**Theoretical Questions**

1. What are the HepMC and LHE formats used for in particle physics simulations? Briefly describe their structure and key differences.  
   *(For this answer, I heavily relied on the references provided below. I tried to answer them, but found it difficult to exactly understand the differences between the two. I tried to research online but couldn’t find clear answers.)*
   1. The Les Houches Events (LHE) file format is an agreement between Monte Carlo event generators and theorists to define Matrix Element level event listings in a common language. It is primarily used to store output from parton level event generators for further use by general purpose ones. It allows for easier interchangeability and separation of different stages of event generation
   2. Monte Carlo event generators are used to simulate interactions of high energy particles. MCEG event records store the information on the simulated particles and their relationships for analysis. HepMC is a framework for this MCEG event record encoding and manipulation.
   3. Differences in PyLHE and HepMC - HepMC is very restrictive. Only predefined information can be stored. From the diagram below, the LHE seems like an input file for a parton shower and the HepMC looks like an output file. LHE files are stored as XML. HepMC is an object-oriented event record written in C++.
   4. The LHE file structure starts with comments, <init> tags with a line describing the beams and after that the events where ach event is enclosed in a <event> tag.
   5. Not exactly sure of the HepMC structure, but the ROOT HepMC follows a Tree structure.A black text on a white background

      AI-generated content may be incorrect.
   6. References
      1. <https://indico.global/event/8315/contributions/78779/attachments/36780/68756/CST2010-MC.pdf>
      2. <https://www.sciencedirect.com/science/article/abs/pii/S0010465520301181?via%3Dihub>
      3. <https://indico.cern.ch/event/277733/contributions/1623403/attachments/506390/699108/CERN13a.pdf>
      4. <https://theory.gsi.de/~smash/userguide/current/doxypage_output_hepmc.html>
2. What are the benefits of visualizing Monte Carlo events in particle physics? Name two potential use cases.
   1. Monte Carlo events are random physical processes that are simulated to understand what’s happening. Multiple iterations of the simulations are used to come up on a conclusion. For this, we need to know the PDFunctions defined for an event. For instance, to know what happens when 2 particles are collided, we generate multiple simulations of the event and process the data so that we can come up with exact information on what happens when the event occurs. Simulating the events would give information on the highly probable events and this would help us come to conclusions on if a theory is valid or not and so on.
   2. Use cases:
      1. Physics analysis - New predicted physics
      2. Simulation of particle interactions with detector’s material
      3. Detector design and optimization
   3. References
      1. <https://indico.global/event/8315/contributions/78779/attachments/36780/68756/CST2010-MC.pdf>
      2. <https://physics.stackexchange.com/questions/723916/how-do-monte-carlo-event-generators-work-in-general>
3. What is the purpose of jet clustering in particle physics? Name one common jet clustering algorithm.
   1. The main purpose of the jet clustering algorithm is to understand what happened after a collision. Because there is a stream of particles that come out at different directions in the form of Jets. These algorithms reconstruct the path of these particles to understand their origins and get more insights into the collision. They provide a link between the observed stable particles and the underlying physics.
   2. Common Jet clustering algorithms
      1. Cambridge/Aachen and kt
   3. References
      1. <https://iopscience.iop.org/article/10.1088/1742-6596/645/1/012008/pdf>
4. Explain how JSON or a similar structured format could be useful when transmitting event data from a server to a web visualization tool.
   1. JSON is useful because of how data is structured as key-value pairs. It is in a format that is both human and machine readable which allows it to be used across hosts and programming languages. And since it is lightweight, it efficiently facilitates communication between a client and a server. For example, transmitting particle data, like the coordinates and their energy is very simple. All we must do is get a unique key to identify the particles, which would be a key-value pair in on itself and then a list of values. Since its just plain text, transmitting over the network makes it simple, fast and efficient.

**Coding Questions**

1. Parsing an MC event file – ***code file name = question\_2a.py***
   1. We need to print basic information about the particles for each event in the given LHE file. Also, we need to plot the transverse momentum distribution for all the top quarks in the events. I have printed the number of total and final particles, the energy and the momentum sums for all the particles in the event.
   2. The formula for calculating the momentum distribution is Once we have the transverse momentum, we plot it as a histogram. The x-axis represents the transverse momentum in GeV and the y-axis represents the frequency of the quarks. This histogram will allow us to visualize how many top quarks with a specific range of momentum have resulted from the collision. This would give information like how often particles (Top quarks in this case) are produced with momentum between a range. If it was a standard collision, the histogram would represent nothing special, but if there is something new happening in the collision, it can give insights on what’s happening during the collision because we would see an anomaly. From the distribution, we can see that there is a peak between the 100 and 200 GeV range, this can say something about the collision. Histogram is one way. A normal distribution is overlayed along the histogram showing that the momenta are mostly situated around the mean. These plots could show us the deviations from expected behavior.
   3. I also tried to plot a box plot of the transverse momenta, but most of them were outliers and I don’t know what to make of it. Also tried to plot a violin plot (sounds interesting), the plot shows that most of the top quarks have momenta ranging from 50 – 200 GeV.
   4. References
      1. <https://pmc.ncbi.nlm.nih.gov/articles/PMC10099715/>
2. Basic event-graph rendering - ***code file name = question\_2b.py***
   1. I chose Plotly to render the particle interaction graph. NetworkX, they have mentioned it on their website itself that their main goal is for graph analysis rather than graph visualization, more like graph algorithms and stuff. PyDot is tightly coupled with the Graphviz world, and it is great for visualizing graphs, but it’s mostly static and does not allow graphs to be interactive. Matplotlib is mainly used along pandas data frames, I used it before for statistical analysis and even these graphs are statistic. Plotly seemed best, it is robust, supports interactive graphs and integrates easily with the JavaScript library that makes it easier for rendering it on the web. For the task, I used Jupyter notebooks. It renders graphs in the cell output itself, which makes it easier to run and debug. For jupyter notebooks, they are usually run on the web, but the IDE (PyCharm) I used, makes it easier to run them locally through the IDE without going through the hassle of setting it up on the browser.
   2. I chose the parameters in the Plotly visualization to clearly distinguish between different particles and interactions in the graph. Color was used as the main differentiator: red for the electron and the positron, purple for the Z boson to represent the intermediate force carrier, and blue for the outgoing photons. The line styles were used to convey different types of particles or interactions—solid lines for the incoming particles, dotted for the force particle (Z boson), and dashed for the photons. Symbols are also used as the differentiating elements: circles for the electron and positron, squares for the Z boson, and X’s for the photons. Similarly, we could use differentiating elements and colors to identify the different particles and interactions. Well, adding in more information might make the graph messy, but we could have a checkbox on what properties to include in the graph and what not to according to the user. So that the user has an option to choose what properties to display. The code snippet also exports the plot to a HTML file, which is interactive and better visualizes the plot.
3. Basic Web Rendering - ***code file name = app.py and index.html***
   1. The particle collider probably generates an exponential number of events, and most of them are results that have already been seen before. So, rather than trying to visualize all the events, we could differentiate the common occurrences from the non-common occurrences and visualize them.
   2. To efficiently visualize the next event, one idea would be the brute force approach to iterate through the events and render it, which would probably be an inefficient solution. Another idea is to use a generator to get the events, and we cannot possibly use a Python list to store all of them. Depending on the memory, when one event is viewed, we could store the information of the surrounding 10 / 100 / 1000 events. For example, if I am viewing event 24, I could call the events from 14–34 and store them in memory. This way, we would pass the next\_event\_id through the JSON structure and that would help us identify the next event that occurred. Also, rather than rendering all the 10 or 100 events at once, we could display minimal information from that event to help a physicist decide if that event is worth visualizing. We could do this step after we pick out the uncommon interactions. Navigating between different events is important to visualize what’s happening when the particles collide. I have used a pretty simple visualization of an event using a 3D graph and plotted the particles that are involved.
   3. The plotting happens from the pX, pY, and pZ variables that are extracted from the file. Although, the pX, pY and pZ are the momentum values, I have used them as coordinates, even though they wouldn’t exactly arrive at that point after the collision, they would travel along that path. And this would give an idea on where the particles are moving along. We can see that the particles are named using their respective symbols. I drew the edges using the mother 1 and mother 2 points from the LHE file—I tried to use them as the origin points for a particle since the PyLHE library extracts this information from the file. I also tried to plot the arrows that would show the directional information for these particles. But I couldn’t plot them because it required a bit more understanding of the Plotly library fast. The arrows would convey a lot more information. Also, we could use the energy to display the size of the particle, conveying the energy information. The lifetime value from the LHE file could help in visualizing how far the particle has traveled once the collision happens. This could be represented as the length of the edge—if the edge length is long, it could mean that the particle has traveled a long distance before decaying, and so on.
   4. Coming to the case of a user supplied file, if it’s represented in the LHE format, it wouldn’t be an issue. But if it’s not in the LHE format, we might have to do additional processing to include the required information and modify the code to include a wide variety of formats to get the information we need. If the information we need is altogether not existing in the file provided, that would be a difficult task and might involve incorporating additional logic to render a graph from the information available at hand. But this should not be that big of an issue.