Cadenas de decaimiento radiactivo

Proyecto Final

Clemente Ferrer
`clemente.ferrer@usm.cl`

Universidad Técnica Federico Santa Maria
Departamento de Física

10 de agosto de 2022

Índice

Enunciado del problema

Radioactive decay. An initially pure sample containing N_0 atoms of ${}^{213}_{83}\mathrm{Bi}$ atoms decays according to:

$$ {}^{213}_{83}\mathrm{Bi} \rightarrow {}^{209}_{81}\mathrm{Tl} + \alpha $$

with a half-life of $45{,}6$ minutes. This is followed by the decay:

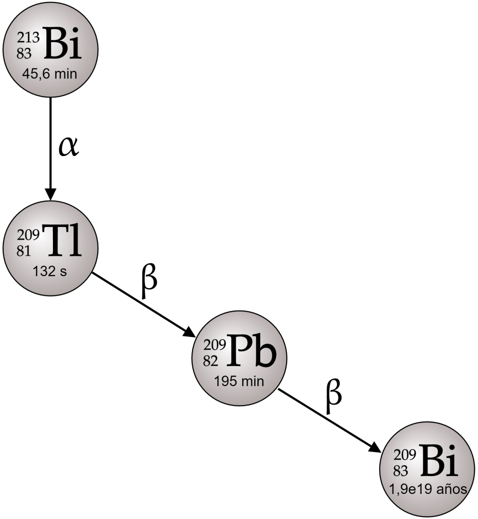
$$ {}^{209}_{81}\mathrm{Tl} \rightarrow {}^{209}_{82}\mathrm{Pb} + \beta $$

with a half-life of 132 seconds, then:

$$ {}^{209}_{82}\mathrm{Pb} \rightarrow {}^{209}_{83}\mathrm{Bi} + \beta $$

with a half-life of 195 minutes. The nuclide ${}^{209}_{83}\mathrm{Bi}$ has a half-life so long $(1{,}9 \times 10^{19}$ years) that it can be considered stable.
Obtain curves for the populations of ${}^{213}_{83}\mathrm{Bi}$, ${}^{209}_{81}\mathrm{Tl}$, ${}^{209}_{82}\mathrm{Pb}$, and ${}^{209}_{83}\mathrm{Bi}$. First derive a set of coupled first-order differential equations for $N_i, i = 1, 2, 3, 4$ for each of the four species; solve these numerically and then plot N_i/N_0.

Cadenas de decaimiento radiactivo



- Los 4 elementos son metáles.
- Emision α: núcleos de helio-4, ${}^{4}_{2}\mathrm{He}^{2+}$.
- Emision β^{-}: e^{-} y $\overline{\nu}_{\mathrm{e}}$.

Ecuaciones de Bateman

Sea N_i el número de átomos del isótopo i en el instante t medido en minutos. Luego

$$\frac{\mathrm{d}N_1}{\mathrm{d}t} = -\lambda_1 N_1,$$

donde $\lambda_i = \ln(2)/T_i$, siendo T_i la vida media del isótopo i. Observemos ahora que

$$\frac{\mathrm{d}N_2}{\mathrm{d}t} = \underbrace{\lambda_1 N_1}_{\text{Producido por } N_1} - \underbrace{\lambda_2 N_2}_{\text{Decaimiento de } N_2}$$

y de manera análoga

$$\frac{\mathrm{d}N_3}{\mathrm{d}t} = \lambda_2 N_2 - \lambda_3 N_3, \quad \frac{\mathrm{d}N_4}{\mathrm{d}t} = \lambda_3 N_3.$$

Ecuaciones de Bateman

El modelo anterior es conocido como **ecuaciones de Bateman**, cuya expresión general es

$$\begin{cases} \dfrac{\mathrm{d}N_1}{\mathrm{d}t} = -\lambda_1 N_1 \\[2ex] \dfrac{\mathrm{d}N_i}{\mathrm{d}t} = \lambda_{i-1} N_{i-1} - \lambda_i N_i \\[2ex] \dfrac{\mathrm{d}N_k}{\mathrm{d}t} = \lambda_{k-1} N_{k-1} \end{cases}$$

donde $i \in \{1, \ldots, k\}$.

En particular, de nuestro problema obtenemos el siguiente sistema de ecuaciones diferenciales acopladas

$$\begin{cases} \dfrac{\mathrm{d}N_1}{\mathrm{d}t} = -\lambda_1 N_1 \\[2ex] \dfrac{\mathrm{d}N_2}{\mathrm{d}t} = \lambda_1 N_1 - \lambda_2 N_2 \\[2ex] \dfrac{\mathrm{d}N_3}{\mathrm{d}t} = \lambda_2 N_2 - \lambda_3 N_3 \\[2ex] \dfrac{\mathrm{d}N_4}{\mathrm{d}t} = \lambda_3 N_3 \end{cases}$$

con $N_1(0) = N_0$ y $N_i(0) = 0$.

Modelo del problema y notación matricial

Ahora bien, podemos escribir el sistema de ecuaciones diferenciales usando notación matricial

$$\begin{pmatrix} N_1' \\ N_2' \\ N_3' \\ N_4' \end{pmatrix} = \begin{pmatrix} -\lambda_1 & 0 & 0 & 0 \\ \lambda_1 & -\lambda_2 & 0 & 0 \\ 0 & \lambda_2 & -\lambda_3 & 0 \\ 0 & 0 & \lambda_3 & 0 \end{pmatrix} \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{pmatrix}$$

o equivalentemente, $\boldsymbol{N'} = A\boldsymbol{N}$. En particular, $\boldsymbol{N'}(0) = (N_0, 0, 0, 0)^\top$ y

$$\lambda_1 = 0,0152 \left[\frac{1}{\text{min}}\right], \quad \lambda_2 = 0,315 \left[\frac{1}{\text{min}}\right] \text{ y } \lambda_3 = 0,00355 \left[\frac{1}{\text{min}}\right].$$

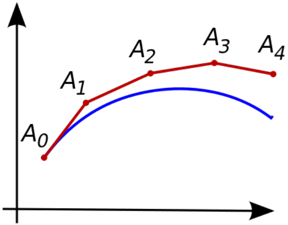
Método de Euler

El método de Euler para resolver ecuaciones del tipo

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y(t)), \quad y(t_0) = y_0.$$

Consiste en escoger un valor h como el tamaño de cada paso y definir $t_n = t_0 + nh$. Ahora bien, un paso del método de Euler desde t_n a $t_{n+1} = t_n + h$ está dado por

$$y_{n+1} = y_n + hf(t_n, y_n)$$



Implementación en C++/ROOT

Pseudocódigo:

Algorithm 1: Explicit Euler's method

Data: number N of parts in which we divide the interval $[a, b]$
Result: Sequence of values $u_0, u_1, \dots, u_n$
Function *Euler*(N)**:**

$$h := \frac{b-a}{N} \; ;$$

for $k = 1..N$ **do**

$t_k = a + hk$;

$u_{k+1} := u_k + hf(t_k, u_k)$;

return $[u_0, u_1, \dots, u_N]$;

- Los gráficos fueron generados usando ROOT.
- Para futuros análisis, los códigos fueron subidos al repositorio público `FIS205`.

Implementación método de Euler

Declaración de librerías, condiciones iniciales y primer método de Euler.

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <iomanip>
using namespace std;

const double t_inicial = 0;
const int n = 1000;
const double t_max = 100;
const double h = (t_max-t_inicial)/n;
const double c[4] = {100,0,0,0};
const double l1 = 0.0152;
const double l2 = 0.315;
const double l3 = 0.00355;

double N_1(double t,double y);
double N_2(double t,double y, vector<double> &n1, int i);
double N_3(double t,double y, vector<double> &n2, int i);
double N_4(double t,double y, vector<double> &n3, int i);
void generar(vector<double> &vect, string input);
```

```
int main() {
  double t0, y0, tn, yn, slope;
  int i;
  vector<double> t_array, n1_array, n2_array, n3_array, n4_array;

  t0 = t_inicial;
  y0 = c[0];

  /* Método de Euler N1 */
  for(i = 0; i < n; i++){
    slope = N_1(t0, y0);
    yn = y0 + h * slope;
    t_array.push_back(t0);
    n1_array.push_back(y0/c[0]);
    y0 = yn;
    t0 = t0+h;
  }

  t0 = t_inicial;
  y0 = c[1];

```

Implementación método de Euler

Métodos de Euler restantes.

```
/* Método de Euler N2 */
for(i=0; i < n; i++){
  slope = N_2(t0, y0, n1_array, i);
  yn = y0 + h * slope;
  n2_array.push_back(y0/c[0]);
  y0 = yn;
  t0 = t0+h;
}

t0 = t_inicial;
y0 = c[2];

/* Método de Euler N3 */
for(i=0; i < n; i++){
  slope = N_3(t0, y0, n2_array, i);
  yn = y0 + h * slope;
  n3_array.push_back(y0/c[0]);
  y0 = yn;
  t0 = t0+h;
}

t0 = t_inicial;
y0 = c[3];

/* Método de Euler N4 */
for(i=0; i < n; i++){
  slope = N_4(t0, y0, n3_array, i);
  yn = y0 + h * slope;
  n4_array.push_back(y0/c[0]);
  y0 = yn;
  t0 = t0+h;
}

generar(t_array, string("times"));
generar(n1_array, string("N1"));
generar(n2_array, string("N2"));
generar(n3_array, string("N3"));
generar(n4_array, string("N4"));
}
```

Implementación método de Euler

Funciones auxiliares.

```
double N_1(double t,double y){
  return -l1*y;
}

double N_2(double t,double y, vector<double> &n1, int i){
  return l1*n1.at(i)-l2*y;
}

double N_3(double t,double y, vector<double> &n2, int i){
  return l2*n2.at(i)-l3*y;
}

double N_4(double t,double y, vector<double> &n3, int i){
  return l3*n3.at(i);
}

```

```
void generar(vector<double> &vect, string input){
  ofstream myfile (input);
  if (myfile.is_open()){
    for(int count = 0; count < n; count ++){
      if (count == n-1)
        myfile << vect.at(count);
      else
        myfile << vect.at(count) << "\n";
    }
    myfile.close();
  }
}
```

Hasta este punto, se han creado cinco archivos de texto diferentes: `times.txt`, `N1.txt`, `N2.txt`, `N3.txt` y `N4.txt`.

Archivos de texto generados

- Los archivos se encargan de almacenar los valores de N_i para cada partición de tiempo.
- El número de lineas es igual a n (número de pasos).

.

	N1	N2	N3	N4	times
1	1	0	0	0	0
2	0.99848	1.52e-05	0	0	0.1
3	0.996962	2.98981e-05	4.788e-09	0	0.2
4	0.995447	4.41101e-05	1.42042e-08	1.69974e-14	0.3
5	0.993934	5.78515e-05	2.80938e-08	6.74223e-14	0.4
6	0.992423	7.11369e-05	4.63071e-08	1.67155e-13	0.5
7	0.990915	8.39809e-05	6.86988e-08	3.31546e-13	0.6
8	0.989408	9.63974e-05	9.51284e-08	5.75426e-13	0.7
9	0.987904	0.0001084	1.2546e-07	9.13132e-13	0.8
10	0.986403	0.000120001	1.59561e-07	1.35851e-12	0.9
11	0.984904	0.000131215	1.97305e-07	1.92496e-12	1
12	0.983406	0.000142052	2.38568e-07	2.62539e-12	1.1
13	0.981912	0.000152525	2.83229e-07	3.47231e-12	1.2
14	0.980419	0.000162646	3.31174e-07	4.47777e-12	1.3
15	0.978929	0.000172425	3.8229e-07	5.65344e-12	1.4
16	0.977441	0.000181873	4.36468e-07	7.01057e-12	1.5
17	0.975955	0.000191001	4.93603e-07	8.56003e-12	1.6
18	0.974472	0.000199819	5.53593e-07	1.03123e-11	1.7
19	0.972991	0.000208337	6.1634e-07	1.22776e-11	1.8
20	0.971512	0.000216564	6.81747e-07	1.44656e-11	1.9

Implementación en ROOT

plot.c

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

void plot(int pasos, string file, string file2) {
  TCanvas *c1 = new TCanvas("c1","Grafico",200,10,700,500);
  Double_t x[pasos], y[pasos];
  Int_t i=0;
  Int_t j=0;
  string linea;
  ifstream archivo(file.c_str());
  while (getline(archivo, linea)) {
    x[i] = stod(linea);
    i=i+1;
  }
  archivo.close();
  ifstream archivo2(file2.c_str());
  while (getline(archivo2, linea)) {
    y[j] = stod(linea);
    j=j+1;
  }
  archivo2.close();
  TGraph* gr = new TGraph(pasos,x,y);
  gr->SetTitle(" ");
  gr->SetLineColor(2);
  gr->SetLineWidth(2);
  gr->Draw("AC");
  c1->SaveAs("ploteo.png");
}
```

Ejecución por consola del código anterior:

```
Microsoft Windows [Versión 10.0.19042.1415]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\ccfer>ssh -XY exphys11@ui.hpc.utfsm.cl
exphys11@ui.hpc.utfsm.cl's password:
Last login: Sat Dec 25 17:30:14 2021 from 170.82.189.242
----------------------------------------------
          Bienvenidos al cluster HPC!
          Welcome to the HPC cluster!
            http://www.hpc.utfsm.cl
----------------------------------------------
(base) [exphys11@ui02 ~]$ cd cf
(base) [exphys11@ui02 cf]$ use root
Package "root" is ready
(base) [exphys11@ui02 cf]$ root -v
   ------------------------------------------------------------------
  | Welcome to ROOT 6.22/06                        https://root.cern |
  | (c) 1995-2020, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Nov 27 2020, 15:14:08                 |
  | From tags/v6-22-06@v6-22-06                                      |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0] .L plot.c
root [1] plot(1000, string("times.txt"), string("N1.txt"))
Info in <TCanvas::Print>: png file ploteo.png has been created
```

Gráficas de N_i/N_0

A continuación se muestran las gráficas generadas por ROOT:

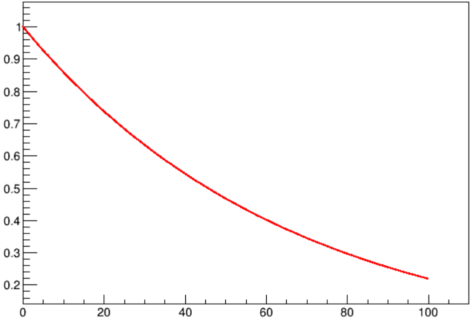


Figura: N_1/N_0.

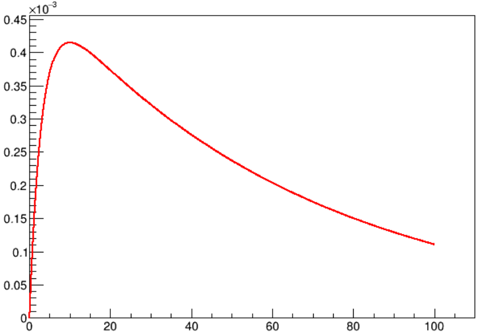


Figura: N_2/N_0.

Gráficas de N_i/N_0

A continuación se muestran las gráficas generadas por ROOT:

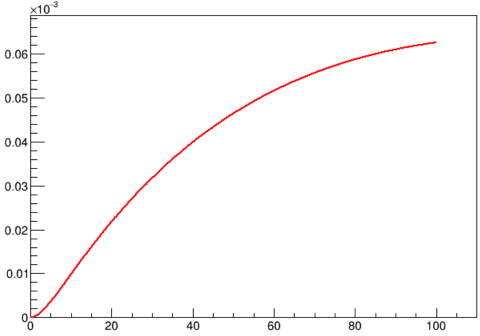


Figura: N_3/N_0.

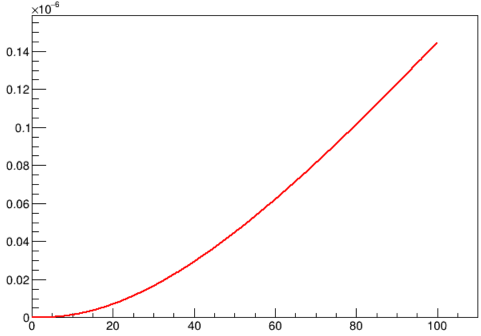


Figura: N_4/N_0.

Conclusiones

- El decaimiento de ${}^{213}_{83}\mathrm{Bi}$ se produce más rápido que cualquiera de los otros isótopos.
- El número de átomos de ${}^{209}_{81}\mathrm{Tl}$ tiende a crecer hasta alcanzar un máximo local y luego decaer.
- La convexidad de la curva creciente de ${}^{209}_{82}\mathrm{Pb}$ vislumbra intenciones de estabilizarse.
- Finalmente, la concavidad hacia arriba creciente que presenta ${}^{209}_{83}\mathrm{Bi}$ denota la estabilidad enunciada del problema.

Para **futuros trabajos** relacionados, podría considerarse una cadena de decaimiento radiactiva más larga, resuelta con otro método númerico.

Bibliografía

- H. Bateman.
  Solution of a system of differential equations occurring in the theory of radioactive transformations.
  Proc. Cambridge Phil. Soc., 15:423–427, 1910.

- J. F. Boudreau, R. M. Bianchi, and E. S. Swanson.
  Applied computational physics.
  Oxford University Press, 2018.

- H. D. Young, R. A. Freedman, A. L. Ford, and H. D. Young.
  Sears and Zemansky's University physics with modern physics.
  Pearson Education, 2020.

Cadenas de decaimiento radiactivo

Proyecto Final

Clemente Ferrer
clemente.ferrer@usm.cl

Universidad Técnica Federico Santa Maria
Departamento de Física

10 de agosto de 2022