Feynman diagrams

Alexander Huss

July 23, 2023

Contents

1 Introduction

Let's have some fun with Feynman diagrams; we'll use the tool `qgraf` for their generation. If you want to reproduce the results in this document, you can get the source from here: qgraf (note that you'll need a `Fortran` compiler for this).

2 Drell–Yan

Let's start simple and see how we can generate the diagrams for the Drell–Yan process that we already know. The example files are located in `../../tools/qgraf/dy` and the program is controlled using the input file `qgraf.dat`. In this example, the Feynman rules are encoded in a simple model file `dyqcd` and the output style is set by `custom.sty`. We can run `qgraf` in that directory to generate the diagrams

```
#loops    v-degrees          #diagrams

   0
           -   4^1     ....     0
          3^2   -      ....     2


     total =  2 connected diagrams
```

This will generate the output inside the file `qlist.out`:

```
*--#[ d1:
*
 a(1):= (-1)*
  in[-1](quark(p1))*
  in[-3](antiquark(p2))*
 out[-2](l_plus(q1))*
 out[-4](l_minus(q2))*
 prop[1,2](photon(-p1-p2))*
 vrtx[-3,-1,1](antiquark(p2),quark(p1),photon(-p1-p2))*
 vrtx[-4,-2,2](l_plus(-q2),l_minus(-q1),photon(p1+p2));
*
*--#] d1:
*--#[ d2:
*
 a(2):= (-1)*
  in[-1](quark(p1))*
  in[-3](antiquark(p2))*
 out[-2](l_plus(q1))*
 out[-4](l_minus(q2))*
 prop[1,2](Z(-p1-p2))*
 vrtx[-3,-1,1](antiquark(p2),quark(p1),Z(-p1-p2))*
 vrtx[-4,-2,2](l_plus(-q2),l_minus(-q1),Z(p1+p2));
*
*--#] d2:
*
* end
*
```

which are precisely the two s-channel diagrams we already know for this process.

3 Gluon scattering

We will now consider the case of $2 \to n$ gluon scattering, $gg \to g \ldots g$. To make this more generic, we have a template file from which we generate an input file for `qgraf` to automatically excract the number of diagrams `qgraf` has generated for us.

```
#> create an input file from the tempalte
out_string=""
sep=""
for i in $(seq 1 ${ngluons}); do
    out_string="${out_string}${sep} gluon"
    sep=","
done
sed -e "s/&out&/${out_string}/g" -e "s/&loops&/${nloop}/g" qgraf.template > qgraf.dat
#> run qgraf and look for the line
#>   `total =  <N> connected diagrams`
#> to extract the number of diagrams qgraf has generated
```

```
if [[ -f qlist.out ]]; then rm qlist.out; fi
qgraf | awk '$1~/^total$/&&$(NF)~/^diagrams$/{print $3}'
```

We can then fill this table for different gluon multiplicities at tree and 1-loop level:

n	# diagrams (tree)	# diagrams (1-loop)
2	4	223
3	25	2105
4	220	25120
5	2485	362510

We can appreciate the rapid increase in the number of diagrams as more legs or loops are added to the diagrams. Note that this only consitites a very naive estimate for the scaling of the complexity as each individual diagram further becomes more complicated to evalute, especially with each additional loop that involves and additional integration that must be performed.

For the tree diagrams, we can write a very simple recursive formula for the number of diagrams:

$$N^{\text{diags}}_{2\to n} = \mathcal{N}_{n+2}(g,t)\big|_{\substack{g=1\\t=1}} \tag{1}$$

where $\mathcal{N}_{n+2}(g,t)$ is a polynomial that "counts" the number of gluons and three-gluon vertices for each diagram and can be defined recursively:

$$\mathcal{N}_{m+1}(g,t) = \Big(g^3\,t\,\frac{\partial}{\partial g} + g\,\frac{\partial}{\partial t}\Big)\mathcal{N}_m(g,t)\,, \qquad \mathcal{N}_3(g,t) = g^3\,t \tag{2}$$

The first term corresponds to an insertion of a new gluon into an existing gluon line, while the second term describes the insertion of a gluon into an existing three-gluon vertex and converting it to a four-gluon vertex.