

LU-TP 20-35
June 2020

A PROGRAM FOR $SU(N)$ COLOR STRUCTURE DECOMPOSITION INTO MULTIPLY BASES USING WIGNER $3J$ AND $6J$ COEFFICIENTS

Andris Potrebko

Master's thesis

Department of Astronomy and Theoretical Physics,
Lund University

Master thesis supervised by Malin Sjö Dahl



LUND
UNIVERSITY

Abstract

The increased capacity of elementary particle accelerators raises the demand for the simulation data of the experiments. One of the bottlenecks in the simulations is the QCD color structure calculation, which is usually treated using non-orthogonal and overcomplete sets of bases. The computational cost could be decreased significantly if orthogonal bases, such as the multiplet bases, were used instead. However, no computation tool performing calculations using these bases is available yet. In this thesis, we present a Mathematica program as proof-of-principle demonstrating the color structure decomposition into the multiplet bases. For a given amplitude, the corresponding multiplet basis states can be created and the scalar product between the amplitude and each of the basis states can be evaluated whenever the required Wigner 6j coefficients are available. The program offers tools for visualization of the tensor expressions in the birdtrack notation as well as a syntax similar to how the tensor expressions would be defined on paper. The available functions and replacement rules allow performing operations on $SU(N_c)$ tensor expressions including index contraction, tensor conjugation, and scalar product of tensors.

Popular science abstract

It might be stunning to realize that researching the smallest constituents of the world we see requires building the largest constructions people have created. On the border of Switzerland and France one can find the greatest of the examples, the Large Hadron Collider, a more than eight kilometers in diameter large ring filled with a vacuum where scientists let bunches of around 100 million protons collide almost 40 million times per second. Despite all the great discoveries (you might have heard of the Higgs boson found in 2012, for example), particle physicists often claim that further explorations of the fundamentals of the Universe require building even larger colliders and increasing the number of collisions even more. However, something is often left out in this demand for faster, bigger, and stronger. In order to find new unknown physics, we must obtain just as much data from the computer simulations as we get from the experiments. Like with two fingerprints, we analyze all the different curves and shapes in the graphs obtained from these two data sets. Any discrepancy found in them would give a clue on where to search for discoveries such as new types of elementary particles, or even change our view on how our Universe works.

My contribution to particle physics is connected to speeding up the methods of mathematical simulations of the collision data. A calculation of each clash of particles, like a collision of two asteroids, involves keeping track of a tremendous mess of collision products — where they fly, how fast and how they interact with each other. What is even more complicated to calculate are predictions about what kind of particles get to be created in each of these collisions. This is determined by the laws of *Quantum Field Theory*, a theory stating that all particles can be viewed as energetic bumps in some invisible fields spanning the whole Universe. The theory predicts how particles get created and destroyed in the interaction points like waves on a drum membrane when it is hit.

Some of the particles created in these tiny interaction points attract each other so strongly that they combine long before hitting the detectors. These particles are called *quarks* and *gluons* and the force that creates these immensely strong bounds is called, well, the *strong force*. An even more peculiar feature of this force is that unlike the electromagnetic charge, the strong force charge comes in three types. Gluons and quarks interact differently depending on what charge they possess. Even though simplified methods to simulate particle collisions approximately exist, for exact calculations, all of the possible color charge combinations have to be considered. This makes one of the greatest bottlenecks of the whole simulation process and is a challenge that this thesis offers a possible solution to.

In my thesis, I computationally implemented a new technique that uses abstract mathematical objects called *multiplet bases* that could potentially speed up the strong force calculations. We have already validated our method for the simplest collisions. However, the highest hope is to update the multiplet bases method to be able to calculate collisions where eight or more gluons and quarks appear and where the speed differences would become more significant. When this is achieved, it should immensely increase the capacity of simulating particle collisions. In this way, physicists would be able to search for even more complicated processes and in this complexity maybe some great discoveries hide.

Populärvetenskaplig beskrivning

Det är häpnadsväckande att för att utforska de minsta delarna som vår värld består av krävs de största maskinerna vi har skapat. På gränsen mellan Schweiz och Frankrike kan man hitta det bästa exemplet, Large Hadron Collider, en 27 kilometer lång ring där forskare låtar knippen av runt 100 miljoner protoner kollidera nästan 40 miljoner gånger per sekund. Trots alla betydande upptäckter (du har möjligtvis hört om Higgsbosonen som upptäcktes 2012) påstår partikelfysiker ofta att ännu djupare forskning av det grundläggande i universum kräver ännu större experiment och ännu fler kollisioner. Någonting är däremot ofta osagt i denna jakt efter snabbare, större och kraftigare experiment. För att hitta ny, okänd fysik, måste vi skaffa lika mycket data från våra datorsimuleringar som vi får från våra experiment. Precis som med två fingeravtryck, analyserar vi olika kurvor och former i graferna som vi skapar från dessa två datauppsättningar. Varje avvikelse kan ge oss en ledtråd om var vi borde leta efter nya partiklar eller till och med förändra vår syn på hur världen fungerar.

Mitt bidrag till partikelfysiken är att implementera en metod som ska öka hastigheten på simulationerna av partikelfysikexperimenten. Att beräkna varje partikelkollision är som att bokföra en krock mellan två asteroider, det kräver att man håller reda på en enorm mängd små bitar från asteroiderna — vart de flyger, hur snabbt och hur de påverkar varandra. Det som är ännu mer komplicerat är att beräkna vilka partiklar som kommer att skapas i dessa kollisioner. Det bestäms av *kvantfältteorins* lagar, en teori som säger att en partikel kan beskrivas som ett knippe energi i ett osynligt fält som sträcker sig genom hela universum. Teorin förutsäger hur partiklar skapas och förstörs liksom vågor på trumman som slås.

Några av partiklarna som skapas i kollisionerna hålls samman så starkt att de har satt sig ihop länge innan de träffar detektorn. Dessa partiklar kallas *kvarkar* och *gluoner*. Den starka kraften som binder dem till varandra så starkt kallas för den *starka kärnkraften*. En märklig egenskap med denna kraft är att till skillnad från den elektromagnetiska laddningen kommer den starka laddningen i tre olika sorter. Gluoner och kvarkar påverkar varandra olika beroende på vilka laddningar de har. Trots att enklare metoder finns för att ungefärligt beräkna partikelkollisioner, måste man för exakta beräkningar överväga alla kombinationer av laddningen i partiklar. Det utgör en av de största utmaningarna i hela beräkningsprocessen.

I mitt examensarbete, implementerade jag en ny metod som använder abstrakta matematiska objekt som kallas *multiplet baser*, för att öka hastigheten på beräkningar av den starka kärnkraften. Vi har redan validerat vår metod för enkla processer. Det slutliga målet är däremot att uppdatera metoden så att det blir möjligt att beräkna kollisioner där åtta eller fler gluoner och kvarkar skapas och där hastighetsskillnaderna mellan beräkningarna är som tydligast. När det är uppnått, kommer det att öka vår förmåga att simulera partikelkollisioner med exakt färgstruktur oerhört. På detta sätt, ska fysiker kunna leta efter ännu mer komplicerade processer och i denna komplexitet finns kanske några stora upptäckter.

Contents

1	Introduction	1
2	Theory	3
2.1	Group theory	3
2.1.1	Special unitary group $SU(N_c)$	4
2.2	Quantum chromodynamics	6
2.3	The diagrammatic birdtrack notation	8
2.3.1	Generalized vertices	10
3	The color structure calculation using multiplet bases	14
3.1	Trace bases	14
3.2	Multiplet bases	16
3.3	Important group theoretical identities	18
3.4	Decomposing color-invariant quantities into the Wigner 3j and 6j coefficients	21
3.4.1	An arbitrary structure	21
3.4.2	QCD amplitude contracted with a basis vector of the form as in figure 1a	22
4	The implementation of the program	25
4.1	Conventions and the normalization	25
4.2	The implementation	27
5	Usage of the program and examples	29
5.1	The building blocks of the program	29
5.2	Color structure simplification	31
5.3	Full color structure calculation	33
5.4	Simplifying amplitudes containing arbitrary representations	35
5.5	Evaluating amplitude squares using multiplet bases	36
5.6	The built-in consistency and validity checks	37
5.7	Treatment of the number of colors N_c	38
5.8	Validation	39

6	Conclusion and Outlook	41
A	The available <i>ColorMath</i> input	44
B	Alternatives to the three vertex loop contracting functions	44
C	Additional functions available in the program	45

1 Introduction

The ongoing upgrade of the Large Hadron Collider (LHC) is going to enable probing particle physics processes with lower cross-sections and more complicated signatures [1, 2]. These processes include the search for beyond the standard model physics, such as the production of hypothetical supersymmetric particles like gluinos and charginos [3] or probing CP-violating standard model processes [4]. Moreover, it would allow reducing the statistical uncertainties on parameters determining standard model processes such as the cross-sections of Higgs boson production and its couplings [5].

Tests of the standard model and beyond the standard model predictions require comparing the experimental data with the Monte Carlo simulations with the same accuracy [6]. In this way, experimental advances require increasing both the speed of numerical calculations and data accuracy. To achieve the latter, for several processes it is also suggested to increase the order of the perturbative expansion of Quantum Chromodynamics (QCD) and electroweak calculations¹ from next-to leading order (NLO) to next-to-next-to leading order (NNLO) [7, 8]. Similarly, higher energies of collisions allow processes with higher multiplicities of partons in this way increasing the complexity of the calculations.

A major challenge for simulating events with color charged particles is the treatment of the color structure. The need for it appears already when calculating the transition matrix element of the hard scattering, that is, of the processes involving momentum transfer of more than 1 GeV, where the strong coupling constant is small. In addition, the particles created in the hard scattering emit virtual gluons and quark-antiquark pairs. In the calculations of this radiation, large logarithms often appear that invalidate the perturbative calculations. Hence, all the orders of perturbation have to be considered. All order perturbative physics of these emissions is usually approximated through parton showers [9, 10] or by resummation [11, 12, 13], which both contain evaluation of the color structure. Despite being present in these steps of particle physics event generation, the common methods for the exact calculations of the color structure using different types of non-orthogonal bases, which we discuss later in this section, break down due to non-sufficient computation power when the number of gluons reaches around 7 or 8 [14]. Increasing this number and satisfying the demand caused by the increased experimental opportunities is a problem that the multiplet bases used in this thesis offer a possible solution.

The complication of treating the color structure arises due to the non-Abelian nature of the strong force, i.e., that there are three types of color charges, called red, green, and blue. In addition, QCD includes also gluon self-interactions. Hence, the color structure of the QCD processes is non-trivial, that is, the result depends on the colors of the interacting particles in a non-trivial way. Experimental evidence shows that color charged particles are

¹We refer to calculations of the transition matrix element and, consequently, of the cross-section of a process at the momentum scale where the coupling constants (electromagnetic, weak, strong) involved are small. Hence the result can be approximated by taking terms up to a finite order in the perturbative expansion in the coupling constants.

always confined in color neutral hadrons, a fact called color confinement. In other words, one can only measure color invariant quantities, that is, color singlets. Consequently, an exact treatment of the cross-section calculations of a QCD process involves averaging the matrix element over all the possible combinations of the incoming colors of particles and summing over all the possible outgoing colors. In addition, the sum over the colors of all virtual particles is hidden inside the matrix element. Since each gluon comes in eight different types and quarks in three, the color sum gets immensely large very quickly. For example, a process involving 5 gluons, 2 quarks, and 2 antiquarks would involve a sum over $8^5 \cdot 3^4 = 2654208$ possible indices. For this reason, alternative methods for doing this color sum are usually used.

The usual treatment of the color sum is to express the color structure of the process in sets of tensors, called bases. The most common and conceptually simplest bases are the trace bases [14, 15, 16, 17, 18, 19], and other non-orthogonal bases [20, 21, 22]. The non-orthogonality of these bases is their main drawback, which causes an increase in the number of terms when performing amplitude squares from the number of basis states to the square of that. Moreover, the set of basis states is often overcomplete, meaning that one requires more basis states than the dimensionality of the color space.

In this way, increasing the complexity of processes for which the color sum can be calculated exactly beyond the seven partons mentioned earlier can be achieved using orthogonal sets of bases. One such option is expressing the color structure in its irreducible representations (multiplets), that by construction are orthogonal [13, 23, 24]. A general recipe for constructing these so-called multiplet bases has been described recently [24]. Moreover, it has been found that expressing the color structure in these bases can be done without explicitly constructing them but instead using the so-called Wigner 3j and 6j coefficients, see, for example [25, 26]. These coefficients have all been calculated for decomposing LO and NLO processes having up to six gluons and quark-antiquark pairs in total [26, 27]. Despite all this theoretical effort, no computational implementation of using the Wigner 3j and 6j coefficients for automatic color structure decomposition onto the multiplet bases has so far been published. The multiplet bases have been shown to be suitable also for the recursion, a different approach of dealing with the color structure where more complicated structures are derived from simpler ones [28]. This thesis, however, is focused on the full color structure decomposition in the multiplet bases.

In this thesis, we present a program written in Mathematica [29] demonstrating as a proof-of-principle a computational implementation of the color structure decomposition into the multiplet bases. Given an input of a QCD amplitude and the corresponding multiplet basis, the components of the amplitude on the basis states can be calculated. Furthermore, the corresponding multiplet basis can be constructed automatically. Lastly, the color sum can be evaluated by squaring all of the components on the basis states and summing them. Consequently, the multiplet bases are shown to be a possible alternative to the non-orthogonal bases currently used in the event generators. Nevertheless, a full replacement of the non-orthogonal bases with the multiplet bases would require further optimization

of the program, as well as extending the available list of Wigner 6j coefficients that are currently only available for calculations of up to 6 gluon and quark-antiquark pairs in total [26, 27].

Additionally to the color structure calculations, the program can be used as a platform for performing color summed tensor calculations with the irreducible representations of $SU(N_c)$, where $N_c \geq 2$ is the number of colors. Any fully color-summed (contracted) tensor can be evaluated in terms of the Wigner 3j and 6j coefficients whenever these coefficients are available, and several identities for simplifying partially contracted color amplitudes are available. Finally, a work with tensors of arbitrary representations of $SU(N_c)$ is permitted, as well as visualizing tensor expression in the diagrammatic birdtrack notation. In this way, the program can be viewed as an extension of the Mathematica package *ColorMath* [18]. The program is available upon request and a publication of it in a package form is possible.

This thesis is organized as follows. First, in section 2 we provide the most relevant theoretical concepts from group theory and QCD. Furthermore, we introduce the birdtrack notation, a technique for group theoretical calculations which we use afterwards in section 3 to derive and justify the methods used in the created program. The explanations in section 2 and section 3 are often kept general, without adapting any particular convention, while the later sections are narrowed down to our particular solution of color structure decomposition process. In section 4 we turn theory into practice by describing how color structure decomposition is computationally implemented in the Mathematica program. Afterwards, in section 5 we document the usage of the program, provide examples of the application and explain the tests we made to validate the program.

2 Theory

In this chapter, we introduce the theoretical base for this thesis. We begin by stating the most important parts of group theory, particularly paying attention to the special unitary groups $SU(N_c)$. Using these concepts, we take a look at the Lagrangian of QCD and the vertices following from it. Finally, we define the birdtrack notation, a powerful tool for performing tensor calculations and state the advantages of this notation in comparison to the equivalent matrix and tensor notations.

2.1 Group theory

Symmetries in physics are of great importance since studying them can provide information about what conserved currents and charges the physical system has, what are the allowed physical states and what are their properties. In particle physics, the electromagnetic, weak, and strong force arise as a consequence of imposing three symmetries: $U(1)$, $SU(2)$, and $SU(3)$. Moreover, requiring new symmetries that are eventually broken in the world

that we observe is the usual method for the search of the physics beyond the standard model.

Symmetries are described as transformations that leave a physical system invariant. These symmetry operations fulfill the properties of a group [30, 31].

Def. A group G is a set of elements $\{g_i, g_j, g_k \dots\}$ and a bilinear operation \circ between the set elements, called group multiplication that satisfies closure, associativity, the existence of an identity element e , and the existence of an inverse element.

A homomorphism from abstract group elements to matrices is called a group representation. First, by a general linear group $GL(V)$ we denote all matrices that act on a vector space V over a field F and map it back into V . Using that, we define a group representation $D(g)$ as a homomorphism from G to $GL(V)$, that is,

$$D(g): G \rightarrow GL(V).$$

Every group possesses the so-called singlet or trivial representation 1, which is just a number 1 and might possess other representations. Moreover, for a given representation one can find an often infinite set of equivalent representations, related by similarity transformations,

$$D'(g) = S^{-1}D(g)S, \tag{2.1}$$

where S is a matrix satisfying $S^{-1}S = I$ and I is the identity matrix. Such transformations correspond to a change of basis in V .

In some cases, one can find a basis of the vector space V such that a non-trivial subset of this basis is closed under the action of the representation. In other words, there exists $W \subset V$, $W \neq \{\}$, $W \neq V$, so that $\forall g \in G, \forall w \in W: D(g)w \in W$. In such cases, we say that the representation is reducible and otherwise — irreducible [30].

A representation, $D(g)$, is reducible if and only if it can be brought into a block-diagonal form:

$$S^{-1}D(g)S = \begin{pmatrix} D_1(g) & 0 & \dots & 0 \\ 0 & D_2(g) & \dots & 0 \\ \vdots & & \ddots & 0 \\ 0 & \dots & & D_n(g) \end{pmatrix},$$

where $D_1(g), \dots, D_n(g)$ are the irreducible representations $D(g)$ reduces into. Such a decomposition is denoted by the Direct sum of the corresponding irreducible representations

$$D(g) = D_1(g) \oplus D_2(g) \oplus \dots \oplus D_n(g).$$

2.1.1 Special unitary group $SU(N_c)$

As will be apparent from the QCD Lagrangian, to be defined in section 2.2, this theory is invariant under the continuous transformations according to $SU(N_c)$ group with $N_c = 3$.

The method of color structure decomposition implemented in this thesis fully relies on the irreducible representations of this group.

We define a vector v^i to belong to the N_c dimensional complex vector space $V = \mathbb{C}^{N_c}$ and its conjugate v_i to belong to its dual space \bar{V} . Then group elements of $SU(N_c)$ are matrices g that preserve the bilinear product of vectors $v^i v_j \longrightarrow v^i g g^\dagger v_j = v^i v_j$ and do not change phase or reflect v^i . The first condition requires unitary $g \cdot g^\dagger = I$, while the second condition requires $\det(g) = 1$. Matrices with $\det(g) = 1$ are called special.

In this way, the definition of $SU(N_c)$ group is:

Def.

$$SU(N_c) := \{g \in GL(\mathbb{C}^{N_c}) \mid g \cdot g^\dagger = I, \det(g) = 1\} \quad (2.2)$$

The two conditions on g reduce the number of independent parameters, needed to specify g uniquely, to $N_c^2 - 1$, which is called the dimension of $SU(N_c)$. We are interested in group elements of a continuous and smooth group (Lie group) which can be written as

$$g = \exp \left(\sum_{a=1}^{N_c^2-1} i\theta^a t^a \right),$$

where t^a are matrices called infinitesimal generators of the Lie group and θ^a are real parameters [30]. The generators, t^a , satisfy the Lie algebra commutation relation (where we adapt Einstein's summation convention):

$$[t^a, t^b] = i f^{abc} t^c, \quad (2.3)$$

where f^{abc} is an anti-symmetric matrix called the structure constant of $SU(N_c)$. In other words, the Lie algebra is closed under the action of the Lie algebra commutation relation, the so-called Lie bracket. For $SU(N_c)$, the unitarity of g implies the Hermiticity of t^a , while the unit determinant implies the tracelessness of t^a , that is, $\text{Tr}[t^a]=0$.

The t^a matrices are orthogonal and can be normalized arbitrarily

$$\text{Tr}[t^a \cdot t^b] = T_R \delta^{ab}. \quad (2.4)$$

Since $SU(2)$ Pauli matrices are usually taken to be $t^a = \frac{1}{2}\sigma^a$ and normalized as $\text{Tr}[\sigma^a \cdot \sigma^b] = 2\delta^{ab}$ (and analogically for $SU(3)$ Gell-Mann matrices $t^a = \frac{1}{2}\lambda^a$), then the generator normalization constant is usually fixed to $T_R = \frac{1}{2}$. In the Mathematica program created in this thesis we, however, let the user define the normalization or leave it as a constant.

Since in QCD $N_c = 3$, it is convenient to adapt a common notation, and label the most frequently encountered representations of $SU(N_c)$ by their dimension in the $N_c = 3$ case [26, 27]. The vector space V , which can itself be viewed as an N_c dimensional representation — the so-called defining representation — we thus denote by $\mathbf{3}$, a triplet.

Using group theoretical methods, such as Young tableaux, it is possible to determine how a two representation direct product (tensor product) decomposes into irreducible representations, see, for instance [25, 31]. One important example is the direct product of a triplet with an anti-triplet, that is,

$$3 \otimes \bar{3} = 1 \oplus 8, \quad (2.5)$$

where 1 is the singlet representation. The octet representation, 8, also called the adjoint representation of the group, can be defined in terms of the structure constants, f^{abc} [25, 30]. For the case of $N_c = 2$, the decomposition corresponding to Eq. (2.5) is the usual two spin $\frac{1}{2}$ state decomposition into a singlet and a triplet [32]. In other words, spin $\frac{1}{2}$ is the corresponding fundamental representation in the $N_c = 2$ case and the triplet is the corresponding adjoint representation.

Not all representations can be denoted by integers, since there are in general several representations having the same dimensions for $N_c = 3$. While for all the contents of this thesis, the simple notation explained above is sufficient, in section 4.1 we explain a unique and N_c -independent label for the representations that we have adapted in the created program.

2.2 Quantum chromodynamics

QCD is a part of the standard model of particle physics — the currently most successful theory predicting the processes involving the elementary particles — that describes the strong interaction. This interaction acts on particles possessing a color charge (or color), namely, quarks and gluons. There are $N_c = 3$ colors in QCD and the strong force is discovered to be invariant under the simultaneous change (rotation) of all three colors, motivating the SU(3) symmetry of the theory. Moreover, writing QCD as an SU(3) invariant gauge field theory turns out to explain many experiments precisely. While in this section, we describe QCD as a theory satisfying SU(3) invariance, a general SU(N_c) invariant theory with an arbitrary N_c and the same vertices as QCD can be written similarly.

The theory of QCD consists of quarks $q^i \in 3$, antiquarks $\bar{q}_i \in \bar{3}$ and gluons $G^a \in 8$, where $a \in 1, \dots, 8$ is the gluon index. Using this notation, the QCD part of the standard model Lagrangian can be written as,

$$\begin{aligned} \mathcal{L} = & \sum_{q=u,d,s,c,b,t} (\bar{q}_i (i\not{\partial} - m) q^i - \frac{1}{4} (\partial_\mu G_\nu^a - \partial_\nu G_\mu^a)^2 + g_s G_\mu^a \bar{q}_i \gamma^\mu (t^a)^i_j q^j \\ & + g_s f^{abc} (\delta_\mu G_\nu^a) G^{\mu b} G^{\nu c} - \frac{1}{4} g_s^2 (f^{eab} G_\mu^a G_\nu^b) (f^{ecd} G^{\mu c} G^{\nu d}), \end{aligned} \quad (2.6)$$

where the sum runs over all quark types; g_s is the strong force coupling constant and $\bar{q}_i = (q^i)^\dagger \gamma^0$, see, for example [33]. Here and in what follows, we put the quark indices (indices transforming as triplets) upstairs and the anti-quark indices (transforming as anti-triplets) downstairs.

2.3 The diagrammatic birdtrack notation

For performing group theoretical calculations with tensors in this thesis, we will be using a powerful diagrammatic notation called birdtracks [24, 25, 35, 36]. Visually, birdtracks often look similar to Feynman diagrams. However, while a Feynman diagram is a diagrammatic representation of contributions to a transition matrix element in a perturbative expansion, birdtracks represent tensors. Moreover, the birdtrack notation allows performing tensor calculations without converting to the equivalent tensor or matrix notations. The QCD vertices encountered in Eq. (2.7) to Eq. (2.9), where we assigned a tensor to a particular diagram, are thus examples of birdtracks. In this chapter, after explaining the advantages of birdtracks and presenting the necessary definitions, we discuss how to apply the birdtrack notation to perform such tensor operation as tensor scalar products and other operations that are going to be used in the further sections.

All the tensor operations can be equivalently defined in the birdtracks, but it has several advantages over the ordinary matrix and tensor notations. First, unlike matrices, it permits working with tensors of an arbitrary rank and contracting them arbitrarily. Furthermore, unlike the ordinary tensor notation, birdtracks do not require writing and keeping track of dummy indices and in some cases even of the free indices. This is particularly handy for the later parts of this thesis where we describe tensors corresponding to many different representations and their tensor products. For such tensors, each of the different representations must have its own set of indices assigned and keeping track of all those would get confusing and cumbersome very fast (see the scalar product in Eq. (2.15) for example). Finally, it is often more straightforward to find and depict patterns in a diagrammatic representation in comparison to a large block of symbols.

We are going to define the birdtrack notation for $SU(N_c)$ tensors. With analogy to Feynman diagrams, we define two different kinds of lines, one for triplets and one for octets. Hence we draw the delta functions as propagators:

$$\delta_j^i \equiv \begin{array}{c} i \\ \longrightarrow \\ j \end{array}, \quad \delta_{a,b} \equiv \begin{array}{c} a \\ \overbrace{}^b \\ \end{array}, \quad (2.10)$$

where $i = 1, \dots, N_c$ is the triplet index, $j = 1, \dots, N_c$ is the anti-triplet index, and $a, b = 1, \dots, N_c^2 - 1$ the octet indices. Different representations will require different lines to distinguish them. While various conventions can be chosen for complex representations like triplets and anti-triplets, in this thesis, we define the arrow to point from the upper index (triplet index in Eq. (2.10)) into the lower index (anti-triplet index in Eq. (2.10)). Since the octet representation is real, the conjugated representation of an octet is the same as the representation itself, so there is no reason to differentiate between them by using different index positions. This transforms into discarding the arrow for octet lines.

It is useful to define a shorthand notation for n direct products of the same representation as

$$R^{\otimes n} \equiv \underbrace{R \otimes R \otimes \dots \otimes R}_{n \text{ times}}. \quad (2.11)$$

such as the j_1 line on Y in Eq. (2.14). Similarly, the contraction can be performed with all the lines in a birdtrack.

In birdtracks, there is no momentum flow, that is, no “in” or “out” and thus no distinction between an incoming quark and an outgoing antiquark (both are triplets, i.e. upper indices), as well as no distinction between an incoming antiquark and an outgoing quark (both anti-triplets, i.e. lower indices). As a consequence, the birdtrack can be turned and twisted as long as it keeps the same topology while still representing the same tensor. If the free indices are discarded, it has to be ensured that the external lines still exit the image in the same order as before twisting.

An example of twisting that is not allowed is interchanging the order in which two legs enter the vertex as shown below for the three gluon vertex.

$$\begin{array}{c}
 \begin{array}{c} a \\ | \\ \bullet \\ / \quad \backslash \\ b \quad c \end{array}
 \quad \longrightarrow \quad
 \begin{array}{c} a \\ | \\ \bullet \\ \backslash \quad / \\ b \quad c \end{array}
 \quad = \quad
 \left(\begin{array}{c} a \\ | \\ \bullet \\ / \quad \backslash \\ b \quad c \end{array} \right)^T
 \quad \equiv \quad
 \begin{array}{c} a \\ | \\ \bullet \\ / \quad \backslash \\ b \quad c \end{array}
 \quad = \quad (-1) \cdot \begin{array}{c} a \\ | \\ \bullet \\ \backslash \quad / \\ b \quad c \end{array} .
 \end{array}
 \tag{2.16}$$

Here we have written the indices explicitly to indicate that in the transformed term the lines exit the vertex in a reverse order (acb instead of abc). Such an interchange corresponds to transposing the vertex. In the term after the equivalence sign we have adapted the so-called Yutsis notation. According to this notation, a minus sign by the vertex denotes that the vertex is transposed or, in other words, that the order of lines is exchanged (lines are read clockwise instead of anti-clockwise)[37]. Since the three-gluon vertex corresponds to the anti-symmetric tensor then $if^{abc} = -if^{cba}$, and transposing the three-gluon vertex obtains a minus sign. In section 2.3.1, after introducing the generalized vertices, we show that the signs obtained by transposing the vertices encountered in this thesis are always ± 1 .

2.3.1 Generalized vertices

We will define the generalized vertex in the birdtrack notation as a tensor proportional to the Clebsch-Gordan coefficient and then discuss some important properties of these vertices.

We saw previously in Eq. (2.5) how a direct product of a triplet and an anti-triplet decomposes into the irreducible representations of a singlet and an octet, $3 \otimes \bar{3} = 1 \oplus 8$. This expression decodes the information that the $3 \times 3 = 9$ dimensional direct product space $3 \otimes \bar{3}$ can be spanned by two sets of bases. This space can, on the one hand, be spanned by the eigenvectors of the matrices belonging to $3 \otimes \bar{3}$. On the other hand, it can be spanned by the total set of the eigenvectors of the representations 1 and 8. In general, it is possible to write the exact expansion of the direct product of two vectors $|\alpha_1 i_1\rangle$ and $|\alpha_2 i_2\rangle$ as

$$|\alpha_1 i_1\rangle \otimes |\alpha_2 i_2\rangle = \sum_{\alpha, i} |\alpha i\rangle \langle \alpha_1 i_1; \alpha_2 i_2 | \alpha i\rangle, \quad (2.17)$$

where $\langle \alpha_1 i_1; \alpha_2 i_2 | \alpha i\rangle$ are the Clebsch-Gordan coefficients, α_1 and α_2 are two irreducible representations and i_1 and i_2 are the corresponding vector indices, α are all irreducible representations such that $\alpha \in \alpha_1 \otimes \alpha_2$ and i is the vector index in α [25, 30, 37]. An example of this expansion is the addition of two angular momenta in quantum mechanics, where it is more common to denote the Clebsch-Gordan coefficients and the vectors using the angular momenta of the states j , which is related to the dimension of the representation by $\alpha = 2 \cdot j + 1$, and using the z component of the angular momentum m instead of the vector index [32].

Based on this, we define a generalized vertex to be a tensor proportional to the Clebsch-Gordan coefficient, that is, in the birdtrack notation

$$\begin{array}{c}
 \alpha_1, i_1 \\
 \nearrow \\
 \alpha_3, i_3 \\
 \searrow \\
 \alpha_2, i_2
 \end{array}
 \propto \langle \alpha_1 i_1; \alpha_2 i_2 | \alpha_3 i_3\rangle. \quad (2.18)$$

The normalization of the generalized vertex can be chosen arbitrarily, but, as described later in this chapter, we normalize them to unity. Since in this thesis we calculate the color summed tensor expressions, the indices i_1 , i_2 and i_3 will always be summed over and, as explained above in section 2.3, can be discarded.

From $3 \otimes \bar{3} = 1 \oplus 8$ and Eq. (2.18), we see that triplet and anti-triplet lines connecting can create two distinct vertices — the quark-gluon vertex as in Eq. (2.7) (where the outgoing representations is an octet) and a delta function as in the left equation in Eq. (2.10) (the outgoing representations is a singlet). The non-existence of any other vertex can be seen as a consequence of a vanishing Clebsch-Gordan coefficient.

The direct product of two octets is

$$8 \otimes 8 = 1 \oplus 8 \oplus 8 \oplus 10 \oplus \bar{10} \oplus 27 \oplus 0, \quad (2.19)$$

where this decomposition into the irreducible representations contains also representations not present in the standard model, see, for example, [31]. Note that the representations of dimension 8 appear twice in Eq. (2.19). This means that two orthogonal eight-dimensional vector spaces have to be used to characterize it; we need two different vertices connecting three octets.

In QCD, only one of such vertices is found, namely the three-gluon vertex if^{abc} as in Eq. (2.8). It is conveniently to choose if^{abc} as one of the two vertices and an orthogonal vertex to it — as the other one. The later turns out to be the symmetric structure constant of $SU(N_c)$, d^{abc} , defined by the anticommutator of the generators, $\{t^a, t^b\} = \frac{2T_R}{N_c} \delta^{ab} + d^{abc} t^c$.

In birdtracks we denote d^{abc} by a three-gluon vertex with a gray circle in the middle

$$d^{abc} \equiv \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \quad . \quad (2.20)$$

In particular, it can be shown that the scalar product between the two three-gluon vertices is

$$\left\langle \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \middle| \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \right\rangle = \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} = - \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} = (-1) \cdot \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} , \quad (2.21)$$

where in the second step we only twisted the inner lines without changing the order of them, but in the third step related the vertices with their transposed vertices. These two steps together correspond to renaming two of the dummy indices in the tensor notation. In the final step, we used the anti-symmetry of if^{abc} from Eq. (2.16) and the symmetry of d^{abc} , which together acquire a minus sign. Hence, in Eq. (2.21) we see that a scalar product is equal to a minus-itself, so it has to vanish and d^{abc} is thus orthogonal to if^{abc} .

We refer to different kinds of vertices connecting the same representations as different instances of the same vertex. We will indicate the instance of the vertex whenever it has several of them and when identifying them is important. We will, for example, order the instances and indicate them by an integer a on the birdtracks as

$$\begin{array}{c} \gamma \\ \diagup \quad \diagdown \\ \text{---} \\ \diagup \quad \diagdown \\ \beta \end{array} \quad \alpha \quad . \quad (2.22)$$

The vertex if^{abc} we identify with the instance $a = 1$ and d^{abc} with $a = 2$. In general, vertices with different instances do not have to be orthogonal. Nevertheless, it is always possible to choose them to be orthogonal as we do in this thesis. In addition to vertex instances, the vertices are also orthogonal unless all their representations are equal. that is, taking the scalar product of two vertices, we get

$$\left\langle \begin{array}{c} \gamma' \\ \diagup \quad \diagdown \\ \text{---} \\ \diagup \quad \diagdown \\ \beta' \end{array} \quad \alpha' \quad \middle| \quad \begin{array}{c} \gamma \\ \diagup \quad \diagdown \\ \text{---} \\ \diagup \quad \diagdown \\ \beta \end{array} \quad \alpha \quad \right\rangle = \begin{array}{c} \alpha' \\ \diagup \quad \diagdown \\ \text{---} \\ \diagup \quad \diagdown \\ \beta' \end{array} \cdot \begin{array}{c} \gamma \\ \diagup \quad \diagdown \\ \text{---} \\ \diagup \quad \diagdown \\ \beta \end{array} = \delta_{ab} \delta_{\alpha'}^{\alpha} \delta_{\beta'}^{\beta} \delta_{\gamma'}^{\gamma} a \begin{array}{c} \alpha \\ \text{---} \\ \beta \end{array} \quad . \quad (2.23)$$

The normalization constant in Eq. (2.23) is called Wigner 3j coefficient² and the representations γ in it corresponds to the line inside the circle. For the theoretical description of color structure decomposition in section 3 we keep the normalization general. However,

²The “3j” in *Wigner 3j coefficients* and, in general, “3n j” in *Wigner 3n j coefficients* and *3n j symbols*, where n is a positive integer, refers to their applications in atomic physics, where they are used for recoupling angular momenta of quantum systems that in total contain $3n$ angular momenta.

for the created Mathematica program, as explained in section 4.1, we choose to normalize the generalized vertices to one, absorbing the constant appearing from Eq. (2.23) in the definition of the vertices. In other words, all Wigner 3j coefficients in this can be set to unity.

From the birdtrack of the vertex in Eq. (2.18), it can be seen that it possesses symmetries that are not present in the Clebsch-Gordan coefficients. Since reversing the direction of the arrow corresponds to conjugating the corresponding representation, the vertex for representations α_1 and α_2 decomposing into α_3 is the same as for representations $\bar{\alpha}_3$ and α_2 decomposing into $\bar{\alpha}_1$. In this way, the generalized vertex turns out to be proportional to the so-called 3j symbols (not Wigner 3j coefficients in terminology used in this thesis) that are commonly used in atomic physics as a more symmetric alternative to Clebsch-Gordan coefficients for three angular momenta recoupling [25, 37, 38]. In section 3.3 we introduce also the Wigner 6j coefficients that are products of four generalized vertices (which are summed over all the dummy indices).

The generalized vertices can be transposed, similarly as was shown on the three-gluon vertex example, Eq. (2.16),

$$\begin{array}{c} \gamma \\ \diagdown \quad \diagup \\ \text{---} \quad \text{---} \\ \diagup \quad \diagdown \\ \beta \end{array} \quad \alpha = \sigma_T^{\alpha\beta\gamma,a} \begin{array}{c} \gamma \\ \diagdown \quad \diagup \\ \text{---} \quad \text{---} \\ \diagup \quad \diagdown \\ \beta \end{array} \quad \alpha \quad (2.24)$$

and conjugated

$$\left(\begin{array}{c} \gamma \\ \diagdown \quad \diagup \\ \text{---} \quad \text{---} \\ \diagup \quad \diagdown \\ \beta \end{array} \quad \alpha \right)^* = \sigma_C^{\alpha\beta\gamma,a} \begin{array}{c} \gamma \\ \diagdown \quad \diagup \\ \text{---} \quad \text{---} \\ \diagup \quad \diagdown \\ \beta \end{array} \quad \alpha \quad , \quad (2.25)$$

where the coefficients $\sigma_T^{\alpha\beta\gamma,a}$ and $\sigma_C^{\alpha\beta\gamma,a}$ are ± 1 . The absolute square of the vertex, which is the product of the vertex with its Hermitian conjugate (complex conjugate and transpose), should always be positive. This puts a constraint on the transposition and conjugation coefficients $\sigma_T^{\alpha\beta\gamma,a} \cdot \sigma_C^{\alpha\beta\gamma,a} = 1$. Just like with the normalization, the sign of the vertex can be chosen freely and absorbed by the redefinition of the vertex, unless the conjugated vertex or the transposed vertex is equal to the vertex itself. For such cases, all the signs of the vertices that are used in program created have been determined in [26, 27].

As shown in Eq. (2.19), the decomposition of the two-octet tensor product also includes a zero-dimensional representation. This representation appears and has a non-zero dimension only for $N_c > 3$. Similarly, for the $N_c = 2$ case even the representations 10, $\bar{10}$ and one of the 8 disappear from Eq. (2.19). These examples, show that there exist representations that are only present for large enough N_c . In section 5.7 we describe how the representations with vanishing dimensions are treated in the program created.

3 The color structure calculation using multiplet bases

The main goal of this thesis is to simplify the calculations of the color structure of $SU(N_c)$ processes. A distinct property of the QCD vertices (Eq. (2.7) - Eq. (2.9)) is that they always contain the same number of triplet as anti-triplet lines, $n_q = n_{\bar{q}}$. In other words, even though $SU(N_c)$ symmetry permits vertices proportional to the totally anti-symmetric tensor, ϵ^{ijk} , with three triplet indices, such vertices do not appear in QCD. Furthermore, as discussed in section 2.2, all of the vertices and their combinations like the one in Eq. (2.13) are color singlets. Such amplitudes satisfying these two properties are said to belong to the color space, that is, the singlet subspace of $8^{\otimes N_g} 3^{\otimes n_q} \bar{3}^{\otimes n_{\bar{q}}}$ and we denote such amplitudes by \mathbf{c} .

Calculating measurable quantities involves taking the absolute square of the amplitude that contains contributions from all the possible Feynman diagrams with the given external partons. In addition, as explained in section 1, one has to sum (average) over all the colors of the outgoing (incoming) particles. For the color structure of the amplitude, this corresponds to taking its absolute square, $|\mathbf{c}|^2$, according to the definition of the scalar product in Eq. (2.15). As mentioned in section 1, the usual treatment of the color structure is to express \mathbf{c} in different sets of so-called bases, that is, sets of tensors into which all the color structures with the same external partons can be expressed.

In this section, we first give an overview of the most frequently used bases, which are all non-orthogonal and thus ineffective when squaring the color structure. This main drawback of non-orthogonal bases is solved by the multiplet bases. The ways of constructing the multiplet bases are explained in section 3.2. Afterwards, in section 3.3, we introduce several useful group theoretical identities, which are right away applied in section 3.4 where we present a method for decomposing a color structure in the multiplet bases. An important part of this will be showing that any group-invariant quantity can be expressed in the Wigner 3j and 6j coefficients. Moreover, we also explain how the decomposition recipe has to be changed to put constraints on the required 6j coefficients, when decomposing scalar products of QCD amplitudes with multiplet basis states as in Eq. (3.5). This decomposition method together with the group theoretical identities presented in section 3.3 is the foundation of the calculations performed by the Mathematica program created in this thesis.

3.1 Trace bases

The most popular bases to express the color structure \mathbf{c} in are the so-called trace bases [15, 14, 18, 19, 24]. The two major properties of the trace bases are the existence of a straightforward general algorithm for decomposing a color amplitude into these bases and a simple method for constructing them for a general color amplitude. For leading order (LO, tree level) color structures consisting only of N_g gluons, the basis vectors are all the possible distinct traces over the products of N_g generators of $SU(3)$. Defining σ as a

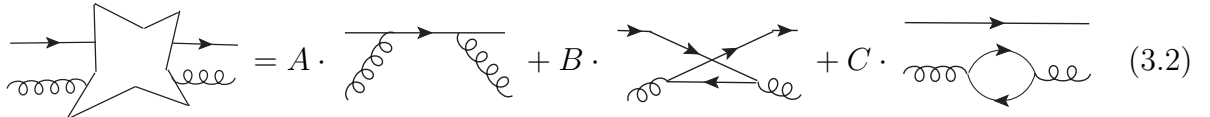
permutation from the permutation group S_{N_g-1} , an arbitrary amplitude \mathcal{A} that besides color structure contains also kinetic factors can be written in the trace basis as:

$$\mathcal{A} = \sum_{\sigma \in S_{N_g-1}} \mathcal{A}_\sigma \text{tr}[t^1 t^{\sigma(2)} \dots t^{\sigma(N_g)}], \quad (3.1)$$

where \mathcal{A}_σ do not contain any color index and are often called color ordered amplitudes.

The number of basis states is equal to $(N_g - 1)!$, that is, the size of S_{N_g-1} . However, it can be reduced by approximately a factor of two by imposing the charge conjugation invariance, i.e., that a charge conjugated gluon only amplitude is equal to the amplitude itself [28]. Since no color invariant quantity can be built from one gluon, the size of the basis space for $N_g = 1$ is 0. This is reflected in Eq. (3.1) by the fact that the $SU(N_c)$ generators are traceless and all the basis states vanish. In the birdtrack notation, each of the basis states is a closed quark loop with N_g gluons attached to it. Generalization of the trace basis exists for amplitudes that also contain loops and external quark lines, where in the former case basis vectors must be extended to contain all possible products of quark loops [16, 17], while in the latter case, open quark lines with gluons attached must be added.

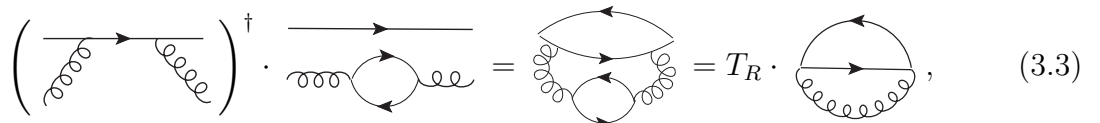
To demonstrate the simplicity of constructing the trace bases, we consider an arbitrary color structure consisting of two gluons and two quarks $\mathbf{c} \in 8^{\otimes 2} \otimes 3 \otimes \bar{3}$ as shown on the left-hand side in Eq. (3.2). The first step is to attach a quark-gluon vertex to each of the external gluons while the second step is to list all of the possible ways to connect the quark lines (the external ones and those attached to the gluon lines in the first step). For a given \mathbf{c} , there are three ways of doing this, therefore \mathbf{c} can be decomposed into the trace basis as shown in Eq. (3.2).



$$= A \cdot \text{diagram 1} + B \cdot \text{diagram 2} + C \cdot \text{diagram 3} \quad (3.2)$$

Some other advantages of the trace bases are the simplicity of describing the effects of gluon emission and gluon exchange [39, 40]. One merely has to attach new gluons to the existing basis.

Nevertheless, already in the example shown in Eq. (3.2) one can notice a crucial property of the trace bases, namely that the basis states are not always orthogonal. Taking the scalar product of the first and the third basis vector from the right-hand side we find



$$\left(\text{diagram 1} \right)^\dagger \cdot \text{diagram 3} = \text{diagram 4} = T_R \cdot \text{diagram 5}, \quad (3.3)$$

where in the last step we removed one of the quark loops by using Eq. (2.4), which expressed in birdtracks means that a quark loop with two attached gluons can be turned into a gluon

delta. The 3j coefficient on the right-hand side of the Eq. (3.3) is the normalization of t^a , which is greater than zero and thus the scalar product between basis vectors is non-zero. When calculating amplitude squares the non-zero scalar product generates cross terms, significantly complicating the computations.

Moreover, the trace bases turn out to be overcomplete for amplitudes with $N_c < N_g$ and, in this way, they would more accurately be described as spanning sets instead of bases [15]. Recursive methods for determining the size of the sets of the trace basis states exist, which reveal an approximately factorial scaling with N_g [15, 24]. From another perspective, the trace bases become orthogonal when $N_c \rightarrow \infty$ and the size of the trace basis can be seen as the dimension of the orthogonal basis spanning the color space in the case when $N_c \rightarrow \infty$. For this reason, the trace bases become more advantageous for theories with a larger N_c . Other non-orthogonal bases exist, such as Del Duca-Dixon-Maltoni [20] and color-flow bases [21, 22]. They all, however, possess the same drawbacks as the trace bases, namely, the same factorial scaling of the basis space with N_g and non-orthogonality.

Similar techniques for calculating the actual dimension of the color space demonstrate a more favorable exponential behavior with respect to N_g for finite N_c [24, 28]. As a comparison, for amplitudes with 5 gluons and no quarks or anti-quarks the length of the trace basis set is 44 (for all orders of perturbation series) and the size of the full vector space is 32, while for 8 gluons the ratio between these two becomes 14833/3598, rapidly increasing for a larger number of partons. Since calculating amplitude squares requires to square number of non-orthogonal basis states, this ratio of computational expenses for two kinds of bases becomes $14833^2/3598$. In the next section, we present the multiplet bases that are both minimal and orthogonal, solving these scaling problems.

3.2 Multiplet bases

Multiplet bases correspond to the irreducible representations (multiplets) of the color structure \mathbf{c} . We showed earlier in Eq. (2.5) and Eq. (2.19) how the direct products of two triplets and of two octets decompose into the irreducible representations. Similarly, the whole color structure \mathbf{c} can also be decomposed. In particular, we can connect all the external lines in vertices and determine what $SU(N_c)$ representations are possible for the internal lines that are created. There is, however, no unique method of connecting the lines and basis vectors might look different.

In this thesis, we use the kind of multiplet bases first presented in [27]. These are shown in figure 1a. They consist of all of the representations attached to a backbone of arbitrary representations $(\alpha_1, \alpha_2 \dots)$, where all N_p partons are arbitrarily grouped so that $\lceil \frac{N_p}{2} \rceil$ are on the left-hand side and $\lfloor \frac{N_p}{2} \rfloor$ are on the right-hand side. By $\lfloor x \rfloor$ ($\lceil x \rceil$) we denote the floor (ceiling) functions, i.e, the greatest (smallest) integer that is less (greater) than or equal to x . By construction, such bases are proper bases, that is, linearly independent and complete, and can be constructed to be orthogonal as well.

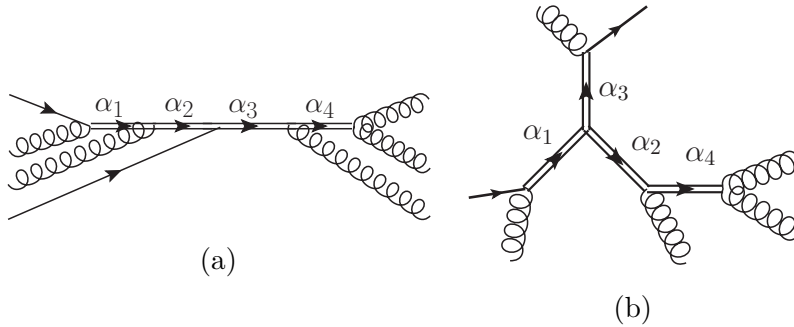


Figure 1: Two possible ways of constructing the orthogonal multiplet bases for the case of 7 partons. Bases used in this thesis are built by attaching the external lines to a backbone of irreducible representations (a). A more general kind of bases can be built by an arbitrary grouping of partons (b), but requires additional Wigner 6j coefficients for the color-structure decomposition onto these bases using the methods of this thesis

As explained in section 2.3.1, even if the decomposition of a direct product of representations into the direct sum of irreducible representations is known, the exact transformation of basis vectors between two spaces requires Clebsch-Gordan coefficients. Similarly, an exact projection of vectors onto a representation α requires a projection operator P_α . This operator P_α can be constructed by two Clebsch-Gordan coefficients (vertices), connected by a delta line of α and summing over all the states in α

$$P_\alpha = \begin{array}{c} \mu \qquad \mu \\ \diagdown \quad \diagup \\ \qquad \alpha \\ \diagup \quad \diagdown \\ \nu \qquad \nu \end{array} . \quad (3.4)$$

The projection operators satisfy idempotency, $P_\alpha P_\alpha = P_\alpha$, that is, projecting a projected vector does not change the vector, and completeness, $\sum_\alpha P_\alpha = 1$ where α runs through all the irreducible representations in $\mu \otimes \nu$. In other words, the completeness relation states that the total action of the sum of projection operators on a vector maps this vector back to itself.

To expand an arbitrary color structure in the corresponding multiplet basis, one has to find the projection operators onto this basis. The recipe for finding the Hermitian projection operators for the multiplet bases of the form figure 1a has been described in [27]. Nonetheless, as will be shown in section 3.4.1, the decomposition of \mathbf{c} into the multiplet basis can be done even without the projection operators if all of the required Wigner 3j and 6j coefficients are known. Consequently, we will not discuss the techniques of constructing the projection operators. In particular, to find the components of \mathbf{c} in its corresponding multiplet basis, it is first needed to find the possible representations on the backbone of the basis vector shown in figure 1a for the specific case of external partons. Afterwards,

with d_α according to the dimensionality, Eq. (3.6), and the both sides of Eq. (3.7) become equal.

A special case of Schur's lemma is used particularly often in this thesis. In this case, we choose the matrix mapping from α to β to consist of two-vertices

$$\begin{array}{c} \delta \\ \circlearrowleft \\ \alpha \rightarrow \beta \\ \circlearrowright \\ \gamma \end{array} = \frac{\begin{array}{c} \delta \\ \circlearrowleft \\ \gamma \\ \circlearrowright \\ \beta \end{array}}{d_\alpha} \begin{array}{c} \alpha \\ \rightarrow \\ \beta \end{array}, \quad (3.8)$$

As mentioned in section 2.3.1, the 3j coefficient appearing in the numerator in Eq. (3.8) is the vertex normalization, that in this chapter is kept arbitrary but in section 4.1 we choose set to one. In what follows, we are going to refer to Eq. (3.8) as Schur's lemma.

The completeness of the projection operators $\sum_\alpha P_\alpha = 1$ stated in section 3.2 can be written in the birdtrack notation

$$\begin{array}{c} \mu \\ \rightarrow \\ \nu \\ \rightarrow \end{array} = \sum_{\alpha \in \mu \otimes \nu} \frac{d_\alpha}{\begin{array}{c} \mu \\ \rightarrow \\ \alpha \\ \rightarrow \\ \nu \end{array}} \begin{array}{c} \mu \\ \rightarrow \\ \alpha \\ \rightarrow \\ \nu \end{array}, \quad (3.9)$$

where the normalization again can be proven by taking the trace on both sides. Taking the trace in birdtracks for structures with many indices, according to our conventions (see section 2.3 and [25]), is to connect the lines exiting from the right-hand side to the lines exiting from the left-hand side with no crossings of lines. For Eq. (3.9) it means connecting the both ends of μ line and the both ends of ν line. In this way, the left-hand side of Eq. (3.9) gives the product of two dimensions $d_\mu d_\nu$ while the right-hand side gives $\sum_\alpha d_\alpha$. These are just two different expressions for the dimension of the tensor product space $\mu \otimes \nu$, hence the normalization in Eq. (3.10) is correct.

Applying the completeness relation, Eq. (3.9), on a loop connecting three vertices, we can derive the so-called vertex correction relation:

$$\begin{array}{c} \alpha \\ \rightarrow \\ \epsilon \\ \rightarrow \\ \zeta \\ \rightarrow \\ \beta \end{array} \rightarrow \gamma = \sum_\lambda \frac{d_\lambda}{\begin{array}{c} \alpha \\ \rightarrow \\ \lambda \\ \rightarrow \\ \beta \end{array}} \begin{array}{c} \alpha \\ \rightarrow \\ \lambda \\ \rightarrow \\ \beta \end{array} \begin{array}{c} \epsilon \\ \rightarrow \\ \zeta \\ \rightarrow \\ \beta \end{array} \rightarrow \gamma = \sum_a \frac{\begin{array}{c} \zeta \\ \rightarrow \\ \alpha \\ \rightarrow \\ \epsilon \\ \rightarrow \\ \beta \end{array}}{\begin{array}{c} \alpha \\ \rightarrow \\ a \\ \rightarrow \\ \beta \end{array}} \begin{array}{c} \alpha \\ \rightarrow \\ a \\ \rightarrow \\ \beta \end{array} \rightarrow \gamma, \quad (3.10)$$

where in the first step the completeness relation was inserted between the representations α and β and in the second step we used the generalized Schur's lemma Eq. (3.7) to simplify the expression between the lines with representations λ and γ . Since the representation γ

might appear several times in the sum over representations $\lambda \in \alpha \otimes \beta$, the delta function appearing from Schur's lemma transforms the sum over representations λ into the sum over a , all the instances of the vertex between α , β and γ . This sum is most often a sum over only one element. An exception is, for example, $8 \otimes 8$ into 8 , where it is a sum over the anti-symmetric and symmetric vertex. Furthermore, it can be shown that for $N_c = 3$ any vertex with 8 has at most two instances [24]. Therefore, whenever at least one of the representations α , β or γ in Eq. (3.10) is 8, the sum over the instances for $N_c = 3$ is at most over two elements. Similarly, it can be shown that if any of these representations is either 3 or $\bar{3}$, the sum in Eq. (3.10) is only over one element.

The numerator on the right-hand side of Eq. (3.10) is another kind of constant called Wigner 6j coefficient. These coefficients are rich in various kinds of symmetries as can be seen by the form of their birdtrack:

1. Since changing the orientation of birdtracks does not affect them, the Wigner 6j coefficients are invariant under rotations by $2\pi/3$ and $4\pi/3$.
2. Mirroring the 6j coefficient is equal to transposing all the vertices and interchanging two of them.

$$\begin{array}{c} \alpha \\ \nearrow \quad \searrow \\ \eta \quad \epsilon \\ \nwarrow \quad \nearrow \\ \beta \quad \delta \quad \gamma \end{array} = \begin{array}{c} -\alpha \\ \nwarrow \quad \nearrow \\ \epsilon \quad \eta \\ \swarrow \quad \searrow \\ -\gamma \quad \delta \quad -\beta \end{array} . \quad (3.11)$$

Since, as explained below Eq. (2.24), transposing the vertex in our conventions gives at most a minus sign, then the transformed 6j differs from the original by at most a minus sign. Similarly, any two vertices can be interchanged by obtaining at most a minus sign. By these two symmetries, that is, rotating and interchanging two vertices, the 6j coefficient can be related to all of the other 6j coefficients consisting of the same four vertices. In other words, these two symmetries correspond to all the 24 symmetries of a tetrahedral graph that preserve the connectivity of the vertices.

3. Each Wigner 6j coefficient can be related to the same coefficient with the vertices conjugated. As was shown in section 2.3.1, the vertex conjugation also brings at most a minus sign in the vertex.

In general, Wigner 3j and 6j coefficients (we will sometimes refer to them as simply 3j and 6j coefficients) are not known and must be calculated. However, the symmetries these coefficients possess make them convenient for group theoretical calculations. Moreover, as we will show in the next section, an important property of 3j and 6j coefficients is that any group theoretical invariant quantity can be decomposed into them. Due to this fact, the procedure for calculating the components of the $SU(N_c)$ color structure on the multiplet bases that the created program is based on, is possible.

3.4 Decomposing color-invariant quantities into the Wigner 3j and 6j coefficients

In this section, different methods for decomposing color-invariant quantities in the form of a vacuum bubble, Eq. (3.5), into Wigner 3j and 6j coefficients are discussed. We begin in section 3.4.1 by showing a universal method, presented in [26] and similar to the one shown in [25] that can be applied to any color singlet. This general method, nevertheless, does not guarantee to obtain a numerical value if the required 3j and 6j coefficients are not known. These coefficients must be calculated in some other ways. An improved method used in the Mathematica program created in this thesis is presented in section 3.4.2. This method puts constraints on the required 6j coefficients and is similar to the method presented in [27]. The main differences between the two methods are explained at the end of section 3.4.2.

3.4.1 An arbitrary structure

The algorithm we present here works by applying the identities introduced in the previous section to remove (contract) all of the loops in birdtracks of scalar quantities that are in the form of vacuum bubbles, such as Eq. (3.5) [25, 26]. We point that writing all the tensors in the birdtrack notation is not necessary, since birdtracks are equivalent to the ordinary tensor notation. However, the loop patterns can often be found simpler in birdtracks. For this reason, we also choose to explain the method using birdtracks.

First, note that Schur's lemma, Eq. (3.8), and the vertex correction relation, Eq. (3.10), can always be applied on structures containing small loops (by small loops in this section we denote loops connecting two or three vertices). Each time applied, these identities remove a small loop by removing two vertices from the color structure and multiply with some constants including Wigner 3j and 6j coefficients. Using these identities, the expression can be simplified until all of the small loops are removed, that is, the smallest loop left connects four or more vertices.

The next step is to reduce the size of any of the loops that are left by displacing one of the vertices out of the loop

$$\begin{aligned}
 & \text{Diagram of a six-vertex loop with two red edges} \\
 &= \sum_{\alpha} \frac{d_{\alpha}}{\text{Diagram of a circle with a horizontal line}} \text{Diagram of a triangle with a red edge} \\
 &= \sum_{\alpha} \frac{d_{\alpha}}{\text{Diagram of a circle with a horizontal line}} \frac{\text{Diagram of a triangle with a red edge}}{\text{Diagram of a circle with a horizontal line}} \text{Diagram of a pentagon with a red edge}
 \end{aligned}
 \tag{3.12}$$

This step can be applied on any loop, so we demonstrate it on a six-vertex loop as shown on the left-hand side in Eq. (3.12). First, the action of the completeness relation, Eq. (3.9), on two edges of the loops splits it into two smaller loops (and turns it into a sum over the representations) where the total number of vertices is increased by two. In any loop that

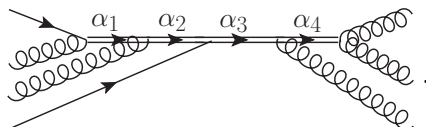
is larger than a two-vertex loop (and only such loops are left after the previous steps), we can always find next-to-neighboring vertices which are marked in Eq. (3.12), in other words, edges that have one more edge in-between. Acting on such edges, Eq. (3.9) splits out one three-vertex loop that can be removed by the vertex correction relation, Eq. (3.10), obtaining the expression after the second equality in Eq. (3.12).

Even though the rightmost expression in Eq. (3.12) contains the same number of vertices as the leftmost, the new expression contains a loop that is one vertex smaller. After repeating the process of splitting off the three-vertex loops and removing it using Eq. (3.10), in the last step one obtains a four-vertex loop. Splitting this loop gives two three-vertex loops that can now both be removed and in this way reduce the total number of vertices by two. Such a process can be repeated on every loop, each time removing two vertices, until either the expression again contains small loops to be removed by Eq. (3.6), Eq. (3.8) and Eq. (3.10) or is fully contracted. So, we conclude that any scalar quantity can be expressed in Wigner $3j$ and $6j$ coefficients using this strategy.

As can be shown, this strategy does not depend on the order of steps made [25, 26, 27]. Also, note that the application of the completeness relation, Eq. (3.9), gives a sum over all the representations in the tensor product of the two representations that it is applied on. This sum, unless one of the representations is a singlet, involves several terms (see, for example, $8 \otimes 8$ in Eq. (2.19)). The removal of the small loops, on the other hand, in the worst case gives a sum over all the instances of a vertex (from Eq. (3.10)) that most often is a sum over only one instance. As a consequence, the computationally most efficient method is to use the completeness relation, Eq. (3.9), as little as possible and remove all of the small loops that have appeared after every step.

3.4.2 QCD amplitude contracted with a basis vector of the form as in figure 1a

We will show how the method presented in the previous section can be changed to put a constraint on the form of the obtained $6j$ coefficients if the decomposition is done for a scalar product between a QCD amplitude and a basis state of the form


(3.13)

Since to form a scalar product each of the external lines of Eq. (3.13) will connect to the external lines of the color structure, no loop containing only the vertices coming from the basis state are possible. In other words, every loop is guaranteed to contain at least one vertex that has only QCD representations (only 3 , $\bar{3}$ and 8). The limiting case is when the loop has only one vertex coming from the initial color structure as depicted in the leftmost

expression in Eq. (3.14),

$$\begin{aligned}
 & \text{Diagram with loop } \alpha, \beta, \gamma \text{ and vertex } \psi \\
 &= \sum_{\psi} \frac{d_{\psi}}{\text{Diagram with loop } \psi, \gamma \text{ and vertex } \alpha} \text{Diagram with loop } \alpha, \psi, \beta, \gamma \text{ and vertex } \gamma \\
 &= \sum_{\psi, a} \frac{d_{\psi}}{\text{Diagram with loop } \psi, \gamma \text{ and vertex } \alpha} \frac{d_{\psi}}{\text{Diagram with loop } a, \psi, a \text{ and vertex } \alpha} \text{Diagram with loop } \alpha, \psi, \beta, \gamma \text{ and vertex } a,
 \end{aligned} \tag{3.14}$$

where the loop can be of an arbitrary length, including a two-vertex loop. The thin lines with no arrows can represent both 3 and 8, while the gray cycle, $\text{triangle with circle}$, can depict the anti-symmetric three-gluon vertex, Eq. (2.8), or the quark-gluon vertex, Eq. (2.7). In what follows, this vertex might also represent the symmetric three-gluon vertex, Eq. (2.20). In this way, we can always choose to insert the completeness relation, Eq. (3.9), between representations where at least one of them belongs to a QCD vertex (vertex only containing 3, $\bar{3}$ and 8) coming from the initial color structure. Following the steps shown in Eq. (3.14), similarly as in Eq. (3.12), we can reduce the size of the loop by one vertex, where the new loop still contains a QCD vertex. Hence, the process can be repeated, each time obtaining a 6j coefficient of the form as in the numerator in the rightmost expression of Eq. (3.14).

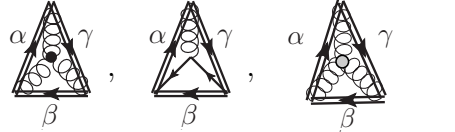
Another kind of 6j coefficient appears if we choose to insert the completeness relation between the non-highlighted thin line and the line with the β representation in the loop on the left-hand side in Eq. (3.14). The three-vertex loop split out then brings a 6j coefficient of a different shape

$$\begin{aligned}
 & \text{Diagram with loop } \alpha, \beta, \psi \text{ and vertex } \alpha \\
 &= \sum_a \frac{\text{Diagram with loop } a, \alpha, \psi \text{ and vertex } \beta}{\text{Diagram with loop } a, \psi, a \text{ and vertex } \alpha} \text{Diagram with loop } a, \psi \text{ and vertex } \alpha.
 \end{aligned} \tag{3.15}$$

By choosing to insert the completeness relation in loops containing at least one QCD vertex, we put a constraint such that the only 6j coefficients that appear in the result are of the form as in Eq. (3.14) and Eq. (3.15). Replacing the thin lines by all the possible combinations of representations 3 and 8 for the 6j coefficient in Eq. (3.14) we obtain

$$\begin{aligned}
 & \text{Diagram with loop } \alpha, \beta, \gamma \text{ and vertex } \delta \\
 & \text{Diagram with loop } \alpha, \beta, \gamma \text{ and vertex } \delta \\
 & \text{Diagram with loop } \alpha, \beta, \gamma \text{ and vertex } \delta,
 \end{aligned} \tag{3.16}$$

where we have excluded the 6j coefficient similar to the second coefficient but octet and triplet representations exchanged since these two 6j coefficients are related by symmetries. Similarly, the second 6j coefficient appearing from the contraction in Eq. (3.15), gives three other distinct types of 6 coefficients


(3.17)

where for clarity we point that the gray vertex appearing in the last 6j coefficient in the symmetric three gluon vertex. Likewise in the process described in section 3.4.1, the full contraction of color scalars includes contracting all the small loops first. When no small loops are present, then the method is to choose one of the loops and reduce its size as depicted Eq. (3.14) until the loop can be contracted using Eq. (3.10). By repeating the whole process of small loop removal and using Eq. (3.14), when no small loops are available, a QCD amplitude can be decomposed into constants and Wigner 6j coefficients.

The representations α , β , γ and δ in Eq. (3.16) and Eq. (3.17) can still be arbitrary. However, as described in [26, 27], constraints on these representations can be put in terms of the so-called first occurrences for each number of external partons in the amplitude and for each order of perturbation expansion in the strong coupling constant. Furthermore, in [26] all the necessary 6j coefficients for decomposing an LO and NLO QCD amplitude with up to six gluons and quark-antiquark pairs in total were calculated for a special case if quark and anti-quark pairs are first combined into the octets and singlets. For the program created in this thesis, we, nevertheless, use the coefficients that are suitable for using the basis shown in Eq. (3.13) and were calculated in [27]. These coefficients are sufficient to decompose amplitudes of up to six gluons and quark-antiquark pairs, but only for LO [27].

The method we presented above, that is, always choosing to insert completeness relation in loops with at least one QCD vertex, is a simplified case to the method presented in [27], where the completeness relation has to be inserted in a loop where exactly one vertex is a QCD vertex and always has to come from the initial color structure. Consequently, our method does not put the same constraints on the representations α , β , γ and δ in Eq. (3.16) and Eq. (3.17). In this way, there could be cases of LO QCD amplitudes with up to six external partons that would require more 6j coefficients than the ones available. Nevertheless, the breaking point of the current computational capabilities of exact computations of the color structure using the ordinary methods is at around eight gluons and for such amplitudes, the required 6j coefficients have not been calculated yet. In this way, the method chosen here does not limit the applications of the program. Moreover, it is often possible to decompose more complicated vacuum bubbles than created by an NLO amplitude with 6 partons, and applying the completeness relation is often unnecessary.

vertices, dividing by the square root of its norm. For example, the absolute square of the triple-gluon vertex is the 3j coefficient below

$$\|if^{abc}\|^2 = \left(\text{triple-gluon vertex} \right)^\dagger \text{triple-gluon vertex} = \text{gluon loop} = 2N_c T_R(N_c^2 - 1) \quad (4.2)$$

whereas, for the generalized three-octet vertex, as mentioned above, we normalize the 3j to identity, that is

$$\text{generalized three-octet vertex} = 1. \quad (4.3)$$

To convert the usual three-gluon vertex to the generalized three-gluon vertex, we divide by the square root of the normalization

$$\text{usual three-gluon vertex} = \frac{1}{\sqrt{2N_c T_R(N_c^2 - 1)}} \text{generalized three-gluon vertex}. \quad (4.4)$$

The advantage of this normalization is avoiding keeping track of the 3j coefficients appearing from almost all of the identities used for the decomposition process, such as Eq. (3.8), Eq. (3.9) and Eq. (3.10). All normalization factors for each vertex can be multiplied into the expression before or after doing the decomposition.

Consequently, the normalization of the multiplet basis states used in the program can be obtained by taking its square and using Eq. (3.8) to one by one remove all the two vertex loops that appear

$$\text{two-vertex loop} = \frac{1}{d_{\alpha_1} d_{\alpha_2} \dots d_{\alpha_n}}, \quad (4.5)$$

where d_α is the dimension of the representation α .

In the program, we adapt a unique and N_c -independent labeling for the $SU(N_c)$ representations, which was used for the 6j coefficients that the program operates with [24, 27]. According to this notation, the representations are labeled by two sets of integers, the lengths of the columns of the quark Young tableaux and the lengths of the columns of the anti-quark Young tableaux of the representation. For simplicity, the representations with a given dimension in the $N_c = 3$ case that appear first in the tensor products of octets and triplets we denote by their dimensions in $N_c = 3$.

In this subsection, we have thus introduced the most essential conventions defined in the program. Other conventions regarding the generalized vertices and the representations can be seen from the supplementary material added to [27], which also contains the 6j coefficients used in the program.

4.2 The implementation

The decomposition of the color-invariants follows the recipe described in section 3.4.2. The Wigner 6j coefficients used in the program are loaded from a separate file that can be updated or replaced by a new file if more 6j coefficients are calculated. In general, the program simplifies not only vacuum bubbles of the form Eq. (3.5) but also any given tensor containing loops (dummy indices) as far as possible with the given 6j coefficients.

As mentioned in section 4.1, the $SU(N_c)$ tensor in the program is given as a product of generalized vertices, Eq. (4.1). In addition, the program contains notation for delta function lines (propagators)

$$\begin{array}{c} i_1 \\ \xrightarrow{r} \\ i_2 \end{array} \equiv \delta_{\{r\}}^{\{i_1, i_2\}}, \quad (4.6)$$

where r is defined to point from i_1 to i_2 .

We denote C to represent a tensor product of generalized vertices, Eq. (4.1), and deltas, Eq. (4.6). The information about how the vertex or delta is connected to the other vertices and deltas in C is given in the color indices (indices i_j in Eq. (4.1) and Eq. (4.6), where $j = 1, \dots$). Specifically, if the index i_j appears only once in C , then it is a free index, while if it appears twice, then the two vertices containing this index are connected. An index appearing more than twice in C is not permitted and the program, by the procedure described in section 5.6, produces a warning when such a color structure is defined or simplified. For implementing the identities Eq. (3.8) to Eq. (3.10) and delta functions, we use the pattern search built into Mathematica to see if C contains vertices that are connected according to the corresponding pattern. For simplifying deltas it, for instance, corresponds to finding an index appearing twice in C , where at least one of the repeated indices belongs to the generalized delta. In the example below

$$\begin{array}{c} r_1, i_1 \\ \swarrow \\ r_2, i_2 \end{array} \begin{array}{c} \swarrow \\ \searrow \\ a \end{array} \begin{array}{c} r_3, i_3 \\ \xrightarrow{r_3, i_4} \\ r_3, i_4 \end{array} \equiv V_{a, \{r_1, r_2, r_3\}}^{\{i_1, i_2, i_3\}} \delta_{\{r_3\}}^{\{i_3, i_4\}} \quad (4.7)$$

it would thus correspond to finding the index i_3 that is repeated twice in the expression and belongs to one delta. Similarly, for Schur's lemma, Eq. (3.8), it involves finding two vertices that contain two common indices and for the vertex correction relation — finding three vertices where each two have an index in common. When the pattern corresponding to the left-hand side of an identity is found, the function replaces the found vertices with the new vertices and constants corresponding to the right-hand side of the identity.

The implementation of the completeness relation, Eq. (3.9), according to Eq. (3.14) involves finding a loop of at least four vertices, where at least one vertex contains only representations from $\{3, \bar{3}, 8\}$ (is a QCD vertex). The loop search exploits the function **FindCycle** built into Mathematica, which searches for cycles in the graph. First, the information about connections between vertices, which is given by the color indices in the vertices, is converted into a Mathematica graph object with generalized vertices as graph vertices. Then the loop search is performed using **FindCycle** and, finally, the loops found in terms of the graph edges are converted back into the generalized vertices.

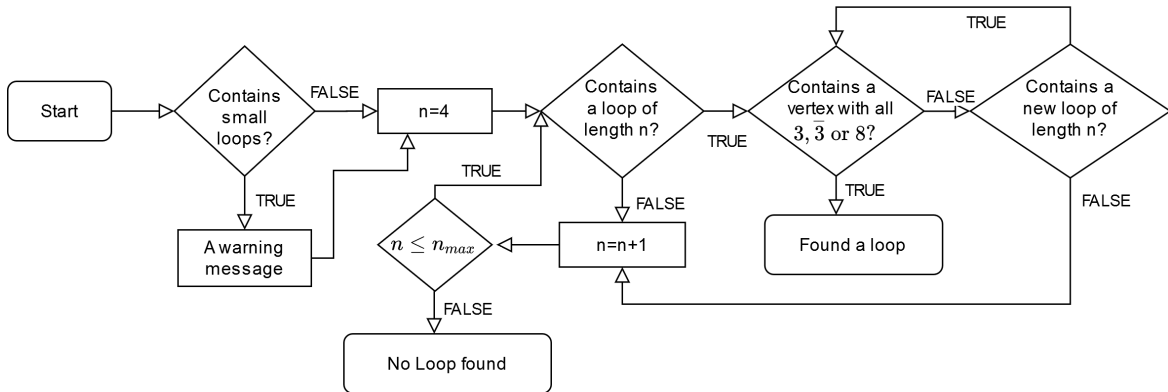


Figure 2: The flow chart of the loop search algorithm. Given an input color structure it finds a loop, where the completeness relation, Eq. (3.9), can be applied ensuring that only the available $6j$ coefficients are used. After a verification that no small loops (connecting up to three vertices) are found, the algorithm enters an iterative search where for every loop length n it searches for the first loop containing vertex with all of its representations belonging to $\{3, \bar{3}, 8\}$.

The flow chart of the loop search in the program is shown in figure 2. First, since according to the method described in section 3.4.2, the completeness relation should be inserted only when no small loops (loops containing up to three vertices) are found, the algorithm involves an additional check to verify that no small loops are left before searching for larger loops. If a small loop is found, it prints an error message. After that, the algorithm enters an iterative process. Starting with the loop length $n = 4$, it searches through all the loops found in the structure to be simplified and for every new loop found, tests if any of the vertices is a QCD vertex. If such a loop is found, the algorithm outputs a list of the color indices of the vertices in the loop. If no such loop is found, then n increases by one and the loop search with length n is repeated until either an appropriate loop is found or until n reaches n_{max} . The default value of n_{max} is 6 as it is the maximal possible loop length appearing in color structures with a maximum of six partons but this value can be changed by adding an additional argument in the function performing the loop search (see **LoopForCR** in section 5.3).

Together with the color indices of the vertices in the loop, the algorithm outputs two first found color indices of the lines where the completeness relation can be applied. The first of the indices belongs to a QCD vertex and the second one is chosen to be the index of one of the next-to neighboring lines in the loop. Additionally, if any of the representations in the loop belonging to QCD vertices is either 3 or $\bar{3}$, this representation is chosen as the first one since any direct product with these representations usually contains fewer terms than with 8. This can be seen, for example, from the Young tableaux of these representations [25]. Compare also, for instance, Eq. (2.5) and Eq. (2.19).

When one required loop is found, the completeness relation, Eq. (3.9), is applied which adds two new vertices to C and replaces the indices of the connected vertices accordingly.

If no such loop is found, then either the final result is obtained or the given structure cannot be simplified anymore with the available 6j coefficients.

5 Usage of the program and examples

In this section, we describe the usage of the created program. The program itself is available upon request. Throughout the chapter we present examples that can simply be performed and adapted in the program while reading the chapter. The main functions and replacement rules are listed in tables 1-4, while other functions that can be useful, but are not mentioned in the text are found in appendix B and appendix C. We begin this section by showing how $SU(N_c)$ tensors are defined and visualized in the program, which includes tools for drawing birdtracks from the defined tensors. Afterwards, we explain the usage of the simplest functions performing tensor index contractions using the identities that were introduced in section 3.3. The main function of the program, **TheGreatSimplifier**, performing full color singlet decomposition algorithm given in section 3.4.2 is introduced in section 5.3. Creation and color structure decomposition in the multiplet bases is discussed in section 5.5. Subsequently we introduce additional tools implemented in the program for simpler use, discuss the usage of specific number of colors N_c and the validation of the program we made.

5.1 The building blocks of the program

The building blocks of the program, the generalized vertices and the generalized delta functions, were defined in Eq. (2.18) and Eq. (4.6) respectively. Table 1 shows how these objects are accessed and defined in Mathematica. The simplest way of defining the input is in the function form (see, table 1) that automatically converts into the Mathematica **StandardForm**. Another way of input is copying other expressions in **StandardForm** and changing the corresponding indices. This form has the advantage of resembling the way tensors are written on paper. Finally the Mathematica **FullForm** can be used that also converts into the **StandardForm** automatically.

The $SU(N_c)$ representations in the program are denoted by strings " r ", for example, triplet has to be inserted as "3" and anti-triplet as "3bar". However, in the objects given in table 1, representations r_i can be written in a often more convenient way, namely, as integers representing their dimensions at $N_c = 3$. The corresponding anti-representations are then denoted by negative integers. In this way, the triplet can be written as 3 and anti-triplet as -3. When defined in the input, the representations given as integers are converted to strings automatically. For the representations denoted using the generalized notation described in section 4.1 no shorthand notation is created, so c_2c_2 , for example, can only be defined as "c2c2". If any r in the objects from table 1 is a symbol, it will be interpreted

Pictorial representation	Mathematica StandardForm	Function form	Mathematica FullForm
	$\mathbf{V}_{\mathbf{a},\{\mathbf{r1},\mathbf{r2},\mathbf{r3}\}}^{\{\mathbf{i1},\mathbf{i2},\mathbf{i3}\}}$ $\delta_{\{\mathbf{r}\}}^{\{\mathbf{i1},\mathbf{i2}\}}$	$\mathbf{V}[\{\mathbf{r1}, \mathbf{r2}, \mathbf{r3}\}, \mathbf{a}, \{\mathbf{i1}, \mathbf{i2}, \mathbf{i3}\}]$ $\mathbf{delta}[\mathbf{r}, \{\mathbf{i1}, \mathbf{i2}\}]$	$\mathbf{Subscript}[\mathbf{Superscript}[\mathbf{V}, \mathbf{a}, \{\mathbf{r1}, \mathbf{r2}, \mathbf{r3}\}], \{\mathbf{i1}, \mathbf{i2}, \mathbf{i3}\}]$ $\mathbf{Superscript}[\mathbf{Subscript}[\mathbf{Delta}, \{\mathbf{r}\}], \{\mathbf{i1}, \mathbf{i2}\}]$

Table 1: Different input forms of the building blocks of the program — the generalized vertices (Eq. (4.1)) and the generalized deltas Eq. (4.6). The simplest approach of defining input is the Function form that automatically transforms into the **StandardForm**. The expressions in the **StandardForm** have the depicted Mathematica **FullForm**. From the **FullForm** it can be seen that objects in the program are written using **Superscript** instead of **Power**. Since the program uses pattern matching to do simplifications on expressions, using **Power** is not a valid input.

as an arbitrary representation. A totally or partially mixed notation is permitted, as can be verified in the program by

```
In[1]:= V[{r, "3", -3}, {i1, i2, i3}] == V[{r, 3, -3}, 1, {i1, i2, i3}] == V1, {r, "3", "3bar"}^{i1, i2, i3}
Out[1]= True
```

QCD amplitudes are conveniently defined using the notation we adapted from *ColorMath*, a Mathematica package for $SU(N_c)$ color summed calculations [18]. The set of available *ColorMath* input is given in table 1 in [18], which we add in appendix A. Besides all the QCD vertices and delta functions, the *ColorMath* notation involves quark loops with attached gluons $\mathbf{o}^{\{g1, g2, \dots, gn\}}$, and quark lines with attached gluons $\mathbf{t}^{\{g1, g2, \dots, g3\} q1 q2}$. Conversion to the generalized vertices is done by the replacement rules **ColorToMultiplet** as, for instance, an NLO diagram for $g \rightarrow q_1 q_2$ via loop of gluons and quarks can be written as

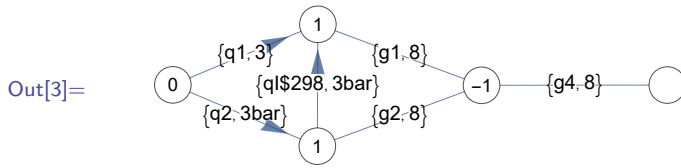
```
In[2]:= t[{g1, g2}, q1, q2] I f[g1, g4, g2] /. ColorToMultiplet
Out[2]= 
$$\frac{\sqrt{2} (N_c (-1 + N_c^2) \text{TR})^{3/2} V_{1, \{8, 3, 3\bar{a}\}}^{\{g1, q1, qI\$2998\}} V_{1, \{8, 3, 3\bar{a}\}}^{\{g2, qI\$2998, q2\}} V_{1, \{8, 8, 8\}}^{\{g1, g4, g2\}}}{N_c}$$

```

where $qI\$2998$ is an automatically generated unique quark index.

As seen above, the generalized vertex notation becomes complicated and unpleasant to read when there are more than few vertices. An alternative is to use **DrawBirdtrack** that uses the built-in **Graph** function to draw a birdtrack of the given expression

```
In[3]:= t[{g1, g2}, q1, q2] I f[g1, g4, g2] delta[q2, q1] /. ColorToMultiplet // DrawBirdtrack
```



Here circles with integers represent vertices with vertex instances and the circles with no numbers represent exiting or incoming lines. Furthermore, transposed vertices (vertices where the indices are ordered anti-clockwise instead of clockwise) are represented by negative vertex instances and a zero instance represents a delta function, that is vertices connecting only two lines. The numbers of the edges represent the color indices and the representations respectively. The directed representations, that is, representations that are not equal to their conjugates are depicted by arrows pointing as defined below Eq. (4.1), otherwise there is no arrow on the lines.

In the cases when the expression contains pairs of vertices that are connected by more than one line and one of the vertices in the pair is transposed, **DrawBirdtrack** might fail displaying which of the two it is. This, nevertheless, never impacts the sign of the expression. Such pairs of vertices connected by two or three lines correspond to the two-vertex loops on the left-hand side of Schur's lemma, Eq. (3.8), and the $3j$ coefficients. It was shown for the special case of $3j$ coefficient in Eq. (2.21), that the value of a $3j$ coefficient is the same if both vertices are transposed. Similarly, two vertices in any $3j$ coefficient or two-vertex loop can be transposed, without changing their value. Another way to see it is that as a consequence of Schur's lemma, the only non-zero case when two vertices can be connected by two or three lines is when the vertices are Hermitian conjugates of each other. This also means that both vertices obtain the same sign under transposition and thus determining the wrong vertex that is transposed does not change the result. In a similar way, if both of the vertices are transposed, **DrawBirdtrack** does not display it. Finally, **DrawBirdtrack** does not show the constant in front of the expression, only the tensorial structure of the expression. When the expression is a sum of terms, only the first one of the terms is displayed.

Since the output of **DrawBirdtrack** is a **Graph** object, it is not a valid input for any of the simplifying functions. However, as a graph it can be modified using the built-in graph visualizing and analysis functions, e.g., emphasizing parts of it, finding cycles. Through the second optional argument **Options**, **DrawBirdtracks**[**Expr**,**Options**] accepts all the arguments of the Mathematica **Graph** function.

5.2 Color structure simplification

The color structure simplification in the program can be done using the replacement rules given in table 2. The application of replacement rules on an expression in Mathematica is done by **Expr**/.**TheRule** for applying a rule **TheRule** once or **Expr**//.**TheRule** for applying **TheRule** repeatedly until the expression does not change. As an example, the Schur's Lemma, Eq. (3.8), is accessed by the rules **SchursLemma** as

In[4]:= V[{8,8,8},{i1,i2,i3}] V[{8,8,8},{i2,i3,i4}]/. SchursLemma

$$\text{Out[4]= } -\frac{\delta_{\{8\}^{\{i1,i4\}}}{-1+Nc^2}$$

where the minus sign appears because one of the vertices has to be transposed to obtain Eq. (3.8) and transposing brings a minus sign in the three-gluon vertex (see section 2.3).

Replacement rule	Effect
SchursLemma	Applies Schur's lemma, Eq. (3.8).
VertexCorrectionRel	Applies vertex correction relation, Eq. (3.10).
DeltaRules	Simplifies expressions containing deltas, e.g, replacing a dummy index or dimensionality, Eq. (3.6).
InsertCompletenessHere	InsertCompletenessHere [{i1,i2},Loop] is a more flexible alternative to CompletenessRelation (see table 4), which applies completeness relation between the lines with indices i1 and i2 for vertices in the loop Loop. Loop has to be specified according to the output of FindLoop and LoopForCR . The case if i1 and i2 belong to the same vertex is not included in InsertCompletenessHere , since it is not required in any of the simplification algorithms.
SimplifyDelayedTables	Simplifies sums, given in terms of the built-in Table function, over the vertices that are created when the simplification rules given in this table are applied on expressions containing arbitrary representations.

Table 2: Replacement rules performing simplifications on expressions. Given that all the necessary Wigner 6j coefficients are available, an expression can be simplified by continuously applying these. All the rules can be applied on vertices with arbitrary representations, but **InsertCompletenessHere** must be applied between lines where at least one of the representations is either of $\{3,\bar{3},8\}$. In the arbitrary case, sums over vertices given in terms of **Table** do not simplify automatically and have to be simplified using **SimplifyDelayedTables**.

Vertex correction relation, Eq. (3.10), is accessed via the replacement rules **VertexCorrectionRel**

In[5]:= o^{g1,g2,g3}/. ColorToMultiplet/. VertexCorrectionRel//Simplify

$$\text{Out[5]= } \frac{\text{TR} \sqrt{(-1 + Nc^2)} \text{TR} \left(\sqrt{Nc} V_{1,\{8,8,8\}}^{\{g1,g2,g3\}} + \sqrt{-\frac{4}{Nc} + Nc} V_{2,\{8,8,8\}}^{\{g1,g2,g3\}} \right)}{\sqrt{2}}$$

The completeness relation, Eq. (3.9), is simplest applied using **CompletenessRelation**[Expr] (see table 4), that inserts it in the first appropriate loop found, that is, the first loop

Function	Usage
LoopForCR	$\{\{i1,i2\}, Cycle\} = \text{LoopForCR}[\text{Expr}, \text{MaxLoopLen}]$ finds a loop in the expression Expr according to the flow chart in figure 2. The maximal loop length, MaxLoopLen has the default value MaxLoopLen = 6. The first output $\{i1,i2\}$ is a list of two next-to neighboring line indices in the loop, where at least i1 is one of $\{3, \bar{3}, 8\}$. Cycle , a list of all the indices of vertices that make the loop found.
FindLoop	FindLoop [Expr , k] finds a loop in expression Expr . FindLoop [Expr , k] finds loop of a length at most k . FindLoop [Expr , $\{k_{max}\}$] finds a loop of exactly length k_{max} FindLoop [Expr , $\{k_{min}, k_{max}\}$] finds a loop of a length between k_{min} and k_{max} FindLoop [Expr , karg , s] finds at most s loops with loop length specifications karg , according to the cases given above. FindLoop [$\{\text{Expr}, \text{ind}\}, \dots$] finds a loop that includes a line with the index ind

Table 3: Functions for finding loops in the expression with generalized vertices.

containing at least one QCD vertex. If the completeness relation needs to be inserted at a specific place in a specific loop, the rule **InsertCompletenessHere**[$\{i1,i2\}, \text{Loop}$] can be used. It requires to specify the two indices **i1** and **i2** of the lines where the completeness relation should be inserted, as well as **Loop**, the list of the indices of the vertices in the loop. **Loop** is in the form $\{I_1, I_2, \dots, I_n\}$, where I_i is a list of the three indices in the i -th vertex in the loop.

Loops can be found using **FindLoop** and **LoopForCR**. The former finds a loop in the expression but does not verify if the completeness relation can be inserted in the loop. To help finding an appropriate loop, **FindLoop** accepts several optional arguments listed in table 3. **LoopForCR**, on the other hand, finds the first loop where the completeness relation can be inserted.

Since applying **VertexCorrectionRel** and **InsertCompletenessHere** can insert sums in the expression, a repeated use of these rules requires expanding the expression after each time applying them. However, using the built-in **Expand** can significantly increase the number of terms and slow down the simplification process, since it expands also the brackets that do not contain any vertices. This can be avoided by using **ExpandVertices** instead, which expands only the brackets containing vertices or deltas. **RemoveAllTriangles** is a function that simplifies all of the three-vertex loops found in the expression. Additionally, for expressions containing lengthy sums, it automatically finds the terms of the sum with common vertices and simplifies them simultaneously, which makes it a faster alternative to `//VertexCorrectionRel`.

5.3 Full color structure calculation

The main function of this thesis is **TheGreatSimplifier**[**Expr**] that, whenever the 6j coefficients appearing in the results of calculations are available, simplifies the expression **Expr**

Function	Usage
TheGreatSimplifier	TheGreatSimplifier [Expr] performs the full loop removal algorithm according to the method described in section 3.4.2.
CompletenessRelation	CompletenessRelation [Expr , MaxLoopLen] inserts a completeness relation, Eq. (3.9), in the first found loop where it is possible to insert it. The loop search is done until the maximal loop length, MaxLoopLen , which has the default value MaxLoopLen = 6.
RemoveAllTriangles	RemoveAllTriangles [Expr] contracts all the three-vertex loops in Expr .
ConjugateRep	ConjugateRep [Rep] conjugates the representation Rep . If Rep is a symbol then ConjugateRep does not evaluate the expression until it is substituted by a string.
ExpandVertices	ExpandVertices [Expr] expands only the brackets containing generalized vertices or deltas in Expr .
DrawBirdtrack	DrawBirdtrack [Expr , { Options }] draws a graph of the form of a birdtrack from expression Expr . The optional argument Options accepts all the arguments from the built-in Graph function.

Table 4: Helpful functions for simplifying and visualizing the expressions.

by contracting all of the loops in it. If **Expr** has no free indices, then **TheGreatSimplifier** outputs a fully contracted expression, that is, containing no generalized vertices. **TheGreatSimplifier** unites the replacement rules defined in the previous section and implements the algorithm described in section 4.2.

As an example consider $q_1\bar{q}_2 \rightarrow q_3\bar{q}_4$ via gluon exchange with two channels, s- and t-channel,

$$\text{In[6]:= } \text{Ampl} = \text{S } t^{\{g\}q^1}_{q_2} t^{\{g\}q^4}_{q_3} + \text{T } t^{\{g\}q^1}_{q_3} t^{\{g\}q^4}_{q_2} /. \text{ColorToMultiplet}; \quad .$$

The absolute square of an amplitude is calculated by multiplying the amplitude with its conjugate. It is then evaluated by applying **TheGreatSimplifier**

$$\text{In[7]:= } \text{TheGreatSimplifier}[\text{Conjugate}[\text{Ampl}] \text{ ReplaceDummyIndices}[\text{Ampl}]]$$

$$\text{Out[7]= } \frac{(-1 + Nc^2) \text{TR}^2 ((Nc \text{S} - \text{T}) \text{S}^* + (-\text{S} + Nc \text{T}) \text{T}^*)}{Nc}$$

where **ReplaceDummyIndices** has to be applied to any of the two terms to replace the dummy indices of the expression with unique symbols and thus make sure that no dummy index would appear more than twice in the expression. The combined action of **Conjugate** and **ReplaceDummyIndices** is united in one function, **ScalarProduct**, so the scalar product above can be written simpler as **ScalarProduct**[**Ampl**,**Ampl**].

We can generalize the amplitude above and substitute the triplets and the octets by arbitrary representations. For example, the same quark process but with an unspecified mediator a in between them is written as


```
In[8]:= Amp12 = TR (-1 + Nc^2) (T V_{1,\{a,3,-3\}}^{\{g,q1,q3\}} V_{1,\{ConjugateRep[a],3,-3\}}^{\{g,q4,q2\}} +
S V_{1,\{a,3,-3\}}^{\{g,q1,q2\}} V_{1,\{ConjugateRep[a],3,-3\}}^{\{g,q4,q3\}})
```

For simplification of the vertices with arbitrary representations the extension of **TheGreatSimplifier** has to be used, namely **TheGreatSimplifierArb**. The result is thus obtained by

```
In[9]:= Result = TheGreatSimplifierArb[ScalarProduct[Amp12, Amp12]];
```

In this way, we calculate several color structures simultaneously — for each representation a . Because $3 \otimes \bar{3} = 1 \oplus 8$, there are only two vertices connecting 3 and $\bar{3}$, that is, 1 and 8. For all the other representations a , the result at `In[9]` is zero. The value of the absolute square for a specific case can be obtained by substituting a in **Result** by specific representations

```
In[10]:= {Result/. a -> "1", Result/. a -> "3", Result/. a -> "8"}//Simplify

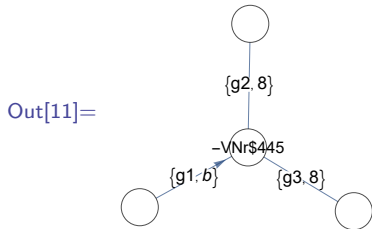
Out[10]= {
  (-1 + Nc^2)^2 TR^2 ((Nc S + T) Conjugate[S] + (S + Nc T) Conjugate[T]) / Nc,
  0, (-1 + Nc^2) TR^2 ((Nc S - T) Conjugate[S] + (-S + Nc T) Conjugate[T]) / Nc }
```

where for $a = 8$ we obtain the same result as calculated in `Out[7]` without using arbitrary representations.

5.4 Simplifying amplitudes containing arbitrary representations

Unless stated otherwise, all of the rules and functions in the program can be applied to expressions containing arbitrary representations. The result of such expressions is not always human-readable, but it can be drawn using **DrawBirdtrack**. Below we show, for example, how a three-vertex loop is removed from an object containing two unspecified representations, a and b ,

```
In[11]:= V_{1,\{b,3,3bar\}}^{\{g1,q1,q2\}} V_{1,\{8,3,ConjugateRep[a]\}}^{\{g2,q2,q3\}} \times
V_{1,\{8,a,3bar\}}^{\{g3,q3,q1\}}//. VertexCorrectionRel//DrawBirdtrack
```



where the arbitrary vertex instance appears because the number of instances of the vertex with representations $\{8, 8, b\}$ is unknown before a specific representation b is supplied. This shows that the sums over the vertex instances and representations appearing from **VertexCorrectionRel** and **CompletenessRelation** are left non-evaluated in cases when

the terms of the sum are not known. If an expression containing non-evaluated sums has to be simplified further the terms inside these sums have to be expanded. However, the usual **ExpandVertices** does not recognize such non-evaluated sums and the replacement rules **SimplifyDelayedTables** have to be used instead.

For a similar reason, **TheGreatSimplifier** cannot be used to simplify expressions containing arbitrary representations. However, it is particularly optimized for dealing with lengthy sums in the expressions. Just like **RemoveAllTriangles** it simplifies terms of these sums containing common vertices simultaneously and is thus a faster alternative to **TheGreatSimplifierArb**, when all of the representations in the vertices are specified.

5.5 Evaluating amplitude squares using multiplet bases

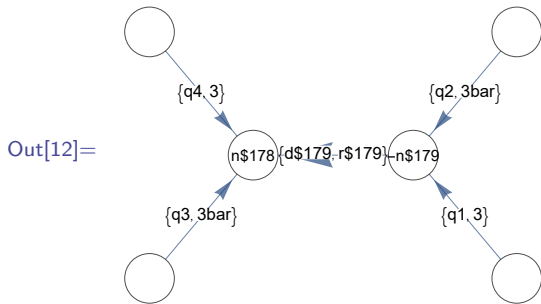
For any amplitude having external lines with only QCD representations, we can create the corresponding multiplet basis, $\{\mathbf{Basis}, \mathbf{RepKeys}\} = \mathbf{CreateMultipletBasisStates}[\mathbf{Ampl}]$. In the output, **Basis** is a basis of the form


(5.1)

described in section 3.2. The representations on its backbone in **Basis** are unique symbols and can be replaced by any representations. The set of all the allowed (non-vanishing) combinations of the representations on the backbone is given in the replacement rules **RepKeys**. Specifically, we can obtain a list of all of the possible basis states if for each key in **RepKeys** we apply this key on **Basis** as $\mathbf{Basis}/.\# \&/\@ \mathbf{RepKeys}$. However, if the representations on the backbone of **Basis** are kept arbitrary, the contractions only have to be done once. When the simplification process is completed, the representations can be inserted in the expression.

As an example, for the amplitude defined in `In[6]` we obtain

```
In[12]:= {Basis, RepKeys} = CreateMultipletBasisStates[Ampl];
Basis//DrawBirdtrack
RepKeys
```



```
Out[13]= {{r$179 -> "1", n$178 -> 1, n$179 -> 1}, {r$179 -> "8", n$178 -> 1, n$179 -> 1}}
```

The indices `r$179`, `n$178`, `n$179` and `d$179` are unique symbols representing an arbitrary representation, two arbitrary vertex instances and a unique color index respectively. In other words, **RepKeys** tell that the representation `r$179` in `Out[12]` can be either "1" or "8" and in both cases the only possible vertex instances `n$178` and `n$179` are 1 and 1 respectively. The scalar product between the amplitude and the basis vector can be simplified without the knowledge about what representations are possible on the backbone of the basis vector

```
In[14]:= SimplifiedScalarProduct = TheGreatSimplifierArb[ScalarProduct[Ampl, Basis]]
```

After that, the keys for the representations and vertex instances can be inserted in the simplified expression, obtaining a list of components of the amplitude on each of the basis vectors

```
In[15]:= Components = SimplifiedScalarProduct /. # & /@ RepKeys
```

The basis states created are normalized. Hence, to obtain the scalar product of the amplitude, a sum of the absolute squares of the components has to be taken

```
In[16]:= Simplify[ReleaseHold[Total[Components Conjugate[Components]]]]
Out[16]= 
$$\frac{(-1 + Nc^2) TR^2 ((Nc S - T) Conjugate[S] + (-S + Nc T) Conjugate[T])}{Nc}$$

```

were we obtain the same expression as we got in section 5.3 by taking the absolute square of the amplitude directly.

All the steps from creating the multiplet basis states up to calculating the amplitude square are included in `AmplitudeSquare[Expr, {Basis, RepKeys}, NcVal]`, where `{Basis, RepKeys}` and `NcVal` are optional arguments specifying the basis and the number of colors respectively. If `{Basis, RepKeys}` is not specified, a new basis is created.

5.6 The built-in consistency and validity checks

The great number of indices and different representations in the tensor expressions make the process of defining color structures in the program susceptible to errors. For this reason, several measures were created that warn the user when at some point an invalid input is given. Here we shortly present some of them.

Attempting to define objects with representations that are not recognized in the program will create an error message and not output any result. Defining one of the representations in the vertex to be 25, which does not exist, will, for example, cause such a message

```
In[17]:= V[{8,8,25},{i1,i2,i3}]
```


V:One or more of the representations {8,8,25} is not recognized in the system.

All of the replacement rules, except **InsertCompletenessHere**, contain an index control. It sees if in the expression to be simplified, the same dummy indices correspond to representations that are conjugated of each other. Instead of adding this check to **InsertCompletenessHere**, it is implemented in the loop search in **LoopForCR**, as well as in **DrawBirdtrack**, making it a good tool not only for visualizing the results but also for searching for the errors in the input.

These input tests include also arbitrary representations in the expressions, even if inconsistencies appear only later in the simplification process

```
In[18]:= Obj=V[{8,ConjugateRep[a],8},{i1,i2,i3}] V[{a,8,8},{i2,i3,i4}]
```

```
In[19]:= Obj /. SchursLemma //DrawBirdtrack
```

```
Out[19]= 
```

```
In[20]:= Obj /. SchursLemma /. a->"25"
```

NVertHold: The vertex with representations {8,8,25} is not known.

Here we see that, since result of Schur's lemma, Eq. (3.8), does not depend on the representation a in the loop, the result simplifies to delta regardless on what is the value of a . However the result still contains a condition, that outputs a warning if any of the initial vertices did not exist.

5.7 Treatment of the number of colors N_c

As argued in section 2.2, all the treatment of color structures done in the program is independent of N_c , which only appears as a parameter in the coefficients like Wigner 6j coefficients and representation dimensions as well as determines what representations are possible for the given N_c . The dimensions of the representations used in the created program for the general N_c case were calculated in [27] using the Young tableaux [25]. As an artifact of the calculation procedure, the non-present representations for a given N_c case appear to have non-positive dimensions. This gives a possibility to distinguish such non-existent representations and to remove them. As explained below, terms containing zero-dimensional representations vanish when calculating the components of a color structure on a multiplet basis. However, the negative dimensional representations have to be removed manually when choosing a particular value of N_c . In addition, some of the representations appearing repeatedly in the tensor products of representations, such as the two 8 in $8 \otimes 8$ might appear only once at low enough N_c and have to be taken out manually as well.

The program always performs simplifications assuming that all the representations are possible, that is, $N_c \rightarrow \infty$. In this way, applying the completeness relation, Eq. (3.9), and vertex correction relation, Eq. (3.10), or creation of the multiplet basis states, bring representations that are not present at low enough N_c . We denote one of the vanishing

representations at some N_c by α and its dimension by d_α . Since the normalization of the basis states is proportional to $\sqrt{d_\alpha}$ (see Eq. (4.5)), the scalar product between a QCD amplitude containing only 3, $\bar{3}$ and 8 and a multiplet basis state containing α always vanishes in the case $d_\alpha = 0$. Even though from applying Schur's lemma, Eq. (3.8), d_α might appear in the denominator in front of the expression, for Schur's lemma to be applied, the expression has to contain at least two representations α . The second representations α can appear in the expression only by applying the completeness relation, Eq. (3.9), which has d_α in the numerator. In this way, the d_α in the numerator and denominator cancel each other and the obtained component on the basis vector stays zero. However, it might happen that the built-in **Simplify** does not find and cancel the two d_α , e.g when one of them is conjugated and the calculation might encounter an infinity

```
In[21]:= Obj = f[g1, g2, g3] f[g3, g4, g9] f[g5, g4, g6] f[g6, g7, g8] /.ColorToMultiplet;
```

```
In[22]:= AmplitudeSquare[Obj] /. Nc -> 3;
```

Power:Infinite expression $\frac{1}{\sqrt{0}}$ encountered.

In the calculation above, **Simplify** is used at the end of **AmplitudeSquare**. To avoid non-canceled divergences, if the program is applied for calculations at a specific N_c , it is recommended either to use **AmplitudeSquare** specifying the number of colors in the last argument as **AmplitudeSquare**[Expr, NcVall] or to define N_c in the beginning of the calculation as

```
In[23]:= Nc = 3;
```

```
In[24]:= Obj = f[g1, g2, g3] f[g3, g4, g9] f[g5, g4, g6] f[g6, g7, g8] /.ColorToMultiplet;
```

```
In[25]:= AmplitudeSquare[Obj]
```

```
Out[25]= 10368 TR4
```

The arbitrariness of N_c can be restored at the end of the calculation by **Clear**[Nc]. After that all of the replacement rules and function will work as if N_c is arbitrary again.

If at the given N_c , representations with negative dimensions appear, they have to be removed manually by choosing only appearing basis states. At the time of publication of this thesis, the program offers no treatment for such representations.

5.8 Validation

In this section, we explain the validation and consistency tests that have been performed on the program and the results they gave.

As seen from the algorithm described in section 3.4.2, the order of contractions, that is, in what order and on which loops Eq. (3.8), Eq. (3.9) and Eq. (3.10) are applied may lead to

6 Conclusion and Outlook

In this thesis, a program in Mathematica performing color structure decomposition in the multiplet bases was presented. In this way, the multiplet bases were demonstrated to be a possible alternative to the non-orthogonal bases commonly used for the color structure calculations in particle physics event generators. The usage of orthogonal bases, such as the multiplet bases, could significantly speed up the color structure calculations since squaring the color structure in these bases does not require calculating the cross terms between different basis states. In this way, multiplet bases is a promising option for exceeding the current limit of approximately eight external gluons for the exact color sum evaluation available in the common event generators. However, as explained below, using the program as a fully-fledged alternative to the current color structure calculation algorithms used in particle physics event generators requires an optimization of the program and extending the list of available Wigner 6j coefficients. The program is available upon request to the author of the thesis and a publication of it in a package form is possible. Even though the thesis did not focus on the recursive methods of color structure calculations, multiplet bases are applicable there as well.

For a given QCD amplitude, the program can construct the corresponding multiplet basis states and obtain the components of the amplitude in this basis. Moreover, the program offers a variety of tools for manipulating the color-summed tensor expressions such as automatic contraction of loops over the tensor indices, conjugation and scalar product for tensors with arbitrary representations of $SU(N_c)$. A simple input form that resembles the way tensors are written on paper is available, as well as options for drawing tensors in the birdtrack notation. A list of the available replacement rules is given in table 2 while the main functions are summarized in tables 3 and 4. Even though inconsistencies in the calculations have been found when performing the operations in different order, a comparison with calculations done on paper indicates that the error may be caused by wrong signs in some of the Wigner 6j coefficients we use.

The program will succeed in completing the tensor contraction whenever the required Wigner 6j coefficients are available. However, the list of previously calculated 6j coefficients is not sufficient to guarantee the amplitude decomposition of more than six gluons and quark-antiquark pairs in total and of orders higher than the NLO in the strong coupling constant. For the multiplet bases to be convenient for usage in particle physics event generators the limit where application of the non-orthogonal bases breaks down has to be exceeded, that is, as mentioned before, around eight gluons. In addition, decomposition of up to NNLO has to be ensured. As a solution to this challenge, the functions available in the program performing tensor index contraction can be used to calculate the 6j coefficients required for extending the scope where the multiplet bases can be applied. This can be done by, for instance, evaluating the 6j coefficients from the projection operators as shown in [26, 27]. Moreover, as was seen in section 3.4.2 and section 5.2, different orders of index contractions on the color singlets lead to expressions containing different combinations of

6j coefficients. This suggests that a recursive method for computing the 6j coefficients from the already known ones might be possible. If such a method is found, the program would be a possible tool to perform the required calculations.

Another application of the program is $SU(N_c)$ tensor calculations at any N_c . This includes working with $SU(2)$ generators, that is, Pauli matrices, and $SU(2)$ structure constants. Using the notation adapted from Mathematica package *ColorMath* (see table A), these can be defined as their $SU(3)$ counterparts, i.e. t^a and if^{abc} matrices. The other $SU(2)$ tensors can be defined using the generalized vertices. Dealing with $SU(N_c)$ with a high N_c , encountered in, for instance, beyond the standard model theories might also be a field where the program could be applied. Since the program operates with arbitrary representations of $SU(N_c)$, it can be viewed as an extension of *ColorMath*, where only calculations with the defining representation and the adjoint representation of $SU(N_c)$ are supported [18]. Furthermore, the input of the created program is compatible with the *ColorMath* input (see section 5.1) and thus allows combining both programs for computations. An example of this would be first applying our program to contract such QCD tensors that can be decomposed using the known 6j coefficients. Afterwards, the tensors left could be simplified using the additional QCD tensor identities and index contraction tools available in *ColorMath*. Furthermore, as our program permits converting *ColorMath* objects into the generalized vertices, the options for visualizing tensor expressions in birdtracks and finding loops in them would be useful tools also in *ColorMath*. A merge between the two programs is a potential option for future development and would simplify their combined application.

Since the aim of this thesis was a computational demonstration of the color structure decomposition in the multiplet bases rather than creating a high-speed alternative in the common color structure decomposition programs such as [14, 19, 22], our implementation of the algorithm might not be fully optimized. Challenges that should be tackled for speeding up the calculations include dealing with cases when the expression to be simplified is a long sum containing several repeated color structures with different coefficients. In order to avoid performing the calculations on the same expressions several times, they should be collected together. For the vertex correction relation, such a procedure is implemented in **RemoveAllTriangles**, but similarly, other replacement rules could be upgraded. Furthermore, to require a smaller set of 6j coefficients, the algorithm searching for loops shown in figure 2 could be improved to follow, for example, the method presented in [27]. The conditions for the pattern search, which the program performs to find parts of the expression to simplify, might be optimized as well. Finally, more user-friendly treatment of small N_c could be implemented, including filtering out the basis states that are non-existent at a given N_c .

Acknowledgments

I would like to thank Malin Sjö Dahl for trusting in my strengths and knowledge to do this exciting project and showing how research is conducted in this field. When answering my questions and explaining the topic, she always made sure that I understand it thoroughly. Moreover, she inspired and encouraged me giving a sense of significance for my work.

I thank Hristina Hristova for making sure that every sentence in this thesis both, sounds well and makes sense. Thanks to Anja Langheld for long-distance discussions about Group Theory and to other colleagues at the university for support.

My sincere gratitude goes to my dad, working hard to make my studies in Sweden possible, as well as, to my supporters in Latvia praying for me and encouraging me!

Thanks to everybody who considered me to be worthy enough to read my thesis and even to have a look in my acknowledgments. Of course, big thanks to my bike Swifty for making my travels to university possible and finally thanks to Corona for making this last semester at university unforgettable.

A The available *ColorMath* input

This appendix lists the definitions of building blocks of the *ColorMath* package [18] that are available in the program created in this thesis.



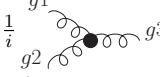
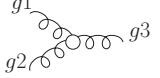
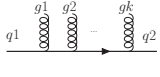
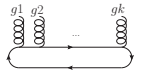
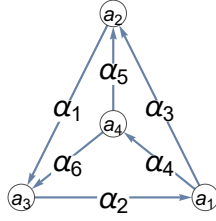
Pictorial representation	Mathematica Standard-Form	Function form	Mathematica FullForm
	$\delta^{q^1}_{q^2}$	$\delta[\mathbf{q1}, \mathbf{q2}]$	<code>Subscript[Superscript[\[Delta], q1], q2]</code>
	$\Delta\{g^1, g^2\}$	$\Delta[\mathbf{g1}, \mathbf{g2}]$	<code>Superscript[\[CapitalDelta], List[g1, g2]]</code>
	$\mathbf{f}\{g^1, g^2, g^3\}$	$\mathbf{f}[\mathbf{g1}, \mathbf{g2}, \mathbf{g3}]$	<code>Superscript[f, List[g1, g2, g3]]</code>
	$\mathbf{d}\{g^1, g^2, g^3\}$	$\mathbf{d}[\mathbf{g1}, \mathbf{g2}, \mathbf{g3}]$	<code>Superscript[d, List[g1, g2, g3]]</code>
	$\mathbf{t}\{g^1, \dots, g^k\}_{q^1, q^2}$	$\mathbf{t}[\{\mathbf{g1}, \dots, \mathbf{gk}\}, \mathbf{q1}, \mathbf{q2}]$	<code>Subscript[Superscript[Superscript[t, List[g1, ..., gk]], q1], q2]</code>
	$\mathbf{o}\{g^1, \dots, g^k\}$	$\mathbf{o}[\{\mathbf{g1}, \dots, \mathbf{gk}\}]$	<code>Superscript[o, List[g1, ..., gk]]</code>

Table 5: The accepted notation used in the Mathematica package *ColorMath* [18]. Converting the *ColorMath* objects into the generalized notation can be done using the replacement rules `/.ColorToMultiplet`. Similarly as for the generalized vertex and delta shown in table 1, the simplest input way is using the function form.

B Alternatives to the three vertex loop contracting functions

To suppress evaluating the 6j coefficients numerically, two other replacement rules instead of `VertexCorrectionRel` are available, namely, `VertexCorrectionRelWrite6j` and `VertexCorrectionRelNonZero6js`. The former outputs the result keeping all the 6j coefficients non-evaluated while the later keeps only non-zero 6j coefficients non-evaluated. The non-evaluated 6j coefficients are in the form $\mathbf{W6C}_{\{\{\alpha_1, \dots, \alpha_6\}, \{a_1, \dots, a_4\}\}}$, where α_i and a_i are the representations of the lines and vertex instances as defined in Eq. (B.1).



(B.1)

The non-evaluated coefficients **W6C** can be replaced by their numerical values using the replacement rules **Evaluate6js**.

C Additional functions available in the program

This appendix includes additional functions and replacement rules which could be helpful for tensor calculations but are not mentioned in the text. Table 6 lists several useful functions, while table 7 gives two replacement rules.

Function	Usage
RepDirectProduct	RepDirectProduct $[Rep_1, Rep_2]$ returns the irreducible representations of $Rep_1 \otimes Rep_2$.
VertexToGraph	VertexToGraph $[Expr]$ converts vertices to Mathematica Edges to be read by the built-in function Graph .
GetVertexInfo	GetVertexInfo $[Expr]$ returns three lists $\{I, V, N\}$ — all indices, representations and vertex instances — in the expression Expr .
CountVertices	CountVertices $[Expr]$ returns the number of vertices in an expression. If the expression can be expanded into a sum or is a list, then it returns the number of vertices for each term as a list $\{Term1, Term2, \dots\}$, otherwise returns an integer.
Coef3J	Coef3J $\{\{\alpha_1, \alpha_2, \alpha_3\}\}$ returns the normalization of a QCD vertex, that is, the Wigner 3j coefficient, with representations $\{\alpha_1, \alpha_2, \alpha_3\}$, where $\alpha_i \in \{3, \bar{3}, 8\}$.
GetIndicesList	GetIndicesList $[Expr]$ returns all of the indices in the expression.
DummyIndices	DummyIndices $[Expr]$ returns all the dummy indices in the expression.
FreeIndices	FreeIndices $[Expr]$ returns all the dummy indices in the expression.
GetRepresentations	GetRepresentations $[Expr]$ returns all the representations found in the expression.

Table 6: Useful functions for tensor summed calculations available in the program.

Rule	Usage
SingletToDelta	Converts a vertex containing a singlet into a delta function, including the normalization.
DeltaToSinglet	Converts a delta function into a vertex containing a singlet. Conversion includes the normalization.

Table 7: Additional replacement rules available for tensor calculations and for comparison between deltas and vertices with singlets.

References

- [1] “High-Luminosity Large Hadron Collider (HL-LHC).” <https://home.cern/science/accelerators/high-luminosity-lhc>, 2020.
- [2] The HSF Physics Event Generator WG, A. Valassi, E. Yazgan, J. McFayden et al., *Challenges in Monte Carlo event generator software for High-Luminosity LHC*, CERN-LPCC-2020-002 (2019) [[arXiv:2004.13687](#)].
- [3] X. C. Vidal, M. d’Onofrio, P. J. Fox, R. Torre et al., *Beyond the Standard Model Physics at the HL-LHC and HE-LHC*, CERN-LPCC-2018-05, HL-LHC Workshop (2018) [[arXiv:1812.07831](#)].
- [4] LHCb collaboration, I. Bediaga, M. C. Torres, J. M. D. Miranda, A. Gomes et al., *Physics case for an LHCb Upgrade II - Opportunities in flavour physics, and beyond, in the HL-LHC era*, CERN-LHCC-2018-027 (2018) [[arXiv:1808.08865](#)].
- [5] M. Cepeda, S. Gori, P. Ilten, M. Kado et al., *Higgs Physics at the HL-LHC and HE-LHC*, CERN-LPCC-2018-04, HL/HE-LHC Workshop (2019) [[arXiv:1902.00134](#)].
- [6] P. Azzi, S. Farry, P. Nason et al., *Standard Model Physics at the HL-LHC and HE-LHC*, CERN-LPCC-2018-03, HL-LHC Workshop (2019) [[arXiv:1902.04070](#)].
- [7] S. Dittmaier, *Standard Model Theory*, *Eur. Phys. Soc. Conf. High Energy Phys.* (2017) [[arXiv:1709.08564](#)].
- [8] G. Heinrich, *QCD calculations for the LHC: status and prospects*, in *5th Large Hadron Collid. Phys. Conf.*, (2017), [arXiv:1710.04998](#).
- [9] T. Sjostrand, S. Mrenna and P. Z. Skands, *A Brief Introduction to PYTHIA 8.1*, *Comput. Phys. Commun.* **178** (2008) 852 [[0710.3820](#)].
- [10] S. Platzer and M. Sjodahl, *The Sudakov Veto Algorithm Reloaded*, *Eur. Phys. J. Plus* **127** (2012) 26 [[1108.6180](#)].
- [11] G. Sterman, *Summation of large corrections to short-distance hadronic cross sections*, *Nucl. Phys. B* **281** (1987) 310.
- [12] S. Catani and L. Trentadue, *Resummation of the QCD perturbative series for hard processes*, *Nucl. Phys. B* **327** (1989) 323.
- [13] M. Sjodahl, *Color evolution of $2 \rightarrow 3$ processes*, *JHEP* **12** (2008) 83 [[arXiv:0807.0555](#)].
- [14] J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer and T. Stelzer, *MadGraph 5: going beyond*, *JHEP* **2011** (2011) 128 [[arXiv:1106.0522](#)].

- [15] P. Cvitanović, *Group theory for Feynman diagrams in non-Abelian gauge theories*, *Phys. Rev. D* **14** (1976) 1536.
- [16] Z. Bern and D. A. Kosower, *Color decomposition of one-loop amplitudes in gauge theories*, *Nucl. Phys. B* **362** (1991) 389.
- [17] Z. Nagy and D. E. Soper, *Parton showers with quantum interference*, *JHEP* **2007** (2007) 114.
- [18] M. Sjö Dahl, *ColorMath—a package for color summed calculations in $SU(N_c)$* , *Eur. Phys. J. C* **73** (2013) 2310 [[arXiv:1211.2099](#)].
- [19] M. Sjö Dahl, *ColorFull – a C++ library for calculations in $SU(N_c)$ color space*, *Eur. Phys. J. C* **75** (2014) 236 [[arXiv:1412.3967](#)].
- [20] V. Del Duca, L. Dixon and F. Maltoni, *New color decompositions for gauge amplitudes at tree and loop level*, *Nucl. Phys. B* **571** (2000) 51.
- [21] F. Maltoni, K. Paul, T. Stelzer and S. Willenbrock, *Color-flow decomposition of QCD amplitudes*, *Phys. Rev. D* **67** (2003) 14026.
- [22] E. Bothmann, G. Singh Chahal, S. Höche, J. Krause et al., *Event generation with Sherpa 2.2*, *SciPost Physics* **7** (2019) 034.
- [23] M. G. Sotiropoulos and G. Sterman, *Color exchange in near-forward hard elastic scattering*, *Nucl. Phys. B* **419** (1994) 59.
- [24] S. Keppeler and M. Sjö Dahl, *Orthogonal multiplet bases in $SU(N_c)$ color space*, *JHEP* **2012** (2012) 124 [[arXiv:1207.0609](#)].
- [25] P. Cvitanović, *Group Theory*. Princeton University Press, feb, 2008.
- [26] M. Sjö Dahl and J. Thorén, *Decomposing color structure into multiplet bases*, *JHEP* **2015** (2015) 55 [[arXiv:1507.03814](#)].
- [27] M. Sjö Dahl and J. Thorén, *QCD multiplet bases with arbitrary parton ordering*, *JHEP* **2018** (2018) 198 [[arXiv:1809.05002](#)].
- [28] Y.-J. Du, M. Sjö Dahl and J. Thorén, *Recursion in multiplet bases for tree-level MHV gluon amplitudes*, *JHEP* **2015** (2015) 119.
- [29] Wolfram Research, Inc., “Mathematica, Version 12.1.”
- [30] A. Zee, *Group Theory in a Nutshell for Physicists*. Princeton University Press, 2016.
- [31] H. F. Jones, *Groups, representations, and physics*. Insitute of Physics Pub., 2nd ed. ed., 1998.

- [32] J. J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*. Cambridge University Press, 2 ed., 2017, [10.1017/9781108499996](https://doi.org/10.1017/9781108499996).
- [33] G. Kane, *Modern Elementary Particle Physics*. Cambridge University Press, feb, 2017, [10.1017/9781316691434](https://doi.org/10.1017/9781316691434).
- [34] M. E. Peskin and D. V. Schroeder, *An Introduction to quantum field theory*. Addison-Wesley, Reading, USA, 1995.
- [35] S. Keppeler, *Birdtracks for $SU(N)$, QCD Master Cl. (2017)* [[arXiv:1707.07280](https://arxiv.org/abs/1707.07280)].
- [36] S. Chadha and M. E. Peskin, *Implications of chiral dynamics in theories of technicolour: (I). Elementary couplings*, *Nucl. Phys. B* **185** (1981) 61.
- [37] A. B. Yutsis, I. B. Levinson and V. V. Vanagas, *Theory of angular momentum*. Israel Program for Scientific Translations, 1962.
- [38] A. Messiah, *Quantum Mechanics*, no. v. 2 in Dover books on physics. Dover Publications, 1999.
- [39] M. Mangano, *The color structure of gluon emission*, *Nucl. Phys. B* **309** (1988) 461 [[arXiv:1503.00530](https://arxiv.org/abs/1503.00530)].
- [40] M. Sjödal, *Color structure for soft gluon resummation: a general recipe*, *JHEP* **2009** (2009) 87 [[arXiv:0906.1121](https://arxiv.org/abs/0906.1121)].