HACETTEPE UNIVERSITY

Department of Nuclear Engineering

NEM 294 ENGINEERING PROJECT II

Assignment 2: Investigation of energy loss of protons and alpha particles in different environments using Bethe formulas

Student No: 2230386062

Student Name : Emre Sakarya

Delivery Date: 8.12.2025

Due Date: 8.12.2025

1 ABSTRACT

This project investigated the energy loss of heavy charged particles such as protons and alpha particles in various materials. The Bethe-Bloch formula was used to perform this analysis. Two different media of critical importance for nuclear reactor physics ordinary water (H_2O) and hydrogen gas (H_2) were considered, and both stopping power and mass stopping power were calculated for protons in the 0.01–1000 MeV range and alpha particles in the 0.2–400 MeV range.

The differential equations expressing the energy loss were solved with high precision with an adaptive step size approach using a code block/simulation generated from Python codes.

In addition, the project also examined the effect of sudden changes in the properties of matter, such as those that can occur in nuclear reactors under normal conditions or even during accidents, on energy loss. These phase transitions, including evaporation and condensation, are modelled and solved accordingly, occurring over a very short distance range of approximately 10^{-4}. This allows us to observe the effects of density changes, which can be frequently encountered in nuclear safety analyses, on particle range, energy loss trajectory, and dE/dx behaviour.

The results confirm the known relationship between stopping power and the density of the medium and the particle charge, and also show that phase transitions cause significant and sharp jumps in particle energy deposition. These results could significantly contribute to the understanding of fundamental radiation physics and, in particular, to reliable calculations in nuclear reactors and their associated infrastructure.

2 INTRODUCTION

In nuclear engineering applications, the interaction of charged particles with matter forms the basis of analyses such as reactor shield design and radiation safety. Heavy charged particles, such as alpha particles and protons, lose their kinetic energy through atomic collisions as they pass through a medium. This is called stopping power. This quantity, which represents the energy lost per unit of motion, depends on both the properties of the incident particle and, of course, the physical and chemical structure of the medium through which it passes.

This project examined the energy loss mechanisms and calculations for these two heavily charged particles, using Bethe formulations and code blocks. Bethe theory assumes that energy loss is directly proportional to the square of the electron density and particle charge, and inversely proportional to the square of the particle's speed. Indeed, the project's results further demonstrate the validity of this theorem. The project modelled normal water and hydrogen gas environments; these environments are frequently used and continue to be preferred as both coolants and moderators in nuclear systems.

Additionally, hydrogen gas, due to its low density, causes particle energy to deplete much more slowly; therefore, it provides a good contrast to clarify the difference between water and gas.

Choosing the ranges of 0.01–1000 MeV for protons and 0.2–400 MeV for alpha particles allows for the examination and graphing of a very broad energy range, covering most applications encountered in nuclear engineering.

How particles behave not only in the initial environment but also when the environment's properties change suddenly is particularly important in nuclear power plants, as undesirable power increases and resulting effects can occasionally occur. In other words, understanding how phase transitions occurring over short distances within the system (e.g., the sudden transformation of water into vapor or the condensation of gas into liquid) affect particle range and energy deposition is crucial, especially in safety analyses. Therefore, in this study, two scenarios where the phase transition occurs over a short distance of approximately 10^{-4} m were modelled, and the impact of this sudden change on dE/dx was investigated.

In this context, the code block developed in Python provides high precision in energy loss calculations using adaptive step spacing. This allows calculations with longer steps in low-density regions, while also keeping numerical errors under control by minimizing the steps in regions near the Bragg peak.

3 METHODS AND CALCULATIONS

In this project, the energy loss and range calculations of heavy charged particles in matter were performed using the Bethe–Bloch approach. The model includes both energy-dependent particle velocity calculations and closed-form stopping power calculations.

- Relativistic Kinematics

Since the kinetic energy range for protons extends up to 1000 MeV, relativistic effects are important. The velocity parameter β and the Lorentz factor γ are derived from the total energy.

- $\boldsymbol{E_{total} = E_{rest} + E_{kin}}$
- $\boldsymbol{E_{rest} = mc^2}$
- $\boldsymbol{\gamma = \frac{E_{total}}{E_{rest}}}$
- $\boldsymbol{\beta = \sqrt{1 - \frac{1}{\gamma^2}}}$

- The stopping power of heavy charged particles in a uniform medium is calculated using the Bethe formula:

$$-\frac{dE}{dx} = \frac{4\pi k_0^2 z^2 e^4 n}{m_e c^2 \beta^2}\left[ln\left(\frac{2m_e c^2 \beta^2}{I(1-\beta^2)}\right) - \beta^2\right]$$

$$E_{(i+1)} = E_i \cdot (1-f)$$

$$\Delta x = (f \cdot E_i)/S(E_i)$$

$$x_{(i+1)} = x_i + \Delta x$$

Where the physical constants and variables are defined as follows:

k : Coulomb constant

z: Atomic number of the incident particle

e: Elementary charge magnitude

n: Electron number density of the medium

m_e: Electron rest mass

I: Mean excitation energy of the medium

- Mass Stopping Power:

$$(1/\rho) \cdot (dE/dx)$$

- Numerical Integration with Adaptive Step Size

To calculate the range and simulate the trajectory, the differential equation for energy loss is discretized. Instead of a fixed step size (n), an adaptive step size algorithm is applied to minimize integration errors near the Bragg peak.

$$E_{(i+1)} = E_i \cdot (1-f)$$

$$\Delta E = f \cdot E_i$$

$$\Delta x = (\Delta E)/S(E_i)$$

$$x_{(i+1)} = x_i + \Delta x$$

- Phase Change Modelling (Boundary Conditions)

For the scenarios involving vaporization and condensation, the medium properties are defined as a function of position. The transition occurs at a critical distance $x_c = 10^{-4}m$

$$n(x) = n_water, x < x_c$$

$$n(x) = n_H2, x \geq x_c$$

$$I(x) = I_water, x < x_c$$

$$I(x) = I_H2, x \geq x_c$$

$$x_c = 10^-4$$

4 RESULTS

In this section of the project, stopping power curves, mass stopping power results, and particle range graphs obtained from numerical calculations based on the Bethe–Bloch formula are examined and analysed in detail. The behaviour of protons and alpha particles in both water and hydrogen gas environments is examined in detail and comparatively, encompassing all of the above. Furthermore, the energy and dE/dx profiles for special scenarios where the environment changes phase abruptly are also evaluated under a separate subsection, and their graphs are also drawn.

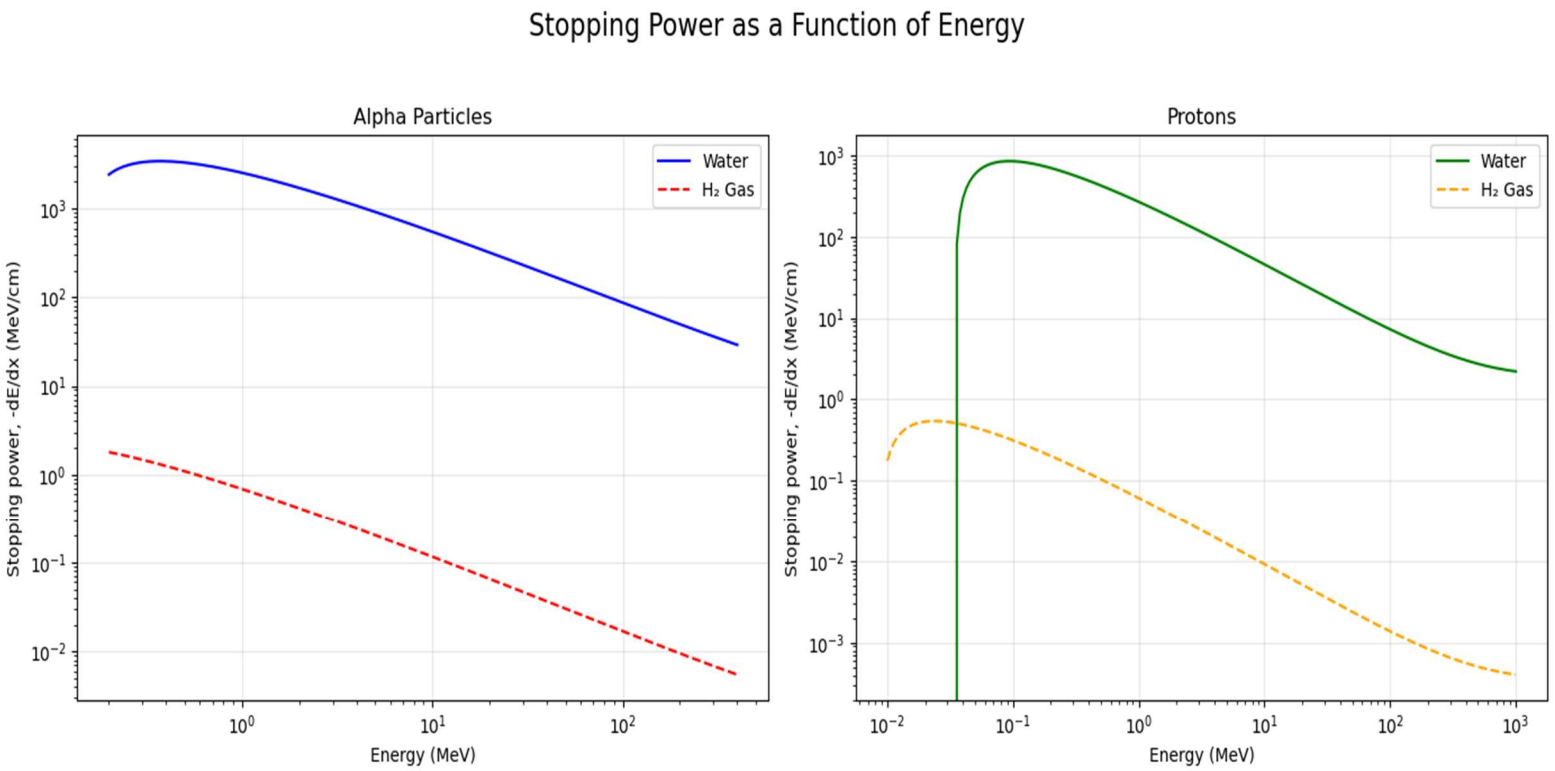

Figure 1: Shows stopping power trends for both alpha particles and protons in water and hydrogen gas media.

The stopping power curves for alpha particles and protons in water and hydrogen gas are plotted on a logarithmic-logarithmic axis. An examination of the graphs reveals that the most striking difference between the two media arises directly from the density of the media. The electron density of liquid water is approximately 6,000 times greater than that of hydrogen gas, creating a four-order difference in stopping power values on the logarithmic axis. This confirms the effect of the "n" term in the Bethe formula.

From another perspective, comparing the curves for water alone in the same medium reveals that the stopping power values of alpha particles are significantly higher than those of protons. The charge of an alpha particle is twice that of a proton. This explains the difference. As can be seen from the formula, since energy loss is directly proportional to the square of the particle's charge, alpha particles lose approximately four times more energy than protons.

As can be seen in the graphs, the stopping power tends to increase at lower energies for both particles. This is due to the interaction cross section increasing as the particle slows down.

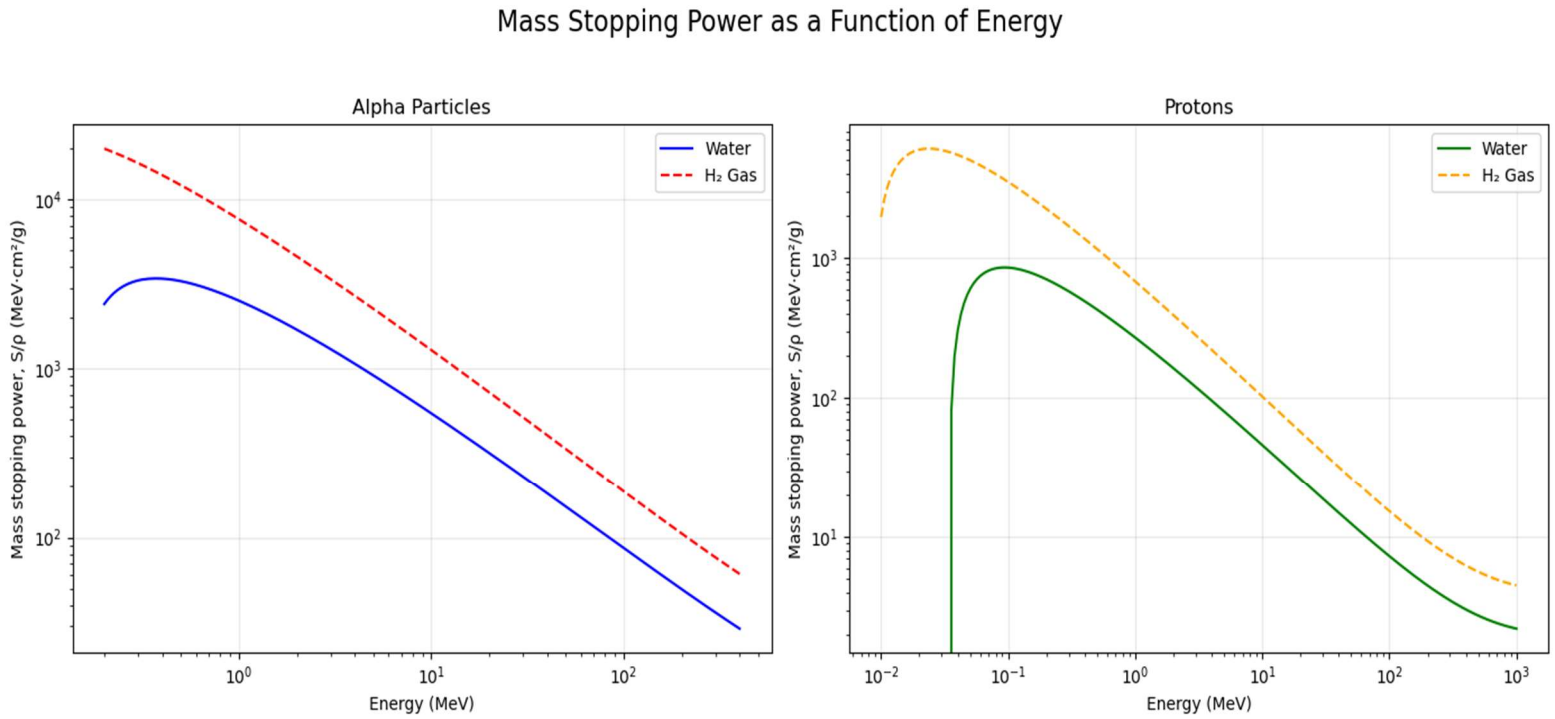

Figure 2: This figure uses the same energy ranges as in Figure 1 and shows the mass stopping power (S/ρ) results on the graph accordingly.

The difference analysed in this case is that the density effect is eliminated. These data, obtained by dividing the stopping power values by the ambient density, allow us to derive conclusions based **solely** on the atomic and chemical structure of the material.

In contrast to the significant difference in Figure 1, Figure 2 shows that the curves for water and hydrogen gas are very close. While the curves are close, they will not be exactly the same, as each medium has its own unique properties. The characteristic that gives rise to the current state is the average excitation energy. Water has a higher ionization energy than hydrogen, and when we look at the graphs, we see that the magnitude of this energy value is negatively correlated with the energy loss. In other words, stopping power decreases as the excitation energy value increases.

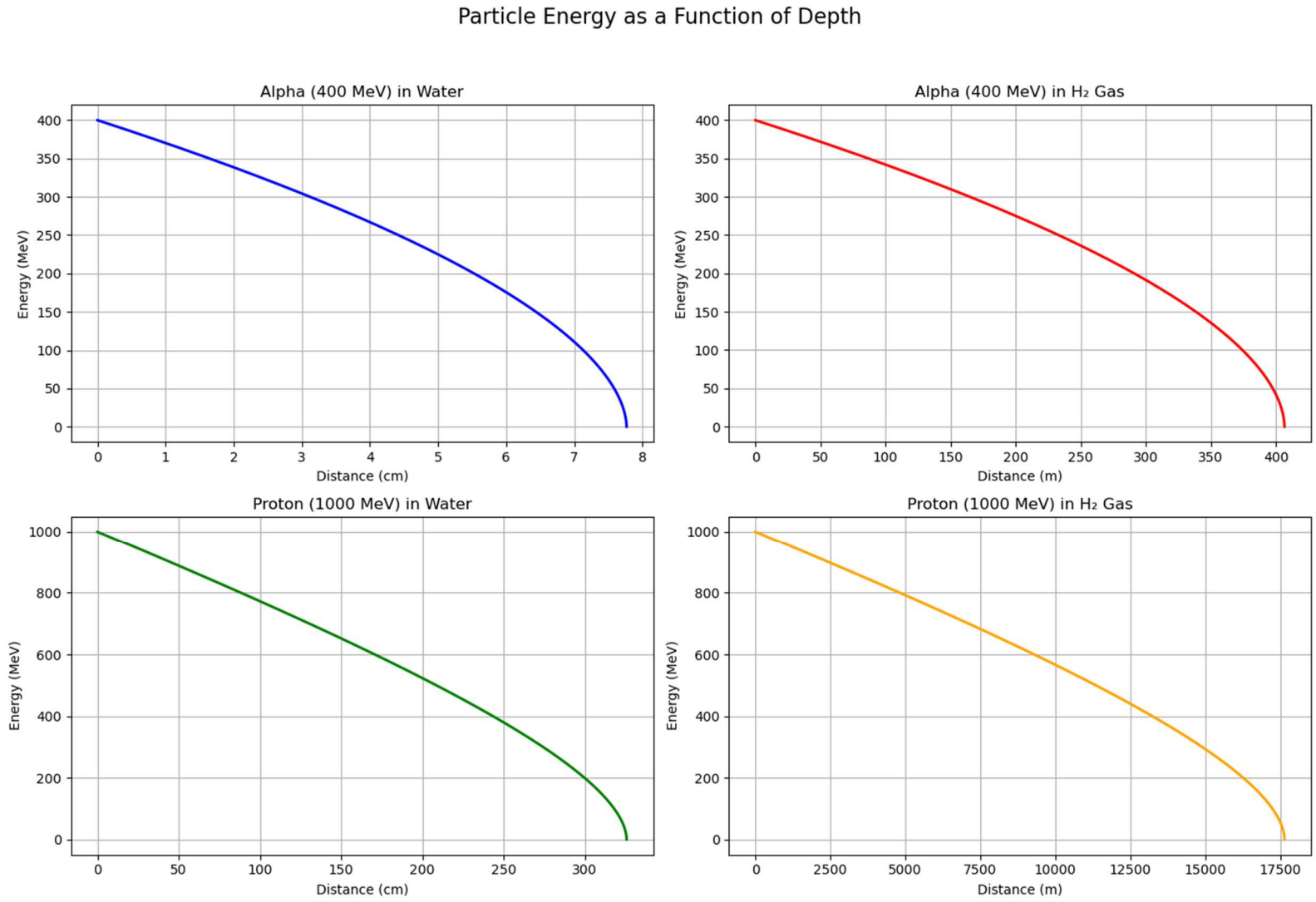

Figure 3: It shows the distance that particles travel in water and hydrogen gas environments until their energy is completely exhausted.

Here, we see remarkably similar slopes for two particles with different energy levels and different charges. Of course, the distances travelled are different for the reasons explained above, but it's clear that particles lose energy incredibly quickly in water compared to the hydrogen layer. Furthermore, this enormous difference, that is, in the water environment, is exploited in nuclear reactors to convert fast neutrons into thermal neutrons (thermal neutrons have a much higher probability of fission).

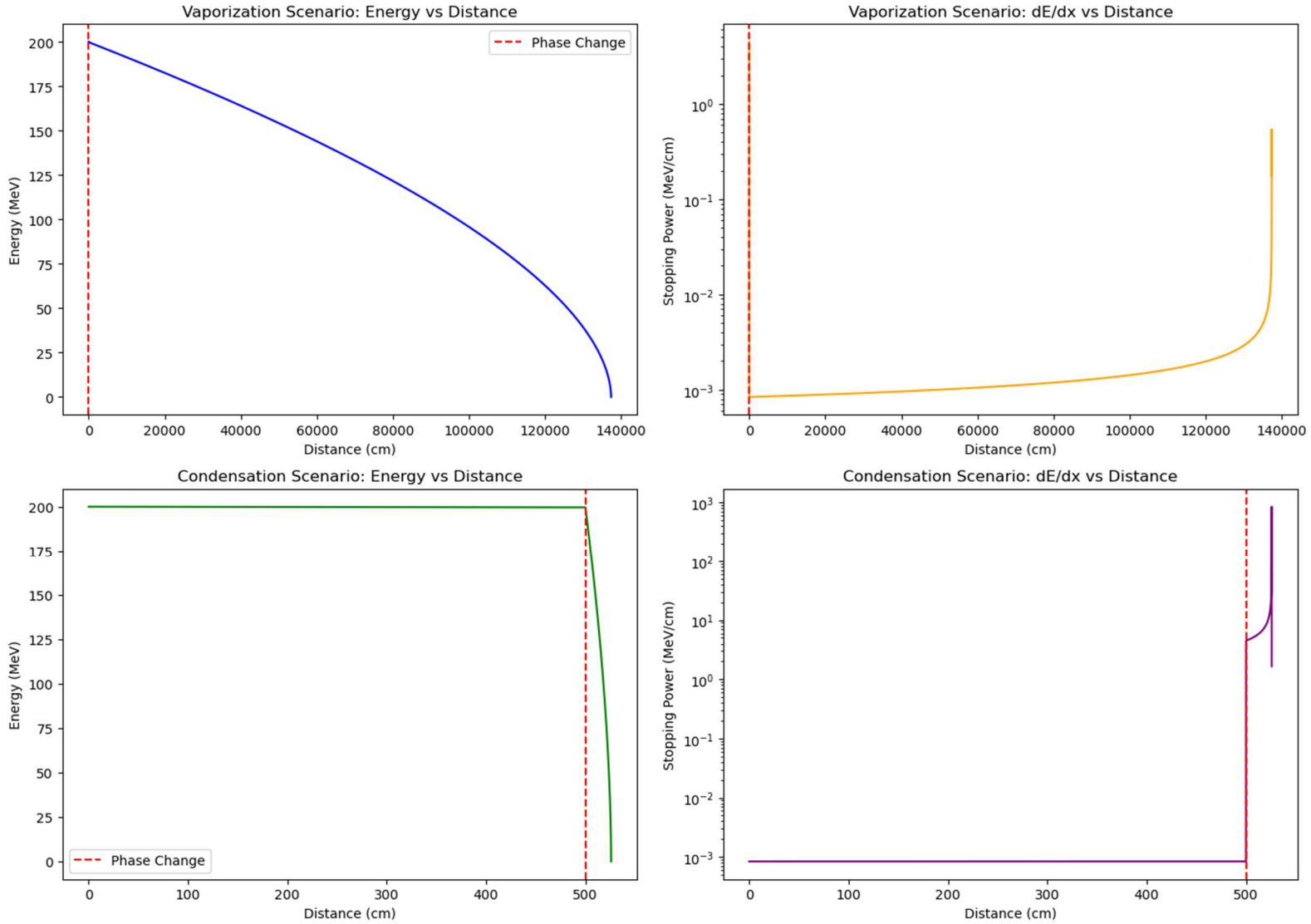

Figure 4: Phase change scenarios occurring at very small distances ({10}^{-4}).

In the evaporation scenario, the particle initially passes through water, but immediately after the phase transition, the medium suddenly changes to gas. After this point, as we can see and predict from the graph, the stopping power decreases significantly. The dE/dx graph shows a sudden drop at the phase change point, indicating that the gas has a much lower stopping power than water. The particle begins to travel a much longer distance with the same energy.

In the condensation scenario, the situation is the opposite: the particle initially moves in the gas phase, but after a certain distance, the medium suddenly changes to water. While the energy-distance graph shows a very slowly decreasing energy profile up to this point, after the phase transition, the curve steepens abruptly, and the particle rapidly loses energy over a short additional distance.

The dE/dx graph shows a sharp upward jump at the phase transition point, clearly demonstrating the sudden increase in stopping power during the transition from gas to water. Considering all this, the critical effects of sudden changes in parameters such as density and ionization potential on particle range and energy are clearly evident.

5 CONCLUSION

In this project, the energy loss processes of protons and alpha particles in different media were numerically investigated using the Bethe-Bloch formulation. The stopping power, mass stopping power, and particle spacing for water and hydrogen gas were analysed using comparative graphs. All results clearly demonstrated that stopping power is clearly dependent on both the particle's charge and the density and ionization properties of the medium.

- It was confirmed that stopping power is directly proportional to the **square** of the particle's charge, and it was observed that alpha particles disperse their energy over a much shorter distance than protons.

- Furthermore, it was determined that the density difference between water and gas media causes an approximately **four-order** change in energy transfer.

- Then, in the mass stopping power analysis, this difference is reduced to a secondary level arising **solely** from the atomic structure (ionization).

- Phase change scenarios, explored in another part of the project, demonstrate the critical impact of sudden changes in medium density on particle energy. During evaporation, a sudden decrease in stopping power causes the particle to lose energy much more slowly along the remaining path, while during condensation, the opposite increase occurs. This quantitatively demonstrates how sudden changes in the physical properties of the medium effect energy deposition.

One of the most critical outcomes of the study is the abrupt phase change (vaporization) scenario modelled at a distance of 10^{-4} m. The graphical results show that the stopping power disappears abruptly as the medium changes from water to gas, and the particle can move with virtually no resistance. This also applies to neutrons and is directly related to the concept of the "Positive Void Coefficient" in nuclear reactor kinetics.

This situation is one of the main reasons for the explosion in the RBMK type reactor in Chernobyl. The coolant water in the reactor suddenly boiled, and neutrons continued to carry out the fission reaction instead of being absorbed by the water. The energy loss stop profile at the moment of transition to gas obtained in the simulation explains very well at the microscopic level why the reactivity increase uncontrollably in this accident. (IAEA, 1992) ((IAEA), 1992)

As a result, this project successfully produced both stopping power curves and range calculations, accurately reflecting abrupt changes in energy loss even in ideal situations such as phase transitions. This study demonstrates the sensitivity of particle-matter interactions to both the particle itself and the properties of the medium, while also reiterating why density variations are critical in nuclear engineering applications.

6 REFERENCES

1. (IAEA), I. A. (1992). *RBMK Reactors: Safety Issues and Recommendations.* Vienna: IAEA.

2. (NIST), N. I. (2018). https://physics.nist.gov/cuu/Constants/

3. Attix, F. H. (2004). *Introduction to Radiological Physics and Radiation Dosimetry.* New York: Wiley.

4. Epperson, J. F. (2013). *An Introduction to Numerical Methods and Analysis.* New York: Wiley.

5. IAEA. (1992). *The Chernobyl Accident: Updating of INSAG-1 (INSAG-7).* Vienna.

6. Lamarsh, J. R. (2001). *Introduction to Nuclear Engineering.* Prentice Hall.

7. Leo, W. R. (1994). *Techniques for Nuclear and Particle Physics Experiments.* Berlin: Springer.

7 APPENDIX

```
import numpy as np
import matplotlib.pyplot as plt


#
================================================================================
==
# 0. CONSTANTS AND MATERIAL DATA
#
================================================================================
==


# Fundamental constants
k0 = 8.9875517923e9          # Coulomb constant: 1 / (4π ε0) [N·m²/C²]
e_charge = 1.60217663e-19    # Elementary charge [C]
m_e = 9.10938356e-31         # Electron mass [kg]
c_light = 2.99792458e8       # Speed of light [m/s]
```

```
    # Unit conversions
    MeV_to_Joule = 1e6 * e_charge
    Joule_to_MeV = 1.0 / MeV_to_Joule

    # Material parameters (Water, Hydrogen Gas)
    # Number density n [1/m^3], Mean excitation potential I [eV]

    rho_water = 1.0             # g/cm^3
    n_water = 3.3428e29         # 1/m^3
    I_water = 75.0              # eV

    rho_h2 = 8.988e-5           # g/cm^3
    n_h2 = 5.3745e25            # 1/m^3
    I_h2 = 19.0                 # eV

    # Particle masses
    m_proton = 1.6726219e-27    # kg
    m_alpha = 6.644657e-27      # kg



    #
================================================================================
==
    # 1. RELATIVISTIC KINEMATICS AND BETHE-BLOCH
    #
================================================================================
==


    def calculate_beta_gamma(energy_MeV: float, mass_kg: float):
        """Calculates relativistic beta and gamma factors from kinetic
energy."""
        e_kin_J = energy_MeV * MeV_to_Joule
        e_rest_J = mass_kg * c_light**2
        e_total_J = e_kin_J + e_rest_J

        gamma = e_total_J / e_rest_J
        # Protect against numerical errors when gamma is very close to 1
        if gamma < 1.0:
            gamma = 1.0

        beta = np.sqrt(1.0 - 1.0 / gamma**2)
        return beta, gamma
```

```
    def bethe_bloch_exact(energy_MeV: float,
                          particle_Z: int,
                          particle_mass_kg: float,
                          n_m3: float,
                          I_eV: float) -> float:
        """
        Calculates Stopping Power (-dE/dx) using the Bethe-Bloch formula.
        Returns: Stopping Power in [MeV/cm]
        """
        beta, gamma = calculate_beta_gamma(energy_MeV, particle_mass_kg)

        # Avoid numerical singularities at very low energies
        if beta < 1e-4:
            return 0.0

        I_Joule = I_eV * e_charge

        # Prefactor: (4 π k0^2 z^2 e^4 n) / (m_e c^2 β^2)
        prefactor = (
            4.0 * np.pi * k0**2 * particle_Z**2 * e_charge**4 * n_m3
            / (m_e * c_light**2 * beta**2)
        )

        # Relativistic argument inside the logarithm: 2 m_e c^2 β^2 / (I
(1 - β^2))
        argument = (2.0 * m_e * c_light**2 * beta**2) / (I_Joule * (1.0 -
beta**2))

        if argument <= 1.0:
            # Logarithm would be negative or zero; outside model validity
            return 0.0

        bracket = np.log(argument) - beta**2
        dE_dx_J_m = prefactor * bracket  # Result in J/m

        # Convert J/m -> MeV/cm
        # 1 J/m = (Joule_to_MeV) MeV / 100 cm
        dE_dx_MeV_cm = (dE_dx_J_m * Joule_to_MeV) / 100.0

        return dE_dx_MeV_cm
```

```
    def calculate_beta(energy_MeV: float, mass_kg: float):
        """Helper function to calculate only beta."""
        beta, _ = calculate_beta_gamma(energy_MeV, mass_kg)
        return beta

    def bethe_formula(energy_MeV: float,
                      particle_Z: int,
                      particle_mass_kg: float,
                      n_m3: float,
                      I_eV: float) -> float:
        """
        Alternative Bethe wrapper returning J/m directly.
        Used for batch calculations in arrays.
        """
        beta = calculate_beta(energy_MeV, particle_mass_kg)
        if beta < 1e-5:
            return 0.0

        I_Joule = I_eV * e_charge

        prefactor = (
            4.0 * np.pi * k0**2 * particle_Z**2 * e_charge**4 * n_m3
            / (m_e * c_light**2 * beta**2)
        )

        argument = (2.0 * m_e * c_light**2 * beta**2) / (I_Joule * (1.0 -
beta**2))
        if argument <= 1.0:
            return 0.0

        bracket = np.log(argument) - beta**2
        dE_dx_J_m = prefactor * bracket  # J/m
        return dE_dx_J_m



    #
==========================================================================
==
    # 2. HIGH-PRECISION PHASE-CHANGE SIMULATION
```

```
    #
================================================================================
==

    def simulate_high_precision(start_energy_MeV: float,
                                mode: str,
                                transition_cm: float):
        """
        Simulates energy loss of a proton with a phase change at
'transition_cm'.

        Args:
            mode: 'vaporization' (Water -> Gas) or 'condensation' (Gas ->
Water)
        """
        current_E = start_energy_MeV
        current_x = 0.0  # cm

        positions = [0.0]
        energies = [current_E]
        dedx_values = []

        # Define Initial Medium
        if mode == 'vaporization':
            current_n, current_I = n_water, I_water
            target_n, target_I = n_h2, I_h2
        else:  # condensation
            current_n, current_I = n_h2, I_h2
            target_n, target_I = n_water, I_water

        switched = False

        # Calculate initial stopping power for t=0
        initial_sp = bethe_bloch_exact(current_E, 1, m_proton, current_n,
current_I)
        dedx_values.append(initial_sp)

        # Main integration loop
        while current_E > 0.01: # Stop if energy drops below 10 keV

            # Check for phase change
            if (not switched) and (current_x >= transition_cm):
```

```
                current_n, current_I = target_n, target_I
                switched = True

                # Recalculate dE/dx immediately after transition
                new_sp = bethe_bloch_exact(current_E, 1, m_proton,
current_n, current_I)
                positions.append(current_x)
                energies.append(current_E)
                dedx_values.append(new_sp)

            # Calculate Stopping Power at current state
            dedx = bethe_bloch_exact(current_E, 1, m_proton, current_n,
current_I)
            if dedx <= 0.0:
                break

            # Adaptive Step Size Control
            fractional_loss = 0.001 # Loss 0.1% of energy per step
            delta_E = current_E * fractional_loss

            if delta_E < 1e-5:
                delta_E = 1e-5

            delta_x = delta_E / dedx  # cm

            # Handle boundary crossing precisely
            if (not switched) and (current_x + delta_x > transition_cm):
                delta_x = transition_cm - current_x
                if delta_x < 1e-12: # Avoid infinite small steps
                    delta_x = 0.0
                    current_x = transition_cm
                else:
                    delta_E = delta_x * dedx

            current_x += delta_x
            current_E -= delta_E

            positions.append(current_x)
            energies.append(current_E)
            dedx_values.append(dedx)
```

```
        return np.array(positions), np.array(energies),
np.array(dedx_values)


    #
================================================================================
==
    # 3. RANGE AND STOPPING POWER UTILITIES
    #
================================================================================
==


    def get_stopping_power_arrays(energies_MeV, Z, mass_kg, n_m3, I_eV,
rho_g_cm3):
        """Calculates Stopping Power (SP) and Mass Stopping Power (MSP)
for an energy array."""
        sp = []
        msp = []
        for E in energies_MeV:
            dE_dx_J_m = bethe_formula(E, Z, mass_kg, n_m3, I_eV)
            dE_dx_MeV_cm = (dE_dx_J_m * Joule_to_MeV) / 100.0

            sp.append(dE_dx_MeV_cm)
            msp.append(dE_dx_MeV_cm / rho_g_cm3)

        return np.array(sp), np.array(msp)

    def simulate_trajectory(start_E_MeV, Z, mass_kg, n_m3, I_eV):
        """Simple range simulation without phase change."""
        distances = [0.0]
        energies = [start_E_MeV]

        current_E = start_E_MeV
        current_x = 0.0

        while current_E > 0.05:
            dE_dx_J_m = bethe_formula(current_E, Z, mass_kg, n_m3, I_eV)
            dE_dx_MeV_cm = (dE_dx_J_m * Joule_to_MeV) / 100.0

            if dE_dx_MeV_cm <= 0.0:
                break

            dE = current_E * 0.005 # 0.5% energy loss per step
```

```
            dx = dE / dE_dx_MeV_cm

            current_x += dx
            current_E -= dE

            distances.append(current_x)
            energies.append(current_E)

        return np.array(distances), np.array(energies)


    #
================================================================================
==
    # 4. MAIN EXECUTION AND PLOTTING
    #
================================================================================
==

    if __name__ == "__main__":
        print("Running Bethe-Bloch Energy Loss Simulation...")

        # --- SIMULATION 1: Phase Change ---
        # Case A: Vaporization (Water -> Gas) at x near 0
        dist_vap, en_vap, sp_vap = simulate_high_precision(
            start_energy_MeV=200.0,
            mode="vaporization",
            transition_cm=1e-30 # Effectively at 0
        )

        # Case B: Condensation (Gas -> Water) at x = 500 cm
        dist_cond, en_cond, sp_cond = simulate_high_precision(
            start_energy_MeV=200.0,
            mode="condensation",
            transition_cm=500.0
        )

        # Plot 1: Phase Change Analysis
        fig_pc, axs_pc = plt.subplots(2, 2, figsize=(14, 10))

        # Vaporization Plots
        ax = axs_pc[0, 0]
```

```
        ax.plot(dist_vap, en_vap, color='blue', linewidth=1.5)
        ax.axvline(x=0.0,   color='red',   linestyle='--',   label='Phase
change')
        ax.set_title("Vaporization (Water -> Gas): Energy vs Distance")
        ax.set_xlabel("Distance (cm)")
        ax.set_ylabel("Energy (MeV)")
        ax.legend()
        ax.grid(True, alpha=0.3)

        ax = axs_pc[0, 1]
        ax.plot(dist_vap, sp_vap, color='orange', linewidth=1.5)
        ax.axvline(x=0.0, color='red', linestyle='--')
        ax.set_title("Vaporization: dE/dx vs Distance")
        ax.set_xlabel("Distance (cm)")
        ax.set_ylabel("Stopping Power (MeV/cm)")
        ax.set_yscale('log')
        ax.grid(True, alpha=0.3)

        # Condensation Plots
        ax = axs_pc[1, 0]
        ax.plot(dist_cond, en_cond, color='green', linewidth=1.5)
        ax.axvline(x=500.0,   color='red',   linestyle='--',   label='Phase
change')
        ax.set_title("Condensation (Gas -> Water): Energy vs Distance")
        ax.set_xlabel("Distance (cm)")
        ax.set_ylabel("Energy (MeV)")
        ax.legend()
        ax.grid(True, alpha=0.3)

        ax = axs_pc[1, 1]
        ax.plot(dist_cond, sp_cond, color='purple', linewidth=1.5)
        ax.axvline(x=500.0, color='red', linestyle='--')
        ax.set_title("Condensation: dE/dx vs Distance")
        ax.set_xlabel("Distance (cm)")
        ax.set_ylabel("Stopping Power (MeV/cm)")
        ax.set_yscale('log')
        ax.grid(True, alpha=0.3)

        plt.tight_layout()
```

```
        # --- SIMULATION 2: Stopping Power Curves ---
        E_alpha_range = np.logspace(np.log10(0.2), np.log10(400.0), 200)
        E_proton_range = np.logspace(np.log10(0.01), np.log10(1000.0),
200)

        # Alpha Particles
        sp_a_w, msp_a_w = get_stopping_power_arrays(E_alpha_range, 2,
m_alpha, n_water, I_water, rho_water)
        sp_a_h, msp_a_h = get_stopping_power_arrays(E_alpha_range, 2,
m_alpha, n_h2, I_h2, rho_h2)

        # Protons
        sp_p_w, msp_p_w = get_stopping_power_arrays(E_proton_range, 1,
m_proton, n_water, I_water, rho_water)
        sp_p_h, msp_p_h = get_stopping_power_arrays(E_proton_range, 1,
m_proton, n_h2, I_h2, rho_h2)

        # Plot 2: Stopping Power
        fig_sp, axs_sp = plt.subplots(1, 2, figsize=(14, 6))
        fig_sp.suptitle("Stopping Power vs Energy", fontsize=16)

        # Alpha Plot
        ax = axs_sp[0]
        ax.plot(E_alpha_range, sp_a_w, 'b-', label='Water')
        ax.plot(E_alpha_range, sp_a_h, 'r--', label='H2 Gas')
        ax.set_title("Alpha Particles")
        ax.set_xlabel("Energy (MeV)")
        ax.set_ylabel("-dE/dx (MeV/cm)")
        ax.set_xscale('log')
        ax.set_yscale('log')
        ax.legend()
        ax.grid(True, alpha=0.3)

        # Proton Plot
        ax = axs_sp[1]
        ax.plot(E_proton_range, sp_p_w, 'g-', label='Water')
        ax.plot(E_proton_range, sp_p_h, 'orange', linestyle='--',
label='H2 Gas')
        ax.set_title("Protons")
        ax.set_xlabel("Energy (MeV)")
        ax.set_ylabel("-dE/dx (MeV/cm)")
        ax.set_xscale('log')
```

```
        ax.set_yscale('log')
        ax.legend()
        ax.grid(True, alpha=0.3)

        plt.tight_layout(rect=[0, 0.03, 1, 0.95])



        # --- SIMULATION 3: Range Analysis ---
        dist_aw, en_aw = simulate_trajectory(400.0, 2, m_alpha, n_water,
I_water)
        dist_ah, en_ah = simulate_trajectory(400.0, 2, m_alpha, n_h2,
I_h2)
        dist_pw, en_pw = simulate_trajectory(1000.0, 1, m_proton, n_water,
I_water)
        dist_ph, en_ph = simulate_trajectory(1000.0, 1, m_proton, n_h2,
I_h2)

        # Plot 3: Range
        fig_range, axs_range = plt.subplots(2, 2, figsize=(14, 10))
        fig_range.suptitle("Particle Energy as a Function of Depth",
fontsize=16)

        ax = axs_range[0, 0]
        ax.plot(dist_aw, en_aw, 'b-', linewidth=2)
        ax.set_title("Alpha (400 MeV) in Water")
        ax.set_xlabel("Distance (cm)")
        ax.set_ylabel("Energy (MeV)")
        ax.grid(True)

        ax = axs_range[0, 1]
        ax.plot(dist_ah / 100.0, en_ah, 'r-', linewidth=2)
        ax.set_title("Alpha (400 MeV) in H2 Gas")
        ax.set_xlabel("Distance (m)")
        ax.set_ylabel("Energy (MeV)")
        ax.grid(True)

        ax = axs_range[1, 0]
        ax.plot(dist_pw, en_pw, 'g-', linewidth=2)
        ax.set_title("Proton (1000 MeV) in Water")
        ax.set_xlabel("Distance (cm)")
        ax.set_ylabel("Energy (MeV)")
        ax.grid(True)
```

```
ax = axs_range[1, 1]
ax.plot(dist_ph / 100.0, en_ph, 'orange', linewidth=2)
ax.set_title("Proton (1000 MeV) in H2 Gas")
ax.set_xlabel("Distance (m)")
ax.set_ylabel("Energy (MeV)")
ax.grid(True)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```