Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2018-0028

Vol. **25**, No. 5, pp. 1547-1563
May 2019

The Adaptive Monte Carlo Toolbox for Phase Space Integration and Generation

R. A. Kycia[1,2,*], J. Turnau, J. J. Chwastowski[3], R. Staszewski[3] and M. Trzebiński[3]

[1] *Tadeusz Kościuszko Cracow University of Technology, Warszawska 24, 31-155 Kraków, Poland.*
[2] *Faculty of Science, Masaryk University, Kotlářská 2, 602 00 Brno, Czech Republic.*
[3] *Institute of Nuclear Physics Polish Academy of Sciences, Radzikowskiego 152, 31-342 Kraków, Poland.*

Received 28 January 2018; Accepted (in revised version) 5 May 2018

Abstract. An implementation of the Monte Carlo (MC) phase space generators coupled with adaptive MC integration/simulation program FOAM is presented. The first program is a modification of the classic phase space generator GENBOD interfaced with the adaptive sampling integrator/generator FOAM. On top of this tool the algorithm suitable for generation of the phase space for an reaction with two leading particles is presented (double-peripheral process with central production of particles). At the same time it serves as an instructive example of construction of a self-adaptive phase space generator/integrator with a modular structure for specialized particle physics calculations.

AMS subject classifications: 65C05, 65Z05, 65Y20, 81U35

Key words: GenExLight, diffractive QCD, phase space integration.

1 Introduction

The need for an efficient phase space generation of multiparticle final states resulted in a large number of programs. These tools offers various degree of generality of the application that improve the integration of the differential cross sections. In addition, they allow for efficient generation of events with the unit weight. Such feature is needed especially when major time consuming detector simulation is involved. These programs employ

*Corresponding author. *Email addresses:* kycia.radoslaw@gmail.com (R. A. Kycia), turnauster@gmail.com (J. Turnau), Janusz.Chwastowski@ifj.edu.pl (J. J. Chwastowski), Rafal.Staszewski@ifj.edu.pl (R. Staszewski), Maciej.Trzebinski@ifj.edu.pl (M. Trzebiński)

the sampling methods which aim at the minimisation of the integrand variance. One example of efficient strategy is described in [5]. The n-body phase space parametrised by $3n-4$ independent variables is divided adaptively into the cells, and techniques which reduce the variance of the result are applied (e.g. importance sampling [12] or [3] and the references therein, or the adaptive integration like VEGAS [11] or FOAM [4]). This strategy described in [5] was implemented as the parton level Monte Carlo Event Generator AcerMC [12] with interfaces to PYTHIA 6.4 [14] and other hadronisation programs. Program described in this paper follows in principle the same strategy but it employs the non-perturbative, e.g. Regge, hadron level amplitudes. Moreover, it is designed to efficiently generate the exclusive final states produced in the diffraction processes at very high energy as measured for example at RHIC or at LHC. To illustrate our approach the program GenExLite, discussed in this paper, has been applied to the reaction

$$p+p\to 2p+2\pi^{+}+2\pi^{-}, \tag{1.1}$$

that represents the continuum production of the pion pairs for which one sets restrictive bounds on the transverse momenta of the final state protons – the leading particles. For a thorough physics discussion of the above reaction see [6].

The matrix elements for reactions described by the Regge model are strongly localised in a small volume of the phase space. Therefore the event generation process applying the adaptive scanning of the integrand function gives one of the best, most general and flexible approach to solve such problems. To the class of the most general adaptive MC integrators belong VEGAS [11] and FOAM [4]. In our toolbox FOAM was selected as it is easily available within the ROOT [17] framework. However, the generators presented here can be easily adapted to use other MC integrators such as e.g. VEGAS – the control flow is similar, classes that describe generation of phase space and possible matrix element are unchanged, and the only difficulty is to implement interfaces for other adaptive MC integrators.

The paper is organized as follows. Section 2 outlines the augmented algorithm of the phase space generation based on the Raubold-Lynch spherical decay algorithm [2] implemented in the TDecay class and interfaced with the external program FOAM for adaptive Monte-Carlo integration. In Section 3 a test of the considered spherical decay tool for efficiency is presented. In Section 4 an example of the specialized self-adapting phase space generator (employing the TDecay class as a basic module, which is also described there) called GenExLight is shown. This code designed for the efficient generation of the double-peripheral processes with two leading particles. A test of its efficiency is presented in Section 5. The kinematical formulae derivations and the program technical details are described in the Appendices A, B and C.

The paper is the continuation of the description of algorithms, tools and their tests started in [7] and used in simulations of diffractive processes in High Energy Physics. However in [7] the emphasis was put on the class structure of expanded GenEx (the Generator for Exclusive processes) MC generator. In the present paper we present simplified version of the generator with completely new design principles that are aimed

at fast prototyping of new reaction models and embedding in the software frameworks dedicated for large scale experiments like ATLAS at LHC. It was created on user demand for simplified version of the tool presented in [7]. In addition, the details of the principles that were used to the selected algorithm to phase-space generation with additional tests were given. Detailed hints on the potential scope of applications of given methods are also outlined.

2 The TDecay – spherical phase space generation algorithm

In this section modifications of the original Raubold-Lynch algorithm [2] implemented in ROOT [17] as the TGenPhaseSpace class will be presented. This modification will be called further the TDecay. The changes allow to interface the algorithm with external adaptive Monte Carlo simulators e.g FOAM. In addition, instead of the relative weight (i.e. probability of the particular phase space configuration of the final particles) implemented in the TGenPhaseSpace, the TDecay provides the absolute phase space weight according to Pilkuhn convention [13] (compare with [1]).

It should be stressed that the current implementation of the class TGenPhaseSpace in ROOT software cannot be used for interfacing with external random number generator and returns only probabilities of the configuration and not the phase space volume element. Therefore the TDecay class presented here can be seen as significant refactorization of the TGenPhaseSpace class aimed at efficiency of use.

The description of the Raubold-Lynch algorithm [2] of the TGenPhaseSpace with modifications incorporated in the TDecay will follow. First, let us consider the integral of some function $f(P;p_1,\cdots,p_N)$ over the phase space[†]

$$\int dLips(s;p_1,\cdots,p_N)f(P;p_1,\cdots,p_N), \tag{2.1}$$

where the sum P of the four-momenta of N particles $\{p_i\}_{i=1}^N$ is conserved, the Lorentz invariant phase space measure is given by

$$dLips(s;p_1,\cdots,p_N)=(2\pi)^4\delta^4\left(P-\sum_{i=1}^N p_i\right)dLips(p_1,\cdots,p_N), \tag{2.2}$$

where $s=P^2$ and, finally,

$$dLips(p_1,\cdots,p_N)=\prod_{i=1}^N\frac{d^3p_i}{(2\pi)^3 2E_i}. \tag{2.3}$$

With substitution

$$f(P;p_1,\cdots,p_N)=\frac{|\mathcal{M}|^2}{2[\lambda(s,m_a^2,m_b^2)]^{1/2}}, \tag{2.4}$$

[†]Note that in the whole paper the integration is written as "an operator" that "acts" on integrated function. It means that instead of $\int f(x)dx$ we write $\int dx f(x)$.

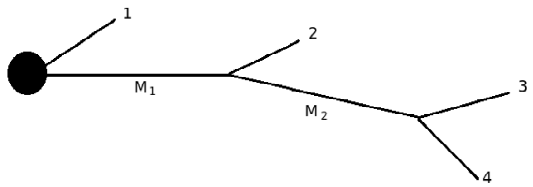

Figure 1: Decay of a central mass object into four final particles. For $n>2$ final state particles there are $n-2$ intermediate particles which masses are ordered $M_1>M_2>\cdots$.

where $\mathcal{M}$ denotes the matrix element for process $p_a+p_b\to p_1+p_2+\cdots+p_N$, $P=p_a+p_b$ and $\lambda(s,m_a^2,m_b^2)=[s-(m_a+m_b)^2][s-(m_a-m_b)^2]$, the integral (2.1) represents its cross section $\sigma(s)$. The substitution

$$f(P;p_1,\cdots,p_N)=\frac{|\mathcal{M}|^2}{2M}, \tag{2.5}$$

where $M=\sqrt{P^2}$ and $\mathcal{M}$ denotes the matrix element for the decay $P\to p_1+p_2+\cdots+p_N$, the integral in Eq. (2.1) represents its total width Γ.

The original Raubold-Lynch idea is sketched in Fig. 1. For a given set of the final particles and the decaying initial state object (marked as a black blob in Fig. 1) of specified four-momentum, one should generate sequential binary decays of intermediate particles until all final state particles are created (see Fig. 1). Each step produces a factor derived from the recurrence relation for phase space [13]:

$$dLips(s;p_1,\cdots,p_m,p_{m+1},\cdots,p_N)=dLips(s;p_1,\cdots,p_m,P_r)\frac{1}{2\pi}dLips(s_r;p_{m+1},\cdots,p_N), \tag{2.6}$$

where the intermediate particle has four-momentum $P_r=\sum_{i=m+1}^{N}p_i$ and $s_r=P_r^2$. Each of $N-1$ binary decays requires two random variables to generate the angle ϕ and $\cos(\theta)$ that fix the direction of the momentum vector of decaying particles. In addition $N-2$ random variables are needed to generate the intermediate masses $M_1>M_2>\cdots$, which have to be ordered to fulfil the energy conservation constraints. This is achieved using the following formulae:

$$M_k=\sum_{i=1}^{k}M_i+\left(\sqrt{s}-\sum_{i=1}^{N}M_i\right)x_k;\quad k=2,3,\cdots,N-1, \tag{2.7}$$

where $0\le x_{k-1}\le x_k\le 1$ for $k=3,\cdots,N-1$. Thus TDecay requires on input the set of $2(N-1)+N-2=3N-4$ random numbers (transferred from FOAM) and $N-2$ of them are required to be in the ascending order. The random set which is not properly ordered can be either rejected or sorted in ascending order. In the latter case $(N-2)!$ permutations of the sorted random variables correspond to $(N-2)!$ points of the FOAM probabilistic space which are mapped to a single point in the physical phase space. It is therefore like the original FOAM probabilistic space was divided into $(N-2)!$ pieces and every one of

them corresponded to the same physical phase space. Therefore, when integrating over such "multiplied" FOAM probabilistic space one has to divide the integrand by the factor $(N-2)!$ to avoid multiple counting [12]. In the former case, when not properly ordered random sets are rejected, only $1/(N-2)!$ of the whole FOAM probabilistic space is used. Then, the efficiency of the exploration drops significantly, especially when the integrand has a very small support in the phase space. For this reasons TDecay sorts $n-2$ variables with additional correction by combinatorial factor $1/(N-2)!$.

The algorithm of TDecay is realized in six steps:

1. Input: rand$[3N-4]$ – the set of $3N-4$ random numbers; N – the numbers of particles to generate; m$[N]$ – array of masses of final particles; P – the decaying particle four-momenta;
2. Check if the decay is energetically allowed: $\sqrt{P^2} > \sum_{i=1}^{N} m[i]$;
3. Sort in ascending order the first $N-2$ random numbers; the additional weight factor is $\frac{1}{(N-2)!}$;
4. Generate $N-2$ intermediate masses using sorted random numbers and Eq. (2.7);
5. For all final particles:
   (a) Calculate $|\vec{p}_i|$ from the two body decay of the $(i-1)$-th intermediate object to i-th particle and i-th intermediate object;
   (b) Generate $cos(\theta)$ and ϕ angle using two random number and rotate $\vec{p}_i$ from the OZ axis into a given direction given by these variables;
   (c) Perform a Lorentz boost from the centre of mass frame of $i-1$ intermediate object into rest frame of the whole event;
   (d) Update the weight for the binary decay;
6. For all final particles perform boost from the centre of mass frame of P to the rest frame.

The above described algorithm represents only a minor modification of the TGenPhaseSpace algorithm from ROOT library. Its implementation requires a detailed description which is provided in Appendix A.

3 TDecay – the efficiency test

First part of the tests compares spherical decay provided by the standard TGenPhaseSpace and its improved version called TDecay.

A program using the TDecay was tested for the generation efficiency of the unweighted events (i.e. events with unit weight) measured by the time t_{gen} needed to

generate one such event. The performance of TDecay is compared to that of the TGenPhaseSpace, the non-adaptive implementation of the Raubold-Lynch algorithm, using the standard rejection sampling for generation of the unweighted events.

Both programs perform their task in two steps. In case of the TGenPhaseSpace in order to obtain unweighted events rejection sampling algorithm was used. In the first step the maximal value of the integrand distribution $WtMax$ (weight maximum) is searched for in the exploration step, using random sampling points in the whole domain. The next step is the rejection sampling – an event is rejected if its *weight* fulfils

$$weight < randRej \times maxWt, \tag{3.1}$$

where $randRej \in [0;1]$ is the random number sampled from an uniform distribution defined on the unit interval. Otherwise the event is accepted with $weight = 1$. The efficiency of the random sampling increases with diminishing weight variance.

In the case of TDecay the FOAM returns unweighted events. It subdivides the domain of the integrand into *ncell* cells (see Appendix A) in such a way that the weight variance within cell is minimised during the exploration step. The rejection sampling performed cell by cell in the generation step is then more effective and with enough cells it may reach almost 100% efficiency.

When comparing the TDecay with the TGenPhaseSpace it is important to set the exploration step parameters for the TGenPhaseSpace in such a way that $WtMax$ (global and by cell) is sufficiently well determined. The event generation time, t_{gen}, does not include the time needed to perform the exploration step. For the TDecay-TGenPhaseSpace benchmark the number of events generated in the TGenPhaseSpace exploration step was equal to $n_{cell} \times n_{sample} = 10^8$ where $n_{cell} = 10^4$ and $n_{sampl} = 10^4$ are FOAM settings for TDecay run. The number of generated events was set to 10^5 for both programs, $|\mathcal{M}|^2 = 1$, $\sqrt{(s)} = 200$ GeV. Calculation has been performed for 4, 5, and 6 final state particles (π mesons). Results are presented in the Table 1[‡].

One can see that for the unit matrix element the adaptive sampling provided by FOAM offers some advantage for N=4 and almost identical generation time, t_{gen}, for $N = 5,6$ due to insufficient number of cells in the exploration step. The advantage becomes evident when the phase space is restricted by the matrix element or for much larger $n_{cell} \times n_{sampl}$ FOAM setting. This is illustrated in the next benchmark test, in which the matrix element represents the Gaussian distribution of particles with the transverse momenta p_{ti} with respect to OZ axis:

$$\mathcal{M}(P;p_1,p_2,\cdots,p_N) = \prod_{i=1}^{N} \exp\left(\frac{-p_{ti}^2}{2\rho^2}\right). \tag{3.2}$$

We use the same program settings as above and change both the number of final particles and the departure from spherical symmetry of the matrix element, regulated by the

[‡]A note on errors: for example, the result 8.02 means that the true value is 8.02 ± 0.01.

Table 1: Comparison of times of generation of events with unit weight and values of integral using an adaptive (A) TDecay program with FOAM setting $n_{cell}=10^4$; $n_{sample}=10^4$ and non-adaptive (NA) the TGenPhaseSpace with 10^8 events in the exploration step. In both cases the matrix element $\mathcal{M}$ is set to 1 and $\sqrt{s}=200$ GeV. N is the number of generated final state particles, $t...$ with subscripts genA and genNA for adaptive (A) and non-adaptive (NA) sampling is the generation time of a single event. The errors for adaptive (σ_A) and non-adaptive (σ_{NA}) integral (in arbitrary units) are indicated by the last significant digit and result from usual formula for MC error (NA case) or from adaptive FOAM algorithm (A case). Tests were performed using Intel Xeon E5-2680 v3 @ 2.5 GHz processor.

N	t_{genA}[ms]	t_{genNA}[ms]	t_{genNA}/t_{genA}	σ_A	σ_{NA}
4	5.3	21.6	0.25	8.02	8.1
5	5.7	5.5	1.04	1410	1410
6	4.0	4.0	1.00	46080	46060

Table 2: Comparison of the generation times of events having the unit weight and the values of integral obtained using an adaptive (A) TDecay program with FOAM settings $n_{cell}=10^4$; $n_{sample}=10^4$ and non-adaptive (NA) the TGenPhaseSpace with 10^8 events in the exploration step. In both cases the matrix element $\mathcal{M}$ is set to the multi-gauss distribution (3.2), and $\sqrt{s}=200$ GeV. N is the number of generated final state particles, $t...$ with subscripts genA and genNA for adaptive (A) and non-adaptive (NA) sampling is the generation time of single event. The errors of adaptive (σ_A) and non-adaptive (σ_{NA}) integral (in arbitrary units) are indicated by the last significant digit and result from usual formula for MC error (NA case) or from adaptive FOAM algorithm (A case). The tests were performed using Intel Xeon E5-2680 v3 @ 2.5 GHz processor.

N	ρ [GeV]	t_{genA}[ms]	t_{genNA}[ms]	t_{genA}/t_{genNA}	σ A	σ NA
4	5	1.56	1231.3	0.0013	7.25	7.58
4	10	0.75	159.03	0.0047	2.08	2.08
4	15	0.64	74.63	0.0080	7.91	7.91
5	10	4.03	456.5	0.009	1.928	1.93
5	15	2.66	211.54	0.013	1.363	1.35
6	10	7.39	426.25	0.017	1.442	1.44
6	15	4.33	457.30	0.009	1.796	1.79

parameter ρ. The advantage of the adaptive sampling increases with number of particles and depending on the ρ-value the improvement can reach many orders of magnitude in generation time. The results of the test are presented in the Table 2. For $N=5,6$ and $\rho\leq 5$ the TGenPhaseSpace could not produce the required number of events (10^5) within 3 days so the calculations were abandoned.

4 GenExLight – adaptive phase space integration/generation

In principle TDecay as an adaptive sampling phase space generator/integrator is an universal tool which can deal with any matrix element, provided that FOAM is set up with a sufficient number of cells and samplings per cell. However, in many situations, application of the strategy outlined in [5] (see Introduction), which combines the adaptive

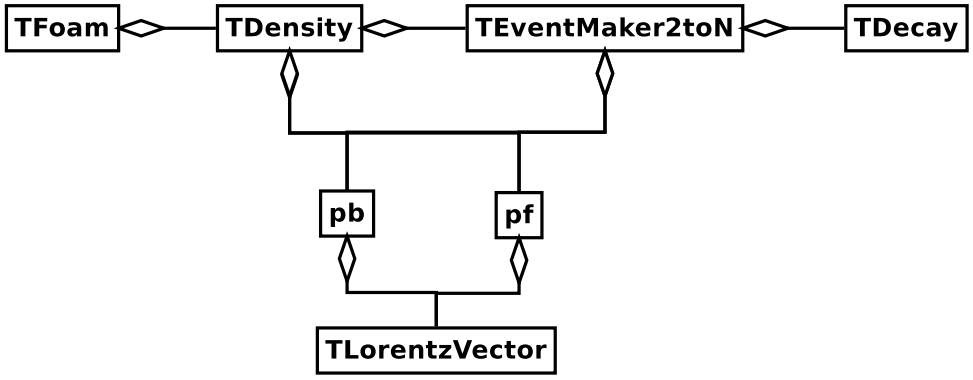

Figure 2: Class diagram of GenExLight generator.

sampling with the appropriate change of some integration variables, appears to be necessary for practical reasons (computation time). In the following, we describe GenExLight, the phase space generator with a modular structure and the adaptive integrator. It can be treated as a light version of GenEx [7] that can be quickly adapted to specialized particle physics calculations. The code we describe below is adapted for the high energy processes in which the four-momentum transfer to two final state particles, so-called leading particles, is strongly (e.g. exponentially) restricted. The central particle production in the multi-peripheral processes [6, 8, 10] is an example. This code, without a complicated object oriented structure, allows even beginners to modify it to the particular applications. At the end of this section we provide examples of possible modifications.

The structure of GenExLight is presented in Fig. 2. It is completely redesigned and simplified compared to full GenEx of [7]. Therefore it can be used for fast prototyping of new theoretical models and embedding in more versatile software packages (or frameworks). As a base it uses TDecay class described previously. The TFoam class of adaptive Monte Carlo simulator requires TDensity class which provides integrand – a function of the four-momenta of the particles that are derived from the random numbers provided by TFoam in TEventMaker2toN. This class generates two leading particles and a central object which then is decayed by TDecay into $N-2$ remaining particles (see Fig. 3). Events are accessible in the tables pb for beam particles and pf for final particles containing their four-momenta as TLorentzVector objects.

Construction of the TEventMaker2toN class uses kinematics derived in [8] for the description of the double peripheral process $pp \to ppf_0$. The cross section formula for the reaction $p_a + p_b \to \sum_{i=1}^{N} p_i$ is:

$$\sigma = \int (2\pi)^4 \delta^{(4)} \left(p_a + p_b - \sum_{i=1}^{N} p_i \right) \prod_{i=1}^{N} \frac{d^3 p_i}{(2\pi)^3 2E_i} \frac{|\mathcal{M}|^2}{2s}. \tag{4.1}$$

It can be transformed into the form appropriate for the factorizable matrix element of the

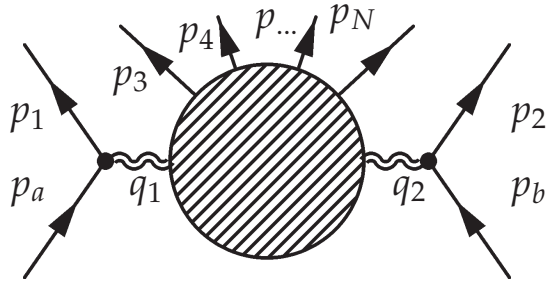

Figure 3: The illustration of the kinematics built by TEvantMaker2toN class adapted to integration/generation of the double peripheral central production processes. The peripherally scattered objects a and b do not enter into the phase space occupied by centrally produced particles $3,4,\cdots,N$.

process depicted in Fig. 3,

$$\sigma=\overbrace{\int_{M^2_{min}}^{M^2_{max}}dM^2\int(2\pi)^4\delta^{(4)}(p_a+p_b-(p_1+p_2+P_M))\frac{d^3p_1}{(2\pi)^3 2E_1}\frac{d^3p_2}{(2\pi)^3 2E_2}\frac{d^3P_M}{(2\pi)^3 2E_M}}^{\text{B}}}^{\text{A}}$$
$$\underbrace{\frac{1}{2\pi}\int(2\pi)^4\delta^{(4)}\left(P_M-\sum_{i=3}^{N}p_i\right)\prod_{i=3}^{N}\frac{d^3p_i}{(2\pi)^3 2E_i}}_{\text{C}}\underbrace{\frac{|\mathcal{M}|^2}{2s}}_{\text{D}}, \tag{4.2}$$

where p_1, p_2 are peripheral particles and $P_M=\sum_{i=3}^{N}p_i$ is the total four-momentum of the centrally produced object which decays into $N-2$ centrally produced particles. In Eq. (4.2) part A represents integration over the invariant mass square of the centrally produced object, B – the integration over the 5-dimensional phase space of the $2\to3$ process depicted in Fig. 3 and C – the integration over the phase space of the decay of the central object into $N-2$ final particles. Finally, the D part is the square of an absolute value of a matrix element divided by the flux. It is also assumed that the matrix element factorizes as:

$$\mathcal{M}(s,p_1,p_2,\cdots,p_N)=\mathcal{M}_{2\to3}(s,p_a,p_b,p_1,p_2,p_M)\mathcal{M}_{1\to N-2}(P_M,p_3,p_4,\cdots,p_N). \tag{4.3}$$

This assumption excludes the possibility of an interference between the identical leading and centrally produced particles as well as any matrix element factors which break factorization in Eq. (4.3) (e.g. some models of absorption [9]). For the double-peripheral process at sufficiently high energy, the interference is indeed excluded, because the leading and central particles occupy different regions of phase space. The integral in part B has to be transformed into the final variables appropriate for the efficient integration:

$$B=\int\sum_{p'_{1z}\in\{p''_{1z}:f(p''_{z1})=0\}}\frac{(2\pi)^4}{J}\frac{|\vec{p}_{1t}|d|\vec{p}_{1t}|d\phi_1}{(2\pi)^3 2E_1}\frac{|\vec{p}_{2t}|d|\vec{p}_{2t}|d\phi_2}{(2\pi)^3 2E_2}\frac{dy_M}{(2\pi)^3 2}. \tag{4.4}$$

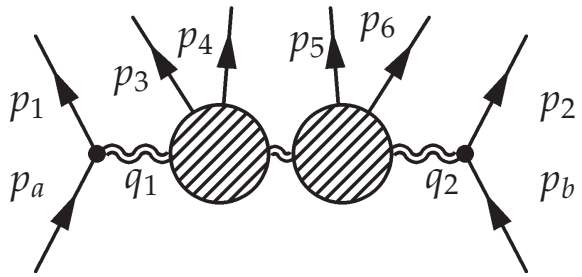

Figure 4: The illustration of possible modification of the kinematics to be built by TEventMaker2toN class, adapted to integration/generation of the triple peripheral process with the central production of two objects.

where ϕ_1 and ϕ_2 are the polar angles, y_M is the rapidity of the central object and $J = \left|\frac{p_{1z}}{E1}+\frac{p_{2z}}{E_2}\right|$. The summation goes over the solutions of the quadratic equation $f(p_{1z})=0$ where

$$f(p_{1z}) = \sqrt{s} - \sqrt{p_{1t}^2+p_{1z}^2} - \sqrt{p_{2t}^2+p_{2z}^2} - E_M. \tag{4.5}$$

If $\sqrt{s} \gg E_M$ we deal with almost two-body process, the solutions correspond to two parts of the phase space in which $p_{az}p_{1z}>0$ (and $p_{bz}p_{2z}>0$) and $p_{az}p_{1z}<0$ (and $p_{bz}p_{2z}<0$). In the very high energy double peripheral processes the initial particles do not "bounce" from each other and only the first solution contributes to the integral. However, in a general case both solutions should be equally treated. This is achieved by adding an additional random variable that will select one of the two of solutions for each generated event. The derivation of the Eq. (4.4) is shown in Appendix B. For the technical description of GenExLight see Appendix C.

GenExLight should be considered as a particular realization of the strategy for a modular construction of the effective generators with an adaptive sampling using TDecay as a basic module. Let us reiterate the essential elements of the program structure. TEventMaker2toN class builds kinematics adapted to the process of interest. In our example such process is double peripheral central production of particles, as depicted in Fig. 3. The simplest modification which can be introduced is to allow for a dissociation of the leading particles i.e. by replacing each of them by intermediate objects of variable mass which are subsequently decayed using TDecay. In order to efficiently generate a triple peripheral process with production of two central objects and two leading particles, Fig. 4, one could construct TEventMaker2toN based on kinematics of $2\to4$ using formulae derived in [10]. Again, one should remember about the factorization assumption of the matrix element we have made. It excludes possibility of the interference of identical final state particles which belong to different decaying objects. For this reason in Ref. [6] $2\to3$ kinematics of TEventMaker2toN was employed. Although at sufficiently high invariant mass of the four-pion system $2\to4$ kinematics would be acceptable approximation resulting in much better generator efficiency.

5 GenExLight – performance test

The last performance test compares the spherical decay given by the TDeacy class and non-spherical decay provided by the TEventMaker2toN class in the presence of non-spherical cuts. It shows that when an incorrect phase space generation scheme is chosen then the adaptivity will not significantly improve the convergence unless the finer exploration will be setup.

For GenExLight performance test we select reaction $p_a(p)+p_b(p)\to p_1(p)+p_2(p)+p_3(\pi^+)+p_4(\pi^-)$ at $\sqrt{s}=\sqrt{(p_a+p_b)^2}=200$ GeV with the unit matrix element and the following cuts:

- $|t_1|=|(p_a-p_1)^2|<t_{cut}$, $|t_2|=|(p_b-p_2)^2|<t_{cut}$;
- $M_{3+4}\in[0.280\text{ GeV};100\text{ GeV}]$;
- $|y_{3+4}|<10$,

where t_{cut} is a parameter. The cuts restrict the four-momentum transfers $a\to 1$ and $b\to 2$ and indirectly transverse momenta of particles 1 and 2, as well as, the variables directly generated in TEventMaker2toN. In addition, the mass and rapidity of the centrally produced object are restricted, so the resulting kinematics resembles that of the typical double peripheral process e.g. [10]. We compare performance of GenExLight and TDecay in terms of exploration t_{expl} and generation t_{gen} times of a single event with unit weight. The FOAM exploration parameters are set to 10^4 cells and 10^4 samples in each cell and the number of generated events is 10^5. The results are presented in Table 3. The table shows that for large values the cut $t_{cut}>700\,\text{GeV}^2$ i.e. when the final state sphericity is not too small, both algorithms converge nd produce results that differ by less than 3%. However, agreement within uncertainty indicated by FOAM is reached only for $t_{cut}>1000\,\text{GeV}^2$.

Table 3: Efficiency comparison of GenExLight (GEL) and TDecay (TD) with identical FOAM parameter settings (see text). $t_{expl\cdots}$ is the exploration time and $t_{gen\cdots}$ is the generation time of single event the with unit weight. Results were obtained for $\mathcal{M}=1$ and $\sqrt{s}=200$ GeV, cuts and the process described in the text. Tests were performed using Intel Xeon E5-2680 v3 @ 2.5 GHz processor.

t_{cut}[GeV]	σ_{GEL}	$t_{explGEL}$[s]	σ_{TD}	t_{explTD}[s]	t_{genTD}/t_{genGEL}
200	4.469	101.73	1.399	301.76	0.024
300	9.53	90.5	6.54	295.04	0.040
400	1.540	102.11	1.254	321.5	0.046
500	2.109	99.41	2.501	320.69	0.054
600	2.69	91.18	3.710	306.96	0.07
700	3.298	83.93	4.99	303.1	0.07
800	6.73	109.96	6.52	298.55	0.09
900	8.44	122.43	8.25	272.06	0.13
1000	1.02	117.91	1.01	266.41	0.13

Table 4: Efficiency comparison of GenExLight (GEL) and TDecay (TD) with two different FOAM parameter n_{cell} and n_{sampl} settings. $t_{expl\cdots}$ is the exploration time and $t_{gen\cdots}$ is the generation time of single event with the unit weight. Results were obtained for $\mathcal{M}=1$, $\sqrt{s}=200$ GeV, cuts and the process described in the text. Tests were performed on an Intel Xeon E5-2680 v3 @ 2.5 GHz processor.

$n_{cell}\times n_{sampl}$	σ_{GEL}	$t_{explGEL}$ [min]	σ_{TD}	t_{explTD}[min]	t_{genTD}/t_{genGEL}
$10^4\times10^4$	4.469	1.7	1.399	5.02	0.024
$10^5\times10^5$	4.468	94.5	4.30	349.8	0.04

For smaller values of t_{cut}, i.e. more asymmetric final state, TDecay program performs poorly and to get a meaningful result the FOAM parameters have to be set to much more cells and samplings/cell, e.g., for $t_{tcut}=200\text{GeV}^2$ with $n_{cell}=10^5$, $n_{sampl}=10^5$ the TDecay-GenExLight convergence is achieved within indicated statistical uncertainty (see Table 4). The exploration time for TDecay increased by a factor 70 while the generation time of a single event with an unit weight is 25 times longer than in the GenExLight case. GenExLight performs perfectly with $n_{cell}=10^4$, $n_{sampl}=10^4$ FOAM settings even for the realistic experimental cut $t_{cut}<1$. It should be noted that the uncertainties indicated by FOAM are not reliable when the numbers of cells and samplings/cell are too small. In practice, one should use the generator with a few set-ups of exploration increasing the number of cell until results get stable.

6 Conclusions

In this paper a simple and robust toolbox for phenomenological studies of exclusive hadronic reactions was discussed. Its basic element, the TDecay class, the module for the specialized self adapting event generators, was described. The TDecay is a significant modification of the classic phase space generator GENBOD (implemented as the TGenPhaseSpace in ROOT) interfaced with the adaptive sampling integrator/generator FOAM.

A practical example of its usage in construction of GenExLight, the program designed for integration/generation of the central production in double peripheral process, was presented, together with comparative performance tests. It is aimed at prototyping of new models in High Energy Physics and ease of its embedding in software available in large experiments of particle physics. The tests show not only the power of adaptive sampling, but also its limitations. It appears that for purely practical reasons (the computation time) appropriate change of variables which enables the program to limit the number of empty cells is often a necessity.

Acknowledgments

We would like to thank Piotr Lebiedowicz and Antoni Szczurek for discussions. This work was supported in part by Polish National Science Center grant: UMO-

2015/17/D/ST2/03530. J. J. C. was partially supported by Polish National Science Center grant: UMO-2015/19/B/ST2/00989. The calculations presented in this article were supported in part by PLGrid and Cracow Cloud One infrastructures. R. K. was partially supported by GACR Grant 17-19437S, and the grant MUNI/A/1138/2017 of Masaryk University.

A Appendix: Technical description of TDecay class

The TDeacy generates n-body decay events $P \to p_1, p_2, \cdots, p_n$ and is interfaced with FOAM, the adaptive sampling Monte-Carlo generator/integrator. It is also an utility class which can be combined with dedicated sampling algorithms, as described in Section 4. The program can be downloaded from [16] in the directory TDecayTFoam. The directory contains the TDecay class (header and implementation files), the main file as well as a Makefile.

The TDecay interface consists of three functions.

- Function that specifies the decay configuration is

  ```
  Bool_t SetDecay (TLorentzVector &P, Int_t nt, const Double_t *mass)
  ```

  where P is the four-momentum of the decaying particle, *nt*, *mass* – the number of final particles and their masses in an array.

- Function

  ```
  Double_t  Generate (std::queue<double> &rnd)
  ```

  derives the four-momenta and the phase space weight (see B in Eq. (4.2)) from *rnd* – standard C++ library queue of $3 \times nt - 4$ double precision random numbers. The weight contains also the factor $\frac{1}{2\pi}$ as it is indicated in Eq. (2.6). The four-momenta of the final particles can be retrieved by the function

  ```
  TLorentzVector *GetDecay (Int_t n)
  ```

  where n is the index of n-th final particle starting from 0.

The basic parameters defining reaction are at the top of the main file.

```
//center of mass energy
const double tecm = 200.0;

//incoming particles
const int Nip = 2;
```

```
//outgoing particles
const int Nop = 4;

//pdg codes of particles starting from index 1
int idIn[Nip+1] = {0, 2212,2212};      //PDGID incoming particles
int idOut[Nop+1] = {0, 2212,2212,211,-211}; //PDGID outgoing particles
```

The TDensity class inherits from TFoamIntegrand. It is required by FOAM, defines how a set of random numbers is transformed into the value of integrand; for details of this structure see [4]. The most important method of this class is the TDensity::Density function which uses TDecay object in order to generate events written in pb and pf tables, place experimental cuts and calculate integrand function – matrix element for the given reaction multiplied by phase space weight.

At the beginning of the main function there is a setup for FOAM including the number of events

```
long NevTot   =     100000;   // Total MC statistics
```

and the exactness of the exploration procedure

```
Int_t  nCells   =     10000;   // Number of Cells
Int_t  nSampl   =     10000;   // Number of MC events per cell in build-up
```

The next section defines sample histograms and ROOT tree used to collect data. After this part the main integration loop follows. First part of the loop is the generation of the event and associated weight

```
 // generate MC event
FoamX->MakeEvent ();
FoamX->GetMCvect ( MCvect);
FoamX->GetMCwt ( MCwt );
```

then the histograms and tree are filled with the generated event and then the integrand is added to the sum of weights generated in the previous rounds of the loop. At the end of the program the booking actions are taken – printing the integral, saving histograms, tree and freeing memory from the allocated objects.

The program is supplied with Makefile [15], with the following commands:

- make run – compile and run the program;
- make clean – clean directory from executables;
- make cleanest – clean directory from executables, eps figures and root tree with events;

B Appendix: $2\to 3$ kinematics in TEventMaker2toN

In this Appendix we show the steps transforming the part B of the cross section formula Eq. (4.2) to Eq. (4.4) which is kinematical basis of the TEventMaker2toN class. The calculations are made in the reaction centre of mass ($\vec{p}_a+\vec{p}_b=0$) and the beam is aligned along OZ axis. Integration over the transverse momentum of the central object $\vec{P}_{Mt}$ gives

$$B=\int (2\pi)^4\delta(\sqrt{s}-E_1-E_2-E_M)\delta(p_{1z}+p_{2z}+P_{Mz})\frac{d^3p_1}{(2\pi)^3 2E_1}\frac{d^3p_2}{(2\pi)^3 2E_2}\frac{dP_{Mz}}{(2\pi)^3 2E_M}, \quad \text{(B.1)}$$

with the constraint $\vec{P}_{mt}=-(\vec{p}_{1t}+\vec{p}_{2t})$. In the next step, integration over p_{2z} gives

$$B=\int (2\pi)^4\delta(\sqrt{s}-E_1-E_2-E_M)\frac{d^3p_1}{(2\pi)^3 2E_1}\frac{d^2p_{2t}}{(2\pi)^3 2E_2}\frac{dP_{Mz}}{(2\pi)^3 2E_M}, \quad \text{(B.2)}$$

with the constraint $p_{1z}+p_{2z}+P_{Mz}=0$. In order to perform the last integration over p_{1z} the argument of the remaining Dirac delta has to be expressed in terms of this variable. Let us introduce

$$f(p_{1z})=\sqrt{s}-\sqrt{p_{1t}^2+p_{1z}^2}-\sqrt{p_{2t}^2+p_{2z}^2}-E_M, \quad \text{(B.3)}$$

where $p_{2z}(p_{1z})=-(p_{1z}+p_{Mz})$. Now $\delta(\sqrt{s}-E_1-E_2-E_M)=\delta(f(p_{z1}))$. The solution of the quadratic equation $f(p_{1z})=0$ generates two solutions discussed in Eq. (4).

Using the standard formula, the Dirac delta can be transformed in the following way

$$\delta(f(p_{z1}))=\sum_{p'_{1z}\in\{p''_{1z}:f(p''_{z1})=0\}}\frac{\delta(p_{1z}-p'_{z1})}{J}, \quad \text{(B.4)}$$

where $J=\left|\frac{p_{1z}}{E1}+\frac{p_{2z}}{E_2}\right|$. At this stage integration over p_{1z} is easily performed and results in

$$B=\int\sum_{p'_{1z}\in\{p''_{1z}:f(p''_{z1})=0\}}\frac{(2\pi)^4}{J}\frac{d^2p_{1t}}{(2\pi)^3 2E_1}\frac{d^2p_{2t}}{(2\pi)^3 2E_2}\frac{dP_{Mz}}{(2\pi)^3 2E_M}. \quad \text{(B.5)}$$

Last part is transformation to polar variables in $\vec{p}_{1t}$ and $\vec{p}_{2t}$ and to $P_{Mz}\to y_M$, where y_M is the rapidity of the central object. That gives

$$B=\int\sum_{p'_{1z}\in\{p''_{1z}:f(p''_{z1})=0\}}\frac{(2\pi)^4}{J}\frac{|\vec{p}_{1t}|d|\vec{p}_{1t}|d\phi_1}{(2\pi)^3 2E_1}\frac{|\vec{p}_{2t}|d|\vec{p}_{2t}|d\phi_2}{(2\pi)^3 2E_2}\frac{dy_M}{(2\pi)^3 2}, \quad \text{(B.6)}$$

where ϕ_1 and ϕ_2 are polar angles.

C Appendix: Technical description of GenExLight program

In this appendix the interface of the classes in Fig. 2 is described. The constructors of TDensity and TEventMaker2toN are as follows

```
TEventMaker2toN (double tecm, double p_min, double p_max, double y_min,
double y_max, int nop, double mass_min, double mass_max, double* mass,
int* idIn, int* idOut, TLorentzVector* pb, TLorentzVector* pf, int isol)

TDensity (double tecm, double p_min, double p_max, double y_min,
double y_max, int nop, double mass_min, double mass_max, double* mass,
int* idIn, int* idOut, TLorentzVector* pb, TLorentzVector* pf, int isol);
```

where the meaning of the variables is given in Table 5.

Table 5: Variables in constructors of TEventMaker2toN and TDecay.

Variable	Meaning
tecm	centre of mass energy $\sqrt{(pb_1+pb_2)^2}$;
p_min, *p_max*	the range of transverse momenta of leading particles 1 and 2;
y_min, *y_max*	the range of rapidity for the central object;
nop	number of outgoing particles;
mass_min, *mass_max*	range of the mass of central object;
mass	mass matrix for central particles; first particle has index 1; the index 0 is unused;
idIn	table of PDG identifiers for incoming particles pb_1 and pb_2;
idOut	table of PDG identifiers for outgoing particles $pf_1,\cdots,pf_N$;
pb	array of four momenta of incoming particles;
pf	array of four momenta of outgoing particles;
isol	selecting solution for kinematics: 0 – nonbouncing, 1 – bouncing

These constructors get a few parameters and some interface/wrapping classes that allow to pack all the arguments into a single data structure can be designed. However we limited ourselves to such simple approach as it simplifies the code structure required for fast prototyping and further embedding in larger software.

Generator can be started and cleaned using GNUMake system [15] with the following commands:

- make run – compile and run generator;
- make clean – clean executables and intermediate files;
- make cleanests – clean executables and results of simulation;

Generator produces the following output files:

- events.root – ROOT file with events as TLorentzVector;
- CM.eps – plot of central mass;
- etaPhi.eps – plot of η vs ϕ for all particles in the events;
- rapidity.eps – plot of particles;
- histograms.root – histograms in ROOT file;

References

[1] R. Hagedorn, *Relativistic kinematics*, Literary Licensing, LLC 2012.
[2] F. James, *Monte Carlo Phase Space*, CERN 68-15, 1968.
[3] S. Jadach, Comput. Phys. Commun. 9 297 (1975).
[4] S. Jadach, Comp. Phys. Commun. 152 55 (2003).
[5] B. P. Kersevan, E. Richter-Was, Eur. Phys. J. C39 439 (2005).
[6] R. Kycia, P. Lebiedowicz, A. Szczurek, J. Turnau, Phys. Rev. D 95, 094020 (2017); arXiv:1702.07572 [hep-ph].
[7] R. A. Kycia, J. Chwastowski, R. Staszewski, J. Turnau, Commun. Comput. Phys. 24 860 (2018).
[8] P. Lebiedowicz, A. Szczurek, Nucl. Phys. A826 101 (2009).
[9] P. Lebiedowicz, A. Szczurek, Acta Phys. Polon. Supp. 8 835 (2015).
[10] P. Lebiedowicz, A. Szczurek, Phys. Rev. D 81 036003 (2010).
[11] T. Ohl, Comp. Phys. Commun. 120 13 (1999).
[12] O. Pene, A. Krzywicki, Nuclear Physics B12 415 (1969).
[13] H. Pilkuhn *The interactions of hadrons*, North Holland Publishing Company, 1967.
[14] T. Sjöstrand, S. Mrenna and P. Skands, PYTHIA 6.4 Physics and Manual, JHEP**05** 026 (2006).
[15] GNU Make, `http://www.gnu.org/software/make/`
[16] Program repository, `https://github.com/rkycia/GenExLight`
[17] ROOT software, `http://root.cern.ch/drupal/`