PROJET INFORMATIQUE N°3

Désintégration Radioactive

Sommaire

Bibliographie

Utilisation de Matplotlib avec représentations graphiques : https://courspython.com/introduction-courbes.html

Site web pour l'équation différentielle : https://www.codingame.com/playgrounds/53303/apprendre-python-dans-le-secondaire/resolution-numerique-dequations-differentielles

Utilisation de Numpy dans les fonctions : www.numpy.org

Utilisation de la fonction random dans les fonctions : https://docs.python.org/fr/3/library/random.html

Utilisation des couleurs : https://matplotlib.org/stable/tutorials/colors/colormaps.html

Désintégration Radioactive

Nous allons ici, Simuler la désintégration radioactive d'un échantillon, cette expérience sera traitée sur un intervalle de temps. Nous allons représenter cet ensemble par une matrice carrée où chaque atome sera représenté par un chiffre. Le 1 représentera l'atome à l'état initial, le 0 lui représentera l'atome désintégré. Comme la désintégration d'un atome est lié au hasard, nous allons utiliser une loi statique, une loi de probabilité, cette probabilité est obtenue par la formule :

$$p = \ln(2)\frac{\mathrm{d}t}{T}$$

Le programme de la désintégration radioactive est le suivant :

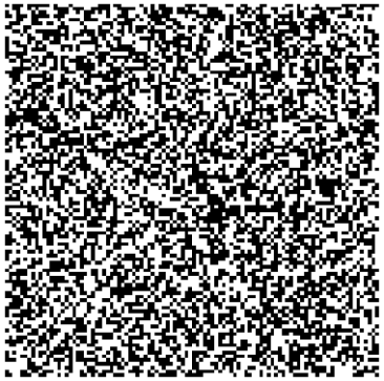
```
def desintegration_radioactive(Taille:int, Demi_vie:float, pas_t:float, duree:float)->np.ndarray:
    taille_matrice = (Taille, Taille) # Taille de la matrice
    Matrice = np.ones(taille_matrice) # Création de la matrice avec que des 1
    proba_desintegration = np.log(2)*pas_t/Demi_vie # Définition de la probabilité de désintégration
    Releve_Temporel = int(duree/pas_t) # Relever temporel pour les boucles du programme
    L_Restant = [] # Liste des noyaux restant dans la matrice
    Nbr_tours = 0
    while Nbr_tours < Releve_Temporel: # On exécute la boucle avec le relevé temporel
        for J in range(Taille):
            for I in range(Taille): # Balayage complet de la matrice
                if Matrice[I][J] == 1: # Si le point de la matrice vaut 1
                    if random.random() < proba_desintegration:
                        Matrice[I][J] = 0 # En fonction de la probabilité, on met un 0
                    else:
                        pass
                else:
                    pass
        image = plt.imshow(Matrice, cmap=cm.binary) # binary nous permet d'avoir le graphique en noir et blanc
        plt.title("En cours de chargement ...")
        plt.axis('off') # On enlève les axes pour un meilleur rendu graphique
        plt.draw() # On trace la matrice sur le graphique
        plt.pause(0.001) # On met une pause entre chaque actualisation de la matrice
        image.set_data(Matrice) # On actualise l'image de la matrice
        L_Restant.append(np.count_nonzero(Matrice))
        Nbr_tours = Nbr_tours + 1 # On ajoute un tour et on passe au balayage suivant
    print("Le relevé des nombres de pas restant à chaque balayage est :", L_Restant)
    print("A la fin de l'expérience, nous avons la matrice suivante : ")
    print(Matrice)
    plt.title("Terminer")
    plt.show() # On affiche la matrice
    plt.close()
    return Matrice # On retourne la matrice avec son animation
```

Pour la désintégration radioactive, nous devons utiliser le cas de l'uranium 238 de demi-vie de 4,5 milliards d'années se désintégrant en thorium 234.

En exécutant la fonction avec la première interprétation, nous avons au début une matrice rempli de 1 pour l'uranium 238 puis on exécute la fonction avec les paramètres suivant :

- Taille de la matrice : 100 (afin d'avoir un bon rendu visuel)
- Demi-vie : 4500 (on passe en millions d'années pour avoir des valeurs plus précise)
- Pas de temps : 100 (Donc 100 millions d'années)
- Durée de l'expérience : 4500 (Identique à la demi-vie avec d'avoir une désintégration vers l'élément suivant, ici, le thorium 234)

On obtient l'image finale de la matrice avec le nombre de noyaux restants :



Les nombres de points restants à chaque pas de temps sont :

[9840, 9682, 9540, 9409, 9237, 9085, 8914, 8770, 8620, 8479, 8356, 8244, 8119, 7996, 7889, 7774, 7663, 7543, 7423, 7307, 7196, 7092, 6991, 6894, 6788, 6689, 6591, 6476, 6377, 6279, 6193, 6108, 6012, 5928, 5836, 5764, 5663, 5590, 5499, 5406, 5329, 5249, 5176, 5093, 5017]

Equation différentielle de la désintégration radioactive

Nous avons une équation différentielle qui vérifie théoriquement l'évolution de la désintégration sur l'intervalle de temps et qui doit avoir des similitudes avec l'observation graphique précédente de la désintégration radioactive. En effet nous devons retrouver le même nombre de points à chaque pas de temps de la fonction. On a l'équation différentielle suivante :

$$\frac{dN}{dt} = -\frac{\ln(2)}{T}N$$

La fonction de l'équation différentielle est la suivante :

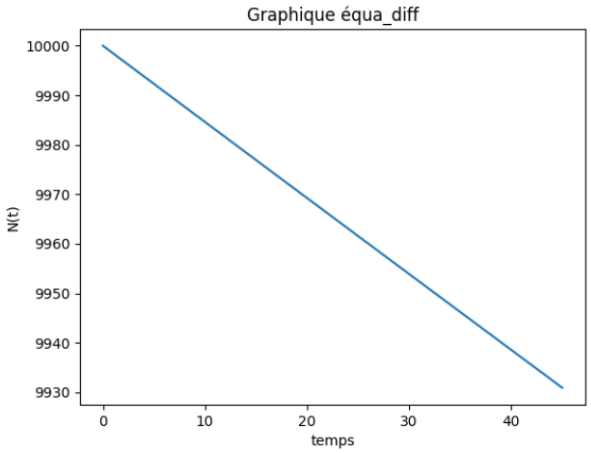
```
# Fonction pour l'équation différentielle de la désintégration radioactive
a = -np.log(2)/Demi_vie #On définit la constante de l'équa_diff
t = np.linspace(0, int(duree/pas_t), int(duree/pas_t)) # On précise le nombre de point en fonction du temps

def equation(N, t): # On créer la fonction de l'équa_diff
    return a*N

N0 = Taille*Taille
N = integ.odeint(equation, N0, t) # On fait fonctionner ODEINT avec l'équation, la taille de la matrice et le temps
plt.plot(t, N) # On trace la fonction sur le graphique
plt.title('Graphique équa_diff')
plt.xlabel('temps') # Axe x : temps
plt.ylabel('N(t)') # Axe y : nombre de valeur
plt.show() # On affiche le graphique
```

Afin de pouvoir fonctionner avec la fonction odeint. La fonction de l'équation différentielle n'est pas une fonction complètement regroupée dans un def.

En exécutant la fonction avec la première interprétation, nous obtenons le graphique et les nombres de points suivant :



[[10000] [9998.42478957] [9996.84982727] [9995.27511306] [9993.7006469] [9992.12642877] [9990.55245862] [9988.97873641] [9987.4052621] [9985.83203564] [9984.25905701] [9982.68632615] [9981.11384303] [9979.5416076] [9977.96961983] [9976.39787967] [9974.82638708] [9973.25514203] [9971.68414448] [9970.1133944] [9968.54289175] [9966.97263648] [9965.40262856] [9963.83286795] [9962.26335462] [9960.69408851] [9959.12506959] [9957.55629783] [9955.98777318] [9954.41949561] [9952.85146508] [9951.28368154] [9949.71614496] [9948.1488553] [9946.58181253] [9945.01501659] [9943.44846746] [9941.88216509] [9940.31610945] [9938.7503005] [9937.18473819] [9935.6194225] [9934.05435337] [9932.48953077] [9930.92495467]]

Filiation Radioactive

La filiation radioactive, c'est une désintégration radioactive, mais notre ensemble change au cours du temps, le noyau passe par trois états : mère, intermédiaire et fille, on différencie les états par leur demi-vie. Chacune, sont des désintégrations à part entière et on donc des probabilités de désintégration différentes les unes des autres. On a deux désintégrations à chaque fois mais dans certains cas le nombre de désintégration peut varier.

La fonction de filiation radioactive est seulement une adaptation des deux fonctions précédentes.

Pour les désintégrations radioactives par filiation radioactive, la fonction est très similaire à la première, sauf que maintenant nous avons deux probabilités, deux relevés temporels et donc deux boucles While afin d'avoir les deux désintégrations.

Afin de gérer le nombre de boucle et le temps d'expérience, le relevé temporel total a été « couper » en deux en respectant la demi-vie de chacun des éléments présents dans les désintégrations.

Pour la première filiation radioactive, nous avons le cas du molybdène 99 de demi-vie 66h, donnant le technétium 99 métastable de demi-vie 6h donnant lui-même le technétium 99 stable.

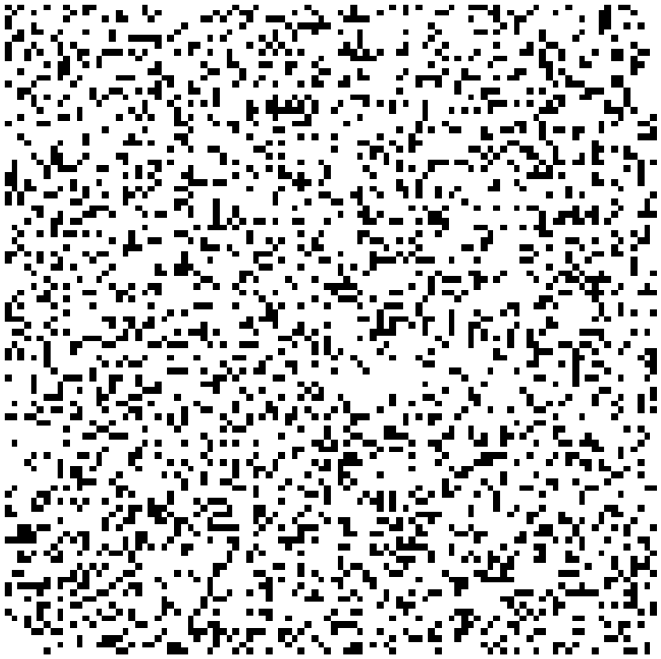
Donc ici nous ne changeons pas les unités et restons en heure pour tout les paramètres de l'expérience :

- Matrice de 100 ;
- Pas de temps de 1 ;
- Durée de l'expérience 72 car 66+6 afin de respecter la demi-vie de chaque élément ;
- $1^{ère}$ demi-vie : 66 (h)
- $2^{ème}$ demi-vie : 6 (h)

Nous obtenons l'image de la première désintégration :

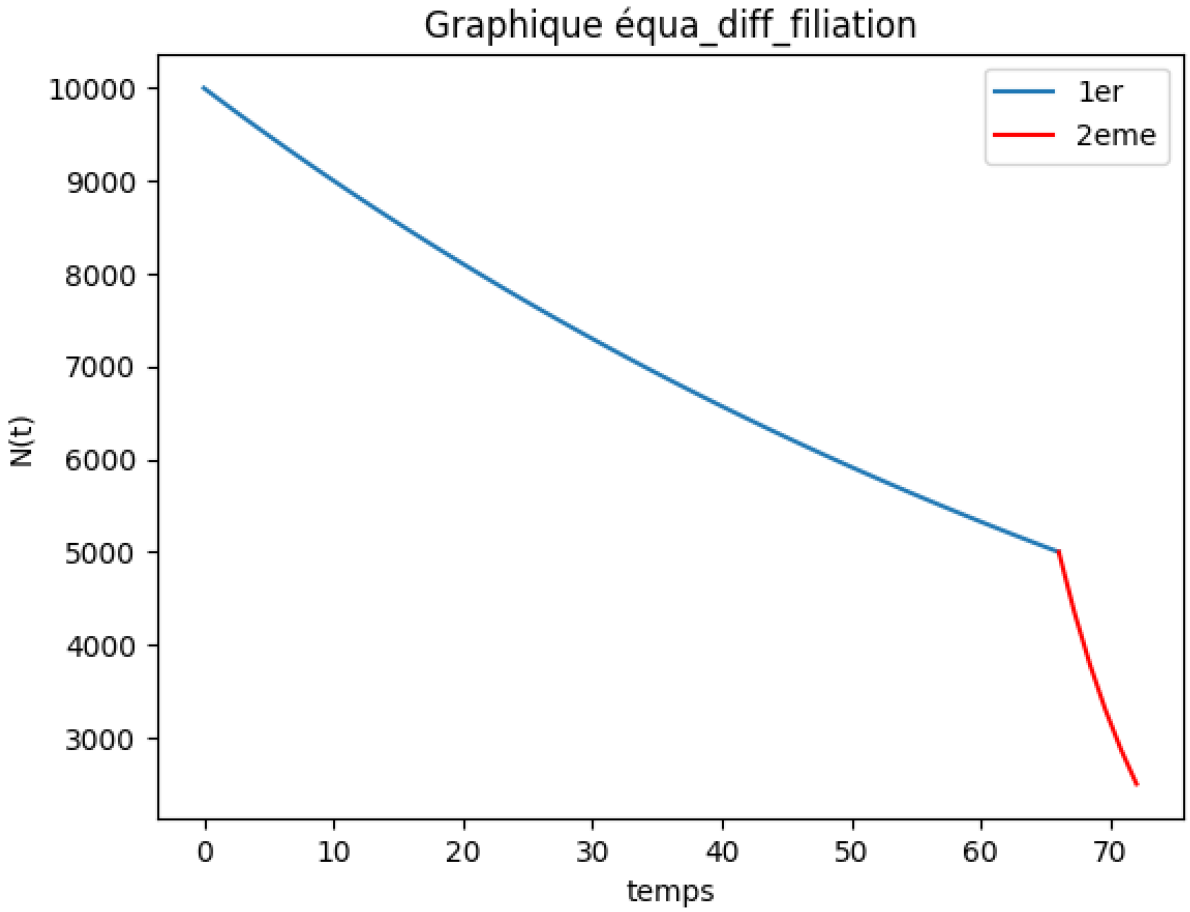
Nous sommes partis du cas du molybdène 99 de demi-vie 66h, donnant le technétium 99 métastable de demi-vie 6h.

Nous obtenons ensuite l'image de la deuxième désintégration :



Le technétium 99 métastable de demi-vie 6h donnant lui-même le technétium 99 stable.

Nous obtenons le graphique de l'équation différentielle :



Le programme de l'équation différentielle est pratiquement le même que la fonction précédente mais cette fois-ci avec tous les paramètres en double et seulement un regroupement des courbes avec la fonction subplot de Matplotlib.

On va étudier le cas de la décomposition du plutonium 240, de demi-vie 6560 ans, en uranium 236, de demi-vie 23,4 millions d'années, donnant lui-même de thorium 232, afin d'étudier l'influence des valeurs respectives des demi-vies du noyau mère et du noyau intermédiaire.

Dans la première désintégration, la valeur de demi-vie est beaucoup trop basse par rapport à la deuxième, ce qui fait qu'il ne restera pas de noyaux à la fin de la première désintégration radioactive. Voilà donc l'influence respective des valeurs de demi-vie sur le système de filiation radioactive.

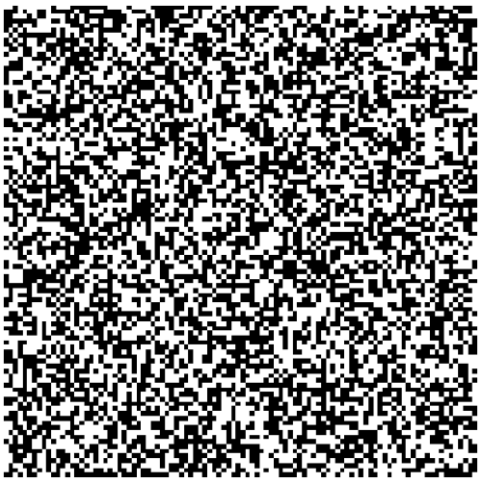
Pour finir, le cas de la filiation radioactive du thorium 227 (demi-vie de 18,7 jours) en radium 223 (demi-vie 11,4 jours) en radon 219.

On reste sur une demi-vie et une durée d'expérience en jours :

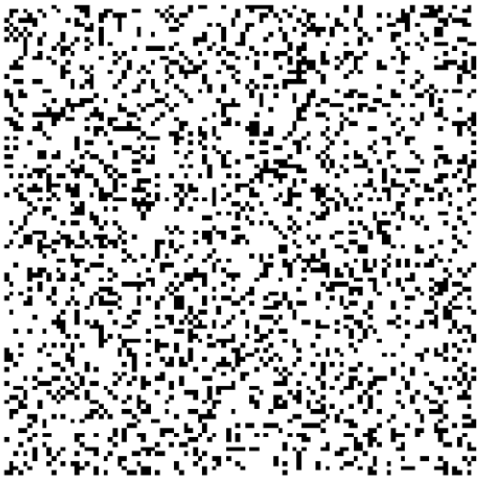
- Taille de la matrice de 100
- Pas de temps de 1 (jour)
- 1ère Demi-vie : 18,7 (jours)
- 2ème Demi-vie : 11,4 (jours)
- Durée totale de l'expérience : 30 (jours)

Nous obtenons les résultats suivants :

1ère désintégration :



2ème désintégration :



Pour finir, le graphique de l'équation différentielle :