Timur Holikov

Kinematics in the VBS/VBF setting

BACHELOR'S THESIS
to achieve the university degree of
Bachelor of Science

in Physics

submitted to
Graz University of Technology
University of Graz

Supervisor
Univ.-Prof. Dipl.-Phys. Dr. rer. nat. Axel Maas
Institute of Physics

Graz, March 2025

Abstract

Vector Boson Scattering (VBS) and Vector Boson Fusion (VBF) are key processes in electroweak physics that produce complex multi-particle final states. Accurately modeling these events is essential for distinguishing different production mechanisms and comparing theoretical predictions with experimental data.

This thesis simulates the kinematics of one potential VBS process involving two Z bosons and three background processes. The goal is to generate theoretical distributions for key observables such as invariant mass and angular correlations, providing insight into how kinematic constraints influence final-state distributions. By constructing kinematically consistent event samples that reflect fundamental physics constraints, such as energy-momentum conservation and the Breit-Wigner distribution, this simulation establishes a theoretical baseline for identifying different event topologies in collider experiments. The results contribute to a deeper understanding of how signal and background processes can be differentiated, supporting the interpretation of future experimental measurements.

Table of Contents

CHAPTER 1

Introduction

Vector boson fusion (VBF) and vector boson scattering (VBS) are important processes in electroweak physics resulting in multi-particle final states with complex kinematic configurations [1]. These interactions involve intermediate vector bosons such as $W^{\pm}$ and Z. A detailed understanding of these processes requires precise modeling and kinematic reconstruction, which is essential to distinguish the different underlying interaction mechanisms.

The comparison of theory and experiment is based on comprehensive experimental data analysis, where measured final-state distributions are reconstructed and fitted against theoretical expectations. It is crucial to precisely understand these patterns in order to identify deviations from Standard Model predictions and to refine event selection criteria in collider experiments. [2, 3]

In this thesis, the kinematics of a potential process involving two Z bosons, along with three background processes, is simulated to obtain theoretical predictions for key observables such as invariant mass and angular distributions. The main goal is to construct kinematically consistent event samples that reflect fundamental physics constraints such as energy-momentum conservation and the Breit-Wigner distribution. This simulation provides a baseline against which experimental data may be compared, allowing to distinguish between different events, such as direct production of final-state particles, production of an intermediate off-shell Z boson, or production of two on-shell Z bosons with subsequent decay into lepton pairs. Moreover, by analyzing different events, this work provides insight into how final-state distributions are affected by various kinematic constraints and decay mechanisms.

The remainder of this thesis is organized as follows. Chapter 2 provides a theoretical background on vector boson scattering (VBS) and vector boson fusion (VBF), along with key concepts such as Monte Carlo methods, Feynman diagrams, the Breit-Wigner distribution, and kinematic constraints. Chapter 3 describes the simulation framework, focusing on the algorithmic approach and event validation. Chapter 4 presents and discusses the results, emphasizing key observables such as invariant mass distributions and angular correlations. Finally, Chapter 5 concludes the thesis with a summary of the findings and suggestions for future research.

CHAPTER 2

Theoretical Background

2.1 Vector Boson Scattering (VBS) and Vector Boson Fusion (VBF)

VBS and VBF processes are studied extensively in high-energy physics as they provide insight into the structure of electroweak interactions, test unitarity constraints, and serve as probes for physics beyond the Standard Model [4, 5]. These processes play a crucial role in understanding weak boson self-interactions and are highly relevant for calibration purposes in collider experiments [6, 7].

These interactions occur in both hadronic and leptonic collisions, involving the exchange of virtual gauge bosons that mediate the process [8]. Theoretical descriptions of VBS and VBF rely on perturbative quantum field theory, where the kinematic distributions of final-state particles depend on the properties of the exchanged bosons, their polarization states, and potential off-shell effects. By analyzing these distributions, experimental measurements can validate theoretical predictions and constrain new physics scenarios [9, 10].

A crucial aspect of VBS/VBF studies is understanding the relevant background processes and decay channels that contribute to similar final-state signatures [11]. In case of electron-positron collisions, several electroweak processes can lead to comparable multi-lepton final states [12]. Among the primary processes relevant to this thesis are:

- **Drell-Yan process**: The production of a virtual Z boson, which subsequently decays into a lepton pair, represents a background in multi-lepton final states. This process serves as a benchmark for electroweak measurements and is important for detector calibration [7, 13].
- **Four-fermion production**: Electroweak four-fermion interactions contribute to the same final states as VBS processes. These include channels where an off-shell Z boson decays into four fermions or where an intermediate gauge bosons produce additional lepton pairs [6, 11].

- **Di-boson production**: Reactions such as $e^+e^- \to ZZ$ or $e^+e^- \to WW$ lead to multi-lepton final states and serve as irreducible backgrounds to VBS/VBF processes.

Identifying VBS/VBF events in collider experiments requires an understanding of possible backgrounds and their kinematic properties. Well-defined event selection strategies and optimized kinematic cuts help separate signal from background, improving the accuracy of the measurements [3, 12, 14].

2.1.1 Mass-shell

A crucial concept in the study of VBS and VBF is the classification of particles as *on-shell* or *off-shell*. The mass-shell condition describes whether a particle satisfies the relativistic energy-momentum relation:

$$E^2 - |\mathbf{p}|^2c^2 = m^2c^4, \tag{2.1}$$

where E is the energy, $\mathbf{p}$ is the three-momentum, and m is the rest mass of the particle. Particles that satisfy this equation are considered *on-shell*, meaning they have their physical, observable mass. In contrast, *off-shell* (or virtual) particles do not satisfy this relation and only exist as intermediate states in Feynman diagrams [15, 16].

2.1.2 Breit-Wigner Distribution

In VBS and VBF, intermediate gauge bosons can be off-shell, affecting their kinematic distributions and the final-state observables [15]. The propagator of a virtual Z boson describes how the particle propagates between interaction points in quantum field theory. It is a fundamental component of Feynman diagrams and influences the probability amplitude of a given process. The propagator for an off-shell Z boson depends on its four-momentum squared, q^2, and is given by [15, Sec. 16.1]:

$$D(q^2) = \frac{1}{q^2 - m_Z^2 + i\Gamma_Z m_Z}, \tag{2.2}$$

where $\Gamma_Z = 2.4955 \pm 0.0023$ GeV [17] represents the decay width of the Z boson, accounting for its lifetime. The farther an intermediate particle is from its mass-shell, the more its contribution to the interaction is suppressed. The propagator enters directly into the squared matrix element $|\mathscr{M}|^2$ [15, Sec. 16.1], and therefore it affects the interaction probability and event rates in collider experiments.

While a full computation of cross-sections is beyond the scope of this work, it is worth mentioning that the squared matrix element, $|\mathscr{M}|^2$, plays a crucial role in determining scattering probabilities. Through integration over the phase space element $d\Phi$, it influences both total and differential cross-sections. The presence of off-shell effects modifies the propagator, impacting these distributions accordingly. Although cross-section calculations are not explicitly performed here, understanding their role provides a fuller description of the theoretical framework underlying VBS and VBF processes.

The Breit-Wigner distribution [18] arises from the propagator when considering the probability of producing an unstable particle like the Z boson:

$$|D(q^2)|^2 = D(q^2)D^*(q^2) = \frac{1}{(q^2 - m_Z^2)^2 + (\Gamma_Z m_Z)^2}. \tag{2.3}$$

Since the four-momentum squared q^2 corresponds to the squared invariant mass m^2 of the produced boson, they are interchangeable, and hence:

$$P(m) \propto |D(q^2)|^2 \propto \frac{1}{(m^2 - m_Z^2)^2 + (\Gamma_Z m_Z)^2}. \tag{2.4}$$

A commonly used representation of the Breit-Wigner distribution is given by [19]:

$$P(E) = \frac{\Gamma}{2\pi} \frac{1}{(E-M)^2 + \frac{\Gamma^2}{4}}. \tag{2.5}$$

where M is the mass (or central energy) of the unstable state and Γ the decay width.

This function describes how the mass of an unstable particle fluctuates around its nominal value due to quantum effects, ensuring accurate modeling of resonance behavior in simulations. However, to be a true probability density function, it must be properly normalized [20, 21].

In practical Monte Carlo simulations, approximations of the Breit-Wigner distribution are often used for efficient sampling [22]. These approximations are used in event generators to handle the mass spectrum of unstable particles, ensuring a realistic description of resonance behavior in simulations. Various techniques, including running-width modifications, alternative parameterizations and inverse transform sampling methods, are implemented to optimize sampling and maintain physical accuracy [23].

The inverse transform method is particularly useful for generating random numbers according to the Breit-Wigner distribution, as it allows direct sampling from the cumulative distribution function, avoiding unnecessary rejections and improving computational speed. [21].

2.2 Feynman Diagrams and Their Interpretation

Feynman diagrams provide a systematic and intuitive way to represent particle interactions. They serve as a visualization for complex mathematical expressions, allowing for a structured approach to calculating scattering processes, decays, and other fundamental interactions. Each diagram represents a specific contribution to the total probability amplitude of a process, where more complex interactions, such as loop corrections or additional virtual particles, correspond to higher-order refinements in the theoretical predictions.

In the context of Vector Boson Scattering (VBS) and Vector Boson Fusion (VBF), Feynman diagrams illustrate how initial-state particles exchange gauge bosons, leading to specific final-state configurations. These diagrams contain both the kinematics and dynamics of the process, making them essential for understanding and cross-section calculations.

2.2.1 Components of a Feynman Diagram

There are different conventions for drawing Feynman diagrams. In this thesis, a coordinate system where the horizontal axis represents time and the vertical axis represents space is used. Particles moving from left to right are interpreted as propagating forward in time, while antiparticles, by convention, can be depicted as moving backward in time.

- **External Lines**: Represent incoming and outgoing real particles such as quarks, leptons, or gauge bosons. These are typically drawn as straight lines for fermions (e.g., electrons, quarks) and wavy lines for photons and electroweak bosons (W, Z).
- **Internal Lines**: Represent virtual particles that mediate interactions. In VBS and VBF, these include the exchange of W and Z. In the case of strong interactions, internal lines can also represent gluons, which are conventionally drawn as curly lines.
- **Vertices**: Points where lines meet, indicating interactions between particles. For example, in electroweak interactions, a vertex could represent a W boson mediating a charged-current interaction. The nature of these interactions is determined by the Standard Model Lagrangian [24].
- **Loops**: Represent higher-order quantum effects, where virtual particles briefly appear and interact within the diagram. These loops contribute to radiative corrections, affecting cross-sections and decay rates. In electroweak interactions, loop diagrams account for effects like self-energy corrections to gauge bosons or vertex corrections in scattering processes.

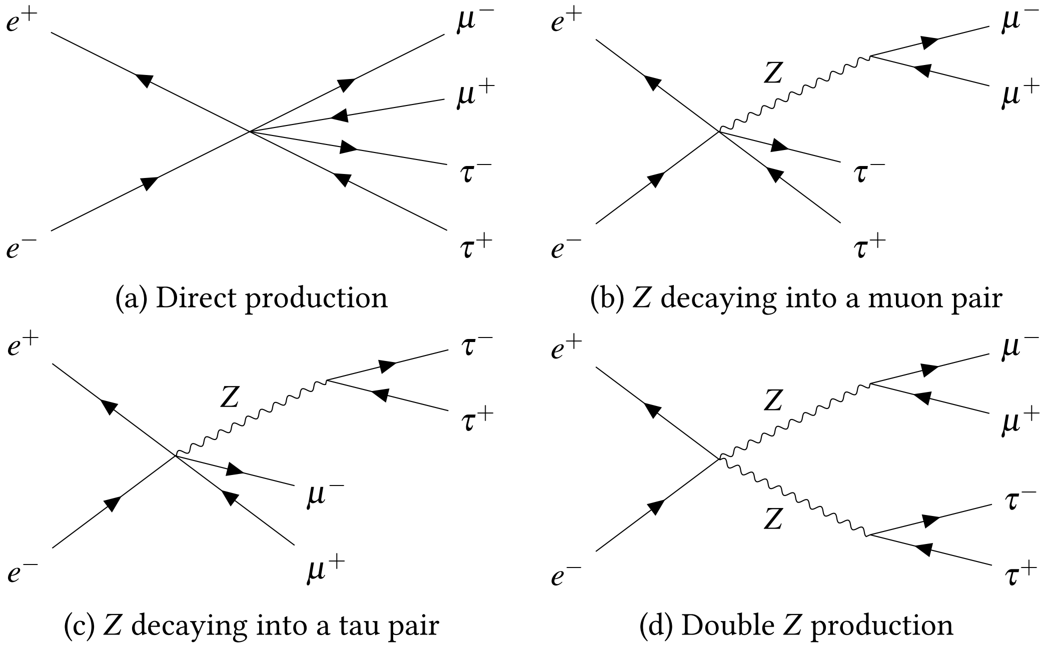
The interpretation of a Feynman diagram follows from applying the Feynman rules, which assign mathematical expressions to each diagram component. These rules dictate how propagators, interaction vertices, and phase-space factors combine to form the probability amplitude of a given process [25].

Understanding these elements is crucial for analyzing the different VBS and VBF scenarios, as they provide insight into kinematic structure, interference effects, and off-shell contributions.

2.2.2 Feynman Diagrams for $e^+e^- \to \mu^+\mu^-\tau^+\tau^-$

The process $e^+e^- \to \mu^+\mu^-\tau^+\tau^-$ can proceed through several different mechanisms, which are best understood through Feynman diagrams. These diagrams provide a visual representation of how initial-state electrons and positrons interact, leading to the production of final-state muons and tau leptons.

In all considered scenarios, the initial electron-positron pair annihilates and produces intermediate states that eventually lead to the observed muon and tau pairs. The difference between these scenarios lies in the intermediate steps: whether the process involves a single or multiple Z bosons, and whether these bosons decay into muon or tau pairs. Since the focus of this thesis is on the kinematics of the initial- and final-state particles, the vertex at which e^+e^- annihilation occurs is treated effectively, and the details of the intermediate interactions are not explicitly analyzed.



(a) Direct production

(b) Z decaying into a muon pair

(c) Z decaying into a tau pair

(d) Double Z production

Figure 2.1: Feynman diagrams for $e^+e^- \to \mu^+\mu^-\tau^+\tau^-$ in different scenarios

In the first diagram Fig. 2.1(a), the annihilation of e^+e^- leads immediately to the production of two muons and two taus. In contrast, diagrams Fig. 2.1(b) and Fig. 2.1(c) illustrate cases where an intermediate Z boson is produced and then decays into either a muon or a tau pair. Finally, diagram Fig. 2.1(d) represents a scenario where two Z bosons are produced, each decaying into a separate lepton pair.

While these scenarios differ in their intermediate states, their overall final-state remains the same. However, their relative probabilities depend on factors such as energy scale and off-shell contributions. The double Z production process, for example, is less probable at lower energies due to the large invariant mass required to produce two Z bosons.

2.3 Monte Carlo Simulations in Collider Experiments

Monte Carlo (MC) event generators are essential tools in high-energy physics, enabling the simulation of scattering processes based on theoretical models and their comparison with experimental data [26, 27]. These generators implement the principles of quantum field theory and integrate kinematic constraints to produce physically meaningful events [28].

In the study of Vector Boson Scattering (VBS) and Vector Boson Fusion (VBF), MC simulations are crucial for modeling weak boson interactions, incorporating off-shell effects, and accounting for background processes [22]. The complexity of these interactions requires precise handling of phase-space sampling and kinematic restrictions to ensure energy and momentum conservation in multi-particle final states.

Several widely used Monte Carlo event generators have been developed, among them: PYTHIA [29], MadGraph [30], Sherpa [31], HERWIG [32].

2.3.1 Phase Space Sampling and Event Kinematics

To accurately simulate particle collisions, MC event generators must efficiently sample phase space while ensuring that generated events satisfy fundamental kinematic constraints [33]. These constraints include:

- **Energy-momentum conservation**: Ensuring that the total energy and momentum remain consistent throughout the interaction.
- **Phase-space validity**: Excluding nonphysical configurations through filtering methods.
- **Off-shell effects**: Accounting for the virtuality of exchanged bosons, which influences kinematic distributions and decay probabilities.
- **Degrees of freedom in multi-particle interactions**: Determining how momentum is shared between initial and final-state particles, particularly in VBS and VBF processes.

By incorporating these constraints, MC simulations can provide reliable predictions for collider experiments, ensuring that theoretical models align with observed data.

CHAPTER 3

Simulation Implementation

In this chapter, the implementation of the simulation framework is described. The main focus is on the practical steps, background calculations, and how physical constraints were enforced using MATLAB.

3.1 Simulation Structure

The simulation consists of several scripts, each handling a different part of the event generation:

- The `main.m` sets up the initial parameters, calls different scenario scripts, and collects and visualizes the output.
- Scenario-specific scripts `Scenario1.m`, `Scenario2a.m`, `Scenario2b.m`, `Scenario4.m` implement specific physical cases, focusing on different decay channels of Z boson into muons and taus.
- Supporting functions perform core computations such as invariant mass sampling, random direction assignment, data visualization, and additional necessary functions.

Each scenario script generates events by randomly assigning the momenta of final-state particles within kinematic constraints. Intermediate Z boson masses are sampled from the Breit-Wigner distribution. Iterative adjustments are performed to enforce strict energy-momentum conservation, and only events meeting all the fundamental criteria are retained for further analysis.

Thus, the main steps of the implemented algorithm are:

1. Random assignment of final-state particle momenta.
2. Sampling of intermediate resonance masses.
3. Iterative adjustment to satisfy conservation laws.

4. Storage of physically valid events for subsequent analysis.
5. Data visualization.

3.2 Simulation Parameters

The simulation models electron-positron collisions at a center-of-mass energy of:

$$\sqrt{s} = 300 \text{ GeV}. \tag{3.1}$$

Particle masses and widths used in the simulation are summarized in Table 3.1:

Table 3.1: Particle parameters used in the simulation.

Particle	Mass / GeV	Decay width / GeV
Muon (μ)	0.105658	–
Tau (τ)	1.776861	–
Z boson	91.1876	2.4952

Simulation parameters were chosen according to known experimental values to ensure realistic simulation results.

Numerical tolerances used in event generation are summarized in Table 3.2.

Table 3.2: Numerical tolerances used in the simulation.

Constraint	Tolerance
Mass-shell condition	10^{-3} GeV^2
Total spatial momentum conservation	5×10^{-2} GeV
Energy conservation	5×10^{-2} GeV
Iteration tolerance for mass-shell adjustment	10^{-4}

3.3 Particle Generation

In this section, a detailed approach to the generation of the final-state particle momenta in each scenario is described, and hence:

1. **Initial Momentum Generation:** The energy of the first particle, E_1, is randomly assigned between its rest mass m_1 and an upper boundary set by half the center-of-mass energy $\sqrt{s}/2$:

$$E_1 = m_1 + r\left(\frac{\sqrt{s}}{2} - m_1\right), \quad r \in [0,1]. \tag{3.2}$$

The corresponding momentum magnitude is determined by:

$$|\mathbf{q}_1| = \sqrt{E_1^2 - m_1^2}. \tag{3.3}$$

Momentum directions for particles $\mathbf{q_1}$, $\mathbf{q_2}$, and $\mathbf{q_3}$ are randomized uniformly in three-dimensional space, defined by spherical coordinates:

$$\theta = \pi r_1, \quad \phi = 2\pi r_2, \quad r_1, r_2 \in [0,1], \tag{3.4}$$

with momentum components given by:

$$\mathbf{q}_i = |\mathbf{q}_i| \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}, \tag{3.5}$$

After generating each particle's momentum, the off-shell condition is verified. If it is not satisfied, the event is discarded and regenerated.

2. **Second Particle Momentum and Invariant Mass Constraints:** For scenarios involving particle pairs from Z resonances, the momentum of the second particle is explicitly calculated to match the target invariant mass m_Z:

$$E_2 = \frac{m_Z^2 - 2m_{\mu,\tau}^2 + 2E_1 m_{\mu,\tau}}{2(E_1 - \mathbf{q_1} \cdot \mathbf{q_2})}, \tag{3.6}$$

where $m_{\mu,\tau}$ represents either muon or tau mass.

If this calculation results in non-physical values, the event is discarded.

3. **Third Particle:** The third particle is generated similarly to the first one, using the remaining energy as an upper boundary.

4. **Fourth Particle:** The fourth particle's momentum is computed explicitly from momentum conservation:

$$\mathbf{q}_4 = -(\mathbf{q_1} + \mathbf{q_2} + \mathbf{q_3}), \tag{3.7}$$

and its energy from:

$$E_4 = \sqrt{m_4^2 + |\mathbf{q_4}|^2}. \tag{3.8}$$

5. **Iterative Energy Adjustment:** Due to numerical precision limitations and random generation, the fourth particle's energy, which is computed explicitly from momentum conservation, does not always satisfy the mass-shell condition exactly. Small numerical deviations may cause $q_4^2 = E_4^2 - |\mathbf{q}_4|^2$ to differ slightly from m_4^2. To account for this, an iterative adjustment is applied:

   a) Compute the initial energy of the fourth particle using Eq. 3.8:

   $$E_4^{(0)} = \sqrt{m_4^2 + |\mathbf{q}_4|^2}. \tag{3.9}$$

   b) Evaluate the squared mass deviation:

   $$\Delta m_4^2 = E_4^{(0)2} - |\mathbf{q}_4|^2 - m_4^2. \tag{3.10}$$

   c) If $|\Delta m_4^2| > \varepsilon$, iteratively refine E_4 using:

   $$E_4^{(n+1)} = \sqrt{m_4^2 + |\mathbf{q}_4|^2}. \tag{3.11}$$

   d) Repeat until the mass-shell condition is satisfied within the defined tolerance:

   $$\left| E_4^{(n)2} - |\mathbf{q}_4|^2 - m_4^2 \right| < \varepsilon. \tag{3.12}$$

   e) If the correction does not converge within a set number of steps, the event is discarded and regenerated.

   This ensures that the final-state particles remain physically valid while allowing for minor numerical variations.[1]

6. **Event Acceptance and Final Checks:**

   After generating all four-momenta and applying iterative corrections, the event undergoes a final validation step. The event is accepted if:

   - The total energy matches the center-of-mass energy within a predefined tolerance ΔE:

   $$|E_{\text{total}} - \sqrt{s}| < \Delta E. \tag{3.13}$$

   - The total spatial momentum is sufficiently small:

   $$|\mathbf{P}_{\text{total}}| < \Delta P. \tag{3.14}$$

   - Each generated particle satisfies its mass-shell condition within the defined tolerance.

   Events that do not meet these criteria are discarded and the process restarts. After generating valid events, the next step is to analyze key kinematic distributions to verify the expected event characteristics.

[1] A brief error analysis is provided in Appendix A.1

3.4 Analysis and Visualization

The generated events are analyzed by examining invariant mass and angular distributions to assess the kinematics of the simulated processes and ensure their consistency with theoretical expectations.

3.4.1 Histogram and Kernel Density Estimation (KDE)

Two methods were used to visualize the distributions:

- **Histograms**: Provide a discrete representation of the data, visualizing the frequency of values within predefined bins.
- **KDE**: Estimates a continuous probability density function, allowing for a smooth representation of the underlying distribution.

Both methods complement each other in visualizing the event distributions. Histograms give a straightforward view of the data structure, while KDE provides a smooth representation that can be useful for comparing distributions without dependence on binning.

KDE is defined as:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right), \tag{3.15}$$

where $K(x)$ is the kernel function, and h is the bandwidth parameter that controls the smoothness of the estimate.

Implementation in MATLAB Initially, KDE was tested manually with different kernel functions [2], and afterward, the built-in `ksdensity()` function was used for efficiency.

3.4.2 Key Distributions and Observables

The following distributions were computed and visualized [3]:

- **Total invariant mass distribution m_{total}**: Represents the reconstructed invariant mass of all four final-state particles. This distribution serves as a crucial check to verify energy-momentum conservation and identify any deviations from expected theoretical values.
- **Pairwise invariant masses $m_{q_1q_2}$, $m_{q_3q_4}$, and $m_{q_1q_3}$**: These distributions provide insight into the correlations between the generated particles. Specifically, they help determine whether particle pairs originate from an intermediate Z boson decay or if they are uncorrelated background contributions.

[2] A detailed comparison is provided in Appendix A.3

[3] Additionally, pseudorapidity distributions were generated. Since they primarily confirm the isotropic nature of the event generation, they are provided in Appendix A.2

- **Angular distributions** $\cos\theta_{12}$ **and** $\cos\theta_{34}$: These distributions help analyze the relative directions of particle pairs, providing insight into the event topology.
- **Summed distributions**: Allow to compare the total event, including both the primary process and background contributions.

The full MATLAB implementation, including all scripts used for event generation and visualization, is provided in the dedicated Code section.

CHAPTER 4

Results and Discussion

The analysis in this chapter follows a realistic experimental approach: starting from the observable final state $e^+e^- \to \mu^+\mu^-\tau^+\tau^-$, the goal is to reconstruct and distinguish the contributing sub-processes. The discussion begins with the full dataset that includes all simulated scenarios, followed by a separate analysis of each scenario to highlight its characteristic features and its role within the overall distribution. In each scenario, 25 000 events were generated, of which 20 827 satisfied all physical constraints and were used in the analysis.

4.1 Overview of Combined Distributions

A simple but essential starting point is the validation of the energy-momentum conservation. Figure 4.1 shows the total invariant mass distribution across all simulated scenarios. The sharp peak at $\sqrt{s} = 300\,\text{GeV}$ confirms that no unphysical contributions are present and that the fundamental constraint was consistently maintained. This serves as a useful sanity check before proceeding to the analysis of more detailed distributions.

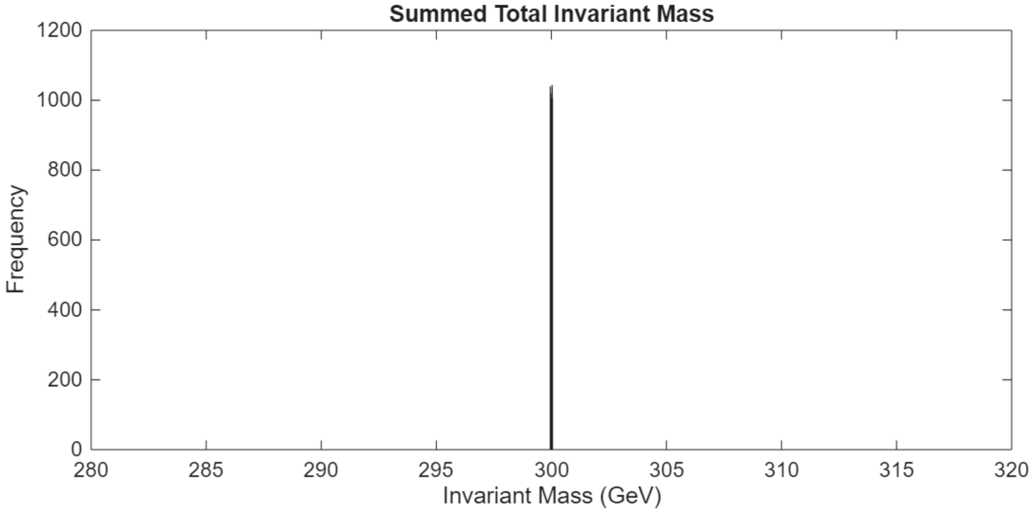
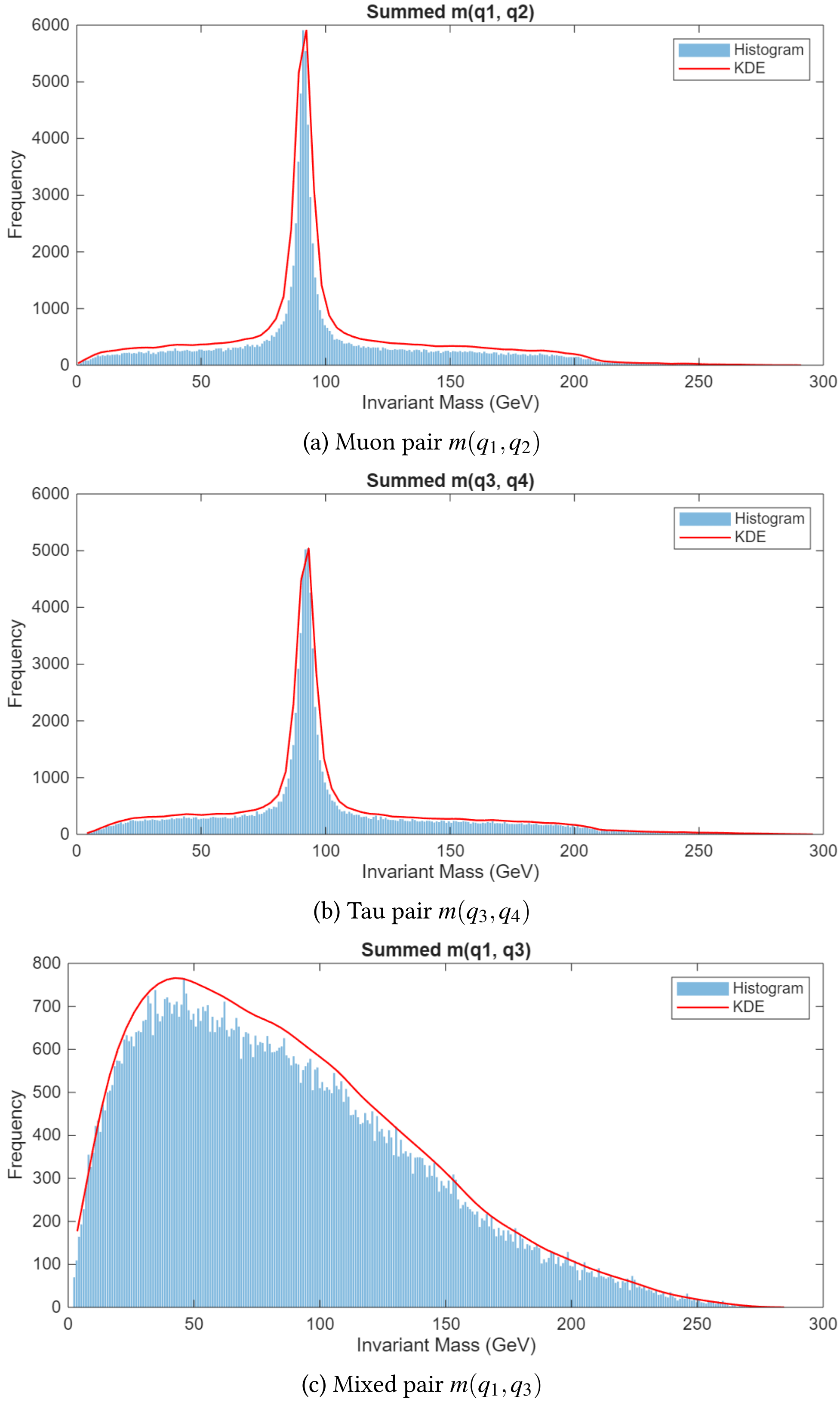


Figure 4.1: Total invariant mass distribution m_{total} across all generated scenarios.



(a) Muon pair $m(q_1, q_2)$

(b) Tau pair $m(q_3, q_4)$

(c) Mixed pair $m(q_1, q_3)$

Figure 4.2: Pairwise invariant mass distributions aggregated across all scenarios. Peaks near $m_Z \approx 91\,\text{GeV}$ indicate intermediate resonances.

The pairwise invariant mass spectra (Fig. 4.2) show narrow peaks near $m_Z \approx 91\,\text{GeV}$, characteristic of Z-mediated decays, as well as broad continua from uncorrelated production. This reflects the mixture of resonant and non-resonant topologies present in the simulated process.

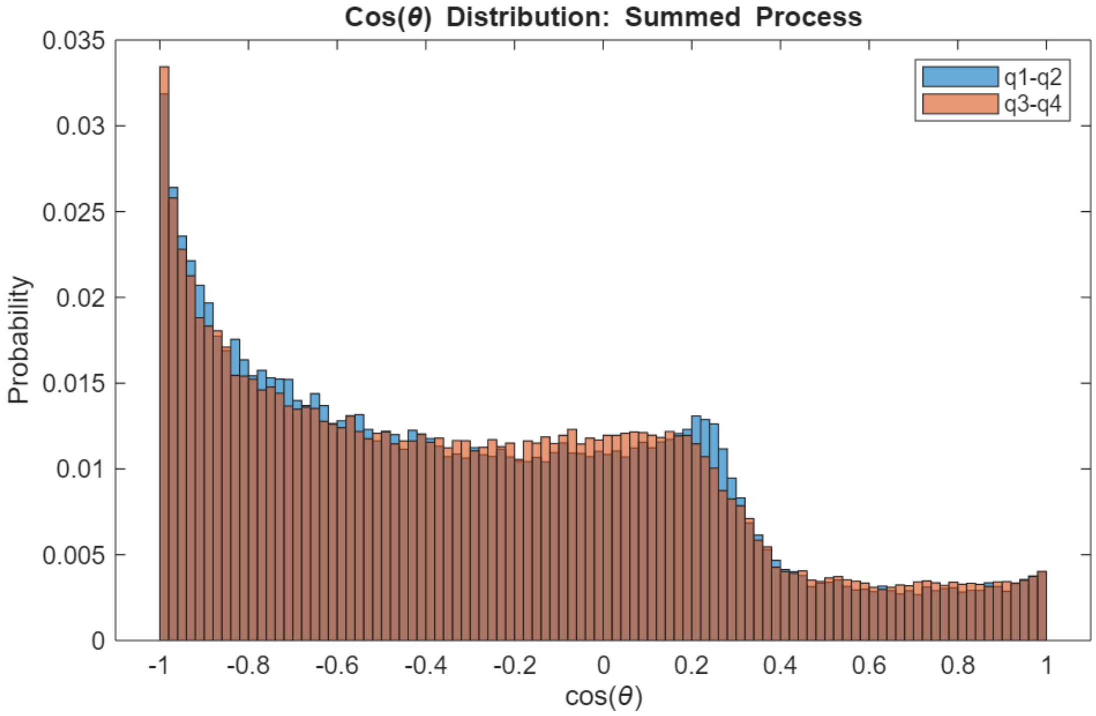


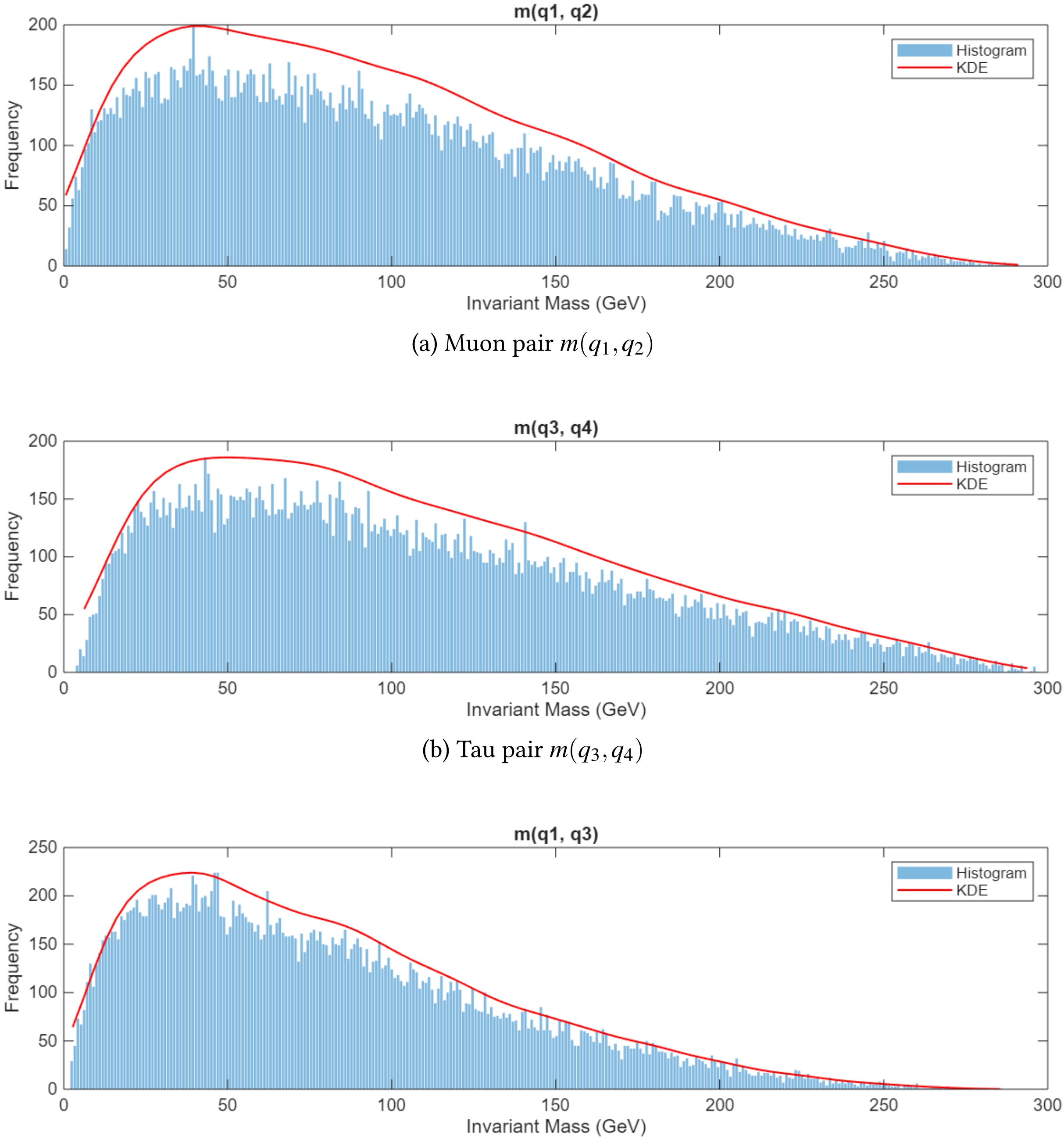
Figure 4.3: Angular distributions $\cos\theta_{12}$ and $\cos\theta_{34}$, where θ_{ij} is the angle between the momentum vectors of particles q_i and q_j.

The angular distributions (Fig. 4.3) provide further characterization of the process by revealing spatial correlations between final-state particles. While all events result in the same observable final state, differences in angular and mass distributions already hint at distinct underlying dynamics. The next sections explore each scenario individually and in more detail.

4.2 Scenario 1: Uncorrelated Production

Scenario 1 serves as a baseline case with fully uncorrelated production, where all final-state particles are generated independently, without intermediate resonances.

The invariant mass distributions in Fig. 4.4 exhibit no sharp peaks, consistent with the absence of intermediate resonances. Instead, the spectra are broad and continuous, shaped purely by kinematic phase space. Lower invariant masses are statistically favored, as they correspond to configurations where the two particles are emitted with small opening angles or where one carries significantly less momentum than the other — both of which occur more frequently within the allowed kinematic range. In addition, the event generation procedure itself enhances this effect (see Sec. 3.3): since the energy of the first particle is sampled uniformly up to $\sqrt{s}/2$, soft momenta are more likely, further shifting the spectrum toward lower invariant mass combinations.



(a) Muon pair $m(q_1, q_2)$

(b) Tau pair $m(q_3, q_4)$

(c) Mixed pair $m(q_1, q_3)$

Figure 4.4: Pairwise invariant mass distributions in Scenario 1. The absence of intermediate resonances results in broad, smooth spectra.

Notably, the distribution for the mixed pair $m(q_1, q_3)$ falls off more steeply and does not reach the kinematic limit of 300 GeV, unlike the same-flavor pairs. This can be understood by considering how the momenta are assigned: since the particles in a mixed pair are generated independently and at different stages of the event construction, they are less likely to simultaneously have large momenta pointing in similar directions. In contrast, the muon and tau pairs (q_1, q_2) and (q_3, q_4), though still uncorrelated in Scenario 1, are generated in closer succession and with more evenly distributed energy budgets, increasing the chance of both particles contributing significantly to the invariant mass. As a result, high-mass configurations are more accessible for same-flavor pairs, while for mixed pairs, such configurations are kinematically disfavored.

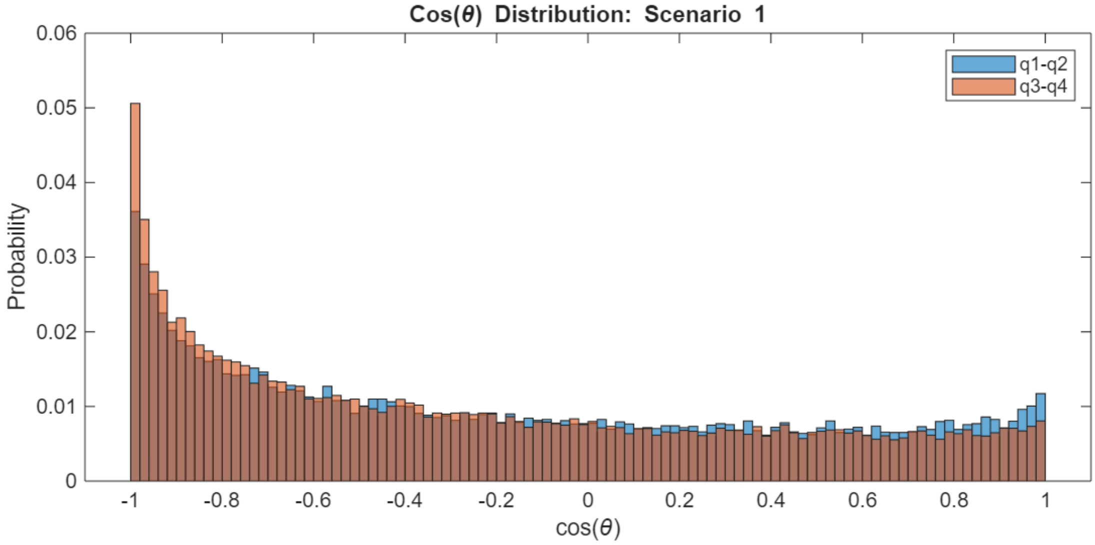


Figure 4.5: Angular distributions $\cos\theta_{12}$ and $\cos\theta_{34}$ in Scenario 1, corresponding to the muon and tau pairs, respectively.

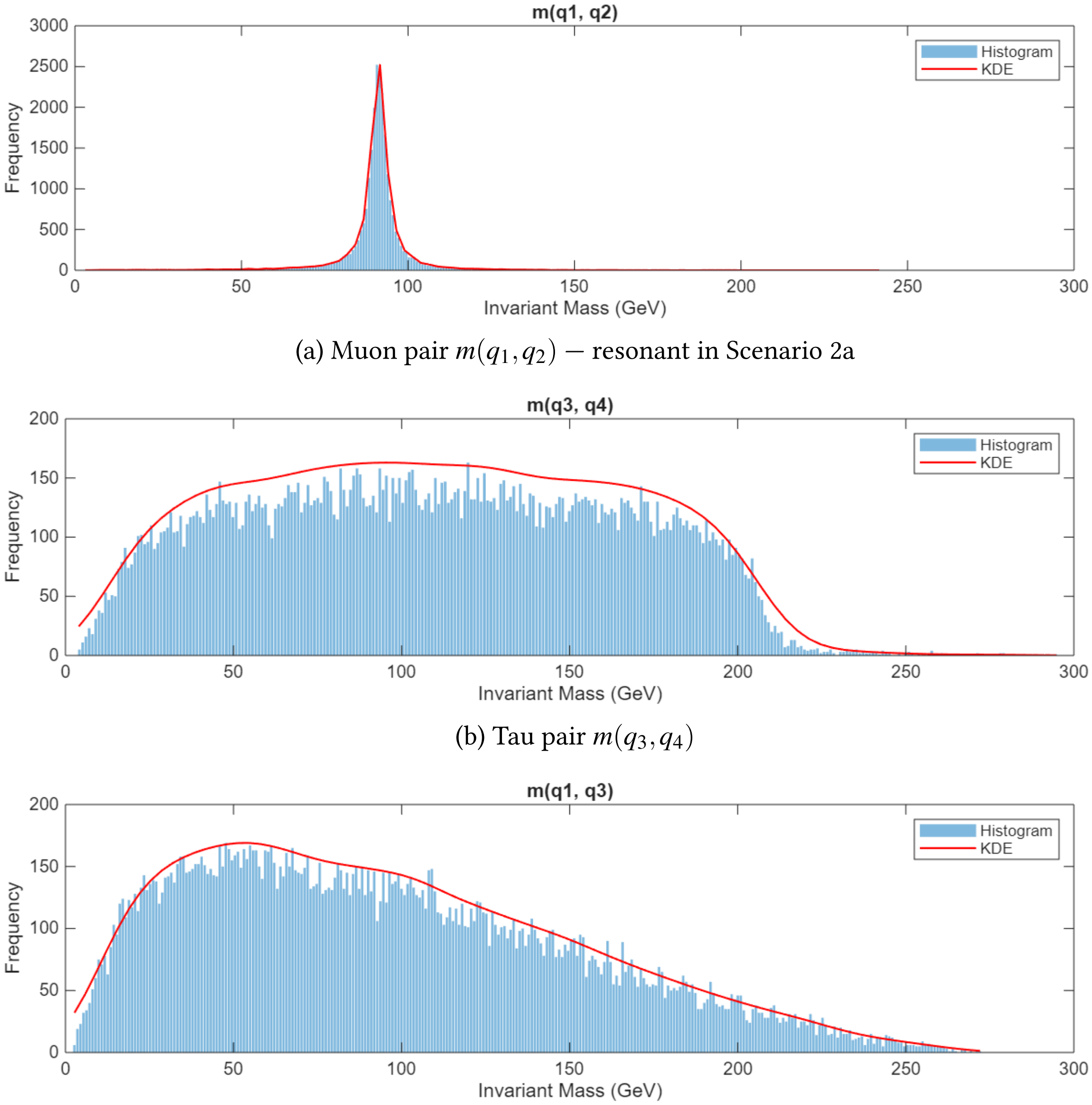
The angular distributions in Fig. 4.5 span the full range from $\cos\theta = -1$ to $+1$, corresponding to back-to-back and collinear particle emission, respectively. In the absence of any intermediate resonances, the resulting shapes are broadly distributed, as expected from unstructured phase-space production.

Both distributions display a mild preference for back-to-back emission (peak near $\cos\theta \approx -1$), which naturally arises when particles are produced with opposite momenta to balance the total system. The effect is particularly noticeable for the tau pair, likely due to their larger mass altering the kinematics slightly. Overall, this distribution acts as a reference, helping to identify angular asymmetries or directional preferences in the scenarios involving intermediate Z bosons.

4.3 Scenario 2a and 2b: Single Resonance

Scenarios 2a and 2b are mirrored cases in which one particle pair originates from a Z-boson decay, while the other remains uncorrelated: in Scenario 2a, the resonance appears in the muon pair (q_1, q_2), and in Scenario 2b, in the tau pair (q_3, q_4). These configurations allow for a direct comparison between resonant and non-resonant pairings within the same final state.

4.3.1 Scenario 2a: $Z \to \mu^+\mu^-$



(a) Muon pair $m(q_1, q_2)$ — resonant in Scenario 2a

(b) Tau pair $m(q_3, q_4)$

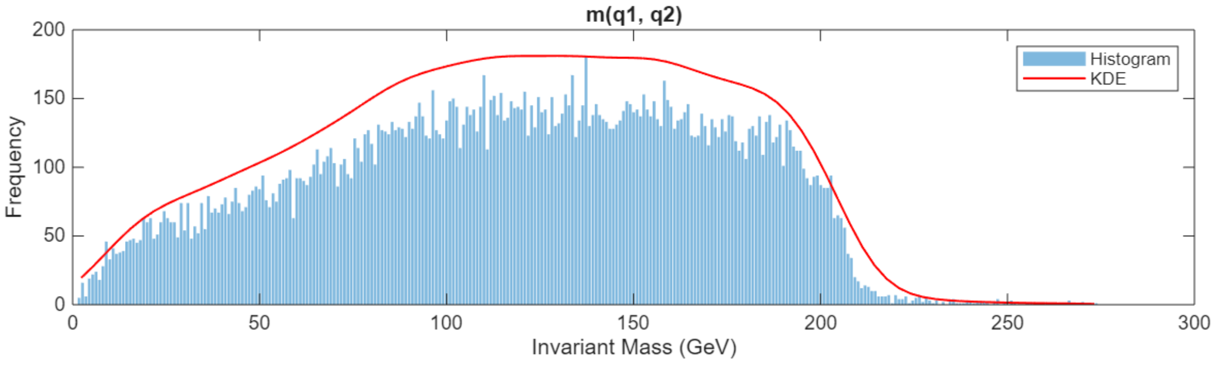
(c) Mixed pair $m(q_1, q_3)$

Figure 4.6: Pairwise invariant mass distributions in Scenario 2a.

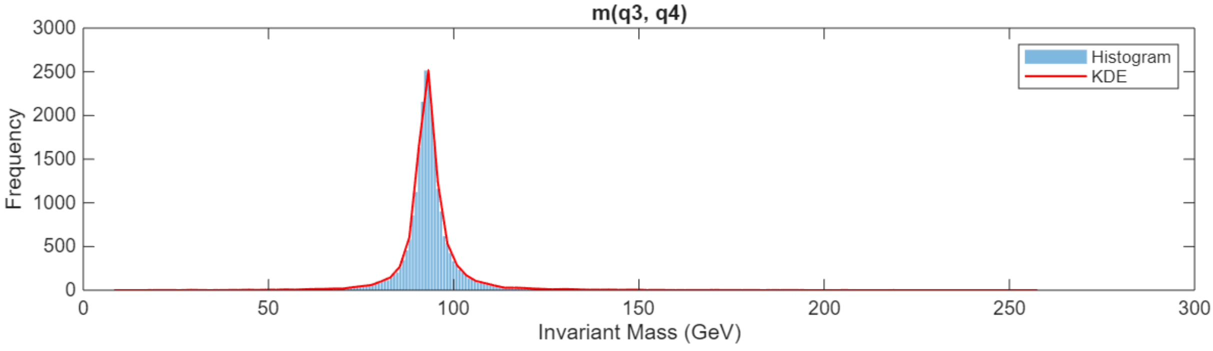
The peak observed in the muon invariant mass $m(q_1, q_2)$ confirms the presence of a Z-boson decay. In contrast, the tau pair $m(q_3, q_4)$, which is generated independently and does not originate from a common intermediate resonance, exhibits a nearly uniform and symmetric distribution across a wide mass range with a drop-off after about 200 GeV. This shape contrasts with the distribution in Scenario 1, where the spectrum is skewed toward lower invariant masses. In Scenario 2a, the tau pair is generated from the remaining 4-momentum after the $Z \to \mu^+\mu^-$ decay, which constrains the available phase space. Unlike the fully uncorrelated production in Scenario 1, this constraint suppresses highly asymmetric configurations and leads to a more uniform distribution. The result is a nearly flat spectrum, where low and high invariant masses occur with comparable probability within the constrained kinematic range.

These differences illustrate how a single resonance can indirectly influence the kinematics of the remaining pair, even in the absence of direct correlations.

4.3.2 Scenario 2b: $Z \to \tau^+\tau^-$

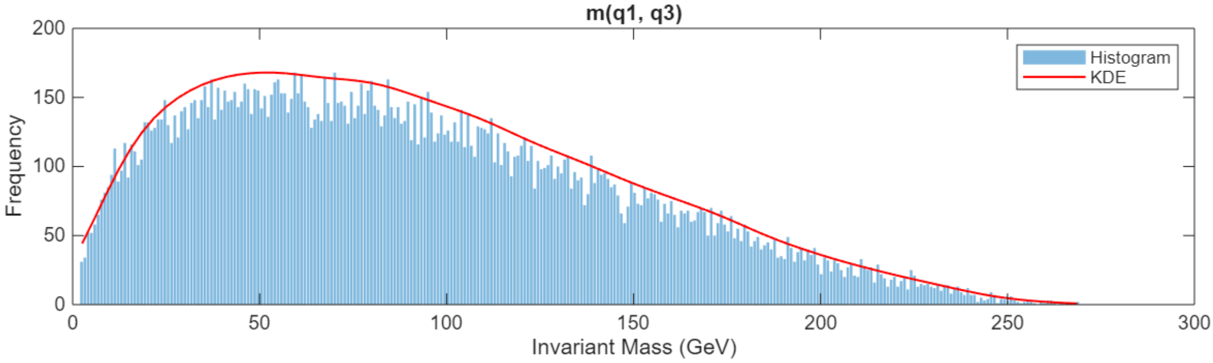


(a) Muon pair $m(q_1, q_2)$



(b) Tau pair $m(q_3, q_4)$ — resonant in Scenario 2b

Figure 4.7: Pairwise invariant mass distributions in Scenario 2b.



(c) Mixed pair $m(q_1, q_3)$

Figure 4.7: (continued) Pairwise invariant mass distributions in Scenario 2b.

The tau invariant mass distribution $m(q_3, q_4)$ in Scenario 2b exhibits a sharp peak near m_Z, confirming the presence of a resonant $Z \to \tau^+\tau^-$ decay. In contrast, the muon pair $m(q_1, q_2)$, which is generated independently, follows a broad, phase-space-like distribution. Compared to the tau spectrum in Scenario 2a, the muon distribution appears skewed toward higher invariant masses and drops off steeply, as expected from the limited energy available after producing a resonant Z-boson. This rightward shift arises from the lower mass of the muons: for the same available momentum, muons are more easily boosted into configurations where both particles carry sizable, similarly directed momenta—conditions that yield high invariant masses.

The mixed pair $m(q_1, q_3)$ remains uncorrelated in both Scenarios 2a and 2b, and its distribution closely resembles that of Scenario 1. This confirms that the resonance only affects the kinematics of directly involved particles, while other pairings retain their phase-space-driven shape.

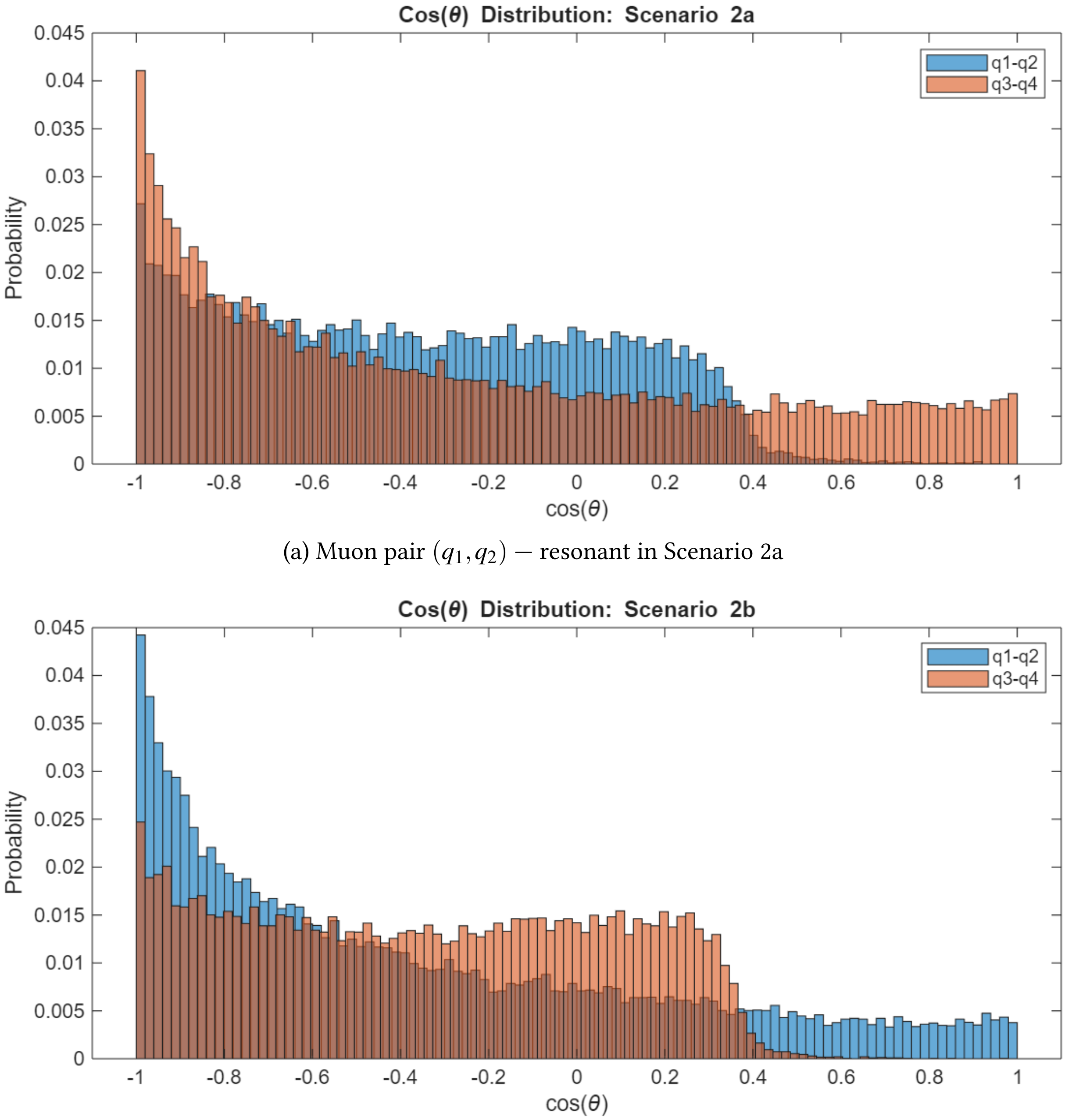
4.3.3 Angular Distributions in Scenarios 2a and 2b

Figure 4.8 shows the angular distributions for the muon and tau pairs in Scenarios 2a and 2b. In both cases, a clear peak near $\cos\theta = -1$ indicates back-to-back emission. This angular structure is characteristic of particle pairs originating from an intermediate Z-boson. Since $Z \to \ell^+\ell^-$ is a two-body decay, the leptons are emitted with opposite momenta in the rest frame of the Z, and this pattern is mostly preserved after boosting. In the full summed angular distribution (Fig. 4.3), these peaks contribute significantly to the rise near $\cos\theta = -1$. The drop around $\cos\theta \approx 0.4$ indicates a kinematic limit: more aligned configurations are not accessible for such decays within the available phase space. [1]

Independently produced pairs, on the other hand, show a broader angular spread from -1 to $+1$, without a strong directional preference. Their shape is dominated by available phase

[1] The exact limit depends on the chosen kinematic configuration and may vary for other setups.

space and closely resembles the distributions observed in Scenario 1 (Sec. 4.2).



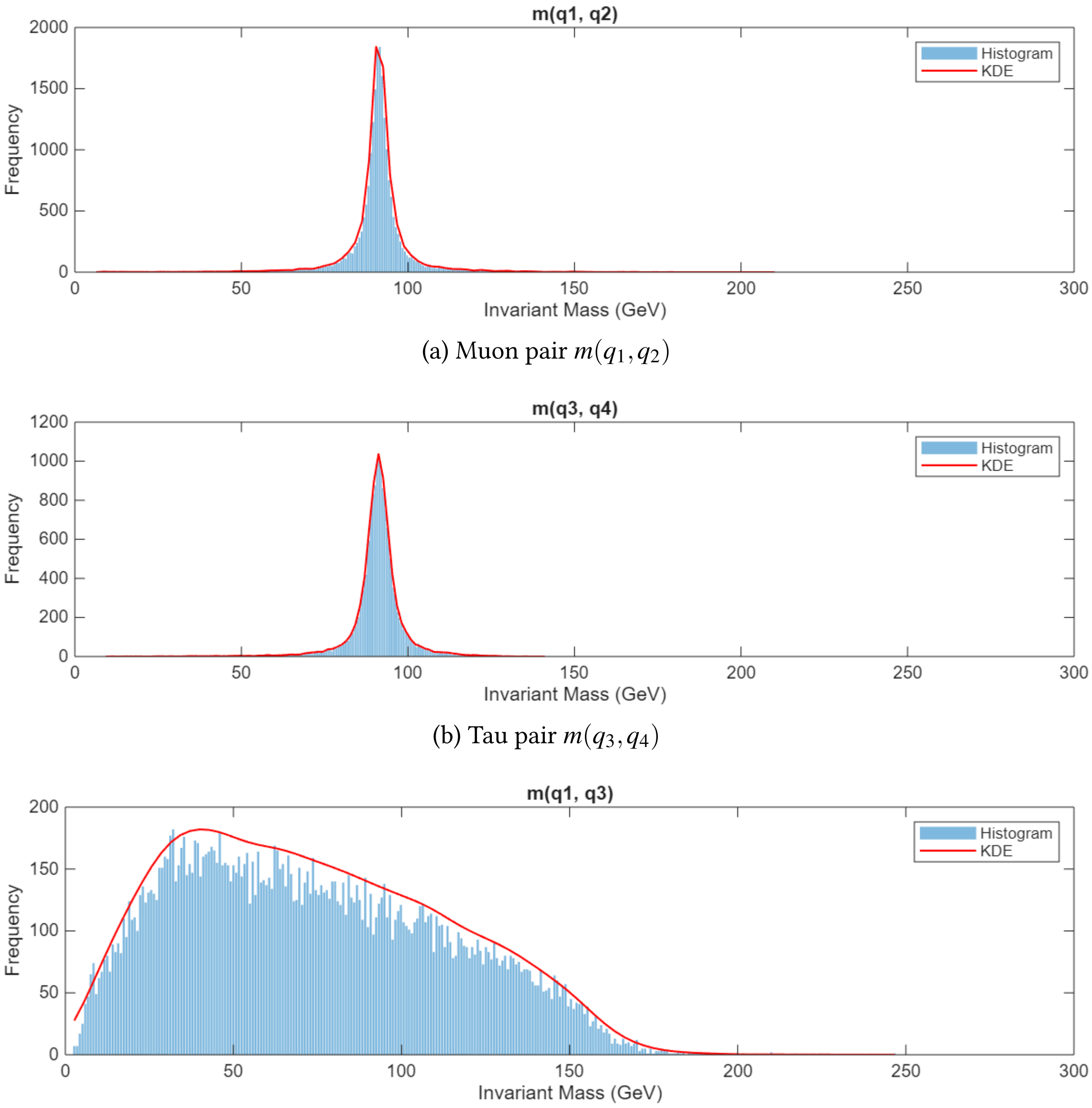
(a) Muon pair (q_1, q_2) — resonant in Scenario 2a

(b) Tau pair (q_3, q_4) — resonant in Scenario 2b

Figure 4.8: Angular distributions of the muon and tau pairs in Scenarios 2a and 2b.

4.4 Scenario 4: Double Resonance

In Scenario 4, both particle pairs originate from intermediate Z-bosons, leading to a fully resonant final state. This configuration represents the cleanest signal topology and is expected to show strong kinematic signatures of two-body decays.



(a) Muon pair $m(q_1, q_2)$

(b) Tau pair $m(q_3, q_4)$

(c) Mixed pair $m(q_1, q_3)$

Figure 4.9: Pairwise invariant mass distributions in Scenario 4. Both same-flavor pairs originate from intermediate Z-bosons.

Both the muon and tau pairs show sharp peaks around m_Z, consistent with the presence of two intermediate resonant decays. The shape of the mixed pair distribution $m(q_1, q_3)$, however, shifts significantly compared to previous cases. It resembles the broad continuum seen in Scenario 1 but drops off earlier, without extending beyond 200 GeV. This reflects a reduced phase space: the combined energy available for the uncorrelated pair is now further limited due to the presence of two massive Z-bosons in the intermediate state.

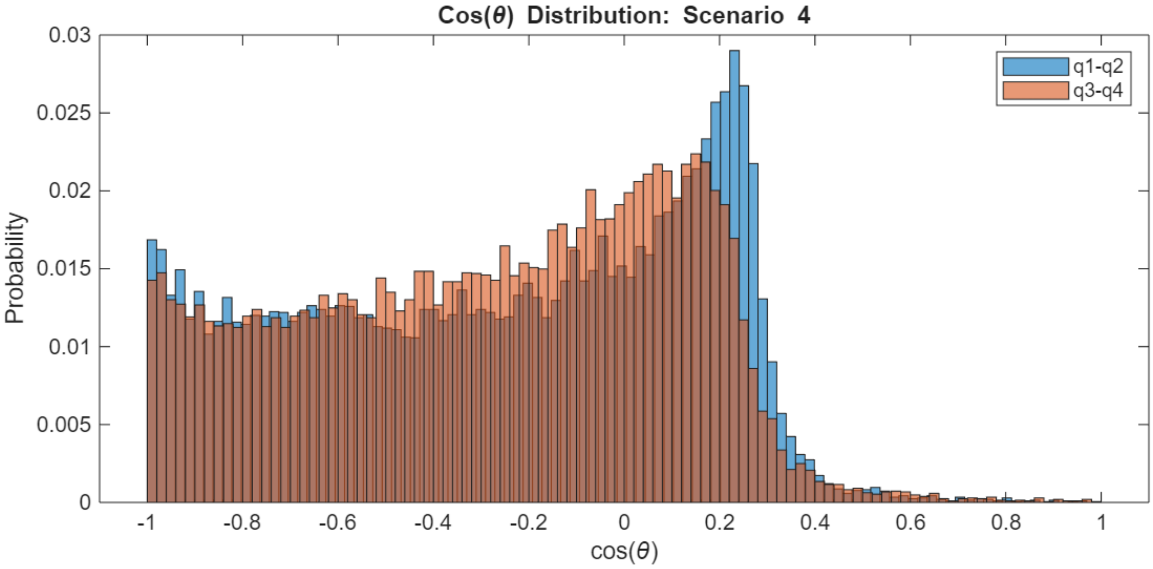


Figure 4.10: Angular distributions of the muon and tau pairs in Scenario 4.

The angular distributions in Fig. 4.10 deviate from the expected back-to-back pattern seen in previous cases. Instead of peaking at $\cos\theta = -1$, both pairs show broad structures with a maximum closer to $\cos\theta \approx 0.2-0.3$. This change arises from the presence of two intermediate Z-bosons: since both must share the available energy and recoil against each other to conserve momentum, their decay products are no longer emitted dominantly back-to-back in the lab frame. Instead, the decay angles are altered by the motion of the parent bosons, leading to a compression of the opening angle, and thus more central values appear. The muon pair distribution appears slightly more shifted toward higher $\cos\theta$, due to the lower mass of the muons allowing a stronger boost in the direction of the recoiling Z. This is also visible in the summed angular distribution (Fig. 4.3) as a secondary peak, providing a hint of events with two resonant Z-decays.

Altogether, the analysis of individual scenarios reveals how invariant mass and angular observables indicate the presence or absence of intermediate resonances. These differences, visible both in isolated plots and the combined distributions, allow for a clear identification of background sub-processes and help disentangle the underlying event topology.

CHAPTER 5

Conclusion and Outlook

In this work, a simulation framework was developed to investigate the kinematics of a vector boson scattering process $e^+e^- \to \mu^+\mu^-\tau^+\tau^-$ involving two intermediate Z bosons and to compare it with three background scenarios. Each process was implemented in MATLAB, and event samples were generated using phase space sampling with the appropriate physical constraints. The resulting invariant mass and angular distributions allowed a direct comparison of the kinematic features of the signal process with those of the backgrounds.

Overall, the thesis demonstrates that kinematic observables, especially invariant mass peaks and angular distributions, carry distinct imprints of the underlying event topology. The signal scenario exhibits the clearest signature, characterized by the presence of two intermediate resonances and modified angular patterns. In contrast, the background scenarios either lack these double-resonant features or display broader, phase-space-driven distributions.

In particular, Scenario 4 shows two sharp invariant mass peaks near m_Z (Fig. 4.9), consistent with the decay of two intermediate Z-bosons. This configuration also leads to modified angular patterns: rather than a dominant back-to-back emission, both lepton pairs show a maximum at angles around $\cos\theta \approx 0.2-0.3$, as seen in Fig. 4.10. This feature persists in the summed angular distribution (Fig. 4.3) as a visible secondary peak, providing an indication of double-resonant contributions.

Background scenarios show a contrasting behavior. The fully uncorrelated case (Scenario 1) leads to smooth invariant mass spectra (Fig. 4.4) and a near-flat angular distribution dominated by phase space (Fig. 4.5). Single-resonant scenarios (2a and 2b) produce hybrid shapes, where one lepton pair forms a peak around m_Z, while the other remains broad. In these cases, the angular distributions of the resonant pair show a strong back-to-back preference (Fig. 4.8), while the uncorrelated pair stays almost uniform. This contrast within a single event helps identify intermediate boson decays even in mixed topologies.

The simulations provide valuable qualitative insights into the expected signatures of VBS-like processes. The modular design of the code and the ability to control individual aspects of the event generation (such as which momenta are sampled directly and which are reconstructed) proved to be useful for tuning the kinematics and exploring different scenarios.

Outlook

Several directions for future development remain open. Most notably, the computational efficiency of the current simulation could be significantly improved. The generation of 25,000 to 40,000 valid events can take up to one hour, which becomes limiting when exploring additional parameter sets or refining the sampling. Possible improvements include optimizing the momentum generation strategy, implementing better rejection sampling techniques, or migrating the code to a faster programming language such as C++.

Additionally, the current approach could be extended to include more realistic features such as matrix element weighting, detector effects, or parton distribution functions. This would bring the simulation closer to experimental conditions and allow for a better comparison between different kinematic configurations. Furthermore, more background processes (e.g. single-resonant or non-resonant Z pair production via gluon fusion) could be added to see whether the distinguishing features identified here remain visible in a broader range of cases.

Finally, another possible extension could involve adapting the simulation to other center-of-mass energies or to processes involving different intermediate states, such as $W^{\pm}$ bosons or hypothetical new resonances. While not intended as a replacement for full-scale event generators, the framework developed here could serve as a lightweight testing ground for examining how specific kinematic features evolve under modified theoretical assumptions.

Bibliography

[1] B. Biedermann, A. Denner, and M. Pellen, “Large Electroweak Corrections to Vector-Boson Scattering at the Large Hadron Collider,” *Physical Review Letters*, vol. 118, p. 261801, jun 2017. → [p1]

[2] D. Green, P. Meade, and M.-A. Pleier, “Multiboson interactions at the LHC,” *Reviews of Modern Physics*, vol. 89, Sep 2017. → [p1]

[3] R. Covarelli, M. Pellen, and M. Zaro, “Vector-Boson scattering at the LHC: Unraveling the electroweak sector,” *International Journal of Modern Physics A*, vol. 36, May 2021. → [p1], [p3]

[4] A. M. Sirunyan, A. Tumasyan, W. Adam, and et al., “Measurement of vector boson scattering and constraints on anomalous quartic couplings from events with four leptons and two jets in proton–proton collisions at $\sqrt{s} = 13$ TeV,” *Physics Letters B*, vol. 774, p. 682–705, Nov 2017. → [p2]

[5] P. Kozów, L. Merlo, S. Pokorski, and M. Szleper, “Same-sign WW scattering in the HEFT: discoverability vs. EFT validity,” *Journal of High Energy Physics*, vol. 2019, p. 021, Jul 2019. → [p2]

[6] S. Chatrchyan, V. Khachatryan, A. M. Sirunyan, and et al., “Observation of Z decays to four leptons with the CMS detector at the LHC,” *Journal of High Energy Physics*, vol. 2012, Dec 2012. → [p2]

[7] M. Bendel, “Production of Z Bosons in Proton–Proton Collisions at $\sqrt{s}$ = 10 TeV: Expectations for Early Measurements at the ATLAS Experiment,” Dec 2010. → [p2]

[8] M. Grossi, J. Novak, B. Kerševan, and D. Rebuzzi, “Comparing traditional and deep-learning techniques of kinematic reconstruction for polarization discrimination in vector boson scattering,” *The European Physical Journal C*, vol. 80, p. 1132, Dec 2020. → [p2]

[9] P. Acton, G. Alexander, J. Allison, and et al., “A test of higher order electroweak theory in Z^0 decays to two leptons with an associated pair of charged particles,” *Physics Letters B*, vol. 287, no. 4, pp. 389–400, 1992. → [p2]

[10] A. Adam, O. Adriani, M. Aguilar-Benitez, and et al., “A study of four-fermion processes at LEP,” *Physics Letters B*, vol. 321, no. 3, pp. 283–294, 1994. → [p2]

[11] A. Denner, S. Dittmaier, M. Roth, and D. Wackeroth, “Predictions for all processes e+e → fermions + ,” *Nuclear Physics B*, vol. 560, p. 33–65, Nov. 1999. → [p2]

[12] F. Berends, R. Pittau, and R. Kleiss, “All electroweak four-fermion processes in electron-positron collisions,” *Nuclear Physics B*, vol. 424, p. 308–342, Aug. 1994. → [p2], [p3]

[13] M. Jindal, D. Bourilkov, K. Mazumdar, and J. Singh, “Drell–Yan process at Large Hadron Collider,” *PRAMANA c Indian Academy of Sciences*, vol. 7615, 03 2011. → [p2]

[14] B. Jäger and S. L. P. Chavez, “Electroweak w+w+ production in association with three jets at nlo qcd matched with parton shower,” *Journal of High Energy Physics*, vol. 2025, Jan. 2025. → [p3]

[15] M. Thomson, *Modern Particle Physics.* Cambridge University Press, 2013. → [p3]

[16] G. Jaeger, “Are Virtual Particles Less Real?,” *Entropy*, vol. 21, no. 2, 2019. → [p3]

[17] P. Janot and S. Jadach, “Improved Bhabha cross section at LEP and the number of light neutrino species,” *Phys. Lett. B*, vol. 803, p. 135319, 2020. → [p3]

[18] G. Breit and E. Wigner, “Capture of Slow Neutrons,” *Phys. Rev.*, vol. 49, pp. 519–531, Apr 1936. → [p4]

[19] F. Giacosa, A. Okopińska, and V. Shastry, “A simple alternative to the relativistic Breit–Wigner distribution,” *The European Physical Journal A*, vol. 57, Dec 2021. → [p4]

[20] R. Hirayama, J. Staudenmaier, and H. Elfner, “Effective spectral function of vector mesons via lifetime analysis,” *Phys. Rev. C*, vol. 107, p. 025208, Feb 2023. → [p4]

[21] H. Brooks, P. Skands, and R. Verheyen, “Interleaved Resonance Decays and Electroweak Radiation in Vincia,” 2022. → [p4]

[22] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale, and S. Schumann, “Exploring phase space with Neural Importance Sampling,” *SciPost Physics*, vol. 8, Apr. 2020. → [p4], [p7]

[23] P. Nason and P. Z. Skands, “Monte Carlo Event Generators,” *Particle Data Group Review*, vol. 39, 2013. Revised September 2013. → [p4]

[24] S. Weinberg, *The Quantum Theory of Fields, Volume 1: Foundations.* Cambridge University Press, 1996. → [p5]

[25] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory.* Addison-Wesley, 1995. → [p5]

[26] A. Buckley, J. Butterworth, L. Lonnblad, H. Hoeth, J. Monk, H. Schulz, and F. Siegert, “Rivet user manual,” *Comput. Phys. Commun.*, vol. 184, pp. 2803–2819, 2011. → [p7]

[27] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, and et al., “An Introduction to PYTHIA 8.2,” *Comput. Phys. Commun.*, vol. 191, pp. 159–177, 2015. → [p7]

[28] T. Gleisberg, S. Hoeche, F. Krauss, M. Schonherr, S. Schumann, F. Siegert, and J. Winter, "Event generation with SHERPA 1.1," *JHEP*, vol. 02, p. 007, 2009. → [p7]

[29] C. Bierlich, S. Chakraborty, N. Desai, and et al., "A comprehensive guide to the physics and usage of PYTHIA 8.3," 2022. → [p7]

[30] J. Alwall, R. Frederix, S. Frixione, and et al., "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations," *Journal of High Energy Physics*, vol. 2014, July 2014. → [p7]

[31] E. Bothmann, L. Flower, C. Gütschow, S. Höche, M. Hoppe, J. Isaacson, M. Knobbe, F. Krauss, P. Meinzinger, D. Napoletano, A. Price, D. Reichelt, M. Schönherr, S. Schumann, and F. Siegert, "Event generation with Sherpa 3," 2024. → [p7]

[32] M. Bähr, S. Gieseke, M. A. Gigg, D. Grellscheid, K. Hamilton, O. Latunde-Dada, S. Plätzer, P. Richardson, M. H. Seymour, A. Sherstnev, and B. R. Webber, "Herwig++ physics and manual," *The European Physical Journal C*, vol. 58, p. 639–707, Nov. 2008. → [p7]

[33] F. James, "Monte Carlo theory and practice," *Rep. Prog. Phys.*, vol. 43, pp. 1145–1189, 1980. → [p7]

Appendix

A.1 Error Propagation and Convergence Rate

The iterative energy correction method updates the energy of the fourth particle as:

$$E_4^{(n+1)} = \sqrt{m_4^2 + |\mathbf{q}_4|^2}. \tag{A.1}$$

If the estimated energy at iteration n deviates from the exact solution E_4, the error is defined as:

$$e_n = E_4^{(n)} - E_4. \tag{A.2}$$

A.1.1 Error Evolution in the Iteration

The function governing the iteration is:

$$f(E) = \sqrt{m_4^2 + |\mathbf{q}_4|^2}. \tag{A.3}$$

Since E_4 is the correct solution, it must satisfy:

$$E_4 = f(E_4). \tag{A.4}$$

Expanding $f(E)$ in a first-order Taylor series around E_4:

$$f(E_4 + e_n) \approx E_4 + f'(E_4)e_n + \mathscr{O}(e_n^2). \tag{A.5}$$

Thus, the error at iteration $n+1$ is:

$$e_{n+1} = f'(E_4)e_n + \mathscr{O}(e_n^2). \tag{A.6}$$

A.1.2 Convergence Rate

The derivative of $f(E)$ is:

$$f'(E) = \frac{1}{2}\frac{1}{\sqrt{m_4^2 + |\mathbf{q}_4|^2}}. \tag{A.7}$$

Substituting $E_4 = \sqrt{m_4^2 + |\mathbf{q}_4|^2}$:

$$f'(E_4) = \frac{1}{2E_4}. \tag{A.8}$$

Thus, the error update equation becomes:

$$e_{n+1} = \frac{1}{2E_4} e_n. \tag{A.9}$$

Since the error shrinks geometrically, the convergence rate is:

$$r = \frac{1}{2E_4}. \tag{A.10}$$

Since E_4 is large, the method converges quickly.

A.2 Pseudorapidity Distributions

Due to the isotropic nature of the generated events, the pseudorapidity distributions follow an approximately Gaussian shape. These distributions do not provide additional insight into event properties but are included here for completeness.

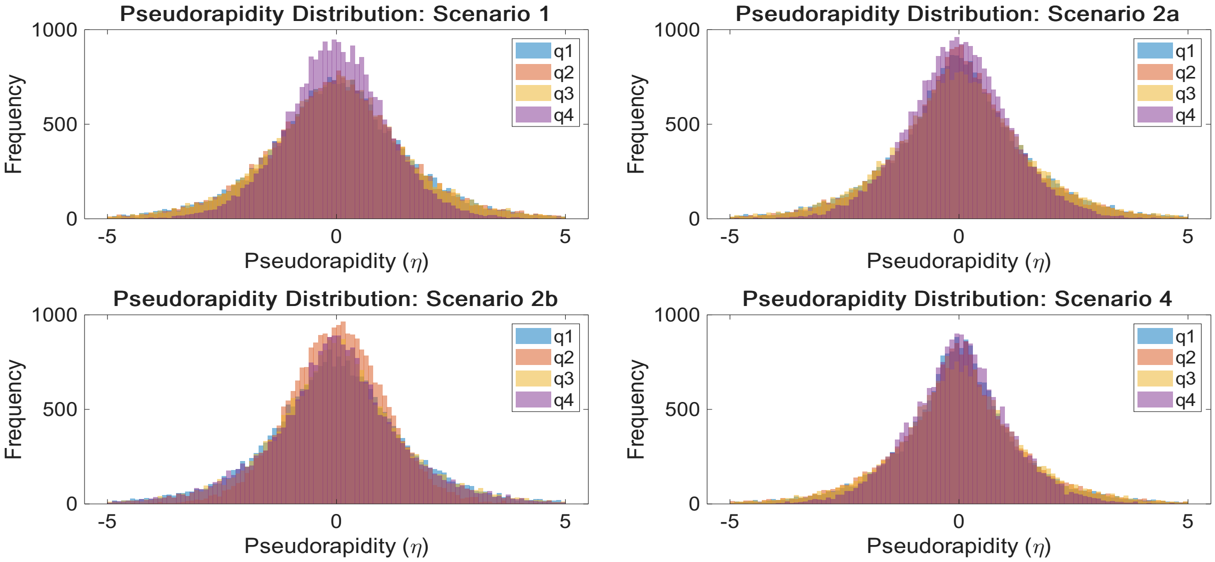


Figure A.1: Pseudorapidity distributions for different event scenarios.

A.3 Comparison of KDE Kernels

Kernel Density Estimation (KDE) is used in this analysis to provide a smooth probability density function for the invariant mass distributions. The choice of the kernel function can influence the shape and accuracy of the estimated distribution. This section presents a comparison of different kernels applied to invariant mass distributions.

A.3.1 Kernel Functions and Implementation

The following kernels were used:

- **Gaussian Kernel**:
$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}, \quad u \in (-\infty, \infty). \tag{A.11}$$
- **Epanechnikov Kernel**: The optimal kernel in terms of minimizing mean integrated squared error (MISE):
$$K(u) = \begin{cases} \frac{3}{4}(1-u^2), & |u| \leq 1, \\ 0, & |u| > 1. \end{cases} \tag{A.12}$$
- **Biweight Kernel**:
$$K(u) = \begin{cases} \frac{15}{16}(1-u^2)^2, & |u| \leq 1, \\ 0, & |u| > 1. \end{cases} \tag{A.13}$$
- **Triweight Kernel**:
$$K(u) = \begin{cases} \frac{35}{32}(1-u^2)^3, & |u| \leq 1, \\ 0, & |u| > 1. \end{cases} \tag{A.14}$$

As an approximation for the bandwidth h Silverman's rule of thumb chosen:

$$h = 1.06 \cdot \sigma \cdot n^{-1/5}. \tag{A.15}$$

A.3.2 MATLAB Implementation

The KDE estimation was implemented in MATLAB for different kernel functions. The process involves computing density estimates at a predefined grid of points and normalizing the results for visualization.

```
        %% Manual KDE Implementation with Different Kernels
        % Generate grid for evaluation
        x_grid = linspace(0, 300, 1000);

        % Bandwidth estimation (Silverman’s rule)
        h = 1.06 * std(mq34_nonzero) * numel(mq34_nonzero)^(-1/5);

        % Define kernels
        kernels = struct(...
            ’Gaussian’, @(u) (1/sqrt(2*pi)) * exp(-0.5 * u.^2), ...
            ’Epanechnikov’, @(u) 0.75 * max(0, (1 - u.^2)), ...
            ’Biweight’, @(u) (15/16) * max(0, (1 - u.^2).^2), ...
            ’Triweight’, @(u) (35/32) * max(0, (1 - u.^2).^3) ...
        );

        % Calculate KDE for each kernel
        kde_results = struct();
        fields = fieldnames(kernels);

        for k = 1:numel(fields)
            kernel_name = fields{k};
            kernel_function = kernels.(kernel_name);
            kde_values = zeros(size(x_grid));

            for j = 1:length(x_grid)
                x = x_grid(j);
                kde_values(j) = sum(kernel_function((x - mq34_nonzero) / h)) / (numel(mq34_nonzero) * h);
            end

            kde_results.(kernel_name) = kde_values;
        end

```

Listing 1: Manual KDE Implementation with Different Kernels

A.3.3 Results and Comparison

To evaluate the effect of different kernels, KDE estimates were plotted alongside histograms of the invariant mass distributions. Two representative cases are shown:

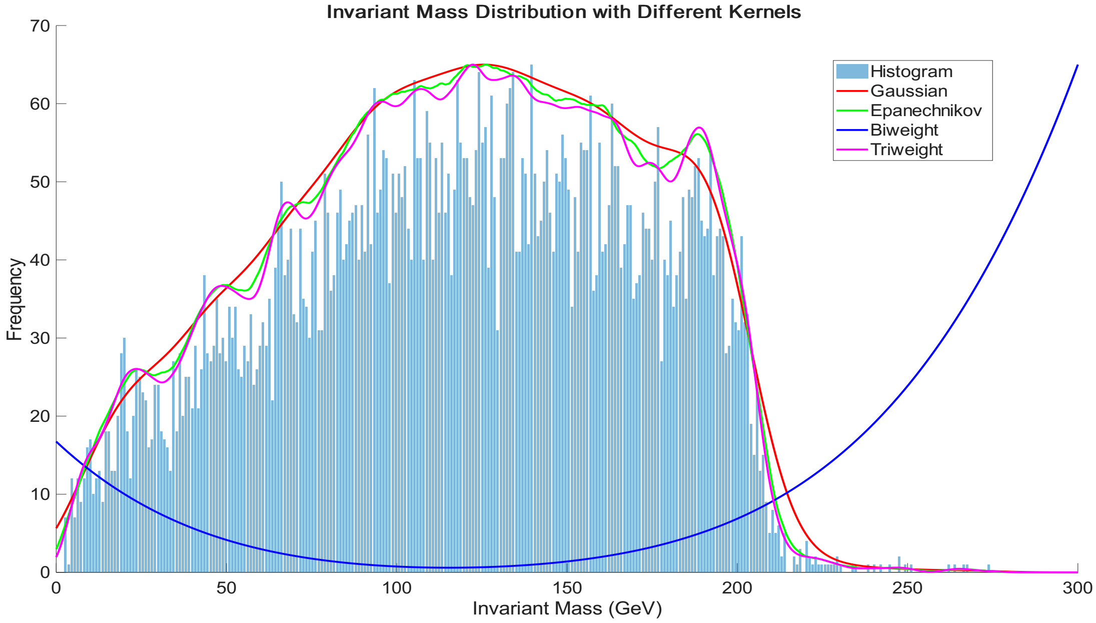


Figure A.2: Comparison of different KDE kernels applied to an invariant mass distribution where the particle pair is produced independently of the Z-boson decay.

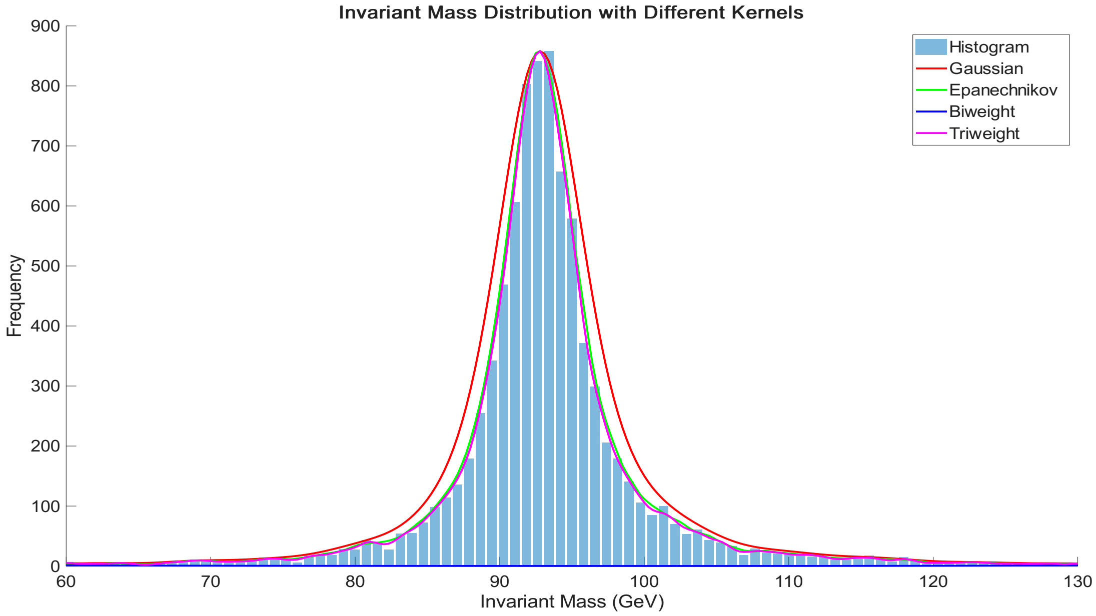


Figure A.3: Comparison of KDE kernels in the invariant mass region around the Z-boson peak.

For the final implementation, the Epanechnikov kernel was selected.

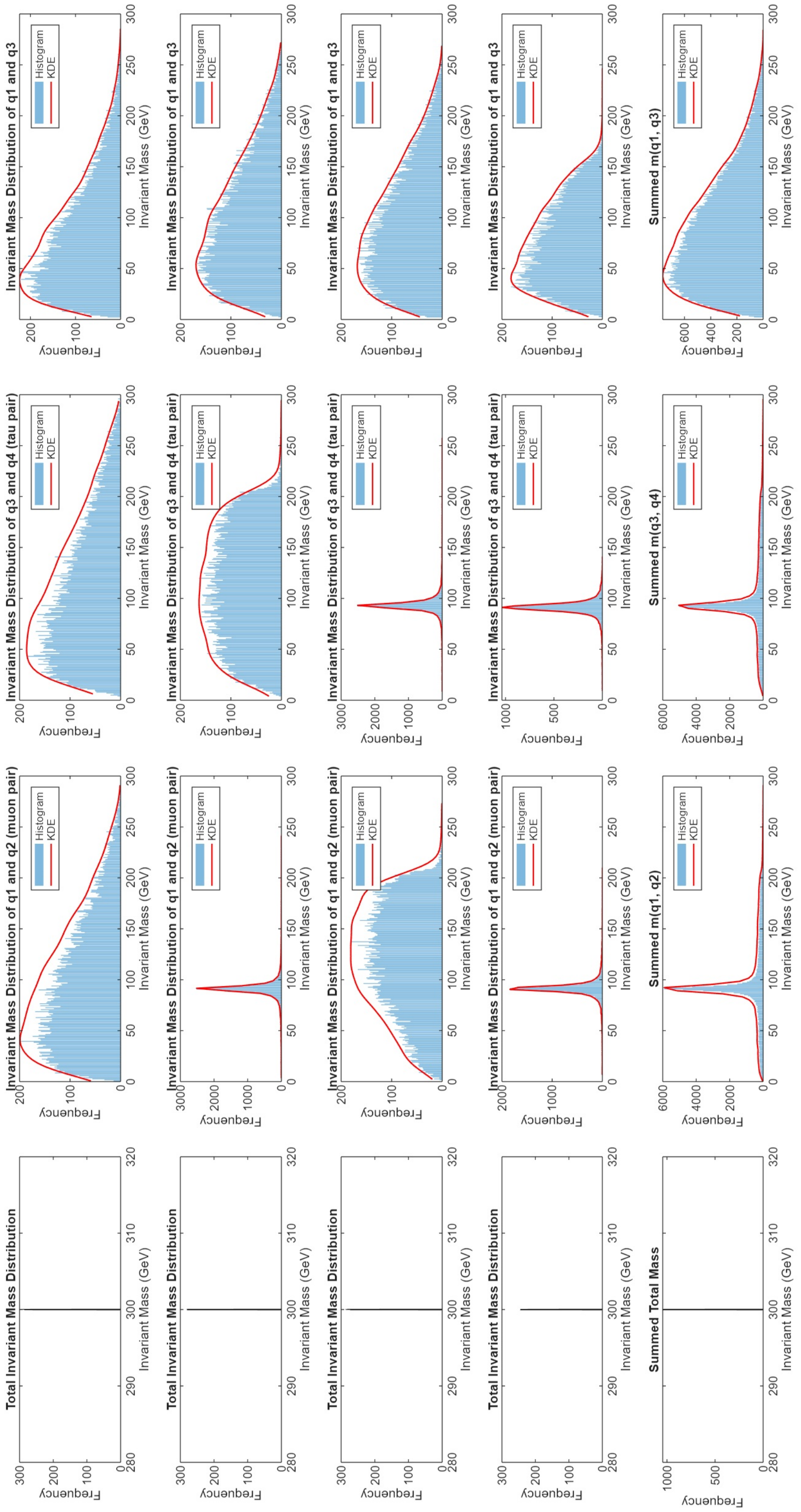


Figure A.4: Overview of all invariant mass distributions used in the analysis, grouped by scenario and pairing. This matrix provides a visual summary of the simulation output.

Code

This section contains the full MATLAB implementation of the simulation framework. The scripts include event generation, kinematic calculations, and data visualization.

Main Simulation Script: main.m

```
clc; clear;

%% Constants
mE = 0.000511;
mMu = 0.105658;
mTau = 1.776861;
mZ = 91.1876;
gammaZ = 2.4952; % decay width for Z boson
eElectron = 150;
sqrtS = 2 * eElectron;

%% Simulation parameters
numEvents = 25000;
scenarios = {@Scenario1, @Scenario2a, @Scenario2b, @Scenario4};
scenarioNames = {’Scenario 1’, ’Scenario 2a’, ’Scenario 2b’, ’Scenario 4’};

allMTotal = [];
allMq12 = [];
allMq34 = [];
allMq13 = [];
allCosTheta12 = [];
allCosTheta34 = [];

%% Main Loop through all Scenarios
figure(’Name’, ’5x4 Grid of Invariant Mass Histograms’);
tiledlayout(5, 4);

figure(’Name’, ’5x4 Grid of Invariant Mass Histograms + KDE’);
tiledlayout(5, 4);

figure(’Name’, ’2x2 Grid of Rapidity Distributions’);
```

```
tiledlayout(2, 2);

figure(’Name’, ’2x2 Grid of Cos(theta) Distributions’);
tiledlayout(2, 2);

for scenarioIdx = 1:length(scenarios)
    scenarioFunction = scenarios{scenarioIdx};
    scenarioName = scenarioNames{scenarioIdx};

    [mTotal, mq12, mq34, mq13, cosTheta12, cosTheta34, pseudorapidity_q1, pseudorapidity_q2, pseudorapidity_q3, pseudorapidity_q4] = ...
    RunScenario(scenarioFunction, sqrtS, mMu, mTau, mZ, gammaZ, numEvents);

    % Filter valid data
    mNon0 = mTotal(mTotal > 0);
    mq12_nonzero = mq12(mq12 > 0);
    mq34_nonzero = mq34(mq34 > 0);
    mq13_nonzero = mq13(mq13 > 0);

    % Remove zeros (empty events)
    pseudorapidity_q1 = pseudorapidity_q1(pseudorapidity_q1 ~= 0);
    pseudorapidity_q2 = pseudorapidity_q2(pseudorapidity_q2 ~= 0);
    pseudorapidity_q3 = pseudorapidity_q3(pseudorapidity_q3 ~= 0);
    pseudorapidity_q4 = pseudorapidity_q4(pseudorapidity_q4 ~= 0);

    cosTheta12 = cosTheta12(cosTheta12 ~= 0);
    cosTheta34 = cosTheta34(cosTheta34 ~= 0);

    % Accumulate invariant masses and angular distributions
    allMTotal = [allMTotal, mNon0];
    allMq12 = [allMq12, mq12_nonzero];
    allMq34 = [allMq34, mq34_nonzero];
    allMq13 = [allMq13, mq13_nonzero];

    allCosTheta12 = [allCosTheta12, cosTheta12];
    allCosTheta34 = [allCosTheta34, cosTheta34];

    % Plot accumulated invariant masses in the 5x4 grid
    figure(1);
    % Column 1: Total invariant mass
    nexttile;
    PlotHistogram(mNon0, [’Total Mass: ’, scenarioName], ’Invariant Mass (GeV)’, [280 320], 100);

    % Column 2: m(q1, q2)
    nexttile;
    PlotHistogram(mq12_nonzero, [’m(q1, q2): ’, scenarioName], ’Invariant Mass (GeV)’, [0 300], 300);

    % Column 3: m(q3, q4)
    nexttile;
    PlotHistogram(mq34_nonzero, [’m(q3, q4): ’, scenarioName], ’Invariant Mass (GeV)’, [0 300], 300);

    % Column 4: m(q1, q3)
    nexttile;
    PlotHistogram(mq13_nonzero, [’m(q1, q3): ’, scenarioName], ’Invariant Mass (GeV)’, [0 300], 300);

    % Plot KDE-enhanced histograms
    figure(2);
    % Total invariant mass KDE
    nexttile;
    PlotHistogram(mNon0, ’Total Invariant Mass Distribution’, ’Invariant Mass (GeV)’, [280 320], 100);

    % m(q1, q2) KDE
    nexttile;
    PlotHistogramWithKDE(mq12_nonzero, ’Invariant Mass Distribution of q1 and q2 (muon pair)’, ’Invariant Mass (GeV)’, [0 300], 300);

    % m(q3, q4) KDE
    nexttile;
    PlotHistogramWithKDE(mq34_nonzero, ’Invariant Mass Distribution of q3 and q4 (tau pair)’, ’Invariant Mass (GeV)’, [0 300], 300);

    % m(q1, q3) KDE
    nexttile;
    PlotHistogramWithKDE(mq13_nonzero, ’Invariant Mass Distribution of q1 and q3’, ’Invariant Mass (GeV)’, [0 300], 300);

    % Pseudorapidity plot for each scenario
    figure(3); % Pseudorapidity distribution plot
    nexttile;
    PlotPseudorapidity(pseudorapidity_q1, pseudorapidity_q2, pseudorapidity_q3, pseudorapidity_q4, scenarioName);

    % Angular distribution plot for each scenario
    figure(4);
    nexttile;
    PlotCosThetaDistribution(cosTheta12, cosTheta34, scenarioName);

    % Individual invariant mass plots
    figure(’Name’, [’Invariant Mass Distributions: ’, scenarioName]);
    tiledlayout(’vertical’);
    nexttile; PlotHistogramWithKDE(mq12_nonzero, ’m(q1, q2)’, ’Invariant Mass (GeV)’, [0 300], 300);
    nexttile; PlotHistogramWithKDE(mq34_nonzero, ’m(q3, q4)’, ’Invariant Mass (GeV)’, [0 300], 300);
    nexttile; PlotHistogramWithKDE(mq13_nonzero, ’m(q1, q3)’, ’Invariant Mass (GeV)’, [0 300], 300);
end

% Plot the summed invariant masses (5th row)
figure(1); % Histogram figure
% Column 1: Total invariant mass
nexttile;
PlotHistogram(allMTotal, ’Summed Total Mass’, ’Invariant Mass (GeV)’, [280 320], 100);

% Column 2: m(q1, q2)
nexttile;
PlotHistogram(allMq12, ’Summed m(q1, q2)’, ’Invariant Mass (GeV)’, [0 300], 300);

% Column 3: m(q3, q4)
nexttile;
PlotHistogram(allMq34, ’Summed m(q3, q4)’, ’Invariant Mass (GeV)’, [0 300], 300);

% Column 4: m(q1, q3)
nexttile;
PlotHistogram(allMq13, ’Summed m(q1, q3)’, ’Invariant Mass (GeV)’, [0 300], 300);

figure(2); % KDE-enhanced figure
% Column 1: Total invariant mass KDE
nexttile;
PlotHistogram(allMTotal, ’Summed Total Mass’, ’Invariant Mass (GeV)’, [280 320], 100);

% Column 2: m(q1, q2) KDE
nexttile;
PlotHistogramWithKDE(allMq12, ’Summed m(q1, q2)’, ’Invariant Mass (GeV)’, [0 300], 300);
```

```matlab

% Column 3: m(q3, q4) KDE
nexttile;
PlotHistogramWithKDE(allMq34, 'Summed m(q3, q4)', 'Invariant Mass (GeV)', [0 300], 300);

% Column 4: m(q1, q3) KDE
nexttile;
PlotHistogramWithKDE(allMq13, 'Summed m(q1, q3)', 'Invariant Mass (GeV)', [0 300], 300);

figure('Name', 'Invariant Mass Distributions: Summed Process');
tiledlayout('vertical');
nexttile; PlotHistogram(allMTotal, 'Summed Total Invariant Mass', 'Invariant Mass (GeV)', [280 320], 100);
nexttile; PlotHistogramWithKDE(allMq12, 'Summed m(q1, q2)', 'Invariant Mass (GeV)', [0 300], 300);
nexttile; PlotHistogramWithKDE(allMq34, 'Summed m(q3, q4)', 'Invariant Mass (GeV)', [0 300], 300);
nexttile; PlotHistogramWithKDE(allMq13, 'Summed m(q1, q3)', 'Invariant Mass (GeV)', [0 300], 300);

% Summed angular distributions
figure('Name', 'Cos() Distribution: Summed Process');
PlotCosThetaDistribution(allCosTheta12, allCosTheta34, 'Summed Process');

%% Functions
function  [mTotal, mq12, mq34, mq13, cosTheta12, cosTheta34, pseudorapidity_q1, pseudorapidity_q2, pseudorapidity_q3, pseudorapidity_q4] = ...
    RunScenario(scenarioFunction, sqrtS, mMu, mTau, mZ, gammaZ, numEvents)
     mTotal = zeros(1, numEvents);
     mq12 = zeros(1, numEvents);
     mq34 = zeros(1, numEvents);
     mq13 = zeros(1, numEvents);
     cosTheta12 = zeros(1, numEvents);
     cosTheta34 = zeros(1, numEvents);
     pseudorapidity_q1 = zeros(1, numEvents);
     pseudorapidity_q2 = zeros(1, numEvents);
     pseudorapidity_q3 = zeros(1, numEvents);
     pseudorapidity_q4 = zeros(1, numEvents);

     %Run scenario simulation
     % Run multiple events
     parpool('local');
     parfor i = 1:numEvents
         switch nargin(scenarioFunction)
             case 3 % Scenario1
                 particles = scenarioFunction(sqrtS, mMu, mTau);
             case 5 % Scenario2a, Scenario2b, Scenario4
                 particles = scenarioFunction(sqrtS, mMu, mTau, mZ, gammaZ);
             otherwise
                 error(’Unexpected number of arguments for the scenario function.’);
         end

         if isempty(particles)
             continue;
         end
         mTotal(i) = ComputeInvariantMass(particles(’q1’), particles(’q2’), particles(’q3’), particles(’q4’));
         mq12(i) = ComputeInvariantMass(particles(’q1’), particles(’q2’));
         mq34(i) = ComputeInvariantMass(particles(’q3’), particles(’q4’));
         mq13(i) = ComputeInvariantMass(particles(’q1’), particles(’q3’));

         % Extract momentum vectors
         q1 = particles(’q1’);
         q2 = particles(’q2’);
         q3 = particles(’q3’);
         q4 = particles(’q4’);

         % Calculate angular distributions for pairs
         cosTheta12(i) = CalculateCosTheta(q1, q2);
         cosTheta34(i) = CalculateCosTheta(q3, q4);

         % Calculate pseudorapidity for all particles
         pseudorapidity_q1(i) = CalculatePseudorapidity(q1);
         pseudorapidity_q2(i) = CalculatePseudorapidity(q2);
         pseudorapidity_q3(i) = CalculatePseudorapidity(q3);
         pseudorapidity_q4(i) = CalculatePseudorapidity(q4);
     end
     delete(gcp);
 end

 function PlotHistogram(data, plotTitle, xLabel, xLimits, numBins)
     histogram(data, numBins);
     if exist(’xLimits’, ’var’) && ~isempty(xLimits)
         xlim(xLimits);
     end
     title(plotTitle);
```

```matlab
    xlabel(xLabel);
    ylabel(’Frequency’);
end

function PlotHistogramWithKDE(data, plotTitle, xLabel, xLimits, numBins)
    % Plot histogram
    edges = linspace(min(data), max(data), numBins + 1); % bin edges
    [counts, ~] = histcounts(data, edges);
    binCenters = (edges(1:end-1) + edges(2:end)) / 2;
    bar(binCenters, counts, ’FaceAlpha’, 0.5, ’EdgeColor’, ’none’);
    hold on;

    % Calculate KDE
    [f, xi] = ksdensity(data); % Unrestricted KDE
    valid_indices = xi >= min(data) & xi <= max(data); % Restrict KDE to data range

    % Overlay KDE
    plot(xi(valid_indices), f(valid_indices) * max(counts) / max(f), ’r-’, ’LineWidth’, 1);

    % Labels and title
    if exist(’xLimits’, ’var’) && ~isempty(xLimits)
        xlim(xLimits);
    end
    title(plotTitle);
    xlabel(xLabel);
    ylabel(’Frequency’);
    legend(’Histogram’, ’KDE’);
    hold off;
end

%% Additional functions
function mass = ComputeInvariantMass(varargin)
    totalMomentum = sum(cat(1, varargin{:}), 1);
    massSquared = totalMomentum(1)^2 - sum(totalMomentum(2:4).^2);
    mass = sqrt(max(massSquared, 0));
end

% Function to calculate pseudorapidity
function eta = CalculatePseudorapidity(p)
    pMag = sqrt(sum(p(2:4).^2));
    theta = acos(p(4) / pMag);
    eta = -log(tan(theta / 2));
end

```

```
% Function to calculate cosine of angle between two particles
function cosTheta = CalculateCosTheta(p1, p2)
    cosTheta = dot(p1(2:4), p2(2:4)) / (norm(p1(2:4)) * norm(p2(2:4)));
end

% Function to plot pseudorapidity distributions for all particles
function PlotPseudorapidity(pseudorapidity_q1, pseudorapidity_q2,
    pseudorapidity_q3, pseudorapidity_q4, scenarioName)
    histogram(pseudorapidity_q1, 100, ’BinLimits’, [-5, 5], ’FaceAlpha’,
    0.5, ’DisplayName’, ’q1’, ’EdgeColor’, ’none’);
    hold on;
    histogram(pseudorapidity_q2, 100, ’BinLimits’, [-5, 5], ’FaceAlpha’,
    0.5, ’DisplayName’, ’q2’, ’EdgeColor’, ’none’);
    histogram(pseudorapidity_q3, 100, ’BinLimits’, [-5, 5], ’FaceAlpha’,
    0.5, ’DisplayName’, ’q3’, ’EdgeColor’, ’none’);
    histogram(pseudorapidity_q4, 100, ’BinLimits’, [-5, 5], ’FaceAlpha’,
    0.5, ’DisplayName’, ’q4’, ’EdgeColor’, ’none’);
    title([’Pseudorapidity Distribution: ’, scenarioName]);
    xlabel(’Pseudorapidity (\eta)’);
    ylabel(’Frequency’);
    legend;
    hold off;
end

% Function to plot angular distribution (cosine) for particle pairs
function PlotCosThetaDistribution(cosTheta12, cosTheta34, scenarioName)
    histogram(cosTheta12, 100, ’Normalization’, ’probability’,
    ’FaceAlpha’, 0.6, ’DisplayName’, ’q1-q2’);
    hold on;
    histogram(cosTheta34, 100, ’Normalization’, ’probability’,
    ’FaceAlpha’, 0.6, ’DisplayName’, ’q3-q4’);
    title([’Cos(\theta) Distribution: ’, scenarioName]);
    xlabel(’cos(\theta)’);
    ylabel(’Probability’);
    legend;
    hold off;
end
```

Scenario1.m: $e^+e^- \to \mu^+\mu^-\tau^+\tau^-$

```
function particles = Scenario1(sqrtS, mMu, mTau)
    deltaE = 5 * 1e-2;
    maxAttempts = 20000;
    particles = [];

    attempt = 0;
    while isempty(particles) && attempt < maxAttempts
        attempt = attempt + 1;

        % Generate q1
        E1 = mMu + (sqrtS/2 - mMu) * rand;
        pMag1 = sqrt(E1^2 - mMu^2);
        p1_direction = RandomDirection();
        q1 = [E1, pMag1 .* p1_direction];

        % Off-shell condition for q1: q1^2 = mMu^2
        assert(abs(q1(1)^2 - norm(q1(2:4))^2 - mMu^2) < 1e-3, ’Off-shell condition for q1 not satisfied.’);

        % Generate q2 (muon 2)
        remainingEnergy1 = sqrtS - q1(1);
        if remainingEnergy1 <= mMu
            continue;
        end

        E2 = mMu + (remainingEnergy1 - mMu) * rand;
        pMag2 = sqrt(E2^2 - mMu^2);
        p2_direction = RandomDirection();
        q2 = [E2, pMag2 .* p2_direction];

        % Off-shell condition for q2: q2^2 = mMu^2
        if abs(q2(1)^2 - norm(q2(2:4))^2 - mMu^2) > 1e-3
            continue;
        end

        % Generate q3 (tau 1)
        remainingEnergy2 = sqrtS - (q1(1) + q2(1));
        if remainingEnergy2 <= mTau
            continue;
        end

        E3 = mTau + (remainingEnergy2 - mTau) * rand;
        pMag3 = sqrt(E3^2 - mTau^2);
        p3_direction = RandomDirection();
        q3 = [E3, pMag3 .* p3_direction];

        % Off-shell condition for q3
        if abs(q3(1)^2 - norm(q3(2:4))^2 - mTau^2) > 1e-3
            continue;
        end

        % Step 6: Calculate q4 to match q3 with invariant mass mq34_target
        totalMomentum_q1q2q3 = q1 + q2 + q3;
        required_momentum_q4 = -totalMomentum_q1q2q3(2:4);

        E4 = sqrt(mTau^2 + norm(required_momentum_q4)^2);
        if E4 < mTau
            continue;
        end

        q4 = [E4, required_momentum_q4];

         % Iterative adjustment to satisfy off-shell condition for q4
        iteration_tolerance = 1e-4;
        max_q4_adjustments = 1000;
        for adj_attempt = 1:max_q4_adjustments
            % Calculate current squared mass of q4
            q4_mass_squared = q4(1)^2 - norm(q4(2:4))^2;
            target_mass_squared = mTau^2;

            % Check if close enough
            if abs(q4_mass_squared - target_mass_squared) < iteration_tolerance
                break;
            end

            % Adjust E4 slightly to bring q4 closer to target mass
            E4_adjusted = sqrt(target_mass_squared + norm(q4(2:4))^2);
            if abs(E4_adjusted - q4(1)) < iteration_tolerance
                break;
            end
            q4(1) = E4_adjusted;
        end

        % Off-shell condition for q4: q4^2 = mTau^2
        if abs(q4(1)^2 - norm(q4(2:4))^2 - mTau^2) > 1e-3
            continue;
        end

        totalMomentum = q1 + q2 + q3 + q4;

        % Check overall energy and momentum conservation
        if norm(totalMomentum(2:4)) > deltaE || abs(totalMomentum(1) - sqrtS) > deltaE
            continue;
        end

        % Store successful particles
        particles = containers.Map({'q1', 'q2', 'q3', 'q4'}, {q1, q2, q3, q4});
    end
end

%% Additional functions
function direction = RandomDirection()
    theta = rand * pi;
    phi = rand * 2 * pi;
    direction = [sin(theta) * cos(phi), sin(theta) * sin(phi), cos(theta)];
end
```

Scenario2a.m: $e^+e^- \to (Z \to \mu^+\mu^-)\tau^+\tau^-$

```
function particles = Scenario2a(sqrtS, mMu, mTau, mZ, gammaZ)
    deltaE = 5 * 1e-2;
    maxAttempts = 20000;
    particles = [];

    attempt = 0;
    while isempty(particles) && attempt < maxAttempts
        attempt = attempt + 1;

        % Sample mq12
        mq12_target = SampleBreitWigner(mZ, gammaZ);

        % Generate q1
        E1 = mMu + (sqrtS/2 - mMu) * rand;
        pMag1 = sqrt(E1^2 - mMu^2);
        p1_direction = RandomDirection();
        q1 = [E1, pMag1 .* p1_direction];

        % Off-shell condition for q1: q1^2 = mMu^2
        assert(abs(q1(1)^2 - norm(q1(2:4))^2 - mMu^2) < 1e-3, 'Off-shell condition for q1 not satisfied.');

        % Generate q2 to match q1 with invariant mass mq12_target
        p2_direction = RandomDirection();
        p2_dot_p1 = dot(p2_direction, q1(2:4));
        E2 = (mq12_target^2 - 2*mMu^2 + 2 * q1(1) * mMu) / (2 * (q1(1) - p2_dot_p1));

        if E2 < mMu
            continue;
        end

        pMag2 = sqrt(E2^2 - mMu^2);
        q2 = [E2, pMag2 .* p2_direction];

        % Off-shell condition for q2: q2^2 = mMu^2
        if abs(q2(1)^2 - norm(q2(2:4))^2 - mMu^2) > 1e-3
            continue;
        end

        if abs(ComputeInvariantMass(q1, q2) - mq12_target) > gammaZ
            continue;
        end

        % Remaining energy after q1 and q2
        remainingEnergy = sqrtS - (q1(1) + q2(1));
        if remainingEnergy <= (2*mTau)
            continue;
        end

        % Generate q3 with a random energy between mTau and remaining energy
        E3 = mTau + (remainingEnergy - mTau) * rand;
        pMag3 = sqrt(E3^2 - mTau^2);
        p3_direction = RandomDirection();
        q3 = [E3, pMag3 .* p3_direction];

        % Off-shell condition for q3: q3^2 = mTau^2
        if abs(q3(1)^2 - norm(q3(2:4))^2 - mTau^2) > 1e-3
            continue;
        end

        % Step 6: Calculate q4 to match q3 with invariant mass mq34_target
        totalMomentum_q1q2q3 = q1 + q2 + q3;
        required_momentum_q4 = -totalMomentum_q1q2q3(2:4);

        E4 = sqrt(mTau^2 + norm(required_momentum_q4)^2);
        if E4 < mTau
            continue;
        end

        q4 = [E4, required_momentum_q4];

        % Iterative adjustment to satisfy off-shell condition for q4
        iteration_tolerance = 1e-4;
        max_q4_adjustments = 100;
        for adj_attempt = 1:max_q4_adjustments
            % Calculate current squared mass of q4
            q4_mass_squared = q4(1)^2 - norm(q4(2:4))^2;
            target_mass_squared = mTau^2;

            % Check if close enough
            if abs(q4_mass_squared - target_mass_squared) < iteration_tolerance
                break;
            end

            % Adjust E4 slightly to bring q4 closer to target mass
            E4_adjusted = sqrt(target_mass_squared + norm(q4(2:4))^2);
            if abs(E4_adjusted - q4(1)) < iteration_tolerance
                break;
            end
            q4(1) = E4_adjusted;
        end

        % Off-shell condition for q4: q4^2 = mTau^2
        if abs(q4(1)^2 - norm(q4(2:4))^2 - mTau^2) > 1e-3
            continue;
        end

        totalMomentum = q1 + q2 + q3 + q4;

        % Check overall energy and momentum conservation
        if norm(totalMomentum(2:4)) > deltaE || abs(totalMomentum(1) - sqrtS) > deltaE
            continue;
        end

        % Store successful particles
        particles = containers.Map({’q1’, ’q2’, ’q3’, ’q4’}, {q1, q2, q3, q4});
    end
end

%% Additional functions
function direction = RandomDirection()
```

```
    theta = rand * pi;
    phi = rand * 2 * pi;
    direction = [sin(theta) * cos(phi), sin(theta) * sin(phi), cos(theta)];
end

function m_sample = SampleBreitWigner(m0, gamma)
    m_sample = m0 + gamma * tan(pi * (rand - 0.5));
end

function mass = ComputeInvariantMass(varargin)
    totalMomentum = sum(cat(1, varargin{:}), 1);
    massSquared = totalMomentum(1)^2 - sum(totalMomentum(2:4).^2);
    mass = sqrt(max(massSquared, 0));
end
```

Scenario2b.m: $e^+e^- \to (Z \to \tau^+\tau^-)\mu^+\mu^-$

```
function particles = Scenario2b(sqrtS, mMu, mTau, mZ, gammaZ)
    deltaE = 5 * 1e-2;
    maxAttempts = 20000;
    particles = [];

    attempt = 0;
    while isempty(particles) && attempt < maxAttempts
        attempt = attempt + 1;

        % Sample mq34
        mq34_target = SampleBreitWigner(mZ, gammaZ);

        % Generate q3
        E3 = mTau + (sqrtS/2 - mTau) * rand;
        pMag3 = sqrt(E3^2 - mTau^2);
        p3_direction = RandomDirection();
        q3 = [E3, pMag3 .* p3_direction];

        % Off-shell condition for q3: q3^2 = mTau^2
        assert(abs(q3(1)^2 - norm(q3(2:4))^2 - mTau^2) < 1e-3, ’Off-shell condition for q3 not satisfied.’);

        % Generate q4 to match q3 with invariant mass mq34_target
        p4_direction = RandomDirection();
        p4_dot_p3 = dot(p4_direction, q3(2:4));
        E4 = (mq34_target^2 - 2*mTau^2 + 2 * q3(1) * mTau) / (2 * (q3(1) - p4_dot_p3));

        if E4 < mTau
            continue;
        end

        pMag4 = sqrt(E4^2 - mTau^2);
        q4 = [E4, pMag4 .* p4_direction];

        % Off-shell condition for q4: q4^2 = mTau^2
        if abs(q4(1)^2 - norm(q4(2:4))^2 - mTau^2) > 1e-3
            continue;
        end

        if abs(ComputeInvariantMass(q3, q4) - mq34_target) > gammaZ
            continue;
        end

        % Remaining energy after q3 and q4
        remainingEnergy = sqrtS - (q3(1) + q4(1));
        if remainingEnergy <= (2*mMu)
            continue;
        end

        % Generate q1 with a random energy between mMu and remaining energy
        E1 = mMu + (remainingEnergy - mMu) * rand;
        pMag1 = sqrt(E1^2 - mMu^2);
        p1_direction = RandomDirection();
        q1 = [E1, pMag1 .* p1_direction];

        % Off-shell condition for q1: q1^2 = mMu^2
        if abs(q1(1)^2 - norm(q1(2:4))^2 - mMu^2) > 1e-3
            continue;
        end

        % Calculate q2 to match q1 with invariant mass mq12_target
        totalMomentum_q3q4q1 = q3 + q4 + q1;
        required_momentum_q2 = -totalMomentum_q3q4q1(2:4);

        E2 = sqrt(mMu^2 + norm(required_momentum_q2)^2);
        if E2 < mMu
            continue;
        end

        q2 = [E2, required_momentum_q2];

        % Iterative adjustment to satisfy off-shell condition for q2
        iteration_tolerance = 1e-4;
        max_q2_adjustments = 100;
        for adj_attempt = 1:max_q2_adjustments
            % Calculate current squared mass of q2
            q2_mass_squared = q2(1)^2 - norm(q2(2:4))^2;
            target_mass_squared = mMu^2;

            % Check if close enough
            if abs(q2_mass_squared - target_mass_squared) < iteration_tolerance
                break;
            end

            % Adjust E2 slightly to bring q2 closer to target mass
            E2_adjusted = sqrt(target_mass_squared + norm(q2(2:4))^2);
            if abs(E2_adjusted - q2(1)) < iteration_tolerance
                break;
            end
            q2(1) = E2_adjusted;
        end

        % Off-shell condition for q2: q2^2 = mMu^2
        if abs(q2(1)^2 - norm(q2(2:4))^2 - mMu^2) > 1e-3
            continue;
        end

        totalMomentum = q1 + q2 + q3 + q4;

        % Check overall energy and momentum conservation
        if norm(totalMomentum(2:4)) > deltaE || abs(totalMomentum(1) - sqrtS) > deltaE
            continue;
        end

        % Store successful particles
        particles = containers.Map({’q1’, ’q2’, ’q3’, ’q4’}, {q1, q2, q3, q4});
    end
end

%% Additional functions
function direction = RandomDirection()
    theta = rand * pi;
    phi = rand * 2 * pi;
    direction = [sin(theta) * cos(phi), sin(theta) * sin(phi), cos(theta)];
end

function m_sample = SampleBreitWigner(m0, gamma)
    m_sample = m0 + gamma * tan(pi * (rand - 0.5));
end

function mass = ComputeInvariantMass(varargin)
    totalMomentum = sum(cat(1, varargin{:}), 1);
    massSquared = totalMomentum(1)^2 - sum(totalMomentum(2:4).^2);
    mass = sqrt(max(massSquared, 0));
end
```

Scenario4.m: $e^+e^- \to (Z \to \mu^+\mu^-)(Z \to \tau^+\tau^-)$

```
function particles = Scenario4(sqrtS, mMu, mTau, mZ, gammaZ)
    deltaE = 5 * 1e-2;
    maxAttempts = 150000;
    particles = [];

    attempt = 0;
    while isempty(particles) && attempt < maxAttempts
        attempt = attempt + 1;

        % Sample mq12 and mq34 from a Breit-Wigner distribution
        mq12_target = SampleBreitWigner(mZ, gammaZ);
        mq34_target = SampleBreitWigner(mZ, gammaZ);

        % Generate q1
        E1 = mMu + (sqrtS/2 - mMu) * rand;
        pMag1 = sqrt(E1^2 - mMu^2);
        p1_direction = RandomDirection();
        q1 = [E1, pMag1 .* p1_direction];

        % Off-shell condition for q1: q1^2 = mMu^2
        assert(abs(q1(1)^2 - norm(q1(2:4))^2 - mMu^2) < 1e-3, ’Off-shell condition for q1 not satisfied.’);

        % Generate q2 to match q1 with invariant mass mq12_target
        p2_direction = RandomDirection();
        p2_dot_p1 = dot(p2_direction, q1(2:4));
        E2 = (mq12_target^2 - 2*mMu^2 + 2 * q1(1) * mMu) / (2 * (q1(1) - p2_dot_p1));

        if E2 < mMu
            continue;
        end

        pMag2 = sqrt(E2^2 - mMu^2);
        q2 = [E2, pMag2 .* p2_direction];

        % Off-shell condition for q2: q2^2 = mMu^2
        if abs(q2(1)^2 - norm(q2(2:4))^2 - mMu^2) > 1e-3
            continue;
        end

        if abs(ComputeInvariantMass(q1, q2) - mq12_target) > gammaZ
            continue;
        end

        % Step 4: Calculate remaining energy after q1 and q2
        remainingEnergy = sqrtS - (q1(1) + q2(1));
        if remainingEnergy <= (2*mTau)
            continue;
        end

        % Step 5: Generate q3 with a random energy between mTau and remaining energy
        E3 = mTau + ((remainingEnergy - mq34_target) - mTau) * rand;
        pMag3 = sqrt(E3^2 - mTau^2);
        p3_direction = RandomDirection();
        q3 = [E3, pMag3 .* p3_direction];


        % Off-shell condition for q3: q3^2 = mTau^2
        if abs(q3(1)^2 - norm(q3(2:4))^2 - mTau^2) > 1e-3
            continue;
        end

        % Step 6: Calculate q4 to match q3 with invariant mass mq34_target
        totalMomentum_q1q2q3 = q1 + q2 + q3;
        required_momentum_q4 = -totalMomentum_q1q2q3(2:4);

        E4 = sqrt(mq34_target^2 + norm(required_momentum_q4)^2 + mTau^2);
        if E4 < mTau
            continue;
        end

        q4 = [E4, required_momentum_q4];

        % Iterative adjustment to satisfy off-shell condition for q4
        iteration_tolerance = 1e-4;
        max_q4_adjustments = 100;
        for adj_attempt = 1:max_q4_adjustments
            % Calculate current squared mass of q4
            q4_mass_squared = q4(1)^2 - norm(q4(2:4))^2;
            target_mass_squared = mTau^2;

            % Check if close enough
            if abs(q4_mass_squared - target_mass_squared) < iteration_tolerance
                break;
            end

            % Adjust E4 slightly to bring q4 closer to target mass
            E4_adjusted = sqrt(target_mass_squared + norm(q4(2:4))^2);
            if abs(E4_adjusted - q4(1)) < iteration_tolerance
                break;
            end
            q4(1) = E4_adjusted;
        end

        % Off-shell condition for q4: q4^2 = mTau^2
        if abs(q4(1)^2 - norm(q4(2:4))^2 - mTau^2) > 1e-3
            continue;
        end

        totalMomentum = q1 + q2 + q3 + q4;
        if abs(ComputeInvariantMass(q3, q4) - mq34_target) > gammaZ
            continue;
        end

        % Check overall energy and momentum conservation
        if norm(totalMomentum(2:4)) > deltaE || abs(totalMomentum(1) - sqrtS) > deltaE
            continue;
        end

        % Store successful particles
        particles = containers.Map({'q1', 'q2', 'q3', 'q4'}, {q1, q2, q3, q4});
    end
```

```
end

%% Additional functions
function direction = RandomDirection()
    theta = rand * pi;
    phi = rand * 2 * pi;
    direction = [sin(theta) * cos(phi), sin(theta) * sin(phi),
   cos(theta)];
end

function m_sample = SampleBreitWigner(m0, gamma)
    m_sample = m0 + gamma * tan(pi * (rand - 0.5));
end

function mass = ComputeInvariantMass(varargin)
    totalMomentum = sum(cat(1, varargin{:}), 1);
    massSquared = totalMomentum(1)^2 - sum(totalMomentum(2:4).^2);
    mass = sqrt(max(massSquared, 0));
end
```

List of Figures

List of Tables

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material that has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present thesis.

Date

Signature