Semiclassical Simulation of the muon pair production

PHYS-512 Computational Physics - Fall 2023

Guillem Alvarez
Natalie Hardin
Shronim Tiwari

October-December, 2023

Contents

Abstract

In this project, our goal was to study the collision of e^+ e^- beams and the resulting cross-section of muon production. This was done by simulating the collisions using an elastic solid body approximation and the differential cross section. The resulting muon production was then compared to another simulation program, Pythia8 [7] and plots show we successfully reproduced its behavior. We also simulated the cross section and recreated a plot containing experimental values at different energies. Further research could include adding different outcomes of e^+ e^- collision like tau production or replacing our general parameters with real paramters.

1 Introduction

In a 1928 paper, Paul Dirac proposed a positively charged counterpart to the electron[3]. Formally discovered by Carl David Anderson in 1932, the positron was the first discovered antiparticle and the first evidence of antimatter[1]. Just four years later, one of the results of an electron-positron collision was discovered. Muons and antimuons, second generation leptons, have a charge of ± 1 e and a spin of $\frac{1}{2}$. They have a mass 206 times larger than the electron and has a long lifetime with a slow decay, as it is only governed by the weak interaction[8].

Muons are of great importance today for tomographical applications and further study in the field of particle physics. Due to their high mass, momentum, and lifespan, muons are able to penetrate solid matter and ensuing coulomb scattering is used to describe what behind or inside that barrier. This has applications for things like detecting traces of radioactive elements in scrap metals and imaging the contents of a blast furnace[2].

To use muons though they must first be created. Muons in most practical applications are harnessed through proton-nuclei collisions, but muons can sometimes be produced through electron-positron collision. With a cross-section (IE probability) of outcomes dependent on the center of mass energy, e^+ e^- collisions at energies on the magnitude of GeV and higher can result in near-elastic collisions, muon and antimuon production, and boson production that decays into tau and antitau particles[8].

This paper's goal is to simulate and visualize muon production from e^+ e^- collisions, based on cross sections from current literature. It will then verify this simulation behavior against Pythia, another high energy particle collision simulator. We show through code tests of the collision mechanism, hard sphere behavior, and cross section calculations that our simulation does resemble the behavior of experimental results and Pythia simulations. Because this paper seeks to simulate the behavior and not directly calculate phenomena, and for computational functionality, generic values are used for some particle components. This paper then discusses the approximations and assumptions made in our method, and how they could be remedied in future projects.

2 Problem Description

Electron-positron collisions have great promise for expanding knowledge beyond the standard model, but experimental developments are costly and will take decades to implement. The Compact Linear Collider (CLIC) is currently in development by CERN and is designed to implement electron-positron collisions at energies and precisions not currently available. It hopes to answer questions regarding the weak force that remain unexplained by the standard model, like the disproportionately large amount of antimatter B mesons decaying into Tau leptons[5]. Despite its importance, CLIC is not projected to start construction until 2026 and will only reach an energy of 3 TeV around 2050[6]. CLIC's facilities will also be unique, meaning time using them will be highly sought after. Therefore, the ability to simulate electron-positron collisions is important for both current theoretical research and guiding future experiments towards maximum efficiency.

This report seeks to determine the likelihood of muon creation resulting from electron-positron collisions and replicate experimental and simulated results from the literature. Experimental calculations have defined a differential cross section as shown in equation1 and the integrated cross section in equation2, from which the expected behavior in our simulation can be calculated. Our simulated behavior should mirror that of Pythia, a "general-purpose Monte Carlo event generator"[7] written in C++ that simulates high-energy collisions. Pythia's longstanding collaboration with CERN and frequent updates underscore the importance of simulations in experimental particle physics.

Differential cross-section[4]:

$$\frac{d\sigma}{d\Omega} = \frac{\alpha^2}{4s}(1 + \cos\theta^2) \tag{1}$$

Where α is the fine-structure constant, s is the Mandelstam variable of the muon after the collision. Mandelstam variables are numerical quantities that encode the energy, momentum, and angles of particles in a scattering process in a Lorentz-invariant fashion. The total cross section is obtained by integrating over the total solid angle and yields to:

$$\sigma = \frac{4\pi\alpha}{3s} \tag{2}$$

3 Code Description

We will explain piece by piece the code segments employed in conducting collision simulations and the subsequent creation of muons. Initially, our code starts by importing the essential library required for computations. However, for visualizing the outcomes and generating movies that illustrate the interaction between particle beams, we leverage the functionalities provided by Matplotlib.

The initial function called *initial_loc_vel_assigner* serves the purpose of organizing all particle-related data into a single dictionary. This dictionary encompasses crucial information such as particle positions, momentum, color for visualization, mass, radius, and charge. We chose dictionaries over NumPy arrays for their compactness and improved timing.

For simulating realistic particle beams within a particle accelerator context, we've positioned all the particles at two opposing locations. These particles are distributed using a Gaussian probability distribution to emulate the expected behavior of particle beams in such an environment.

The next function we've implemented is called *checkCollisions*. This function serves the purpose of generating an array containing particles whose centers are within a distance equal to or less than the combined sum of their radius. This identification helps to highlight potential collisions between particles in the simulation.

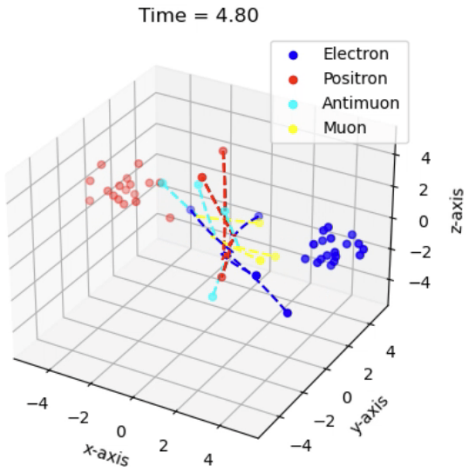


Figure 1: Instance from simulation after collision.

Finally, let's delve into the most intricate aspect of the code, the update mechanism governing both particle positions and velocities. To achieve this, we utilize the function *pos_update*. This function, in turn, incorporates the functionality of *collision_resolver*, aiding in computing the momentum and directional changes of particles' post-collision events.

For position updates, we employ the forward Euler method expressed as $\vec{posi} + 1 = \vec{posi} + dt \cdot \frac{p_i}{m}$, facilitating the evolution of particle positions over time. Simultaneously, the momentum updating process, expressed as $\vec{pi} + 1 = \vec{pi} + dt \cdot f_{Lorentz}(q, \vec{p}, \vec{B}) = \vec{p}_i + dt \cdot q \cdot (\vec{v} \times \vec{B})$, involves utilizing the *lorentz_force* function to compute the magnetic field at the specified position. We'll assume a constant magnetic field, a common scenario in particle accelerators where solenoids are often utilized at collision points. These solenoids generate a nearly constant magnetic field along their central axis. Considering the small size of the particle beam distribution, it's reasonable to assume that all particles will uniformly experience this magnetic field.

Additionally, to ensure the visualization and analysis of collided particles, we save their positions for later plotting and inspection.

Then, we proceed to store the labels of the colliding particles and initiate the computation of their momentum post-collision. The function *collision_resolver* undertakes the computation of relative positions pos_{rel} and velocities v_{rel} between the particles involved in the collision.

A crucial initial step involves relocating the particles to the collision point. This necessitates calculating the angle α, which denotes the triangular angle formed by the relative position, the trajectory of the second atom, and a line from the center of the first atom to the center of the second where both atoms collide. Determining α involves computing the vertical differences between centers $dy = |pos_{rel} \times \frac{v_{rel}}{|v_{rel}|}|$. Subsequently, α is derived from $\alpha = \arcsin(dy/(r_1 + r_2))$, and the distance traveled within the atom from the initial contact is calculated as $d = (r_1 + r_2) \cdot \cos alpha - dx$, where the horizontal distance between centers is $dx = np.dot(pos_{rel} \cdot \frac{v_{rel}}{|v_{rel}|})$. The time interval $\delta t = \frac{d}{|v_{rel}|}$ guides the positional updates necessary to return to the collision point.

Moving forward, we determine the direction and magnitude of the new velocity by calculating the total mass of the system $m_{tot} = m_1 + m_2$ and transforming momentum to the CM. Here, we evaluate the value of the Mandelstam constant $s \approx 2_{\vec{p_1}\vec{p_2}}$ under the relativistic approximation. This constant, when multiplied by the differential cross-section, facilitates the implementation of the muon creation condition. We verify the possibility of muon creation by assessing if $|s| \geq 2 \cdot m_{muon}$. Subsequently, we revert to the lab reference system.

Now, let's see how these functions are integrated to derive the simulation results. Our code has high interactivity, allowing for flexible parameter adjustments. The subsequent lines of code illustrate this flexibility by showcasing all the variables utilized in the simulation. This adaptability enables users to manipulate a wide array of variables, influencing the outcomes of the simulation.

```
Natoms = 50 # Total number of particles
targets = int(Natoms/2) # How many particles are going to be in the second beam

Rbeam = 0.5 # If the beam follows a gaussian distribution is the standard
    seviation.
# If not, it's the radius of the sphere containing all the particles

L = 5  # The distance between the two beems. Must be bigger than Rbeam

me = 1 # simplified electron mass
mu = 2 # simplified muon mass
Re = 0.1 # projectile radius (left beam particle radius)
Ru = 0.1 # target radius (right beam particle radius)
dt =  2*Re*me/p1 # Optimized timestep to avoid crossing from high speed particles

```

```
T = int(((2*L-2*Rbeam)*me/(p1*dt))) # Total time of the simulation. Is set to be
    the optimal time when the two beams
# cross each other and we consider all the possible collisions took place

p1 = 2 #Left beam particle initial momentum
p2 = -2 #Right beam particle initial momentum

q1 = -1 # Simplified left beam particle charge
q2 = 1 # Simplified left beam particle charge

B = np.array([0, 0, 0.09]) # Magnetic field
s = 2*(abs(p1*p2)) #Energy Mandelstam variables
```

Listing 1: Python Code

The time step dt is carefully chosen to prevent collisions by avoiding total overlap of particles within a single step. Similarly, the total number of steps T is calculated to allow complete beam crossing without exceeding this limit, ensuring optimal efficiency.

Following the definition of all variables, the code can be executed to simulate particle collisions and visualize the transformation of electron-positron pairs into muon-antimuons. The resultant visual representation can be seen in the video attached to the repository, titled *"Movie_collisions"*. The video showcases the initial particle distribution, collision events resulting in the creation of new muons, and the subsequent curvature induced by the magnetic field and how this curvature has different radiuses for different masses. In Figure 1, a specific frame from this movie is displayed, illustrating the state of particles moments after the collision. The code utilized to generate this movie is outlined below.

```
colisions = 0 # Colisions counter
muons = 0 # Muons couter

#We first assign the initial properties to all atoms
particles_properties = initial_loc_vel_assigner(Natoms, targets, p1, p2, me, me,
    Re, Re, q1, q2)

# Delete the particles that where overlaping in the begining
# We found this overlaping
hitlist = checkCollisions(Natoms, particles_properties)

# Then, we delete them from de dictionary
if len(hitlist) > 0:
    for i in hitlist[:,0]:
        if i in particles_properties:
            particles_properties.pop(i)

# And we redefine the total number of atoms
Natoms = len(particles_properties)

# We want to store which particles have colided and their trajectories for better
    display
particle_track = []
particle_trajectory = []

# We make the animation

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ani = animation.FuncAnimation(fig, Animation, frames = int(T*1.5), interval=50,
    repeat=False)

print('Number of colisions occurred:', colisions)
```

```
print('Number of muon creations:', muons)

# Display the animation using HTML
HTML(ani.to_jshtml())
```

Listing 2: Python Code

We started the simulation by setting the initial conditions, ensuring the elimination of any overlapping particles at the outset. Then, we applied the *Animation* function to generate a frame-by-frame plot, visualizing each step of the simulation.

However, the primary objective of this project is testing the cross-section dependency concerning collision energy. To achieve this, we conducted numerous simulations. For efficiency purposes, we duplicated the code, removing all display functions except for the final plot that demonstrates the cross-section outcomes. While maintaining identical code and initial variables, we implemented a loop iterating over a predetermined list of chosen momentums. This iterative process allows us to assess the cross-section of these interactions using the formula $\frac{muons \cdot \pi R_{beam}^2}{collisions \cdot atoms}$. In our project, we selected fifteen distinct momentums ranging from zero to six. It's essential to note that these units are arbitrary, chosen specifically to contain the required energy for muon creation. Specifically, since the muon's mass is two, we considered $E_{threshold} = 2 \cdot m_{muon}$.

Pythia8:
For the comparison of our code with more robust scientifically tested code, we used Pythia8 [7] to simulate e^+ e^- collision and produce the graph for a cross-section. Pythia is a tool used to generate high-energy events. Pythia uses Monte-Carlo methods for event generation. Pythia can be used intensively to study LHC physics. It is built using C++, however, for our project, we used the Python interface of Pythia. 'conda install -c conda-forge pythia8 ' is used for the installation of the Python interface of Pythia.

A sample code for event generation (for e^+ e^- collision) using Pythia looks like following

```
#Importing sys_pipes_forever from wurlitzer to capture and display Pythia's
    standard output
from wurlitzer import sys_pipes_forever
sys_pipes_forever()

import pythia8

pythia = pythia8.Pythia() #creating instance of Pythia

pythia.readString("WeakSingleBoson:ffbar2ffbar(s:gmZ) = on") #turning on a process

#setting particle IDs for electron and positron
pythia.readString("Beams:idA = 11")
pythia.readString("Beams:idB = -11")

pythia.init() #initializing event generator

pythia.next() #generating next event
```

Listing 3: Pythia example

Here, we set BeamA as electron and BeamB as positron. The particle naming is based on the "Monte Carlo Particle Numbering Scheme" as found here. After running this, you should be able to see the event and new particles created.

For our study, we activated "WeakSingleBoson:ffbar2ffbar(s:gmZ) = on" and turn off Z-decays except for mu mubar. Later on, we retrieve the QED total cross-section for different energy levels to produce the plot in figure 3.

To verify the integrated cross-section (contained in file *muon_creation_integrated*) for muon collision, a function *muon_prod_sigma* takes the number of electron-positron collisions and the center of mass energy as arguments. Then it uses equation2 to determine the cross section as a probability between 0 and 1 and generates a random value between 0 and 1 for each collision. If that collision 'beats' the probability it will be counted as a muon creation. The remainder of the code tests this function and plots energy as a function of muon creations per collision.

4 Code Tests

We aimed to validate the functionality of our code by computing the cross-section of a hard-sphere scattering, which presents a simpler physical system involving cross-sections. To achieve this, we adjusted the variables within the code to simulate this scenario. Specifically, we set the number of atoms in the target beam to one (*targets = 1*), increased the target's radius significantly, and augmented its mass. Then, we directed numerous small projectiles toward the target, observing the extent of their deflections. A visual representation of this process is available in the attached movie labeled *'movie_hard_sphere'* within the repository. Additionally, a frame from this movie can be seen in Figure 2.

Using the formula $\sigma = \frac{A\#deflections}{\#projectiles}$, where A represents the effective area of the projectile distribution, we computed the cross-section of this simulation. Essentially, this process emulates a computationally expensive Monte Carlo integration.

Theoretically, this cross-section is expected to be $\sigma_{theory} = \pi(R_t + R_p)$, considering the projectile radius R_p as negligible compared to the target radius R_t.

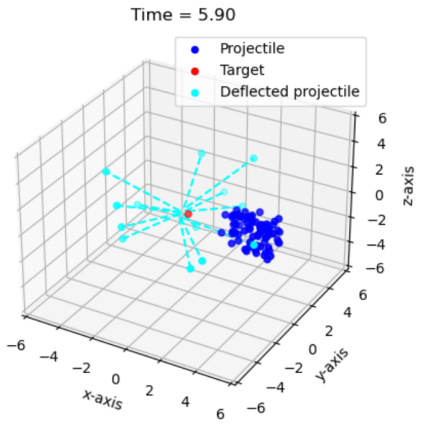


Figure 2: Instance from simulation of a target shoot with small projectiles.

Conducting the simulation with 500 projectiles and a $R_t = 0.5$, we obtained a total cross-section of $\sigma = 0.786$ m^2. This result closely aligns with the theoretical cross-section of $\sigma_{theory} = 0.785$ m^2, indicating that our code accurately reproduces the anticipated theoretical outcome. Therefore, if our final results for the muon cross-section deviate from our expectations, we can deduce that the issue lies within the theoretical approach rather than the implementation of our code.

5 Results and Analysis

The first step in implementing our code was to simulate particle collision. We were able to successfully simulate the elastic collision of particles. Next, we introduced the magnetic field and charge of particles in simulation and obtained the expected result. Furthermore, we simulated the collision-producing pair of particles and observed its motion in the magnetic field. We obtained expected behavior. After checking that the code worked well for the hard-sphere collision case, we studied the cross-section of e^+ e^- for different energy levels. Which also worked as expected. For all these steps, the code was computationally intensive since we needed to check if there was a collision between particles at each step.

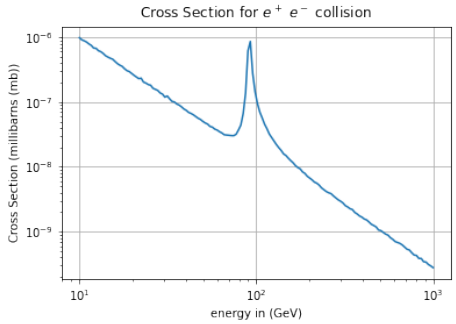
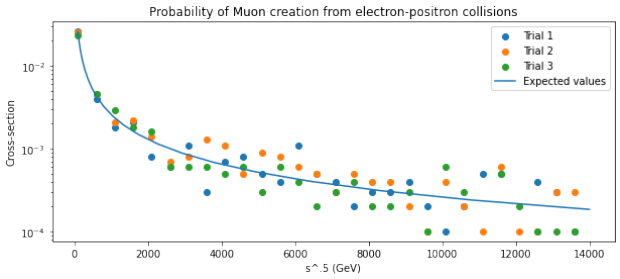


Figure 3: Cross-Section in millibarns VS Energy in GeV for e^+e^- collision produced using Pythia8

Moreover, we used Pythia to simulate LHC-like events for e^+ e^- collision leading to μ^+ μ^- production,. We selected the WeakSingleBoson:ffbar2ffbar(s:gmZ) process in Pythia to study the cross-section of e^+ e^- and we allowed for the Z-boson to decay into a muon. We studied the cross-section for the center of mass energy varying from 1 GeV to 1 TeV using Pythia and produced the plot in figure 3.

We can observe a peak in cross-section with the center of mass energy of around 92 GeV which matches the mass of Z-boson. A similar trend is seen in our code where we can observe an increase in the cross-section at the resonant energy.

Figure 4: Cross-Section VS the square root of the Madelstam momentum variable obtained with our simulation.



(a) Original

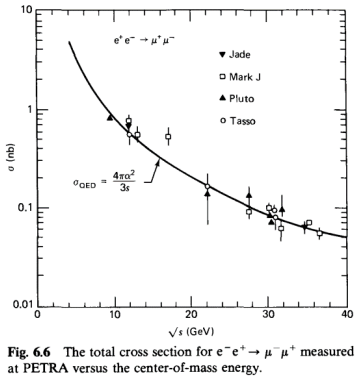


Fig. 6.6 The total cross section for $e^-e^+ \to \mu^-\mu^+$ measured at PETRA versus the center-of-mass energy.

(b) Halzen and Martin

Figure 5: (a) shows this papers recreation of the integrated cross section plot in (b)

In our code, we have not dealt with the underlying physics of particle physics. We applied classical mechanics to update particle positions, while quantum-computed differential cross-sections were used to determine the transformation of an electron pair into a muon. Even tough this approach is considerably far from reality and we are not producing the same plot as with pythia8, we were able to see a similar trend (as in Fig 4) .

By excluding the linear decreasing term of Pytha results our findings exhibit a consistent behavior. We observed that no cross section can be computed for CM energies smaller than the threshold required for the creation of a muon pair. As expected, a peak appears precisely at the energy equal to twice the mass of the two muons. Additionally, as our cross-section diminishes to a constant level, the Pytha cross section also follows the typical linear decrease pattern.

Our code was also succesful in recreating a plot verifying the integrated cross section against real controlled data found in the Petra experiment, as shown in figures 5a and 5b. This suggests that the Pythia8's observed linear trend might be influenced by experimental factors that we have not accounted for, such as the detection of residual muons or the potential creation of muons through other processes, such as Z boson decay.

6 Discussion and Conclusion

The main goal of our project was to simulate particle collisions and study cross-sections for e^+ e^- collision. In our semiclassical simulation, we used equation 1 and equation 2 from Quantum Theory to compute the differential cross-section of the collision. We were able to obtain the expected result. We studied the general behavior using dummy parameters rather than aiming to simulate actual physics using real parameters. For more robust calculation, one needs to consider particle physics in depth, like including real-time quantum field theory commutations in the collision, which is beyond the scope of this project. Moreover, we didn't consider the general relativity in depth in our implementation which can be done further to improve the accuracy of simulation. The code can be optimized, e.g. When checking collision, rather than checking every particle against every particle, divide the whole 3d grid into sub-grids and check collision between particles that are same grids.

The project was a learning opportunity. We had different milestones: including implementation of the collision mechanism; introduction of a magnetic field; confirmation of hard-sphere;

production of muon anti-muon from electron-positron; calculation of cross-section; and neat visualization. We were able to cover most of the milestones however there is room for improvement as mentioned above.

7 Contribution Statements

The tasks were divided among team members. Initially, different parts of coding were divided among our team members. Eventually, as the code grew into being more complex, we divided different sections of coding for each person.

Guillem: Focus on implementing the code and producing a numerical cross-section Natalie: Focus on theoretical cross-section Shronim: Focus on learning Pythia producing a more robust numerical cross-section.

Finally, during the report writing, different sections were again divided into group members. In general, every member of the group contributed equally in all parts of the project.

References

[1] Carl D. Anderson. The positive electron. *Phys. Rev.*, 43:491–494, Mar 1933.

[2] P. Checchia. Review of possible applications of cosmic muon tomography. *Journal of Instrumentation*, 11(12):C12072, dec 2016.

[3] Paul Adrien Maurice Dirac. The quantum theory of the electron. *Proceedings of the Royal Society*, 117(778), 1928.

[4] Francis Halzen and Martin Alan D (Alan Douglas). *6.5 e -mu- Scattering and the Process e + e -*, page 125–125. John amp; Sons, 1984.

[5] BABAR collaboration Lees, et al. Evidence for an excess of $\overline{B} \to D^{(*)}\tau^{-}\overline{\nu}_{\tau}$ decays. *Phys. Rev. Lett.*, 109:101802, Sep 2012.

[6] Eva Sicking and Rickard Ström. From precision physics to the energy frontier with the compact linear collider. *Nature Physics*, 16(4):386–392, 2020.

[7] Torbjörn Sjöstrand, Stefan Ask, Jesper R Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O Rasmussen, and Peter Z Skands. An introduction to pythia 8.2. *Computer physics communications*, 191:159–177, 2015.

[8] R. L. Workman and Others. Review of Particle Physics. *PTEP*, 2022:083C01, 2022.