Practical 5: Understanding the MadGraph Cuts

Last practical, we finished generating data in our python script to plot a histogram of our angular distribution observable. Firstly, we will make a small comment about this. In our method `pass_cuts()` we have both a gluon cut and a theta cut. Why is it that we need both? Recall that for an outgoing quark of momentum p_1 and outgoing gluon of momentum k, the amplitude we desire has a factor of

$$\frac{1}{p_1 \cdot k} = \frac{1}{E_g E_q (1 - \cos\theta_{gq})} \tag{1}$$

From the left, one may naively assuming that imposing a gluon momentum cut would ensure that the denominator is non-zero. However, of course, this does not prevent the colinear divergence. Likewise, fixing the angle does not stop the soft divergence. Thus, it really is necessary to include both cuts.

We would like to compare our previous results to MadGraph. Let's open it up and compute the process

```
1       generate e+ e- > d d~ g / z
2   display diagrams
```

This is now in memory. We will output

```
1       output CHEP_xcheck
```

to save the generated data for reference. We now `launch CHEP_xcheck` (which is then all that one needs if the `CHEP_xcheck` process has been output already during a prior run of MadGraph). We can access the parameter card by typing `1`. If we would to compare to our Python code, we should double check that each of these parameters are the same. These can be found in `parameters.py`. They should in theory already match what is in MadGraph, but it's good to double check.

We would also like to double check some properties for our run card. Let us type `2` in MadGraph to open it. Firstly, we would like to fix the renormalisation scale

```
1    True = fixed_ren_scale  ! if .true. use fixed ren scale
2        True = fixed_fac_scale  ! if .true. use fixed fac scale
```

We would also like to modify the standard cuts. We would firstly like to change the transverse momentum, `ptj`. Note that this transverse momentum refers to the beam axis momentum, rather than the component of the gluon transverse to the quarks. Fixing the minimum value guarantees that we won't have a soft divergences (since it bounds the momentum from below), however this won't fix our collinear divergence. Next we consider the rapidity of the jets, `etaj`. Note that in relativistic physics, angles are not reference

frame invariant. Thus, instead of performing cuts based on angles, MadGraph uses frame-independent pseudorapidity differences (which are equal to rapidity differences in the case of massless particles), which are a function of the angles, but ideal quantities to observe in a collider experiment since one cannot reconstruct the longitudinal boost connecting the center-of-mass frame of the collision to the laboratory frame. The pseudorapidity given by

$$\eta = -\log\left(\tan\left(\frac{\theta}{2}\right)\right), \tag{2}$$

where θ is the angle between the particle spatial momentum and the positive direction of the beam axis. However, the `ptj` cut already implicitly places a similar angular cut as the pseudorapidity cut would, so it would be of no use for the purpose of further screening infrared singularities. The rapidity is measured relative to the beam axis, and thus it is only enforcing that our jets are angled away from the beam axis.

What is more interesting is `drjj`, the separation between two jets, or the pseudorapidity distance. This is what forces us away from the collinear regime. It is given by

$$\Delta R_{j_1 j_2} = \sqrt{\Delta\eta^2_{j_1 j_2} + \Delta\phi^2_{j_1 j_2}}, \tag{3}$$

where $\eta_{j_1 j_2}$ is the pseudorapidity difference between the two jets, and is the azimuthal angle difference between the two jets. This is preferred to just using the angle because it's much easier than computing $\cos\theta$ experimentally. A value of 0.4 (the default) corresponds to an angle of separation of approximately 10°. If we remove this cut and run the experiment in MadGraph, then can see that we get a singular cross section. Indeed, the process takes a long time to run and each iteration is not within the uncertainty bounds of the previous iteration. This suggests a problem with convergence. Of course, keeping the cut ensures that the computation runs smoothly as expected.

Given that MadGraph uses `drjj` for its cuts, we will modify our code in Python to also use this for our cuts. This is not so difficult to do, as we already have the appropriate operations for Lorentz vectors implemented. Your task is to modify the code to compute `drjj` for our generated phase space points, and determine what the appropriate value of the `drjj` cut that corresponds to our current cuts in terms of θ. This will allow us to cross check both our data generation and analysis in the next practical.