscopique.

```
plt.plot(Tr,Nr,'r-',label='Nr(Tr)')
plt.legend(loc='best')
plt.xlabel('t') ; plt.ylabel('$Nombres d atomes restants$')
plt.grid()
plt.show()
plt.figure()
```



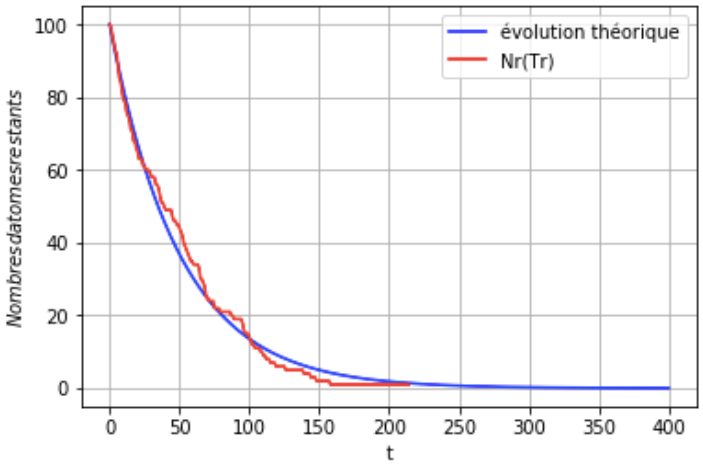




On peut ensuite tenter de comparer la décroissance aléatoire à une loi exponentielle. On illustre alors le fait que si les allures sont similaires, le modèle exponentielle ne traduit qu'imparfaitement la réalité pour de si petites populations. Le module de tracé des deux courbes est précis ci-dessous.

```
def Nthéo(u):
    return n0*(np.exp(-u/50))

t0, tf = 0., 400. # bornes de l'intervalle de temps pour la résolution
N = 10000 # nombre de pas
t = np.linspace(t0,tf,N+1) # liste des dates pour le calcul des solutions aprochées
```



Retrouvez éduscol sur

On peut alors faire varier la taille de la population initiale et vérifier son influence : pour une population initiale de n0=5000 atomes, il n'est plus possible visuellement de faire la différence entre la loi simulée aléatoirement et une modélisation exponentielle ; ce qui est à la fois une illustration de la capacité prédictive d'une modélisation par une équation différentielle et en même temps une mise en garde quant au fait qu'elle n'est pas toujours valable.