Bateman Equation: The Polonium Problem

University of Bologna

Theoretical and Numerical Aspects of Nuclear Physics

A.A. 2021/22

Introduction: the Bateman Equation

The Bateman equations are a set of first order ordinary differential equations of the form:

$$n'(t) = An(t) \qquad n(0) = n_0$$

n(t): nuclide concentration vector
A: Bateman matrix
n_0: initial nuclide concentration

Exact solution $\longrightarrow$ $n(t) = e^{At}n_0$

The Polonium Problem [1]

A simple example of the Bateman equation is represented by the Polonium-210 production which is expressed in the following chain:

$$^{209}\mathrm{Bi} \xrightarrow{(n,\gamma)} {}^{210}\mathrm{Bi} \xrightarrow{5{,}013d} {}^{210}\mathrm{Po} \xrightarrow{138{,}376d} {}^{210}\mathrm{Pb}$$

Lead-210 is a stable nuclide. The change in the nuclides concentration over time is:

$$\frac{dn_{Bi209}}{dt} = -\lambda_{(Bi209)} n_{Bi209}$$

$$\frac{dn_{Bi210}}{dt} = \lambda_{(Bi209)} n_{Bi209} - \lambda_{(Bi210)} n_{Bi210}$$

$$\frac{dn_{Po210}}{dt} = \lambda_{(Bi210)} n_{Bi210} - \lambda_{(Po210)} n_{Po210}$$

$\lambda_{nuclide}$: *decay constant of the nuclide*

The Polonium Problem

The Bateman equations can be rewritten in the matrix form:

$$\frac{d}{dt}\begin{pmatrix} n_{Bi209} \\ n_{Bi210} \\ n_{Po210} \end{pmatrix} = \begin{pmatrix} -\lambda_{Bi209} & 0 & 0 \\ \lambda_{Bi209} & -\lambda_{Bi210} & 0 \\ 0 & \lambda_{Bi210} & \lambda_{Po210} \end{pmatrix}\begin{pmatrix} n_{Bi209} \\ n_{Bi210} \\ n_{Po210} \end{pmatrix}$$

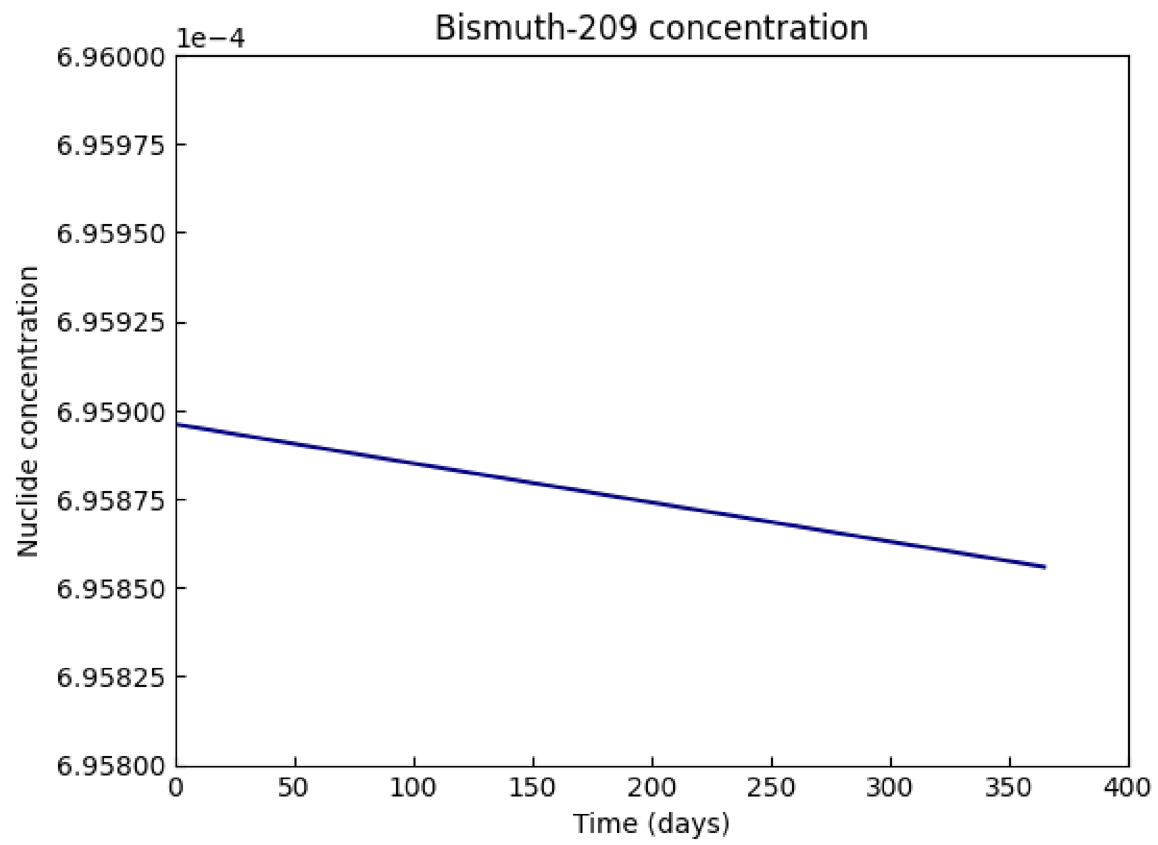
The matrix on the right-hand side is the Bateman matrix, which is used to find the nuclides concentration over a given period of time using the matrix exponential method on Python (v. 3.9.4).

The Polonium Problem

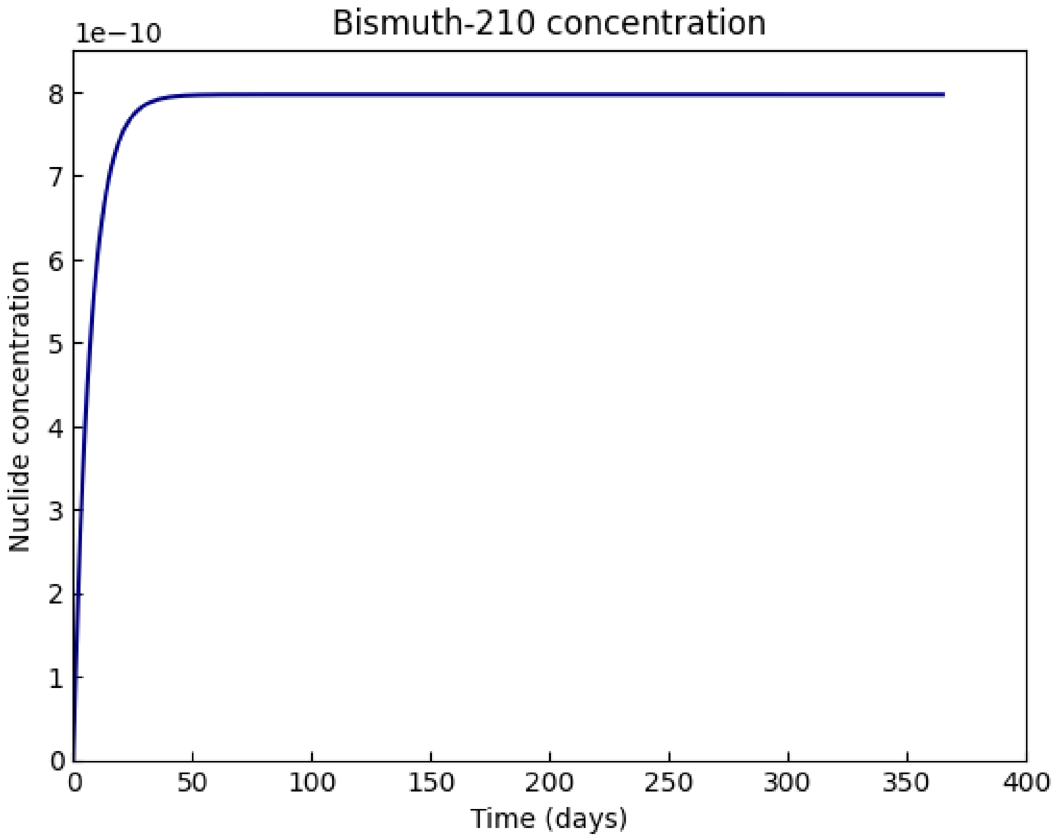
In the following table are the constants used in the Bateman equations:

λ_{Bi209}	$1.83163 \cdot 10^{-12}\ s^{-1}$
λ_{Bi210}	$1.60035 \cdot 10^{-6}\ s^{-1}$
λ_{Po210}	$5.79764 \cdot 10^{-8}\ s^{-1}$
$n_{Bi209}(0)$	$6.95896 \cdot 10^{-4}\ cm^{-3}$
$n_{Bi210}(0)$	$0\ cm^{-3}$
$n_{Po210}(0)$	$0\ cm^{-3}$

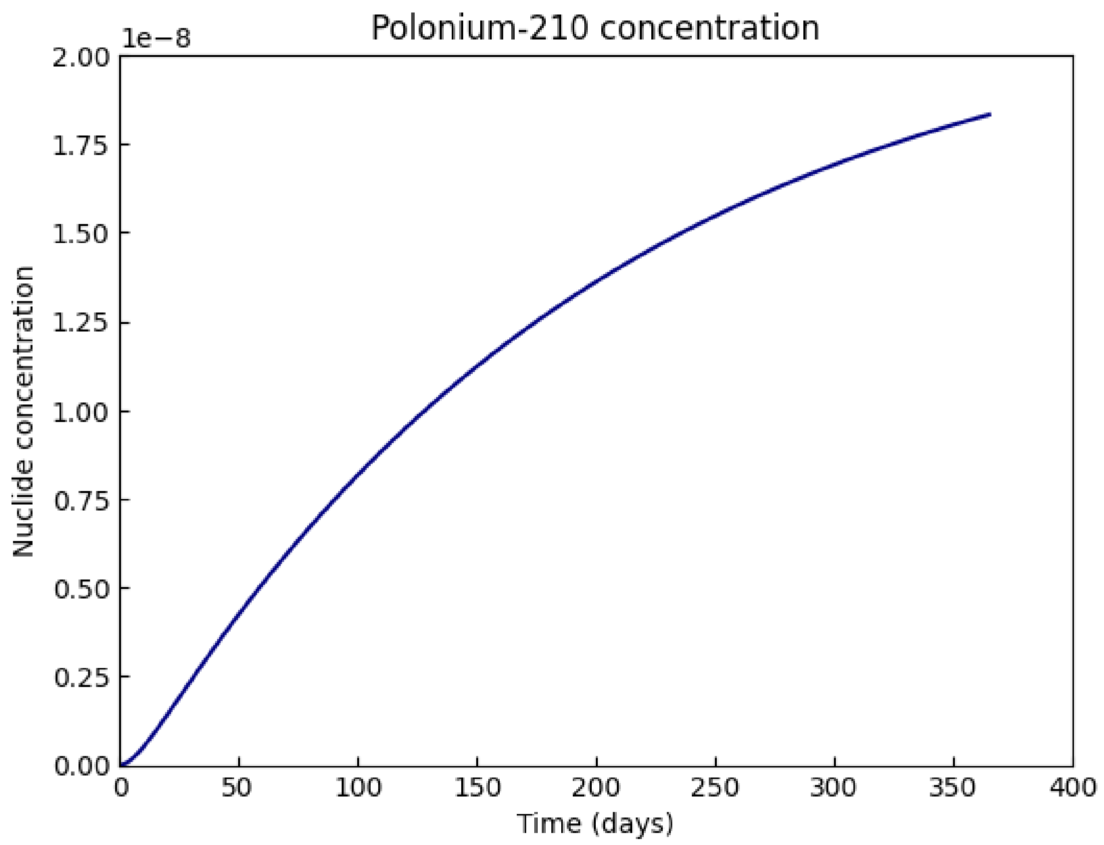
The Polonium Problem: Results



The Polonium Problem: Results



The Polonium Problem: Results



Conclusions

The resulting plots show the variation in the nuclide concentration over a period of 365 days of the different nuclides.

The results are compatible with what is expected to happen: Bismuth-209 decreases over time while Bismuth-210 and Polonium-210 both increases starting from an initial concentration of 0.

Bibliography

[1] Master Thesis: Implementation, validation and comparison of different algorithms to solve the Bateman equations for very large systems – Vranckx Maren 2015/16

Appendix: Python code

```python
def polonium_problem():

    # Decay constants of the nuclides
    lambdaBi_209 = 1.83163e-12
    lambdaBi_210 = 1.60035e-6
    lambdaPo_210 = 5.79764e-8

    # Time unit: 1 day
    T = 365 # Simulate for one year (365 days)
    dt = 60*60*24 # seconds in a day

    # Initial concentration of the different nuclides
    Bi209_0 = 6.95896e-4
    Bi210_0 = 0
    Po210_0 = 0
    init_value = [Bi209_0, Bi210_0, Po210_0]

    # Bateman Matrix
    A = np.array([[-lambdaBi_209, 0, 0],
                  [lambdaBi_209, -lambdaBi_210, 0],
                  [0, lambdaBi_210, -lambdaPo_210]])

    # Compute the solution to the Bateman's system
    bateman_sol = matrix_exponential_method(init_value,dt,T,A)
```

```python
    Bi209 = bateman_sol[:,0]
    Bi210 = bateman_sol[:,1]
    Po210 = bateman_sol[:,2]

    #Print results
    print_results(T, Bi209, Bi210, Po210)

    # Plots
    plot(0,69580.0e-8,69600.0e-8,Bi209,'Bismuth-209 concentration')
    plot(1,0,8.5e-10,Bi210,'Bismuth-210 concentration')
    plot(2,0,2e-8,Po210,'Polonium-210 concentration')

    plt.show()

if __name__ == '__main__':
    polonium_problem()
```

Python implementation of the Polonium problem

Appendix: Python code

```python
def matrix_exponential_method(U_0,dt,T,A):
    '''...

    # Definition of u matrix
    u = np.zeros((T+1, len(U_0)))

    # Fill u with initial values
    u[0] = U_0

    # Iteration
    for i in range(T):
        u[i+1] = linalg.expm((dt*A)).dot(u[i])

    return u
```

Python implementation of the matrix exponential method

Appendix: Python code

```python
def print_results(T, Bi209, Bi210, Po210):

    table = []

    for i in range(T):
        table.append([i, Bi209[i], Bi210[i], Po210[i]])

    print("Results")
    print(" ")
    print(tabulate(table, headers=['t', 'Bismuth-209', 'Bismuth-210', 'Polonium-210']))
    print(" ")

```

Results and plots

```python
def plot(i,y_min,y_max,y,title):
    plt.figure(i)
    plt.ylim(y_min,y_max)
    plt.xlim(0, 400)
    plt.plot(y, '#000080')
    plt.title(title)
    plt.xlabel('Time (days)')
    plt.ylabel('Nuclide concentration')
    plt.ticklabel_format(axis = "y", style = "sci", scilimits=(0,0))
    plt.tick_params('both', direction = 'in')

```