Practical 4: Computing the R Ratio, II - Angular Distribution

For this practical, your task will be to implement the remaining code to reproduce some results for the $e^+e^- \to q\bar{q}g$. Particularly, we would like to reproduce the real emission plot - the distribution of events with respect to angle. Recall that at leading order, with no gluon emission, we expect the relative angle between quarks to be π: they should scatter in opposite directions. As we introduce gluons if we allow gluons in our final state, the angle will be slightly reduced as the gluon will carry some of the momentum. As the gluon becomes soft, the angle will restore to π. We would like to see the exact distribution of angles after implementing a soft cut. Create a new file for our experiment, `rratio_differential.py`. We rename the function to `rratio_differential`. In `run.py`, add this experiment to the cases, as well as the appropriate code to the parser. We will also need to import the file so that `run.py` knows where to find our code,

```
1       from CHEP.experiments.rratio_differential import rratio_differenital
```

We would like to plot a histogram. How do we gather the data we need for plotting? If we look inside our integrand function, we can see the where we generate our data and add to our evaluations. This is where we can start gathering angular data. The simplest way is to add a new argument to our integrand function, which allows it to take an array as input. Note that one should take care when doing this: when we pass an object into a method, depending on the coding language, it can either be a reference to the object's location in memory or copy? If it were a copy, the object will only be modified in the scope of the method, and our original object will be unmodified. This isn't good if we're trying to store data. We want reference for the histogram. This is the default in python - exactly what we want. However, if we want to parallelise, this would make things more difficult as the code would try and modify our array concurrently - though we won't worry about this for now. Let's create an array for our data:

```
1           costheta_histo=[0.0 for _ in range(200)]
```

This gives 200 bins. We could also add an option for number of bins in our experiment run parameters, but for now we will keep things simple. We then edit our integrand constructor to take this vector as an item by adding a parameter `histo`. Inside the integrand method, we can assign our generated phase space point to a set of variables to save the momenta

```
1       p_ep, p_em,p_d,p_db,p_g=ps_point
```

We can then compute the angle between our quarks

```
costhetaqq=p_d.space().dot(p_db.space())/(abs(p_d.space())*abs(p_db.space()))
```

We would then like to assign this to a bin. To do so, we should round the angle down to an appropriate number based on the number of bins. Noting that `int` casting in python is the same as taking the `floor`, then

```
bin_id=int((1+costhetaqq)/2*len(histo))
```

Note that we also need to account for the weight, since we don't have unweighted events. We can save these weights with

```
            histo[bin_id][0]+=wgt
```

A differential distribution is defined for a given observable function $J(\phi)$ of the final-state kinematic configuration ϕ, which in case reads:

$$J(\phi) \equiv \cos\theta_{q\bar{q}}(\phi) := \frac{\vec{p}_q \cdot \vec{p}_{\bar{q}}}{|\vec{p}_q||\vec{p}_{\bar{q}}|} \tag{1}$$

The formal definition of the differential cross-section for this observable function $J(\phi)$ is:

$$\frac{d\sigma}{dJ} := \int d\phi |\mathcal{M}(\phi)|^2 \delta\left(J(\phi) - J\right) \tag{2}$$

In order to solve this integral analytically, one would have to be able to express the observable J in terms of the phase-space parameterization variables supporting ϕ, which is not always possible, especially for complicated observables. Not to mention that one may be interested in monitoring a vast ensemble of different observable at the same time for a given simulation. It is therefore convenient to instead consider a piece-wise constant approximation of the truly continuous differential distribution $\frac{d\sigma}{dJ}$, by constructing histograms instead, with a certain number of bins, each identified with a boundary $b_{i,\min}$ and $b_{i,\max}$. The value of the piece-wise constant function within such an interval, i.e. the "height" h_i of this bin of the histogram, is then:

$$h_i := \int_{b_{i,\min}}^{b_{i,\max}} dJ \left(\frac{d\sigma}{dJ}\right) = \int d\phi |\mathcal{M}(\phi)|^2 \Theta\left[J(\phi) - b_{i,\min}\right] \Theta\left[b_{i,\max} - J(\phi)\right] \tag{3}$$

Notice that it is clear that if the total range of the histogram covers the complete final-state phase-space, then the following propery holds:

$$\sum_i h_i = \sigma_{\text{tot}} \tag{4}$$

This expression (3) is well-suited to be evaluated numerically independently of any choice for the phase space-parameterization. It can essentially be understood as the estimator for the cross-section within a fiducial volume identified by the bin of interest, and during the Monte-Carlo integration, one can simply test whether or not a given sample point belongs to the interior of the bin range.

However, as for any estimator of the central value of an integral, one must normalize the sum of all sample weights by the number of sample points entering that bin, which we must therefore keep track of as follows:

```
        costheta_histo=[(0.0, 0) for _ in range(200)]
```

and change

```
histo[bin_id][0]+=wgt
histo[bin_id][1]+=1
```

We can use this information to appropriately normalise each bin

```
        normalise_histo=[(tot_wgt/n_events) if n_events>0
        else 0 for (tot_wgt,n_events) in costheta_histo ]
```

which we can then plot

```
    bins=np.linspace(-1,1,len(normalise_histo))
    plt.bar(bins,normalise_histo, width=0.01)
    plt.show()
```

As mentioned at the beginning, the resulting distribution will depend on our gluon cut. By setting this to a relatively high energy to remove the soft gluons, for example `GLUON_ENERGY_CUT = 300.0` we can obtain a plot for the angular distribution.

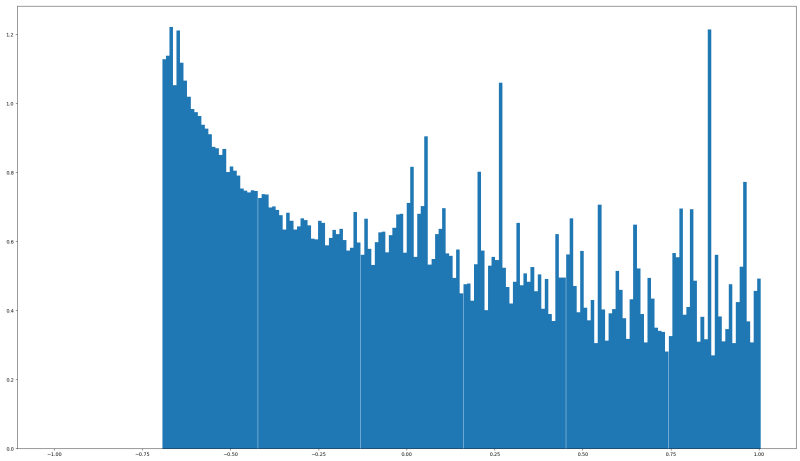

Figure 1: Cross-section angular distribution for the 300MeV soft gluon cut.

Observe that the distribution close to $\cos(\theta) = -1$ is quite well resolved, while at $\cos(\theta) = 1$ the histogram is much more jagged and noisy. This is an artefact of our sampling: we generate far more points closer to $\theta = \pi$ and thus have better resolution.