# An introduction to semi-automated matrix element computation in particle physics

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

Received 9 September 2023; accepted 17 January 2024

This paper presents a semi-automated method to compute matrix elements at the Leading Order and the Next-to-Leading Order in the ABC model of particle physics. The ABC model consists of three scalar particles and they interact only when all three particles are present in the interaction. In the Next-to-Leading Order calculations, one often has to deal with ultraviolet divergences. Some of the techniques such as Wick's rotation, Feynman parameterisation, and dimensional regularization which are useful in Next-to-Leading Order computations are presented by applying them on the ABC model calculations and also implemented in python based programs using Computer Algebra System

*Keywords:* Matrix element; next-to-leading order; ABC model; sympy; particle physics.

DOI: https://doi.org/10.31349/RevMexFisE.21.020207

## 1. Introduction

Matrix element computation is crucial to determine the cross-section of a particle physics process. This requires one to evaluate for example, the so-called tree-level or Leading Order (LO) Feynman diagram contribution. Many a calculations in the present times require particle physicist to evaluate Next-to-Leading Order (NLO) matrix element for understanding the underlying physics that is observed at the collider experiments. These calculations are usually automated in particle physics and makes use of mathematica based softwares such as for example FeynCalc [1]. This paper explores the underlying physics and mathematics that is used when computing both the LO and NLO matrix elements in the ABC model in a semi-automated method using Computer Algebra System (CAS) in `Python`, which is because of the free and easy availability of `Python`. We have provided all the programs within this paper, which readily allows the reader to reproduce the results presented here. All the python programs are also available as independent `jupyter` notebooks from `https://github.com/amanmdesai/ABC_NLO_sympy_calculation`.

For pedagogical reasons, we have used the ABC model to present a way in which one can evaluate the matrix element at the LO and NLO. Before diving into physics framework developed here, we shall highlight major motivation which necessitates NLO computations in particle physics.

Particle physics aims to study the fundamental particles and their interactions. It has been instrumental in developing the so-called Standard Model (SM) of particle physics that describes all known interactions among matter with the exception of gravity. The SM has thus far survived all experimental tests throughout the years. The theoretical prediction of Higgs Boson in the 1960's was corroborated by the recent discovery in 2012 by the ATLAS and CMS collaboration at Large Hadron Collider, thereby establishing the SM as a credible theory [2, 3]. The SM framework includes fermions, which are grouped as leptons ($e,\ \mu,\ \tau,\ \nu_e,\ \nu_\mu,\ \nu_\tau$) and quarks($u,\ d,\ s,\ c,\ b,\ t$) and their anti-particles, and bosons, such as the scalar spin-zero Higgs Boson ($H$) and the spin-1 gauge bosons ($\gamma,\ g,\ Z,\ W^+,\ W^-$). One of the main goals of particle physics experiments since then is to study the SM at the precision frontier at high energy collider experiments *i.e.* to measure the SM observables at a better precision while the other major goal is to look for new physics beyond the SM as several fundamental physics questions remain unanswered within the SM. In recent years the precision studies, as well searches for new physics, has necessitated that the predictions of theory are obtained at higher-orders in perturbation theory rather than the tree-level computations.

One of the motivations for NLO computation is that it leads to newer channels which may be useful in discovery of new physics. As an example, consider one of the clean Higgs boson discovery channel $H \to \gamma\gamma$ [2, 3]. The tree-level SM does not allow this decay as photons are massless, and the Higgs boson does not couple directly to photons. However, this channel is possible if there exists a loop of fermion as shown in Fig. 1. This loop is just one of the NLO contributions to the Higgs decay width.

One of the problems faced in computing matrix elements of scattering processes at higher-orders is that the integrals tend to infinity. Often this leads to integrals of the form (Note that in this paper $p, p_i, q, q_i$ represent the four-momenta):

$$\int^{\infty} d^4q/q^4 \propto \int^{\infty} q^3 dq/q^4 \log(q)|^{\infty}. \tag{1}$$

It is apparent from the above equation that the result is infinity, and this is often referred to as the "ultraviolet divegence" [4]. In general, whether a Feynman diagram (i.e., the amplitude for a given process) is divergent or convergent can be determined using the concept known as Superficial degree of divergence [5]. This is quantified by counting the power of momentum in the numerator subtracted by the power of momentum in the denominator of the amplitude for a given

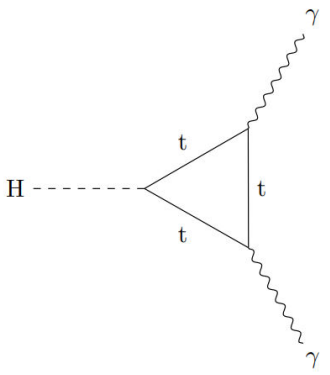


FIGURE 1. Representative Feynman diagram for the decay process $H \to \gamma\gamma$ at NLO.

process and denoted as $D$. Accordingly, if $D$ is positive or zero, the integral is certainly divergent, otherwise if the value of $D$ is negative for the given Feynman diagram and so its sub-diagrams also, then the integral is convergent as required by the Weinberg's theorem; the reason for using 'superficial' is that not only the complete diagram but also the sub-diagrams must have $D < 0$. The parameter D is useful in identifying whether the theory has a finite number of divergent diagrams, then one can 'regularise' (*i.e.*, isolate the ultraviolet divergences) and 'renormalise' the theory to obtain finite amplitudes; if the number of divergent diagrams is not finite then one knows that that the theory is renormalisable. Techniques have been developed to overcome this situation and they require one to invoke the ideas related to dimensional regularization, renormalization, among others [6]. In addition, one often also uses mathematical techniques such as the Feynman parametric integrals, that separates the finite part of the integral from the divergent part.

In this article, we have used the ABC model [4] that consists of three scalar particles to illustrate methods that help in semi-automation of calculation with a CAS such as `sympy` package [7]. We have evaluated the expressions for the amplitudes of decay and the scattering processes at the LO and NLO. The organization of this paper is as follows: In Sec. 2, we introduced the ABC model and some theoretical background. In Sec. 3, we have calculated the matrix element for the decay and scattering processes at the tree-level using the CAS in `Python`. We have also shown how one can obtain the decay-width/cross-section for a given decay/scattering process. In Sec. 4, we have demonstrated how one could obtain the matrix element expression at NLO for decay process, and NLO for scattering processes such as the triangle, bubble, and box diagrams, all of which were implemented with `Python`. In Sec. 5, we have presented some useful mathematical techniques and their computational implementation are useful in isolating the infinities that occur in the computation of NLO diagrams.

## 2. Toy ABC Model

The ABC toy model is primarily used in particle physics when introducing new concepts viz. Feynman diagrams, renormalization, dimensional regularisation [4, 8, 9]. Moreover, recently the model was used to introduce the techniques of Monte Carlo Event Generator at the Leading Order [10]. The model name, "ABC", refers to the three scalar particles (called $A,\ B,\ C$) that comprise it and any interaction between these particles is only allowed when all three particles share a common vertex (interaction point). The Lagrangian for the ABC model is given by the following equation [4]:

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

Each term in the Lagrangian has a dimension of mass 4 (assuming $\hbar = c = 1$) owing to the fact that the action given by $S = \int \mathcal{L}d^4x$ is dimensionless in the natural units. Looking at the term $ig\phi_A\phi_B\phi_C$, we see that for dimensional consistency the coupling constant of the theory ($g$) must also have a dimension of mass. This has direct consequences when computing the matrix elements at NLO [9].

In the ABC theory, the free Lagrangian is given as follows:

$$\mathcal{L}_{\text{free}} = \sum_{i=A,B,C}\left(\frac{1}{2}\partial_\mu\phi_i\partial^\mu\phi_i - \frac{1}{2}m_i^2\phi_i^2\right), \quad (3)$$

and the interaction term is given by:

$$\mathcal{L}_{\text{int}} = ig\phi_A\phi_B\phi_C. \quad (4)$$

In order to obtain the equation of motion of free particles in the ABC model, one uses the Euler-Lagrange equation, which for a Lagrangian $\mathcal{L}(\phi_i, \partial_\mu\phi_i)$ is given by:

$$\frac{\partial}{\partial x^\mu}\frac{\partial\mathcal{L}}{\partial(\partial_\mu\phi_i)} = \frac{\partial\mathcal{L}}{\partial\phi_i}. \quad (5)$$

The Feynman rules of a theory as obtained from Quantum field theory are a mnemonic for calculating the matrix element (dynamics) of a physics process. A summarized procedure to obtaining these rules from the Lagrangian is:

- Remove the fields (particle wave-functions) from the interacting part of the Lagrangian. The quantities which remain form the vertex contributions.
- Apply Euler-Lagrange's equation on the Free Lagrangian. This gives the inverse of the propagator (additional factor of complex number $i$ needs to be multiplied).

The interaction Lagrangian Eq. (4) contains the constant $-ig$, which is effectively the vertex contribution to a Feynman diagram. Applying the Euler-Lagrange Equation (5) to the free Lagrangian Eq. (3), we obtain the Klein-Gordon equation:

$$\partial^\mu\partial_\mu\phi - m^2\phi^2 = 0. \quad (6)$$

Thus, we see that each free particle in the ABC model follows the Klein-Gordon equation. In the momentum space ($p_\mu = -i\partial_\mu$) this reduces to:

$$p^2 - m^2 = 0. \tag{7}$$

The propagator of the theory is then given by taking inverse of the above equation and multiplying it by a factor of $i$:

$$\frac{i}{p^2 - m^2}. \tag{8}$$

The procedure to obtain the Klein-Gordon equation from the free Lagrangian can also be performed using (`sympy`). We illustrate this procedure by considering the free part of the Lagrangian for particle A. The computing environment may be setup as follows:

```
from sympy import Symbol,Function
from sympy import euler_equations
```

where we imported the necessary submodules of `sympy`. We then define the symbols and the wavefunction $\phi$.

```
t = Symbol('t')
x = Symbol('x')
y = Symbol('y')
z = Symbol('z')
m = Symbol('m_A')
g = Symbol('g')
phi = Function('\\phi_A')
```

The Lagrangian for the free particle A can be implemented in `sympy` as follows:

```
L = (1/2)*(phi(x,y,z,t).diff(t)**2
    - phi(x,y,z,t).diff(x)**2
    - phi(x,y,z,t).diff(y)**2
    - phi(x,y,z,t).diff(z)**2
    - (m*phi(x,y,z,t))**2)
print(L)
```

Euler-Lagrange equation is provided as a submodule in `sympy` and this package may be called as follows:

```
result = euler_equations(L, phi(x,y,z,t),
[x,y,z,t])
print(result[0])
```

The output of this program is then given as:

$$-m_A^2\phi_A(x,y,z,t) - \frac{\partial^2}{\partial t^2}\phi_A(x,y,z,t) + \frac{\partial^2}{\partial x^2}\phi_A(x,y,z,t) + \frac{\partial^2}{\partial y^2}\phi_A(x,y,z,t) + \frac{\partial^2}{\partial z^2}\phi_A(x,y,z,t) = 0, \tag{9}$$

TABLE I. Feynman rules of the ABC model.

| Diagram Component | Feynman Rule |
|---|---|
| External Lines | 1.0 |
| Vertex | $-ig$ |
| Propagator | $\frac{i}{q^2-m^2}$ |
| Momentum Conservation | $\delta^4(k_1+k_2+k_3)$ where $k_i$ is momenta of lines at vertex |

where the above equation is the expanded version of the Klein-Gordon Equation $(\partial^\mu\partial_\mu - m^2)\phi = 0$.

The Feynman rules of the ABC model are summarized in Table I. In addition, one also needs to include the Feynman rules that are required to conserve four-momentum at each vertex. This is ensured by introducing a four-dimensional Dirac delta function for each vertex.

In order to automate matrix element calculations at LO and NLO for decay and scattering processes in the ABC model, we adapt the Feynman rules in `sympy` syntax. As in the previously implemented syntax, we import the necessary `sympy` modules and define the symbols:

```
from sympy import Symbol, symbols
from sympy import I, simplify,pi
from sympy import DiracDelta
from sympy import integrate,oo,factor
mA,mB,mC = symbols('m_A m_B m_C') # masses
g = Symbol('g')# coupling
q = Symbol('q') # internal momentum
p1,p2,p3,p4 = symbols('p_1:5') #externalmomenta
```

The module defining the Feynman rules is then developed as a class with each Feynman rule implemented as a function.

```
# Feynman rules for the ABC model
class FeynmanRules:
 def __init__():
     pass

 def propagator(q,m):
     prop = I/(q**2 - m**2)
     return prop

 def vertex():
     return -I*g

 def external_line():
     return 1

 def delta(k1, k2, k3):
     delta = (2*pi)**4*DiracDelta(k1 + k2 + k3)
     return delta

 def remove_factor():
     return I/((2*pi)**4*DiracDelta(0))

 def integral_factor():
     return 1/((2*pi)**4)
```

We have now introduced and also developed the necessary minimum tools/modules that allows us to carry out ma-

trix element computation at the Leading order as well as at the NLO in the ABC model.

Note that one may also use the superficial degree of divergence introduced in the previous section to determine if a diagram is divergent or convergent. For the ABC model, one may use the following expression for $D$ which is derived in polar coordinates (which is simpler to interpret) in Ref. [8]:

$$D = 3 - \frac{3V}{2} - \frac{E}{2}$$

where $V$ is number of vertices, $E$ is the number of external lines. A given Feynman diagram is said to be divergent if $D \geq -1$ (note the condition is slightly different from Weinberg's theorem owing to different coordinate system, the physics interpretation remains unchanged).

## 3. Semi-automating tree-level computation

With the implementation of Feynman rules in the `sympy` package, we now evaluate the matrix element for decay process and for a scattering process at the tree-level. This allows us to test the software and compare the results with books [4].

### 3.1. Decay width process at leading order

A decay process of a particle refers to its transformation into two or more daughter particles. Owing to the type of interaction in the ABC model, a given particle for instance $A$, can only decay to particle's $B$ and $C$ at the tree-level. Let us assume that the mass $m_A > m_B + m_C$ in which case the decay is kinematically feasible. The Feynman diagram for the decay process $A \rightarrow B\ C$ is given in Fig. 2. As given in Grifiths [4], we expect the matrix element for the decay process to be $\mathcal{M} = g$. To evaluate the matrix element for decay process using `sympy`, one can use the following script in conjunction with the `FeynmanRules` module prepared in the previous section:

```
fr = FeynmanRules
ME = fr.external_line()**3
ME = ME*fr.vertex()*fr.delta(p1,-p2,-p3)
ME = ME.subs({p1:p2+p3})*fr.remove_factor()
print(ME)
```

where we have substituted $p_1 = p_2 + p_3$ to enforce the energy-momentum conservation. The output of this program is $\mathcal{M} = g$, which is the expected result. The decay width is given as a product of available phase space and square of absolute matrix element [4]:

$$\Gamma = \frac{|\mathbf{p}_A|}{8\pi m_A^2} * |\mathcal{M}^2|, \tag{10}$$

where $\mathbf{p}_A$ is absolute three-momentum of the decaying particle and $m_A$ is mass of the particle. Substituting the value of matrix element obtained, the decay width $\Gamma$ is given by:

$$\Gamma = \frac{g^2|\mathbf{p}_A|}{8\pi m_A^2}. \tag{11}$$

The above equation can also be implemented using the following `sympy` code:

```
pA = Symbol('|\\textbf{p}_A|')
gamma = pA/(8*pi *(mA**2)) * abs(ME)**2
```

### 3.2. Scattering processes at the leading order

We now evaluate the scattering amplitude for an $s$-channel process in the ABC model, where $s$-channel is defined such that the four-momenta of internal particle is sum of four-momenta of incoming particles. In particular, we consider the following process $A\ B \rightarrow C \rightarrow A\ B$. The Feynman diagram of this process is given in Fig. 3. One can use the `FeynmanRules` module prepared earlier to evaluate the matrix element for this process using the following snippet:

```
from sympy import integrate,oo
fr = FeynmanRules
ME = fr.external_line()**4
ME = ME * fr.vertex()**2
ME = ME * fr.propagator(q,mC)
ME = ME * fr.delta(p1,p2,-q)
ME = ME * fr.delta(-p3,-p4,q)
ME = integrate(ME, (q,-oo,oo))*fr.integral_factor()
ME = ME * fr.remove_factor()
ME_s = ME.subs({p3: -p4 +p1 + p2})
```

where we have integrated over the internal momentum. This integration in principle is carried over a four-dimensional Dirac delta function using a $d^4q$ as the variable for integra-

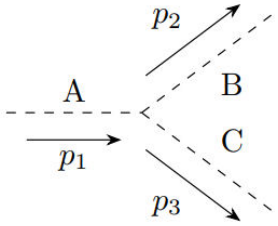


FIGURE 2. Representative Feynman diagram for the decay process in the ABC model.

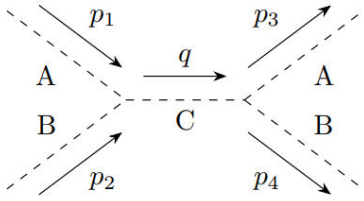


FIGURE 3. Representative Feynman diagram for the scattering process in the $s$-channel.

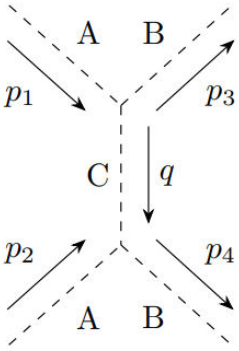


FIGURE 4. Representative Feynman diagram for the scattering process in the $t$-channel

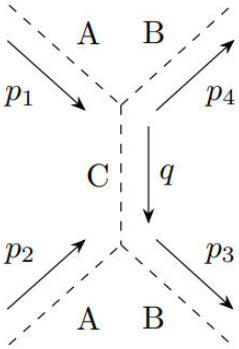


FIGURE 5. Representative Feynman diagram for the scattering process in the $u$-channel.

tion. However, in CAS, we have evaluated the integral over 1D Dirac delta function and $dq$ being the integration variable. Finally, we substitute $p_3 = p_1 + p_2 - p_4$, which ensures the energy-momentum conservation.

The program finally gives as output:

$$\frac{g^2}{-m_C^2 + (p_1 + p_2)^2}, \tag{12}$$

which is the matrix element as one would obtain [4].

As exercises one may repeat the above procedure for calculating $t$-channel and $u$-channel matrix elements. The $t$- and $u$-channel diagrams are presented in Figs. 4 and 5. The $t-$channel refers to the channel where the internal momenta is given as $q = p_1 - p_3$, whereas for the $u-$channel the internal momenta is given as $q = p_1 - p_4$. Moreover, a $u-$channel process is viable only when there are identical particles in the final state.

The differential cross-section for a scattering process is given as a product of phase-space (kinematics) and square of absolute amplitude (dynamics) by [4]:

$$\frac{d\sigma}{d\Omega} = \frac{|\mathbf{p}_f|}{128\pi^2 E_{cm}^2 |\mathbf{p}_i|} |\mathcal{M}|^2, \tag{13}$$

where $E_{cm}$ is the center of mass energy of a given process and $d\Omega$ is the angular area. Here $\mathcal{M}$ refers to the sum of $\mathcal{M}_s$, $\mathcal{M}_t$ if the process occurs in the $s-t$ channel or $\mathcal{M}_t$, $\mathcal{M}_u$ if the process happens in the $t-u$ channel. The $t$-channel matrix element is given by:

$$\mathcal{M}_t = \frac{g^2}{-m_C^2 + (-p_2 + p_4)^2}. \tag{14}$$

Therefore one obtains the following expression for the differential cross section for the process $A\,B \to A\,B$:

$$\frac{d\sigma}{d\Omega} = \frac{|\mathbf{p}_f| \left| \frac{g^2}{m_C^2 - (p_2 - p_4)^2} + \frac{g^2}{m_C^2 - (p_1 + p_2)^2} \right|^2}{128\pi^2 E_{cm} |\mathbf{p}_i|}. \tag{15}$$

The method can be implemented in `sympy` as follows (assuming process occurs in the $s$ and $t$ channels):

```
p_f = Symbol('\\textbf{p}_f')
p_i = Symbol('\\textbf{p}_i')
Ecm = Symbol('E_{cm}')
diff_xsec =abs(p_f)*abs(1/p_i) \
*(1/(128*pi**2 *Ecm))*abs(ME_s + ME_t)**2
```

## 4. Introduction to semi-automated NLO computations

The techniques discussed in the previous sections may also be adapted to work for semi-automated determination of matrix element at NLO. For the tree-level decay process and for the tree-level scattering process discussed in Sec. 3, the number of possible processes rises at NLO. Each diagram at NLO introduces new internal lines and additional vertices. For instance, in the decay process at NLO, there are two additional vertices and three internal lines. On the other hand, an NLO scattering diagram includes four internal lines or two additional vertices. This implies that the possible number of diagrams increases to eight for a given scattering process in a channel.

In this section we will show using `sympy` how one can obtain matrix element expressions for the decay process at NLO as well as the matrix element for the vertex correction, propagator correction and box diagram, which are the possible contributions at the NLO for a given channel in the ABC model.

### 4.1. Decay Width Amplitude at NLO

In Sec. 3, we evaluated the decay width amplitude at the tree-level and the Feynman diagram for the tree-level process is given in Fig. 2. A possible NLO contribution to this process is given in Fig. 6. As a first step, we evaluate the decay width amplitude for the Feynman diagram given in Fig. 6 assuming, as previously that $m_A > m_B + m_C$.

This diagram involves a triangular loop which consists of three internal lines. There are three vertices in this diagram,

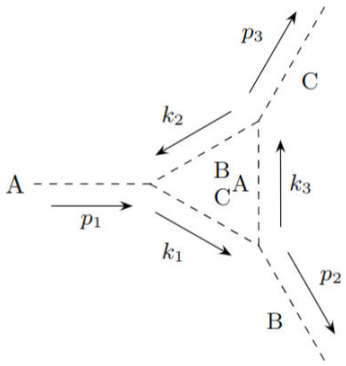


FIGURE 6. Representative Feynman diagram for the decay process at NLO.

which is, two more compared to the Decay width at the Leading Order. Each vertex brings in a factor of $-ig$, and, therefore, we are evaluating the matrix element at an order of $g^3$.

The following scripts are used to construct the matrix element for this diagram with `sympy` package:

```
fr = FeynmanRules
ME = fr.external_line()**3
ME = ME*fr.vertex()**3
ME = ME*fr.delta(p1,-k1,k2)
ME = ME*fr.delta(k1,-p2,-k3)
ME = ME*fr.delta(k3,-k2,-p3)
ME = ME*fr.propagator(k1,mC)
ME = ME*fr.propagator(k2,mB)
ME = ME*fr.propagator(k3,mA)
ME = integrate(ME,(k1,-oo,oo))
ME = ME * fr.integral_factor()
ME = integrate(ME,(k2,-oo,oo))
ME = ME.subs({p1:p2+p3})*fr.remove_factor()
```

The matrix element obtained (before carrying out any integration) is of the form:

$$\mathcal{M} = \int d^4k_1 \int d^4k_2 \int d^4k_3 \frac{ig^3\delta\left(-k_1+k_2+p_1\right)\delta\left(k_1-k_3-p_2\right)\delta\left(-k_2+k_3-p_3\right)}{\left(k_1^2-m_C^2\right)\left(k_2^2-m_B^2\right)\left(k_3^2-m_A^2\right)}. \tag{16}$$

The result before integrating the variable $k_3$ is of the following form[i]:

$$\mathcal{M} \propto -\int d^4k_3 \frac{16i\pi^4 g^3}{F(k_3)}, \tag{17}$$

where the denominator (the function $F(k_3)$) is given by:

$$F(k_3) = \left(k_3-m_A\right)\left(k_3+m_A\right)\left(-k_3+m_B+p_3\right)\left(k_3+m_B-p_3\right)\left(k_3-m_C+p_2\right)\left(k_3+m_C+p_2\right). \tag{18}$$

For carrying out the last integral, we have to take into account that we are integrating over a 4D space where $q^2 = -q_0^2+q_1^2+q_2^2+q_3^2$.

In Sec. 5, we will discuss some ways in which one can evaluate integrals as above. But before going through that we would like to explore the various scattering processes that can occur at one loop in the ABC model.

### 4.2. Scattering at NLO: Vertex Corrections or Triangle diagrams

In scattering processes, there are possible types of contributions viz. triangular, bubble and box diagrams. To begin with, we consider triangular NLO diagram contribution (Fig. 7) to the $s-$channel process ($A\ B \to A\ B$) as in Fig. 3, which is similar to the triangle as seen in the previous section. While we consider the triangular region on the right vertex in the Feynman diagram, there is also another possibility to have the triangular diagram at the left vertex. However, the scattering process discussed here is different from the decay process at NLO. For instance, it has an additional vertex which leads to an additional internal line (propagator).

The triangle diagram that we are considering here is also known as the vertex correction diagram, because the vertex in a tree-level diagram is now replaced by a triangle, which is effectively a correction to the vertex.

The following program is used to obtain expression for matrix element by applying the Feynman rules to the Feynman diagram in Fig. 7:

```
fr = FeynmanRules
ME = fr.external_line()**4
ME = ME*fr.vertex()**4
ME = ME*fr.delta(p1,p2,-q)
ME = ME*fr.delta(q,-k1,-k2)
ME = ME*fr.delta(k2,k3,-p4)
ME = ME*fr.delta(k1,-k3,-p3)
ME = ME*fr.propagator(q,mC)
ME = ME*fr.propagator(k1,mC)
ME = ME*fr.propagator(k2,mB)
ME = ME*fr.propagator(k3,mA)
ME = ME*fr.integral_factor()**4
ME = integrate(ME,(k1,-oo,oo))
ME = integrate(ME,(k2,-oo,oo))
ME = integrate(ME,(q,-oo,oo))
```

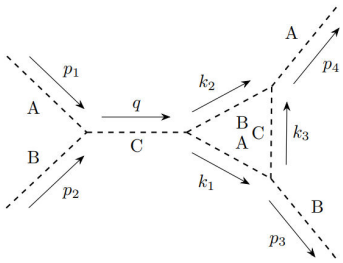


FIGURE 7. Representative Feynman triangle diagram for the scattering process at NLO in the $s$-channel.

$$\mathcal{M} = \int d^4k_1 \int d^4k_2 \int d^4k_3 \frac{g^4 \delta\left(-k_1 - k_2 + q\right) \delta\left(k_1 - k_3 - p_3\right) \delta\left(k_2 + k_3 - p_4\right) \delta\left(p_1 + p_2 - q\right)}{\left(k_1^2 - m_C^2\right)\left(k_2^2 - m_B^2\right)\left(k_3^2 - m_A^2\right)\left(-m_C^2 + q^2\right)}, \tag{19}$$

which after integrating over all internal momenta except $k_3$ gives:

$$-\mathcal{M} \propto \int d^4k_3 \frac{g^4 \delta\left(p_1 + p_2 - p_3 - p_4\right)}{F(k_3)}, \tag{20}$$

where the function $F(k_3)$ is given by:

$$\begin{aligned} F(k_3) = &\left(k_3 - m_A\right)\left(k_3 + m_A\right)\left(k_3 - m_B - p_4\right)\left(k_3 + m_B - p_4\right) \\ &\times \left(m_C - p_1 - p_2\right)\left(m_C + p_1 + p_2\right)\left(k_3 - m_C + p_1 + p_2 - p_4\right)\left(k_3 + m_C + p_1 + p_2 - p_4\right). \end{aligned} \tag{21}$$

As noted earlier in the case of decay width matrix element, we see that the last integral will lead to infinities.

### 4.3. Scattering at NLO: Propagator correction or Bubble diagrams

In this section, we study the corrections to the propagator by evaluating the matrix element of the process given in Fig. 8.

We apply the procedure to evaluating Feynman diagrams and make use of the following python snippet to evaluate the matrix element for the bubble diagram in Fig. 8.

```
fr = FeynmanRules
ME = fr.external_line()**4
ME = ME * fr.vertex()**4
ME = ME * fr.propagator(k1,mC)
ME = ME * fr.propagator(k2,mA)
ME = ME * fr.propagator(k3,mB)
ME = ME * fr.propagator(k4,mC)
ME = ME * fr.delta(p1,p2,-k1)
ME = ME * fr.delta(k1,-k2,k3)
ME = ME * fr.delta(k2,-k3,-k4)
ME = ME * fr.delta(k4,-p3,-p4)
ME = ME *fr.integral_factor()**4
ME = integrate(ME, (k1,-oo,oo))
ME = integrate(ME, (k2,-oo,oo))
ME = integrate(ME, (k4,-oo,oo))
```

where the matrix element before integrating is given as:

$$\begin{aligned} \mathcal{M} = &\int d^4k_1 \int d^4k_2 \int d^4k_3 \int d^4k_4 \\ &\times \frac{g^4 \delta\left(-k_1 + p_1 + p_2\right) \delta\left(k_1 - k_2 + k_3\right) \delta\left(k_2 - k_3 - k_4\right) \delta\left(k_4 - p_3 - p_4\right)}{\left(k_1^2 - m_C^2\right)\left(k_2^2 - m_A^2\right)\left(k_3^2 - m_B^2\right)\left(k_4^2 - m_C^2\right)}, \end{aligned} \tag{22}$$

and after performing the integral over the three internal momenta, namely, $k_1$, $k_2$, and $k_4$, we obtain the following expression:

$$-\mathcal{M} \propto \int d^4k_3 \frac{g^4 \delta\left(p_1 + p_2 - p_3 - p_4\right)}{F(k_3)}, \tag{23}$$

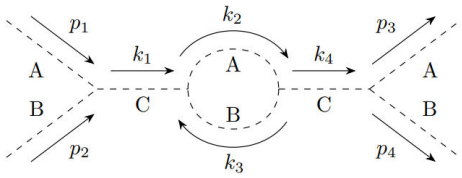


FIGURE 8. Representative Feynman Bubble diagram for the scattering process at NLO in the $s$-channel.

where the function $F(k_3)$ is given by:

$$F(k_3) = (k_3 - m_B)(k_3 + m_B)(m_C - p_1 - p_2)^2 \times (m_C + p_1 + p_2)^2 (k_3 - m_A + p_1 + p_2) \times (k_3 + m_A + p_1 + p_2). \quad (24)$$

It may be noted that this correction is also possible in any of the external lines. Therefore, the total number of bubble diagrams in any given $s-$, $t-$ or $u-$ channels is 5.

### 4.4. Scattering at NLO: Box diagrams

Box diagrams are important in particle physics. In QED for instance, Delbruck scattering or scattering of photon-by-photon ($\gamma\gamma \to \gamma\gamma$) proceeds via the box diagram and hence evaluating such diagrams is important, since photon-photon scattering is not known to occur at the tree-level processes.

In the ABC model, there exists only one box diagram for each $s$-, $t$- and $u$-channel process at the NLO. The following python snippet is used to evaluate the matrix element for the box diagram as given in Fig. 9.

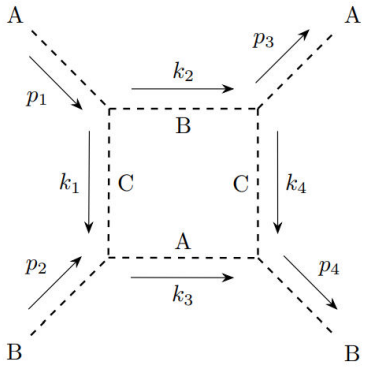


FIGURE 9. Representative Feynman Box diagram for the scattering process at NLO.

```
fr = FeynmanRules
ME = fr.external_line()**4
ME = ME * fr.vertex()**4
ME = ME * fr.propagator(k1,mC)
ME = ME * fr.propagator(k2,mB)
ME = ME * fr.propagator(k3,mA)
ME = ME * fr.propagator(k4,mC)
ME = ME * fr.delta(p1,-k1,-k2)
ME = ME * fr.delta(k2,-p3,-k4)
ME = ME * fr.delta(k4,k3,-p4)
ME = ME * fr.delta(p2,k1,-k3)
ME = ME *fr.integral_factor()**4
ME = integrate(ME, (k1,-oo,oo))
ME = integrate(ME, (k2,-oo,oo))
ME = integrate(ME, (k4,-oo,oo))
```

where the matrix element before integrating is given as:

$$\mathcal{M} = \int d^4k_1 \int d^4k_2 \int d^4k_3 \int d^4k_4 \frac{g^4 \delta(-k_1 - k_2 + p_1)\,\delta(k_1 - k_3 + p_2)\,\delta(k_2 - k_4 - p_3)\,\delta(k_3 + k_4 - p_4)}{(k_1^2 - m_C^2)(k_2^2 - m_B^2)(k_3^2 - m_A^2)(k_4^2 - m_C^2)}, \quad (25)$$

and after performing the integral over the three internal momenta, namely, $k_1$, $k_2$, and $k_4$, we obtain the following expression:

$$\mathcal{M} \propto -\int d^4k_3 \frac{g^4 \delta(p_1 + p_2 - p_3 - p_4)}{F(k_3)}, \quad (26)$$

where the function $F(k_3)$ is given by:

$$F(k_3) = (k_3 - m_A)(k_3 + m_A)(k_3 - m_C - p_4)(k_3 + m_C - p_4)(k_3 - m_B - p_3 - p_4) \times (k_3 + m_B - p_3 - p_4)(k_3 - m_C + p_1 - p_3 - p_4)(k_3 + m_C + p_1 - p_3 - p_4). \quad (27)$$

We will explore methods to integrate the above expression.

## 5. Some Mathematical techniques to overcome infinities at NLO

In this section we present some techniques that aid in separating the infinite part of the integral from the finite part, thereby reducing the answer as a sum of finite and infinite parts. To illustrate the techniques involved, we use the matrix element for the bubble diagram which was obtained in Eq. (23) and factors given in Eq. (24). The reader is encouraged to apply all the techniques discussed in this section to other NLO processes.

### 5.1. Wick's Rotation

The numerator in Eq. (23) has a term of the type: $d^4q$ where $q$ is the four momentum of the propagator. From special relativity we know that $q^2 = -q_0^2 + q_1^2 + q_2^2 + q_3^2$,

where the subscripts 0 stands for the time coordinate and $1, 2, 3$ represent the spatial coordinate. By Wick's rotation we mean to perform a rotation which allow us the reinterpret Minkowski coordinate to the Euclidean coordinate. Accordingly, let $q_0 \rightarrow iq_0$. We therefore see that $q^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2$, which is the 4 dimension equivalent to the 3 dimension radius $R^2 = x^2 + y^2 + z^2$ in the Euclidean space. One can then write $d^4q$ as $q^3 dq d\Omega$.

This leads to Eq. (23) to the following form, where for simplicity we replace the variable $k_3$ with $q$:

$$\mathcal{M} \propto -\int d\Omega \int q^3 dq \frac{g^4 \delta\left(p_1 + p_2 - p_3 - p_4\right)}{F(q)}, \tag{28}$$

with $F(q)$ given as:

$$\begin{aligned} F(k_3) = &\left(q - m_B\right)\left(q + m_B\right)\left(m_C - p_1 - p_2\right)^2 \\ &\times \left(m_C + p_1 + p_2\right)^2 \left(q - m_A + p_1 + p_2\right) \\ &\times \left(q + m_A + p_1 + p_2\right). \end{aligned} \tag{29}$$

Owing to Wick's rotation, the integration limits have changed to 0 to $\infty$ instead of $-\infty$ to $\infty$. The Wick's rotation as yet does not allow us to integrate the quantities, but makes the expression more convenient to deal with than before.

### 5.2. Feynman Integrals

The concept of Feynman integrals (parameters) is introduced here. It is a useful step towards integrating the quantities obtained in the last section. The Feynman parameters are given in the following form [9]:

$$\frac{1}{ab} \equiv \int_0^1 dx \frac{1}{\left[ax + b\left(1 - x\right)\right]^2}. \tag{30}$$

This identity can be verified by integrating the right hand side. Alternatively, one may also use CAS system (`sympy`) to test the identify as illustrated in the program below:

```
from sympy import symbols,simplify,integrate
a, b, x = symbols('a b x')
integrand = 1/(x*a + (1-x)*b)**2
result=integrate(integrand,(x,0,1))
simplify(result)
```

As we see the expression (23), parameters like a, b are useful in bubble diagrams. However, for triangle or box diagrams, we need parameters such a, b, c, and d. We focus on applying the case for two parameters. We now apply the Feynman identity to the bubble diagram under consideration. For simplicity we only write the terms containing a factor of $q$. Therefore the function to be integrated is:

$$\mathcal{M} \propto -\int q^3 dq \frac{1}{F(q)}, \tag{31}$$

with $F(q)$ given as:

$$\begin{aligned} F(k_3) &= \left(q - m_B\right)\left(q + m_B\right)\left(q + p - m_A\right)\left(q + p + m_A\right) \\ &= \left(q^2 - m_B^2\right)\left((q + p)^2 - m_A^2\right)., \end{aligned} \tag{32}$$

where $p = p_1 + p_2$.

We now identify $a \equiv (q^2 - m_B^2)$ and $b \equiv \left((q + p)^2 - m_A^2\right)$, which implies that:

$$\begin{aligned} &\int_0^1 dx \int d\Omega \int_0^\infty q^3 dq \\ &\times \frac{1}{\left(x\left(-m_B^2 + (p+q)^2\right) + (1-x)\left(-m_B^2 + q^2\right)\right)^2}. \end{aligned} \tag{33}$$

One then needs to collect the terms as per the order of $q$ and expand the equation, and then finally, factorize again. The above steps can be carried out in `sympy` with the following program:

```
from sympy import symbols
from sympy import simplify,expand,factor
from sympy import latex
q, p,x =symbols('q p x')
mA, mB = symbols('m_A m_B')

def feynman_integral(A, B):
    integrand = 1/(A*(1-x) + x*B)**2
    return integrand

integrand = feynman_integral(q**2 - mB**2,
(q+p)**2 - mB**2)
factor(expand(integrand).collect('q'))
```

The output of this program is:

$$\begin{aligned} &\int_0^1 dx \int d\Omega \int_0^\infty q^3 dq \\ &\times \frac{1}{\left(-m_B^2 + p^2 x + 2pqx + q^2\right)^2}. \end{aligned} \tag{34}$$

Now we substitute the following variable: $l = q + xp$ and find the following:

$$\frac{1}{\left(-l^2 + m_B^2 + p^2 x^2 - p^2 x\right)^2}. \tag{35}$$

In the above equation, let us name all terms not containing $l$ as some constant $\Delta$.

$$\frac{1}{\left(l^2 - \Delta\right)^2}, \tag{36}$$

with

$$\Delta = mB^2 + p^2 x^2 - p^2 x. \tag{37}$$

### 5.3. Cut-off Method

One of the methods to carry out the momentum integral is the cut-off method. The motivation of the cut-off procedure is to integrate the internal momentum from 0 to $\Lambda$ instead of infinity and take the limit: $\lim_{\Lambda \rightarrow \infty}$ at the end of the calculation. The main idea is that we do not know a priori if we are fully aware of the dynamics of physics beyond certain energies. Carrying out the integration over this $q$ with the new limit, we find (using the definition of $\Delta$ as above):

$$\mathcal{M} \propto \frac{\Lambda}{-2\Delta^2+2\Delta\Lambda^2} + \frac{\sqrt{\frac{1}{\Delta^3}}\log\left(-\Delta^2\sqrt{\frac{1}{\Delta^3}}\right)}{4} - \frac{\sqrt{\frac{1}{\Delta^3}}\log\left(\Delta^2\sqrt{\frac{1}{\Delta^3}}\right)}{4} - \frac{\sqrt{\frac{1}{\Delta^3}}\log\left(-\Delta^2\sqrt{\frac{1}{\Delta^3}}+\Lambda\right)}{4} + \frac{\sqrt{\frac{1}{\Delta^3}}\log\left(\Delta^2\sqrt{\frac{1}{\Delta^3}}+\Lambda\right)}{4}, \tag{38}$$

and the integration: $\int d\Omega$ in 4-dimension is $4\pi$. We also need to take into account the $1/(2\pi)^4$ factor that arises due to the $dq$ integral. The final result will also take the factors from Eq. (32) that did not contain the variable $q$.

One should then be able to isolate the infinities from finite terms. Actually, what we present here is a simple technique to isolate infinities from finite terms. The infinities themselves need to be addressed by renormalization where in the Lagrangian is required to contain additional 'counter-terms' that keep the infinities 'under the rug'.

The `sympy` code to carry out the steps in this section is as follows:

```
from sympy import Symbol,symbols
from sympy import simplify,expand,factor
from sympy import integrate,latex
q, p,x,l =symbols('q p x l')
mA, mB = symbols('m_A m_B')
lamb = Symbol('\Lambda')
delta = Symbol('\Delta')

def feynman_integral(A, B):
    integrand = 1/(A*(1-x) + x*B)**2
    return integrand

integrand = feynman_integral(q**2 - mB**2,
(q+p)**2 - mB**2)
result=factor(expand(integrand).collect('q'))
result=result.subs({q: l -x*p})
result=result.expand().collect('l').factor()
result=result.subs\
({mB**2 + (p*x)**2 - p**2 *x: delta})
result = integrate(result, (l,0,lamb))
```

In the next section we provide a glimpse of implementing dimensional regularization in `sympy`.

### 5.4. Dimensional Regularization

One of the alternative procedures to isolate the infinities of the integral in Eq. (34) involve the technique of dimensional regularization. The motivation here is to write the integral in the $d$-dimensional abstract space and introduce a parameter $\epsilon$ such that $d = 4-\epsilon$ and then at the end of the calculation take the limit $d \to 4$. In this section, we highlight the important steps involved in dimensional regularization and its implementation in `sympy`. The expression that we want to integrate is given as:

$$\int_0^1 dx \int_0^\infty d^d l \frac{1}{(l^2-\Delta)^{2m}}, \tag{39}$$

where $\Delta$ is given by Eq. (37). Our focus will be to integrate over the variable $l$. For generalisation of the problem we have introduced a factor $m$ in the power of denominator. For this case, one can substitute $m = 1$ at the end of the calculation. The angular integral of $\int d^{d-1}\Omega$, which represents a sphere in $d$ dimension is given as $(2\pi^{d/2})/(\Gamma(d/2))$ and one check this function of some known cases as $d = 1, 2, 3...$ [8].

Apart from the angular factor which can be integrated independently, the part of the integral in Eq. (39) is evaluated and found as:

$$\frac{2^{-d}\pi^{-d}\Delta\Delta^{-2m}\Delta^{\frac{d}{2}-1}\Gamma\left(\frac{d}{2}\right)\Gamma\left(-\frac{d}{2}+2m\right)}{2\Gamma\left(2m\right)}, \tag{40}$$

which in our case $m = 1$, reduces to the following form:

$$\frac{2^{-d}\pi\pi^{-d}\Delta^{\frac{d}{2}-1}\left(d-2\right)}{4\Delta\sin\left(\frac{\pi d}{2}\right)}. \tag{41}$$

The following program can be used to integrate:

```
from sympy import Symbol,symbols
from sympy import integrate,simplify
from sympy import pi,oo
l, d, m = symbols('l d m',positive=True)
delta = Symbol('\Delta',positive=True)
eps = Symbol('\epsilon')
integral = l**(d-1)/(l**2 + delta)**(2*m)
integral = integral/(2*pi)**d
integral = integrate(integral,(l,0,oo))
simplify(integral)
```

Note that when $m = 1$ (which is the case we are dealing with), one gets a factor of $\Gamma(\epsilon/2)$ which is defined as (see *e.g.*, [5]):

$$\Gamma\left(\frac{4-d}{2}\right) = \Gamma\left(\frac{\epsilon}{2}\right) = \frac{2}{\epsilon} - \gamma + \mathcal{O}(\epsilon), \tag{42}$$

where, $\gamma$ is the Euler-Macheroni constant and the limit $d \to 4$ is carried at the end of computation. The divergent terms are therefore collected by the $1/\epsilon$ factor and other terms are finite. The idea then is to renormalize the ABC Lagrangian and renormalization factors are defined such that the infinities are absorbed by the counter diagrams. The last part is not covered in this paper.

## 6. Exercises

- Evaluate the matrix element for $t$-channel and $u$-channel processes at Leading Order.

- Obtain an expression for matrix element for a bubble diagram when the "bubble" is present on an external line.

(See Appendix for Solutions to the Exercises)

## 7. Conclusions

We have presented a semi-automated `Python` Computer Algebra System (`sympy`) framework, which can be used to evaluate the LO matrix element as well as the expressions for NLO matrix elements such as the triangle, bubble and box diagrams in the ABC model. Moreover, we explored some of the techniques such as implementing Feynman integrals, cut-off method and dimensional regularization techniques in this semi-automated framework. Given the open-source python-based ecosystem, our computation will be useful in teaching how theory computation in particle physics can be implemented in python at an early stage.

In the future, we aim to present another paper which will be based on carrying out semi-automated `sympy` based computation for Quantum Electrodynamics. Moreover, a further automation can be introduced by building a framework that, by using techniques of combinatorics and conservation of quantum numbers, is able to predict the possible Feynman diagram for a given process.

## Appendix

## A. Solutions to exercises

- For t-channel process (Fig. 4) the matrix element can be computed as follows:

```
fr = FeynmanRules
ME = fr.external_line()**4
ME = ME * fr.vertex()**2
ME = ME * fr.propagator(q,mC)
ME = ME * fr.delta(p1,-p3,-q)
ME = ME * fr.delta(p2,-p4,q)
ME = integrate(ME, (q,-oo,oo)) * fr.integral_factor()
ME = ME * fr.remove_factor()
ME_t = ME.subs({p3: -p4 +p1 + p2})
print(latex(ME_t))
```

which results in

$$\mathcal{M}_t = \frac{g^2}{-m_C^2 + (-p_2 + p_4)^2}. \tag{A.1}$$

- For u-channel process (Fig. 5) the matrix element can be computed as follows:

```
fr = FeynmanRules
ME = fr.external_line()**4
ME = ME * fr.vertex()**2
ME = ME * fr.propagator(q,mC)
ME = ME * fr.delta(p1,-p4,-q)
ME = ME * fr.delta(p2,-p3,q)
ME = integrate(ME, (q,-oo,oo)) * fr.integral_factor()
ME = ME * fr.remove_factor()
ME_u = ME.subs({p3: -p4 +p1 + p2})
print(latex(ME_u))
```

which results in

$$\mathcal{M}_u = \frac{g^2}{-m_C^2 + (p_1 - p_4)^2}. \tag{A.2}$$

- For the bubble diagram with a bubble in an external arm (Fig. 10):

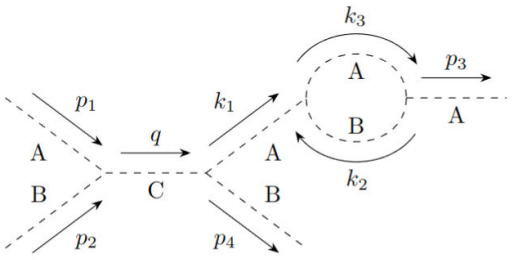


FIGURE 10. Representative Feynman triangle diagram for the scattering process at NLO in the $s$-channel with bubble in an external arm.

The following program can be used:

```
fr = FeynmanRules
ME = fr.external_line()**4
ME = ME * fr.vertex()**4
ME = ME * fr.propagator(q,mC)
ME = ME * fr.propagator(k1,mA)
ME = ME * fr.propagator(k2,mB)
ME = ME * fr.propagator(k3,mC)
ME = ME * fr.delta(p1,p2,-q)
ME = ME * fr.delta(q,-k1,-p4)
ME = ME * fr.delta(k1,k2,-k3)
ME = ME * fr.delta(-k2,k3,p3)
ME = ME *fr.integral_factor()*
ME = integrate(ME, (q,-oo,oo))
ME = integrate(ME, (k1,-oo,oo)
ME = integrate(ME, (k2,-oo,oo)
```

which gives

$$\mathcal{M}=\frac{g^4\delta\left(p_1+p_2+p_3-p_4\right)}{\left(k_3-m_C\right)\left(k_3+m_C\right)\left(m_A-p_3\right)\left(m_A+p_3\right)\left(k_3-m_B+p_3\right)\left(k_3+m_B+p_3\right)\left(m_C-p_3+p_4\right)\left(m_C+p_3-p_4\right)}. \tag{A.3}$$

$i$. The Matrix element presented in this section, has not considered a factor of $1/(2\pi)^4$ which is multiplied after carrying out all integrations. We have therefore used the proportionality symbol.


1. R. Mertig, M. Böhm, and A. Denner, Feyn Calc - Computeralgebraic calculation of Feynman amplitudes, *Computer Physics Communications* **64** (1991) 345, https://doi.org/10.1016/0010-4655(91)90130-D.
2. G. Aad *et al.*, Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC, *Phys. Lett. B* **716** (2012) 1, https://doi.org/10.1016/j.physletb.2012.08.020.
3. S. Chatrchyan *et al.*, Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC, *Phys. Lett. B* **716** (2012) 30, https://doi.org/10.1016/j.physletb.2012.08.021.
4. D. J. Griffiths, Introduction to elementary particles; 2nd rev. version, Physics textbook (Wiley, New York, NY, 2008), https://cds.cern.ch/record/111880.
5. M. E. Peskin and D. V. Schroeder, An Introduction to quantum field theory (Addison-Wesley, Reading, USA, 1995).
6. G. 't Hooft and M. Veltman, Regularization and renormalization of gauge fields, *Nuclear Physics B* **44** (1972) 189, https://doi.org/10.1016/0550-3213(72)90279-9.
7. A. Meurer, et al., SymPy: symbolic computing in Python, PeerJ *Computer Science* **3** (2017) e103, https://doi.org/10.7717/peerj-cs.103.
8. 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.
9. 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.
10. A. Desai, Monte Carlo Event Generator for the ABC Model (pre-print) (2023), https://doi.org/10.5281/zenodo.8181098.