

## Création d'un programme en Python avec un agent conversationnel

### Contexte

A partir d'un TP de spécialité physique-chimie en terminale sur la décroissance radioactive (voir fichier joint), l'objectif est de comparer plusieurs méthodes pour calculer la demi-vie d'un élément :

- la calculatrice ;
- le tableur (Excel ou Libre Office Calc) ;
- la programmation avec un fichier en Python fourni (voir annexe à la fin).

L'objectif du TP est de montrer que la programmation permet d'aboutir plus rapidement au résultat. Un des problèmes de cette méthode est qu'elle nécessite la création du programme (il était ici fourni, l'élève devait juste le compléter).

Nous utilisons un agent conversationnel, ici Chat GPT, pour créer un programme permettant de calculer la demi-vie de notre élément. Cela permettra d'ajouter une 4<sup>ème</sup> méthode, utilisant l'IA, à notre TP.

### Mise en œuvre

- Le TP a été initialement chargé et le prompt initial était le suivant :  
*Bonjour, j'aimerais calculer la demi-vie du potassium 40 avec un programme informatique de niveau lycée de type Python*

Le programme proposé est le suivant :

```
import math
import matplotlib.pyplot as plt

# Données
N0 = 100000 # nombre initial de noyaux
duree_pas = 10 # en millions d'années
t = 0
N = N0

# Calcul de la constante radioactive lambda
reste_pourcent = 99.45
lambda_ = -math.log(reste_pourcent / 100) / duree_pas

# Listes pour stocker les valeurs
temps = [0]
noyaux = [N0]

# Recherche de la demi-vie et stockage pour le graphe
while N > N0 / 100: # on va jusqu'à 1% pour un graphe lisible
    N = N * math.exp(-lambda_ * duree_pas)
    t += duree_pas
    temps.append(t)
    noyaux.append(N)

    if N <= N0 / 2 and 'demi_vie' not in locals():
        demi_vie = t # on récupère le temps quand on atteint la moitié

# Affichage de la demi-vie
print("La demi-vie estimée du potassium-40 est de :", demi_vie, "millions d'années")
```

Le programme fonctionne et l'objectif initial est atteint.

- L'agent conversationnel propose alors de générer une version graphique avec matplotlib pour tracer la décroissance.

```

import math
import matplotlib.pyplot as plt

# Données
N0 = 100000 # nombre initial de noyaux
duree_pas = 10 # en millions d'années
t = 0
N = N0

# Calcul de la constante radioactive lambda
reste_pourcent = 99.45
lambda_ = -math.log(reste_pourcent / 100) / duree_pas

# Listes pour stocker les valeurs
temps = [0]
noyaux = [N0]

# Recherche de la demi-vie et stockage pour le graphe
while N > N0 / 100: # on va jusqu'à 1% pour un graphe lisible
    N = N * math.exp(-lambda_ * duree_pas)
    t += duree_pas
    temps.append(t)
    noyaux.append(N)

    if N <= N0 / 2 and 'demi_vie' not in locals():
        demi_vie = t # on récupère le temps quand on atteint la moitié

# Affichage de la demi-vie
print("La demi-vie estimée du potassium-40 est de :", demi_vie, "millions d'années")

# Tracé du graphique
plt.figure(figsize=(8,5))
plt.plot(temps, noyaux, label="Décroissance du 40K")
plt.axhline(N0/2, color='r', linestyle='--', label='Moitié de N0')
plt.axvline(demi_vie, color='g', linestyle='--', label=f'Demi-vie ≈ {int(demi_vie)} Ma')
plt.xlabel("Temps (millions d'années)")
plt.ylabel("Nombre de noyaux restants")
plt.title("Décroissance radioactive du potassium-40")
plt.legend()
plt.grid()
plt.show()

```

Le programme a permis le tracé graphique de la décroissance du nombre de noyaux de potassium-40 en fonction du temps, en plus de l'estimation de la demi-vie. Le graphique permet aussi de visualiser à quel moment on atteint la moitié de  $N_0$ .

Néanmoins, la partie *Tracé du graphique* n'était pas totalement exacte. Il a fallu enlever la partie entourée sans toucher à `)`.

```

# Tracé du graphique
plt.figure(figsize=(8,5))
plt.plot(temps, noyaux, label="Décroissance du 40K")
plt.axhline(N0/2, color='r', linestyle='--', label='Moitié de N0')
plt.axvline(demi_vie, color='g', linestyle='--', label=f'Demi-vie ≈ {int(demi_vie)} Ma')
plt.xlabel("Temps (millions d'années)")
plt.ylabel("Nombre de noyaux restants")
plt.title("Décroissance radioactive du potassium-40")
plt.legend()
plt.grid()
plt.show()

```

- L'IA propose alors : *Si tu veux, je peux aussi te faire une version où tu choisis la durée des pas ou la quantité initiale directement depuis la console, façon petit TP interactif pour élèves. Ça te dirait ?*

```
# Entrée des données par l'utilisateur
N0 = int(input("Entrez le nombre initial de noyaux de 40K : "))
duree_pas = float(input("Entrez la durée d'un pas de temps (en millions d'années) : "))

# Calcul de la constante radioactive lambda
reste_pourcent = 99.45
lambda_ = -math.log(reste_pourcent / 100) / 10 # car la donnée est pour 10 Ma

# Initialisation
t = 0
N = N0
temps = [0]
noyaux = [N0]

# Recherche de la demi-vie et stockage pour le graphe
while N > N0 / 100: # on va jusqu'à 1% pour un graphe lisible
    N = N * math.exp(-lambda_ * duree_pas)
    t += duree_pas
    temps.append(t)
    noyaux.append(N)

    if N <= N0 / 2 and 'demi_vie' not in locals():
        demi_vie = t # on récupère le temps quand on atteint la moitié

# Affichage de la demi-vie
print(f"La demi-vie estimée du potassium-40 est de : {demi_vie:.1f} millions d'années")

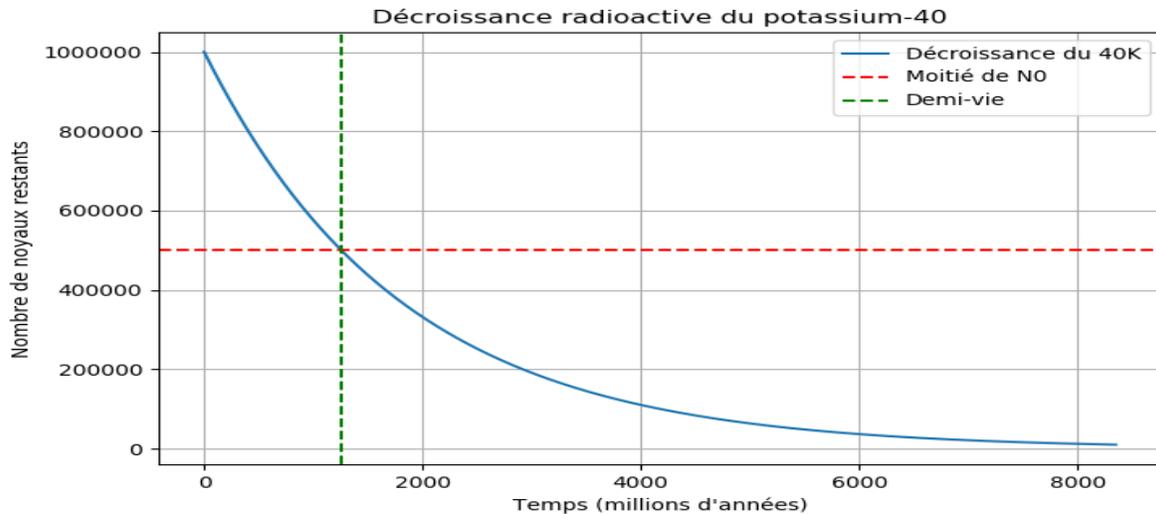
# Tracé du graphique
plt.figure(figsize=(8,5))
plt.plot(temps, noyaux, label="Décroissance du 40K")
plt.axhline(N0/2, color='r', linestyle='--', label='Moitié de N0')
# plt.axvline(demi_vie, color='g', linestyle='--', label=f'Demi-vie ≈ {int(demi_vie)} Ma')
plt.xlabel("Temps (millions d'années)")
plt.ylabel("Nombre de noyaux restants")
plt.title("Décroissance radioactive du potassium-40")
plt.legend()
plt.grid()
plt.show()
```

La aussi il a fallu modifier les 2 lignes où {demi\_vie} apparaissent.

```
# Affichage de la demi-vie
print("La demi-vie estimée du potassium-40 est de : ", demi_vie, "millions d'années")

# Tracé du graphique
plt.figure(figsize=(8,5))
plt.plot(temps, noyaux, label="Décroissance du 40K")
plt.axhline(N0/2, color='r', linestyle='--', label='Moitié de N0')
plt.axvline(demi_vie, color='g', linestyle='--', label='Demi-vie') #≈ 'demi_vie' Ma)
```

- Le résultat est probant. En plus de l'affichage de la demi-vie du potassium 40, on obtient le graphique suivant.



## Bilan

L'IA a permis rapidement et facilement de créer un programme permettant de calculer la demi-vie du potassium 40. Elle a aussi été plus loin que la consigne initiale en suggérant de compléter le programme par le tracé du graphique de la décroissance radioactive. Elle a aussi demandé si on voulait modifier le programme pour le rendre plus interactif.

Ces suggestions n'existaient pas il y a quelques temps et le développement rapide des IA laissent penser qu'elles seront de plus en plus précises à l'avenir.

Elle a également commenté les différents programmes en expliquant ce qu'elle faisait (les commentaires n'ont pas été copiés ici).

## Limites

La partie graphique ne fonctionnait pas au départ. Il y avait une erreur sur une ligne. Il fallait donc un minimum de connaissances pour la corriger.

## Annexe

```

N_0=100
i=0
N=N_0
while N>50 :
    N=
    i=
print('la demie-vie du K40 est')
print(i* )
print('millions d'années')

```