Thème: Désintégration radioactive :

Evolution du nombre de noyaux et demi-vie

Niveau: Première enseignement scientifique

TI-83 Premium CE

python™

Eric Tixidor

Fiche méthode

Référentiel, compétences

Capacités exigibles :

- Utiliser une représentation graphique pour déterminer une demi-vie.

Capacités numériques :

- Faire *un usage explicité des outils numériques*.

Commentaires de l'auteur

Les capacités numériques mises en jeu ici sont *non exigibles*. Mais cette séance permettra de consolider les notions de programmation en *langage python* étudiées dans d'autres matières de la classe de première.

Le BO suggère également de « **Mettre en œuvre des pratiques scientifiques » :** *modéliser, simuler, raisonner …* C'est cette liberté qui rend possible la réalisation de cette activité en classe.

Materiel

- Calculatrice TI-83 Premium CE Edition Python.

Prérequis

Les élèves doivent être déjà familiarisés avec l'environnement python de la calculatrice.

Enoncé

Le sujet : Dans ce travail pratique, on utilisera un script en Python pour simuler l'évolution de la population du carbone 14 au cours du temps.

Les nucléides cosmogéniques, comme le C^{14} peuvent être formés dans l'atmosphère mais aussi directement dans les minéraux des roches après impact des rayons cosmiques.

Leur population radioactive décroit par radioactivité et suit la loi de Poisson :

$$\Delta N = -\lambda \Delta t \times N$$

avec N le nombre de noyaux.

λ est la constante radioactive du radionucléide : Cette valeur représente la *probabilité de désintégration* d'un noyau par unité de temps. Pour C^{14} on a $\lambda = 0{,}000121\ an^{-1}$.

Thème: Désintégration radioactive :

Evolution du nombre de noyaux et demi-vie

Niveau: Première enseignement scientifique

TI-83 Premium CE

python™

Eric Tixidor

Fiche méthode

Cette loi exprime que la variation (diminution) **ΔN** du nombre **N** de noyaux pendant un temps court **Δt** dépend du produit :

- de la constante radioactive λ
- du temps **Δt**
- et du nombre **N** de noyaux restants.

Le temps **Δt** doit être choisi suffisamment court, pour que le produit **λ.Δt** soit inférieur à 0.1, voire, du même ordre de grandeur.

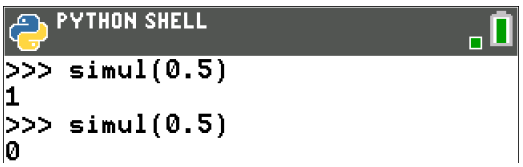
On pourra choisir, par exemple, un intervalle **Δt = 1000**. (soit 1000 ans). Ce qui donnera :

$$\lambda \times \Delta t = 0{,}000121 \times 1000 = 0{,}121$$

Cette valeur se situe alors dans la limite supérieure de validité pour le traitement suivant. Les résultats donnés par ce modèle sont toutefois assez satisfaisants. Il ne faudra pas prendre de valeur plus élevée pour **Δt.**

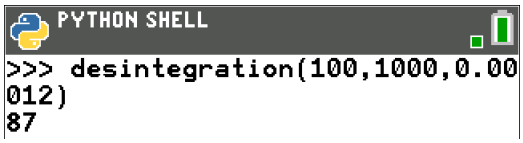
Travail à réaliser :

1) Ecrire une fonction **`simul`** qui prend comme paramètre p et qui revoie 1 avec une probabilité p et 0 avec une probabilité $1-p$.

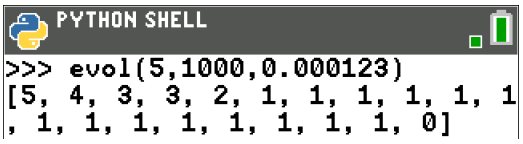


2) Ecrire une fonction **`desintegration`** qui prend comme paramètres N le nombre de noyaux (initial), dt et l (pour λ) et qui renvoie le nombre de noyaux restants après le temps dt.

Pour un temps de $dt = 1000$ la probabilité de désintégration est $p = \lambda \times dt$.



3) Ecrire une fonction **`evol`** qui prend comme paramètres N le nombre de noyaux (initial), dt et l (pour λ) et qui renvoie une liste contenant le nombre de noyaux restant à chaque pas dt (la liste doit commencer par N et se terminer par 0).



4) Utiliser les outils graphiques de la calculatrice pour déterminer le temps de demi-vie du carbone C^{14}.

Thème: Désintégration radioactive :

Evolution du nombre de noyaux et demi-vie

Proposition de résolution

Fonction `simul`

1) Par définition, la probabilité de désintégration est :

$$\mathbf{p} = \boldsymbol{\lambda} \times \boldsymbol{\Delta t}$$

On peut proposer deux façons de programmer cette fonction :

- Une première très classique qui choisit aléatoirement un réel dans l'intervalle $[0,1]$ et qui renvoie 1 si ce nombre est plus petit que la valeur de probabilité **p** et 0 sinon.

```
ÉDITEUR : RADIO
LIGNE DU SCRIPT 0008
from random import *
def simul(p):
  a=random()
  if a<=p:
    return 1
  else:
    return 0
```

- Une seconde qui consiste à choisir aléatoirement un réel dans l'intervalle $[0,1]$ et de lui ajouter **p**. Ce qui revient à choisir un réel **a** dans $[p; 1+p]$. Parmi ces nombres, ceux qui sont dans $[1,1+p]$ ont une partie entière égale à 1. Et il y a une probabilité **p** que le nombre **a** soit dans cette intervalle.

```
ÉDITEUR : RADIO
LIGNE DU SCRIPT 0006
from random import *
def simul(p):
  a=random()+p
  b=int(a)
  return b
```

Dans les deux cas, il faudra importer la librairie random à l'aide de l'instruction :

from random import $*$

Fonction `desintegration`

2) Pour chaque particule on va déterminer si elle se désintègre ou non en appelant **`simul(dt*l)`.**

Si le résultat est nul, alors la particule ne se désintègre pas,
si le résultat vaut 1 alors elle se désintègre.

Il suffit donc de retirer ce résultat à N pour obtenir le nouveau nombre de noyaux restants.

On recommence ce travail pour chaque noyau dans une boucle **`for`.**

```
ÉDITEUR : RADIO
LIGNE DU SCRIPT 0016
def desintegration(N,dt,l):
  for i in range(N):
    N=N-simul(dt*l)
  return N
```

Fonction `evol`

3) Pour obtenir la liste contenant le nombre de noyaux restant à chaque pas dt, on l'initialise avec une première valeur N : **`liste=[N].`**

Puis, tant qu'il reste des noyaux (c'est-à-dire **`while N>0`**) on calcule le nombre de noyaux restant après **dt** et on l'ajoute à la **`liste.`**

```
ÉDITEUR : RADIO
LIGNE DU SCRIPT 0021
def evol(N,dt,l):
  liste=[N]
  while N>0:
    N=desintegration(N,dt,l)
    liste.append(N)
  return liste
```

Détermination graphique du temps de demi vie

4) Ajouter au début du script la librairie ti_system :

```
from ti_system import *
```

Executer le script. Cela aura pour effet de charger les fonctions et de les rendre accessible depuis le *shell.*
Dans le *shell* : lancer la fonction avec les paramètres suivants :

- **N** : nombre de noyaux à l'instant initial : 100
- **dt** : le pas du temps discrétisé : 1000 ans
- **l** : constante radioactive : 0.000121 an^{-1}.

La fonction doit retourner la liste du nombre de radionucléides, comptés pour chaque intervalle de durée **dt**.
On devra stocker ces données dans une nouvelle liste, accessible depuis le shell, par exemple **no**.
L'instruction s'écrira alors :

```
no = eval(100,1000,0.000121)
```

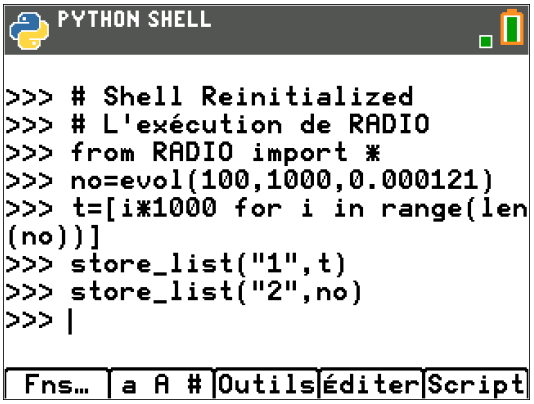
Créer alors une liste avec les repères de temps correspondants. Cette liste, appelons la **t**, aura la même longueur que **no**. Et sera remplie avec des multiples de **dt**, ce qui donne (avec **dt=1000**) :

```
t = [i*1000 for i in
range(len(no))]
```

On suppose pour la suite que les listes L_1 et L_2 du menu STAT de la calculatrice ont été au préalable effacée.

Stocker **t** et **no** dans ces deux listes grâce à la fonction **store_list** de la librairie **ti_system** :

- **store_list**("**1**", **t**)
- **store_list**("**2**", **no**)



Sortir alors de l'application python.

Thème: Désintégration radioactive :

Evolution du nombre de noyaux et demi-vie

Renseigner les listes à prendre pour les axes du graphiques L_1 et L_2 :

Dans le menu graph stat, accessible depuis [2nde] [graph statsf1 f(x)] **1:Graph1…Aff,** choisir :

Afficher la courbe : **Aff**

XListe : remplacer la liste par L1 à l'aide de la combinaison de touches [2nde] [listes stats] **1:L_1**

YListe : choisir **L_2**

Vérifier les valeurs des extrema des deux listes en parcourant les valeurs de celles-ci :

[listes stats] **1:modifier**

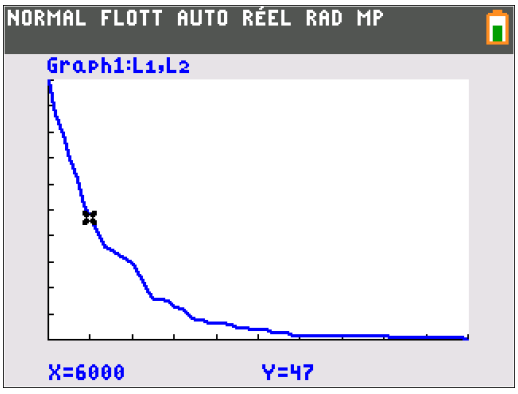
La valeur maximale pour l'axe des Y est connu : il s'agit de la valeur de **N** à l'instant initial (**no**=100).
La dernière valeur de **t** donnera la valeur maximale à renseigner pour l'axe des X. Probablement 30000 avec les conditions initiales utilisés.

Modifier les axes : [déf table f2 fenêtre], puis modifier les valeurs pour afficher au mieux la courbe à l'écran.

Puis afficher le graphique : [table f5 graphe] ainsi que le curseur sur la courbe : [calculs f4 trace]

Déterminer alors (graphiquement) la valeur du temps de demi-vie du carbone C^{14} à partir de cette simulation :

Vous devriez trouver une valeur comprise entre 5000 et 6000 ans.



Thème: Désintégration radioactive :

Evolution du nombre de noyaux et demi-vie

Modification des paramètres

Les détails de la courbe obtenue montrent que la simulation est assez imprécise. La courbe est loin d'être lisse et présente des discontinuités. De plus, l'intervalle de temps qui encadre la durée de demi-vie est assez important (intervalle de 1000 ans). On améliorera la précision de cette valeur, et la qualité de la courbe en prenant d'autres paramètres pour la fonction **`eval(N,dt,l).`** Essayer par exemple :

- ✓ **`N`** : 400
- ✓ **`dt`** : 200
- ✓ **`l`** : 0.000121

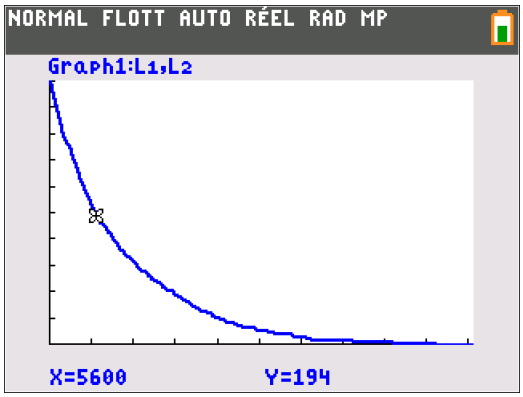
Revenir dans l'application python. Exécuter le script RADIO. Dans le shell, adapter les instructions vues précédemment pour calculer les noyaux avec ces nouveaux paramètres.
Stocker les valeurs de **`t`** et **`no`** dans les listes L_1 et L_2 de la calculatrice.

Quitter l'application python, puis :

- adapter les axes X et Y de la fenêtre graphique à partir des nouvelles valeurs obtenue
- effacer le graphique précédent pour permettre l'affichage de la nouvelle courbe à l'aide du menu DESSIN :

Le nouveau tracé doit alors apparaître à l'écran.

Le temps de demi-vie devrait avoir une valeur plus proche de celle attendue et la courbe plus lissée (valeur theorique : ***5750 ans***).



Prolongement

On pourra adapter ce script au calcul du temps de demi-vie pour d'autre radionucléides : il suffira d'adapter la valeur de la constante radioactive, l, par rapport au tableau suivant :

Et choisir une valeur de **`dt`** adaptée.

nucléides	constante radioactive (an^{-1})
H^3	0.0565
Be^{10}	$4.6*10^7$
Cl^{36}	$2.3*10^6$

Thème: Désintégration radioactive :

Evolution du nombre de noyaux et demi-vie

Script RADIO

ÉDITEUR : RADIO
LIGNE DU SCRIPT 0003

```
from random import *
from ti_system import *

def simul(p):
  a=random()+p
  b=int(a)
  return b

def desintegration(N,dt,l):
  for i in range(N):
    N=N-simul(dt*l)
  return N

def evol(N,dt,l):
  liste=[N]
  while N>0:
    N=desintegration(N,dt,l)
    liste.append(N)
  return liste
```

Fns… | a A # | Outils | Exéc | Script

Documents et script à télécharger à l'adresse :
https://education.ti.com/fr/physique-chimie