Pymcabc: A particle physics toy toolbox for the ABC model

A. Desai

Department of Physics, The University of Adelaide, Adelaide, SA 5005, Australia,
e-mail: amanmukeshdesai@gmail.com

Received 8 January 2024; accepted 26 March 2024

We present the `pymcabc` software, which is a Monte Carlo event generator for the ABC toy model. The ABC model consists of three scalar particles of arbitrary masses. The only interaction among these particles occurs when all three of them are present together. The `pymcabc` software can calculate all the leading-order cross-sections as well as decay widths within the ABC model and it can simulate all the scattering processes within the ABC model. Moreover, it simulates the decays associated with the heavy-particle final state, leading to a $2 \to 3$ or a $2 \to 4$ type final states within the ABC model. We also apply toy detector effects to simulate the detector response of a toy tracker for three-momentum measurements and a toy calorimeter for energy measurements. Using the results of the `pymcabc` software, we also illustrate some well-known physics analyses techniques such as the analysis of the lineshape of a heavy propagator and the recoil mass reconstruction technique.

Keywords: Monte Carlo.

DOI: https://doi.org/10.31349/RevMexFisE.22.010207

1. Introduction

Monte Carlo Simulation tools are essential for the physics program at particle accelerators and detectors. These tools are widely used by the current experimental collaborations such as the ATLAS and the CMS experiments at the Large Hadron Collider (LHC), at CERN, as well as by physicists working in the planning of Future Collider experiments. These tools play an important role in designing experiments as well as in corroborating experimental results. However, it is difficult for a novice to understand the working of these tools. The complexity involved is two-fold: theoretically, the full Lagrangian of the Standard Model of particle physics consists of several interacting terms, which lead to complicated calculations for particle physics observables, and, computationally, the Monte Carlo software is complicated in a way that benefits the software to accommodate many Standard Model processes as well as Beyond the Standard Model processes. In collider experiments, for instance, at LHC, CERN, a single collision would lead to many particles in the final state, and moreover, the number of collisions per second is high. In order to decipher the event of interest there are various Monte Carlo simulations software available, for example, `MadGraph5_aMC@NLO` [1], `Pythia8` [2], etc.

The difficulties faced in teaching the method of computing quantum mechanical amplitudes (or matrix elements) for a given physics process through the technique of Feynman diagrams are eased by particle physics textbooks by introducing a pedagogical model viz. the ABC model. The ABC model assumes that the world consists of three spin-zero particles and the only possible interaction of these particles is when all three particles participate in a given process [3,4]. This model has also been used to introduce canonical quantum field theory, dimensional regularization, as well as renormalization [4,5].

In this paper, the idea of using the ABC model is extended as a means to introduce a Monte Carlo Event Generator as well as to give a glimpse of toy-phenomenological analysis of the ABC model. The `pymcabc` software[i] introduced here can be used as a tool to calculate both the total cross-sections as well as generate simulations of ABC scattering processes. Furthermore, it can be used as a tool to simulate detector effects on final-state particles and the subsequent decay of any heavy particles in the final state. The decay simulation allows the software to populate the final state with three or four particles and allows the user to explore toy detector effects in a multi-particle final state. The use of the ABC model is motivated since the three particles are real scalars with spin zero, which simplifies the computation of the quantum mechanical amplitude of a process.

Given that the three particles are the same (except for their mass), it allows us to generalize the `pymcabc` software to include all the possible ABC scattering processes as well as the decay processes.

A typical workflow in High Energy Physics involves starting from the Lagrangian of the theory and implementing the model in specific file formats that are compatible with computer software for simulating and calculating cross-section of a process. One then passes these events through detector simulation to take into account the effect of detector material and, finally, performs analysis with the events generated by Monte Carlo Simulation. This typical workflow has been presented in here for the ABC toy model. We have attempted to decipher this workflow and made an attempt to allow anyone with some basic knowledge of particle physics to be able to understand the workflow.

The organization of this paper is as follows. In Sec. 2, we present the salient features of the ABC model and discuss the theory involved in the computation of observables. In Sec. 3, we introduce a Monte Carlo software called `pymcabc`^{ii} and discuss the method used in preparing the software. The first steps involved in installing and testing the software are provided in Sec. 3.1. We compare the performance of the software with theory in Sec. 3.2 and present some benchmark calculations with this software in Sec. 3.3. In Sec. 4, we discuss some physics cases, such as the Event analysis in Sec. 4.1.1 for a tu-channel process. Then, in Sec. 4.1.2, we consider the event analysis of an st-channel process with decay products in the final state. In Sec. 4.2, the analysis of the Lineshape of a heavy propagator in an s-channel is given and in Sec. 4.3, we present the recoil mass reconstruction technique that can be explored using cross-section calculated by, and the events generated with, `pymcabc`. Finally, in Sec. 5, we have suggested some exercises to explore further the ABC model using the `pymcabc` software.

2. Theory of the ABC Model

The ABC model consists of three scalar spin-zero particles, called A, B, and C, which are distinguishable owing to differences in their masses. The Lagrangian for the ABC model is [3]:

$$\mathcal{L} = \frac{1}{2}\partial_\mu\phi_A\partial^\mu\phi_A + \frac{1}{2}\partial_\mu\phi_B\partial^\mu\phi_B + \frac{1}{2}\partial_\mu\phi_C\partial^\mu\phi_C - \frac{1}{2}m_A^2\phi_A^2 - \frac{1}{2}m_B^2\phi_B^2 - \frac{1}{2}m_C^2\phi_C^2 - ig\phi_A\phi_B\phi_C, \quad (1)$$

where ϕ_A, ϕ_B, and ϕ_C represent real scalar fields of particles A, B, and C, respectively. In this paper, we adopt the natural units viz. $\hbar = c = 1$, so the dimension of space and time are inverse energy and the dimension of the scalar fields is energy. The coupling constant, g, of the model has the dimension of energy. Note that each scalar particle, if free, would follow the Klein-Gordon Equation.

The ABC Model's Lagrangian as given in Eq. (1) suggests the following Feynman rules [3,4]:

- The external lines, that is, the incoming and outgoing scalar particles, are assigned a factor of unity.
- The vertex has a factor of $-ig$.
- The propagator (internal line with both ends connected) contributes a factor of $1/[q_j^2 - m_j^2 + i\Gamma_x m_x]$, where Γ_x represent the 'decay width' of the propagator. The reason for the additional factor, $i\Gamma_x m_x$, is to avoid divergences in the computation of the total cross-section. The propagator need not be on-shell, it is a virtual state, not physically accessible.

We may note that external lines represent physical states for incoming or outgoing particles whereas internal lines represent virtual states. Only connected diagrams represent physically possible processes. The particles interact only when all three are participating in a given process. This is indicated by the last term in the Lagrangian. This Lagrangian has no self-interacting terms, which, by the way, do occur in the Standard Model of particle physics. Higher-order derivatives terms lead to non-local interactions and are thereby avoided in quantum field theory.

The Feynman rules of the ABC model are used to evaluate the matrix elements (quantum mechanical amplitudes, which give the dynamics) of a given ABC process. Combining appropriately the squared matrix elements with the phase space (which gives the kinematics) results in the cross-section of a process or the decay width of a particle. The Feynman diagrams, along with the Feynman rules, are a mnemonic to write down the amplitude of a given process.

The scattering processes that one considers in particle physics occur via the so-called different 'channels', mainly the s-channel, t-channel, and u-channel. As an example, consider a process $1\ 2 \rightarrow X \rightarrow 3\ 4$, where the numbers 1 to 4 represent the different particle states (external lines) and X represents a propagator (internal line) of the process. In such cases, the possible interaction process can proceed via an s-channel, a t-channel, or a u-channel. The channel in which the initial particles 1, 2 share a common vertex with one end of the internal line of the propagator, X, and the final state particles 3, 4 share the other end of the internal line, another common vertex, with the particle X, is termed as an s-channel process. In this channel, the four-momentum of the propagator, q_X^μ, is given by $q_X^\mu = p_1^\mu + p_2^\mu$, where p_1^μ and p_2^μ are the four-momentum of particles 1 and 2, respectively. The channel in which particles 1, 3 share a vertex with the propagator X and, particles 2, 4 share the other vertex with X, is termed as a t-channel process. For the t-channel, the propagator has four-momentum given as $q_X^\mu = p_1^\mu - p_3^\mu$. A u-channel process is just like a t-channel process but with the particles 3 and 4 exchanged and, therefore, the propagator's four-momentum is given as $q_X^\mu = p_1^\mu - p_4^\mu$. However, note that the u-channel process is possible only when the final state particles are identical. The variables $s,\ t$, and u are also known as Mandelstaam variables, and they satisfy the useful relation $s + t + u = m_1^2 + m_2^2 + m_3^2 + m_4^2$, where m_i is the mass of the ith particle.

The representative Feynman diagrams for the s-channel, t-channel, and u-channel in the context of the ABC model are given in Fig. 1. Considering all combinations of processes in the ABC model and using the three possible channels, one

TABLE I. List of possible processes in the ABC model. A particle denoted with an asterisk refers to a virtual intermediate state (propagator).

s- and t- channel processes	t- and u- channel processes
$A\ B \rightarrow C^* \rightarrow A\ B$	$A\ A \rightarrow B^* \rightarrow C\ C$
$A\ C \rightarrow B^* \rightarrow A\ C$	$A\ A \rightarrow C^* \rightarrow B\ B$
$B\ C \rightarrow A^* \rightarrow B\ C$	$B\ B \rightarrow C^* \rightarrow A\ A$
	$B\ B \rightarrow A^* \rightarrow C\ C$
	$C\ C \rightarrow A^* \rightarrow B\ B$
	$C\ C \rightarrow B^* \rightarrow A\ A$

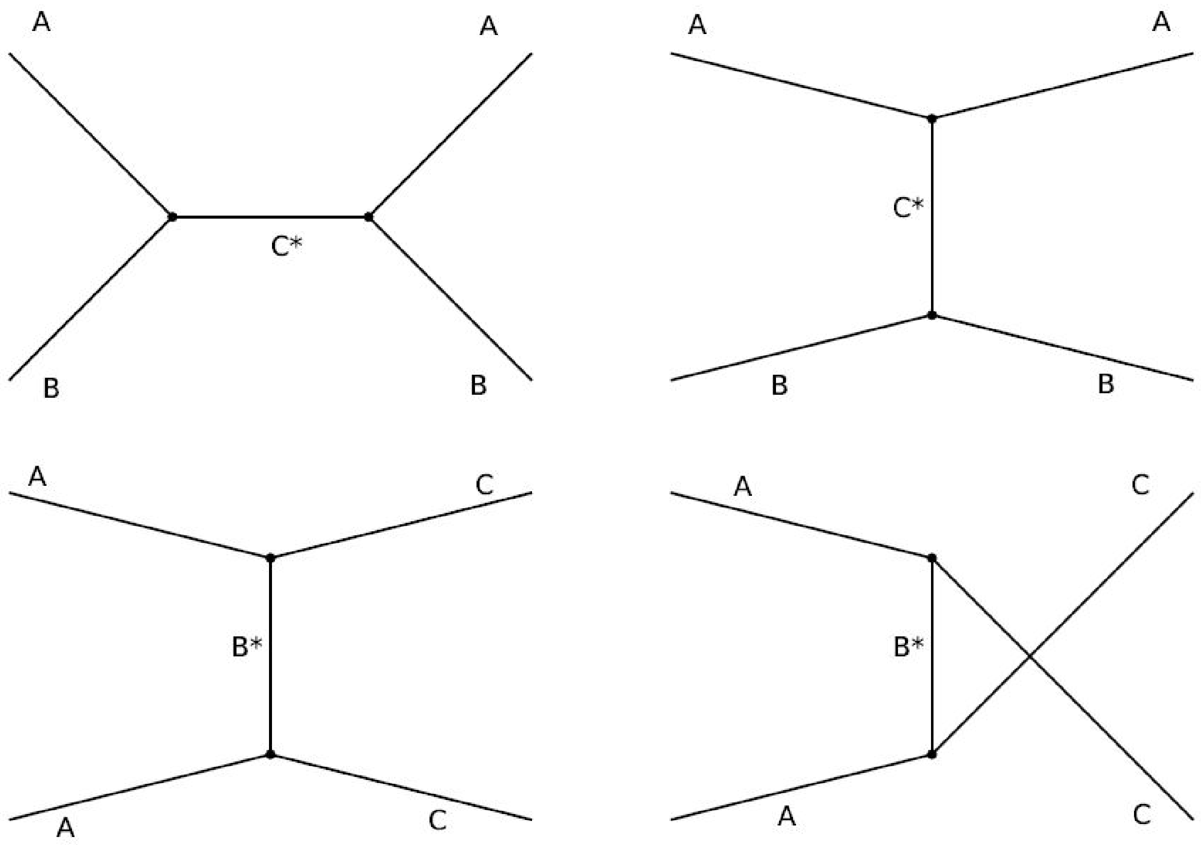


FIGURE 1. Representative Feynman diagrams for a) s-channel, b) t-channel for the process $A\ B \to C^* \to A\ B$ and c) t-channel, d) u-channel for the process $A\ A \to B^* \to C\ C$. The time axis goes from left to right. (Particles denoted with an asterisk refer to virtual intermediate states).

finds that only nine unique processes are possible (at the so-called tree-level), which are listed in Table I. A given process in the ABC model can proceed via a combination of either (i) s- and t- channels as shown in Fig. 1a)-b) or (ii) t- and u-channels as shown in Fig. 1c)-d). Particles denoted with an asterisk denote virtual intermediate state (propagator). Note that energy-momentum is conserved at each vertex of the Feynman diagram.

For the s- and t-channels scattering processes, the differential cross-section in terms of the matrix elements [3] in the center-of-momentum frame of reference is $\mathbf{p_1} = -\mathbf{p_2}$:

$$\frac{d\sigma}{d\Omega} = \frac{|\mathbf{p}_f|}{64\pi^2 E_{cm}^2 |\mathbf{p}_i|} |\mathcal{M}_s + \mathcal{M}_t|^2, \tag{2}$$

where, $E_{\rm cm}$ is the center-of-momentum energy of the reaction, $|\mathbf{p}_i|$ is the magnitude of the three-momentum of the initial particle before scattering, and $|\mathbf{p}_f|$ is the magnitude of the three-momentum of the final particle after scattering. $\mathcal{M}_s$ and $\mathcal{M}_t$ represent the matrix elements (amplitudes) for the s-channel and the t-channel, respectively. In the above equation, the term,

$$|\mathcal{M}_s + \mathcal{M}_t|^2, \tag{3}$$

represents the absolute square of the sum of the matrix elements for the s- and t- channel processes, and the term

$$\frac{|\mathbf{p}_f|}{64\pi^2 E_{cm}^2 |\mathbf{p}_i|}, \tag{4}$$

is the phase space for the process in which two initial particles scatter into two final state particles, *i.e.*, $(2 \to 2)$ process. As we see in Table I, the scattering processes with the t- and u-channel configuration consist of identical final state particles and, therefore, the available phase space needs to be additionally divided by a factor of two to avoid double counting. Thus, for the t- and u-channel processes:

$$\frac{d\sigma}{d\Omega} = \frac{|\mathbf{p}_f|}{128\pi^2 E_{cm}^2 |\mathbf{p}_i|} |\mathcal{M}_t + \mathcal{M}_u|^2. \tag{5}$$

Here, $\mathcal{M}_u$ is the matrix element corresponding to the u-channel. The matrix elements $\mathcal{M}_s, \mathcal{M}_t$, and $\mathcal{M}_u$ are obtained using the Feynman rules listed above and are found to be [4]:

$$\mathcal{M}_s = \frac{g^2}{(p_1+p_2)^2 - m_x^2 + i\Gamma_x m_x} = \frac{g^2}{E_{cm}^2 - m_x^2 + i\Gamma_x m_x}, \tag{6}$$

$$\mathcal{M}_t = \frac{g^2}{(p_1-p_3)^2 - m_x^2 + i\Gamma_x m_x} = \frac{g^2}{m_1^2 + m_3^2 - m_x^2 - 2\mathbf{p}_1 \cdot \mathbf{p}_3 + i\Gamma_x m_x}, \tag{7}$$

$$\mathcal{M}_u = \frac{g^2}{(p_1-p_4)^2 - m_x^2 + i\Gamma_x m_x} = \frac{g^2}{m_1^2 + m_4^2 - m_x^2 - 2\mathbf{p}_1 \cdot \mathbf{p}_4 + i\Gamma_x m_x}, \tag{8}$$

where m_x is the mass of the propagator, p_1, p_2, p_3, and p_4 are the four-momentum of the initial particles 1 and 2, and final particles 3 and 4, respectively (bold represents three-momentum). The s-channel matrix element, $\mathcal{M}_s$, in Eq. (6), is independent of the masses of particles scattered in the process and depends only on the propagator's mass. On the other hand, the matrix elements for the t-channel, $\mathcal{M}_t$, in Eq. (7), and u-channel, $\mathcal{M}_u$, in Eq. (8), relies on the initial and final particle's masses as well as the propagator's mass.

The total cross-section of a process is then obtained by integrating the differential cross-section $d\sigma/d\Omega$ over the solid angle $d\Omega$. The total cross-section obtained thus far is also known as the *Leading-Order* cross-section, as the matrix element computation only used the so-called tree-level diagrams (Fig. 1). A higher-order process leads to additional Feynman diagrams containing particle loops and vertices for the given process. The 'order' is determined by the powers of the coupling constant (each vertex has a factor of g) in the amplitude of the given process. In general, the total cross-section is given as:

$$\sigma_{\text{total}} = \sigma_{\text{LO}} + \sigma_{\text{NLO}} + \sigma_{\text{NNLO}} + ..., \tag{9}$$

where σ_{N^iLO} refers to the total cross-section at the $(i+1)$-th order. In this paper, higher-order processes are ignored as they only complicate the ABC Model calculations and they will necessitate the ideas of renormalization and add extra parameters. Moreover, the higher-order processes increase the number of diagrams that needs to be evaluated. For instance, at NLO (next-to-leading order) in the ABC model, one has to deal with 16 diagrams for a given tu- process, whereas, on the other hand, one deals with only two diagrams at the LO (leading-order) for the tu process [3]. It must, however, be noted that the higher-order processes are important in particle physics and considerable attention is given to automating the higher-order calculations. In the ABC model, one could reduce the contributions of higher-order processes by having the coupling constant $g < 1$ as $\sigma_{N^iLO} \sim g^{2i+2}$. The coupling constant in this paper is, however, set to unity (to simplify calculations). By the way, the ABC model is super-renormalizable [4].

In the ABC model, decays are possible only if one of the three particles has a mass greater than the sum of the masses of the other two particles, since all three particles need to participate in the interaction. Owing to this, only the heaviest particle can decay to the other two lighter particles. Thus, processes like A→B B, etc., are forbidden. As an example, consider the decay of particle A. This decay process is possible only if the mass of particle A is greater than the sum of the masses of particles B and C.

Having gone through the theoretical aspects of the ABC model, we would like to test this model presuming it were experimentally viable. One would need to design the experiment and test the various aspects of the theory. It would be useful to simulate the possible scenarios before carrying out the experiment. In fact, this is what is done in experiments say, at LHC, CERN, which are designed to test the Standard Model or search for physics Beyond the Standard Model.

The `pymcabc`-software introduced in this paper can be taken as a warm-up exercise before tackling the software designed for the LHC, for instance. This is similar to dealing with the Feynman diagrams in the ABC model before dealing with them in QED theory. In the next section, we illustrate some benchmark calculations used to validate the software and also provide a set of commands that are used to run the software.

3. `pymcabc`: Software for the ABC model simulation

The `pymcabc` is a python software developed to understand and simulate the physical processes that can occur in the ABC model. Although it is academic, nevertheless, the ideas presented here are also useful in introducing the workflow carried out by experiments at facilities such as the LHC at CERN. This software comprises dedicated modules used for the identification of input processes, generation of Feynman diagrams, calculations of matrix elements, decay widths of the massive particle, differential and total cross-sections, generating Monte Carlo events, decaying of the heavy particle in final states, applying detector effects, saving events in `.root` or `.csv` format (only `ROOT` and `CSV` formats are supported for now), and for automatic plotting of basic kinematic variables.

The software requires the user to provide the process as a `string` (text), the masses of particles as `float` (real number), and the initial three-momenta as `float` (real number). The software identifies the channel in which the process can proceed and can draw the relevant Feynman diagrams. It then identifies the propagator of the process and assigns masses to the incoming, outgoing, and propagator particles. For example, when the outgoing particles are identical (A A or B B or C C), the software assigns the process as a tu-type process and, otherwise, the process is assigned an st-type process.

For the cases when a heavy particle exists in the final state, the software subsequently also identifies the possible decay channels.

Once the identification of the process is accomplished, the software can be used to evaluate the total cross-section of a process. The total cross-section computation is done by first evaluating the differential cross-section based on Eq. (2) for the $s,\ t$-channel and Eq. (5) for the $t,\ u$-channel. Note that for consistency in computation, and to avoid division by a zero in the denominator, we modify the matrix element by adding a complex term in its denominator viz. $i\Gamma_x m_x$, where Γ_x is the decay width of the mediator, and m_x is the mass of the mediator.

To perform integration over $d\Omega$, we first generate a set of uniform random numbers (r) in the range [0,1], and using these numbers, we define $\cos\ \theta$ as $\cos\ \theta = 2r - 1$. Once $\cos\ \theta$ is defined, we use it to find the differential cross-section at each $\cos\ \theta$. Each result thus obtained is stored as a "weight". We also additionally store the maximum weight $W_{\max}$, so that $W_{\max} = \max(0, W_i)$. The total cross-section is defined as:

$$\sigma = \frac{\Sigma W_i}{N}, \tag{10}$$

where N is the number of points used for the integration process. The variance in the integration, assuming that the average value of the integrand in the problem tends to be a Gaussian distribution, is then defined as:

$$\sigma_{\text{variance}} = \sqrt{\left(\frac{\Sigma W_i^2}{N}\right) - \left(\frac{\Sigma W_i}{N}\right)^2}. \tag{11}$$

The error in σ is defined as (just like the standard error on the mean):

$$\sigma_{\text{error}} = \frac{\sigma_{\text{variance}}}{\sqrt{N}}. \tag{12}$$

Note that the units for σ, and its error, is GeV^{-2} in the Natural Unit. However, the units commonly used to represent the total cross-section is picobarns (pb), which has a dimension of area. [1 barn = 10^{12} pb= 10^{-28} m^2] To convert the units (GeV^{-2}) into the units of area, for instance, to (pb), we revert to the S.I. units for $\hbar = 1.054 \times 10^{-34}$ J s and $c = 3 \times 10^8$ m/s, which were earlier set to unity in the system of Natural Units. For instance:

$$\begin{aligned}\sigma[pb] &= \sigma[\text{GeV}^{-2}] \times (\hbar c[\text{GeV}fm])^2 \\ &= \sigma[\text{GeV}^{-2}] \times 3.89 \times 10^8[\text{GeV}^2 pb].\end{aligned}$$

In addition to computing the total cross-sections for the st- and tu-channels, the software also allows the computation of the total cross-section for a specific s-, t-, or u- channel, even though the results are not guaranteed to be physical. The procedure to evaluate individual channels is as follows: if one wants to compute a specific total cross-section due to the s-channel, say, then we start with the full matrix element $\sigma \propto |\mathcal{M}_s + \mathcal{M}_t|^2$, and expanding it, we find $\sigma \propto |\mathcal{M}_s|^2 + |\mathcal{M}_t|^2 + 2|\mathcal{M}_s\mathcal{M}_t|$. We then remove the contributions of $|\mathcal{M}_t|^2$ as well as the interference term $2|\mathcal{M}_s\mathcal{M}_t|$.

Having evaluated the total cross-section of a process, we now consider the generation of events of a given process. An event refers to a single scattering process that leads to final state particles. Generation of events refers to the simulation of a given event, which involves determining the kinematics of the final state particles of a given process that is consistent with the available phase space of a process as well as the governing dynamics. The generation of Monte Carlo events is based on the acceptance-rejection method, which is a Monte Carlo technique that proceeds as follows (for the general method see, for example, [6]):

- First, we define $\cos\ \theta$ as $2r_1 - 1$ where r_1 is a uniform random number between [0, 1]. Using this value of $\cos\ \theta$, we evaluate the differential cross-section and call it W_i.
- We then generate another uniform random number r_2.
- Using the $W_{\max}$ obtained in cross-section computation, we check if $W_i/W_{\max}$ is less than or greater then r_2.
- The angle $\cos\ \theta$ is then used to generate an event if and only if $W_i/W_{\max} > r_2$.
- The angle $\cos\ \theta$ and the another angle ϕ are used to define the kinematics of the final state using equations defined later in this section. ϕ is a uniform random number in the range $[0, 2\pi]$ other than r_1 and r_2.

The Kinematics of the event is determined as follows. Since the processes we consider are $2 \to 2$ processes, the absolute three-momentum of a given final state particle can be defined completely in terms of the masses of particles and the center-of-momentum energy of the collision. The generation of events relies on defining the final state kinematics and then randomly generating the angles θ and ϕ as discussed above. The kinematics for the outgoing particles are as follows:

$$|\mathbf{p}| = \sqrt{\left(\frac{E_{\text{cm}}^2 + m_3^2 - m_4^2}{2E_{\text{cm}}}\right)^2 - m_3^2}, \tag{13}$$

here $|\mathbf{p}|$ is the magnitude of three-momentum of the outgoing particle.

The energy is assigned to each particle based on the energy-momentum relation:

$$E = \sqrt{|\mathbf{p}|^2 + m^2}. \tag{14}$$

The three-momentum of one of the particles is defined as $(p_x,\ p_y,\ p_z)$ (using spherical coordinates):

$$p_x = |\mathbf{p}| \sin\theta \cos\phi, \tag{15}$$

$$p_y = |\mathbf{p}| \sin\theta \sin\phi, \tag{16}$$

$$p_z = |\mathbf{p}| \cos\ \theta, \tag{17}$$

and the other particle's four-momentum is defined using the momentum-energy conservation laws.

The decay of a particle, in principle, relies on the decay width of that particle, and this quantity is inversely proportional to the particle's lifetime. Therefore, unless a particle is a "long-lived" particle (long lifetime), the particle will decay before leaving the detector. The decay width of a particle is found using the Feynman rules and is given as [3]:

$$\Gamma = \frac{g^2|\mathbf{p}_x|}{8\pi m_x^2}, \tag{18}$$

where $|\mathbf{p}_x|$ is the absolute three-momentum of the decayed fragment and m_x is the mass of that particle; in general, the mass of the decaying particle has to be greater than the sum of the masses of the decaying fragments.

The decay of a heavy particle produced in the final state is handled as follows. Let us consider that particle A is the heaviest of the three scalars, and its mass is greater than the sum of the masses of particles B and C. Then the decay process $A \rightarrow B\ C$ is allowed by the conservation of energy-momentum. The Feynman diagram of this decay process is given in Fig. 2. The absolute three-momentum of the decay fragments, that is, particles B and C, in the rest frame of the particle A is given by [3,7]:

$$|\mathbf{p}_B| = |\mathbf{p}_C| = \frac{1}{2m_A} \times \sqrt{m_A^4+m_B^4+m_C^4-2m_A^2m_B^2-2m_A^2m_C^2-2m_B^2m_C^2}. \tag{19}$$

The absolute three-momentum for particles B and C is the same owing to the conservation of momentum. Using the energy-momentum relations, the energies of particles B and C are:

$$E_B = \frac{m_A^2 - m_C^2 + m_B^2}{2m_A}, \tag{20}$$

$$E_C = \frac{m_A^2 - m_B^2 + m_C^2}{2m_A}. \tag{21}$$

The above equations indicate the decay kinematics of particles, assuming the particles that is decaying is massive and has a mass more than the sum of the masses of the other two particles. These equations are based on the assumption that we are using the frame of reference in which the particle A is at rest. In principle, the produced particle A is not at rest, hence the above equations need to be modified, 'boosted', using the final state three-momentum of the particle A produced in the main $2 \rightarrow 2$ process. The following matrix equation is used to boost a particle (B or C) with four-momentum $[E, p_x, p_y, p_z]$ [8]:

$$\begin{bmatrix} E' \\ p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} \gamma & \gamma q_x & \gamma q_y & \gamma q_z \\ \gamma q_x & 1+\frac{(\gamma-1)q_x^2}{q^2} & \frac{(\gamma-1)q_x q_y}{q^2} & \frac{(\gamma-1)q_x q_z}{q^2} \\ \gamma q_y & \frac{(\gamma-1)q_y q_x}{q^2} & 1+\frac{(\gamma-1)q_y^2}{q^2} & \frac{(\gamma-1)q_y q_z}{q^2} \\ \gamma q_z & \frac{(\gamma-1)q_z q_x}{q^2} & \frac{(\gamma-1)q_z q_y}{q^2} & 1+\frac{(\gamma-1)q_z^2}{q^2} \end{bmatrix} \times \begin{bmatrix} E \\ p_x \\ p_y \\ p_z \end{bmatrix}, \tag{22}$$

where in the above equation q represents the four-vector momentum of the particle A produced in the $2 \rightarrow 2$ process. The algorithm which we have used here has been inspired by the TGenPhaseSpace class of `CERN ROOT` [8], which is based on the Raubold and Lynch method [9] as well as the `MadSpin` software [10]. Note that we do not apply weights to the decaying particles.

Having generated the events and decayed the final state particles, one would need to detect these particles using detectors as in a real collider experiment. The study of detector effects on the toy ABC model could be used as a way to introduce the so-called fast detector simulation, which is a simulation of detectors based on some parameters. In principle, the detector simulation employed by the `pymcabc` software is two-parameter based, since the detector used by the `pymcabc` software has two components: a tracker-like component for the three-momentum measurement and a calorimeter-like component for the energy measurement. In this paper, we have assumed that the error introduced in measurement by the detectors is Gaussian-like. In reality, the errors in detector measurement arise owing to several reasons such as the misidentification of particles, and the number of particles in a shower for energy measurements, among others. Considering the `pymcabc` software, it allows assigning different sensitivities to the two detector components, but by default, the Gaussian width is set to unity for both components. The detector effects are generated as follows: to begin with, a component of the particle's three-momentum is chosen. The component's value is then chosen as the mean of the Gaussian, and the Gaussian's width is chosen by the user; moreover, an additional factor, that worsens or improves the resolution of the calorimeter over the tracker, is also included and may be defined by the user. The same procedure is also applied for energy measurements, albeit, with different Gaussian widths. By default, both the calorimeter and the tracker are assumed to have the same sensitivity. In the next subsec-

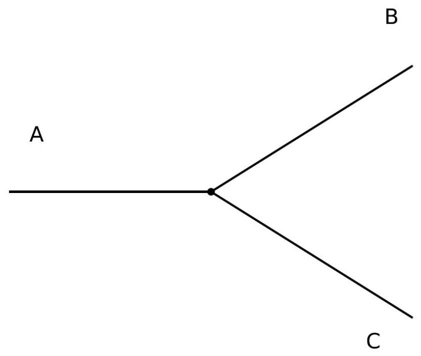


FIGURE 2. Representative Feynman diagram for the Decay process $A \rightarrow B\ C$.

tion, we discuss the steps involved in the installation of the `pymcabc` software.

3.1. Installation and the first run

The `pymcabc` software is available for installation from the Python Package Index (`PyPI`) using the following command in the terminal, assuming that `pip` is available on the operating system:

```
!pip install pymcabc
```

or the following command from the jupyter notebook/session:

To test the installation, the user may open either a jupyter notebook/session or directly use the terminal and try to invoke:

```
import pymcabc
```

If no errors are faced, then the installation has succeeded. We note that `pymcabc` has additional dependencies on `Numpy` [11] to handle arrays, `Uproot` [12] to handle root related scripts within `python`, `Matplotlib` [13] for plots, `feynman`^{iii} for drawing Feynman Diagrams, `Pandas` [14] packages.

A sample code is given here to explain the main commands used in evaluating the total cross-section and generating events for the $A\ B \to C^* \to A\ B$ process:

```
import pymcabc
pymcabc.DefineProcess('A B > A B',
mA=10, mB=1, mC=2, pi=10)
pymcabc.FeynmanDiagram()
pymcabc.CrossSection().calc_xsection()
pymcabc.SaveEvent(10000,boolDecay=True,
boolDetector=True).to_root('name.root')
```

where the expected output includes the total cross-section, set of Feynman Diagrams, and generation of 10k events. Here mA, mB, and mC are the masses of particles A, B, and C, respectively, and `pi` is the magnitude of the initial three-momentum of one of the incident particles. (The `pi` here should not be confused with the $\pi = 3.14$, which is a constant.)

The above script will evaluate the total cross-section of the $A\ B \to C^* \to A\ B$ process. The magnitude of the three-momentum for the initial particle in the collision of particles A and B is set using the parameter `pi`, and, in this case, it is set to 10 GeV. The last statement in the above script generates 10^4 events that have undergone the decay process $A \to B\ C$ as well as toy detector simulation. The name of the file produced at the end is `name.root`.

As far as this software is concerned, these four lines of code are crucial for the total cross-section calculation and event generation for any of the nine possible processes within the ABC universe. We will discuss these steps in detail in this paper. User interaction with other modules is required only when a detailed analysis is sought for, for instance, to access the individual s-channel or t-channel or u-channel contributions, or to change the width of the Gaussian used for toy detector effects. The propagator of the process, in this case, C^*, as well as the decay chain, in this case, $A \to B\ C$ for the final state, are identified by the software internally. We have now checked that the software is installed and in the next subsection we go through the benchmarks of the software. To begin with, we discuss the computation of the differential cross-section with the `pymcabc` software and compare the output with the results from the ABC theory to see if the matrix elements used in the package are implemented correctly.

3.2. Calculation of the Differential Cross-Section for the A A → C* →B B process

We now calculate the differential cross-section for a process and see if the matrix elements computed using the `pymcabc` software are consistent with the theory. To calculate the differential cross-section, we will first need to calculate the matrix element for the given process using the software and ensure that it agrees with the theory.

Here, this test is carried out by evaluating the differential cross-section for the tu-channel using the `pymcabc` software and comparing the results with Griffiths [3]. We consider the $A\ A \to C^* \to B\ B$ process that occurs via the t- and u- channels. The theoretical differential cross-section for this process for the case when $m_A = m_B$ and $m_C = 0$ is given by:

$$\frac{d\sigma}{d\Omega} = \frac{g^2}{2 \times 256\pi^2 |\mathbf{p}|^4 E_{cm}^2 \sin^4\theta}, \tag{23}$$

where $|\mathbf{p}|$ is the magnitude of the initial particle's three-momentum. The following model parameters are chosen for this example: $mA = 10$ GeV, mB $= 10$ GeV, mC $= 0$ GeV, and the magnitude of the initial three-momentum is $|\mathbf{p}| = 10$ GeV. Inserting these values in the equation, we obtain:

$$\frac{d\sigma}{d\Omega} = \frac{9.89 \times 10^{-11}}{\sin^4\theta}. \tag{24}$$

To compute the differential cross-section for a given theta, we use the following code:

```
import pymcabc
import numpy as np
theta = n # define theta in the range [0, 6.28],
#avoiding 0, 3.14, and 6.28, can also use numpy to define theta over a range
# define the physics process
```

```
pymcabc.DefineProcess('A A > B B',mA=10,mB=10,mC=0,pi=10)
#evaluate differential cross-section at a given theta
pymcabc.CrossSection().dsigma_tu(math.cos(theta)
#we fit using the theory prediction,
# Theory prediction function evaluated at a given theta
#parameter is defined as given in the text
def theory_pred(theta, par):
    return _par*1./(np.sin(theta))**4
## and plotting scripts follows this
```

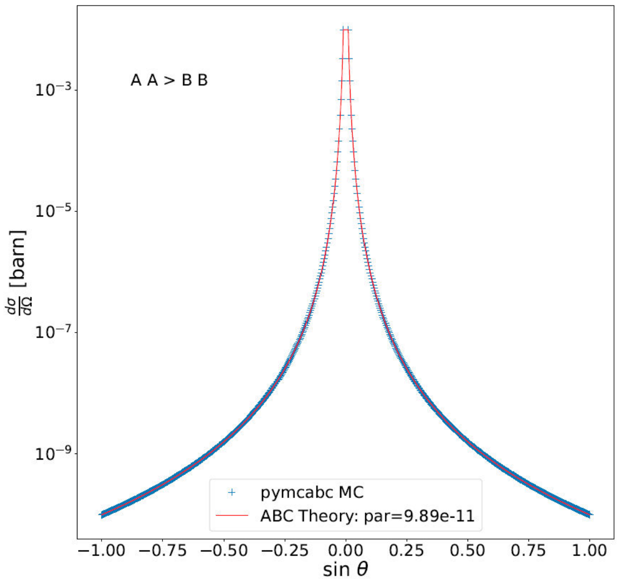


FIGURE 3. The differential cross-section vs. $\sin\theta$ as computed by the pymcabc and compared with the theoretical prediction of the ABC model as calculated from the theory equation for the $A\ A \to C^* \to B\ B$ process.

We have used Numpy package [11] to provide input in terms of arrays. The plotting scripts make use of Matplotlib package [13].

The theoretical prediction given in Eq. (24) is compared with the pymcabc software prediction for different angles. In Fig. 3, we see that the software's calculation agrees with the theory prediction. We have shown this agreement for a region of $\sin\theta$ away from zero since the differential cross-section tends to infinity when $\sin\theta$ is close to zero. We now compare the total cross-section obtained using the pymcabc software with the MadGraph5_aMC@NLO.

3.3. Comparison of the total cross-section results obtained using the pymcabc software with MadGraph5_aMC@NLO

We have checked the consistency of the pymcabc software with the prediction from theory for the differential cross-section. In this section, we test the prediction of the pymcabc software with MadGraph5_aMC@NLO, which is a Monte Carlo Event Generator used in the High Energy Physics (HEP) to corroborate data as well as in other HEP phenomenological studies. MadGraph5_aMC@NLO is a fully automated HEP library that can simulate Standard Model as well as Beyond the Standard Model processes. A physics model can be added to MadGraph5_aMC@NLO by first producing a Universal Feynman Rules Output (UFO) [15] with either FeynRules [16] or LanHEP [17]. The UFO model files of the ABC model, which is implemented in LanHEP, can be downloaded from this link [18]. In this section, we evaluate the total cross-section for all the nine physics processes in the ABC model using both the pymcabc software as well as the MadGraph5_aMC@NLO. For computation, we have chosen the following four "benchmark" sets of parameters as given in Table II along with a brief motivation on using these.

We compare the pymcabc software's final results with the MadGraph5_aMC@NLO results. Note that the MadGraph5_aMC@NLO software takes as input the beam energies for beam 1 and beam 2 while in pymcabc we consider as input the initial three-momentum of one of the initial particles. In order to extract the actual beam energies, we use the ECM() function of pymcabc software, which gives as output a tuple of (beam1energy, beam2energy, TotalCMEnergy). Moreover, for using the MadGraph5_aMC@NLO comparison, we have set the width of the propagator using Eq. (18). The results of this exercise are shown in Fig. 4. The results indicate that the predictions of the pymcabc package are consistent with that of the advanced tools used by particle

TABLE I. Benchmark parameters used for comparison of the pymcabc with MadGraph5_aMC@NLO.

Label	mA	mB	mC	p_i	Motivation
Scenario 1	2	3	5	10	Decay width of particles is zero (no massive particle)
Scenario 2	2	3	5	5	Same as above with different initial three-momentum
Scenario 3	9	3	4	10	Massive particle
Scenario 4	3	6	8	15	Large masses of all particles, high three-momentum

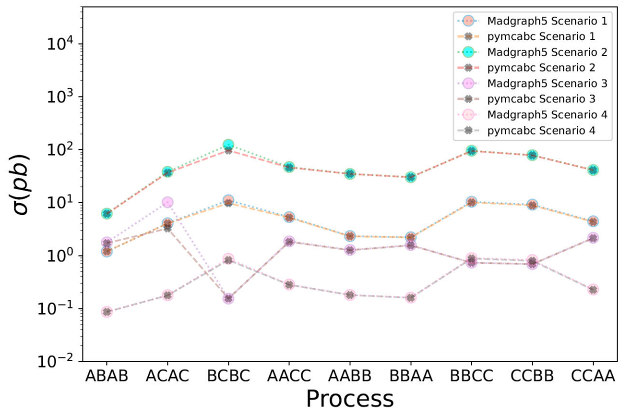


FIGURE 4. Comparison of results of `pymcabc` software with the Madgraph result where Madgraph is run using a UFO model developed here.

physicists in active LHC and other collider studies. There are known divergences, especially when the mass of the propagator is zero. In the last section, we discussed this case and compared the differential cross-section with theory to show that the `pymcabc` calculation is consistent with theory in a given range of angles that avoids $\sin\theta = 0$.

The `pymcabc` software can also be used in the following way to obtain the total cross-section and the integration 'error':

```
import pymcabc
x = pymcabc.DefineProcess('process',mA=10,
mB=1,mC=2,pi=10)
energy_beam1, energy_beam2, cms_energy =
x.ECM()
xsec, error = pymcabc.CrossSection().
calc_xsection()
```

where `'process'` refers to `A A > B B` or `B C > B C`, etc. as listed in Table I. The error arises as one uses the Monte Carlo integration technique.

The error due to Monte Carlo integration can be controlled by varying the number of integration steps/points used. In principle, the error is inversely proportional to the square of a number of points used for integration as shown in Eq. (12). One can change the number of integration points in the following way:

```
"""
Code to evaluate s-,t- channel total cross-section
with different numbers of phase space points
"""
import pymcabc
pymcabc.DefineProcess('A B > A B',mA=10,mB=1,mC=2,pi=10)
sigma_x, error = pymcabc.CrossSection().calc_xsection(N = Nintegrate)
```

where `Nintegrate` refers to the number of points to be used for the Monte Carlo integration. Here the default value for any given calculation is set to 10000 integration points. The same procedure is also used to evaluate the total cross-section for the tu-channel process.

The Figs. 5a) and 5b) indicates that the error in the total cross-section divided by the total cross-section decreases with the increase in the number of Monte Carlo integration points ($\sigma_{\text{error}} \propto 1/N^2$). Although in an ideal situation having more Monte Carlo integration points is beneficial as it tends to reduce the fluctuations, nevertheless, the time taken by the `pymcabc` for

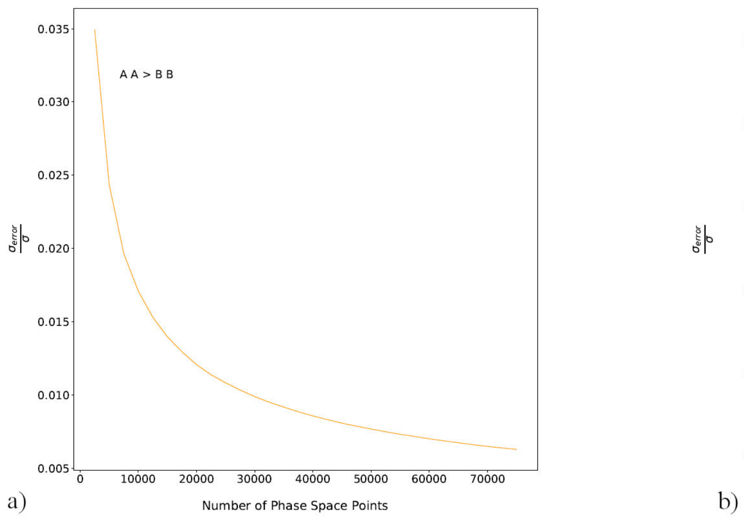
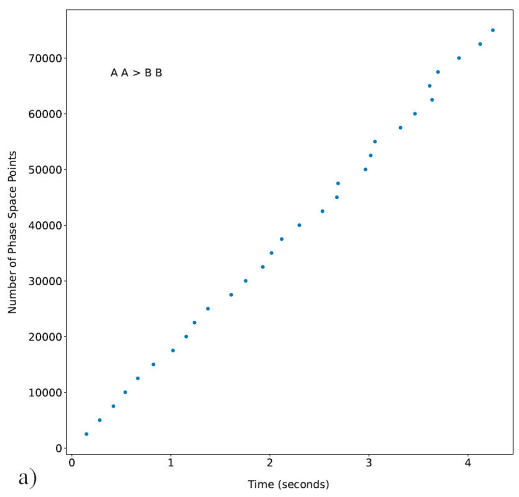


FIGURE 5. The ratio of error in the total cross-section to the total cross-section is evaluated by the `pymcabc` software using different numbers of phase space points that are considered for Monte Carlo integration for a) $A\ A \to C^* \to B\ B$ and b) $A\ B \to C^* \to A\ B$.



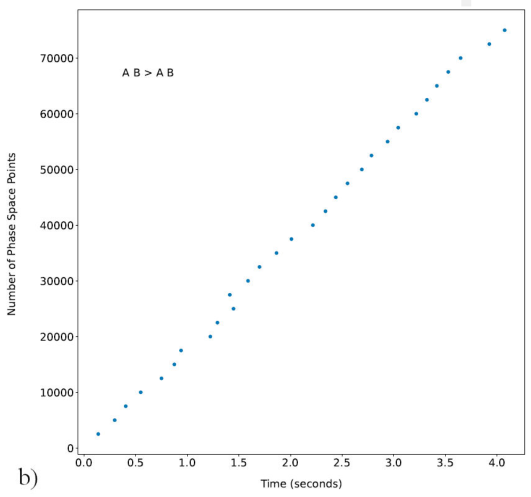


FIGURE 6. The number of space points versus the time taken by the `pymcabc` software in the evaluation of the total cross-section for a) $A\ A \rightarrow C^* \rightarrow B\ B$ and b) $A\ B \rightarrow C^* \rightarrow A\ B$.

calculation increases linearly with the number of Monte Carlo integrations as shown in Figs. 6a), and 6b). Hence, the number of points used in default calculations is set to 10^4, which can be modified by the user using the `N` parameter in the function `CrossSection().calc_xsection` as explained in the `Python` program above. Here, the time taken in the evaluation is calculated using the `python` module `time`.

3.4. Total cross-section's dependence on the Center-of-Mass Energy

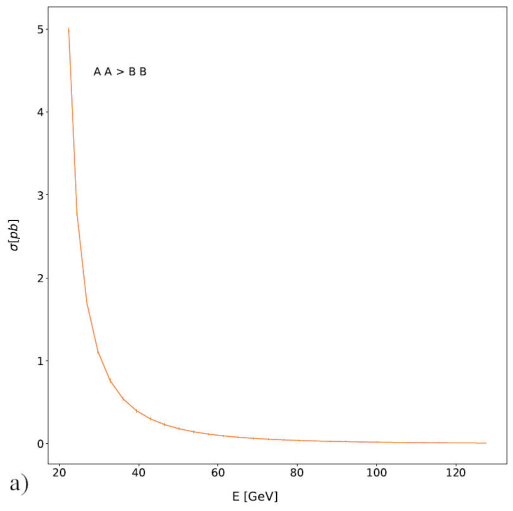
The total cross-section depends on the center-of-mass energy, $\sigma \propto 1/E_{\text{cm}}^2$, as can be seen from Eqs. (2), (5). We now put to test this theoretical prediction with the `pymcabc` software. We see that the calculation for the ABC model indeed shows this dependence as seen from Fig. 7. This computation is performed by varying the initial three-momentum (`pi`) parameter of `DefineProcess`, which, in turn, varies the center-of-mass energy of the collision since,

$$E_{cm} = \sqrt{\mathbf{p}_i^2 + m_1^2} + \sqrt{\mathbf{p}_i^2 + m_2^2}, \tag{25}$$

where 1, and 2 indicate the two initial particles in the system.

The code to perform this computation is:

```
import pymcabc
pi_value = #value of initial three-momentum to be used for computation
x = pymcabc.DefineProcess('process',mA=10,mB=1,mC=2,pi=pi_value)
energy_beam1, energy_beam2, cms_energy = x.ECM() # beam energies
#'process' needs to be updated to "A B > A B" or "A A > B B"
sigma_x, error = pymcabc.CrossSection().calc_xsection()
```



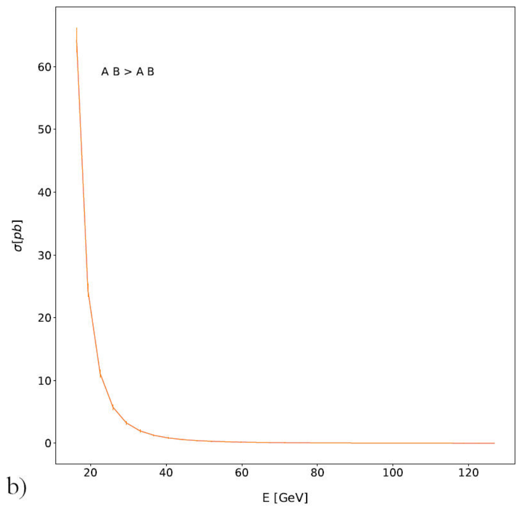


FIGURE 7. The total cross-section for a) $A\ A \rightarrow C^* \rightarrow B\ B$ and b) $A\ B \rightarrow C^* \rightarrow A\ B$ as a function of energy.

3.5. Event generation

When two initial particles scatter they lead to a final state consisting of two or more particles. A physics event is then defined as this interaction of particles. A single collision at real colliders may lead to the production of many particles, which makes the study of such an event very complex; moreover, there are many collisions per second and this aggravates the detection problem. This is among the reasons why event generators play an important role in particle physics. Computationally, the generation of physics events involves finding the final state kinematics of particles provided we are aware of the initial state. The equations involved in final state kinematics were given in Eqs. (13)-(17). In this section, we provide some guidance on how to use the `pymcabc` software to generate events for the st-channel process by providing the program used to generate the events and the options available in generation. The steps involved in generating an st-channel type process are:

```
import pymcabc
pymcabc.DefineProcess('A B > A B',mA=10,mB=1,mC=2,pi=10)
pymcabc.CrossSection().calc_xsection()
pymcabc.SaveEvent(10000,boolDecay=True,boolDetector=True).to_root("ab_ab_decay_detect.root")
```

Here, the produced particle undergoes decay if the variable/flag `boolDecay` is set to `True` and undergoes detector effects if the variable/flag `boolDetector` is set `True`. The options `boolDecay` and `boolDetector` can either be True or False, and thus we have four combinations or ways to produce events (True, True), (True, False), (False, True), (False, False). These statements produce two files in root format: the first file contains truth information, which stores the actual events without decay and detector effects, and this file carries `truth_` as the prefix to the filename; the second file contains the events after the decay of the heavy particle produced in scattering and after the detector effects are applied on the particles. The name of the `ROOT` Tree where events are stored is called `events`. The steps involved in generating a tu-channel type process are the same, except that the name of the process is changed. The basics of the event analysis are provided in the next subsection.

3.6. Basic event analysis using events generated with the `pymcabc`

In this section, we provide some illustrations of how the events generated with the `pymcabc` software may be analyzed. Event analysis refers to analyzing the events generated in the previous subsection and it involves, among other things, producing the plots (histograms) of the distribution of observables such as kinematic variables. The `pymcabc` software comes with an inbuilt script to perform basic event analysis such as drawing the distributions of four-momenta of final state particles. The script to perform this analysis is:

```
import pymcabc
# Initial Setup (same as in previous examples)
pymcabc.DefineProcess("A B > A B", mA=4, mB=10, mC=1, pi=30)
pymcabc.CrossSection().calc_xsection()
pymcabc.SaveEvent(100, boolDecay=True, boolDetector=True).to_root(
    "test_eventGen_detector_decay_st.root")
# This line leads to the automatic analysis of events
#four momenta of final state particles are plotted
pymcabc.PlotData.file("test_eventGen_detector_decay_st.root")
```

This script produces the four-momentum distribution, the user may be interested, for instance, to see the distribution of transverse momentum, which is defined as $p_T = \sqrt{p_x^2 + p_y^2}$, or, some other variable constructed using the events. To handle that, the user might either use `Uproot` [12]^{iv} or use `CERN ROOT`^{v} or the `Python` extension of `ROOT`, which is `PyROOT`^{vi}. For now, we illustrate with a simple example that can be used to open a tree, read events, and plot a given variable with `Uproot` and `PyROOT`.

```
import ROOT
import array
# open the file called aa_bb_decay_detect.root
file = ROOT.TFile.Open("aa_bb_decay_detect.root")
tree = file.Get("events") # tree name is "events"
hist_b1_energy = ROOT.TH1D("b1_energy","b1_energy",30,0,20)
for entry in range(0,tree.GetEntriesFast()): # loop over events
    tree.GetEntry(entry) #retrieve the entry
    # read the energy of final state B's energy
    B1_energy = getattr(tree, "B_1_E")
    B2_energy = getattr(tree, "B_2_E")
    hist_b1_energy.Fill(B1_energy) # fill histogram
c1 = ROOT.TCanvas() # make canvas
hist_b1_energy.Draw() # draw histogram
c1.SaveAs("name.pdf") # save histogram
```

To print the names of all the branches in a given `TTree`, one can do as follows:

```
for the branch in the tree.GetListOfBranches():
    print(branch.GetName())
```

Another alternative procedure to analyze events is to use the `Uproot` software. A simple program to read a tree using 'Uproot' is given here:

```
import Uproot
import matplotlib.pyplot as plt

file = Uproot.open("aa_bb_decay_detect.root") # open file
tree = file["events"] # define tree
branches = tree.arrays() # retrieve name of branches
print(branches) # print a list of branch
plt.hist(branches["B_1_E"],bins=40, color=None) #plot histogram
plt.savefig("name.png") # saving figure
```

One more possibility is to analyze events using the `pandas` module [14]. The user may convert the `.root` file obtained after simulation into a comma separated file (`csv`) using the following `pymcabc` inbuilt feature:

```
import pymcabc
pymcabc.convert_tocsv(inputname="name.root",outputname="name.csv")
```

We may note that the analysis of "events" presented in this paper is handled with `PyROOT`[19], which is the python extension of `CERN ROOT` software [8]. Most plots were produced using either the `Python` module `Matplotlib` [13] or `PyROOT`.

4. Physics cases

We have discussed the generation of events within the ABC model as well as the analysis of events using various tools. In real experiments, the generation of events undergoes many stages such as the generation of the 'hard' process, decays of heavy resonances, parton shower, and fast- or full- detector simulation. Then, at the end, physics analysis follows. In the `pymcabc` software, we have automated some of the steps and attempted to produce the final events as they would be in the large experiments. To illustrate the use of the generated events, we present a few phenomenological cases. In this section, we study a few physics cases such as the basic event analysis, analyzing the lineshape of a massive propagator, and the recoil mass reconstruction technique. This section is not meant to teach physics analysis but to showcase the ways in which the generated events can be used and the reader is encouraged to re-produce these studies.

4.1. Event analysis

4.1.1. *Event analysis of a tu-type process: $A\ A \rightarrow C^* \rightarrow B\ B$*

Section 3.5 discussed about the commands used to generate events with the `pymcabc` software and Sec. 4.1 presented a brief introduction to event analysis with `PyROOT`, and `Uproot`. The analysis of the generated events is crucial to understanding the phenomenology of the ABC model. For the study presented in this section, we have considered the following parameters: $mA = 10$ GeV, $mB = 1$ GeV, $mC = 2$ GeV and $pi = 10$ GeV. The output of event generation gives the four-momentum of the final state particles B (two B) as given in Fig. 8. To distinguish the particle B's, we have assigned the label `_1_` and `_2_` with the variables in the `ROOT TTree`. Since the final state ($B\ B$) does not consist of the heaviest particle, that is, particle (A), the decay is forbidden. For each variable presented in Fig. 8, the detector effects are shown. It may be noted that in a real experiment, particle B will not be differentiated in the way in which the ABC software allows, which is primarily for understanding purposes.

In addition to the particle's four-momentum, we have also shown the common variables such as the transverse momentum and pseudorapidity distribution, where pseudorapidity of a particle is defined as the rapidity in the limit of mass tending to zero:

$$y = \frac{1}{2}\log\left(\frac{E+p_z}{E-p_z}\right), \tag{26}$$

and $\eta = y_{\lim(m\rightarrow 0)}$. The reader is encouraged to prepare plots for more such variables as used at the Hadron and Lepton colliders.

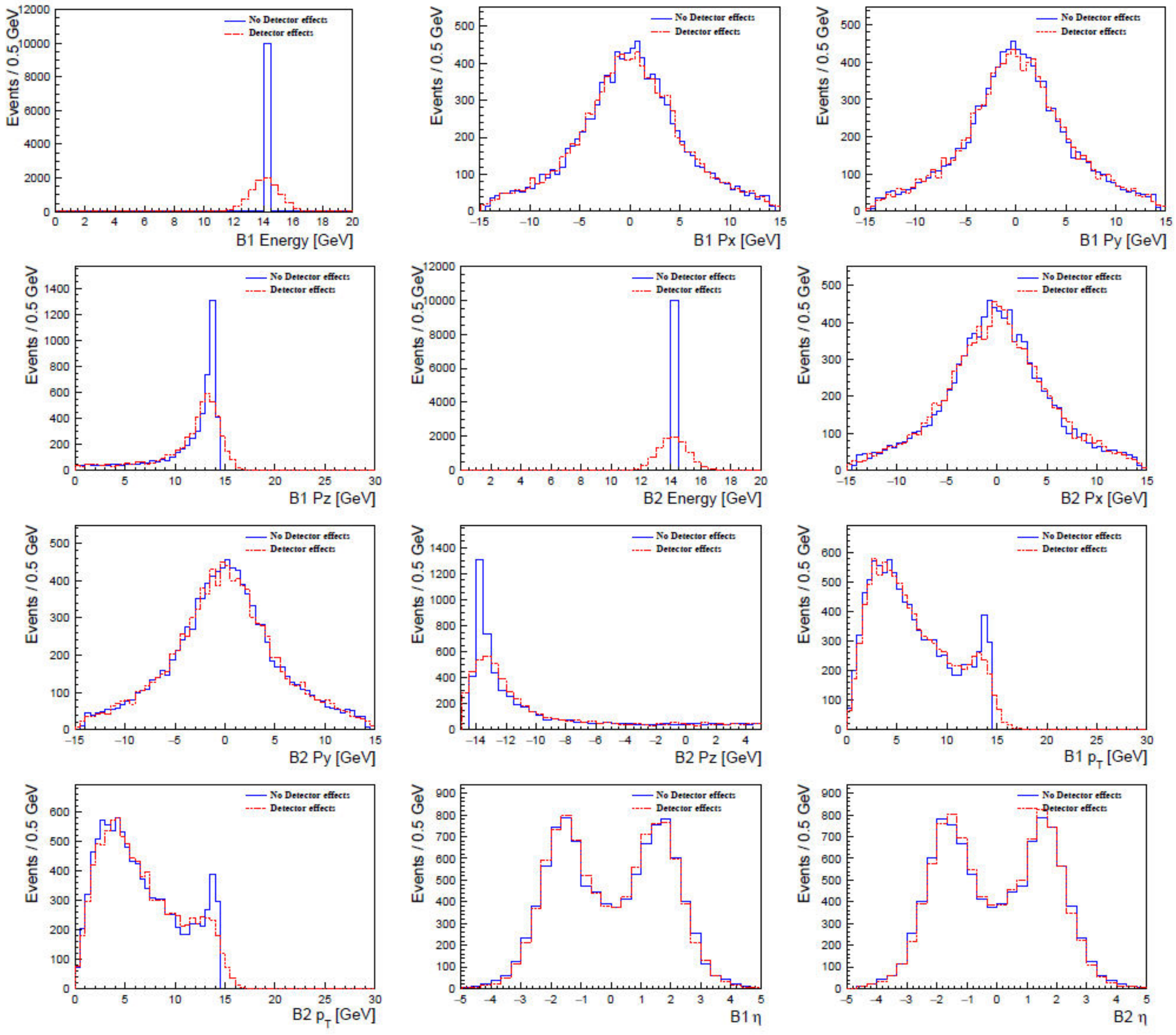


FIGURE 8. The energy, three-momentum p_x, p_y, p_z distributions for particles B1 and B2 as obtained from the `pymcabc` software with and without detector effects in the $A\ A \rightarrow C^* \rightarrow B\ B$ process. The parameters used for these distributions are: $mA = 10$ GeV, $mB = 1$ GeV, $mC = 2$ GeV and $pi = 10$ GeV.

The distributions of the variables, px and py, are Gaussian, as random numbers were used in their generation. The pz of variables for B1 and B2 have a mirror-like symmetry. Since the particles are the same in the final state, and the absolute three-momentum for each is the same owing to the conservation of momentum, the energy of the particles is equal. Lastly, the plots of p_T and η are shown, which have characteristics (shapes) similar to the ones obtained in a HEP analysis of a Standard Model or Beyond the Standard Model scenario.

4.1.2. *Event analysis: Combinatorics background in an st-type process: $A\ B \rightarrow C^* \rightarrow A\ (\rightarrow B\ C)\ B$*

We have shown in Sec. 3.5 the steps to generate the events with the decay process and detector effects for the process $A\ B \rightarrow C^* \rightarrow A\ (\rightarrow B_2\ C)B_1$ that takes place in the st-channel. The analysis of the kinematics of the final state particle is shown in Fig. 9. The interpretations remain the same as in the previous sections. However, there are some changes in the p_T and η distributions owing to the presence of three particles in the final state.

As a physics case study, we consider the combinatorics background, that is, the background that is a result of wrong combinations, which might result when one reconstructs the particle A using the decay products, B and C. In this case, there are two choices: use particles B_1 and C or particles B_2 and C to reconstruct A; the right choice is to use particles B_2 and C. However, a priori, it may not be known which is B_1 and B_2. The events resulting owing to the first choice (B_1 and C) are referred to as the 'background', whereas the second choice (B_2 and C) is termed as a 'signal'.

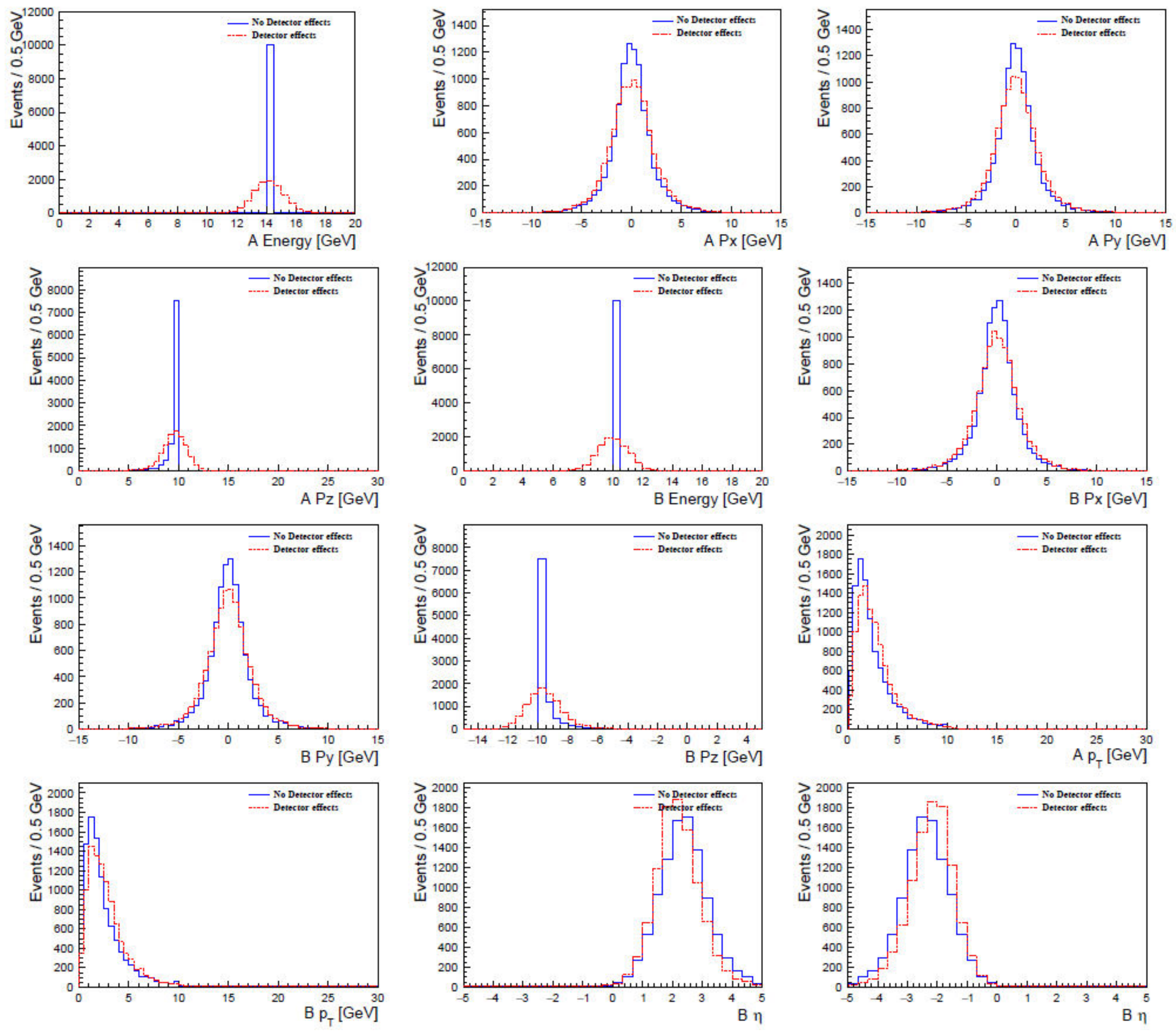


FIGURE 9. The energy and three-momentum, p_x, p_y, p_z distributions for particles A and B_1 as obtained from the pymcabc software with and without detector effects for the $A\,B \to C^* \to A\,B_1$ process. The parameters used for these distributions are: $mA = 10$ GeV , $mB = 1$ GeV, $mC = 2$ GeV and $pi = 10$ GeV.

The distributions of the combined four-momentum of the two possible combinations of B and C are given in Fig. 10. Figure 11 shows the invariant mass of the two combinations, namely, B_2 and C as signal and B_2 and C as background, along with the truth distribution of particle A's mass after applying detector effects.

At the LHC experiments combinatorial background often arises, for instance, when analyzing the decays of B meson ($B_s^0 \to \mu^+\mu^-$) [20] among others. Therefore, this exercise is useful to introduce the concept of combinatorial background.

4.2. Lineshape of a heavy mass intermediate state in an s-channel process

For a process that takes place in the s-channel, one can obtain a lineshape distribution for the propagator. The lineshape distribution of a particle refers to the plot with the y-axis containing the s-channel process's production total cross-section versus the energy on the x-axis, and making sure to cover the energies close to the mass of the propagator. Such studies for determining the lineshapes for the Z boson were conducted at the LEP1 and LEP2. Moreover, future lepton colliders also aim to produce the lineshapes for the Z boson and Higgs boson [21].

One can use the pymcabc software to produce lineshape plots for the heavy propagator in the ABC model by using the st-channel process and discarding the diagrams related to the t-channel and the interference of st-channel using the scheme discussed earlier in the paper. As an example, for the ABC model, we consider the following st-channel process, $AC \to AC$, and consider the contribution of the s-channel

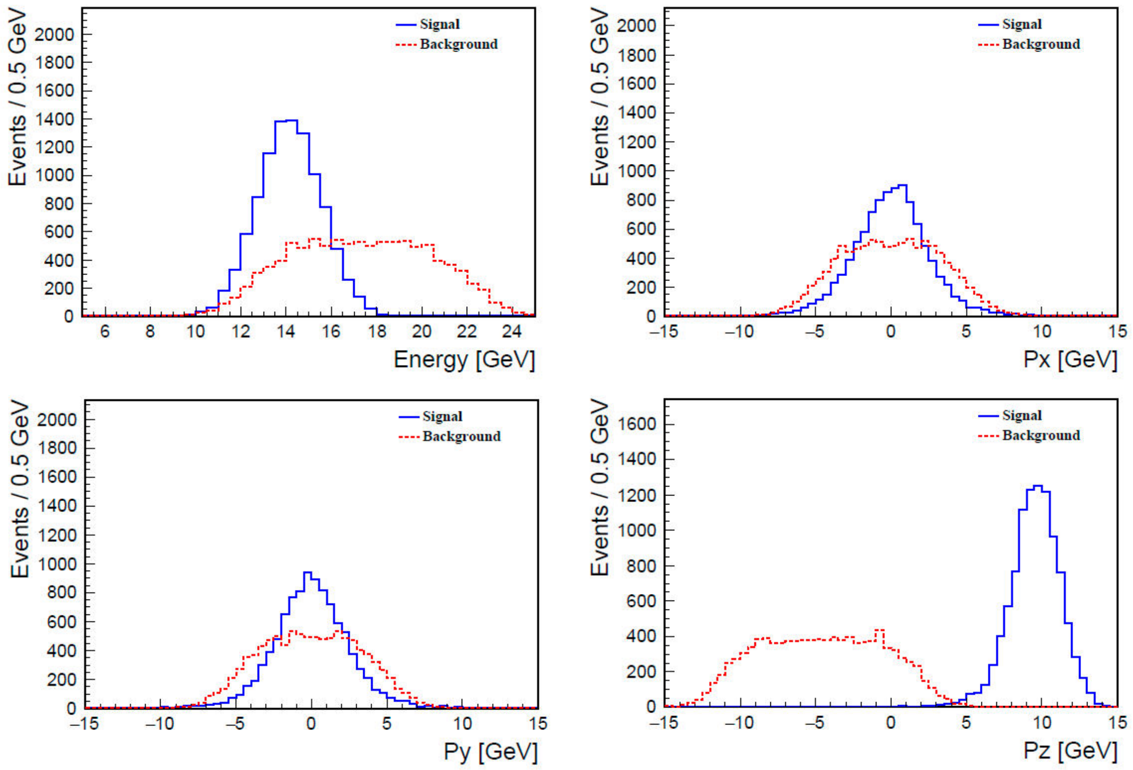


FIGURE 10. For the process A B → A (→ B2 C) B1, the comparison of the combined four-momenta of B2 and C (signal) with B1 and C to illustrate the combinatorics background.

only. Let the masses of particles $mA = 1$ GeV, $mB = X$ GeV, $mC = 1$ GeV, where X is defined as 10 GeV (blue), 20 GeV (orange), 30 GeV (green), 40 GeV (red), and 80 GeV (purple) in Fig. 12. The following code can be used to evaluate a specific channel contribution:

```
import python
get_energy = pymcabc.DefineProcess('A C > A C',mA=1,mB=10,mC=1,pi=p[i],channel='s')
energy_cm = get_energy.ECM()[2]
pymcabc.CrossSection().calc_xsection()
```

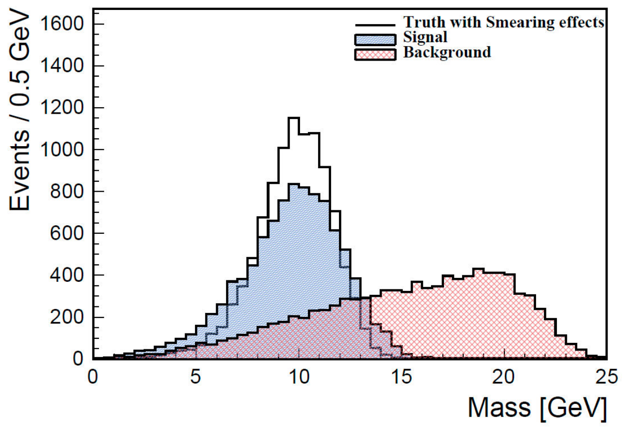


FIGURE 11. For the process A B → A (→ B2 C) B1 comparison of the invariant mass distribution of the signal combination B2 and C with background combination B1 and C. The truth distribution of the expected invariant mass distribution after applying detector effects is also shown.

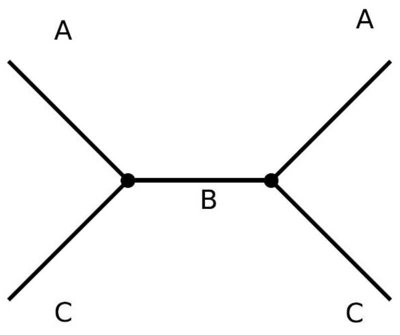


FIGURE 12. Representative Feynman diagram for the $A\,C \rightarrow A\,C$ process in the s-channel.

Here, the energy scan is performed around the mass of particle $B = 10$ GeV. The result of this energy scan is given in Fig. 13. The reason for the high peak in the total cross-section near the mass of the resonance particle B is because in the s-channel, the matrix element tends to infinity as the center-of-mass energy approaches the mass of the propaga-

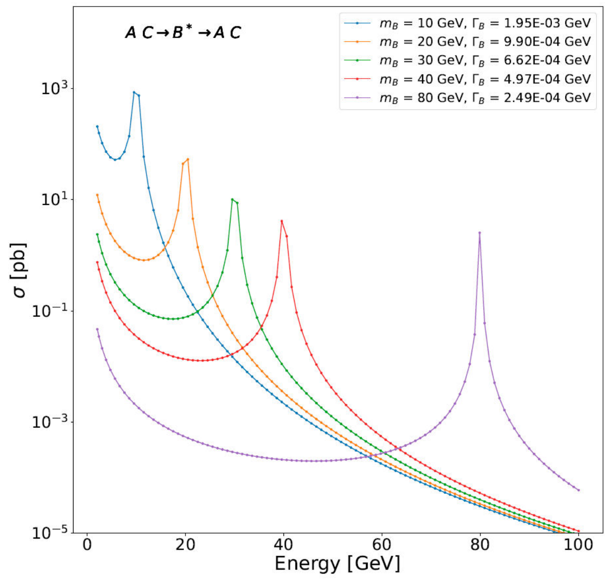


FIGURE 13. Lineshape for particle B with mass 10 GeV. The energy scans were around the center of- mass energy ‘close’ to the mass of particle B.

tor. Within the software, owing to the modified matrix element, the matrix element or the total cross-section does not tend to infinity but to $\propto (1/m^2\Gamma^2)$. The width of the resonance in the plot is owing to the factor of Γ (particle decay width). This plot can then be used to extract the mass of the propagator as well as measure the decay width of the propagator, thereby providing the two most fundamental parameters relevant for a particle (other than its intrinsic spin, color, etc., which are not relevant for the ABC model).

We note from the plots, that the total cross-section reduces with an increase in the mass of the mediator as $\sigma \propto 1/m_B^2$. Moreover, we also notice that the decay width (and therefore the width of the resonance structure on the plot) reduces with an increase in the mass of the mediator; for instance, for $m_B = 10$ GeV, the width of the curve is wider as compared to when $m_B = 80$ GeV. This is due to the decay width being inversely proportional to the mass of the particle B, in this case.

4.3. Recoil mass reconstruction technique

Reconstruction of particles in the final state is crucial for the collider experiment and to achieve this one needs accurate information about the initial state. The Hadron colliders lack the information on initial energies at which the quark or gluon within the proton interacts. That is, although the protons collide at 13.6 TeV at the LHC, the actual energies with which the gluons/quarks interact are different, and are less than 13.6 TeV. This is because the distribution of quark and gluon’s three-momentum is random and is based on the Parton Distribution Function. On the other hand, lepton collider experiments have access to the initial energies and in their cases, this information can be used to determine the properties of one of the final states provided that the other particle can be reconstructed relying on its decay properties. In summary, at lepton colliders, the information on initial energy can be used as a constraint to extract the properties of a particle, for example, to reconstruct the Higgs recoil mass using this technique [21].

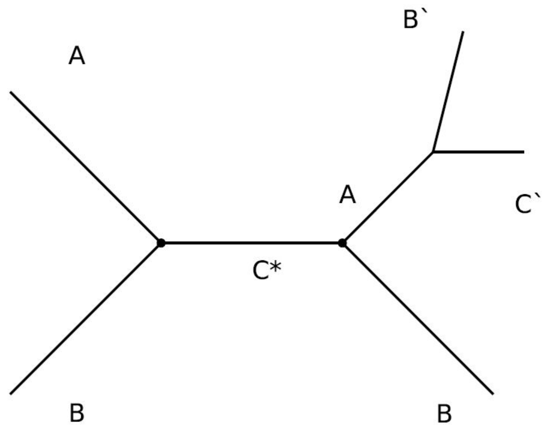


FIGURE 14. Representative Feynman diagram for the $AB \to AB$ process with $A \to B'C'$.

Consider the following process: $A\ B \ \to \ A\ B$, which occurs via the st-channel with the mass of particles $mA = 120$ GeV, $mB = 20$ GeV, and $mC = 40$ GeV. The initial three-momentum is chosen as 180 GeV (high enough to allow the process and give some additional boost to the decay fragments).

The `detector_sigma` parameter of the detector is set to 0.5. Let us consider that only the s-channel contribution remains valid for this production. Owing to its large mass, particle A undergoes the decay: $A \ \to \ B'\ C'$ (Feynman diagram of the full event is in Fig. 14). Our aim in this section is to illustrate the recoil mass reconstruction technique that can help in the reconstruction of the mass of particle B, which was produced along with particle A. The program for generating such events is as follows:

```
import pymcabc
a=pymcabc.DefineProcess('A B > A B',mA=120,mB=20,mC=40,pi=180,channel='s')
Ecm =a.ECM()[2] # extract beam energy for later use in analysis
pymcabc.CrossSection().calc_xsection()
pymcabc.SaveEvent(100000,boolDecay=True,boolDetector=True,detector_sigma=.5)
.to_root("ab_ab_all_strah.root")
```

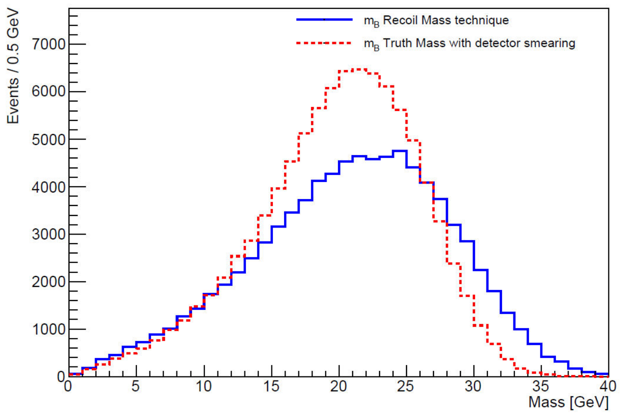


FIGURE 15. Recoil mass reconstruction technique is illustrated for the process $AB \rightarrow AB$ where the recoil mass of particle B is reconstructed using the initial energies and information of the decay products of particle A that undergoes the decay $A \rightarrow B'C'$. Truth measurement also includes detector effects.

The method of recoil mass reconstruction is as follows:

Let $p_1 = (E_{iA}, 0, 0, p_{iAz})$, $p_2 = (E_{iB}, 0, 0, -p_{iAz})$ be the initial four-momenta of initial state particles A, B and let p_3, p_4 be the final four-momenta of the final state particles A, B, respectively. Then, according to the energy-momentum conservation:

$$p_1 + p_2 = p_3 + p_4. \tag{27}$$

The Left-Hand-Side of the above equation leads to the total energy $p_1 + p_2 = (E_{iA} + E_{iB}, 0, 0, 0) = (E_{\text{cm}}, 0, 0, 0)$. Let us further assume that the particle detector can measure the decay information of particle A in the final state. Therefore, p_3 is completely known.

Then

$$p_4 = E_{\text{cm}} - p_3, \tag{28}$$

which upon squaring and using $p_4^\mu p_{4\mu} = m_4^2$ leads to:

$$m_4^2 = E_{\text{cm}}^2 - 2E_{\text{cm}}E_3 + m_3^2. \tag{29}$$

One can therefore determine the mass of the final state particle 4, B, completely using the knowledge of the initial state and the knowledge of the state of particle 3. The recoil mass reconstructed using this technique is compared against the true mass of particle B (after detector response) and is given in Fig. 15. We see from the figure that the reconstructed mass of particle B (blue) shown by the Blue line overlaps the distribution representing the true mass of particle B (red) after detector smearing.

5. Exercises and further project ideas

In this section, we outline some exercises that may be carried out using the `pymcabc` software. Some of the exercises range from being easy to hard and cover many areas within particle physics such as physics analysis, particle physics theory, experiment, and computation speed. These exercises may help the novice to broaden their knowledge of the phenomenology analysis that is undertaken at the collider experiments as well as allow them to go through the complexity that is involved in particle physics theory and experiments. Some of the exercises require modifying the source code, which can be obtained from:

```
wget https://github.com/amanmdesai/
pymcabc/archive/refs/heads/master.zip
```

and then unpack the zip folder.

Another option is to use `Git` as follows:

```
git clone https://github.com/
amanmdesai/pymcabc.git
```

5.1. Physics - analysis idea

- Verify the conservation of energy and momentum using the final state energies and momenta of particles. (Note that, in this case, the detector mode must be turned off for accurate results.)

- Analyze the kinematics of a four-particle final state. The following steps are required: identify the process and parameters that lead to a final state with four particles and, then develop analysis scripts to analyze the final state.

5.2. Physics - theory related ideas

- In the paper, while evaluating the total cross-section for an s-channel process, we removed diagrams related to t-channel as well as interference between the s- and t- channels. There are other schemes that can be used for diagram removal. As an instance, consider the matrix element $\sigma \propto |\mathcal{M}_s + \mathcal{M}_t|^2$, expanding this we found $\sigma \propto |\mathcal{M}_s|^2 + |\mathcal{M}_t|^2 + 2|\mathcal{M}_s\mathcal{M}_t|$. We then remove the contribution of $|\mathcal{M}_t|^2$ and take the sum of $|\mathcal{M}_s|^2 + 2|\mathcal{M}_s\mathcal{M}_t|$ as the total contribution from s-channel.

- Next-to-leading Order (NLO) Processes: An interesting extension of this software is to allow NLO computation within ABC. This requires a major change in the software. However, this will help to learn the difference between NLO and LO contributions. In particular, the student may be introduced to the so-called K_{factors} often mentioned in the HEP literature.

- ABCD model: Extend the ABC model to include one more scalar particle, say, for example, D, with the same characteristics. This will lead to more types of interactions and vertices and, thus, the phenomenology of the ABCD model . The Lagrangian for this model can be written as follows:

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}\partial_\mu\phi_A\partial^\mu\phi_A + \frac{1}{2}\partial_\mu\phi_B\partial^\mu\phi_B + \frac{1}{2}\partial_\mu\phi_C\partial^\mu\phi_C \\ &- \frac{1}{2}\partial_\mu\phi_D\partial^\mu\phi_D - \frac{1}{2}m_A^2\phi_A^2 - \frac{1}{2}m_B^2\phi_B^2 - \frac{1}{2}m_C^2\phi_C^2 \\ &- \frac{1}{2}m_D^2\phi_D^2 - ig_{ABC}\phi_A\phi_B\phi_C - ig_{ABD}\phi_A\phi_B\phi_D \\ &- ig_{ACD}\phi_A\phi_C\phi_D - ig_{BCD}\phi_B\phi_C\phi_D \\ &- ig_{ABCD}\phi_A\phi_B\phi_C\phi_D\end{aligned}$$

where the g_{ABC} and others represent the coupling constants associated with particles as labeled. The last term $ig_{ABCD}\phi_A\phi_B\phi_C\phi_D$ can be mimicked as a contact interaction at a point. The corresponding UFO model for use with `MadGraph5_aMC@NLO` may be obtained from this link [22].

5.3. Detector related ideas

- In the paper, we showed that the keywords `boolDecay, boolDetector` may be used in four possible combinations: (True, True), (False, True), (True, False), and (False, False). The last combination is often referred to in particle physics as the truth sample. The task for the student would be to compare the distribution for an st-type process and a tu-type process.

- Exploring Detector Configuration: The student can also modify the detector configuration. For instance, an easy exercise would be to modify the Gaussian width. A more difficult exercise would be to use a statistical distribution different from the normal distribution for energy measurements and assume a normal distribution for three-momentum measurements. Moreover, one could also re-implement the detector sensitivities to vary with different η and p_T of particles.

- Invisible Particles: In real collider experiments, certain particles such as neutrinos in the Standard Model and Dark Matter particles within the Beyond the Standard Model cannot be directly detected by the detector. These particles lead to the so-called missing p_T in the detector. The idea is that the initial p_T is zero and, therefore, the final p_T should also be zero due to the conservation of energy-momentum. To obtain invisible particles in ABC, the project would focus on modifying the detector configuration such that its sensitivity toward a given particle (identified using its true mass) is set to zero. Then, one could reconstruct this particle using the energy-momentum information of the final state and another particle that is tagged.

5.4. Computation Speed Improvements

- We note that `pymcabc` software can be improved considerably in terms of speed. For instance, one could use a GPU-based computation or parallel computation. Both approaches have the potential to improve the speed of computation as well as the generation of events.

6. Conclusion

In this paper, we have introduced a Monte Carlo based software, `pymcabc` for the ABC model. This software is aimed at a particle physics novice who wishes to learn at an early stage the tools used by, say, the LHC collaborations. By using the ABC model, one can calculate the differential cross-section by pencil and paper and compare the same with the `pymcabc` software prediction. We have also compared the results of the `pymcabc` software against that of the `MadGraph5_aMC@NLO` and found consistency in the results. Moreover, the ability to generate events allows one to understand the kinematical information of events.

We have also shown how the ABC model can be used to make the lineshape of a heavy mediator. This technique has been used in real experiments with the Z bosons at LEP colliders. Moreover, this technique might also be used in the Future Lepton Colliders for the Higgs lineshape. Furthermore, we illustrated an important analysis technique called the Recoil Mass Reconstruction, which will also be deployed in the Future experiments to extract, for example, the Higgs boson properties. The `pymcabc` software can be improved and extended in several ways. As examples, we have presented ideas as exercises that can be used as starting points for the reader.

---

i. The source code can be downloaded from `https://github.com/amanmdesai/pymcabc`.

ii. The version (1.1) used for the publication is archived `https://doi.org/10.5281/zenodo.7792881`.

iii. https://github.com/GkAntonius/feynman

iv. `https://github.com/scikit-hep/uproot5`

v. `https://root.cern/`

vi. `https://root.cern/manual/python/`

1. J. Alwall *et al.*, The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations, *J. High Energ. Physics* 2014 (2014) 79, `https://doi.org/10.1007/jhep07(2014)079`.

2. T. Sjöstrand *et al.*, An introduction to PYTHIA 8.2, *Comput.*

Phys. Commun. **191** (2015) 159, `https://doi.org/10.1016/j.cpc.2015.01.024`.

3. D. J. Griffiths, Introduction to elementary particles; 2nd rev. version, Physics textbook (Wiley, New York, NY, 2008), `https://cds.cern.ch/record/111880`.
4. I. J. R. Aitchison and A. J. G. Hey, Gauge Theories in Particle Physics: A Practical Introduction, Volume 1 : From Relativistic Quantum Mechanics to QED, Fourth Edition (Taylor & Francis, 2013), `https://doi.org/10.1201/b13717`.
5. P. Kraus and D. J. Griffiths, Renormalization of a model quantum field theory, *Am. J. Phys.* **60** (1992) 1013, `https://doi.org/10.1119/1.16980`.
6. A. Papaefstathiou, How-to: write a parton-level Monte Carlo particle physics event generator, *Eur. Phys. J. Plus* **135** (2020) 497, `https://doi.org/10.1140/epjp/s13360-020-00499-1`.
7. R. L. Workman *et al.*, Review of Particle Physics, *Progr. Theor. Exp. Phys.* **2022** (2022) 083C01, `https://doi.org/10.1093/ptep/ptac097`.
8. R. Brun and F. Rademakers, ROOT: An object oriented data analysis framework, *Nucl. Instrum. Meth. A* **389** (1997) 81, `https://doi.org/10.1016/S0168-9002(97)00048-X`.
9. F. E. James, Monte Carlo phase space, Cern Academic Training Lecture (Cern, Geneva, 1968), `https://doi.org/10.5170/CERN-1968-015`.
10. P. Artoisenet *et al.*, Automatic spin-entangled decays of heavy resonances in Monte Carlo simulations, *JHEP* **2013** (2013) 015, `https://doi.org/10.1007/JHEP03(2013)015`.
11. C. R. Harris *et al.*, Array programming with NumPy, *Nature* **585** (2020) 357, `https://doi.org/10.1038/s41586-020-2649-2`.
12. J. Pivarski *et al.*, *Uproot* (2024), `https://doi.org/10.5281/zenodo.10699405`.
13. J. D. Hunter, Matplotlib: A 2D graphics environment, *Comput. Sci. Eng.* **9** (2007) 90, `https://doi.org/10.1109/MCSE.2007.55`.
14. T. pandas development team, pandas-dev/pandas: Pandas (2020), `https://doi.org/10.5281/zenodo.3509134`.
15. C. Degrande *et al.*, UFO - The Universal FeynRules Output, *Comput. Phys. Commun.* **183** (2012) 1201, `https://doi.org/10.1016/j.cpc.2012.01.022`.
16. A. Alloul *et al.*, FeynRules 2.0 - A complete toolbox for tree-level phenomenology, *Comput. Phys. Commun.* **185** (2014) 2250, `https://doi.org/10.1016/j.cpc.2014.04.012`.
17. A. Semenov, LanHEP: A Package for the automatic generation of Feynman rules in field theory. Version 3.0, *Comput. Phys. Commun.* **180** (2009) 431, `https://doi.org/10.1016/j.cpc.2008.10.012`.
18. A. Desai, UFO Files for the Toy ABC Model (2023), `https://doi.org/10.5281/zenodo.8163866`.
19. M. Galli, E. Tejedor, and S.Wunsch, A New PyROOT: Modern, Interoperable and More Pythonic, *EPJ Web Conf.* **245** (2020) 06004, `https://doi.org/10.1051/epjconf/202024506004`.
20. R. Aaij *et al.*, Measurement of the $B_s^0 \to \mu^+\mu^-$ branching fraction and effective lifetime and search for $B^0 \to \mu^+\mu^-$ decays, *Phys. Rev. Lett.* **118** (2017) 191801, `https://doi.org/10.1103/PhysRevLett.118.191801`.
21. G. Bernardi *et al.*, The Future Circular Collider: a Summary for the US 2021 Snowmass Process (2022), `https://doi.org/10.48550/arXiv.2203.06520`.
22. A. Desai, UFO Files for the Toy ABC+D Model (2023), `https://doi.org/10.5281/zenodo.8163874`.
23. A. Desai, An introduction to semi-automated matrix element computation in particle physics (2023), `https://doi.org/10.5281/zenodo.8331825`.
24. A. Desai, amanmdesai/pymcabc (2023), `https://doi.org/10.5281/zenodo.7792881`.
25. M. H. Seymour and M. Marx, Monte Carlo Event Generators, In 69th Scottish Universities Summer School in Physics: LHC Physics (2013) pp. 287-319, `https://doi.org/10.1007/978-3-319-05362-2_8`.
26. J. Pivarski, Uproot, Zenodo `https://doi.org/10.5281/zenodo.8122179`.