ODE for Radioactive Decay

Kaylin Shanahan 2023

This is a python programme to estimate the number of nuclei remaining in a radioactive sample of Cobalt-60 over the next 20 years. This will be done using Euler's technique and display the numerical and exact analytical solutions on a graph.

The rate at which the number of nuclei in a radioactive sample decay is proportional to the number of radioactive nuclei remaining in the sample, as described by the following differential equation:

$$\frac{dN}{dt} = -\lambda N$$

Where N is the number of radioactive nuclei, t is the time and λ is the decay constant.

This formula can then be rearranged for the number of radioactive nuclei N remaining after time t, which will be used to provide the analytical solution:

$$N_t = N_0 e^{-\lambda t}$$

λ, the probability of a given nucleus decaying in one second, is related to the half-life by the following equation:

$$t_{\frac{1}{2}} = \frac{ln(2)}{\lambda}$$

This formula can then be rearranged for the decay constant λ:

$$\lambda = -\frac{\ln(2)}{t_{half}}$$

This particular radioactive sample, Cobalt-60, initially contains 10^{10} nuclei and has a half-life of 5.272 years.

- Input the initial conditions

- Determine the number of steps required
- Initialise x and y values
- Calculate the remaining nuclei using the differential equation above
- Output the x and y values
- Output a plot of the numerical and analytical solutions on a graph

In [142…

```python
# Solve ODE for radioactive decay using Euler's Technique
# dN/dt = -lambda * N
# t_1/2 = ln*2 / lambda

# Numpy is needed for the natuaral log and exponential function
import numpy as np

t_half_years = 5.272                                    # Half-life of Cobalt-60 (years)
t_half_secs = t_half_years*3.154e+7                     # Half-life of Cobalt-60 (secs)
decay_const_secs = -((np.log(2))/(t_half_secs))         # Calculating decay constant (secs)
deacy_const_years = decay_const_secs/(3.154*(10**7))    # Calculating decay constant (years)
Nc_0 = 10 ** 10                                         # Initial amount of radioactive nuclei

t_year_max = 20                                         # Iterate up to a maximum time of t_year_max (years)
delta_t_year = 1                                        # Set the time step in years
N = int(t_year_max/delta_t_year)                        # Calculate the number of time jumps in years

# Set up NumPy arrays to hold the Nuclei Nc and time t values and initialise
# We need one element in each array, than the number of jumps
Nc = np.zeros(N + 1)
t = np.zeros(N + 1)
Nc[0] = Nc_0
t[0] = 0

# Use a for loop to step along the t-axis
for i in range(N):
    slope = -decay_const * Nc[i]                       # Calculation for slope, the rate of radioactive nuclei decay
    Nc[i+1] = Nc[i] + slope * delta_t_year             # Number of nuclei remaining after time t
    t[i+1] = t[i] + delta_t_year                       # Time elapsed

# Print the t and Nc values
for i in range(N+1):
    print("i = {0:3}, t = {1:6.1f}, Nc = {2:6.2f}".format(i, t[i], Nc[i]))
```

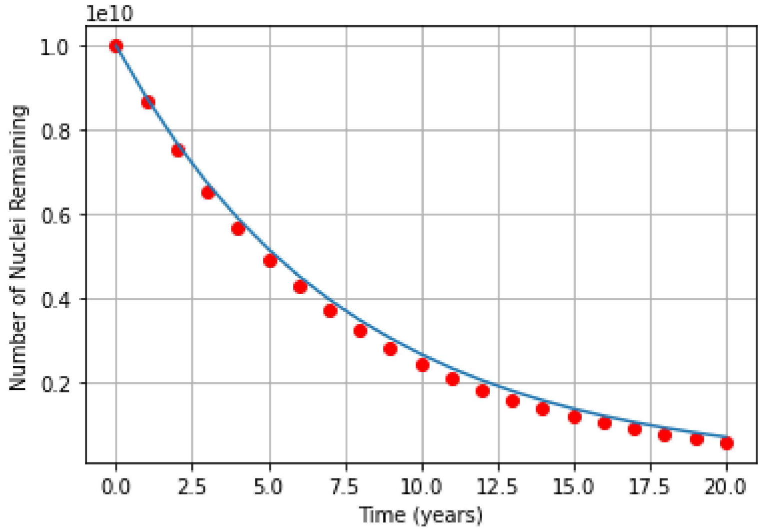
```
i =   0, t =    0.0, Nc = 10000000000.00
i =   1, t =    1.0, Nc = 8685230000.00
i =   2, t =    2.0, Nc = 7543322015.29
i =   3, t =    3.0, Nc = 6551548666.69
i =   4, t =    4.0, Nc = 5690170702.64
i =   5, t =    5.0, Nc = 4942044129.17
i =   6, t =    6.0, Nc = 4292278993.20
i =   7, t =    7.0, Nc = 3727943028.01
i =   8, t =    8.0, Nc = 3237804262.51
i =   9, t =    9.0, Nc = 2812107471.49
i =  10, t =   10.0, Nc = 2442380017.46
i =  11, t =   11.0, Nc = 2121263219.91
i =  12, t =   12.0, Nc = 1842365895.54
i =  13, t =   13.0, Nc = 1600137154.69
i =  14, t =   14.0, Nc = 1389755922.01
i =  15, t =   15.0, Nc = 1207034982.65
i =  16, t =   16.0, Nc = 1048337644.24
i =  17, t =   17.0, Nc = 910505355.78
i =  18, t =   18.0, Nc = 790794843.12
i =  19, t =   19.0, Nc = 686823509.53
i =  20, t =   20.0, Nc = 596522014.97
```

In [143…

```
# Matplotlib is needed to plot graphs
import matplotlib.pyplot as plt

# Plot the numerical solution as point
plt.plot(t, Nc, "ro")
plt.xlabel("Time (years)")                          # X-axis label
plt.ylabel("Number of Nuclei Remaining")            # Y-axis label
plt.grid()                                          # Graph grid

# Plot the analytical solution as a line
# Use the same (N+1) t values
Nc_analytic = Nc[0] * np.exp(-decay_const * t)    # Calculation of the nuber of nuclei
plt.plot(t, Nc_analytic)                            # Plotting time versus number of nuclei
plt.show()                                          # Display graph
```



In [144…

```
# Calculate fractional accuracy
Nc_frac_accuracy = (Nc - Nc_analytic) / Nc_analytic

# Plot the graph of fractional accuracy versus time
plt.plot(t, Nc_frac_accuracy)
plt.xlabel("Time (years)")                          # X-axis label
plt.ylabel("Fractional accuracy")                   # Y-axis label
plt.grid()                                          # Graph grid
```