1 Implementation

1.1 Generating Feynman Diagrams

The first step is to encode the Feynman diagrams in a way that Mathematica can manipulate them fig. 1. We represent each diagram as a list of pairs, for instance, the third diagram in fig. 1 can be represented as:

$$\{\{1,-1\},\{2,-1\},\{3,-2\},\{4,-2\},\{5,-2\},\{-1,-2\}\};$$

with the convention that the external legs are represented by positive integers and the internal vertices by negative integers, the pairs representing the connections between the vertices always ordered in decreasing order.

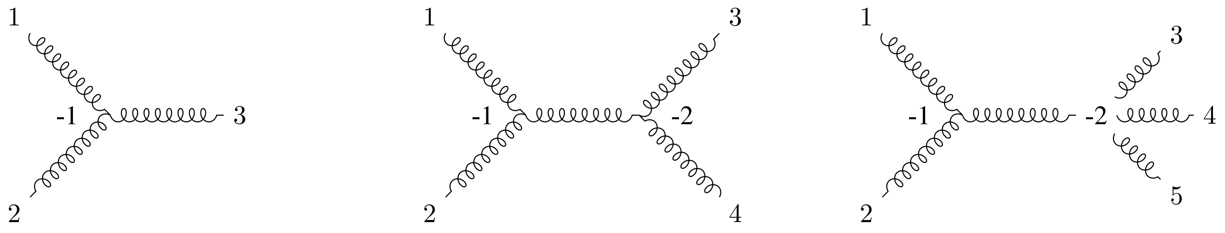

Figure 1: Example of a Feynman diagram for gluon scattering with 3, 4, and 5 external legs. On the left, the diagram with 3 external legs is the skeleton from which the other diagrams are built.

To generate the Feynman diagrams, we use a recursive function that starts with an initial diagram and progressively adds vertices. First, 3-point vertices are introduced at each internal line, and then 4-point vertices are added at each 3-point vertex. The number of 3-point vertices is determined by tallying how many times a negative number appears in the list, with each negative number appearing at most 4 times.

The next step is to assign momenta to the external and internal lines. For a generic diagram with N external lines, P internal lines , L loops and V_3, V_4 the number of 3 and 4 point vertices respectively, the number of loops is equal to the number of independent momenta:

$$L = P - V_3 - V_4 + 1 \tag{1}$$

in tree-level diagrams $L = 0$, so that we have $V_3 + V_4 = P + 1$. Imposing momentum conservation at each vertex provides $P + 1$ equations. However, there are only P unknowns, meaning the system is underdetermined. To solve this, we must include the constraint of global momentum conservation, which removes one of the equations, since momentum conservation in each vertex imply the global momentum consevation. All the previous steps are implemented in the function `GenerateDiagrams` in listing 1.

Listing 1: Generating Feynman diagrams

```
Diagrams[graph_] (* Generates diagrams from a graph, calculating internal momenta and Lorentz and color indices for the vertices and
    propagators *)
generatediagrams[n_] (* Generates diagrams with n external points and only 3 and 4−point vertices *)
generatediagrams3[n_] (* Generates diagrams with n external points and only 3−point vertices *)
```

To complete the Feynman diagrams, we need to include propagators and vertices, each with momenta, color, and Lorentz indices fig. 2. We can achieve this by using Mathematica's list manipulation and pattern matching capabilities, constructing vertices and propagators as functions that take momentum, Lorentz indices, and color indices as arguments.

1.2 Feynman Rules

The QCD Feynman rules are implemented in Mathematica as follows:

Listing 2: Feynman rules for QCD

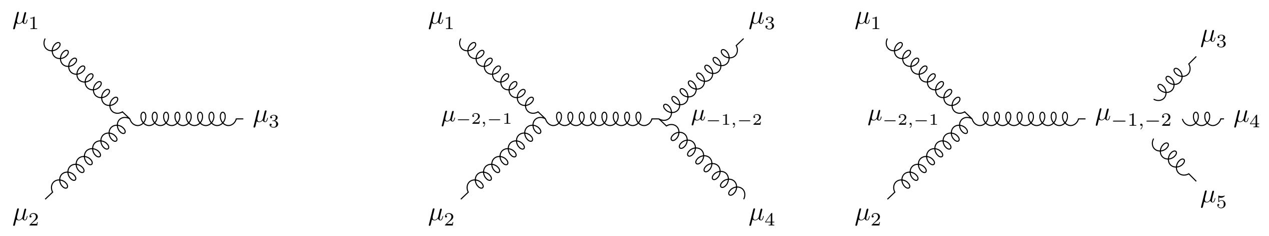

Figure 2: Each external leg is assigned a Lorentz index μ_i and each vertex j a Lorentz indices $\mu_{i,j}$. The same logic is applied for the color indices.

.

```
gluon = prop[p_, {mu1_, mu2_}, {c1_,c2_}] :> -I myDelta[c1, c2] gprop[p, mu1, mu2];
propagator[p_, mu1_, mu2_] := SP[{mu1}, {mu2}]/SP[p, p]
V3QCD = V[i_][{p1_, p2_, p3_}, {mu1_, mu2_, mu3_}, {c1_, c2_, c3_}] :> -g f[c1, c2, c3] V3Lorentz[{p1, p2, p3}, {mu1, mu2,
      mu3}]
V3p[{p1_, p2_, p3_}, {mu1_, mu2_, mu3_}] := SP[p1 - p2, {mu3}] SP[{mu1}, {mu2}] + SP[p2 - p3, {mu1}] SP[{mu2}, {mu3}] +
      SP[p3 - p1, {mu2}] SP[{mu3}, {mu1}]
V4QCD = V[i_][{p1_, p2_, p3_, p4_}, {mu1_, mu2_, mu3_, mu4_}, {c1_, c2_, c3_, c4_}] :> -I g^2
        (f[c1, c2, c[x, i]]  f[c3, c4, c[x, i]]  (SP[{mu1}, {mu3}] SP[{mu2}, {mu4}] - SP[{mu1}, {mu4}] SP[{mu2}, {mu3}])
        - f[c1, c3, c[x, i]]  f[c2, c4, c[x, i]]  (SP[{mu1}, {mu4}] SP[{mu2}, {mu3}] - SP[{mu1}, {mu2}] SP[{mu3}, {mu4}])
        + f[c1, c4, c[x, i]]  f[c2, c3, c[x, i]]  (SP[{mu1}, {mu2}] SP[{mu3}, {mu4}] - SP[{mu1}, {mu3}] SP[{mu2}, {mu4}]))
SetAttributes[myDelta, Orderless]
ColorDelta = # //. {myDelta[a_, b_] myDelta[a_, b_] :> (Nc^2 - 1) ,
    myDelta[a_, a_] :> (Nc^2 - 1),
    myDelta[a_, b_]*expr_ :> (expr /. b -> a),
    c[i_, j_] :> a[i,j]} &
FeynmanRules = (# /. gluon /. V3QCD /. V4QCD /.gprop -> propagator /. V3Lorentz -> V3p // ColorDelta // antisymf // Expand)
      &
```

where Lorentz structure and color structure are factorized when possible so we can manipulate them separately. This property is especially useful for generating all the distinct products of structure constants f^{abc} for a given number of external legs. The Feynman rules are applied to the Feynman diagrams using the `Replace` function, which substitutes the propagators and vertices with their corresponding expressions.

1.3 Color Algebra and Lorentz

The color algebra is handled using the following functions:

- **antisymf**: This functions applies the antisymmetry property of the structure constants f^{abc}, and brings the color indices in decreasing alfanumeric order.
- **frule**: This functions acts on products of structure constants, it counts the number of repeated indices and substitute them with $a[i]$. For $n \geq 5$ there are more than one repeated indices and the degeneracy associated with the permutation of the dummy indices must be taken into account.
- **Color**: This function acts on fully contracted products of structure constants and it calculates the results in terms of N_c, the dimensions of the fundamental representation of the $SU(3)$ group. This function converts the products of structure constants into its fundamental representation, using the following identity:

$$f^{abc} = -2iTr\left([T^a, T^b]T^c\right) \tag{2}$$

where T^a are the generators of the $SU(3)$ group in the fundamental representation. The trace is expressed as a product of matrices with cyclic indices, and the the Fierz identity is utilized to convert the product of T^a into a product of Kronecker

Table 1: Testing Color for various cases

n	product	Result	Time (s)
4	$f^{c_3c_1a_1}f^{c_3c_2b_1}f^{c_4c_1b_1}f^{c_4c_2a_1}$	$\frac{Nc^2}{2}(N_c^2-1)$	0.0625
5	$f^{c_1b_2b_1}f^{c_3c_2a_2}f^{c_3c_2b_1}f^{c_4a_2a_1}f^{c_5c_1a_1}f^{c_5c_4b_2}$	$-\frac{Nc^3}{2}(N_c^2-1)$	0.5868
6	$f^{c_1a_3a_1}f^{c_1b_3b_1}f^{c_3c_2a_1}f^{c_3c_2b_1}f^{c_4b_3b_2}f^{c_5a_3a_2}f^{c_6c_4a_2}f^{c_6c_5b_2}$	$\frac{Nc^4}{2}(N_c^2-1)$	25.6701

deltas:

$$T^a_{i,j}T^b_{k,l} = \frac{1}{2}\left(\delta_{i,l}\delta_{j,k} - \frac{1}{N_c}\delta_{i,j}\delta_{k,l}\right) \tag{3}$$

This conversion allows us to express the color algebra in terms of Kronecker deltas, which can be manipulated in Mathematica.

However this algorithm is not highly efficient. Before all the Kronecker deltas are contracted, the number of terms grows exponentially. For n gluons, there are $2(n-2)$ fully contracted structure constants, converting to traces gives $2^{2(n-2)}$ products of $2(n-2)$ traces each, writing the traces as products of matrices we get $3*2(n-2)$ matrices, and using Fierz it becomes $2^{2(n-2)*3}$ terms, in total we have $2^{2(n-2)*3+2(n-2)} = 2^{8(n-2)}$ Kronecker deltas to contract, making it a very inefficient algorithm table 1. It is possible to reduce the number of terms by using known identities before applying the Fierz identity, for example: $Tr(T^aT^a) = Nc^2/2$.

Lorentz algebra is implemented using the `SP` function, which is a wrapper for the `Dot` function. This function is used to contract Lorentz indices and it is defined as follows:

Listing 3: Lorentz algebra

```
SetAttributes[SP, Orderless]
Contract = (# //. {
  SP[a_, {mu_}] SP[b_, {mu_}] :> SP[a, b], (*Contraction*) SP[p_, {mu_}] SP[p_, {mu_}] :> SP[p, p],(*Square*)
  SP[a_Integer *p_, {mu_}] :> a SP[p, {mu}],(*Homogeneity*) SP[a_Integer *p_, q_] :> a SP[p, q], (*Homogeneity*)
  SP[p_ + q_, {mu_}] :> SP[p, {mu}] + SP[q, {mu}],(*Linearity*) SP[a_, b_ + c_] :> (SP[a, b] + SP[a, c]),(*Distributive property*)
  SP[{mu_}, {mu_}] :> 4 (*Trace*), SP[n, n] :> 0 (*Light Cone Gauge*), SP[p[i_], p[i_]] :> 0 (*On Shell Condition*)
  }) &;
```