ColorMath—a package for color summed calculations in SU($N_c$)

Malin Sjödahl
Department of Astronomy and Theoretical Physics, Lund University, Sölvegatan 14A, 223 62 Lund, Sweden

Received: 29 November 2012 / Revised: 15 January 2013 / Published online: 20 February 2013
© The Author(s) 2013. This article is published with open access at Springerlink.com

Abstract A Mathematica package for color summed calculations in QCD (SU($N_c$)) is presented. Color contractions of any color amplitude appearing in QCD may be performed, and the package uses a syntax which is very similar to how color structure is written on paper. It also supports the definition of color vectors and bases, and special functions such as scalar products are defined for such color tensors.

1 Introduction

With the LHC follows an increased demand of exact calculations in QCD involving many color charged partons. Due to the non-abelian nature of QCD this poses a nontrivial computational problem.

This is the issue the Mathematica® package ColorMath is set out to tackle. The main feature of ColorMath is thus the ability to automatically perform color summed calculations starting from a QCD color structure which is expressed using a syntax very similar to how the color structure would have been written on paper.

ColorMath allows both partial and full dummy index contractions, and thus deals with color structures having an arbitrary set of free indices. In this sense it is thus more general than [1], on the other hand it is a pure SU($N_c$) tool, rather than a high energy physics general purpose tool, such as [2, 3]. ColorMath works for an arbitrary $N_c \geq 2$, and for arbitrary trace convention of the squared SU($N_c$) generators, $T_R$. It may be used by just giving the color structure in the appropriate form and afterwards running CSimplify to contract all repeated indices. For example, consider $q_1 \overline{q}_2 \rightarrow q_3 \overline{q}_4$ via gluon exchange in the s- and t- channels. Writing down the amplitude as

$$\text{Amp} = S t^{i \ell q_1 q_2} q_2 t^{i \ell q_4 q_3} q_3 + T t^{i \ell q_1 q_2} q_2 t^{i \ell q_3 q_4} q_4$$

where $S$ and $T$ represent some s- and t-channel kinematics, and $t^{i \ell q_1 q_2}$ the SU($N_c$) generator in the fundamental representation, otherwise typically denoted $(t^a)^{i q_1 q_2}$, we may calculate the squared amplitude (defined as in the scalar product Eq. (12)) using

$$\text{CSimplify}[\text{Conjugate}[\text{Amp}]] \text{ReplaceDummyIndices}[\text{Amp}],$$

immediately giving the answer

$$T_R^2 \frac{(N_c^2 - 1)}{N_c} \left[ (N_c S - T) S^* + (N_c T - S) T^* \right].$$

Each squared amplitude can in principle can be calculated like this. However, ColorMath also facilitates the usage of color vectors and bases and has special functions for the calculations of scalar products and gluon exchanges.

This paper, which is intended to be the main reference, is organized as follows: First a general introduction to the basic color building blocks is given in Sect. 2. In Sect. 3 the computational strategy is presented, whereas basic examples are given in Sect. 4, and the usage of vectors and their functions are presented in Sect. 5. In Sect. 6 some remarks concerning validation and scalability are made and in Sect. 7 conclusions are drawn.

2 QCD building blocks

From a color perspective, the QCD Lagrangian is built out of

quark–gluon vertices,

$$i \begin{array}{c} \overline{a} \end{array} \begin{array}{c} c \end{array} j = (t^a)^{ij},$$

and

triple-gluon vertices,

$$a \rightarrow b c = i f^{abc},$$

$\overline{a}$ e-mail: malin.sjodahl@thep.lu.se

ColorMath is available at www.thep.lu.se/~malin/ColorMath.html.
where we follow the convention of reading of the fully antisymmetric structure constant indices in counterclockwise order. The four gluon vertices can be rewritten in terms of (one gluon) contracted triple-gluon vertices, and thus need no special treatment.

Due to confinement, we never observe individual colors and it therefore suffices to calculate color summed/averaged quantities for making predictions in quantum chromodynamics. We may thus constrain ourselves to treat QCD amplitudes carrying a set of external indices with values that need never be specified, as they are always summed over in the end.

In principle, the SU(Nc) generators, along with a delta function for indicating a quark and an anti-quark color singlet, δq1q2, constitute a minimal set of objects for treating the color structure in QCD.\(^1\) For convenience, and for performance reasons, it is, however, useful to define a larger set of objects. The complete set of basic building blocks for carrying color structure used by the ColorMath package is given in Table 1.

Apart from a delta function in quarks indices, δq1q2, a delta function in gluon indices, denoted by Δ\[^{g1,g2}\], is also defined. Note the Mathematica List brackets in \[^{g1,g2}\].

The gluon delta function can alternatively be expressed using

\[
\begin{align*}
g^1 & \cdots \cdots g^2 \quad = \quad TR \begin{bmatrix} g^1 \cdots \cdots g^2 \end{bmatrix} \\
\implies \text{tr}[g^1 g^2] & = TR \delta^{g1 g2} \\
\implies o^{g1 g2} & = TR \Delta^{g1 g2}
\end{align*}
\]

(6)

where \(TR\) comes from the normalization of the SU(Nc) generators, and is typically taken to be 1/2 (the Gell-Man normalization) or 1. A rescaling of the normalization of the SU(Nc) generators can always be absorbed into a normalization of the strong coupling constant. To allow for arbitrary normalization \(TR\) is kept as a free parameter, denoted by TR, and may be defined by the user. Also the number of colors, denoted NC in ColorMath, may be set by the user; by default both NC and TR are kept as free parameters. Note the ColorMath notation for a trace over two gluons \(o^{g1,g2}\) in Eq. (6). Similarly, a general trace over \(k\) gluons \(g_1, g_2, \ldots, g_k\) is denoted \(o^{g_1 g_2 \ldots g_k}\), and may be thought of as a closed quark-line with \(k\) gluons attached.

The totally antisymmetric structure constants, which—along with an extra \(i\)—define the triple gluon vertices, are denoted \(f^{g1 g2 g3}\). Similarly, the totally symmetric “structure constants” are defined as \(d^{g1 g2 g3}\). Recall that, starting from the commutation (anticommutation) relations

\[
[t^{g1}, t^{g2}] = if^{g1 g2 g3} t^{g3}, \quad [t^{g1}, t^{g2}] = d^{g1 g2 g3} t^{g3},
\]

(7)

the structure constants can be rewritten in terms of traces over SU(Nc) generators,

\[
if^{g1 g2 g3} / d^{g1 g2 g3} = \frac{1}{TR} \left[ \text{tr}(t^{g1} t^{g2} t^{g3}) \mp \text{tr}(t^{g2} t^{g1} t^{g3}) \right]
\]

(8)

In ColorMath notation \(t\) is used to denote an open quark-line, and the above expression is written similarly,

\[
f^{g1 g2 g3} / d^{g1 g2 g3} = \frac{1}{TR} \left( o^{g1 g2 g3} \mp o^{g2 g1 g3} \right)
\]

(9)

\[
\begin{align*}
\text{Pictorially this represents} \quad & = \frac{1}{TR} \left[ \begin{array}{c} \circ & \circ \\
\end{array} \right] \\
\text{where } i & \text{ is included in the triple gluon vertex. The rationale for putting the gluon index in the SU(Nc) generators } t^{g_1 g_2 q} \text{ inside a Mathematica List in Eq. (9) is to allow for the natural extension of having many gluons attached to an open quark-line, thus}
\end{align*}
\]

\[
t^{g_1 g_2 \ldots g_k q} = t^{g_1 q_1} d_2 t^{g_2 d_2} d_3 \ldots t^{g_k d_k} q_2.
\]

(11)

The left hand side has several advantages compared to the right hand side. An open quark-line with an arbitrary number of gluon indices can be written in a compact form with no dummy indices. This is not only more human readable, but also superior from a computational point of view, as it avoids the contraction of unnecessary dummy indices. A third advantage with the above notation is its direct correspondence to the trace type bases [4–12]. A basis (or spanning set) for the color space for a fixed set of external quarks, anti-quarks and gluons can always be taken to be a sum of products of open and closed quark-lines.
Along with the number of colors, $N_c$, and the trace of an SU($N_c$) generator squared, $\mathbf{TR}$, the basic building blocks are as below. Note that $\langle g^1 g^2 \ldots g^k \rangle$ represents a trace over gluons with indices $g_1 \ldots g_k$, $\text{tr}[g^1 g^2 \ldots g^k]$, and that $\{g^1 g^2 \ldots g^k\}_{q_1 q_2}$ represent the $q_1 q_2$-component in a trace over generators that has been cut open. $(\delta^{g_1 g_2} \ldots \delta^{g_k q^1 q^2})_{q_1 q_2}$. For convenience, also the totally symmetric “structure constants” $d^{(g_1 g_2 g^3)}$ are defined. Note the Mathematica FullForm. ColorMath is built on pattern matching, and it is therefore essential that the expressions have the correct FullForm. In particular, Power may not be used instead of Superscript. To get the right FullForm it is recommended to use the function form, which is just a function returning the corresponding ColorMath object.

### Table 1

| Pictorial representation | ColorMath | Function form | Mathematica FullForm |
|--------------------------|-----------|---------------|----------------------|
| $q_1$ $\cdots$ $q_2$    | $\delta^{q_1 q_2}$ | $\delta[q_1, q_2]$ | Subscript[Superscript[\[Delta], q1], q2] |
| $\cdots$ $q_1 q_2$     | $\Delta^{(g_1 g_2)}$ | $\Delta[g_1, g_2]$ | Superscript[\[CapitalDelta], List[g1, g2]] |
| $\cdots$ $q_1 q_2$     | $f^{(g_1 g_2 g^3)}$ | $f[g_1, g_2, g^3]$ | Superscript[f, List[g1, g2, g3]] |
| $\cdots$ $q_1 q_2$     | $d^{(g_1 g_2 g^3)}$ | $d[g_1, g_2, g^3]$ | Superscript[d, List[g1, g2, g3]] |
| $\cdots$ $q_1 q_2$     | $t^{(g_1 \ldots g_k q_1 q_2)}$ | $t[[g_1, \ldots, g_k], q_1, q_2]$ | Subscript[Superscript[Superscript[t, List[g1, \ldots, g_k]], q1], q2] |
| $\cdots$ $q_1 q_2$     | $o^{(g_1 \ldots g_k)}$ | $o\{g_1, \ldots, g_k\}$ | Superscript[o, List[g1, \ldots, g_k]] |

In this context we also remark that the color tensors defined in Table 1 are color scalars, i.e., they are invariant under SU($N_c$) transformations. This imposes no restriction for our purposes as, for any QCD amplitude, the overall color structure, including both incoming and outgoing particles, always is a color singlet. As the basic building blocks are invariant, each tensor built out of these objects, i.e., each tensor needed for color summed calculations in perturbative QCD is a actually a color scalar.

The scalar product on this vector space is given by summing over all external color indices, i.e.

$$\langle c_1|c_2\rangle = \sum_{a_1, a_2, \ldots} c_1^{a_1 a_2} c_2^{a_1 a_2 \ldots}$$

(12)

with $a_i = 1, \ldots, N_c$ if parton $i$ is a quark or anti-quark and $a_i = 1, \ldots, N_c^2 - 1$ if parton $i$ is a gluon. As long as the color structures in Table 1 are multiplied by real coefficients the scalar product is actually real, which is easy to prove using the computational rules in the next section.

### 3 Basic computational strategy

Having defined all the color carrying objects, we turn to describing the basic strategy for carrying out calculations.

Again we note that we need not treat the four gluon vertex as this can be rewritten in terms of three gluon vertices. To treat an arbitrary color structure in QCD we may always compute any squared amplitude in the following way:

1. **Rewrite the triple gluon vertices using Eq. (8).** This results in a color structure which is a sum of products of open and closed quark-lines, connected to each other via repeated gluon indices.

2. **Contract all internal gluon indices using the Fierz or completeness relation**

3. **If present, remove Kronecker deltas (for quarks and gluons), and internal quark indices using**

4. **Use the tracelessness of the SU($N_c$) generators, and contract quark and gluon delta functions with repeated indices**

\[
\begin{align*}
\sum_{a_1, a_2, \ldots} 
\end{align*}
\]
After this, all dummy indices have been contracted. A squared amplitude is thus represented by a rational function in $T_R$ and $N_c$, whereas a general amplitude is represented as a sum of products of closed and open quark-lines, i.e., as sums of products of $t_{[g_1,g_2,...,g_k]}^{(g_1,g_2,...,g_k)}q_1^g$, including $t_{[g_1]}^{(g_1)}q_2 = \delta g_1 q_2$, and $o_{[g_1,g_2]}^{(g_1,g_2)}$, which for two gluons may be rewritten as $o_{[g_1,g_2]}^{(g_1,g_2)} = TR\Delta_{[g_1,g_2]}$.

By applying these rules, any amplitude square and any interference term appearing in QCD can be calculated [6, 11]. To successfully square an arbitrary QCD amplitude using the above set of rules we see that steps (i)–(ii) have to be performed while keeping the relative order, i.e., by first applying rule (i) and then rule (ii). We also note that these rules increase the number of terms, whereas the rules (iii)–(iv) decrease the number of terms or keep it fixed. In order not to unnecessarily inflate an expression it may therefore be useful to apply the latter rules at any time during the computation. ColorMath utilizes this and tries to contract indices as it initially was. Applying these to an expression thus tend to simplify it, for example

\[ f_{g_11/12} f_{g_22/112}^2 = 2 N_c T_R \delta g_1 g_2 \]

and all rules for contracting three gluons in products of three symmetric or antisymmetric structure constants, such as

\[ d_{g_11/12} d_{g_22/113} f_{g_33/213} = T_R \frac{N_c^2 - 4}{N_c} f_{g_1 g_2 g_3} \]

are implemented. For more intricate gluon contractions involving structure constants ColorMath can contract indices by first applying rule (i).

## 4 Basic calculations

In this section, basic functionality of ColorMath is explored. As always with Mathematica, the package has to be loaded before it can be used. With

\[
\text{Get["/full/path/to/ColorMath1.0.m"]}
\]

where the version number is adjusted to the version in question, this can be done. To start using the package, it is, however, recommended to start from (a version of) the tutorial “ColorMathTutorial.nb” and modify its content according to the desired usage.

The color contractions corresponding to the basic manipulations from Sect. 3 are carried out using Mathematica Rules, i.e., a set of replacement rules based on pattern matching. As always, the rules may be applied using “Expr/.TheRules”, and may be applied repeatedly using “Expr/.TheRules”.

The rules described in (iii)–(iv) are, along with Eq. (6) contained in a set of rules called SimpleRules, as they keep the expression at least as simple (in a term counting sense) as it initially was. Applying these to an expression thus tend to simplify it, for example

\[
t_{[g_1]}^{(g_1)} q_2 t_{[g_2]}^{(g_2)} q_1 \Delta_{[g_1,g_2]} / . \text{SimpleRules}
\]

results in $TR(\Delta_{[g_1,g_2]}^2)$. To fully utilize the rules we (may) need to apply them repeatedly,

\[
t_{[g_1]}^{(g_1)} q_2 t_{[g_2]}^{(g_2)} q_1 \Delta_{[g_1,g_2]} / . \text{SimpleRules}
\]
giving the fully contracted expression \((-1 + \mathbf{N} c^3) \text{TR}\).

The rules defined in (v)–(vi), acting on \( t \) and \( o \), are similarly contained in \texttt{OTSimpleRules}, and the union of \texttt{SimpleRules} and \texttt{OTSimpleRules} and a few rules for rewriting closed quark-lines with zero to two gluons, are contained in \texttt{AllSimpleRules}.

The special rules for gluon index contraction in structure constants, exemplified in Eqs. (18)–(19) are defined in \texttt{FDRules}. The complete set of rules are stated in Table 2.

Rather than thinking about how individual rules have to be applied, it is convenient to have a standard procedure for contracting color indices. This is embodied in the function \texttt{CSimplify}, which does what its name suggests; simplifies the color structure as far as possible. For color structures which do not contain structure constants this always implies contracting all repeated indices.

For color structure involving the structure constants \texttt{CSimplify} first attempts simplification using the \texttt{FDRules}. If, after this, the expression still contains structure constants, the structure constants are by default rewritten in terms of traces using Eq. (9), and the indices are fully contracted, resulting in a sum of products of open and closed quark-lines. Sometimes it may, however, be desirable not to rewrite the structure constants, as expressions may be more compact if they are kept. This can be achieved by using the option \texttt{RemoveFD} → \texttt{False}.

The most useful set of functions are given in Table 3. Apart from \texttt{CSimplify} we especially note the function \texttt{ReplaceDummyIndices} for replacing all repeated indices in a color structure with a new unique set of color indices.

For the purpose of calculating amplitudes square, we need, as in Eq. (12), the complex conjugated version of the color structures. We note that

\[
\text{Conjugate}[g^{q_1 g_2}] = \overline{g}^{g_2 q_1} \\
\text{Conjugate}[t^{g,1,\ldots,g_l} q_1 g_2] = t^{g_1,\ldots,g_l} q_2 g_2 \\
\text{Conjugate}[o^{g,1,\ldots,g_k}] = o^{g_1,\ldots,g_k}
\]

whereas \( \Delta^{[g_1,g_2]} \), \( f^{g_1,g_2,g_3} \) and \( d^{g_1,g_2,g_3} \) are real. This is implemented in ColorMath via redefinition of the Mathematica’s built in function \texttt{Conjugate}. With this in mind, we are ready to perform calculations of the type in Eq. (2).

Finally we remark that to Mathematica \( \Delta^{[g_1,g_2]} \neq \Delta^{[g_2,g_1]} \), and similarly, \( o^{[g_1,g_2,g_3,g_4]} \neq o^{[g_1,g_2,g_3,g_4]} \) etc. For this reason a function \texttt{SortIndices}, which writes indices in Mathematica default order is defined, s.t. for example

\[
\text{SortIndices}[o^{g_2,g_3,g_4,g_1}] = o^{g_1,g_2,g_3,g_4}.
\]

This function is used by \texttt{CSimplify}, and by some of the rules in Table 2.

Sometimes more detailed control over the calculation may be desired. For this purpose, and for internal usage, functions manipulating indices and probing the color structure are given in Table 4, in the Appendix.

While each squared amplitude in principle can be calculated as outlined above, ColorMath also offers more efficient tools for dealing with vectors in color space.

### 5 Defining and using vectors

For the purpose of studying color space it is often convenient to define a basis for the color space, and sometimes also projection operators. Both of these are examples of color (singlet) tensors, and ColorMath has a set of tools for working directly with such tensors.

As an example, let us consider the color structure for \( q_1 \overline{q}_2 \rightarrow q_3 \overline{q}_4 g_5 \).

\[
\text{Vector18}\{q_1,q_2,q_3,q_4,q_5\} := g^{q_1 q_2} t^{[5]} q_4 q_3 \\
\text{Vector818}\{q_1,q_2,q_3,q_4,q_5\} := t^{[5]} q_4 q_3 \\
\text{Vector888}\{q_1,q_2,q_3,q_4,q_5\} := \text{Module}[\{t^{[1,2]}, t^{[1,2]} q_4 q_3 d^{[1,2,5]}\}] \\
\text{Vector888a}\{q_1,q_2,q_3,q_4,q_5\} := \text{Module}[\{t^{[1,2]}, t^{[1,2]} q_4 q_3 f^{[1,2,5]}\}]
\]

where the basis vectors are labeled using first the overall multiplet of \( q_1 \overline{q}_2 \), then the overall multiplet of \( q_3 \overline{q}_4 g_5 \), and finally the overall multiplet of \( q_3 \overline{q}_4 g_5 \), which—due to color conservation—must equal the multiplet of \( q_1 \overline{q}_2 \), (implying that the notation is somewhat redundant).

From a Mathematica perspective we note a few things. First, on the left hand side, we see that the indices inside the \texttt{List} in the \texttt{Subscript} are followed by underscore to indicate pattern matching. This is standard in Mathematica and makes it possible to use any symbol to denote the indices in later calculations. Then we note that the last two tensors in Eq. (25) are defined using \texttt{Module}. This is to ensure that each time the tensor is used, it comes with a fresh set of dummy indices. This is also the reason why set delayed “\(:=\)” is used.

This basis is orthogonal since at least one set of partons transform under different representations in the various tensors. It is, however, not normalized. Finding the normalization is easy using ColorMath. To calculate for example \texttt{Vector18} squared we could enter

\[
\text{CSimplify[Conjugate[Vector18}\{q_1,q_2,q_3,q_4,q_5\}] \\
\text{Vector18}\{q_1,q_2,q_3,q_4,q_5\} \\
\]

In this user guide indices representing incoming quarks and outgoing anti-quarks are placed upstairs, whereas outgoing quarks and incoming anti-quarks are placed downstairs. Note, however, that we could as well have used the opposite convention.
but it is yet much easier to use the tensor functions for calculating scalar products. Instead we could simply write

\[ \text{CDot}[\text{Vector181}, \text{Vector181}] \]  

resulting in \(Nc(-1 + Nc^2)\text{TR}\). Having a basis we naturally want to calculate all norms (square). ColorMath has special functions for this as well. If we define

\[ \text{OurBasis} \rightarrow \{\text{Vector181}, \text{Vector818}, \text{Vector888s}, \text{Vector888a}\} \]  

we may calculate the squares of the basis vectors using

\[ \text{CDot}[\text{OurBasis}]. \]  

This results in a List containing the scalar products between each basis vector and itself

\[ \{Nc(-1 + Nc^2)\text{TR}, Nc(-1 + Nc^2)\text{TR}, 2(4 - 5Nc^2 + Nc^4)\text{TR}^3/Nc, 2Nc(-1 + Nc^2)\text{TR}^3\}. \]  

\[ \text{(29)} \]

Often non-orthogonal bases are used, in particular, this is the case for the trace type bases. In this case, to square a general amplitude, all the scalar products between all the basis vectors are needed. For calculating the scalar product matrix the command \text{CDotMatrix} may be used. This returns a List of Lists, i.e. a matrix, where the \(ij\)th element is the scalar product between the \(i\)th and \(j\)th vector in the list of (basis) vectors. For larger vector spaces with complicated scalar products, it may be desirable to get progress information on the calculations. This can be obtained by setting the option \text{Verbose} \rightarrow \text{True} for \text{CDotMatrix},

\[ \text{CDotMatrix}[\text{OurBasis}, \text{Verbose} \rightarrow \text{True}]. \]  

\[ \text{(31)} \]

Similarly, it may be useful to be able to simplify potential (normalization) roots assuming a large \(Nc\). This can be done by using the option \text{NcMin}. The scalar product functions, along with their options, are listed in Table 5.

Often it is also of interest to investigate the effect of gluon exchange on the color structure expressed in a basis, i.e. starting in a basis vector \(j\), what is the effect on the basis vector of exchanging a gluon between parton1 and parton2. This is useful both for soft gluon resummation and for one-loop corrections via gluon exchange. The result can be expressed in terms of a matrix whose element \(ij\) is the \(i\)th component resulting after such an exchange in the initial vector \(j\). This is calculated by the function

\[ \text{CGamma}[\text{Basis}, \text{parton1}, \text{parton2}] \]  

\[ \text{(32)} \]

where \text{Basis} is a List of basis vectors, defined using the syntax in Eq. (25) and parton1 and parton2 are the numbers of the partons in the basis vectors, i.e., in this case numbers between 1 and 5. The sign conventions are such that quark–gluon vertex always comes without additional signs, and the triple-gluon vertex have the indices appearing in the order: external index, internal dummy index and index of the exchanged gluon.\(^3\) For example, we may calculate the effect of gluon exchange between parton 1 and 3 in the above basis using

\[ \text{CGamma}[\text{OurBasis}, 1, 3, \text{Verbose} \rightarrow \text{False}, \] 

\[ \text{BasisType} \rightarrow \text{OrthogonalBasis}]. \]  

\[ \text{(33)} \]

Here we have supplied optional information about the basis type, that the basis is orthogonal, to speed up the calculations. By default \text{CGamma} does a few consistency checks. It checks that the vector resulting after gluon exchange, when squared, has the same value as the basis decomposed version. For orthonormal bases (BasisType \rightarrow \text{OrthonormalBasis}) it is also checked that the resulting matrix is symmetric \[14\]. These checks may, however be turned off (MakeChecks \rightarrow \text{False}). The set of options, with default values are listed in Table 5.

\[ 6 \text{ Validation and scalability} \]

The computational rules and functions in this package have been used for calculating the three gluon projection operators presented in \[15\]. This imposes highly nontrivial consistency checks, as it is verified that every projector square equals itself, which at intermediate steps often involve several ten thousand terms. Additional consistency checks on the color contraction rules have been made by using the \text{CGamma} function which checks that vectors square and basis decomposed vectors square agree, and by changing the order in which the color structure contraction rules are applied. The scalar product matrices have also been compared to the ColorFull code \[16\] for tree level trace bases with up to six partons out of which one parton is a quark and one an anti-quark. Selected results have been compared against \[13\], and the package has been tested both in Mathematica 7, 8 and 9.

The computational effort needed for exact treatment of the color space grows very quickly with the number of partons. The dimension of the vector space grows roughly as a factorial in the number of gluons plus \(qq\)-pairs \[15\] (strictly speaking an exponential for finite \(N_c\) in a multiplet basis).

\(^3\)The sign conventions thus differ from the typical eikonal choice.
The computational effort for ColorMath, or any program operating by direct manipulation of quark-lines, tends to grow roughly as the square of this, as the quark-lines are non-orthogonal. ColorMath (in its current form) is thus rather intended to be an easy to use package for calculations of low and intermediate complexity than a competitive tool for processes with very many colored partons.

7 Conclusion

In this paper a Mathematica package ColorMath for performing color summed calculations in QCD is presented. This package allows for simple evaluation of QCD color amplitudes which are expressed in a format which very much resembles how the color structure would have been written on paper, see Table 1. The idea is that the user—for simple cases—just should give the expression, and then run CSimplify[Expr] rather than Simplify[Expr]. The package is based on advanced pattern matching rules, and a list of rules is given in Table 2, whereas functions acting on color structures are given in Table 3.

For calculations of intermediate or high complexity it is often beneficial to use a basis for performing color space calculations. ColorMath allows for definition of color tensors of form $C_1^{ij_1...i_k_1}$, carrying an arbitrary set of quark, anti-quark and gluon indices. Special functions for calculating scalar products, and investigating the effect of gluon exchange, are given in Table 5. ColorMath is, however, not intended for high speed calculations involving many colored partons. For this purpose a separate C++ package is written [16].

Acknowledgements Terrance Figy, Johan Grönqvist, Simon Plätzer, Stefan Prestel, Johan Rathsman and Konrad Tywoniuk are thanked for useful comments on the ColorMath code and/or paper. This work was supported by a Marie Curie Experienced Researcher fellowship of the MNet Research Training network, contract MRTN-CT-2006-035606, by the Helmholtz Alliance “Physics at the Terascale” and by the Swedish Research Council, contract number 621-2010-3326.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

Appendix: Rules and functions

In this appendix rules and functions for index and color structure probing are stated. First the basic rules for ma-

| Rule               | Effect                                                                                           |
|--------------------|--------------------------------------------------------------------------------------------------|
| AllSimpleRules     | AllSimpleRules is the set of all rules involving $\Delta^{[1,2]}$, $\delta^{[1]}$, $\delta^{[3]}$ and $t^{[1,2]} q_2$, which do not increase the number of terms, i.e., the union of SimpleRules and OTSimpleRules |
| ExpandThenRules    | The rules ExpandThenRules first uses Expand, then applies OTThenAllSimpleRules and finally restores default index order with the SortIndices function |
| FDRules            | Rules for gluon contraction for terms involving up to three structure constants, $f^{[1,2,3]}$ or $d^{[1,2,3]}$, i.e., expressions of form Eqs. (18) and (19). The terms resulting after contraction are expressed in terms of structure constants and gluon deltas, $\Delta^{[1,2]}$. |
| FDTToORules        | Rules for replacing structure constants, $f^{[1,2,3]}$ and $d^{[1,2,3]}$, with sums of closed quark-lines $o^{[1,2,3]}$ using the intermediate expression in Eq. (9). |
| FDTToTRules        | Replaces structure constants, $f^{[1,2,3]}$ and $d^{[1,2,3]}$, with a sum of products of SU($N_c$) generators $t^{[1]} q_1 q_2$ using the last form in Eq. (9). |
| OTGluonRules       | Rules for contracting repeated gluon indices in $f^{[1,2,3]}$ and $d^{[1,2,3]}$, with products of SU($N_c$) generators $t^{[1]} q_1 q_2$, The Fierz identity, Eq. (13), is included in these rules. |
| OTSimpleRules      | Rules for contracting neighboring and next to neighboring gluons in closed and open quark-lines, $o^{[1,2,3]}$ and $t^{[1,2]} q_1 q_2$, Eqs. (16) and (17). |
| OTThenAllSimpleRules| The rules OTThenAllSimpleRules first applies OTGluonRules and then repeatedly AllSimpleRules |
| OTToRules          | Rules for replacing open and closed quark-lines, $o^{[1,2,3]}$ and $t^{[1,2]} q_1 q_2$, with products of SU($N_c$) generators $t^{[1]} q_1 q_2$. |
| Remove0ORules      | Rule for simplifying closed quark-lines with 0 gluons, $o^0 = N_c$ |
| Remove0To1ORules   | Rules for simplifying closed quark-lines with 0 or 1 gluons, $o^0 = N_c, o^{[1]} = 0$ |
| Remove0To2ORules   | Rules for simplifying closed quark-lines with 0 to 2 gluons, $o^0 = N_c, o^{[1]} = 0$ and $o^{[1,2]} = T R A^{[1,2]}$ |
| SimpleRules        | Basic rules for quark and gluon contraction. These rules involve $\Delta^{[1,2]}$, $\delta^{[1]} q_2$ or quark contraction, and never increase the number of terms. These rules thus contain the rules in (iii)–(iv) as well as in Eq. (6). |
Table 3 The most useful functions, to be used on any expression carrying color structure of the form given in Table 1. Recall to relabel dummy indices, manually, or by using ReplaceDummyIndices, and to place quark indices that are to be contracted such that one index sits upstairs and one downstairs. Furthermore, check that the correct FullForm is used, cf. Table 1. The function WhatIsWrong may be useful for identifying common input mistakes in the basic color objects

| Function with usage | Effect |
|---------------------|--------|
| CSimplify [Expr]    | The most general function for simplifying color structure. If Expr contains structure constants, \( f(g_1, g_2, g_3) \) or \( d(g_1, g_2, g_3) \), FDRules are first applied. If, after this, the expression still contains structure constants they are—by default—removed using FDTORules, and all repeated indices are subsequently contracted by repeatedly using AllSimpleRules, then OTThenAllSimpleRules, and finally ExpandThenRules. For color structure containing structure constants, it could happen that it is desired not to replace the structure constants. This can be achieved by setting the option RemoveFD \( \rightarrow \) False. |
| Options with default value: RemoveFD \( \rightarrow \) True |
| GluonContract[Expr, Gs] | Contracts a set (List) of gluons \( G_s = \{g_1, \ldots, g_n\} \) or a single gluon \( G_s = g_1, \) in the expression Expr, while leaving other indices unaffected. This function is intended for quark-lines and will replace structure constants with quark-lines. |
| ReplaceDummyIndices[Expr] | Replaces the dummy indices in Expr with a new set of unique dummy indices. |
| SortIndices[Expr] | Sorts the gluon indices appearing in \( \Delta_{g_1, g_2}, f(g_1, g_2, g_3) \) or \( d(g_1, g_2, g_3) \) and \( o(g_1, \ldots, g_k) \) such that they stand in Mathematica default order. This is needed to ensure that one color structure only is represented in one form. |
| SplitConstAndColor[Expr] | Splits an expression Expr into a List of Lists of color structures and corresponding multiplicative factors \([\{\text{constants} 1, \text{color structure} 1\}, \{\text{constants} 2, \text{color structure} 2\}, \ldots]\). This is done by first expanding the expression and then splitting the terms separately. |
| WhatIsWrong[Expr] | Checks if anything is obviously wrong with an expression, for example if Power is used instead of Superscript or if gluon indices are placed downstairs. The check is performed by first expanding the expression, and then checking each term. Read error messages from above. |

Table 4 The below set of functions are used for probing the color structure, and identifying indices. Concerning the convention of what (lower or upper) quark-indices are referred to as incoming quarks and outgoing anti-quarks, as opposed to incoming anti-quarks and outgoing quarks, ColorMath does not specify any preference, but simply refers to the indices as upper quarks and lower quarks when needed.

| Function | Effect |
|----------|--------|
| AllIndices[Expr] | Returns a List of all (external and dummy) indices in Expr |
| ContainsColor[Expr] | Returns True if the expression Expr contains any color structure, i.e., any of the terms in Table 1, and False otherwise |
| ContainsFD[Expr] | Returns True if the expression Expr contains structure constants and False otherwise |
| ContainsGluonDelta[Expr] | Returns True if the expression Expr contains a gluon delta-function, \( \Delta_{g_1, g_2} \), and False otherwise |
| ContainsO[Expr] | Returns True if the expression Expr contains closed quark-lines, \( o(g_1, \ldots, g_k) \), and False otherwise |
| ContainsQuarkDelta[Expr] | Returns True if the expression Expr contains a quark delta-function, \( \delta_{q_1, q_2} \), and False otherwise |
| ContainsT[Expr] | Returns True if the expression Expr contains open quark-lines, \( t(g_1, \ldots, g_k)q_1 q_2 \), and False otherwise |
| DummyIndices[Expr] | Finds the dummy indices in an expression Expr, by first expanding it and then finding all dummy indices in all terms |
| GluonIndices[Expr] | Returns a List of all gluon indices (external and dummy) in the expression Expr |
| LowerQuarkIndices[Expr] | Returns a List of all (external and dummy) quark-type indices placed downstairs, i.e., \( q_2 \) in \( t(g_1, \ldots, g_k)q_1 q_2 \) and \( \delta_{q_1, q_2} \) |
| UpperQuarkIndices[Expr] | Returns a List of all (external and dummy) quark-type indices placed upstairs, i.e., \( q_1 \) in \( t(g_1, \ldots, g_k)q_1 q_2 \) and \( \delta_{q_1, q_2} \) |

Manipulations are stated in Table 2, then the most important functions, operating directly on the basic color objects from Table 1 are listed in Table 3, whereas special functions for indices and color structure probing are stated in Table 4, and the functions dealing with color vector objects are stated in Table 5.
Table 5  Special functions for color structures expressed in the form of Eq. (25), i.e., the corresponding tensors need to be defined using the pattern matching underscores, potential dummy indices should be hidden inside Modules, and the evaluation should be delayed for expressions containing dummy indices, i.e. “:=” should be used. The vector indices should sit inside a List in the Subscript. Additional options may be supplied to these functions to control for example the verbosity. Thus, CNorm may for example be called using CDot[[C1, . . . , Cn], Verbose → True], to get progress information or simply as CDot[[C1, . . . , Cn]] to use the default options

Vector function with usage

| Function                | Usage                                                                 | Effect                                                                 |
|-------------------------|----------------------------------------------------------------------|------------------------------------------------------------------------|
| CDot[C1, C2, Options]   | Options with default value:                                        | Calculates the scalar product between two color tensors C1 and C2. Before using CDot the color tensors \[C1_{i_1, i_2, \ldots, i_k}\] and \[C2_{i_1, i_2, \ldots, i_k}\] should have been defined. For simplifications (in particular of roots) it is by default assumed that Nc is at least 3. However, this may be manually changed by setting the option NeMin, for example CDot[C1, C2, NeMin → 100] |
| NeMin → 3               |                                                                      |                                                                        |
| CDot[[C1, . . . , Cn], Options] | Options with default value:                                        | Similar to CDot[C1, C2] but calculates the scalar product of a set of color tensors and themselves and returns the result as a List where the \[i\]th element is CDot[C1, Cj]. In addition to assuming a minimal value of Nc for simplifications, it is also possible to get progress information by setting the option Verbose to True |
| NeMin → 3               |                                                                      |                                                                        |
| CDotMatrix[[C1, . . . , Cn], Options] | Options with default value:                                        | Calculates the scalar product matrix given a List of color vectors, i.e., element \[ij\] is CDot[Ci, Cj]. If the list contains a basis, this function thus returns the scalar product matrix |
| NeMin → 3               |                                                                      |                                                                        |
| CNorm[C1, Options]      | Options with default value:                                        | Calculates the norm of a color tensors using CDot. For simplifying the result, it may be useful to set the option NeMin to a large value |
| NeMin → 3               |                                                                      |                                                                        |
| CNorm[[C1, . . . , Cn], Options] | Options with default value:                                        | Similar to CNorm above, but calculates the norms of a List of color vectors \[C1, . . . , Cn\], and returns a List containing the norms. Progress information is available by supplying the option Verbose → True |
| NeMin → 3               |                                                                      |                                                                        |
| CGamma[[C1, . . . , Cn], p1, p2, Options] | Options with default value:                                        | Describes the effect of gluon exchange between partons \[p1\] and \[p2\], on basis vector Cj as a column vector \[j\] in a resulting matrix (technically List of List), i.e., element \[ij\] is the resulting color structure’s \[i\]th component. The BasisType option should assume one of the values GeneralBasis, OrthogonalBasis, OrthonormalBasis or TraceBasis. The latter is defined to be any basis where the basis vectors consist of one product of closed and open quark-lines, not a sum. By default it is checked that the color tensor, resulting after gluon exchange, is the same when squared directly, as when basis decomposed, and then squared. This checks that the basis is complete and that the basis decomposition is correct. For orthonormal bases, it is also checked that the resulting matrix is symmetric. Using MakeChecks → False these checks are turned off. By default, progress information is also written out, which can be changed with Verbose → False |
| NeMin → 3               |                                                                      |                                                                        |
| MakeChecks → True       |                                                                      |                                                                        |
| BasisType → GeneralBasis|                                                                      |                                                                        |

References

1. J. Hakkinen, H. Kharrazia, COLOR: a computer program for QCD color factor calculations. Comput. Phys. Commun. 100, 311–321 (1997). hep-ph/9603229
2. J. Alwall, M. Herquet, F. Maltoni, O. Mattelaer, T. Stelzer, MadGraph 5: going beyond. J. High Energy Phys. 1106, 128 (2011). arXiv:1106.0522
3. J. Kuipers, T. Ueda, J. Vermaseren, J. Vollinga, FORM version 4.0. arXiv:1203.6543
4. J.E. Paton, H.-M. Chan, Generalized Veneziano model with isospin. Nucl. Phys. B 10, 516–520 (1969)
5. P. Dittner, Invariant tensors in SU(3). II. Commun. Math. Phys. 27, 44–52 (1972)
6. P. Cvitanović, Group theory for Feynman diagrams in non-Abelian gauge theories. Phys. Rev. D 14, 1536–1553 (1976)
7. P. Cvitanović, P. Lauwers, P. Scharbach, Gauge invariance structure of quantum chromodynamics. Nucl. Phys. B 186, 165–186 (1981)
8. M.L. Mangano, S.J. Parke, Z. Xu, Duality and multi-gluon scattering. Nucl. Phys. B 298, 653 (1988)
9. M.L. Mangano, The color structure of gluon emission. Nucl. Phys. B 309, 461 (1988)
10. Z. Nagy, D.E. Soper, Parton showers with quantum interference. J. High Energy Phys. 09, 114 (2007). arXiv:0706.0017
11. M. Sjodahl, Color structure for soft gluon resummation—a general recipe. J. High Energy Phys. 0909, 087 (2009). arXiv:0906.1121
12. S. Platzer, M. Sjodahl, Subleading Nc improved parton showers. J. High Energy Phys. 1207, 042 (2012). arXiv:1201.0260
13. M. Sjodahl, Color evolution of 2 → 3 processes. J. High Energy Phys. 12, 083 (2008). arXiv:0807.0555
14. M.H. Seymour, M. Sjodahl, Symmetry of anomalous dimension matrices explained. J. High Energy Phys. 12, 066 (2008). arXiv:0810.5756
15. S. Keppeler, M. Sjodahl, Orthogonal multiplet bases in SU(Nc) color space. J. High Energy Phys. 1209, 124 (2012). arXiv:1207.0609
16. M. Sjodahl, ColorFull—a C++ package for color space calculations (2013, in preparation)