**Project Report - Particle Collider Simulation**

**Team-1**

**1. Introduction**

The Particle Collider Simulation project is designed to model high-energy particle collisions, akin to those conducted in large-scale experiments such as the CERN Large Hadron Collider (LHC). The simulation incorporates key concepts from particle physics, including the conservation of energy, momentum, and charge, as well as probabilistic transformations (Monte Carlo simulations) and fusion events.

The primary objective of this project is to create a realistic simulation using Python, leveraging principles of physics and computational modelling to study particle interactions and transformations.

**2. Objectives**

The main goals of this project are:

1. To simulate collisions between particles based on proximity.
2. To implement conservation laws of momentum, energy, and charge.
3. To incorporate stochastic particle transformations using Monte Carlo simulations.
4. To model fusion events when colliding particles exceed an energy threshold.
5. To visualize and analyze the behavior and properties of particles post-collision.

**3. Methodology**

**3.1 Particle Initialization**

* **Attributes**: Each particle is characterized by position (3D vector), velocity (3D vector), mass, charge, and type (e.g., Proton, Electron, Muon).
* **Randomization**: For testing, particles are initialized with random positions, velocities, masses, and charges within realistic ranges.

**3.2 Collision Detection**

* **Proximity-Based Detection**: Collisions are identified when the distance between two particles falls below a predefined threshold.
* **Distance Calculation**: The Euclidean distance formula is used to compute particle separations.

**3.3 Collision Handling**

* **Momentum Conservation**: The simulation ensures conservation of total momentum during elastic collisions by recalculating particle velocities.
* **Energy Conservation**: The total energy (rest energy + kinetic energy) is conserved for each collision or transformation.

**3.4 Stochastic Transformations**

* **Monte Carlo Simulation**: Post-collision, particles may stochastically transform into new particles with a predefined probability.
* **Transformations**: Particles can split into two smaller particles with half their original mass.

**3.5 Fusion Events**

* **Energy Threshold**: If the total energy of colliding particles exceeds a predefined threshold, they fuse to form a single, heavier particle.
* **Fusion Properties**: Combined mass of colliding particles plus additional mass from excess energy (E=mc^2), while conserving charge and momentum.

**3.6 Force Calculations**

* **Electric Force**: Coulomb’s law is used to compute forces between charged particles. The resulting accelerations modify particle velocities over time.

**4. Implementation**

The project was implemented using Python. Key components include:

**4.1 Particle Class**

Encapsulates properties and methods for individual particles:

* **Attributes**: position, velocity, mass, charge, type.
* **Methods**:
  + move(): Updates the particle's position based on its velocity.
  + bounce(width, height, depth, restitution): Handles collisions with the simulation boundaries.

**4.2 Collider Class**

Handles interactions between particles:

* **Detect Collisions**: Identifies particle pairs within the collision threshold.
* **Handle Collisions**:
  + **Momentum and Energy Conservation**: Updates particle velocities using conservation laws.
  + **Stochastic Transformations**: Applies Monte Carlo simulations to probabilistically transform particles post-collision.
  + **Fusion Events**: Combines particles into a single, heavier particle if the energy threshold is exceeded.
  + **Force Application**: Computes Coulombic forces between charged particles.

**4.3 Simulation Controller Class**

Manages the overall simulation:

* **Initialization**: Creates particles and sets up the simulation environment.
* **Run Simulation**: Updates particle positions, checks for collisions, and applies forces at each timestep.

**4.4 DataLogger Class (Optional)**

Logs and calculates kinetic energy:

* **Calculate\_kinetic\_energy(particles)**: Computes the total kinetic energy of all particles in the simulation.

**5.Results**

**5.1 Sample Output**

The following output shows the state of particles after simulated collisions:

**yaml**

Copy code

Detecting collisions...

Collision with scattering angle: 45.00 degrees.

Particles transformed into DecayProduct1 and DecayProduct2.

Fusion created a new particle with mass 3.35e-27 kg and charge 0.

Applying forces...

Final particle states:

Particle 1:

Type: Proton

Position: [ 5.44e-10 -4.39e-10 3.77e-10]

Velocity: [ 2.45e+04 -1.23e+04 8.76e+03]

Mass: 1.67e-27 kg

Charge: 1.60e-19 C

Total Energy: 1.50e-10 J

Particle 2:

Type: FusionParticle

Position: [ 0.00e+00 0.00e+00 0.00e+00]

Velocity: [ 1.12e+04 -3.14e+03 6.78e+03]

Mass: 3.35e-27 kg

Charge: 0.00e+00 C

Total Energy: 1.80e-10 J

**5.2 Observations**

* **Collision Detection**: Successfully identified and processed collisions for particles within proximity.
* **Momentum and Energy Conservation**: Verified through debug statements in the simulation.
* **Stochastic Transformations**: Decay products were created probabilistically.
* **Fusion Events**: A new particle was created with combined properties when the energy threshold was met.

**6. Conclusion**

The particle collider simulation successfully models high-energy particle interactions, adhering to fundamental physics principles:

* Conservation of momentum, energy, and charge.
* Stochastic transformations and decay processes using Monte Carlo methods.
* Fusion events based on total energy thresholds.

This project serves as a robust foundation for further exploration in computational particle physics, such as:

* Visualizing particle trajectories.
* Extending simulations to include more complex interactions (e.g., quantum effects).

**7. Future Enhancements**

* **3D Visualization**: Implement real-time visualization of particle trajectories and collisions using libraries like Matplotlib or Plotly.
* **Particle Decay**: Incorporate more realistic decay chains and probabilistic lifetime models.
* **Large-Scale Simulations**: Optimize the simulation for thousands of particles using parallel processing.
* **Machine Learning**: Use AI to predict collision outcomes based on initial conditions.

*Presented by:*

Abhishek Nagendra 100002342

Hemanth Kumar Ganchi 100002352

Akshatha Varadaraya Swamy 100003581

Kamini sharma 100002280

Jiacheng Tao 11036416

Tian Kairong 100002389

Rohit Patil 100003554