Simulación de la cadena de decaimiento del Uranio 238 y cálculo de la Ecuación Diferencial correspondiente

V. Mira Ramírez[1]

[1] *Departmento de Física Aplicada – Facultad de Ciencias, Universidad de Alicante (UA), Alicante, España.*

1 CADENA DE DECAIMIENTO

Visualizamos la cadena de decaimiento del Uranio 238 al Plomo 206, pero calcularemos el caso específico de una de las cadenas, y acoplaremos las ecuaciones diferenciales asociadas para calcular la cantidad de plomo que se genera por unidad de tiempo con una cantidad de uranio dada.

$$
\begin{aligned}
& {}^{238}_{92}\mathrm{U} \xrightarrow{\alpha} {}^{234}_{90}\mathrm{Th} + {}^{4}_{2}\mathrm{He} && \xrightarrow{\beta^-} \\
\xrightarrow{\beta^-} \quad & {}^{234}_{91}\mathrm{Pa} + {}^{4}_{2}\mathrm{He} + \mathrm{e}^- + \overline{\nu}_e && \xrightarrow{\beta^-} \\
\xrightarrow{\beta^-} \quad & {}^{234}_{92}\mathrm{U} + {}^{4}_{2}\mathrm{He} + 2\,\mathrm{e}^- + 2\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\alpha} \quad & {}^{230}_{90}\mathrm{Th} + 2\,{}^{4}_{2}\mathrm{He} + 2\,\mathrm{e}^- + 2\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\beta^-} \quad & {}^{226}_{88}\mathrm{Ra} + 3\,{}^{4}_{2}\mathrm{He} + 2\,\mathrm{e}^- + 2\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\alpha} \quad & {}^{222}_{86}\mathrm{Rn} + 4\,{}^{4}_{2}\mathrm{He} + 2\,\mathrm{e}^- + 2\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\beta^-} \quad & {}^{218}_{84}\mathrm{Po} + 5\,{}^{4}_{2}\mathrm{He} + 2\,\mathrm{e}^- + 2\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\alpha} \quad & {}^{214}_{82}\mathrm{Pb} + 6\,{}^{4}_{2}\mathrm{He} + 2\,\mathrm{e}^- + 2\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\beta^-} \quad & {}^{214}_{83}\mathrm{Bi} + 6\,{}^{4}_{2}\mathrm{He} + 3\,\mathrm{e}^- + 3\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\alpha} \quad & {}^{210}_{81}\mathrm{Tl} + 7\,{}^{4}_{2}\mathrm{He} + 3\,\mathrm{e}^- + 3\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\beta^-} \quad & {}^{210}_{82}\mathrm{Pb} + 7\,{}^{4}_{2}\mathrm{He} + 4\,\mathrm{e}^- + 4\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\alpha} \quad & {}^{210}_{83}\mathrm{Bi} + 7\,{}^{4}_{2}\mathrm{He} + 5\,\mathrm{e}^- + 5\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\beta^-} \quad & {}^{206}_{81}\mathrm{Tl} + 8\,{}^{4}_{2}\mathrm{He} + 5\,\mathrm{e}^- + 5\,\overline{\nu}_e && \xrightarrow{\alpha} \\
\xrightarrow{\alpha} \quad & {}^{206}_{82}\mathrm{Pb} + 8\,{}^{4}_{2}\mathrm{He} + 6\,\mathrm{e}^- + 6\,\overline{\nu}_e &&
\end{aligned}
$$

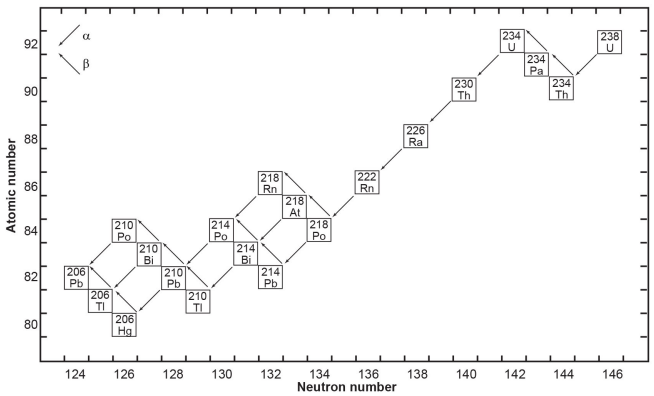

Figura 1: Esquema del decaimiento. *Dr. Andrew A. Snelling, Problems with the U-Pb Radioisotope Dating Methods—1*

2 ECUACIONES DIFERENCIALES

$$
\begin{aligned}
\frac{dN_{U_{238}}}{dt}(t) &= -\lambda_{U_{238}} N_{U_{238}}(t) \\
\frac{dN_{Th_{234}}}{dt}(t) &= \lambda_{U_{238}} N_{U_{238}}(t) - \lambda_{Th_{234}} N_{Th_{234}}(t) \\
\frac{dN_{Pa_{234}}}{dt}(t) &= \lambda_{Th_{234}} N_{Th_{234}}(t) - \lambda_{Pa_{234}} N_{Pa_{234}}(t) \\
\frac{dN_{U_{234}}}{dt}(t) &= \lambda_{Pa_{234}} N_{Pa_{234}}(t) - \lambda_{U_{234}} N_{U_{234}}(t) \\
\frac{dN_{Th_{230}}}{dt}(t) &= \lambda_{U_{234}} N_{U_{234}}(t) - \lambda_{Th_{230}} N_{Th_{230}}(t) \\
\frac{dN_{Ra_{226}}}{dt}(t) &= \lambda_{Th_{230}} N_{Th_{230}}(t) - \lambda_{Ra_{226}} N_{Ra_{226}}(t) \\
\frac{dN_{Rn_{222}}}{dt}(t) &= \lambda_{Ra_{226}} N_{Ra_{226}}(t) - \lambda_{Rn_{222}} N_{Rn_{222}}(t) \\
\frac{dN_{Po_{218}}}{dt}(t) &= \lambda_{Rn_{222}} N_{Rn_{222}}(t) - \lambda_{Po_{218}} N_{Po_{218}}(t) \\
\frac{dN_{Pb_{214}}}{dt}(t) &= \lambda_{Po_{218}} N_{Po_{218}}(t) - \lambda_{Pb_{214}} N_{Pb_{214}}(t) \\
\frac{dN_{Bi_{214}}}{dt}(t) &= \lambda_{Pb_{214}} N_{Pb_{214}}(t) - \lambda_{Bi_{214}} N_{Bi_{214}}(t) \\
\frac{dN_{Tl_{210}}}{dt}(t) &= \lambda_{Bi_{214}} N_{Bi_{214}}(t) - \lambda_{Tl_{210}} N_{Tl_{210}}(t) \\
\frac{dN_{Pb_{210}}}{dt}(t) &= \lambda_{Tl_{210}} N_{Tl_{210}}(t) - \lambda_{Pb_{210}} N_{Pb_{210}}(t) \\
\frac{dN_{Bi_{210}}}{dt}(t) &= \lambda_{Pb_{210}} N_{Pb_{210}}(t) - \lambda_{Bi_{210}} N_{Bi_{210}}(t) \\
\frac{dN_{Tl_{206}}}{dt}(t) &= \lambda_{Bi_{210}} N_{Bi_{210}}(t) - \lambda_{Tl_{206}} N_{Tl_{206}}(t) \\
\frac{dN_{Pb_{206}}}{dt}(t) &= \lambda_{Tl_{206}} N_{Tl_{206}}(t) - \lambda_{Pb_{206}} N_{Pb_{206}}(t)
\end{aligned}
$$

3 REPRESENTACIÓN GRÁFICA

Una vez tenemos el sistema de EDO, podemos buscar en la literatura el valor de las lambda de cada isótopo y usar una rutina de resolución de ecuaciones diferenciales de *python* como puede ser *odeint*. Basta con definir el sistema en una función y pasarle a *odeint* unas condiciones iniciales (una cantidad arbitraria de Uranio-238 relativamente baja), el sistema de ecuaciones y un intervalo de tiempo.

Usando como condiciones iniciales una cantidad de 10^5 núcleos de Uranio-238 y usando un intervalo de tiempo de $1{,}3^{18}$ segundos ($\approx$ 41 mil millones de años) obtenemos una animación que nos muestra la evolución de la cantidad de Uranio-238 frente a la de Plomo-206 que es el primer isótopo estable que nos encontramos en la cadena de decaimiento.

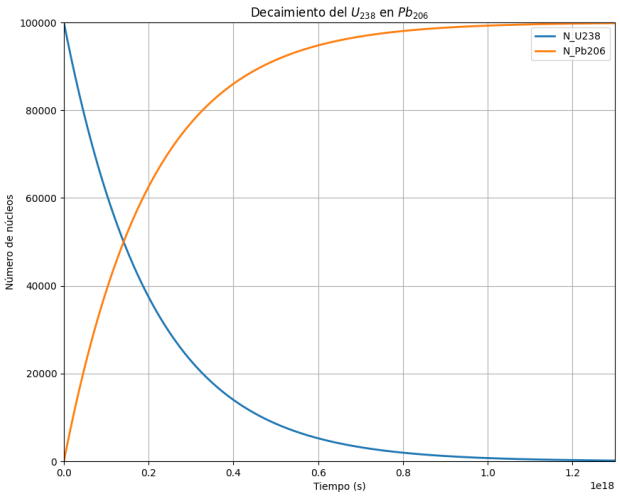

Figura 2: Núcleos de Uranio-238 y Plomo-206 en función del tiempo

Encontré un problema poniendo una cantidad inicial más grande de Uranio, que provoca que *odeint* tenga problemas con el tamaño de los números con los que tratamos. Una forma de tratar con ello es hacer que el sistema represente millones de núcleos en vez de núcleos, para así poder introducir números más bajos en el script de python.

En el script también se incluye la generación de una animación con ambas cantidades en el tiempo, pero no se puede incluir en este documento debido a las limitaciones del formato pdf.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from matplotlib.animation import FuncAnimation

# Constantes de desintegración
l_U238  = 4.916e-18
l_Th234 = 2.88e-6
l_Pa234 = 1.88e-4
l_U234  = 2.46e-5
l_Th230 = 9.19e-6
l_Ra226 = 4.33e-4
l_Rn222 = 2.10e-3
l_Po218 = 1.53e-3
l_Pb214 = 1.14e-3
l_Bi214 = 3.73e-3
l_Tl210 = 2.2e-4
l_Pb210 = 7.21e-4
l_Bi210 = 2.17e-4
l_Tl206 = 1.55e-4
l_Pb206 = 0  # estable

# Sistema de EDOs
def sistema(N, t):
  N_U238, N_Th234, N_Pa234, N_U234, N_Th230, N_Ra226,
  N_Rn222, N_Po218, N_Pb214, N_Bi214, N_Tl210,
  N_Pb210, N_Bi210, N_Tl206, N_Pb206 = N

dN_U238_dt  = -l_U238 * N_U238
dN_Th234_dt = l_U238  * N_U238  - l_Th234 * N_Th234
dN_Pa234_dt = l_Th234 * N_Th234 - l_Pa234 * N_Pa234
dN_U234_dt  = l_Pa234 * N_Pa234 - l_U234  * N_U234
dN_Th230_dt = l_U234  * N_U234  - l_Th230 * N_Th230
dN_Ra226_dt = l_Th230 * N_Th230 - l_Ra226 * N_Ra226
dN_Rn222_dt = l_Ra226 * N_Ra226 - l_Rn222 * N_Rn222
dN_Po218_dt = l_Rn222 * N_Rn222 - l_Po218 * N_Po218
dN_Pb214_dt = l_Po218 * N_Po218 - l_Pb214 * N_Pb214
dN_Bi214_dt = l_Pb214 * N_Pb214 - l_Bi214 * N_Bi214
dN_Tl210_dt = l_Bi214 * N_Bi214 - l_Tl210 * N_Tl210
dN_Pb210_dt = l_Tl210 * N_Tl210 - l_Pb210 * N_Pb210
dN_Bi210_dt = l_Pb210 * N_Pb210 - l_Bi210 * N_Bi210
dN_Tl206_dt = l_Bi210 * N_Bi210 - l_Tl206 * N_Tl206
dN_Pb206_dt = l_Tl206 * N_Tl206 - l_Pb206 * N_Pb206

return [dN_U238_dt, dN_Th234_dt, dN_Pa234_dt,
  dN_U234_dt, dN_Th230_dt, dN_Ra226_dt, dN_Rn222_dt,
  dN_Po218_dt, dN_Pb214_dt, dN_Bi214_dt, dN_Tl210_dt,
  dN_Pb210_dt, dN_Bi210_dt, dN_Tl206_dt, dN_Pb206_dt]

# Condiciones iniciales, Solo U238 ~4e-17g
N0 = [1e5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
t_max =  1.3e18 # ~41 mil millones de años
t = np.linspace(0, t_max, 1000)
solucion = odeint(sistema, N0, t)

# Animación
fig, ax = plt.subplots(figsize=(10, 8))
line_U238, = ax.plot([], [], label='N_U238', lw=2)
line_Pb206, = ax.plot([], [], label='N_Pb206', lw=2)

ax.set_xlim(0, t_max)
ax.set_ylim(0, 1e5)
ax.set_xlabel('Tiempo (s)')
ax.set_ylabel('Número de núcleos')
ax.legend(loc='upper right')
ax.grid()

def init():
    line_U238.set_data([], [])
    line_Pb206.set_data([], [])
    return (line_U238, line_Pb206)
def animate(i):
    line_U238.set_data(t[:i], solucion[:i, 0])
    line_Pb206.set_data(t[:i], solucion[:i, 14])
    return (line_U238, line_Pb206)

anim = FuncAnimation(fig, animate, init_func=init,
            frames=len(t), interval=5, blit=True)

# Gráfica del último frame
fig_final, ax_final = plt.subplots(figsize=(10, 8))
ax_final.plot(t, solucion[:, 0], label='N_U238')
ax_final.plot(t, solucion[:, 14], label='N_Pb206')

ax_final.set_xlim(0, t_max)
ax_final.set_ylim(0, 1e5)
ax_final.set_xlabel('Tiempo (s)')
ax_final.set_ylabel('Número de núcleos')
ax_final.set_title('Decaimiento del U238 en Pb206')
ax_final.legend(loc='upper right')
ax_final.grid()
plt.show()
```

Enlace al script de python que genera la gráfica y la animación: github.com/vmr48-ua

Enlace al código de LaTeX que genera este documento: www.overleaf.com