HACETTEPE UNIVERSITY

Department of Nuclear Engineering

NEM 393 ENGINEERING PROJECT II

Multi-stage decay chain analysis of uranium.

Student No. : **2230386062**

Student Name : **Emre Sakarya**

Delivery Date: 26.10.2025

Due Date: 26.10.2025

1 ABSTRACT

This project investigates the decay chain from Uranium 238 to Lead 206, as well as other isotopes in this decay chain with half-lives longer than one year, using both analytical and numerical methods.

In the first part of the project, an analytical solution using the Bateman equation was performed to determine the present-day masses of uranium isotopes at the beginning of the Earth. The result not only ensures conservation of mass, but also the mass fraction of uranium from the beginning of the Earth to the present is the same as the mass fraction calculated in the third part of the project when we calculated the mass of Uranium 238 from the mass of Lead 206. This ensures accuracy.

In the second part, a numerical solution was implemented. This section demonstrates the importance of the time interval chosen for the numerical solution. Furthermore, no single numerical method was used; calculations were made using the Implicit Euler method in addition to the Explicit Euler method, and the results were compared. This project demonstrates how the decay chain should be calculated in general, the considerations to be taken into account when calculating the chosen method, and highlights their importance.

2 INTRODUCTION

Radioactive isotopes generally do not decay directly into stable isotopes. Instead, they transform into new isotopes by emitting alpha and beta scattering reactions. If the newly formed isotopes are also radioactive, this process continues until the isotope becomes stable. This process can be complex and lengthy because each isotope has its own unique properties. One of the best-known examples is Uranium-238. U-238 undergoes a series of decay chains to form the stable isotope Lead-206 (Pb-206). This chain contains many intermediate isotopes with very different half-lives, ranging from seconds to billions of years.

Such half-life, isotope studies are used in many processes, from determining the age of the Earth or a substance on Earth to managing the waste released in nuclear reactors.

In this project, assuming that all uranium at the time of Earth's formation (approximately 4.503 billion years ago) was U-238 and that no other isotopes existed, the present-day U-238 amount is given in kilograms.

Therefore, the first part of the project determined the amount of uranium in the early Earth and its present-day isotopes. This solution is based on an analytical solution using Bateman's equations.

In the second part of the project, numerical solutions were used for the uranium mass at the beginning of the Earth and the isotope mass at present, again using Bateman's equations. These differential equations were discretized at specific time intervals (Δt = 100 years, 10,000 years, and 1,000,000 years) to solve the problem. This allowed for a comparison between the analytical and numerical solutions. Furthermore, the numerical solution was further analyzed internally based on the selected Δt value.

Finally, as additional example, a different approach was adopted; analytical solutions for each isotope were made by considering the mass of Pb-206 on Earth today, and the mass of Uranium-238 at the beginning of Earth and today was calculated.

3 METHODS AND CALCULATIONS

The project assumed that there were 2.892 x 10^15 kg of U-238 on Earth, all of which were U-238. The Earth was estimated to be 4.503 billion years old. Therefore, in the first part of the project, the differential equations were solved analytically using the Bateman equations below, and the results were obtained.

Radioactive decay is a process, and radioactive isotopes have a definite half-life. The half-life is the average time required for half of the atoms in a sample to decay. The rate of decay, depending on the atomic number and the decay constant, is expressed as follows:

$$\frac{dN_1}{d_t} = -\lambda_1 N_1$$

The decomposition constant varies depending on the substance, but is formulated as follows:

$$\lambda = \frac{ln\,2}{t_{1/2}}$$

The equation used to solve this differential equation is as follows:

$$N_1(t) = N_1(0) \cdot \left(e^{(-\lambda_1 t)}\right)$$

And finally, the general form for the k th isotope in the chain is:

$$\frac{dN_k}{d_t} = \lambda_{k-1}N_{k-1} - \lambda_k N_k$$

After finding the value of : N_k , the mass of each isotope is calculated as follows:

$$m_{k(t)} = \frac{N_k(t)A_{(k)}}{N_A}$$

$A_{(k)}$: Atomic mass of the isotope (g/mol)

N_A: $Avagadro\ number$ = 6.022×10^23

In the second part of the project, the initial values were fixed, but the solution was done numerically rather than analytically.

In this method, derivative expressions are converted to finite difference form:

$$\frac{N(t+\Delta t)-N(t)}{\Delta t}=\lambda_{k-1}N_{k-1}(t)-\lambda_k N_k(t)$$

In the third part of the project, the analytical equations in the first part were used.

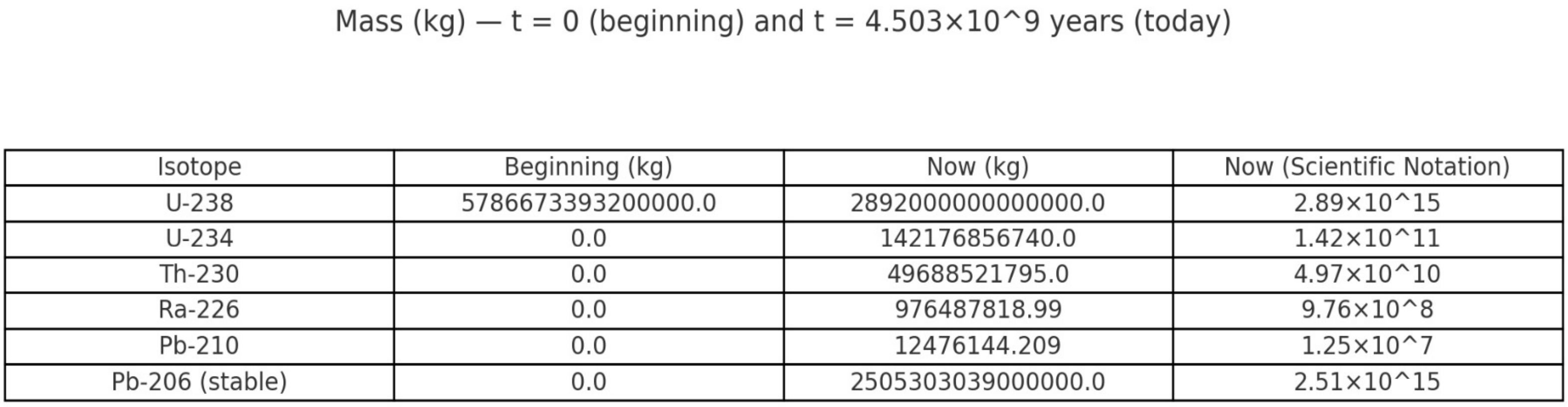
4 RESULTS

This section examines the results obtained from analytical solutions. Initially, only U-238 was assumed to be present, and the masses of all other isotopes were set to zero. The calculations determined both the mass of uranium at the beginning of the world and the masses of the isotopes in the chain now, (4.503×10^9 years after the beginning of the world).

The mass of U-238 at Earth's beginning and the present-day masses of the other isotopes are clearly shown in the table below. Total mass conservation was checked, and the total mass of the system was found to be constant. This conservation was tested during the verification of the results.

Table 1:This table showing the mass of U-238 and its isotopes in the beginning the world and today, both in real terms and scientifically.

.



Mass (kg) — t = 0 (beginning) and t = 4.503×10^9 years (today)

Isotope	Beginning (kg)	Now (kg)	Now (Scientific Notation)
U-238	5786673393200000.0	2892000000000000.0	2.89×10^15
U-234	0.0	142176856740.0	1.42×10^11
Th-230	0.0	49688521795.0	4.97×10^10
Ra-226	0.0	976487818.99	9.76×10^8
Pb-210	0.0	12476144.209	1.25×10^7
Pb-206 (stable)	0.0	2505303039000000.0	2.51×10^15

In the graph below, we see the masses of uranium and its isotopes as a bar chart. The important point to note about this graph is that it is logarithmic. In a linear graph, the distance between 1 and 2 units is the same as the distance between 10 and 11 units. In a logarithmic graph, the distance between 1 and 10 is the same as the distance between 10 and 100 (the ratio is 10 times). The reason for using this method is purely for legibility. A linear graph wouldn't tell us much because the mass difference between the isotopes is so enormous

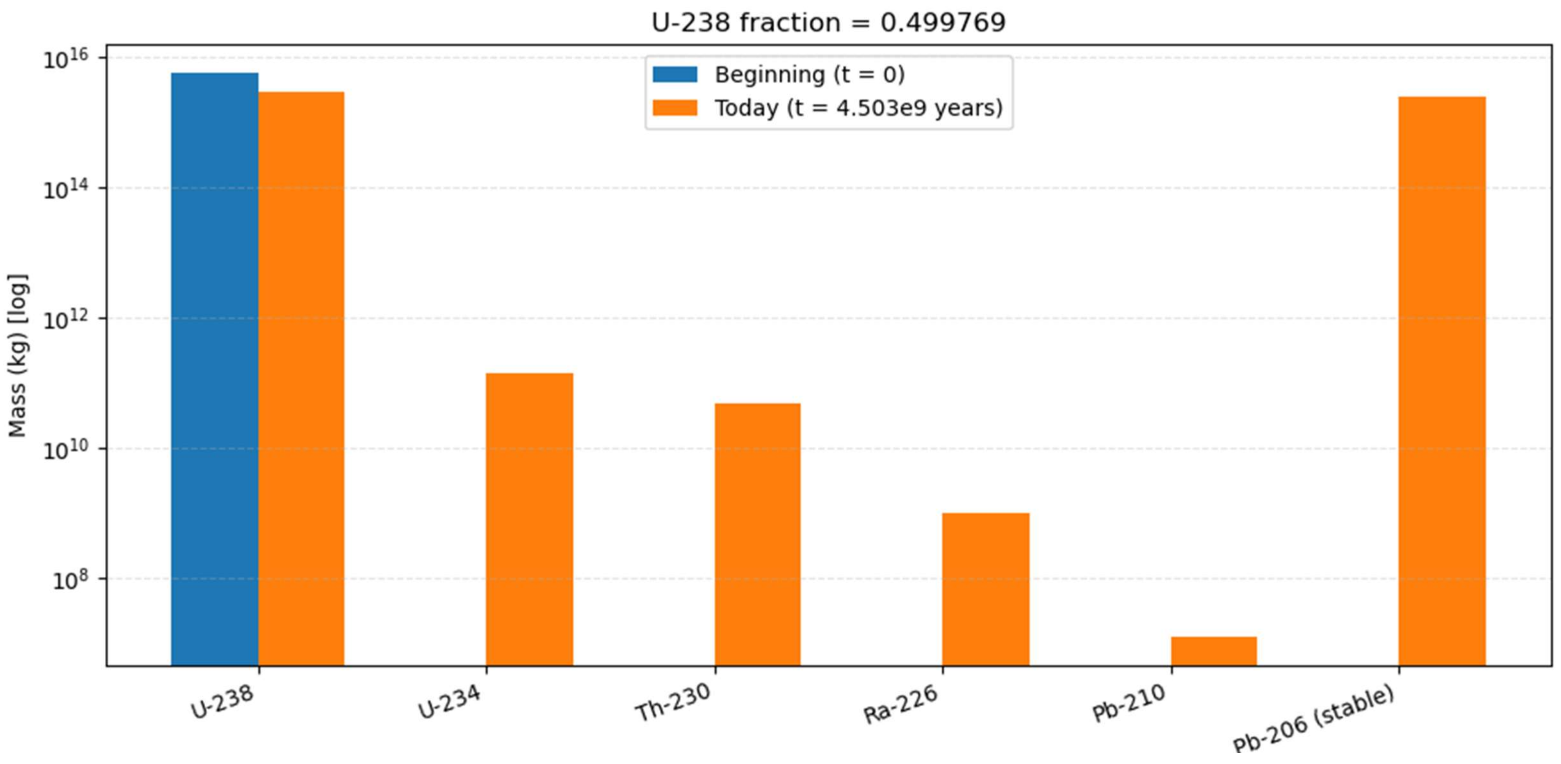


Figure 1:This graph shows the masses of uranium and its isotopes today and at the beginning of the Earth on a log scale.

In this and the graph below, we can clearly see the mass change of uranium and its isotopes over time. As expected, Pb-206 has been increasing continuously since its inception because it is stable. In addition, the graphs show that Ra-226 and Pb-210, with their shorter half-lives, decay more rapidly, thus creating less accumulation in this mass balance.

Furthermore, the decrease in uranium mass is equal to the total increase in other isotopes.

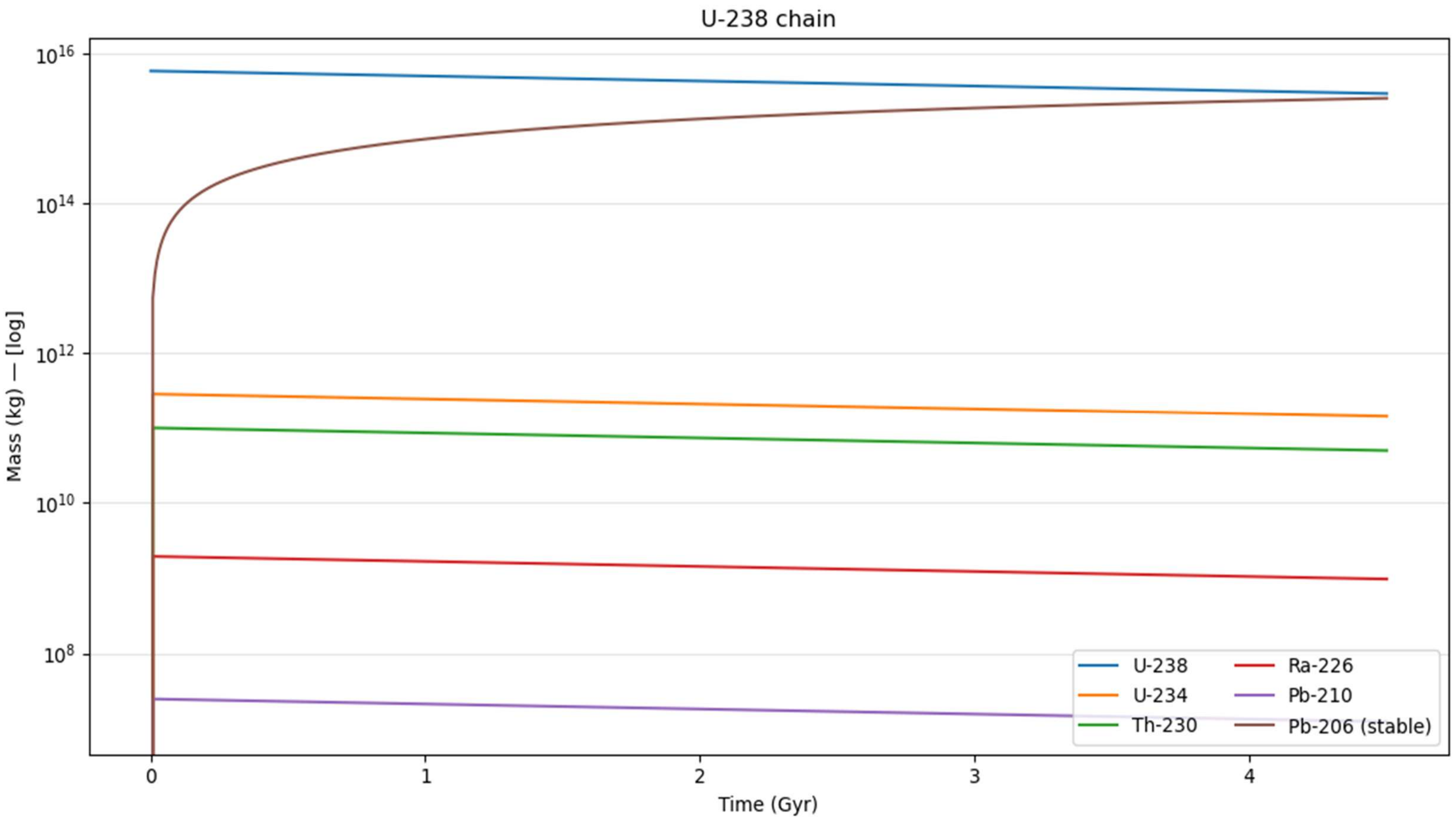


Figure 2: Graph showing the mass change of U238 and some isotopes released in the chain reaction of U238 with time on a logarithmic scale.

While we see that the uranium mass has almost halved in Table 1, we cannot quite see it in the graph. Therefore, the importance of both the table and the graph in a project becomes even more apparent here. Below is an additional graph, primarily showing U-238, to illustrate that the uranium mass has nearly halved.

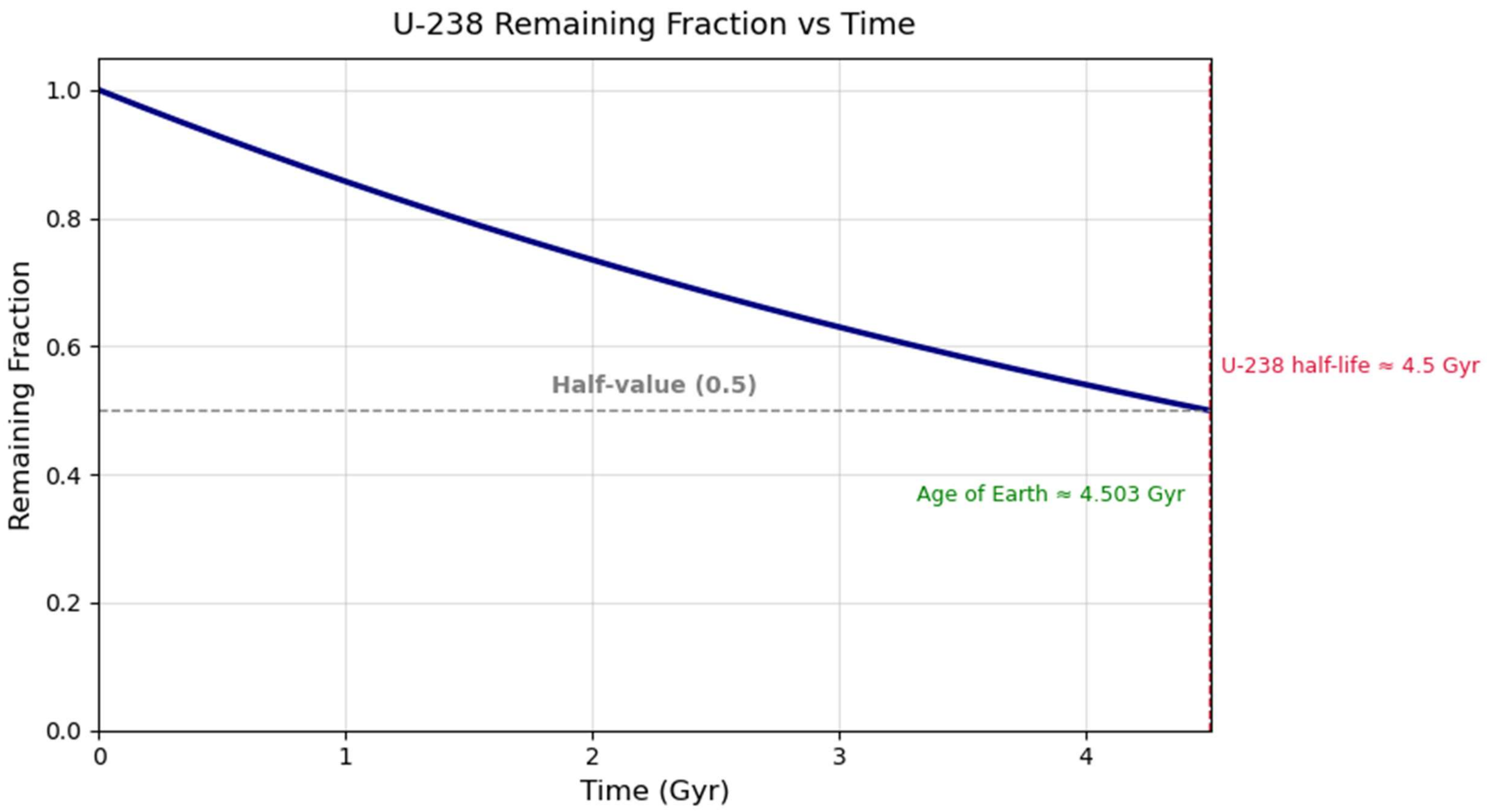


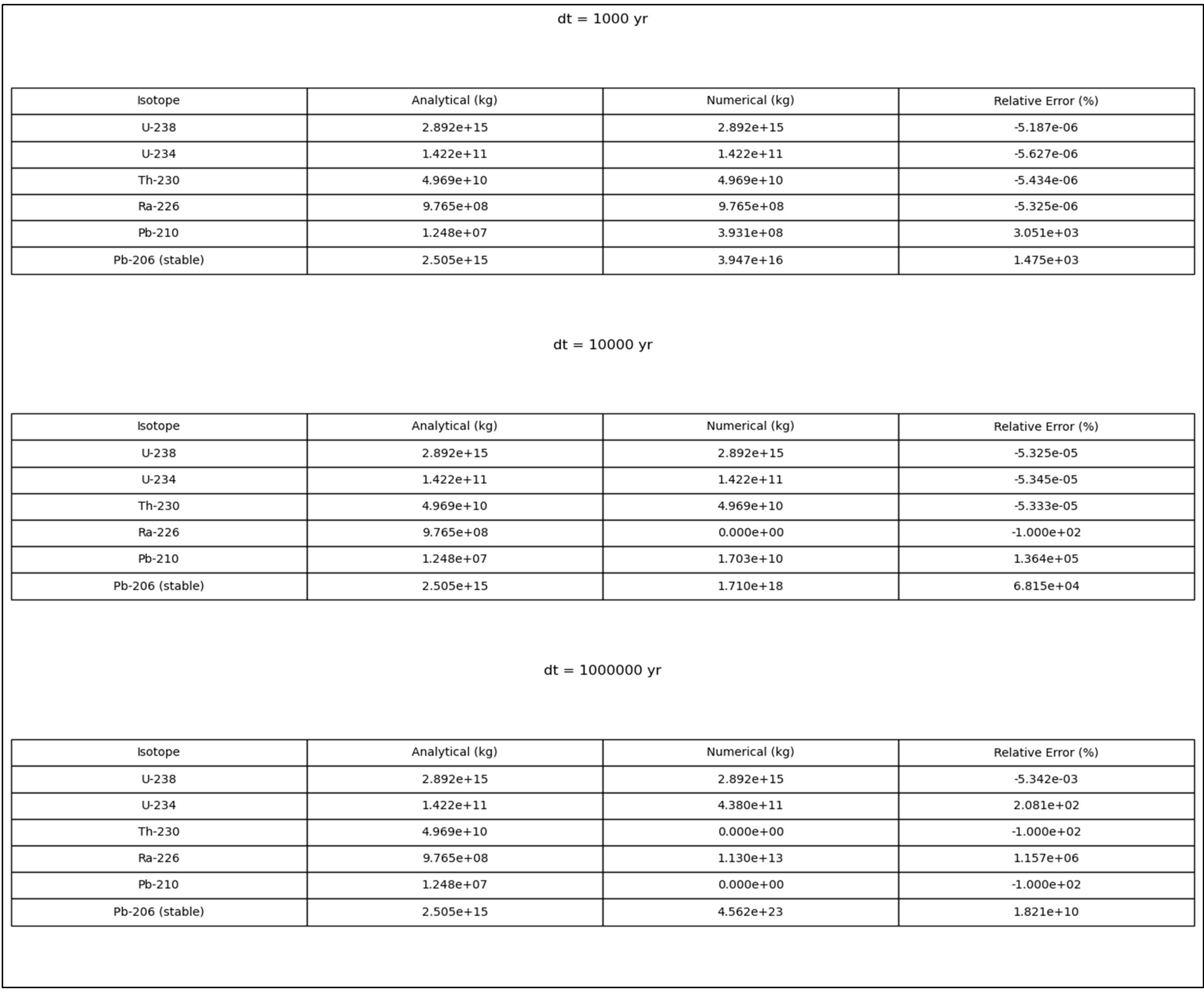
Figure 3: Graph showing the mass fraction of Uranium-238 versus time

While in some cases the results were almost identical to the analytical ones, in some cases very serious deviations were observed depending on the isotope and the chosen Δt.

This numerical solution is used in the following equation:

$$\frac{N(t+\Delta t)-N(t)}{\Delta t}=\lambda_{k-1}N_{k-1}(t)-\lambda_k N_k(t)$$

Table 2 :Table showing the masses of isotopes found by numerical methods for varying Δt values, along with the relative error.



dt = 1000 yr

Isotope	Analytical (kg)	Numerical (kg)	Relative Error (%)
U-238	2.892e+15	2.892e+15	-5.187e-06
U-234	1.422e+11	1.422e+11	-5.627e-06
Th-230	4.969e+10	4.969e+10	-5.434e-06
Ra-226	9.765e+08	9.765e+08	-5.325e-06
Pb-210	1.248e+07	3.931e+08	3.051e+03
Pb-206 (stable)	2.505e+15	3.947e+16	1.475e+03

dt = 10000 yr

Isotope	Analytical (kg)	Numerical (kg)	Relative Error (%)
U-238	2.892e+15	2.892e+15	-5.325e-05
U-234	1.422e+11	1.422e+11	-5.345e-05
Th-230	4.969e+10	4.969e+10	-5.333e-05
Ra-226	9.765e+08	0.000e+00	-1.000e+02
Pb-210	1.248e+07	1.703e+10	1.364e+05
Pb-206 (stable)	2.505e+15	1.710e+18	6.815e+04

dt = 1000000 yr

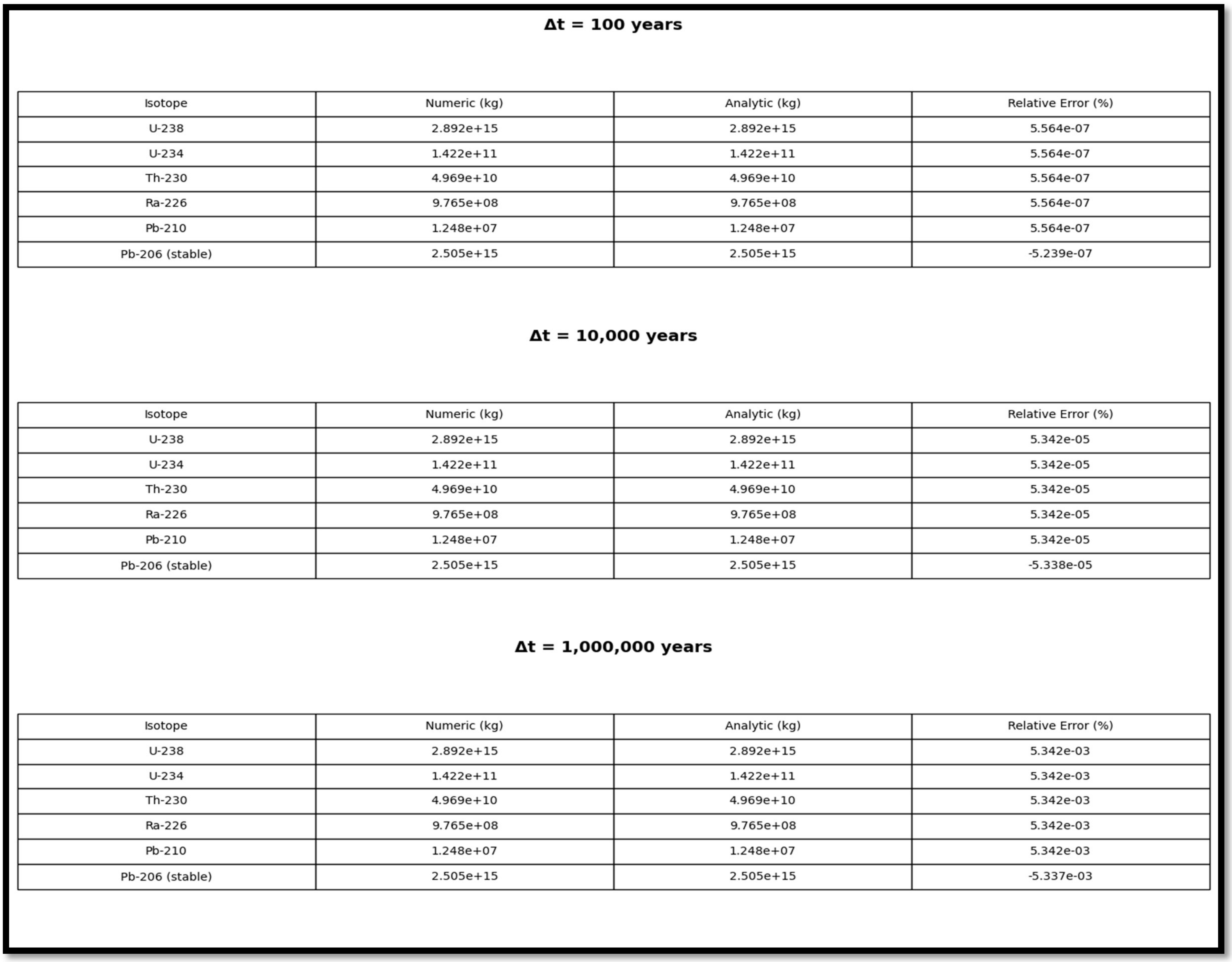
Isotope	Analytical (kg)	Numerical (kg)	Relative Error (%)
U-238	2.892e+15	2.892e+15	-5.342e-03
U-234	1.422e+11	4.380e+11	2.081e+02
Th-230	4.969e+10	0.000e+00	-1.000e+02
Ra-226	9.765e+08	1.130e+13	1.157e+06
Pb-210	1.248e+07	0.000e+00	-1.000e+02
Pb-206 (stable)	2.505e+15	4.562e+23	1.821e+10

There are two main factors that cause the errors in this numerical result to be high: the chosen Δt and the half-life of the isotopes. According to the equation and solution used in the question, the isotope must meet the condition $0<\lambda\Delta t<2$ in order to give a stable result in the numerical solution.

For example, when the time interval is 1000 years, the numerical and analytical solutions agree with each other except for the Pb-210 isotope. This discrepancy is due to the relatively short half-life of Pb-210. Consequently, the λ value increases, and accurate results cannot be obtained. The error in the Pb 206 isotope increases each time because it is a stable isotope and we can see that it is directly affected by the decays in other isotopes.

The answer to the question is given in Table 2. However, if we also want an exact numerical solution, we can do so using more detailed differential equations. The results in Table 2 are called the Forward Euler method. For a more detailed and complex solution, the Implicit Euler method is used. The results are as follows.

Table 3: The table shows the results of numerical analysis of the time-dependent change of isotope masses according to the Implicit Euler method.



Δt = 100 years

Isotope	Numeric (kg)	Analytic (kg)	Relative Error (%)
U-238	2.892e+15	2.892e+15	5.564e-07
U-234	1.422e+11	1.422e+11	5.564e-07
Th-230	4.969e+10	4.969e+10	5.564e-07
Ra-226	9.765e+08	9.765e+08	5.564e-07
Pb-210	1.248e+07	1.248e+07	5.564e-07
Pb-206 (stable)	2.505e+15	2.505e+15	-5.239e-07

Δt = 10,000 years

Isotope	Numeric (kg)	Analytic (kg)	Relative Error (%)
U-238	2.892e+15	2.892e+15	5.342e-05
U-234	1.422e+11	1.422e+11	5.342e-05
Th-230	4.969e+10	4.969e+10	5.342e-05
Ra-226	9.765e+08	9.765e+08	5.342e-05
Pb-210	1.248e+07	1.248e+07	5.342e-05
Pb-206 (stable)	2.505e+15	2.505e+15	-5.338e-05

Δt = 1,000,000 years

Isotope	Numeric (kg)	Analytic (kg)	Relative Error (%)
U-238	2.892e+15	2.892e+15	5.342e-03
U-234	1.422e+11	1.422e+11	5.342e-03
Th-230	4.969e+10	4.969e+10	5.342e-03
Ra-226	9.765e+08	9.765e+08	5.342e-03
Pb-210	1.248e+07	1.248e+07	5.342e-03
Pb-206 (stable)	2.505e+15	2.505e+15	-5.337e-03

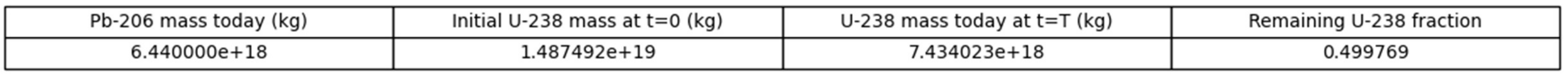
In the third part of the project, a specific amount of Pb-206 was given and, accordingly, each isotope was included in the process, as in the first question, and two things were asked:

1. The mass of Uranium 238 at the beginning of the world

2. The mass of Uranium 238 today

What I find important here is that the mass fraction of uranium 238 has almost halved, as in Figure 3. Even if our starting values are different, the ratios do not change if the calculations are correct.

Table 4: Table showing the mass and proportion of uranium from the beginning of the world to the present day, with data obtained from the present amount of lead, using the analytical solution method.



Pb-206 mass today (kg)	Initial U-238 mass at t=0 (kg)	U-238 mass today at t=T (kg)	Remaining U-238 fraction
6.440000e+18	1.487492e+19	7.434023e+18	0.499769

5 CONCLUSION

In summary, this study analyzed many aspects of the U-238 → … → Pb-206 decay chain. Both analytical and numerical results were obtained. These results were compared internally, and some errors were identified. The results are as follows.

The first part of the project addressed the analytical Bateman solution. The resulting change in the mass of the isotopes was reported. These results were consistent with conservation of mass, and no errors were detected. Although no errors were detected, the U-238 mass fraction was found for reconfirmation in the third part of the project. Part 3 presented a reverse approach to the question in the first part. Similar calculations were performed in reverse, with different values. Consequently, the mass fraction for U-238 remained unchanged, even though the numerical values were different. This demonstrates that the Bateman analytical solution was correct in both parts 1. and 3. of the project.

A numerical approach was applied to the second part of the problem. This numerical approach was performed over three different time intervals, and the results were examined. It was observed that the accuracy of the solution, or rather the accuracy with which it approximates the result, depends on both the half-life of the isotope and the chosen time interval. Because we cannot change the properties of the isotope or sample used in this and likely many other problems, the chosen time interval should be kept as short as possible. As the time interval increases, both the error rate increases and, for some isotopes, measurements become impossible. This project clearly demonstrates the importance of the time interval in numerical solutions.

A different approach to the second question is the Implicit Euler approach. Unlike the former, it is more precise because it calculates the derivative at the next time step and directly incorporates the future state into the equation. This yield results much closer to reality. Of course, this process is much more difficult and complex than the first. The critical point here, as in all areas of engineering, is efficiency. Depending on the task at hand, one of these two analytical approaches can be used. For situations where the margin of error in numerical results is acceptable or for isotopes with very long half-lives, the Explicit method, which is less energy and time-consuming, can be used. For isotopes with no margin of error or with very short half-lives, similar to the project, the Implicit Euler method should be used. After that, it all comes down to calculating efficiency, determining the work to be done, and the energy consumed.

6 REFERENCES

- Agency, I. A. (2019). *Nuclear Data Services*. Retrieved from https://www-nds.iaea.org/
- Bateman, H. (1910). The solution of a system of differential equations occurring in the theory of radioactive transformations. *Proceedings of the Cambridge Philosophical Society*, 423-427.
- Center, N. N. (2024). *Chart of Nuclides*. https://www.nndc.bnl.gov/nudat3/ adresinden alındı
- Chapman, S. J. (2017, pp. 63-66). Fortran for Scientists and Engineers. McGraw-Hill.

- Chapra, S. C., & Canale, R. P. (2015). *Numerical Methods for Engineers*. McGraw-Hill.
- Knoll, G. F. (2010). *Radiation Detection and Measurement* (4th ed.). Wiley.
- Lamarsh, J. R., & Baratta, A. J. (2001). *Introduction to Nuclear Engineering* (3rd ed.). Prentice Hall.

7 APPENDIX

```
import numpy as np
import math

# --- CONFIGURATION & CONSTANTS ---
try:
    # Use higher precision if available
    DT = np.longdouble
except AttributeError:
    DT = np.float64

EPS_REL = DT(1e-28)
EPS_ABS = DT(0.0)
EPS_ZERO_DIV = DT(1e-99)

NA = DT('6.02214076e23')   # Avogadro constant (mol^-1)
G_PER_KG = DT(1000)        # grams per kg
T_END = DT('4.503e9')      # Age of Earth (years)

# Isotope Data: (Name, Molar Mass [g/mol], Half-life [years])
ISOTOPES = [
    ("U-238",           DT(238.0), DT('4.5e9')),
    ("U-234",           DT(234.0), DT('2.25e5')),
    ("Th-230",          DT(230.0), DT('8.0e4')),
    ("Ra-226",          DT(226.0), DT('1.6e3')),
    ("Pb-210",          DT(210.0), DT('22.0')),
    ("Pb-206 (stable)", DT(206.0), np.inf),
]

NAMES = [n for n, _, _ in ISOTOPES]
A_MASS = np.array([a for _, a, _ in ISOTOPES], dtype=DT)
```

```
T12 = np.array([t for _, _, t in ISOTOPES], dtype=object)

# Decay Constants (lambda = ln(2) / t_1/2)
LAM = np.array([
    (DT(math.log(2.0)) / DT(t)) if np.isfinite(t) else DT(0)
    for t in T12
], dtype=DT)

NUM_ISOTOPES = len(LAM)

# --- HELPER FUNCTIONS ---

def mass_to_atoms(mass_kg, molar_mass):
    """Converts mass (kg) to number of atoms."""
    moles = (mass_kg * G_PER_KG) / molar_mass
    return moles * NA

def atoms_to_mass(N, molar_mass):
    """Converts number of atoms to mass (kg)."""
    moles = N / NA
    return (moles * molar_mass) / G_PER_KG

def clamp_nonnegative(x, ref):
    """Clips tiny negative values caused by floating point errors."""
    tol = EPS_ABS + EPS_REL * abs(ref)
    return DT(0) if (x < 0 and abs(x) <= tol) else x

# --- SOLVERS ---

def bateman_solution(t, lam, N0_parent):
    """
    Computes analytical solution using Bateman equations.
    Assumes only the first parent (N0) is present at t=0.
    """
    N = np.zeros(NUM_ISOTOPES, dtype=DT)

    if t == 0:
        N[0] = N0_parent
        return N
        # Parent Nucleus
        N[0] = N0_parent * np.exp(-lam[0] * t)

        # Daughter Nuclei
        for k in range(1, NUM_ISOTOPES - 1):
            num = DT(1)
            for r in range(k):
                num *= lam[r]

            total_sum = DT(0)
            for j in range(k + 1):
                denom = DT(1)
                for r in range(k + 1):
                    if r != j:
                        diff = lam[r] - lam[j]
                        # Avoid division by zero for very close decay
constants
                        denom *= diff if abs(diff) > DT(1e-100) else
DT(1e-100)

                total_sum += np.exp(-lam[j] * t) / denom

            N[k] = N0_parent * num * total_sum

        # Stable End Product (Conservation of atoms)
        N[-1] = N0_parent - np.sum(N[:-1], dtype=DT)

        return N

    def forward_euler_solver(dt, t_total, lam, N0_parent):
        """
        Numerical solution using Explicit (Forward) Euler method.
        """
        steps = int(np.ceil(t_total / dt))
        N = np.zeros(NUM_ISOTOPES, dtype=DT)
        N[0] = N0_parent

        for _ in range(steps):
            prev = N.copy()

            # Parent
            N[0] = prev[0] + (-lam[0] * prev[0]) * dt
            if N[0] < 0: N[0] = DT(0)

            # Daughters
            for k in range(1, NUM_ISOTOPES - 1):
                production = lam[k-1] * prev[k-1]
                decay = lam[k] * prev[k]
                N[k] = prev[k] + (production - decay) * dt
                if N[k] < 0: N[k] = DT(0)

            # Stable Product
            N[-1] = prev[-1] + (lam[NUM_ISOTOPES-2] * prev[NUM_ISOTOPES-
2]) * dt

        return N

    # --- MAIN EXECUTION ---

    def run_analysis():
        print("=== PROJECT 1: URANIUM SERIES DECAY ANALYSIS ===\n")

        # --- PART 1: MASS EVOLUTION (Analytic) ---
        M_NOW_U238_KG = DT('2.892e15')
        N_now_U238 = mass_to_atoms(M_NOW_U238_KG, A_MASS[0])

        # Back-calculate initial atoms at t=0
        N0_U238 = N_now_U238 * np.exp(LAM[0] * T_END)

        N_start = bateman_solution(DT(0), LAM, N0_U238)
        N_now = bateman_solution(T_END, LAM, N0_U238)

        m_start = [atoms_to_mass(N_start[i], A_MASS[i]) for i in
range(NUM_ISOTOPES)]
        m_now = [atoms_to_mass(N_now[i], A_MASS[i]) for i in
range(NUM_ISOTOPES)]

        print("--- Q1: Analytic Masses (kg) ---")
        print(f"{'Isotope':<18} {'At Formation (t=0)':<25} {'Today
(t=4.5e9)':<25}")
        for i in range(NUM_ISOTOPES):
            print(f"{NAMES[i]:<18}                    {float(m_start[i]):.6e}
{float(m_now[i]):.6e}")
```

```
        print("\n")

        # --- PART 2: NUMERICAL vs ANALYTIC (Euler Method) ---
        print("--- Q2: Numerical Stability Analysis (Forward Euler) ---")
        dt_list = [DT('1000'), DT('10000'), DT('1000000')]

        ref_mass = np.sum(m_now) # Conservation of mass check

        for dt in dt_list:
            print(f"\n[Time Step dt = {int(dt)} years]")

            # Stability Check
            unstable_isotopes = []
            for i in range(NUM_ISOTOPES - 1):
                if float(LAM[i] * dt) >= 1.0:
                    unstable_isotopes.append(NAMES[i])
            if unstable_isotopes:
                print(f"Warning: Solution likely unstable for: {',
'.join(unstable_isotopes)}")

            N_num = forward_euler_solver(dt, T_END, LAM, N0_U238)
            m_num = [atoms_to_mass(N_num[i], A_MASS[i]) for i in
range(NUM_ISOTOPES)]

            print(f"{'Isotope':<18} {'Analytic (kg)':<18} {'Numeric
(kg)':<18} {'Rel Err (%)':<18}")
            for i in range(NUM_ISOTOPES):
                a = float(m_now[i])
                n = float(m_num[i])
                diff = abs(n - a) / a * 100 if a > 0 else 0.0
                print(f"{NAMES[i]:<18} {a:.4e}                    {n:.4e}
{diff:.4f}%")

        # --- PART 3: BACK-CALCULATION FROM PB-206 ---
        print("\n--- Q3: Reverse Calculation (Pb-206 -> U-238) ---")

        M_PB206_TODAY = DT('6.44e18')
        N_Pb206_today = mass_to_atoms(M_PB206_TODAY, A_MASS[-1])

        # Calculate what fraction of U-238 has decayed into Pb-206 by
today
```

```
        # We use Bateman to find a[-1] which is the coefficient for the
last daughter
        # Note: Using simplified scalar logic for the last chain member
        N_normalized = bateman_solution(T_END, LAM, DT(1.0))
        fraction_converted = N_normalized[-1]

        # Calculate Initial U-238 required to produce this much Pb-206
        N0_required = N_Pb206_today / fraction_converted
        M_U238_start_calc = atoms_to_mass(N0_required, A_MASS[0])

        # Calculate remaining U-238 today from that start
        M_U238_today_calc = M_U238_start_calc * np.exp(-float(LAM[0]) *
float(T_END))

        print(f"Given Pb-206 mass:   {float(M_PB206_TODAY):.6e} kg")
        print(f"Calculated U-238(0): {float(M_U238_start_calc):.6e} kg")
        print(f"Calculated U-238(T): {float(M_U238_today_calc):.6e} kg")

    if __name__ == "__main__":
        run_analysis()
```

```
program q2_numeric_implicit
    implicit none

    ! --- Variable Declarations ---
    integer, parameter :: dp = selected_real_kind(15, 307)
    integer, parameter :: m  = 6  ! Number of isotopes (U-238 -> ... -> Pb-
206)

    ! Isotope Data
    character(len=15), parameter :: names(m) = [ &
        'U-238          ', &
        'U-234          ', &
        'Th-230         ', &
        'Ra-226         ', &
        'Pb-210         ', &
        'Pb-206 (stable)']

    real(dp), parameter :: A_molar(m) = [238.0_dp, 234.0_dp, 230.0_dp,
226.0_dp, 210.0_dp, 206.0_dp]
```

```
    real(dp), parameter :: t12(m)       = [4.5e9_dp, 2.25e5_dp, 8.0e4_dp,
1.6e3_dp, 22.0_dp, huge(1.0_dp)]

    real(dp) :: lam(m), ln2
    integer  :: i, j

    ! Constants
    real(dp), parameter :: NA       = 6.02214076e23_dp
    real(dp), parameter :: G_PER_KG = 1000.0_dp
    real(dp), parameter :: T_END    = 4.503e9_dp

    ! Initial Condition (Mass of U-238 today)
    real(dp), parameter :: M_today_U238_kg = 2.892e15_dp

    ! Arrays for calculation
    real(dp) :: N0(m), N_T_num(m), N_T_ana(m)
    real(dp) :: M0(m), M_T_num(m), M_T_ana(m)
    real(dp) :: N_today_U238, N0_parent

    ! Time steps to test
    real(dp), parameter :: dts(3) = [1.0e6_dp, 1.0e4_dp, 1.0e2_dp]
    integer :: nsteps

    ! --- Initialization ---
    ln2 = log(2.0_dp)
    do i = 1, m
        if (t12(i) < huge(1.0_dp)) then
            lam(i) = ln2 / t12(i)
        else
            lam(i) = 0.0_dp
        end if
    end do

    ! 1. Calculate Initial Parent (N0) at t=0 based on Today's U-238
    N_today_U238 = masskg_to_atoms(M_today_U238_kg, A_molar(1))
    N0_parent    = N_today_U238 * exp(lam(1) * T_END)

    ! Set initial vector (only U-238 exists at t=0)
    N0 = 0.0_dp
    N0(1) = N0_parent
    ! Calculate Analytic Solution for Reference
    call bateman_solution(T_END, lam, N0_parent, N_T_ana)

    ! Convert analytic results to mass
    do i = 1, m
        M_T_ana(i) = atoms_to_masskg(N_T_ana(i), A_molar(i))
    end do

    print *, '--- IMPLICIT EULER SIMULATION ---'

    ! --- Main Loop for Different Time Steps ---
    do j = 1, 3
        print *, ' '
        print '(A, ES10.1, A)', '>>> Simulation with dt = ', dts(j), '
years'

        call implicit_euler_forward(N0, dts(j), nsteps, N_T_num)

        ! Convert results to mass
        do i = 1, m
            M_T_num(i) = atoms_to_masskg(N_T_num(i), A_molar(i))
        end do

        print '(A, I0)', '    Steps: ', nsteps
        print '(A)', '    Isotope           Numeric (kg)      Analytic (kg)
Rel. Err (%)'
        print '(A)', '    --------------------------------------------------
---------------'

        do i = 1, m
            print '(4X, A15, 2(ES16.6), F12.4)', &
                  names(i), M_T_num(i), M_T_ana(i), relerr_pct(M_T_num(i),
M_T_ana(i))
        end do
    end do

contains

    ! --- Helper Functions ---

    pure real(dp) function masskg_to_atoms(mass_kg, M_gpmol) result(N)
        real(dp), intent(in) :: mass_kg, M_gpmol
        N = (mass_kg * G_PER_KG / M_gpmol) * NA
    end function

    pure real(dp) function atoms_to_masskg(N, M_gpmol) result(mkg)
        real(dp), intent(in) :: N, M_gpmol
        mkg = (N / NA) * M_gpmol / G_PER_KG
    end function

    pure real(dp) function relerr_pct(num, ref) result(p)
        real(dp), intent(in) :: num, ref
        if (ref == 0.0_dp) then
            p = 0.0_dp
        else
            p = 100.0_dp * abs(num - ref) / ref
        end if
    end function

    ! --- Solvers ---

    subroutine bateman_solution(Time, lambdas, N_initial, N_out)
        real(dp), intent(in)  :: Time, N_initial
        real(dp), intent(in)  :: lambdas(m)
        real(dp), intent(out) :: N_out(m)
        real(dp) :: num, total, denom
        integer  :: r, j2, k2

        N_out = 0.0_dp
        N_out(1) = N_initial * exp(-lambdas(1) * Time)

        do k2 = 2, m-1
            num = 1.0_dp
            do r = 1, k2-1
                num = num * lambdas(r)
            end do

            total = 0.0_dp
            do j2 = 1, k2
                denom = 1.0_dp
                do r = 1, k2
```

```
                if (r /= j2) denom = denom * (lambdas(r) - lambdas(j2))
            end do
            total = total + exp(-lambdas(j2) * Time) / denom
        end do
        N_out(k2) = N_initial * num * total
    end do

    ! Conservation for stable element
    N_out(m) = N_initial
    do k2 = 1, m-1
        N_out(m) = N_out(m) - N_out(k2)
    end do
end subroutine

subroutine implicit_euler_forward(N_in, dt, steps_out, N_out)
    real(dp), intent(in)  :: N_in(m), dt
    integer,  intent(out) :: steps_out
    real(dp), intent(out) :: N_out(m)
    integer  :: step, k
    real(dp) :: denominator(m)

    steps_out = ceiling(T_END / dt)
    N_out = N_in

    ! Pre-calculate denominators (1 + lambda*dt) for efficiency
    denominator = 1.0_dp
    do k = 1, m-1
        denominator(k) = 1.0_dp / (1.0_dp + lam(k)*dt)
    end do

    do step = 1, steps_out
        ! Parent: N1 = N1_prev / (1 + lam1*dt)
        N_out(1) = N_out(1) * denominator(1)

        ! Daughters: Nk = (Nk_prev + Production) / (1 + lamk*dt)
        do k = 2, m-1
            N_out(k)   =   (N_out(k)   +   lam(k-1)*dt*N_out(k-1))   *
denominator(k)
        end do
```

```
            ! Stable Product: N_stable += Production
            N_out(m) = N_out(m) + lam(m-1)*dt*N_out(m-1)
        end do
    end subroutine

end program q2_numeric_implicit
```