Package-X: A Mathematica package for the analytic calculation of one-loop integrals

Hiren H. Patel

1 Particle and Astro-Particle Physics Division
Max-Planck Institute for Nuclear Physics (MPIK)
Saupfercheckweg 1, 69117 Heidelberg, Germany

Package-X, a Mathematica package for the analytic computation of one-loop integrals dimensionally regulated near 4 spacetime dimensions is described. Package-X computes arbitrarily high rank tensor integrals with up to three propagators, and gives compact expressions of UV divergent, IR divergent, and finite parts for any kinematic configuration involving real-valued external invariants and internal masses. Output expressions can be readily evaluated numerically and manipulated symbolically with built-in Mathematica functions. Emphasis is on evaluation speed, on readability of results, and especially on user-friendliness. Also included is a routine to compute traces of products of Dirac matrices, and a collection of projectors to facilitate the computation of fermion form factors at one-loop. The package is intended to be used both as a research tool and as an educational tool.

CONTENTS

I. Introduction 1
II. Structure and Design of package 2
III. LoopIntegrate: Covariant tensor decomposition 3
IV. LoopRefine: Reduction to elementary functions 3
A. Reduction of A and B functions 4
B. Reduction of auxiliary \( b^k \) functions 4
C. Reduction of C functions 4
Case 1: \( \text{det} Z \neq 0 \) 5
Case 2: Ellis-Zanderighi triangle 6
Case 3: \( \text{det} Z = 0 \) but \( \tilde{X}_0j \neq 0 \) 6
Case 4: vanishing \( \text{det} Z \) and \( \tilde{X}_0j \) 7
Case 5: All elements of \( Z \) vanishing 7
Case 6: All elements of \( Z \) and \( f_k \) vanishing 7
V. The scalar \( C_0 \) function: Analytic expressions and numerical implementation 8
VI. Handling the \(+i\varepsilon\) prescription and simplifying logarithms 9
VII. Spur: Computation of traces of Dirac matrices 10
VIII. Projector: Projection onto fermion form factors 10
IX. Conclusions and further development 11
Acknowledgments 11

A. Conventions 11
B. Feynman parameter integral representations of Passarino-Veltman coefficient functions 12
C. Derivation of reduction formulae for C functions Case 2 13
D. Derivation of reduction formulae for C functions Case 4 14
E. Coefficient B functions with \( r = -1 \) 15
References 16

I. INTRODUCTION

Many packages are available to assist with the evaluation of one-loop integrals that appear in higher order calculations of perturbative quantum field theory. The most widely used ones are the Mathematica packages FeynCalc[1], FORMCalc[2] and the Fortran program Golem95[3]. These packages compute one-loop integrals using the Passarino-Veltman reduction algorithm[4], and form just a small part of a much larger collection of routines designed to streamline the numerical computation of a differential cross section. As such, they do substantially more than to simply compute one-loop integrals.

Nevertheless FeynCalc falls short in that it gives results of one-loop computations in terms of basis scalar functions which cannot be evaluated on their own. Instead, it is up to the user to supply their analytical forms from an external source, or to link them to yet another package (such as LOOPTools[2]).

Moreover, one-loop integrals have many more applications than to calculate cross sections and decay rates. Examples are the computation of ultraviolet counterterms, pole positions, residues, Peskin-Takeuchi oblique param-
eters, electromagnetic moments, etc. Many of these applications require the calculation of Feynman integrals at singular kinematic points such as at physical thresholds or zero external momenta. Since the Passarino-Veltman reduction algorithm typically breaks down at these points, it is nearly impossible to obtain results with FeynCalc or FormCalc (GOLEM95 can give numerical results). But, it is also at these points where compact analytic expressions exist.

Although smaller-scale packages are available that are designed around a particular application (such as LOOL [5] and ANT [6]), there is no general-purpose software that gives analytic results to one-loop integrals for all kinematic configurations. In this regard, Package-X serves to fill this gap.

Package-X calculates dimensionally regulated \((d = 4 - 2\epsilon)\) rank-\(P\) one-loop tensor integrals of the form

\[
T^{\mu_1 \cdots \mu_P}(p_1, \ldots, p_N; m_0, m_1, \ldots, m_N) = \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{k^{\mu_1} \cdots k^{\mu_P}}{[k^2 - m_0^2 + i\epsilon][(k+p_1)^2 - m_1^2 + i\epsilon] \cdots [(k+p_N)^2 - m_N^2 + i\epsilon]},
\]

with up to \(N = 3\) denominator factors, and finds compact analytic expressions for arbitrary configurations of external momenta \(p_i\) and real-valued internal masses \(m_i\). The functional paradigm of the Wolfram Language together with the supplementary trace-taking routines included in Package-X allows one to compute an entire one-loop diagram at once. All output is ready for numerical evaluation and symbolic manipulation with Mathematica’s internal functions.

This article details the technical aspects of Package-X, and assumes familiarity in the use of the package. The application files are found at the Hepforge webpage \(\text{http://packagex.hepforge.org}\), where the software will be maintained and periodically updated. Included among the package files is a tutorial that provides an introduction, and a complete set of documentation files that becomes embedded with the Wolfram Documentation Center upon installation and provides details and examples of all functions defined in Package-X.

II. STRUCTURE AND DESIGN OF PACKAGE

The subroutines in this package belong to one of three Mathematica contexts organized as in Fig. 1. The module IndexAlg\(^{\text{®}}\) contains the rudimentary tensor-algebraic routines and serves as the backbone of Package-X. OneLoop\(^{\text{®}}\) contains the algorithms and look-up tables for the computation of one-loop integrals, and Spur\(^{\text{®}}\) includes the algorithms to perform traces over products of Dirac matrices and contains a catalog of fermion form factor projectors.

The basic Package-X workflow for the computation of a one-loop integral consists of three steps:

1. Call LoopIntegrate to carry out the covariant tensor decomposition (section [III]).
2. Apply on-shell conditions and other kinematic relations with Mathematica’s built-in functions ReplaceAll (/.) and Rules (\(\rightarrow\)).
3. Call LoopRefine to convert coefficient functions into explicit expressions (section [VI]).

The reasoning behind the three-step design is as follows: kinematic configurations of external invariants \(p_i, p_j\) and internal masses \(m_i\) relevant to many physical applications occur at singular points of one-loop integrals, such that if they were applied \(\text{after}\) obtaining the general expressions, errors like \(0/0\) or \(0 \times \ln(0)\) would inevitably occur. To avoid such errors and to facilitate the generation of compact results, LoopRefine uses algorithms depending critically on the kinematic configuration supplied by the user \textit{beforehand}.

Two other supplementary functions are provided to streamline computations involving fermions:

- Spur (German for ‘trace’) computes traces of Dirac matrices that may appear in the numerators of one-loop integrals (section [VII]).
- Projector is used to project fermion self-energy and vertex form factors out of the loop integrals (section [VIII]).

The algorithms used by these functions are detailed in the aforementioned sections below.
III. LOOPINTEGRATE: COVARIANT TENSOR DECOMPOSITION

The evaluation of an integral is initiated with LoopIntegrate, which carries out its covariant tensor decomposition in terms of scalar coefficient functions. For example (omitting the $+i\varepsilon$),

\[
\text{LoopIntegrate}[k_\mu k_\nu k_\rho k_\lambda p_1 m_0 m_1] : \quad (\frac{i}{16\pi^2})^{-1} \mu^{2\epsilon} \int \frac{dk^4 k}{(2\pi)^4} \frac{k^\mu k^\nu (k,k)^5}{[k^2-m_0^2][(k+p_1)^2-m_1^2]} \rightarrow \{[p_1]^\mu [g]\} B_{001} + \{[p_1]^3\} B_{111}, \tag{2}
\]

\[
\text{LoopIntegrate}[k_\mu k_\nu k_\rho p_1 p_2 m_0 m_1 m_2] : \quad (\frac{i}{16\pi^2})^{-1} \mu^{2\epsilon} \int \frac{dk^4 k}{(2\pi)^4} \frac{k^\mu k^\nu}{[k^2-m_0^2][(k+p_1)^2-m_1^2][(k+p_2)^2-m_2^2]} \rightarrow \{[g]\} (\mu^\mu C_{00} + \{[p_1]^2\}^\mu C_{11} + \{[p_1][p_2]\}^\mu C_{12} + \{[p_2]^2\}^\mu C_{22}) \tag{3}
\]

Here, $B_{001}$, $B_{111}$, $C_{00}$ etc. are coefficient functions that depend only on Lorentz invariants, $p_i$, $p_j$ and $m_i$. Note that as indicated in the left hand sides, an overall constant $(\frac{i}{16\pi^2})$ is factored out of the natural integration measure $\mu^{2\epsilon} \frac{dk^4 k}{(2\pi)^4}$ to simplify the output. Each coefficient function multiplies a totally symmetric tensor, denoted $\{\ldots\}^\mu_{\ldots}$ in the notation of [5], containing products of external momentum four-vectors $p_i^\mu$ and the metric tensor $g^{\mu\nu}$. These tensors are generated by a Package-X internal function (inside IndexAlgl), which utilizes Mathematica’s built-in function Permutations. The time to generate the corresponding symmetric tensors grows factorially with the rank of tensor integrals.

For integrals with high powers of contracted loop momenta, such as

\[
\mu^{2\epsilon} \int \frac{dk^4 k}{(2\pi)^4} \frac{k^\alpha k^\beta (k,k)^5}{[k^2-m_0^2][(k+p_1)^2-m_1^2]}, \tag{4}
\]

it is necessary to obtain explicit expressions of self-contracted symmetrized high-rank tensors like $\{[p_1]^3\}^\alpha_\beta^\mu_\nu g_{r\sigma\rho\nu}$. It would be wasteful to first generate the totally symmetric high-rank tensors, only to subsequently contract indices down to lower-rank symmetric tensors. Instead, the contraction formulae

\[
(p_k)_{\mu_1} \{[p_1]^n_1 \ldots [p_N]^n_N [g]^r\}^{\mu_1_{\ldots \mu}_P} = \sum_{\ell=1}^N p_k \cdot p_{\ell} \{[p_{\ell}] [p_1]^n_1 \ldots [p_N]^n_N [g]^r\}^{\mu_2_{\ldots \mu}_P} + (n_k + 1) \{[p_k] [p_1]^n_1 \ldots [p_N]^n_N [g]^{r-1}\}^{\mu_2_{\ldots \mu}_P} \tag{5}
\]

\[
g_{\mu_1\mu_2} \{[p_1]^n_1 \ldots [p_N]^n_N [g]^r\}^{\mu_3_{\ldots \mu}_P} = \sum_{i,j} p_i \cdot p_j \{[p_i] [p_j] [p_1]^n_1 \ldots [p_N]^n_N [g]^r\}^{\mu_3_{\ldots \mu}_P} + \delta_r (d+P-2+\sum k) \{[p_1]^n_1 \ldots [p_N]^n_N [g]^{r-1}\}^{\mu_3_{\ldots \mu}_P}, \tag{6}
\]

are employed to carry out the self-contractions symbolically before converting any remaining symmetric tensors with free indices into explicit forms in terms of $p_i^\mu$ and $g^{\mu\nu}$. The time to construct self-contracted tensors in this way is reduced to follow a power law.

IV. LOOPREFINE: REDUCTION TO ELEMENTARY FUNCTIONS

Once the covariant decomposition is made, and any on-shell or kinematic conditions are applied, the final step is to feed the results into LoopRefine, which replaces the coefficient functions with explicit expressions. The basic algorithm followed by LoopRefine is as follows:

Step 1: For each coefficient function ($pvA$, $pvB$, $pvB$, or $pvC$) encountered by LoopRefine, symbols for internal masses are recorded (for Step 4), and the appropriate reduction routine (see corresponding subsections below) is called.

Step 2: The reduction of $C$ functions for more general kinematic configurations end with the scalar function $C_0$. If the $C_0$ function is IR-divergent or has an explicit form that is sufficiently compact (as controlled by the option ExplicitCO), the explicit form is substituted.

Step 3: All instances of the spacetime dimension $d$ is replaced by $4 - 2\epsilon$, and Mathematica’s built-in function Series is called to keep the leading terms in the $\epsilon$ expansion. UV-divergences appear as $1/\epsilon$ poles, and IR-divergences appear as $1/\epsilon$ and/or $1/\epsilon^2$ poles.

Step 4: Combine and simplify logarithms, organize the expression by the logarithms, and group the ‘t Hooft parameter $\mu^2$-dependent logarithm with the $1/\epsilon$ pole in the expression (see section V).

In the following subsections, the algorithms and accompanying formulae used by LoopRefine to reduce the
coefficient functions are summarized. It should be noted that nearly all algorithms are drawn from the 2005 paper by Denner and Dittmaier \[7\], and will be referenced henceforth as [DD]. The only formulae not taken directly from their paper are those for the auxiliary $b^k$ functions in Section IVB (which is only a slight modification of the reduction formulae for $B$ functions), and those of two additional algorithms for the reduction of $C$ functions in special kinematic configurations (Cases 2 and 4 in section IVC).

### A. Reduction of $A$ and $B$ functions

The Passarino-Veltman coefficient $A$ functions are simple enough to be obtained by direct integration (eqn 3.4 of [DD]):

$$A_{0...0}(m_0) = \frac{(m_0^2)^{r+1}}{2(r+1)!} \left( \frac{1}{\epsilon} + \ln \left( \frac{\mu^2}{m_0^2} \right) + H_{r+1} \right), \quad (7)$$

where $1/\epsilon = 1/\gamma_E + \ln(4\pi)$, and $H_n$ is the $n^{th}$ harmonic number.

The $B_{0...01...1}$ functions, with at least one pair of 00 indices are obtained iteratively in terms of those with fewer number of 00 indices using (eqn 4.5 of [DD]):

$$B_{0...01...1}(p^2; m_0, m_1) = \frac{-1}{2(n+1)} \left[ (-1)^{n+1} A_{0...0}(m_1) \right] + (p^2 - m_1^2 + m_0^2)B_{0...01...1}(p^2; m_0, m_1) + 2p^2 B_{0...01...1}(p^2; m_0, m_1), \quad r \geq 1 \quad (8)$$

Then the $B_{1...1}$ integrals (with no 00 index pairs) are obtained by explicit integration over the single Feynman parameter in \[12\]. Results are given in (eqn 4.8 of [DD]), but the form that tends to generate most compact expressions is

$$B_{1...1}(p^2; m_0, m_1) = \frac{(-1)^n}{n+1} \left[ \frac{1}{\epsilon} + \ln \left( \frac{\mu^2}{m_1^2} \right) + \sum_{k=0}^{\frac{n+1}{2}} \frac{2}{n+1} \sum_{j=0}^{\frac{n-k-1}{2}} \left( \frac{p^2 + m_0^2 - m_1^2}{2p^2} \right)^{n-k-2j} \left( \frac{\lambda(p^2, m_0^2, m_1^2)}{4(p^2)^2} \right)^j \right. \left. \ln \left( \frac{m_0^2}{m_1^2} \right) \right] + \sum_{k=0}^{\frac{n+1}{2}} \frac{2}{n+1} \left( \frac{p^2 + m_0^2 - m_1^2}{2p^2} \right)^{n-k-2j} \left( \frac{\lambda(p^2, m_0^2, m_1^2)}{4(p^2)^2} \right)^j \ln \left( \frac{m_0^2}{m_1^2} \right) \right] \quad (9)$$

Here $\lambda(a, b, c) = a^2 + b^2 + c^2$ is the Källén function, implemented as $\text{Kallen\lambda}[a, b, c]$, and $\Lambda(p^2; m_0, m_1)$ is the abbreviation

$$\Lambda(p^2; m_0, m_1) = \sqrt{\lambda(p^2, m_0^2, m_1^2)} \ln \left( \frac{2m_0 m_1}{-p^2 + m_0^2 + m_1^2 - \sqrt{(p^2, m_0^2, m_1^2)}} + i\epsilon \right), \quad (10)$$

implemented as $\text{Disc\B}[s, m_0, m_1]$. In order to access $B_{1...1}(p^2; m_0, m_1)$ at its singular points, a limiting procedure would need to be made at runtime in order to avoid errors such as 0/0 or $0 \times \ln(0)$. While Mathematica’s function Limit can eventually generate an expression, computation time is long, and output expressions are always unwieldy. Instead, a catalog of explicit expressions (also obtained by direct integration) of $B_{1...1}$ at all its singular points (see Table I) is included in the source code. They may be accessed directly within Package-X using $\text{LoopRefine}[p\text{V}[0, n, s, m_0, m_1]]$.

### B. Reduction of auxiliary $b^k$ functions

In covariant gauges, the propagator for massless vector fields

$$i\hat{D}^{\mu\nu}(k) = -\frac{i}{k^2} \left[ g^{\mu\nu} - (1 - \xi) \frac{k^{\mu} k^{\nu}}{k^2} \right], \quad (11)$$

contains a gauge term that leads to an additional factor in the denominator of one-loop integrals. Package-X can handle such propagators inside bubble integrals, with the coefficient functions given by the auxiliary Passarino-Veltman $b^k$ functions \[8\]. For example,

$$\left( \frac{4}{16\pi^2} \right)^{-1} \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{k^\mu k^\nu k^\rho}{[k^2]^2((k + p)^2 - m^2)} = \{ \{ p | g \} \}^{\mu\nu} \hat{b}^{001}_1 + \{ \{ p | g \} \}^{\mu\nu p} \hat{b}^{111}_1.$$ 

The reduction formulae for these functions essentially mirror those for the standard $B$ functions. Auxiliary $b^{0...01...1}$ functions with at least one pair of 00 indices are iteratively determined in terms of functions with fewer
00 index pairs using
\[
b_{0\ldots011}^\xi(\xi^2; m) = -\frac{1}{2(n+1)} B_{0\ldots0}(\xi^2; 0, m) \\
+ (\xi^2 - m^2) b_{0\ldots011}^\xi(\xi^2; m) \\
+ 2\xi^2 b_{0\ldots011}^\xi(\xi^2; m), \quad r \geq 1, \quad (12)
\]
and the \( b_{1\ldots1}^\xi \) functions with no 00 index pairs are obtained by direct integration over the single Feynman parameter in [DD]. The integral is finite if \( n \geq 1 \), with the result
\[
b_{1\ldots1}^\xi(\xi^2; m) = \\
\frac{(-1)^{n-1}}{p^2} \left[ -\frac{1}{n} \sum_{k=1}^{n-1} \frac{1}{n-k} \frac{m^2}{p^2 - m^2} \left( \frac{p^2 - m^2}{p^2} \right)^k \\
+ \frac{m^2}{p^2 - m^2} \left( \frac{p^2 - m^2}{p^2} \right)^n \ln \left( \frac{m^2}{p^2 - m^2} + i\varepsilon \right) \right].
\]
If \( n = 0 \) (a case that is not met in practice since the gauge part of the spin-1 propagator guarantees two powers of momenta in the numerator), the auxiliary \( b_{1\ldots1}^\xi \) function is IR-divergent.

Explicit expressions at the various singular points of \( b_{1\ldots1}^\xi \) (see Table I) are included in the Package-X source code.

C. Reduction of C functions

The reduction of coefficient \( C \) functions is significantly complicated by its numerous singular kinematic points. Although the standard Passarino-Veltman reduction algorithm is applicable at most points (Case 1 below), different formulae are needed to handle the various singular cases (Cases 2 – 6). LoopRefine identifies the nature of the kinematic configuration and applies the appropriate reduction method.

Cases 1, 3, 5 and 6 are taken from [DD]. Note that since the emphasis of [DD] is on numerical stability and not on generating analytic expressions, the algorithms presented there do not automatically give compact expressions. The algorithm under Case 2 is new, and while technically it is covered by Case 1, it leads to more compact expressions. Furthermore, an algorithm to handle the reduction at physical thresholds is not completely covered by [DD] because an expansion around that point is not known. This gap is filled by the formulae under Case 4.

The arguments of the coefficient \( C \) functions are ordered differently in Package-X as compared to those used by other authors. See Appendix A for details.

In the reduction formulae below, the following kinematic abbreviations are used (which differ slightly from [DD] by numeric factors):
\[
f_j = p_j^2 - m_j^2 + m_0^2, \quad j = \{1, 2\}
\]
\[
Z = \left( \begin{array}{cc} p_1^2 & p_1.p_2 \\
     p_2.p_1 & p_2^2 \end{array} \right) \quad \text{(Gramian matrix)}
\]
\[
q^2 = p_1^2 + p_2^2 - 2p_1.p_2 \\
\det Z = \frac{1}{4} \lambda(q^2, p_1^2, p_2^2)
\]
\[
\tilde{Z} = \left( \begin{array}{cc} p_2^2 & -p_1.p_2 \\
     -p_1.p_2 & p_1^2 \end{array} \right) \quad \text{(cofactor matrix)}
\]
\[
\tilde{X}_{0j} = \left( \begin{array}{cc} p_2^2 f_1 - p_1.p_2 f_2 \\
     -p_1.p_2 f_1 + p_1^2 f_2 \end{array} \right) \quad j = \{1, 2\}
\]
Furthermore, hatted indices on coefficient functions (e.g. \( B_{0\ldots011}^\xi \)) indicate the removal of those indices. Coefficient \( B \) functions derived by canceling denominators from three-point integrals are abbreviated by
\[
B_{..}(\tilde{D}_1) = B_{..}(p_2^2; m_0, m_2) \quad (14)
\]
\[
B_{..}(\tilde{D}_2) = B_{..}(p_1^2; m_0, m_1). \quad (15)
\]
If the denominator \((k^2 - m_0^2)^{-1}\) independent of an external momentum vector is cancelled, a shifted form of the \( B \) function is used:
\[
B_{0\ldots0112\ldots2}(\tilde{D}_0) = \\
\frac{(-1)^{n_1}}{p_1^2} \sum_{j=0}^{n_1} \binom{n_1}{j} B_{0\ldots011}(q^2; m_1, m_2). \quad (16)
\]
Whenever \( n_1 > n_2 \) the invariance property
\[
B_{0\ldots0112\ldots2}(\tilde{D}_0) = B_{0\ldots0112\ldots2}(\tilde{D}_0)|_{m_1 \leftrightarrow m_2} \quad (17)
\]
is used to keep the number of terms in the sum to a minimum. Cases 2 and 4 require expressions for the \( B \) functions with the number of 00 index pairs continued to \( r = -1 \). Details of this function are found in Appendix E.

Finally, formulae for Cases 1, 3 and 5 below contain explicit dependence on spacetime dimension \( d = 4 - 2\epsilon \) appearing in denominators of certain prefactors. In the course of reduction, the \( O(\epsilon) \) part multiplying any lower coefficient functions combines with their UV 1/\( \epsilon \) pole.

\[\text{A. Denner, private correspondence}\]

\[\text{For the argument that they are only of UV origin (and not IR), see the argument in Sec. 5.8 of [DD]}\]
and gives rise to finite polynomials in kinematic variables. Although this can be automatically handled by Series at Step 3, the reduction algorithm performs much faster if these polynomials are explicitly supplied. They are obtained by integration over the Feynman parameters as described at the end of Appendix [B].

\[ \text{Case 1: } \det Z \neq 0 \]

At non-singular kinematic configurations with \( \det Z \neq 0 \), the original Passarino-Veltman reduction formula is used (eqns 5.10, 5.11 of [DD]):

\[
\begin{align*}
& C_{0\ldots 0\underbrace{1_{n_1} 2_{n_2}}_{2r}} = \frac{1}{2\det Z} \sum_{k=1}^{2r} \tilde{Z}_{jk} \left[ \delta_{nk} \delta_{jk} B_{0\ldots 0\underbrace{1_{n_1} 2_{n_2}}_{2r-1}} (\tilde{D}_k) - B_{0\ldots 0\underbrace{1_{n_1} 2_{n_2}}_{2r}} (\tilde{D}_0) \
& \quad - f_k C_{0\ldots 0\underbrace{1_{n_1-1} 2_{n_2}}_{2r-2}} (\tilde{D}_0) - 2(n_k - \delta_{jk}) C_{0\ldots 0\underbrace{1_{n_1-1} 2_{n_2}}_{2r-2}} \right], \quad n_1 \geq 1 \quad (18)
\end{align*}
\]

where \( \tilde{k} = \begin{cases} 1, & k = 2 \\ 2, & k = 1 \end{cases} \). In the first equation, \( j = 1 \) is taken, although the choice \( j = 2 \) would give equivalent results. If \( n_1 = 0 \) with \( n_2 > 0 \), then the relation [B5] is used and the first equation is applied.

\[ \text{Case 2: Ellis-Zanderighi triangle } 6 \]

Coefficient \( C \) functions for which arguments are \((m_0^2, s, m_s^2; m_2, 0, m_0)\)—or an equivalent permutation thereof—are already covered by Case 1. However, final expressions obtained from it tend not to give the most compact formulae for this kinematic configuration. More compact formulae are obtained by directly integrating over the Feynman parameters in [B4]; see Appendix [C] for derivation. It is of note that the corresponding scalar function \( \tilde{C}_0 \) is the IR-divergent three-point function, ‘triangle 6’, as classified by Ellis and Zanderighi [9].

\[
\begin{align*}
& C_{0\ldots 0\underbrace{1_{n_1} 2_{n_2}}_{2r}} (m_0^2, s, m_s^2; m_2, 0, m_0) = \\
& \quad \frac{(-1)^{n_1} n_1! (n_2 + 2r - 1)!}{2} \frac{(1 + 2\epsilon(H_{n_1 + n_2 + 2r} - H_{n_2 + 2r - 1}))}{(n_1 + n_2 + 2r)!} B_{0\ldots 0\underbrace{1_{n_1} 2_{n_2}}_{2r-2}} (s; m_0, m_2), \quad n_2 \neq 0 \text{ or } r \neq 0 \quad (19)
\end{align*}
\]

where \( H^{(r)}_n \) is the \( n \)th harmonic number of order \( r \). If the arguments take the form \((s, m_0^2, m_s^2; 0, m_2, m_0)\), then the identity [B5] is applied, and the equations above are valid.

A different formula is needed if the off-shell momentum \( s \) is in the third position:

\[
C_{0\ldots 0\underbrace{1_{n_1} 2_{n_2}}_{2r}} (m_0^2, m_s^2, s; m_0, m_2, 0) = \frac{(-1)^{n_2}}{2} \frac{1}{n_1 + n_2 + 2r} \sum_{k=0}^{n_2} \left( \begin{array}{c} n_2 \\ k \end{array} \right) \frac{2\epsilon}{n_1 + n_2 + 2r} B_{0\ldots 0\underbrace{1_{n_1+k} 2_{n_2}}_{2r-2}} (s; m_0, m_2) \quad (20)
\]

To apply Eqs. [19] and [20] above, explicit forms of the scalar \( B \) function with the number of 00 indices taken to \( r = -2 \) is occasionally needed. These functions are discussed in Appendix [E].
Case 3: det \(Z = 0\) but \(\tilde{X}_{0j} \neq 0\)

With \(\det Z = 0\), the primary reduction formulae are rearranged to give: (eqns 5.38 and 5.40 of [DD])

\[
\begin{align*}
\left\{ \begin{array}{l}
C_{0\ldots 01\ldots 12\ldots 2} = \frac{1}{X_{0j}} \sum_{k=1}^{2} \tilde{Z}_{jk} \left( \delta_{n_10} B_{0\ldots 01\ldots 1} (\hat{D}_k) - B_{0\ldots 01\ldots 12\ldots 2} (\hat{D}_0) - 2n_k C_{0000\ldots 01\ldots 12\ldots 2} \right), \\
\sum_{n_1, n_2 \geq 1} \end{array} \right. \\
C_{0\ldots 0} = \frac{1}{d+2r-3} \left( B_{0\ldots 0} (\hat{D}_0) - m_n^2 C_{0\ldots 0} \right) + \frac{1}{2(d+2r-3)} \sum_{n, m=1}^{2} \left( \delta_{nm} \delta_{nl} - \delta_{kl} \delta_{nm} \right) \\
\times \left\{ \begin{array}{l}
\sum_{j=1}^{2} \left( (1-\delta_{mj}) B_{0\ldots 01} (\hat{D}_m) - B_{0\ldots 01} (\hat{D}_0) \right) + \frac{1}{2} f_m \left[ -B_{0\ldots 01} (\hat{D}_n) + B_{0\ldots 01} (\hat{D}_0) + f_n C_{0\ldots 0} \right] \right\}
\end{align*}
\]

(21)

The value of \(j\) chosen (1 or 2) is the one for which the corresponding \(\tilde{X}_{0j}\) is non-vanishing. If both elements are vanishing, then Case 4 is applied.

Case 4: vanishing det \(Z\) and \(\tilde{X}_{0j}\)

When the physical threshold coincides with the boundary of the physical region, a new set of reduction formulae are used. They are derived in Appendix D. If \(p_2^2 \neq 0\),

\[
\begin{align*}
C_{0\ldots 01\ldots 12\ldots 2} &= \frac{(-1)^{n_1+n_2}}{2} \sum_{j=0}^{n_2} \left( \frac{n_2}{j} \right) \alpha^{n_2-j} \left\{ \frac{n_1!(n_2-j)!}{(n_1+n_2-j+1)!} \sum_{k=0}^{j} \left( \frac{j}{k} \right) (-\alpha)^{j-k} (-1)^k B_{0\ldots 01\ldots 1} (\hat{D}_1) \right. \\
&+ \left\{ \sum_{k=0}^{n_1} \frac{1}{k!} \left( \frac{(-1)^{n_1-k}}{n_2-j+k+1} \right) \left( 1-\alpha \right)^{j+1} (-1)^{n_2+k} B_{0\ldots 01\ldots 1} (\hat{D}_0) + (-\alpha)^{j+1} (-1)^{n_2+k+1} B_{0\ldots 01\ldots 1} (\hat{D}_2) \right\} 
\end{align*}
\]

(22)

where \(\alpha = -q^2 + p_1^2 + p_2^2/(2p_2^2)\), and \((n)_k\) is the Pochhammer symbol.

If \(p_2^2 = 0\), then \(\det Z = 0\) implies \(q^2 = p_1^2\), and the formula

\[
\begin{align*}
C_{0\ldots 01\ldots 12\ldots 2} &= \frac{(-1)^{n_1+1}}{2(n_2+1)} \sum_{k=0}^{n_1} \left( \frac{n_1}{k} \right) B_{0\ldots 01\ldots 1} (\hat{D}_2)
\end{align*}
\]

(23)

is used. When \(r = 0\) in either (22) or (23), the \(B\) functions continued to \(r = -1\) are needed (see Appendix E).

Case 5: All elements of \(Z\) vanishing

If all external invariants are vanishing \(p_1^2 = p_2^2 = q^2 = 0\), then the following are applied (eqns 5.62 and 5.63 of [DD]):

\[
\begin{align*}
\left\{ \begin{array}{l}
\frac{C_{0\ldots 01\ldots 12\ldots 2}}{2r} = \frac{1}{d+2(n_1+n_2+r-1)} \left[ B_{0\ldots 01\ldots 12\ldots 2} (\hat{D}_0) + m_n^2 C_{0\ldots 01\ldots 12\ldots 2} \right], \\
\sum_{n_1, n_2 \geq 1} \end{array} \right. \\
\frac{C_{1\ldots 12\ldots 2}}{n_2} = \frac{1}{f_k} \left[ -2n_k C_{k001\ldots 12\ldots 2} + \delta_{n_10} B_{1\ldots 12\ldots 2} (\hat{D}_k) - B_{1\ldots 12\ldots 2} (\hat{D}_0) \right]
\end{align*}
\]

(24)

In the second equation, the index \(k\) is chosen such that \(f_k\) is non-vanishing.
TABLE I. Special kinematic cases of the Passarino-Veltman coefficient functions $B$, $b^\ell$ and $C$ for which explicit expressions are included in the source file OneLoop.m. Further information for these function is found in the indicated sections.

| Special kinematic cases of the Passarino-Veltman coefficient functions | B-functions | $b^\ell$-functions | C-functions |
|--------------------------------------------------------------|-------------|------------------|-------------|
| $B_{1,1}(0; 0, 0)$                                           | $B_{1,1}(p^2; 0, 0)$ | $b^\ell_{1,1}(0, m)$ | $C_0(0, 0, 0, 0, 0, 0)$ |
| $B_{1,1}(m_0^2; m_0, m_0)$                                  | $B_{1,1}(0; m_0, m_0)$ | $b^\ell_{1,1}(m_0^2, m_0)$ | $C_0(0, m_0^2; m_0, 0, 0)$ |
| $B_{1,1}(0; 0, m_1)$                                        | $B_{1,1}(m_1^2; 0, m_1)$ | $b^\ell_{1,1}(0, m_1)$ | $C_0(0, 0, m_1, 0, 0, 0)$ |
| $B_{1,1}(0; m_0, m_0)$                                      | $B_{1,1}(0; m_0, m_0)$ | $b^\ell_{1,1}(m_0^2, m_0)$ | $C_0(0, m_0^2; m_0, 0, 0)$ |

$B$-functions with $r = -1$ — Appendix E

| $B_{1,1}(0; 0, 0)$                                           | $B_{1,1}(p^2; 0, 0)$ | $B_{1,1}(m_0^2; m_0, 0)$ | $B_{1,1}(0; m_0, m_1)$ |
| $B_{1,1}(0; 0, m_1)$                                        | $B_{1,1}(m_1^2; 0, m_1)$ | $B_{1,1}(p^2; 0, m_1)$ |
| $B_{1,1}(0; m_0, m_0)$                                      | $B_{1,1}(0; m_0, m_0)$ | $B_{1,1}(m_0^2; m_0, 0)$ | $B_{1,1}((m_0 + m_1)^2; m_0, m_1)$ |

$C_0(0, 0, 0, 0, 0, 0)$ for $k = 1$, $B_{1,1}(0; 0, 0)$ for $k = 2$.

Case 6: All elements of $Z$ and $f_k$ vanishing

Finally, if also the $f_k$’s are vanishing, the following formulae are used (eqns 5.71 and 5.72 of [DD]):

$$
\begin{align*}
C_{0,0,1,1,1,2,2} &= \frac{-1}{2(n_k+1)} B_{k\rightarrow\infty} \left( \hat{D}_0 \right) , \quad r \geq 1 \\
C_{1,1,2,2} &= \frac{1}{m_0^2} \left[ (d+2n_1+2n_2)C_{00,1,1,1,2,2} - B_{1,1,2,2} \right] \\
\end{align*}
$$

Equivalent results are obtained for $k = 1$ or 2 in the first equation. The choice $k = 1$ is used in Package-X.

V. THE SCALAR $C_0$ FUNCTION: ANALYTIC EXPRESSIONS AND NUMERICAL IMPLEMENTATION

Many algorithms for the reduction of coefficient $C_0$-functions listed in the previous section end with the UV-finite scalar function $C_0(p_1^2, p_2^2, q^2, m_1, m_0)$. To complete the computation of the one loop integral and to make the final output useable, the scalar function must be replaced. For this purpose, a complete catalog of analytic expressions for $C_0$ at numerous kinematic cases—each one obtained by direct integration—is included in the source file (see Table I). Many such expressions are scattered throughout the literature. The general formula with non-zero kinematic variables appears in [10]. All IR-divergent three-point formulae are given in [11], and some expressions valid for vanishing momenta are found in [12]. Others still, appear in unpublished notes [12].

Although a complete catalog of analytic expressions of $C_0$ is available, not all cases are automatically sub-
masses and external momenta are as follows: succes- sively slow numerical evaluations. When numerics are required, a brute-force evaluation of the dilogarithm function is computationally very expensive. The main features of the code for the rapid numerical evaluation of the three-point scalar function for real masses and external momenta are as follows:

- The imaginary part of $C_0$ in the physical region (defined by $\lambda(q^2, p^2_1, p^2_2) > 0$) is obtained by applying Cutkosky’s rule, and with a straightforward continuation into the unphysical region (defined by $\lambda(q^2, p^2_1, p^2_2) < 0$). Its computation requires the evaluation of a single logarithm.
- The real part of $C_0$ requires evaluations of only the real part (in the physical region) or only the imaginary part (in the unphysical region) of the dilogarithm, but not both. Calling PolyLog would lead to needless computation of both parts by the Mathematica Kernel. Following [15], the real and imaginary parts of the dilogarithm function are implemented separately.
- For the real part of $C_0$ in the physical region, the $+i\varepsilon$ prescription is irrelevant (since it influences only the imaginary part which is anyway evaluated using Cutkosky’s rule). Then, either the arguments of the 12 dilogarithms come in complex-conjugate pairs (for which the real part of the dilogarithms are identical and are added reducing the number of dilog evaluations), or the arguments are purely real (for which the real parts of the dilogarithms are rapidly evaluated using real arithmetic).
- The code is compiled to the Wolfram Virtual Machine (using Compile), leading to a substantial boost in computation speed.

In the physical region, up to a 200-fold increase in speed is achieved compared to brute-force Mathematica evaluation by the Kernel. In the unphysical region, up to a 20-fold increase in speed is obtained. If the option CompilationTarget → “C” to Compile is set, its performance rivals that of the Fortran implementation in LOOPTOOLS, with Package-X generating results approximately twice as fast.

VI. HANDLING THE $+i\varepsilon$ PRESCRIPTION AND SIMPLIFYING LOGARITHMS

The $+i\varepsilon$ prescription appearing in the denominators of propagator functions enforce causality in the time-ordered Green functions of a relativistic quantum field theory. In one-loop computations, it determines the branch on which the logarithms are to be evaluated. All output expressions of LoopRefine observe the $+i\varepsilon$ prescription and are consistent with the analytic conventions of the built-in Mathematica functions Log and PolyLog, which are

$$\begin{align*}
\text{Log}[x] &\rightarrow \lim_{\varepsilon \to 0^+} \ln(x+i\varepsilon), \\
\text{PolyLog}[2, x] &\rightarrow \lim_{\varepsilon \to 0^+} \text{Li}_2(x-i\varepsilon).
\end{align*}$$

Because Package-X assumes real external invariants and internal masses, almost all analytic formulae can be expressed compactly in terms of the built-in functions. Whenever LoopRefine generates a logarithm containing the ratio of two internal masses, the ratio may be flipped to bring the logarithm to ‘canonical form’, i.e.

$$\begin{align*}
\text{Log}\left[\frac{m_1^2}{m_0^2}\right] &\rightarrow -\text{Log}\left[\frac{m_0^2}{m_1^2}\right].
\end{align*}$$

Since internal masses are assumed to be positive real, this is allowed, and helps to keep the logarithmic parts compact.

In the course of reduction, regardless of whether the final expression is divergent or finite, multiple logarithms of ratios of several scales with the ’t Hooft parameter $\mu^2$ are typically generated, i.e.

$$a \text{Log}\left[\frac{\mu R^2}{-s}\right] + b \text{Log}\left[\frac{\mu R^2}{m^2}\right] + c \text{Log}\left[\frac{\mu R^2}{m^2-s}\right].$$

(27)

It is found that by consistently keeping $\mu^2$ in the numerator, the $+i\varepsilon$ prescription is always observed – even when the other scales are external invariants that may become time-like. The $\mu^2$ from each logarithm are brought into a single logarithm by forming the ratio with a variable that is known to be positive (which were recorded at STEP 1):

$$\begin{align*}
\begin{bmatrix} 27 \end{bmatrix} &= a \text{Log}\left[\frac{m^2}{-s}\right] + (a+b+c) \text{Log}\left[\frac{\mu R^2}{m^2}\right] \\
&\quad + c \text{Log}\left[\frac{m^2}{m^2-s}\right].
\end{align*}$$

(28)

That way, the coefficient—$(a+b+c)$ in this example—of the $\mu^2$-dependent logarithm always matches that of the

\footnote{If the explicit analytic form is desired, the option ExplicitCO → All can be supplied to LoopRefine.}
1/\epsilon$ pole elsewhere in the expression, and are grouped before presenting the results. If the final expression were in fact finite without a $1/\epsilon$ pole, the coefficient would cancel exactly.

In more complicated cases, expressions cannot be given compactly assuming a universal sign for the infinitesimal imaginary part. In this case, Mathematica’s built-in functions Log and PolyLog are unsuitable. For this purpose, two new analytic functions are defined in Package-X (within OneLoop):

\[
\begin{align*}
\text{Ln}[x,a] & \longrightarrow \lim_{\epsilon \to 0^+} \text{ln}(x + i\epsilon a), \text{ and} \\
\text{DiLog}[x,a] & \longrightarrow \lim_{\epsilon \to 0^+} \text{Li}_2(x + i\epsilon).
\end{align*}
\]

The (real part of the) second argument $a$ controls the side of the branch on which these functions evaluate. A simple example that uses DiLog in its output can be found by running

\[\text{LoopRefine[pvC0[0,m^2,s,0,m,m]}.\]

VII. SPUR: COMPUTATION OF TRACES OF DIRAC MATRICES

To assist in the evaluation of one-loop integrals with internal (closed) fermion lines, Package-X includes the rudimentary function Spur (inside the module Spur*) to evaluate traces over products of Dirac gamma matrices appearing in numerators. The function Projector helps to handle loop integrals with open fermion lines and is described in the next section. Because the primary function of Package-X is to compute loop integrals, with the computation of traces being a secondary feature, only a cursory description of the algorithms are given in the following two sections.

As with the rest of the algorithms in Package-X, the calculation of traces is rule-based at its core, and bears some resemblance to that of a much earlier Mathematica package Tracer [16]. However there a number of differences listed below that lead to greater computation speed.

- Throughout the evaluation of the trace, expressions can grow very large containing many terms. Groups of terms are temporarily enclosed within a List to prevent the Mathematica kernel from automatically simplifying the large expression at each step of the computation process.
- While more complicated trace formulae such as those for products of numerous gamma matrices are recursive, non-iterative rules are used for simpler tasks such as for collecting $\gamma_5$ and $\gamma\mu$ within each term.
- Products of gamma matrices with repeated Lorentz indices (such as $\gamma^\nu \gamma^\rho \gamma^\rho \gamma_\mu$) are related to products with fewer gamma matrices. With more gamma matrices interposed between contracted matrices, the number of terms in the identity grows. On account of the cyclic property of the trace, these contraction identities may be applied in one of two directions. Additional rules are included so as to apply the identity in the direction with fewer number of interposed gamma matrices.
- Traces that multiply $\gamma_5$ are tagged differently to set it apart from those without it. This way, rules for computing traces with $\gamma_5$ and those without $\gamma_5$ are separated, and saves some time when the kernel searches for the appropriate rules.

When compared to the other Mathematica packages FeynCalc and Tracer, Package-X generally gives results around 10 times faster. As an example, the trace

\[
\begin{align*}
\text{Tr} \left[ (\slashed{k} - \slashed{p}_1 - \slashed{p}_2 + m)\gamma^\nu (gl \slashed{P}_L + gr \slashed{P}_R)(\slashed{k} - \slashed{p}_2 + m) \\
\gamma^\rho (gl \slashed{P}_L + gr \slashed{P}_R)(\slashed{k} + m)\gamma^\lambda (gl \slashed{P}_L + gr \slashed{P}_R) \right] 
\end{align*}
\]

was calculated with each package and computation times were recorded (with Timing). The results on a 2.93 GHz Intel i7 processor are:

| Package             | Time    |
|---------------------|---------|
| Package-X           | 0.096 s |
| FeynCalc 9.2.0      | 1.03 s  |
| Tracer 1.1          | 0.81 s  |

Part of the motivation for refining the trace-taking algorithms is due to the inclusion of fermion projectors described in the next section. When a Projector is included inside Spur, the number of terms within the trace is increased to an extent that a noticeable slowdown is observed. However, with the refinements described above, projections onto form factors are nearly instantaneous on a modern computer.

VIII. PROJECTOR: PROJECTION ONTO FERMION FORM FACTORS

Package-X does not directly handle expressions involving open fermion chains that are relevant for fermion self energy and form factor calculations. In order to provide some support for such computations, Package-X comes equipped with a set of projectors. The projectors permit the projection of a loop integral with an open fermion line onto specific form factors functions.

For example, the one-loop expression for the off-shell fermion self-energy function takes the form

\[
I(p) = \mu^{2\epsilon} \int \frac{d^dk}{(2\pi)^d} \frac{\mathcal{M}(k,p)}{[k^2 - m^2][k + p]^2},
\]

where $\mathcal{M}(k,p)$ is a Dirac matrix structure that depends on the integration variable $k$ and external momentum $p$. Parity conservation and Lorentz matrix structure allow $I$ to be written in the form

\[
I(p) = A(p^2)\slashed{p} + B(p^2)m,
\]
where the form factors $A$ and $B$ depend on Lorentz invariants $p^2$ and $m^2$ only. By multiplying the appropriate projectors

$$\mathcal{F}^{[A]}(p, m) = \frac{1}{4p^2}$$

and

$$\mathcal{F}^{[B]}(p, m) = \frac{1}{4m^2}$$

with the numerator of (30), and taking the trace, the form factors are obtained:

$$A(p^2) = \mu^2 \int \frac{d^d k}{(2\pi)^d} \frac{\text{Tr}[\mathbb{M}(k, p) \mathcal{F}^{[A]}(p, m)]}{|k^2 - m^2|(|k + p|^2)}$$

$$B(p^2) = \mu^2 \int \frac{d^d k}{(2\pi)^d} \frac{\text{Tr}[\mathbb{M}(k, p) \mathcal{F}^{[B]}(p, m)]}{|k^2 - m^2|(|k + p|^2)}.$$

The trace over the projectors convert the expressions into ordinary tensors integrals that are readily computed with Package-X.

A large set of pre-programmed projectors for off-shell self energy functions and on-shell scalar- and vector-vertex functions in various bases (L/R-chiral or Vector/Axial-vector) are available (as Projector) to streamline the computation of such integrals. These projectors are generalizations of those used in [17] for the calculation of lepton anomalous magnetic moments, and in [18] for dipole moments. A comprehensive list of available projectors is given in the built-in documentation files.

**IX. CONCLUSIONS AND FURTHER DEVELOPMENT**

The utility of Package-X is in obtaining compact analytic expressions for one-loop integrals that can be symbolically manipulated and numerically evaluated with Mathematica’s built-in routines. Results to loop integrals have been extensively hand-checked against known results in the literature, and cross-checked against numerical integration. Listed below are important limitations of Package-X that guides its current line of development.

1. An analytic series expansion of the loop integral in kinematic variables is not generally possible. Currently, the only available method is to use Mathematica’s Series on the output of LoopRefine. However, if the result of loop integral contains specially defined function like pvC0, then Series will not work. Given that much information about a loop-integral can be gleaned from its expansion, the omission of this feature is most conspicuous.

2. Package-X currently supports loop integrals with up to only three denominator factors. But, as the number of denominator factors increases, so does the complexity of their analytic forms. Thus, it would not be so practical to work with such expressions for higher-point functions even if Package-X were to provide them. However, at special kinematic points such as at zero external momenta or at thresholds, compact expressions could be obtained.

3. Gamma-5 is implemented naively in dimensional regularization. This means that the VVA or AAA three-point functions may not automatically satisfy Ward identities appropriate to the physical problem. However, the versatility of Package-X makes it easy to apply Adler’s method [19] (see also [20]) to enforce the Ward identities.

4. Loop integrals with open fermion chains are not directly supported. As explained in Section VIII there is no way to input an open string of Dirac matrices. Instead, the computation of fermion form factors can be done by projecting out the needed form factors.

**ACKNOWLEDGMENTS**

I express my gratitude to the members of the Mathematica StackExchange community for providing countless answers to my questions regarding the technical aspects of Mathematica and Wolfram Workbench. I thank Ansgar Denner for clarifying discussions regarding the reduction methods and for providing helpful information regarding the literature. Numerous colleagues have provided useful feedback: Huaike Guo for cross-checking several results during the early stages of the development, and Xunjie Xu for encouraging me to refine the trace-taking algorithms.

I also acknowledge my debt to beta tester Johannes Welter for cross-checking the fermion form factor projectors, and to beta testers Jurij Smirnov and Michael Durr for identifying bugs. Special thanks goes to Michael Durr for meticulously hand-checking various results of the reduction algorithms, and for suggesting improvements to the user interface, the tutorial, and the accompanying documentation pages.

**Appendix A: Conventions**

For reference, the conventions for spacetime quantities are summarized in Table II. Conventions for the Passarino-Veltman functions are displayed in Table III. Note that a slightly-unconventional ordering and form of the arguments of the Passarino-Veltman functions is

| Quantity                        | Convention |
|---------------------------------|------------|
| Metric signature $g_{\mu\nu}$   | $\text{diag}(+,-,-,-)$ |
| Spacetime dimension $d$         | $d = 4 - 2\epsilon$ |
| Dirac matrix commutator $\sigma_{\mu\nu} = \frac{i}{2}[\gamma_{\mu}, \gamma_{\nu}]$ | |
| Fifth gamma matrix $\gamma_5 = \gamma^0\gamma^1\gamma^2\gamma^3$ | |
| Chiral Projectors $\hat{P}_L = \frac{1}{2}(1 - \gamma_5)$, $\hat{P}_R = \frac{1}{2}(1 + \gamma_5)$ | |
| Levi-civita symbol $\epsilon^{0123} = +1$ | |

**TABLE II. Conventions for spacetime quantities**
TABLE III. Conventions for the arguments of the Passarino-Veltman functions

| Function     | Diagram |
|--------------|---------|
| $A_0(m_0)$   | ![Diagram](image) |
| $B_0(p^2, m_0, m_1)$, and $b_0^2(p^2, m_0, 0)$ | ![Diagram](image) |
| $C_0(p^2, p_1^2, q^2, m_2, m_1, m_0)$, $q^2 = (p_2 - p_1)^2$ | ![Diagram](image) |

The coefficient integrated with the help of the multinomial theorem. The needed UV-divergent parts are those of the $B_1$, $B_2$, and $B_3$ functions. They are controlled by the leading gamma function which for large enough $\epsilon$ pole as

$$A_{0,0,m_0}(m_0) = (4\pi\mu^2)^\epsilon \frac{(-1)^{1+r}}{2^r} \Gamma(1+\epsilon-r) m_0^{1-\epsilon+r} \quad (B1)$$

$$B_{0,0,1,1,m_0,m_1}(p^2; m_0, m_1) = (4\pi\mu^2)^\epsilon \frac{(-1)^{2+r+n}}{2^r} \Gamma(\epsilon - r)$$

$$\times \int_0^1 \frac{dx x^n}{(p^2x^2 + (-p^2 + m_1^2) x + m_0^2 - i\epsilon)^{\epsilon - r}} \quad (B2)$$

$$b_{0,0,1,1}^2(p^2; m) = (4\pi\mu^2)^\epsilon \frac{(-1)^{3+r+n}}{2^r} \Gamma(1 + \epsilon - r)$$

$$\times \int_0^1 \frac{dx x^n (1 - x)}{(p^2x^2 + (-p^2 + m_1^2 - m_0^2) x + m_0^2 - i\epsilon)^{1+\epsilon-r}} \quad (B3)$$

The coefficient $C$ function exhibits an invariance under the simultaneous interchange of indices $n_1 \leftrightarrow n_2$, external momenta $p_1^2 \leftrightarrow p_2^2$ and internal masses $m_1 \leftrightarrow m_2$.

$$C_{0,0,0,1,1,1,2,2}(p^2, p_1^2, p_2^2, q^2; m_2, m_1, m_0) = C_{0,0,0,1,1,1,2,2}(p_1^2, p_2^2, q^2; m_2, m_1, m_0) = C_{0,0,0,1,1,1,2,2}(p^2, p_1^2, p_2^2, q^2; m_1, m_2, m_0) \quad (B5)$$

and is frequently employed during its reduction in the most general kinematic case ($\det Z \neq 0$).

In certain reduction formulae of $C$-functions, some terms are multiplied by $\epsilon$, which in the $\epsilon \to 0$ limit, pick up the UV-divergent parts of the coefficient functions in those terms. The UV divergent parts are readily obtained from the integral representation. They are controlled by the leading gamma function which for large enough $r$ develops a $1/\epsilon$ pole as $\epsilon \to 0$. When $r$ is large, the integrand becomes polynomial in the Feynman parameters and are readily integrated with the help of the multinomial theorem. The needed UV-divergent parts are those of the $B$ and $C$ functions, shown below.

$$B_{0,0,0,1,1,1,1}^{\text{UV, Div.}}(p^2; m_0, m_1) = \sum_{k_1+k_2+k_3=r} \left( \frac{1}{2k_1 + k_2 + n + 1} \right)^\epsilon (B6)$$
where \( a = p^2, b = -p^2 + m_1^2 - m_0^2 \), and \( c = m_0^2 \), are polynomial coefficients of the integrand in \([B2]\).

\[
C_{0,0 \underbrace{1,1 \ldots 2,2}_{n_1} \underbrace{2,2}_{n_2}}(p_1^2, p_2^2, q^2; m_2, m_1, m_0) \bigg|_{\text{UV-Div.}} = \frac{(-1)^{n_1+n_2}}{2^r (r-1)!} \sum_{k_1 + \ldots + k_6 = r-1} \left( r-1 \right) a^{k_1} b^{k_2} c^{k_3} d^{k_4} e^{k_5} f^{k_6} \frac{(2k_1 + k_3 + k_4 + n_1)!(2k_2 + k_3 + k_5 + n_2)!(2k_1 + 2k_2 + 2k_3 + k_4 + k_5 + n_1 + n_2 + 2)!}{(2k_1 + 2k_2 + 2k_3 + k_4 + k_5 + n_1 + n_2 + 2)!} \frac{1}{\varepsilon},
\]

(B7)

where \( a, b, c, d, e, \) and \( f \) are polynomial coefficients of the integrand in \([B4]\) in the order displayed.

**Appendix C: Derivation of reduction formulae for \( C \) functions Case 2**

The derivation of the first equation in \([19]\) begins with the Feynman parameter representation of the coefficient \( C \) function,

\[
C_{0,0 \underbrace{1,1 \ldots 2,2}_{n_1} \underbrace{2,2}_{n_2}}(m_0^2, s, m_2; m_2, 0, m_0) = \frac{(4\pi\mu^2)^r}{2^r} \frac{(-1)^{3+r+n_1+n_2}}{2^r} \Gamma(1 + \epsilon - r) \times \int_0^1 dy \int_0^{1-y} dz \ y^{n_1} z^{n_2} \left[ m_0^2 y^2 + s z^2 + (-m_0^2 + m_2^2 + s) y z + (-m_0^2 + m_2^2 - s) z + m_0^2 - i\varepsilon \right]^{-1-\epsilon+r}. \tag{C1}
\]

Upon making a change of integration variables \( y = 1 - y' \) and \( z = y' z' \), the nested integrals are factored:

\[
\int_0^1 dy' y'^{-1+n_2-2\epsilon+2\epsilon}(1-y')^{n_1} \int_0^1 dz' z'^{n_2} \left[ s z'^2 + (-s + m_2^2 - m_0^2) z' + m_0^2 - i\varepsilon \right]^{-1-\epsilon+r}. \tag{C2}
\]

The \( y' \) integral gives the Euler Beta function, while the \( z' \) integral is identified as the integral representation of coefficient \( B \)-function \([B2]\).

\[
C_{0,0 \underbrace{1,1 \ldots 2,2}_{n_1} \underbrace{2,2}_{n_2}}(m_0^2, s, m_2; m_2, 0, m_0) = \frac{(-1)^{n_1}}{2} B(n_2 - 2\epsilon + 2\epsilon, n_1 + 1) B_{0,0 \underbrace{1,1 \ldots 2,2}_{n_1} \underbrace{2,2}_{n_2}}(s; m_0, m_2) \tag{C3}
\]

As long as one of \( n_2 \) or \( r \) is non-zero, the Beta function is finite, and its expansion to \( O(\epsilon) \) may be inserted yielding the first equation in \([19]\).

On the other hand, if \( n_2 = r = 0 \), the Beta function develops a \( 1/\epsilon \) pole, so that to \( O(\epsilon) \),

\[
B(-2\epsilon, n_1 + 1) = \frac{1}{-2\epsilon} - H_{n_1} - \epsilon \left( H_{n_1}^{(2)} - H_{n_1}^{(0)} \right). \tag{C4}
\]

In this case, \((C1)\) is written as

\[
C_{1,1 \underbrace{1,1 \ldots 2,2}_{n_1}}(m_0^2, s, m_2; m_2, 0, m_0) = (4\pi\mu^2)^r (-1)^{n_1+1} \Gamma(1 + \epsilon) \frac{1}{2\epsilon} \int_0^1 dz \left[ s z'^2 + z' (-s + m_2^2 - m_0^2) + m_0^2 - i\varepsilon \right]^{-1-\epsilon} + (4\pi\mu^2)^r (-1)^{n_1+1} \Gamma(1 + \epsilon) \left( H_{n_1} + \epsilon \left( H_{n_1}^{(2)} - H_{n_1}^{(0)} \right) \right) \int_0^1 dz \left[ s z'^2 + z' (-s + m_2^2 - m_0^2) + m_0^2 - i\varepsilon \right]^{-1-\epsilon}. \tag{C4}
\]

While the \( z' \) integral in the second line can be identified with the integral representation of the coefficient \( B \) function, the first line is identified as the integral representation of the scalar function \( C_0(m_0^2, s, m_2^2; m_2, 0, m_0) \) classified by Ellis and Zanderighi \([19]\) as IR-divergent triangle 6. These identifications lead to the second equation of \([19]\).

If the off-shell momentum \( s \) is in the third argument, the derivation starts with the change of variables \( z = 1 - y - x \) in \([B4]\) followed by an interchange of the \( x \) and \( y \) integrals to give

\[
C_{0,0 \underbrace{1,1 \ldots 2,2}_{n_1} \underbrace{2,2}_{n_2}}(m_0^2, m_0^2, s; m_0, m_0, m_0) = (4\pi\mu^2)^r \frac{(-1)^{3+r+n_1+n_2}}{2^r} \Gamma(1 + \epsilon - r)
\]

\[
\times \int_0^1 dx \int_0^{1-x} dy \ y^{n_1} (1-x-y)^{n_2} \left[ m_0^2 x^2 + s y^2 + (s + m_0^2 - m_2^2) x y - 2m_0^2 x + (-s - m_0^2 + m_2^2) y + m_0^2 - i\varepsilon \right]^{-1-\epsilon+r}. \tag{C5}
\]

\(^4\) see \(\text{http://qcdloop.fnal.gov/tridiv6.pdf}\)
The nested integrals are factored by making a further change of variables \( x = 1 - x' \) and \( y = y' x' \) to give

\[
\int_0^1 dx' \int_0^{y'} \left[ \left( 1 - y' \right)^{n_2} \left( a y'^2 + bz'^2 + c y' z' + dy' + e z' + f \right)^{-1-\epsilon+r} \right] dy' \]  
\]

The \( x' \) integral is straightforward. The \( y' \) integral can be brought to a recognizable form after expanding the factor \( \left( 1 - y' \right)^{n_2} \) as a binomial series

\[
= \sum_{k=0}^{n_2} \binom{n_2}{k} \left( -1 \right)^k \int_0^1 dy' \int_0^{y'} \left[ sy'^2 + (-s + m_2^2 - m_0^2)y' + m_0^2 - i \varepsilon \right]^{-1-\epsilon+r}. 
\]

The \( y' \) integral is now identified as the integral representation of the coefficient \( B \) function, yielding (20).

**Appendix D: Derivation of reduction formulae for \( C \) functions Case 4**

Although the steps below leading to (22) and (23) appear complicated, they essentially follow that of [10] for the evaluation of the scalar function \( C_0 \). Beginning with the integral representation (B4), a change of integration variables \( y = 1 - y' \) brings the Feynman integrals to the form

\[
\int_0^1 dy' \int_0^{y'} dz \left( 1 - y' \right)^{n_1} \left[ ay'^2 + bz'^2 + cy'z' + dy' + ez' + f \right]^{-1-\epsilon+r} 
\]

where \( a = p_1^2, b = p_2^2, c = q^2 - p_1^2 - p_2^2, d = -p_1^2 + m_0^2 - m_2^2, e = p_1^2 - q^2 - m_0^2 + m_2^2, \) and \( f = m_1^2 - i \varepsilon \). A second change of variables is made \( z = z' + ay' \), with \( \alpha = \frac{1}{2 \varepsilon} \) chosen to make the coefficient of \( y'^2 \) in square brackets to vanish.

\[
\int_0^1 dy' \int_0^{(1-\alpha)y} dz \left( 1 - y' \right)^{n_1} \left( z' + ay' \right)^{n_2} \left[ bz'^2 + (c + 2b \alpha) y' z' + (d + e \alpha) y' + e z' + f \right]^{-1-\epsilon+r} 
\]

The choice for \( \alpha \) implies that \( c + 2b \alpha \) vanishes, and the kinematic relations \( \det Z = X_0 \) = 0 implies that \( d + e \alpha \) vanishes, yielding

\[
\int_0^1 dy \int_0^{(1-\alpha)y} dz \left( 1 - y \right)^{n_1} \left( z + ay \right)^{n_2} \left[ bz^2 + e z + f \right]^{-1-\epsilon+r}, 
\]

where the primes have been omitted. The binomial theorem is applied to the factor \( (z+ay)^{n_2} = \sum_j (\binom{n_2}{j} a^{n_2-j} y^{n_2-j} z^j) \) and the order of integrations are interchanged so that

\[
\sum_{j=0}^{n_2} \binom{n_2}{j} a^{n_2-j} \int_0^1 dz \int_0^{z/(1-\alpha)} dy - \int_0^{-\alpha} dz \int_0^{1-\alpha} dy \left( 1 - y \right)^{n_1} y^{n_2-j} z^j \left[ bz^2 + e z + f \right]^{-1-\epsilon+r}. 
\]

The \( y \) integrals in both terms yield terminating hypergeometric series most compactly written in terms of the incomplete Beta function:

\[
\int_0^1 dy (1 - y)^{n_1} y^{n_2-j} = \frac{n_1!(n_2-j)!}{(n_1 + n_2 - j + 1)!} - B_X(n_2-j+1, n_1+1) = \frac{n_1!(n_2-j)!}{(n_1 + n_2 - j + 1)!} \sum_{k=0}^{n_1} \frac{X^{n_2-j+k+1}}{(n_2-j+k+1) (n_1)_k} (-1)_k. 
\]

Since the first term of (D5) is common to both integrations in (D4), they are combined to yield a total of three terms:

\[
\int_0^{1-\alpha} dz B_{z/(1-\alpha)}(n_2-j+1, n_1+1) = \int_0^{1-\alpha} dz B_{z/(1-\alpha)}(n_2-j+1, n_1+1) z^j + \int_0^{1-\alpha} dz B_{z/(1-\alpha)}(n_2-j+1, n_1+1) z^j 
\]

(D6)
A change of integration variables is carried out in each term to stretch their ranges to $0 \to 1$: In the first integral, $z = z' - \alpha$, in the second integral $z = (1 - \alpha)z'$, and in the third integral $z = -\alpha z'$. Consequently, the polynomials $b z^2 + c z + f$ take the shape of integrands for the $B$ functions.\footnote{These quadratic polynomials may be recognized as the ‘pinch functions’ originating from the three cut channels of the triangle graph.}

First term: \[ p^2 z'^2 + (-p^2 + m^2 - m^2_0)z' + m^2_0 - i\epsilon \quad := \quad P_2(z') \]

Second term: \[ q^2 z'^2 + (-q^2 + m^2 - m^2_0)z' + m^2_0 - i\epsilon \quad := \quad P_2(z') \]

Third term: \[ p^2 z'^2 + (-p^2 + m^2 - m^2_0)z' + m^2_0 + i\epsilon \quad := \quad P_1(z') \]

Upon inserting the series representation of the incomplete Beta function \((D5)\) the result is (after dropping the primes on \(z\))

\[
\text{integrals} = \sum_{j=0}^{n_2} \binom{n_2}{j} \alpha^{n_2-j} \left\{ \frac{n_1!(n_2-j)!}{(n_1+n_2-j+1)!} \int_0^1 dz (z-\alpha)^j P_2(z)^{-1-\epsilon+r} \right. \\
+ \sum_{k=0}^{n_1} \frac{1}{n_2-j+k+1} \left( -1 \right)^{j+1} \int_0^1 dz z^{n_2+k+1} P_2(z)^{-1-\epsilon+r} + (-\alpha)^{j+1} \int_0^1 dz z^{n_2+k+1} P_1(z)^{-1-\epsilon+r} \right\} 
\]

(E7)

In the first term, the binomial theorem is applied to \((z-\alpha)^j = \sum_k \binom{j}{k} (-\alpha)^{j-k} z^k\), and the three integrals are finally identified as integral representations of the coefficient $B$ functions upon which \((22)\) is obtained.

If $p^2 = 0$, then $\det X = 0$ implies $p^2 = q^2$ and $X_{0j} = 0$ implies $m_0 = m_2$. In this case, the integrals in \((D1)\) are already factored:

\[
\text{integrals} = \int_0^1 dy \int_0^{y'} dz (1-y')^{n_1} z^{n_2} \left[ p_1^2 y'^2 + (-p_1^2 + m_0^2 - m_0^2) y' + m_0^2 - i\epsilon \right]^{-1-\epsilon+r} 
\]

(E8)

The $z$ integration gives a factor $1/(n_2+1)$, and the factor $(1-y')^{n_1} = \sum_k \binom{n_1}{k} (-y)^k$ is expressed as a binomial series.

\[
\text{integrals} = \frac{1}{n_2+1} \sum_{k=0}^{n_1} \binom{n_1}{k} (-1)^k \int_0^1 dy' y'^{n_2+k+1} \left[ p_1^2 y'^2 + (-p_1^2 + m_0^2 - m_0^2) y' + m_0^2 - i\epsilon \right]^{-1-\epsilon+r} 
\]

(E9)

Eqn \((23)\) is obtained after identifying the $y'$ integration as the integral representation of the coefficient $B$ function.

Appendix E: Coefficient $B$ functions with $r = -1$

The two new reduction algorithms for $C$ functions (Cases 2 and 4) require extending the set of basis functions to include $B$ functions in which the index $r$ in \((12)\) is continued to $-1$. In a certain sense, these new reduction formulae may be closely related to those in \([22]\). The authors present different reduction formulae for coefficient functions which require extending the set of basis functions to scalar functions with repeated propagators.

A set of explicit expressions for the general case and at singular points is constructed and included in the Package-X source file (see Table\(\ref{tab:coefficients}\)). The integration is straightforward in most cases. The functions are UV-finite for all $n \geq 0$, but with many kinematic configurations developing IR-divergent $1/\epsilon$ poles.

However, there are three kinematic cases, all corresponding to physical threshold for which the Feynman parameter integral diverges even for finite $\epsilon$. To handle these cases, $\epsilon$ is taken sufficiently large and negative so that the integral converges, and then analytically continued to $\epsilon \to 0$. The results of these integrations are given below:

\[
B_{0\ldots 0 \ldots 1 \ldots 1}(m^2_0; 0, m_1) = -\frac{2}{m_1^2} \left( \frac{4\pi\mu^2}{m_1^2} \right) \Gamma(1+\epsilon) \int_0^1 dx x^{-2-2\epsilon} = \frac{2}{m_1^2} 
\]

(E1)

\[
B_{0\ldots 0 \ldots 1 \ldots 1}(m^2_0; m_0, 0) = \frac{2}{m_0^2} \left( \frac{4\pi\mu^2}{m_0^2} \right)^n (-1)^{n+1} \Gamma(1+\epsilon) \int_0^1 dx x^n(x-1)^{-2-2\epsilon} = \left\{ \begin{array}{ll}
\frac{(-1)^{n+1} m_0^2}{2} n \left( \frac{1}{\epsilon} + \ln \left( \frac{\mu^2}{m_0^2} \right) + 2H_{n-1} - 2 \right), & n \geq 1 \\
\frac{2}{m_0^2}, & n = 0
\end{array} \right. 
\]

(E2)
Among these integrals, only (E3) gives numerical results that are related to limiting values as threshold is reached: the real part of (E3) corresponds to the limiting value of $\text{Re} B(s; m_0, m_1)$ when approached from above threshold, and the imaginary part corresponds to the limiting value of $\text{Im} B(s; m_0, m_1)$ when approached below threshold.

That the integrals (E1-E3) give numerical results that do not match their limiting values as threshold is reached are not of any concern. The results above should be viewed as ill-defined divergent integrals arising at intermediate stages in the reduction of coefficient $C$ functions. They serve to facilitate the analytic cancellation of these integrals at the end of a physically meaningful computation, such as for the electromagnetic contribution to the electron anomalous magnetic moment.

[1] R. Mertig, M. Bohm, and A. Denner, Comput. Phys. Commun. 64, 345 (1991).
[2] T. Hahn and M. Perez-Victoria, Comput. Phys. Commun. 118, 153 (1999), arXiv:hep-ph/9807565 [hep-ph].
[3] T. Binoth, J.-P. Guillet, G. Heinrich, E. Pilon, and T. Reiter, Comput. Phys. Commun. 180, 2317 (2009).
[4] G. Passarino and M. Veltman, Nucl. Phys. B160, 151 (1979).
[5] A. Ilakovac and L. Popov, (2014), arXiv:1407.2727 [hep-ph].
[6] P. W. Angel, Y. Cai, N. L. Rodd, M. A. Schmidt, and R. R. Volkas, JHEP 1310, 118 (2013), arXiv:1308.0463 [hep-ph].
[7] A. Denner and S. Dittmaier, Nucl. Phys. B734, 62 (2006), arXiv:hep-ph/0509141 [hep-ph].
[8] D. Y. Bardin and G. Passarino, The standard model in the making: Precision study of the electroweak interactions (Oxford Science Publications, 1999).
[9] R. K. Ellis and G. Zanderighi, JHEP 0802, 002 (2008), arXiv:0712.1851 [hep-ph].
[10] G. ’t Hooft and M. Veltman, Nucl. Phys. B153, 365 (1979).
[11] L. G. Cabral-Rosetti and M. A. Sanchis-Lozano, J. Phys. Conf. Ser. 37, 82 (2006), arXiv:hep-ph/0206081 [hep-ph].