Generating QCD antennas

A. van Hameren[a], R. Kleiss[b]

University of Nijmegen, Nijmegen, The Netherlands

Abstract. An extension of the SARGE algorithm of van Hameren, Kleiss and Draggiotis is introduced, which includes the incoming momenta in the kinematical pole structure of the density with which the momenta are generated. The algorithm is compared with RAMBO in the integration of QCD amplitudes in the SPHEL approximation, and the computing times are extrapolated to those for the calculation with exact matrix elements.

1 Introduction

In future experiments with hadron colliders, such as the LHC, many multi-jet events will occur. These can be divided into interesting events (IE) and the background. The main difference between the two classes is that the standard model shall not have proven yet its capability of dealing with the description of the IE at the moment when they are analyzed. The background shall not manifest itself as such a strong test for the standard model. However, we still need to know the cross sections of the background in order to compare the ratio of these and those of the IE with the predictions of the standard model.

For a large part of the background, a piece of the transition amplitude consists of a multi-parton QCD amplitude, and it is well known [5] that it contributes to the cross section with a singular behavior in phase space (PS), given by the so-called *antenna pole structure* (APS). In particular, for processes involving only n gluons the most important contribution comes from the sum of all permutations in the momenta of

$$\frac{1}{(p_1 \cdot p_2)(p_2 \cdot p_3)(p_3 \cdot p_4) \cdots (p_{n-1} \cdot p_n)(p_n \cdot p_1)}, \quad (1)$$

and the singular nature stems from the fact that the scalar products $p_i \cdot p_j$ can become very small. For the calculation of integrals over PS, the Monte Carlo (MC) method is the only option, so that an algorithm to generate random momenta is needed. For processes at high energy, the momenta may be massless, and RAMBO [4] generates any number of them distributed uniformly in PS. This uniform distribution, however, has the disadvantage that, for the integration of an integrand containing the APS, a large number of events is needed to reach a result to acceptable precision. As an illustration, we present in Table 1 the number of PS points needed to integrate the single

Table 1. Typical number of PS points for the integration of a single antenna to an estimated precision of 1%

# momenta	$\dfrac{\text{cut-off}}{\text{CM energy}}$	# PS points
3	0.183	10, 069
4	0.129	26, 401
5	0.100	58, 799
6	0.0816	130, 591
7	0.0690	240, 436
8	0.0598	610, 570

antenna of (1), so not even the sum of its permutations, to an expected error of 1%. The antenna cannot be integrated over the whole of PS because of the singularities, so these have to be cut out. This is done through the restriction $(p_i + p_j)^2 \geq s_0$ for all $i, j = 1, \ldots, n$, and in the table the ratio between $s_0^{1/2}$ and the total energy $s^{1/2}$ is given. These numbers are based on the reasonable choice $s_0/s = 0.2/[n(n-1)]$.

Performing MC integration with very many events is not a problem if the evaluation of the integrand in each PS point is cheap in computing time. This is, for example, the case for algorithms to calculate the squared multi-parton amplitudes based on the so-called SPHEL approximation, for which only the kinematical structure of (1) is implemented [5]. Nowadays, algorithms to calculate the *exact* matrix elements exist, which are far more time consuming [7,8]. As an illustration of what is meant by "more time consuming", we present Table 2 with the typical cpu time needed for the evaluation in *one* PS point of the integrand for processes of two gluons going to more gluons, both for the SPHEL approximation and the exact matrix elements [11]. It is expected, and observed, that the exact matrix elements reveal the same kind of singularity structures as the APS, so that, according to the tables, the PS integra-

Table 2. Typical computing times in seconds for the evaluation of the integrand in 1 PS point

# final gluons	SPHEL	exact
3	2.83×10^{-5}	1.60×10^{-1}
4	9.76×10^{-5}	5.54×10^{-1}
5	4.88×10^{-4}	1.945
6	3.26×10^{-3}	6.06
7	2.57×10^{-2}	19.91
8		64.45

tion for a process with eight final gluons would take in the order of 400 days ...

The solution to this problem is importance sampling. Instead of RAMBO, a PS generator should be used which generates momenta with a density including the APS. In [9], we introduced an algorithm that does part of the job, which is called SARGE (from "staggered antenna radiation generator"). It generates n random momenta with a density proportional to the sum of all permutations of (1), and because they are all random, they should be interpreted as outgoing momenta. In the pole structure of a "real" QCD amplitude, however, also the incoming momenta occur. In this paper, we introduce an extension of the SARGE algorithm, which includes these pole structures. We compare it with RAMBO in the calculation of integrals of QCD amplitudes in the SPHEL approximation, and extrapolate the computing times to those for the calculation with exact matrix elements. The conclusion will be that SARGE takes care of a substantial reduction in computing time.

For the sake of completeness, we describe the full algorithm in this paper, including the piece that was introduced in [9]. What we actually presented there was only the algorithm and no proof at all of its correctness. In this paper, however, we shall meet our engagements with respect to this. We do this with the help of the unitary algorithm formalism, which we introduce in the following section through its application to the RAMBO algorithm.

2 Notation
and the unitary algorithm formalism

The relativistic momentum $p = (p^0, p^1, p^2, p^3) = (p^0, \mathbf{p})$ of an elementary particle is a vector in $\mathbf{R}^4$. The momentum with the opposite 3-momentum is denoted by

$$\tilde{p} = (p^0, -\mathbf{p}). \tag{2}$$

We shall need the first and the fourth canonical basis vectors, which we denote

$$e_0 = (1, 0, 0, 0) \quad \text{and} \quad e_3 = (0, 0, 0, 1). \tag{3}$$

A typical parameterization of a 3-momentum with unit length which we will also need is given by

$$\hat{\mathbf{n}}(z, \varphi) = (\sqrt{1 - z^2} \sin \varphi, \sqrt{1 - z^2} \cos \varphi, z). \tag{4}$$

The Lorentz invariant scalar product shall be denoted with a dot or with parentheses:

$$(pq) = p \cdot q = p^0 q^0 - \mathbf{p} \cdot \mathbf{q}, \tag{5}$$

where

$$\mathbf{p} \cdot \mathbf{q} = p^1 q^1 + p^2 q^2 + p^3 q^3. \tag{6}$$

The product of a vector with itself is denoted as a square

$$p^2 = (pp) = (p^0)^2 - |\mathbf{p}|^2, \quad |\mathbf{p}| = (\mathbf{p} \cdot \mathbf{p})^{1/2}. \tag{7}$$

The same notation for the quadratic form and the 2-component will not lead to confusion, because the 2-component will not appear explicitly anymore after this section. For physical particles, p^2 has to be positive, and in that case, the square root gives the invariant mass of the particle:

$$m_p = \sqrt{p^2} \quad \text{if} \quad p^2 \geq 0. \tag{8}$$

A boost that transforms a momentum p, with $p^2 > 0$, to $m_p e_0$ is denoted $\mathcal{H}_p$, so

$$\mathcal{H}_p p = m_p e_0 \quad \text{and} \quad m_p \mathcal{H}_p e_0 = \tilde{p}. \tag{9}$$

A rotation that transforms p to $p^0 e_0 + |\mathbf{p}| e_3$ is denoted $\mathcal{R}_p$, so

$$\mathcal{R}_p p = p^0 e_0 + |\mathbf{p}| e_3 \quad \text{and} \quad \mathcal{R}_p \tilde{p} = p^0 e_0 - |\mathbf{p}| e_3. \tag{10}$$

Since rotations only change the 3-momentum, we shall use the same symbol if a rotation is restricted to three-dimensional space.

The physical PS of n particles is the $(3n - 4)$-dimensional subspace of $\mathbf{R^{4n}}$, given by the restrictions that the energies of the particles are positive, the invariant masses squared p_i^2 are fixed at given positive values s_i, and that the sum

$$p_{(n)} = \sum_{i=1}^{n} p_i \tag{11}$$

of the momenta is fixed at a given momentum P. The restrictions for the separate momenta shall be expressed with a "PS characteristic distribution"

$$\vartheta_{s_i}(p) = \delta(p^2 - s_i)\theta(p^0), \quad \text{and} \quad \vartheta(p) = \vartheta_0(p). \tag{12}$$

The generic PS integral of a function F of a set $\{p\}_n = \{p_1, \ldots, p_n\}$ of momenta that has to be calculated is then given by

$$\int_{\mathbf{R^{4n}}} \left(\prod_{i=1}^{n} \mathrm{d}^4 p_i \vartheta_{s_i}(p) \right) \delta^4(p_{(n)} - P) F(\{p\}_n). \tag{13}$$

An integral shall always start with a single $\int$ symbol, and for every integration variable, say z, a $\mathrm{d}z$ means "integrate z over the appropriate integration region". If it is not evident what this region is, it shall be made explicit with the help of logical θ functions, which have statements Π as argument, and are defined through

$$\theta(\Pi) = \begin{cases} 1 \text{ if } \Pi \text{ is true,} \\ 0 \text{ if } \Pi \text{ is false.} \end{cases} \tag{14}$$

2.1 The RAMBO algorithm in the UAF

RAMBO was developed with the aim to generate the flat PS distribution of n massless momenta as uniformly as possible, and such that the sum of the momenta is equal to $s^{1/2}e_0$ with s a given squared energy. This means that the system of momenta is in its center-of-mass frame (CMF), and that the density is proportional to the "PS characteristic distribution"

$$\Theta_s(\{p\}_n) = \delta^4(p_{(n)} - \sqrt{s}e_0)\prod_{i=1}^n \vartheta(p_i). \qquad (15)$$

The algorithm consists of the following steps:

Algorithm 1 (RAMBO)

1. generate n massless vectors q_j with positive energy without constraints but under some normalized density $f(q_j)$;
2. compute the sum $q_{(n)}$ of the momenta q_j;
3. determine the Lorentz boost and the scaling transform that bring $q_{(n)}$ to $\sqrt{s}e_0$;
4. perform these transformations on the q_j, and call the result p_j.

Trivially, the algorithm generates momenta that satisfy the various δ constraints, but it is not clear a priori that the momenta have the correct distribution. To prove that they actually do, we apply the unitary algorithm formalism (UAF). We write the generation of a variable as the integral of the density with which that variable is generated, and interpret every assignment as a generation with a density that is given by a Dirac delta distribution. The assignment of the final output should not be written as an integral, but only with the delta distributions. The UAF tells us that Algorithm 1 generates a density

$$\Phi_s(\{p\}_n) = \int \left(\prod_{j=1}^n d^4q_j\vartheta(q_i)f(q_j)\right)$$
$$\times d^4b\delta^4\left(b - \frac{q_{(n)}}{m_{q_{(n)}}}\right)dx\delta\left(x - \frac{\sqrt{s}}{m_{q_{(n)}}}\right)$$
$$\times \prod_{i=1}^n \delta^4(p_i - x\mathcal{H}_bq_i). \qquad (16)$$

The unitarity of the algorithm is expressed by the fact that integration of the above equation over the set of variables $\{p\}_n$ leads to the identity $1 = 1$. To calculate the distribution yielded by this algorithm, the integral has to be evaluated. First of all, some simple algebra using $p_{(n)} = x\mathcal{H}_bq_{(n)}$, $q_{(n)} = x^{-1}\mathcal{H}_b^{-1}p_{(n)}$ and the Lorentz and scaling properties of the Dirac δ distributions leads to

$$\delta^4\left(b - \frac{q_{(n)}}{m_{q_{(n)}}}\right)\delta\left(x - \frac{\sqrt{s}}{m_{q_{(n)}}}\right)$$
$$= \frac{2s^2}{x}\delta^4(p_{(n)} - \sqrt{s}e_0)\delta(b^2 - 1). \qquad (17)$$

Furthermore, since we may write

$$d^4q_j\delta(q_j^2)\delta^4(p_j - x\mathcal{H}_bq_j) = \frac{1}{x^2}\delta(p_j^2) \qquad (18)$$

under the integral, the l.h.s. of (16) becomes

$$\int \Theta_s(\{p\}_n)d^4b\delta(b^2 - 1)dx\frac{2s^2}{x^{2n+1}}$$
$$\times \prod_{i=1}^n f\left(\frac{1}{x}\mathcal{H}_b^{-1}p_i\right)\theta(e_0 \cdot \mathcal{H}_b^{-1}p_j > 0). \qquad (19)$$

In the standard RAMBO algorithm, the following choice is made for f:

$$f(q) = \frac{c^2}{2\pi}\exp(-cq^0), \qquad (20)$$

where c is a positive number with the dimension of an inverse mass. Therefore, if we use that $p_{(n)} = s^{1/2}e_0$ and that $q^0 = e_0 \cdot q$ for any q, then

$$\prod_{i=1}^n f\left(\frac{1}{x}\mathcal{H}_b^{-1}p_i\right)\theta(e_0 \cdot \mathcal{H}_b^{-1}p_i > 0)$$
$$= \left(\frac{c^2}{2\pi}\right)^n\exp\left(-\frac{c}{x}e_0 \cdot \mathcal{H}_b^{-1}p_{(n)}\right)\prod_{i=1}^n \theta(e_0 \cdot \mathcal{H}_b^{-1}p_i > 0)$$
$$= \left(\frac{c^2}{2\pi}\right)^n\exp\left(-\frac{c\sqrt{s}}{x}b^0\right)\theta(b^0 > 0). \qquad (21)$$

As a result of this, the variables p_i, $i = 1, \ldots, n$ only appear in Θ_s, as required. The remaining integral is calculated in the Appendix, with the result that RAMBO generates the density

$$\Phi_s(\{p\}_n) = \Theta_s(\{p\}_n)\left(\frac{2}{\pi}\right)^{n-1}\frac{\Gamma(n)\Gamma(n-1)}{s^{n-2}}. \qquad (22)$$

Incidentally, we have computed here the volume of the PS for n massless particles:

$$\int_{\mathbf{R^{4n}}} d^{4n}p\Theta_s(\{p\}_n) = \left(\frac{\pi}{2}\right)^{n-1}\frac{s^{n-2}}{\Gamma(n)\Gamma(n-1)}. \qquad (23)$$

Note, moreover, that c does not appear in the final answer; this is only natural since any change in c will automatically be compensated by a change in the computed value for x. Finally, it is important to realize that the "original" PS has dimension $3n$, while the resulting one has dimension $3n - 4$: there are configurations of the momenta q_j that are different, but after boosting and scaling end up as the same configuration of the p_j. It is this reduction of the dimensionality that necessitates the integrals over b and x.

3 The basic antenna

As mentioned before, we want to generate momenta that represent radiated partons with a density that has the antenna structure $[(p_1p_2)(p_2p_3)(p_3p_4)\cdots(p_{n-1}p_n)(p_np_1)]^{-1}$.

Naturally, the momenta can be viewed as coming from a splitting process: one starts with two momenta; a third is radiated off creating a new pair of momenta of which a fourth is radiated off and so on. In fact, models similar to this are used in full-fledged Monte Carlo generators like HERWIG. Let us therefore first try to generate a single massless momentum k, radiated from a pair of given massless momenta p_1 and p_2. In order for the distribution to have the correct infrared and collinear behavior, it should qualitatively be proportional to $[(p_1k)(kp_2)]^{-1}$. Furthermore, we want the density to be invariant under Lorentz transformations and scaling of the momenta, keeping in mind that the momenta are three out of possibly more in a CMF and that we have to perform these transformations in the end, like in RAMBO. This motivates us to define the *basic antenna* structure as

$$\mathrm{d}A(p_1, p_2; k) = \mathrm{d}^4 k \vartheta(k) \frac{1}{\pi} \frac{(p_1 p_2)}{(p_1 k)(k p_2)}$$
$$\times g\left(\frac{(p_1 k)}{(p_1 p_2)}\right) g\left(\frac{(k p_2)}{(p_1 p_2)}\right). \quad (24)$$

Here, g is a function that serves to regularize the infrared and collinear singularities, as well as to ensure normalization over the whole space for k: therefore, $g(\xi)$ has to vanish sufficiently fast for both $\xi \to 0$ and $\xi \to \infty$. To find out how k could be generated, we evaluate $\int \mathrm{d}A$ in the CMF of p_1 and p_2. Writing

$$E = \sqrt{(p_1 p_2)/2}, \quad p = \mathcal{H}_{p_1+p_2} p_1, \quad q = \mathcal{H}_{p_1+p_2} k, \quad (25)$$

we have

$$(p_1 p_2) = 2E^2 \quad (26)$$

and

$$(p_1 k) = E q^0 (1 - z), \quad (k p_2) = E q^0 (1 + z), \quad (27)$$

where $z = \mathbf{p} \cdot \mathbf{q}/(|\mathbf{p}||\mathbf{q}|)$. The azimuthal angle of $\mathbf{q}$ is denoted φ, so that $\mathbf{q} = |\mathbf{q}| \mathcal{R}_p^{-1} \hat{\mathbf{n}}(z, \varphi)$. We can write

$$\mathrm{d}^4 k \vartheta(k) = \frac{1}{2} q^0 \mathrm{d}q^0 \mathrm{d}\varphi \mathrm{d}z = \frac{1}{2}(p_1 p_2) \mathrm{d}\varphi \mathrm{d}\xi_1 \mathrm{d}\xi_2, \quad (28)$$

where,

$$\xi_1 = \frac{(p_1 k)}{(p_1 p_2)} \quad \text{and} \quad \xi_2 = \frac{(k p_2)}{(p_1 p_2)}, \quad (29)$$

so that $z = (\xi_2 - \xi_1)/(\xi_2 + \xi_1)$ and $q^0 = E(\xi_2 + \xi_1)$. The integral over $\mathrm{d}A$ takes on the particularly simple form

$$\int \mathrm{d}A(p_1, p_2; k) = \left(\int_0^\infty \mathrm{d}\xi \frac{1}{\xi} g(\xi)\right)^2. \quad (30)$$

The antenna $\mathrm{d}A(p_1, p_2; k)$ will therefore correspond to a unitary algorithm when we let the density g be normalized by

$$\int_0^\infty \mathrm{d}\xi \frac{1}{\xi} g(\xi) = 1. \quad (31)$$

Note that the normalization of $\mathrm{d}A$ fixes the overall factor uniquely: in particular the appearance of the numerator $(p_1 p_2)$ is forced upon us by the unitarity requirement.

For g we want to take, at this point, the simplest possible function we can think of which has a sufficiently regularizing behavior. We introduce a positive non-zero number ξ_{m} and take

$$g(\xi) = \frac{1}{2 \log \xi_{\mathrm{m}}} \theta(\xi_{\mathrm{m}}^{-1} \leq \xi \leq \xi_{\mathrm{m}}). \quad (32)$$

The number ξ_{m} gives a cut-off for the quotients ξ_1 and ξ_2 of the scalar products of the momenta, and not for the scalar products themselves. It is, however, possible to relate ξ_{m} to the total energy $s^{1/2}$ in the CMF and a cut-off s_0 on the invariant masses, i.e., the requirement that

$$(p_i + p_j)^2 \geq s_0 \quad \text{for all momenta } p_i \neq p_j. \quad (33)$$

This can be done by choosing

$$\xi_{\mathrm{m}} = \frac{s}{s_0} - \frac{(n+1)(n-2)}{2}. \quad (34)$$

The invariant masses $(p_1 + k)^2$ and $(k + p_2)^2$ are regularized with this choice, but can still be smaller than s_0 so that the whole of PS, cut by (33), is covered. The s_0 can be derived from physical cuts p_{T} on the transverse momenta and θ_0 on the angles between the outgoing momenta:

$$s_0 = 2p_{\mathrm{T}}^2 \min\left(1 - \cos\theta_0, \left(1 + \sqrt{1 - p_{\mathrm{T}}^2/s}\right)^{-1}\right). \quad (35)$$

With this choice, PS with the physical cuts is covered by PS with the cut of (33). To generate the physical PS, the method of hit-and-miss Monte Carlo can be used, i.e, if momenta of an event do not satisfy the cuts, the whole event is rejected. We end this section with the piece of the PS algorithm that corresponds to the basic $\mathrm{d}A(p_1, p_2; k)$:

Algorithm 2 (Basic antenna)

1. given $\{p_1, p_2\}$, put $p \leftarrow \mathcal{H}_{p_1+p_2} p_1$ and put $E \leftarrow ((p_1 p_2)/2)^{1/2}$;
2. generate two numbers ξ_1, ξ_2 independently, each from the density $g(\xi)/\xi$, and φ uniformly in $[0, 2\pi)$;
3. put $z \leftarrow (\xi_2 - \xi_1)/(\xi_2 + \xi_1)$, $q^0 \leftarrow E(\xi_2 + \xi_1)$ and $\mathbf{q} \leftarrow q^0 \mathcal{R}_p^{-1} \hat{\mathbf{n}}(z, \varphi)$;
4. put $k \leftarrow \mathcal{H}_{p_1+p_2}^{-1} q$.

4 A complete QCD antenna

The straightforward way to generate n momenta with the antenna structured density is by repeated use of the basic antenna. Let us denote

$$\mathrm{d}A_{j,k}^i = \mathrm{d}A(q_j, q_k; q_i); \quad (36)$$

then

$$dA^2_{1,n}dA^3_{2,n}dA^4_{3,n}\cdots dA^{n-1}_{n-2,n} = \qquad (37)$$

$$\frac{(q_1q_n)g_n(\{q\}_n)}{\pi^{n-2}(q_1q_2)(q_2q_3)(q_3q_4)\cdots(q_{n-1}q_n)}\prod_{i=2}^{n-1}d^4q_i\vartheta(q_i),$$

where

$$g_n(\{q\}_n) = g\left(\frac{(q_1q_2)}{(q_1q_n)}\right)g\left(\frac{(q_2q_n)}{(q_1q_n)}\right)g\left(\frac{(q_2q_3)}{(q_2q_n)}\right)$$

$$\times g\left(\frac{(q_3q_n)}{(q_2q_n)}\right)\cdots g\left(\frac{(q_{n-1}q_n)}{(q_{n-2}q_n)}\right). \qquad (38)$$

So if we have two momenta q_1 and q_n, then we can easily generate $n-2$ momenta q_j with the antenna structure. Remember that this differential PS volume is completely invariant under Lorentz transformations and scaling transformations, so that it seems self-evident to force the set of generated momenta in the CMF with a given energy, using the same kind of transformation as in the case of RAMBO. If the first two momenta are generated with density $f(q_1, q_n)$, then the UAF tells us that the generated density $A^{\mathrm{QCD}}_s(\{p\}_n)$ satisfies

$$A^{\mathrm{QCD}}_s(\{p\}_n)$$

$$= \int d^4q_1\vartheta(q_1)d^4q_n\vartheta(q_n)f(q_1,q_n)$$

$$\times dA^2_{1,n}dA^3_{2,n}dA^4_{3,n}\cdots dA^{n-1}_{n-2,n}$$

$$\times d^4b\delta^4(b - q_{(n)}/m_{q_{(n)}})dx\delta(x - \sqrt{s}/m_{q_{(n)}})$$

$$\times \prod_{i=1}^{n}\delta^4(p_i - x\mathcal{H}_bq_i). \qquad (39)$$

If we apply the same manipulations as in the proof of the correctness of RAMBO, we obtain the equation

$$A^{\mathrm{QCD}}_s(\{p\}_n)$$

$$= \Theta_s(\{p\}_n)\frac{(p_1p_n)g_n(\{p\}_n)}{\pi^{n-2}(p_1p_2)(p_2p_3)(p_3p_4)\cdots(p_{n-1}p_n)}$$

$$\times \int d^4b\delta(b^2 - 1)dx\frac{2s^2}{x^5}f\left(\frac{1}{x}\mathcal{H}_b^{-1}p_1,\frac{1}{x}\mathcal{H}_b^{-1}p_n\right). (40)$$

Now we choose f so that q_1 and q_n are generated back-to-back in their CMF with total energy $s^{1/2}$, i.e.,

$$f(q_1, q_n) = \frac{2}{\pi}\delta^4(q_1 + q_n - \sqrt{s}e_0). \qquad (41)$$

If we evaluate the second line of (40) with this f, we arrive at

$$\frac{4s^2}{\pi}\int dx\frac{1}{x^5}d^4b\delta(b^2 - 1)\delta^4\left(\frac{1}{x}\mathcal{H}_b^{-1}(p_1 + p_n) - \sqrt{s}e_0\right)$$

$$= \frac{4}{\pi}\int_0^\infty dx\frac{1}{x^5}\delta\left(\frac{(p_1 + p_n)^2}{sx^2} - 1\right) = \frac{s^2}{2\pi(p_1p_n)^2}, \quad (42)$$

so that the generated density is given by

$$A^{\mathrm{QCD}}_s(\{p\}_n) = \Theta_s(\{p\}_n)\frac{s^2}{2\pi^{n-2}}$$

$$\times \frac{g_n(\{p\}_n)}{(p_1p_2)(p_2p_3)\cdots(p_{n-1}p_n)(p_np_1)}. (43)$$

Note that, somewhat surprisingly, also the factor $(p_np_1)^{-1}$ comes out, thereby making the antenna even more symmetric. In fact, if the density $f(q_1, q_2) = c^4\exp(-cq_1^0 - cq_2^0)/4\pi^2$ is taken instead of the one we just used, the calculation can again be done exactly, with exactly the same result. The algorithm to generate n momenta with the above antenna structure is given by

Algorithm 3 (QCD antenna)

1. generate massless momenta q_1 and q_n;
2. generate $n-2$ momenta q_j by the basic antennas $dA^2_{1,n}$ $dA^3_{2,n}dA^4_{3,n}\cdots dA^{n-1}_{n-2,n}$;
3. compute $q_{(n)} = \sum_{j=1}^n q_j$, and the boost and scaling transforms that bring $q_{(n)}$ to $s^{1/2}e_0$;
4. for $j = 1,\ldots,n$, boost and scale the q_j accordingly, into the p_j.

Usually, the event generator is used to generate a cut PS. If a generated event does not satisfy the physical cuts, it is rejected. In the calculation of the weight coming with an event, the only contribution coming from the functions g is, therefore, their normalization. In total, this gives a factor $1/(2\log\xi_{\mathrm{m}})^{2n-4}$ in the density.

5 Incoming momenta and symmetrization

The density given by the algorithm above is not quite what we want. First of all, we want to include the incoming momenta p_0 and $\tilde{p}_0$ in the APS, so that the density becomes proportional to $[(p_0p_1)(p_1p_2)\cdots(p_{n-1}p_n)(p_n\tilde{p}_0)]^{-1}$ instead of $[(p_1p_2)\cdots(p_{n-1}p_n)(p_np_1)]^{-1}$. Then we want the sum of all permutations of the momenta, including the incoming ones.

5.1 Generating incoming momenta

The incoming momenta can be generated after the antenna has been generated. To show how, let us introduce the following "regularized" scalar product:

$$(pq)_\delta = (pq) + \delta p^0q^0, \qquad (44)$$

where δ is a small positive number. This regularization is not completely Lorentz invariant, but that does not matter here. Important is that it is still invariant under rotations, as we shall see. Using this regularization, we are able to generate a momentum k with a probability density

$$\frac{1}{2\pi I_\delta(p_1, p_2)}\frac{\vartheta(k)\delta(k^0 - 1)}{(p_1k)_\delta(\tilde{k}p_2)_\delta}. \qquad (45)$$

To show how, we calculate the normalization $I_\delta(p_1, p_2)$. Using the Feynman representation of $1/[(p_1k)_\delta(\tilde{k}p_2)_\delta]$, it is easy to see that

$$I_\delta(p_1, p_2) = \frac{1}{4\pi p_1^0 p_2^0} \int \mathrm{d}z\mathrm{d}\varphi \int_0^1 \frac{\mathrm{d}x}{(1 + \delta - |\mathbf{p}_x|z)^2}, \quad (46)$$

where $\mathbf{p}_x = x\hat{\mathbf{p}}_1 + (x-1)\hat{\mathbf{p}}_2$. The integral over z and φ can now be performed, with the result that

$$\begin{aligned} I_\delta(p_1, p_2) &= \frac{1}{p_1^0 p_2^0} \int_0^1 \frac{\mathrm{d}x}{(1+\delta)^2 - |\mathbf{p}_x|^2} \\ &= \frac{1}{2(p_1\tilde{p}_2)} \int_0^1 \frac{\mathrm{d}x}{(x_+ - x)(x - x_-)}, \quad (47) \end{aligned}$$

where

$$x_\pm = \frac{1}{2} \pm \frac{1}{2}\sqrt{1 + \frac{2p_1^0 p_2^0(2\delta + \delta^2)}{(p_1\tilde{p}_2)}} \quad (48)$$

are the solutions for x of the equation $1 + \delta = |\mathbf{p}_x|$. Further evaluation finally leads to

$$I_\delta(p_1, p_2) = \frac{(p_1\tilde{p}_2)^{-1}}{x_+ - x_-} \log\left|\frac{x_+}{x_-}\right|. \quad (49)$$

Notice that there is a smooth limit to the case in which p_1 and p_2 are back-to-back:

$$I_\delta(p, \tilde{p}) = \lim_{q \to \tilde{p}} I_\delta(p, q) = \frac{1}{(p^0)^2(2\delta + \delta^2)}. \quad (50)$$

The algorithm to generate k can be derived by reading the evaluations of the integrals backwards.

Because k and $\tilde{k}$ are back-to-back, they can serve as the incoming momenta. To fix them to $e_0 + e_3$ and $e_0 - e_3$, the whole system of momenta can be rotated. If we generate momenta with the density A_s^{QCD}, use the first two momenta to generate the incoming momenta and rotate, we get a density

$$\begin{aligned} D_s(\{p\}_n) &= \int \mathrm{d}^{4n}q A_s^{\mathrm{QCD}}(\{q\}_n)\mathrm{d}^4k \frac{1}{2\pi I_\delta(q_1, q_2)} \\ &\quad \times \frac{\vartheta(k)\delta(k^0 - 1)}{(q_1k)_\delta(q_2\tilde{k})_\delta} \prod_{i=1}^n \delta^4(p_i - \mathcal{R}_k q_i) \\ &= A_s^{\mathrm{QCD}}(\{p\}_n)I_\delta(p_1, p_2)^{-1} \\ &\quad \times \int \mathrm{d}^4k \frac{\vartheta(k)\delta(k^0 - 1)}{2\pi(p_1\mathcal{R}_k k)_\delta(p_2\mathcal{R}_k\tilde{k})_\delta}, \quad (51) \end{aligned}$$

where we used the fact that the whole expression is invariant under rotations, and that these are orthogonal transformations. The last line of the previous expression can be evaluated further with the result that

$$D_s(\{p\}_n) = A_s^{\mathrm{QCD}}(\{p\}_n)\frac{I_\delta(p_1, p_2)^{-1}}{(p_1p_0)_\delta(\tilde{p}_0p_2)_\delta}, \quad (52)$$

with

$$p_0 = e_0 + e_3, \quad \tilde{p}_0 = e_0 - e_3. \quad (53)$$

The algorithm to generate the incoming momenta is given by

Algorithm 4 (Incoming momenta)

1. given a pair $\{p_1, p_2\}$, calculate x_+ and x_-;
2. generate x in $[0, 1]$ with density $\sim [(x_+ - x)(x - x_-)]^{-1}$, and put $\mathbf{p}_x \leftarrow x\hat{\mathbf{p}}_1 + (x - 1)\hat{\mathbf{p}}_2$;
3. generate φ uniformly in $[0, 2\pi)$, z in $[-1, 1]$ with density $\sim (1 + \delta - |\mathbf{p}_x|z)^{-2}$;
4. put $\mathbf{k} \leftarrow \mathcal{R}_{p_x}^{-1}\hat{\mathbf{n}}(z, \varphi)$ and $k^0 \leftarrow 1$;
5. rotate all momenta with $\mathcal{R}_k$;
6. put $p_0 \leftarrow \frac{1}{2}\sqrt{s}(e_0 + e_3)$ and $\tilde{p}_0 \leftarrow \frac{1}{2}\sqrt{s}(e_0 - e_3)$.

Notice that $I_\delta(p_1, p_2)(p_1p_0)_\delta(\tilde{p}_0p_2)_\delta$ is invariant under the scaling $p_1, p_2 \to cp_1, cp_2$ with a constant c, so that scaling of p_0 and $\tilde{p}_0$ has no influence on the density.

The pair (q_1, q_2) with which k is generated one is free to choose because we want to symmetrize in the end anyway. We should only choose it in such a way that we get rid of the factor (q_1q_2) in the denominator of $A_s^{\mathrm{QCD}}(\{q\}_n)$.

5.2 Choosing the type of antenna with incoming momenta

A density which is the sum over permutations can be obtained by generating random permutations, and returning the generated momenta with permutated labels. This, however, only makes sense for the outgoing momenta. The incoming momenta are fixed, and should be returned separately from the outgoing momenta by the event generator. Therefore, a part of the permutations has to be generated explicitly. There are two kinds of terms in the sum: those in which $(p_0\tilde{p}_0)$ appears, and those in which it does not.

Case 1: antenna with $(p_0\tilde{p}_0)$. To generate the first kind, we can choose a label i at random with a weight equal to $(p_ip_{i+1})/\Sigma_1(\{p\}_n)$, where $\Sigma_1(\{p\}_n)$ is the sum of all scalar products in the antenna[1]:

$$\Sigma_1(\{p\}_n) = \sum_{i=1}^n (p_ip_{i+1}). \quad (54)$$

This is a proper weight, since all scalar products are positive. The total density then gets this extra factor, so that (p_ip_{i+1}) cancels. The denominator of the weight factor does not give a problem, because its singular structure is much softer than the one of the antenna. The pair $\{p_i, p_{i+1}\}$ can then be used to generate the incoming momenta, as shown above. So in this case, a density $A_s^{\mathrm{QCD}}(\{p\}_n)B_1(\{p\}_n)/\Sigma_1(\{p\}_n)$ is generated, where

$$B_1(\{p\}_n) = \sum_{i=1}^n \frac{(p_ip_{i+1})I_\delta(p_i, p_{i+1})^{-1}}{(p_ip_0)_\delta(\tilde{p}_0p_{i+1})_\delta}. \quad (55)$$

Case 2: antenna without $(p_0\tilde{p}_0)$. To generate the second kind, we can choose two non-equal labels i and j with weight $(p_ip_{i+1})(p_jp_{j+1})/\Sigma_2(\{p\}_n)$, where

---

[1] Read $i + 1 \mod n$ when $i + 1$ occurs in this section

$$\Sigma_2(\{p\}_n) = \sum_{i \neq j}^{n} (p_i p_{i+1})(p_j p_{j+1}). \qquad (56)$$

Next, a pair (k, l) of labels has to be chosen from the set of pairs

$$\{(i,j)\}_+ = \{(i,j),(i,j+1),(i+1,j),(i+1,j+1)\}. \qquad (57)$$

If this is done with weight $I_\delta(p_k, p_l)/\Sigma_{i,j}(\{p\}_n)$, where

$$\Sigma_{i,j}(\{p\}_n) = \sum_{(k,l) \in \{(i,j)\}_+} I_\delta(p_k, p_l), \qquad (58)$$

then the density $A_s^{\text{QCD}}(\{p\}_n)B_2(\{p\}_n)/\Sigma_2(\{p\}_n)$ is generated, where

$$
\begin{aligned}
B_2(\{p\}_n) &= \sum_{i \neq j}^{n} (p_i p_{i+1})(p_j p_{j+1}) \\
&\times \sum_{(k,l) \in \{(i,j)\}_+} \frac{I_\delta(p_k, p_l)}{\Sigma_{i,j}(\{p\}_n)} \cdot \frac{I_\delta(p_k, p_l)^{-1}}{(p_k p_0)_\delta (\tilde{p}_0 p_l)_\delta} \\
&= \sum_{i \neq j}^{n} \frac{(p_i p_{i+1})(p_j p_{j+1})}{(p_i p_0)_\delta (p_{i+1} p_0)_\delta (\tilde{p}_0 p_j)_\delta (\tilde{p}_0 p_{j+1})_\delta} \\
&\times \frac{\sum_{(k,l) \in \{(i,j)\}_+} (p_k p_0)_\delta (\tilde{p}_0 p_l)_\delta}{\sum_{(k,l) \in \{(i,j)\}_+} I_\delta(p_k, p_l)}. \qquad (59)
\end{aligned}
$$

Before all this, we first have to choose between the two cases, and the natural way to do this is with relative weights $(1/2)s\Sigma_1(\{p\}_n)$ and $\Sigma_2(\{p\}_n)$, so that the complete density is equal to

$$
\begin{aligned}
S_s^{\text{QCD}}(\{p\}_n) = \frac{1}{n!} \sum_{\text{perm.}} A_s^{\text{QCD}}(\{p\}_n) \\
\times \frac{\frac{1}{2}sB_1(\{p\}_n) + B_2(\{p\}_n)}{\frac{1}{2}s\Sigma_1(\{p\}_n) + \Sigma_2(\{p\}_n)}, \qquad (60)
\end{aligned}
$$

where the first sum is over all permutations of $(1, \ldots, n)$. One can, of course, try to optimize the weights for the two cases using the adaptive multi-channel method (cf. [3]). The result of using the sum of the two densities is that the factors $(p_i p_{i+1})$ in the numerator of B_1 and $(p_i p_{i+1})(p_j p_{j+1})$ in the numerator of B_2 cancel with the same factors in the denominator of A_s^{QCD}, so that we get exactly the pole structure we want. The "unwanted" singularities in B_1, B_2 and Σ_1, Σ_2 are much softer than the ones remaining in A_s^{QCD}, and cause no trouble. The algorithm to generate the incoming momenta and the permutation is given by

Algorithm 5 (Choose incoming pole structure)

1. choose case 1 or 2 with relative weights $\frac{1}{2}s\Sigma_1(\{p\}_n)$ and $\Sigma_2(\{p\}_n)$;

2. in case 1, choose i_1 with relative weight $(p_{i_1} p_{i_1+1})$ and put $i_2 \leftarrow i_1 + 1$;
3. in case 2, choose (i, j) with $(i \neq j)$ and relative weight $(p_i p_{i+1})(p_j p_{j+1})$, and then choose (i_1, i_2) from $\{(i,j)\}_+$ with relative weight $I_\delta(p_{i_1}, p_{i_2})$;
4. use $\{p_{i_1}, p_{i_2}\}$ to generate the incoming momenta with Algorithm 4;
5. generate a random permutation $\sigma \in S_n$ and put $p_i \leftarrow p_{\sigma(i)}$ for all $i = 1, \ldots, n$.

An algorithm to generate the random permutations can be found in [1]. An efficient algorithm to calculate a sum over permutations can be found in [6].

6 Improvements

When doing calculations with this algorithm on a PS, cut such that $(p_i + p_j)^2 > s_0$ for all $i \neq j$ and some reasonable $s_0 > 0$, we notice that a very high percentage of the generated events does not pass the cuts. An important reason why this happens is that the cuts, generated by the choices of g, (32), and ξ_m, (34), are implemented only on quotients of scalar products that appear explicitly in the generation of the QCD antenna:

$$\xi_1^i = \frac{(p_{i-1} p_i)}{(p_{i-1} p_n)} \quad \text{and} \quad \xi_2^i = \frac{(p_i p_n)}{(p_{i-1} p_n)}, \qquad (61)$$
for all $i = 2, 3 \ldots, n-1$.

The total number of these ξ variables is

$$n_\xi = 2n - 4, \qquad (62)$$

and the cuts are implemented such that $\xi_m^{-1} \leq \xi_{1,2}^i \leq \xi_m$ for $i = 2, 3 \ldots, n-1$. We show now how these cuts can be implemented on *all* quotients

$$\frac{(p_{i-1} p_i)}{(p_{j-1} p_j)}, \quad \frac{(p_{i-1} p_i)}{(p_j p_n)} \quad \text{and} \quad \frac{(p_i p_n)}{(p_j p_n)}, \qquad (63)$$
for all $i, j = 2, 3, \ldots, n-1$.

We define the m-dimensional convex polytope

$$
\begin{aligned}
\mathbf{P_m} = \{(x_1, \ldots, x_m) \in [-1,1]^m \big| \\
|x_i - x_j| \leq 1 \; \forall i, j = 1, \ldots, m\}, \qquad (64)
\end{aligned}
$$

and replace the generation of the ξ variables by the following

Algorithm 6 (Improvement)

1. generate $(x_1, \ldots, x_{n_\xi})$ distributed uniformly in $\mathbf{P_{n_\xi}}$;
2. define $x_0 = 0$ and put

$$\xi_1^i \leftarrow e^{(x_{2i-3} - x_{2i-4})\log \xi_m}, \quad \xi_2^i \leftarrow e^{(x_{2i-2} - x_{2i-4})\log \xi_m} \qquad (65)$$
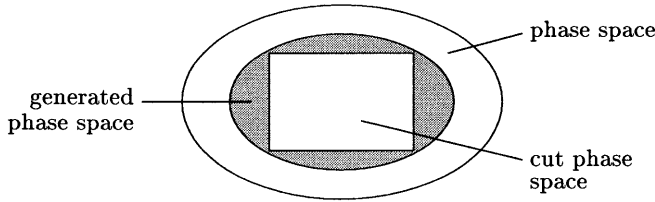
for all $i = 2, \ldots, n-1$.

Fig. 1. Schematic view on phase space

Because all the variables x_i are distributed uniformly such that $|x_i - x_j| \leq 1$, *all* quotients of (63) are distributed such that they are between ξ_m^{-1} and ξ_m. In terms of the variables x_i, this means that we generate the volume of $\mathbf{P}_{\mathbf{n}_\xi}$, which is $n_\xi + 1$, instead of the volume of $[-1,1]^{n_\xi}$, which is 2^{n_ξ}. In [10], we give the algorithm to generate variables distributed uniformly in $\mathbf{P_m}$. We have to note here that this improvement only makes sense because the algorithm to generate these variables is very efficient. The total density changes such, that the function g_n in (43) has to be replaced by

$$g_n^{\mathbf{P}}(\xi_m; \{\xi\}) = \frac{\theta((x_1, \ldots, x_{n_\xi}) \in \mathbf{P}_{\mathbf{n}_\xi})}{(n_\xi + 1)(\log \xi_m)^{n_\xi}}, \qquad (66)$$

where the variables x_i are functions of the variables $\xi_{1,2}^i$ as defined by (65). Because hit-and-miss MC is used to restrict generated events to cut PS, again only the normalization has to be calculated for the weight of an event.

With this improvement, still a large number of events does not pass the cuts. The situation with PS is depicted in Fig. 1. Phase space contains generated phase space which contains cut phase space. The problem is that most events fall in the shaded area, which is the piece of generated PS that is not contained in cut PS. To get a higher percentage of accepted events, we use a random variable $\xi_v \in [0, \xi_m]$, instead of the fixed number ξ_m, to generate the variables $\xi_{1,2}^i$. This means that the size of the generated PS becomes variable. If this is done with a probability distribution such that ξ_v can, in principle, become equal to ξ_m, then the whole of cut phase space is still covered. We suggest the following, tunable, density: introduce a parameter $\alpha \geq 0$ and take

$$h_\alpha(\xi_v) = \frac{\alpha n_\xi + 1}{(\log \xi_m)^{\alpha n_\xi + 1}} \frac{(\log \xi_v)^{\alpha n_\xi}}{\xi_v} \theta(1 \leq \xi_v \leq \xi_m). \quad (67)$$

If $\alpha = 0$, then $\log \xi_v$ is distributed uniformly in $[0, \log \xi_m]$, and for larger α, the distribution peaks more and more towards $\xi_v = \xi_m$. Furthermore, the variable is easy to generate and the total generated density can be calculated exactly: $g_n^{\mathbf{P}}(\xi_m; \{\xi\})$ should be replaced by

$$\begin{aligned} & G_n^{\mathbf{P}}(\alpha, \xi_m; \{\xi\}) \\ & = \int \mathrm{d}\xi_v h_\alpha(\xi_v) g_n^{\mathbf{P}}(\xi_v; \{\xi\}) \qquad (68) \\ & = \frac{1}{n_\xi + 1} \frac{\alpha n_\xi + 1}{(\log \xi_m)^{\alpha n_\xi + 1}} \int_{\log \xi_{low}}^{\log \xi_m} \mathrm{d}x \, x^{(\alpha - 1)n_\xi}, \end{aligned}$$

where ξ_{low} is the maximum of the ratios of scalar products in (63).

Table 3. Cpu times (τ_{SPHEL}) in seconds needed to evaluate the SPHEL integrand one time with a 300 MHz UltraSPARC-IIi processor, and the cpu times (τ_{exact}) needed to evaluate the exact integrand, estimated with the help of Table 1

n	τ_{SPHEL} (s)	τ_{exact} (s)
4	5.40×10^{-5}	3.07×10^{-1}
5	2.70×10^{-4}	1.08
6	1.80×10^{-3}	3.35
7	1.41×10^{-2}	10.92

7 Results and conclusions

We compare SARGE with RAMBO in the integration of the SPHEL integrand for processes of the kind gg $\to$ ng, which is given by

$$\sum_{\text{perm.}} \frac{2 \sum_{i \neq j}^{n+1} (p_i p_j)^4}{(p_1 p_2)(p_2 p_3)(p_3 p_4) \cdots (p_n p_{n+1})(p_{n+1} p_{n+2})(p_{n+2} p_1)}, \qquad (69)$$

where p_1 and p_2 are the incoming momenta, and the first sum is over all permutations of $(2, 3, \ldots, n+2)$ except the cyclic permutations. The results are presented in Table 4.

The calculations were done at a CM energy $s^{1/2} = 1000$ with cuts $p_T = 40$ on each transverse momentum and $\theta_0 = 30°$ on the angles between the momenta. We present the results for $n = 4, 5, 6, 7$, calculated with RAMBO and SARGE with different values for α, see (68). The value of σ is the estimate of the integral at an estimated error of 1% for $n = 4, 5, 6$ and 3% for $n = 7$. These numbers are only printed to show that different results are compatible. Remember that these are not the whole cross sections: flux factors, color factors, sums and averages over helicities, and coupling constants are not included. The other data are the number of generated events (N_{ge}), the number of accepted events (N_{ac}) that passed the cuts, the cpu time consumed (t_{cpu}), and the cpu time the calculation would have consumed if the exact matrix element had been used (t_{exa}), both in hours. This final value is estimated with the help of Table 3 and the formula

$$t_{exa} = t_{cpu} + N_{ac}(\tau_{exact} - \tau_{SPHEL}), \qquad (70)$$

where τ_{exact} and τ_{SPHEL} are the cpu times it takes to evaluate the squared matrix element once. Remember that the integrand only has to be evaluated for accepted events. The calculations have been performed with a single 300 MHz UltraSPARC-IIi processor.

The first conclusion we can draw is that SARGE outperforms RAMBO in computing time for all processes. This is especially striking for a lower number of outgoing momenta, and this behavior has a simple explanation: we kept the CM energy and the cuts fixed, so that there is less energy to distribute over the momenta if n is larger, and the cuts become relatively tighter. As a result, RAMBO gains on SARGE if n becomes larger. This effect would not appear if the energy, or the cuts, would scale with n

Table 4. Results for the integration of the SPHEL integrand with $n = 4, 5, 6, 7$ outgoing gluons. The CM energy and the cuts used are $s^{1/2} = 1000$, $p_T = 40$ and $\theta_0 = 30°$. Presented are the finial result (σ), the number of generated (N_{ge}) and accepted (N_{ac}) events, the cpu time (t_{cpu}) in hours, and the cpu time (t_{exa}) it would take to integrate the exact matrix element, estimated with the help of Table 3. In the calculation of this table, adaptive multi-channeling in the two cases of Sect. 5.2 was used, and $\delta = 0.01$ (Sect. 5.1)

error	n	algorithm	σ	N_{ge}	N_{ac}	$t_{cpu}(h)$	$t_{exa}(h)$
1%	4	RAMBO	4.30×10^8	4,736,672	3,065,227	0.198	262
		SARGE, $\alpha = 0.0$	4.31×10^8	296,050	111,320	0.0254	9.52
		SARGE, $\alpha = 0.5$	4.37×10^8	278,702	40,910	0.0172	3.51
		SARGE, $\alpha = 10.0$	4.32×10^8	750,816	23,373	0.0348	2.03
1%	5	RAMBO	3.78×10^{10}	4,243,360	1,712,518	0.286	514
		SARGE, $\alpha = 0.0$	3.81×10^{10}	715,585	167,540	0.133	51.6
		SARGE, $\alpha = 0.5$	3.80×10^{10}	1,078,129	36,385	0.0758	11.7
		SARGE, $\alpha = 10.0$	3.81×10^{10}	6,119,125	21,111	0.277	9.10
1%	6	RAMBO	3.07×10^{12}	3,423,981	700,482	0.685	653
		SARGE, $\alpha = 0.0$	3.05×10^{12}	2,107,743	276,344	1.32	258
		SARGE, $\alpha = 0.5$	3.13×10^{12}	6,136,375	34,095	0.471	32.2
		SARGE, $\alpha = 10.0$	3.05×10^{12}	68,547,518	17,973	3.17	19.9
3%	7	RAMBO	2.32×10^{14}	605,514	49,915	0.224	152
		SARGE, $\alpha = 0.0$	2.16×10^{14}	710,602	42,394	1.86	130
		SARGE, $\alpha = 0.5$	2.20×10^{14}	5,078,153	3,256	0.452	10.3
		SARGE, $\alpha = 10.0$	2.28×10^{14}	125,471,887	1,789	6.74	12.2

like in Table 1. Another indication for this effect is the fact that the ratio N_{ac}/N_{ge} for RAMBO, which estimates the ratio of the volumes of cut PS and whole PS, decreases with n.

Another conclusion that can be drawn is that SARGE performs better if α is larger. Notice that the limit of $\alpha \to \infty$ is equivalent with dropping the improvement of the algorithm using the variable ξ_v (68). Only if the integrand becomes too flat, as in the case of $n = 7$ with the energy and the cuts as given in the table, smaller values are preferable. Then, too many events do not pass the cuts if α is large.

As an extra illustration of the performance of SARGE, we present in Figs. 2 and 3 the evaluation of MC integrals as a function of the number of accepted events.

Depicted are the integral σ with the bounds on the expected deviation coming from the estimated expected error, and the relative error. Especially the graphs with the relative error are illustrative, since they show that it converges to zero more smoothly for SARGE then for RAMBO. Notice the spike for RAMBO around $N_{ac} = 25000$, where an event obviously hits a singularity.

8 Other pole structures

The APS of (1) is not the only pole structure occurring in the squared amplitudes of QCD processes; not even in purely gluonic processes. For example, in the case of $gg \to 4g$, also permutations of
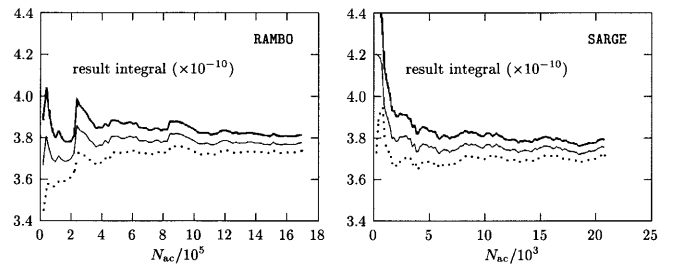

Fig. 2. The convergence of the MC process in the integration of the SPHEL integrand for $n = 5$, with $s^{1/2} = 1000$, $p_T = 40$ and $\theta_0 = 30°$. The graphs show the integral itself as function of the number of accepted events, together with the estimated bounds on the expected deviations. SARGE was used with adaptive multi-channeling in the two cases of Sect. 5.2, with $\delta = 0.01$ (Sect. 5.1) and without the variable ξ_v. The number of generated events was $6,699,944$, and the cpu time was 0.308 hours
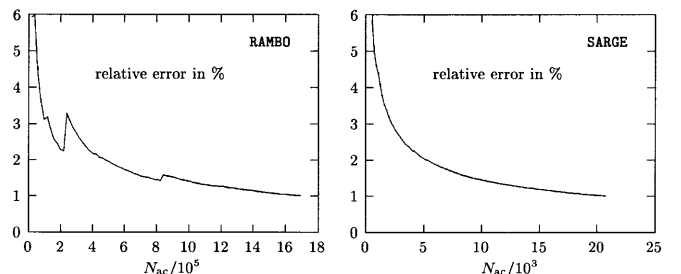
Fig. 3. The relative error in the same processes as in Fig. 2

$$\frac{1}{(p_1 p_3)(p_2 p_4)(p_0 p_1)(\tilde{p}_0 p_2)(p_0 - p_1 - p_2)^2} \quad (71)$$

occur [5]. If one is able to generate momenta with this density, it can be included in the whole density with the use of the adaptive multi-channel technique. In the interpretation of the transition amplitude as a sum of Feynman diagrams, this kind of pole structures typically comes from t-channel diagrams, which are of the type
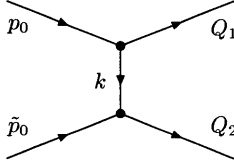
and where, for this case, $Q_1 = p_1 + p_3$ and $Q_2 = p_2 + p_4$, so that $k = p_0 - p_1 - p_3$. The natural way to generate a density with this pole structure is by generating $s_i = Q_i^2$ with a density proportional to $1/s_i$, a variable t that plays the role of $(p_0 - p_1 - p_3)^2$, construct with this and some generated angles the momenta Q_i, and then split new momenta from each of these. For $n = 4$, only two momenta have to split off each Q_i, and there is a reasonable simple algorithm to generate these.

We shall now just present the algorithm, and then show its correctness using the UAF. If we mention the generation of some random variable x "with a density $f(x)$" in the following, we mean a density that is proportional to $f(x)$, and we shall not always write down the normalization explicitly. Furthermore, s denotes the square of the CM energy and $\lambda = \lambda(s, s_1, s_2)$ the usual Mandelstam variable

$$\lambda = s^2 + s_1^2 + s_2^2 - 2ss_1 - 2ss_2 - 2s_1 s_2. \quad (72)$$

Of course, a cut has to be implemented in order to generate momenta following (71), and we shall be able to put $(p_i p_j) > (1/2)s_0$ for the scalar products occurring in the denominator, where s_0 only has to be larger than zero. To generate the momenta with density (71), one should use

Algorithm 7 (t-Channel)

1. generate s_1 and s_2 between s_0 and s with density $1/s_1$ and $1/s_2$;
2. generate t between $s - s_1 - s_2 \pm \sqrt{\lambda(s, s_1, s_2)}$ with density $1/[t(t + 2s_1)(t + 2s_2)]$;
3. put $z \leftarrow (s - s_1 - s_2 - t)/\sqrt{\lambda}$ and generate φ uniformly in $[0, 2\pi)$;
4. put $Q_1 \leftarrow (\sqrt{s_1 + \lambda/(4s)}, \sqrt{\lambda/(4s)}\hat{\mathbf{n}}(z, \varphi))$ and $Q_2 \leftarrow \sqrt{s}e_0 - Q_1$;
5. for $i = 1, 2$, generate $z_i > 1 - 4s_0/(t + 2s_i)$ with density $1/(1 - z_i)$ and φ_i uniformly in $[0, 2\pi)$, and put $q_i \leftarrow \frac{1}{2}\sqrt{s_i}(1, \hat{\mathbf{n}}(z_i, \varphi_i))$;
6. for $i = 1, 2$, rotate q_i to the CMF of Q_i, then boost it to the CMF of $Q_1 + Q_2$ to obtain p_i, and put $p_{i+2} \leftarrow Q_i - p_i$.

As a final step, the incoming momenta can be put to $p_0 \leftarrow (1/2)s^{1/2}(e_0 + e_3)$ and $\tilde{p}_0 \leftarrow (1/2)s^{1/2}(e_0 - e_3)$. The variables s_i and z_i can easily be obtained by inversion (cf. [2]). The variable t can best be obtained by generating $x = \log(2(s_1 s_2)^{1/2}) - \log t$ with the help of the rejection method (cf. [2]). In the UAF, the steps of the algorithm read as follows. Denoting

$$\varepsilon_1 = e_0 + e_3. \quad \varepsilon_2 = e_0 - e_3, \quad (73)$$

$$h_\pm = s - s_1 - s_2 \pm \sqrt{\lambda}, \quad (74)$$

and

$$\begin{aligned}
\mathrm{nrm}(s, s_1, s_2) &= \int \frac{\mathrm{d}t}{t(t + 2s_1)(t + 2s_2)}\theta(h_- < t < h_+) \\
&= \frac{1/4}{s_1 - s_2}\left[\frac{1}{s_2}\log\frac{1 + 2s_2/h_-}{1 + 2s_2/h_+}\right. \\
&\quad \left. - \frac{1}{s_1}\log\frac{1 + 2s_1/h_-}{1 + 2s_1/h_+}\right],
\end{aligned} \quad (75)$$

we have

1. $\displaystyle\int \frac{\mathrm{d}s_1}{s_1}\frac{\mathrm{d}s_2}{s_2}\frac{\theta(s_0 < s_{1,2} < s)}{\left(\log\frac{s}{s_0}\right)^2}$

2. $\displaystyle\int \frac{\mathrm{d}t}{t(t + 2s_1)(t + 2s_2)}\frac{\theta(h_- < t < h_+)}{\mathrm{nrm}(s, s_1, s_2)}$

3. $\displaystyle\int \mathrm{d}z\delta\left(z - \frac{s - s_1 - s_2 - t}{\sqrt{\lambda}}\right)\frac{\mathrm{d}\varphi}{2\pi}$

4. $\displaystyle\int \mathrm{d}^4 Q_1\delta\left(Q_1^0 - \sqrt{s_1 + \frac{\lambda}{4s}}\right)\delta^3\left(\mathbf{Q}_1 - \sqrt{\frac{\lambda}{4s}}\hat{\mathbf{n}}(z, \varphi)\right)$
   $\times \mathrm{d}^4 Q_2\delta^4(Q_1 + Q_2 - \sqrt{s}e_0)$

5. $\displaystyle\int \prod_{i=1}^{2}\frac{\mathrm{d}z_i}{1 - z_i}\frac{\theta\left(1 - z_i > \frac{4s_0}{t + 2s_i}\right)}{\log\frac{t + 2s_i}{2s_0}}\frac{\mathrm{d}\varphi_i}{2\pi}$
   $\times \mathrm{d}^4 q_i\delta\left(q_i^0 - \frac{1}{2}\sqrt{s_i}\right)\delta^3(\mathbf{q}_i - q_i^0\hat{\mathbf{n}}(z_i, \varphi_i))$

6. $\displaystyle\int \prod_{i=1}^{2}\mathrm{d}^4 b_i\delta^4(b_i - \mathcal{H}_{Q_i}\varepsilon_i)$
   $\times \delta^4(p_i - \mathcal{H}_{Q_i}^{-1}\mathcal{R}_{b_i}^{-1}q_i)\delta^4(p_{i+2} + p_i - Q_i)$.

The various assignments imply the following identities. First of all, we have

$$(p_i + p_{i+2})^2 = Q_i^2 = s_i. \quad (76)$$

Using that $4ss_1 + \lambda = (s + s_1 - s_2)^2$ we find

$$\sqrt{4s}(\varepsilon_1 \cdot Q_1) = s + s_1 - s_2 - z\sqrt{\lambda} = t + 2s_1, \quad (77)$$

and the same for $(1 \leftrightarrow 2)$, so that

$$t = 4(p_0 \cdot Q_1) - 2(p_1 + p_3)^2 = -2(p_0 - p_1 - p_3)^2. \quad (78)$$

Denote $\mathcal{L}_{Q_i} = \mathcal{R}_{b_i}\mathcal{H}_{Q_i}$, so that $q_i = \mathcal{L}_{Q_i}p_i$. Because $\mathcal{L}_{Q_i}\varepsilon_i \sim \varepsilon_1$, we find that

$$1 - z_i = \frac{2(\varepsilon_1 \cdot q_i)}{\sqrt{s_i}} = 2\frac{(\varepsilon_1 \cdot \mathcal{L}_{Q_i}p_i)}{(\varepsilon_1 \cdot \mathcal{L}_{Q_i}Q_i)} = 2\frac{(\varepsilon_i \cdot p_i)}{(\varepsilon_i \cdot Q_i)}, \quad (79)$$

so that

$$(t + 2s_1)(1 - z_1) = 8(p_0 \cdot p_1) \quad (80)$$

and

$$(t + 2s_2)(1 - z_2) = 8(\tilde{p}_0 \cdot p_2). \quad (81)$$

We can conclude so far that the algorithm generates the correct pole structure. For the further evaluation of the integrals one can forget about the factors s_i, t, $t + 2s_i$ and $1 - z_i$ in the denominators. Using that

$$\mathrm{d}^4 q_i \delta\left(q_i^0 - \frac{1}{2}\sqrt{s_i}\right)\delta^3(\mathbf{q}_i - q_i^0\hat{\mathbf{n}}(z_i, \varphi_i))$$
$$= 2\mathrm{d}^4 q_i \vartheta(q_i)\delta^3\left(\frac{2}{\sqrt{s_i}}\mathbf{q}_i - \hat{\mathbf{n}}(z_i, \varphi_i)\right), \quad (82)$$

and replacing step 4 by

$$\left(\prod_{i=1}^2 2\sqrt{s_1 + \frac{\lambda}{4s}}\mathrm{d}^4 Q_i \vartheta_{s_i}(Q_i)\right)\delta(z(\mathbf{Q}_1) - z)$$
$$\times \delta(\varphi(\mathbf{Q}_1) - \varphi)\delta^4(Q_1 + Q_2 - \sqrt{s}e_0), \quad (83)$$

the integrals can easily be performed backwards, i.e., in the order q_i, φ_i, z_i, b_i, Q_i, φ, z, t, s_1, s_2. The density finally is

$$\frac{\Theta_s(\{p\}_4)}{(p_0 p_1)(\tilde{p}_0 p_2)(p_1 p_3)(p_2 p_4)[-(p_0 - p_1 - p_3)^2]}$$
$$\times \frac{s}{24(2\pi)^3}$$
$$\times \left[\left(\log\frac{s}{s_0}\right)^2 \log\frac{t + 2s_1}{2s_0} \log\frac{t + 2s_2}{2s_0}\mathrm{nrm}(s, s_1, s_2)\right]^{-1}$$
$$\times \theta(2(p_0 p_1) > s_0)\theta(2(\tilde{p}_0 p_2) > s_0)$$
$$\times \theta(2(p_1 p_3) > s_0)\theta(2(p_2 p_4) > s_0), \quad (84)$$

where $s_i = (p_i + p_{i+2})^2$ and $t = -2(p_0 - p_1 - p_3)^2$.

Appendix

We have to calculate the integral

$$2s^2 \left(\frac{c^2}{2\pi}\right)^n \int \mathrm{d}x\mathrm{d}^4 b\delta(b^2 - 1)\theta(b^0 > 0)$$
$$\times \frac{1}{x^{2n+1}} \exp\left(-\frac{c\sqrt{s}}{x}b^0\right) = \frac{2\Gamma(2n)B(n)}{(2\pi)^n s^{n-2}}, \quad (85)$$

where

$$B(n) = \int \mathrm{d}^4 b\delta(b^2 - 1)\theta(b^0 > 0)(b^0)^{-2n}$$
$$= 2\pi \int_1^\infty \mathrm{d}b^0 (b^0)^{-2n}\sqrt{(b^0)^2 - 1}. \quad (86)$$

The "Euler substitution" $b^0 = (1/2)(v^{1/2} + v^{-1/2})$ casts the integral in the form

$$B(n) = 2^{2n-2}\pi \int_1^\infty \mathrm{d}v\frac{(v-1)^2 v^{n-2}}{(v+1)^{2n}}. \quad (87)$$

By the transformation $v \to 1/v$ it can easily be checked that the integral from 1 to ∞ is precisely equal to that from 0 to 1, so that we may write

$$B(n) = \frac{2^{2n-2}\pi}{2} \int_0^\infty \mathrm{d}v\frac{v^n - 2v^{n-1} + v^{n-2}}{(1+v)^{2n}}$$
$$= 4^{n-1}\pi\frac{\Gamma(n-1)\Gamma(n)}{\Gamma(2n)}, \quad (88)$$

where we have used, by writing $z = 1/(1+v)$, that

$$\int_0^\infty \mathrm{d}v v^p(1+v)^{-q} = \int_0^1 \mathrm{d}z z^{q-p-2}(1-z)^p$$
$$= \frac{\Gamma(q-p-1)\Gamma(p+1)}{\Gamma(q)}. \quad (89)$$

References

1. D.E. Knuth, The art of computer programming, vol. 2, 2nd ed. (Princeton, 1991)
2. L. Devroye, Non-uniform random variate generation (Springer, 1986)
3. R. Kleiss, R. Pittau, Comp. Phys. Comm. **83**, 141 (1994)
4. W.J. Stirling, R. Kleiss, S.D. Ellis, Comp. Phys. Comm. **40**, 359 (1986)
5. J.G.M. Kuijf, Multiparton production at hadron colliders, PhD thesis (University of Leiden, 1991)
6. F. Berends, H. Kuijf, Nucl. Phys. B **353**, 59 (1991)
7. P. Draggiotis, R. Kleiss, C.G. Papadopoulos, Nucl. Phys. B **439**, 157 (1998)
8. F. Caravaglios, M.L. Mangano, M. Moretti, R. Pittau, Nucl. Phys. B **539**, 215 (1999)
9. A. van Hameren, R. Kleiss, P. Draggiotis, Phys. Lett. B **483**, 124 (2000)
10. A. van Hameren, R. Kleiss, preprint physics/0003078
11. P. Draggiotis, private communication