2.1.1 by Hiren H. Patel
hpatel6@ucsc.edu

Primer
March 29, 2018

Contents

A message from the author

In this tutorial, I provide an introduction to the use of *Package*-**X**. This tutorial contains a series of simple examples to illustrate basic usage. To follow them, I expect that the reader has some familiarity in the evaluation of dimensionally regulated one-loop integrals, and in the use of *Mathematica*. But because I designed this package with a shallow learning curve in mind, they are (hopefully) easy to follow, and it should not take too long before becoming acquainted with the package.

1 Installing and loading the package

Package-**X** can be installed on any computer that runs *Mathematica* 8.0 or higher. However, future releases of *Package*-**X** will require *Mathematica* 10.1 or higher. The latest version of the program, including expansion packs, is hosted at the hepforge project page `http://packagex.hepforge.org`.

After downloading and uncompressing the zip file, place the folder `/X` and all its contents within the directory *$UserBaseDirectory*`/Applications/`. Here, *$UserBaseDirectory* can be found by running **$UserBaseDirectory** within a *Mathematica* session:

In[1]:=   **$UserBaseDirectory**

Out[1]=   `/Users/xxxx/Library/Mathematica`

It is necessary to restart *Mathematica* for the system to find and merge the *Package*-**X** help files with the *Mathematica Documentation Center*. For uninstallation, simply remove the folder from the directory; the help files will automatically be removed.

*On an American keyboard, the grave accent ` is found just below the* `esc` *key.*

You can load the package in the current session with the command:

In[1]:=   **<< X`**

        Package-X v2.1.1, by Hiren H. Patel
        For more information, see the guide

This will activate all definitions made by *Package*-**X**, and the functions defined within are ready for use.

2 Using the Wolfram Documentation Center

Included with *Package*-**X** is a set of documentation files which is automatically merged with the Wolfram Documentation Center upon starting *Mathematica*. The documentation files contain details and examples relevant to every symbol/function defined by *Package*-**X**. One way to view these files is by visiting the guide page via the printed link when *Package*-**X** is initialized.

Alternatively, pressing `F1` when the cursor is within a function name will bring up the relevant help page for that function.

Finally, brief information about a function can be obtained by using '**?**' followed by its name. For example,

In[1]:=   **? LoopRefine**

> LoopRefine[*expr*] converts the Passarino-Veltman coefficient functions
> in *expr* to analytic expressions that can be numerically evaluated. ≫

Tensor	Standard-Form	FullForm
p_μ	$\mathbf{p}_\mu$	`LTensor[p, `μ`]`
γ_μ	$\mathbf{\gamma}_\mu$	`LTensor[DiracG, `μ`]`
$g_{\mu\nu}$	$\mathfrak{g}_{\mu,\nu}$	`LTensor[MetricG, `μ`, `ν`]`
$\varepsilon_{\mu\nu\rho\varsigma}$	$\mathbf{\varepsilon}_{\mu,\nu,\rho,\varsigma}$	`LTensor[LeviCivitaE, `μ`, `ν`, `ρ`, `ς`]`

Table 1: Some *Package*-**X** tensors in `StandardForm` (suitable for notebook input) and `FullForm` (suitable for command-line input)

And additional information can be obtained by following the link '≫' which takes you to the help page in the Documentation Center.

3   Preliminaries: Input

Package-**X** is designed with increased readability of input code by having its syntax closely resemble corresponding expressions that would be written by hand. Therefore, it is natural that special input methods and symbols would be an integral part of the input for functions defined by *Package*-**X**. In this section, the preliminaries are introduced.

3.1   Entering Lorentz dot products (e.g. `k.p`)

Once *Package*-**X** is loaded, the shortcut infix operator '`.`' is remapped to a new function **LDot** that represents the Lorentz dot product.

```
In[1]:=  (*Lorentz dot product in Package-X*)
         p.k
Out[1]=  k.p
```

The built-in *Mathematica* function **Dot** is still available to you to perform standard matrix multiplication or inner products, but it can no longer be invoked by the shortcut.

```
In[2]:=  (*Mathematica's inner product*)
         Dot[{a, b, c}, {d, e, f}]
Out[2]=  a d + b e + c f
```

You can input dotted four-vectors in any order; *Mathematica* will canonically order them freely as in the example above. To save input time, you can enter dot products of vector sums or differences, *e.g.* `(p + k).(q - k)`.

3.2   Entering four-vectors (e.g. $\mathbf{p}_\mu$, $\mathbf{\gamma}_\mu$) and tensors (e.g. $\mathfrak{g}_{\mu,\nu}$)

Computations performed with *Package*-**X** are intended to be fully Lorentz covariant. Thus the distinction between covariant p^μ and contravariant p_μ indices is unnecessary. All indices will appear in "contravariant" position in your notebook (*e.g.* $\mathbf{p}_\mu$). You can enter four-vectors and higher rank tensors (see Table 1) by placing their indices in subscript position (short-cut `ctrl`+`–`), and *Package*-**X** will automatically parse the syntax as **LTensor**.

The function **Contract** is defined to recognize repeated indices and will form the appropriate dot-products.

```
In[1]:=  Contract[p_μ k_ν k_μ]
Out[1]=  k.p k_ν
```

Glyph	Alias	Description
$\mathbb{1}$	`esc` 11 `esc`	unit Dirac matrix
$\boldsymbol{\gamma}$	`esc` g `esc`	Dirac matrix
$\boldsymbol{\sigma}$	`esc` s `esc`	Dirac sigma matrix
$\boldsymbol{\gamma}5$	`esc` g `esc` 5	Gamma-5 matrix
$\mathbb{P}\mathbf{L}$	`esc` PL `esc`	left chiral projector
$\mathbb{P}\mathbf{R}$	`esc` PR `esc`	right chiral projector
$\mathbf{g}_{\square,\square}$	`esc` gg `esc`	metric tensor
$\boldsymbol{\varepsilon}_{\square,\square,\square,\square}$	`esc` levi `esc`	Levi-Civita symbol

Table 2: Aliases for notebook entry of commonly used symbols

Note that subscripts tied to sums/differences of symbols are understood to be four-vector sums/differences (*e.g.* $(\mathbf{p} + 2\,\mathbf{k})_\mu \equiv (p_\mu + 2k_\mu)$), and can save you time during input.

Pay particular attention to the syntax for higher rank tensors, such as $\mathbf{g}_{\mu,\nu}$ and $\boldsymbol{\varepsilon}_{\mu,\nu,\rho,\varsigma}$ (see next section), for which the indices are delimited by commas. Forgetting to include the comma separators would be interpreted by *Mathematica* as multiplication of the corresponding symbols inside the subscript.

4 Tutorial 1: Taking traces

4.1 Entering special glyphs (e.g. $\mathbb{1}$, $\boldsymbol{\gamma}$, $\mathbf{g}$)

Consult the Package-**X** *guide page in the Wolfram Documentation Center for a complete list.*

Let us begin using *Package*-**X** to evaluate traces over products of Dirac matrices with the function **Spur**. In order to maintain readability of code, certain glyphs have been defined for input. Special aliases (keystroke combination `esc`, $\cdots$, `esc`) have been reserved for rapid notebook entry. Some commonly used symbols are in table 2.

We begin by evaluating the following trace:

$$\mathrm{Tr}\Big[(\not{p}_1 - m)\,\gamma_\mu(\not{p}_2 - m)\,\gamma_\nu\Big]$$

The corresponding input is as follows (consult table 2 for entering $\boldsymbol{\gamma}$ and $\mathbb{1}$, and section 3.2 for entering subscripted indices)[1]:

In[1]:= **Spur[$\boldsymbol{\gamma}$. p1 − m$\mathbb{1}$, $\boldsymbol{\gamma}_\mu$, $\boldsymbol{\gamma}$. p2 − m$\mathbb{1}$, $\boldsymbol{\gamma}_\nu$]**

Out[1]= $4\ \mathrm{p1}_\nu\,\mathrm{p2}_\mu + 4\ \mathrm{p1}_\mu\,\mathrm{p2}_\nu + 4\ \mathrm{m}^2\ \mathbf{g}_{\mu,\nu} - 4\ \mathrm{p1.p2}\ \mathbf{g}_{\mu,\nu}$

Note that you must enter the Feynman slash $\not{p}$ by typing out the scalar dot-product of γ and p in full. Also, the glyph $\mathbb{1}$ stands for the unit 4×4 Dirac matrix which, although conventionally implicit in written form, must be supplied explicitly for robustness of the syntax. It is to be used multiplicatively when a constant is present (in this example **m**) that does not multiply any other object in spinor space (such as $\boldsymbol{\gamma}$, $\boldsymbol{\sigma}$, or $\boldsymbol{\gamma}5$).

As a general rule, you can add objects in spinor-space together *e.g.*

$$\mathrm{Tr}\Big[1 + c\gamma_5\Big] \equiv \mathbf{Spur[\mathbb{1} + c\,\boldsymbol{\gamma}5]}\,,$$

---

[1]Greek symbols can also be inserted rapidly using *Mathematica*'s pre-defined aliases. For example, Greek letter μ is obtainable from `esc` m `esc`; symbol ν from `esc` n `esc`, *etc*. Although Greek symbols are not required for entering subscripts, they make for readable code.

but you must set off products by comma delimiters to preserve ordering of matrix-multiplication *e.g.*

$$\text{Tr}\big[\gamma_\mu\gamma_\nu\gamma_\lambda\gamma^\mu\big] \equiv \texttt{Spur[}\boldsymbol{\gamma}_\mu\texttt{, }\boldsymbol{\gamma}_\nu\texttt{, }\boldsymbol{\gamma}_\lambda\texttt{, }\boldsymbol{\gamma}_\mu\texttt{]} .$$

4.2 Tree-level $Z^0 \to f\bar{f}$ decay width

In this next example, we will compute the partial width of the Z^0 gauge boson disintegration into two fermions f. Remember that *Package-X* is *not* FEYN-CALC: it will not give a partial width starting from the Feynman rules. Instead, **Spur** can be used to evaluate the traces that appear in the intermediate steps of the computation. Using the Feynman rules for electroweak theory, we find that the tree-level amplitude is given by:

$$i\mathcal{M} = \bar{u}_{s'}(\mathbf{p}')\frac{-ie\gamma^\mu}{\sin 2\theta_W}(g_V^f - g_A^f\gamma_5)v_s(\mathbf{p})\,\epsilon_\mu^{[\lambda]}(\mathbf{k})$$

To obtain the unpolarized decay width, square the amplitude, average over initial helicity states $\frac{1}{3}\sum_\lambda$, and sum over final spins and colors $\sum_{s,s'}\sum_{\text{color}}$.

$$|\overline{\mathcal{M}}|^2 = \frac{e^2}{\sin^2(2\theta_W)}N_c\sum_{s,s'}\Big(u_{s'}(\mathbf{p}')\bar{u}_{s'}(\mathbf{p}')\gamma^\mu(g_V^f - g_A^f\gamma_5)$$

$$\times v_s(\mathbf{p})\bar{v}_s(\mathbf{p})(g_V^f + g_A^f\gamma_5)\gamma^\nu\Big)\frac{1}{3}\sum_\lambda\epsilon_\mu^{[\lambda]}(\mathbf{k})\epsilon_\mu^{*[\lambda]}(\mathbf{k}) \quad (1)$$

$$= \frac{e^2}{\sin^2(2\theta_W)}\frac{N_c}{3}\text{Tr}\Big[(p\!\!\!/' + m_f)\,\gamma^\mu\,(g_V^f - g_A^f\gamma_5)$$

$$\times (p\!\!\!/ - m_f)(g_V^f + g_A^f\gamma_5)\,\gamma^\nu\Big]\Big(-g_{\mu\nu} + \frac{k_\mu k_\nu}{m_Z^2}\Big) \quad (2)$$

Now, use *Package-X* to compute the trace appearing in equation 2 (we represent $p = $**p1** and $p' = $**p2**):

In[1]:= **Spur[$\boldsymbol{\gamma}$.p2 + mf 1, $\boldsymbol{\gamma}_\mu$, gV 1 – gA $\boldsymbol{\gamma}$5, $\boldsymbol{\gamma}$.p1 – mf 1, gV 1 + gA $\boldsymbol{\gamma}$5, $\boldsymbol{\gamma}_\nu$]**

Out[1]= $4\,\text{gA}^2\,\text{p1}_\nu\text{p2}_\mu + 4\,\text{gV}^2\,\text{p1}_\nu\text{p2}_\mu + 4\,\text{gA}^2\,\text{p1}_\mu\text{p2}_\nu + 4\,\text{gV}^2\,\text{p1}_\mu\text{p2}_\nu + 4\,\text{gA}^2\text{mf}^2\,\mathfrak{g}_{\mu,\nu} -$

$4\,\text{gV}^2\,\text{mf}^2\mathfrak{g}_{\mu,\nu} - 4\,\text{gA}^2\,\text{p1}.\text{p2}\,\mathfrak{g}_{\mu,\nu} - 4\,\text{gV}^2\,\text{p1}.\text{p2}\,\mathfrak{g}_{\mu,\nu} + 8\,\text{i}\,\text{gA}\,\text{gV}\,\varepsilon_{\mu,\nu,\{\text{p1}\},\{\text{p2}\}}$

This is the SCHOONSCHIP *notation, introduced by Veltman*

The symbol $\varepsilon_{\mu,\nu,\{\text{p1}\},\{\text{p2}\}} = \epsilon^{\mu\nu\alpha\beta}p1_\alpha p2_\beta$ in the output represents the double contraction of the Levi-Civita antisymmetric symbol.

Then, we can contract the result with the Z^0 polarization sum $\big(-g_{\mu\nu} + k_\mu k_\nu/m_Z^2\big)$

In[2]:= **Contract$\big[$Spur[$\boldsymbol{\gamma}$.p2 + mf 1, $\boldsymbol{\gamma}_\mu$, gV 1 – gA $\boldsymbol{\gamma}$5, $\boldsymbol{\gamma}$.p1 – mf 1, gV 1 + gA $\boldsymbol{\gamma}$5, $\boldsymbol{\gamma}_\nu$]**

$\big(-\mathfrak{g}_{\mu,\nu} + k_\mu k_\nu / \text{mZ}^2\big)\big]$

Out[2]= $-4\,d\,\text{gA}^2\,\text{mf}^2 + 4\,d\,\text{gV}^2\,\text{mf}^2 - 8\,\text{gA}^2\,\text{p1}.\text{p2} + 4\,d\,\text{gA}^2\,\text{p1}.\text{p2} -$

$8\,\text{gV}^2\text{p1}.\text{p2} + 4\,d\,\text{gV}^2\,\text{p1}.\text{p2} + \frac{1}{\text{mZ}^2}\big(4\,\text{gA}^2\,\text{mf}^2\,\text{k}.\text{k} - 4\,\text{gV}^2\text{mf}^2\,\text{k}.\text{k} +$

$8\,\text{gA}^2\,\text{k}.\text{p1}\,\text{k}.\text{p2} + 8\,\text{gV}^2\,\text{k}.\text{p1}\,\text{k}.\text{p2} - 4\,\text{gA}^2\,\text{k}.\text{k}\,\text{p1}.\text{p2} - 4\,\text{gV}^2\,\text{k}.\text{k}\,\text{p1}.\text{p2}\big)$

The output contains a number of dot-products of the external momenta. We can give the on shell conditions for the various 4-vectors by first storing them as a list of **Rule**s (infix shortcut **->** which gets auto-replaced with **→**)

In[3]:= **onShell = {k.k → mZ², p1.p1 → mf², p2.p2 → mf²,**

p1.p2 → (mZ² - 2 mf²)/2, k.p1 → mZ²/2, k.p2 → mZ²/2};

and then applying them by using **ReplaceAll** (suffix shortcut **/.**).

Finally, the symbol d (alias: esc dim esc) in the output represents the number of space-time dimensions. For purposes of this tree-level computation we would like to set $d = 4$. In *Package-***X**, the recommended way to achieve this is to apply **LoopRefine** on the output. In the next section, we will use **LoopRefine** frequently to compute one-loop integrals.

So our program to compute the $Z \to f\bar{f}$ matrix element (square) looks like this:

In[3]:= **onShell = {k.k → mZ², p1.p1 → mf², p2.p2 → mf², p1.p2 → (mZ² - 2 mf²)/2,**

k.p1 → mZ²/2, k.p2 → mZ²/2};

In[4]:= **Contract[Spur[γ.p2 + mf 𝟙, γ_μ, gV 𝟙 - gA γ5, γ.p1 - mf 𝟙, gV 𝟙 + gA γ5, γ_ν]**

(-g_{μ,ν} + k_μ k_ν /mZ²)] /. onShell // LoopRefine

Out[4]= $-4 \left(4\, gA^2\, mf^2 - 2\, gV^2\, mf^2 - gA^2\, mZ^2 - gV^2\, mZ^2\right)$

There are two other (but more precarious) ways to set $d = 4$:

- Appending a replacement at the end of the line **/.d → 4**. Be warned that making this replacement at an intermediate stage of computing one-loop integrals can lead to incorrect finite results.

- Declare **d = 4;** . This method is especially dangerous, and should be used only if tree-level computations will be carried out in the current *Mathematica* session. Be sure to **Clear[d]** before using **LoopIntegrate** and **LoopRefine** to compute one-loop integrals!

We can now manipulate this result using *Mathematica*'s built-in algebraic manipulation routines. For example, we can reorganize the output using **Collect**:

In[5]:= **Collect[%, {mf, mZ}]**

Out[5]= $4 \left(-4\, gA^2 + 2\, gV^2\right) mf^2 + 4 \left(gA^2 + gV^2\right) mZ^2$

Therefore, the spin-averaged matrix element is

$$|\overline{\mathcal{M}}|^2 = \frac{e^2}{\sin^2(2\theta_W)} \frac{N_c}{3} 4\left[\left((g_V^f)^2 + (g_A^f)^2\right)m_Z^2 + 2\left((g_V^f)^2 - 2\,(g_A^f)^2\right)m_f^2\right] \qquad (3)$$

from which we can finally deduce the partial width:

$$\Gamma = \frac{\alpha \, m_Z}{\sin^2(2\theta_W)} \left(\frac{N_c}{3}\right) \sqrt{1 - \frac{4m_f^2}{m_Z^2}} \left[\left((g_V^f)^2 + (g_A^f)^2\right) + 2\left((g_V^f)^2 - 2\,(g_A^f)^2\right) \frac{m_f^2}{m_Z^2} \right]. \qquad (4)$$

In section 5.6, I will show you how to arrive at the same result using unitarity of the S-matrix.

5 Tutorial 2: Computing one-loop diagrams

In this section, I provide examples to compute one-loop integrals. The basic procedure of computing the integrals using *Package-X* consists of three steps:

1. Run **LoopIntegrate** to express the integral in terms of scalar coefficient functions. (If needed, projections onto form factors should be made at this stage)

2. Supply on shell and other kinematic conditions by calling **Replace[**$expr$**,**$rules$**]**, (or equivalently $expr$**/.{**$rules$**}**). Never manually make the $d \to 4$ replacement at this stage.

3. Run **LoopRefine** to safely take the $d \to 4$ limit, and to obtain the integral in terms of analytic expressions.
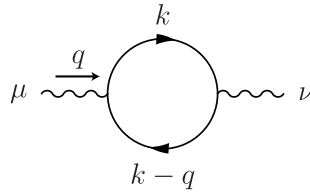
5.1 QED vacuum polarization (**Spur**)

Figure 1: One-loop vacuum polarization in QED

Calling **?LoopIntegrate** *will display its syntax information.*

We will begin with the concrete example of computing the one-loop QED vacuum polarization function $\Pi^{\mu\nu}(q)$. The Feynman diagram along with the momentum routing is shown in Fig. 1. The usage information for **LoopIntegrate** is summarized below:

LoopIntegrate[num, k, {q_0, m_0}, {q_1, m_1}, ...**]** $\left(\dfrac{i\, e^{-\gamma_E \epsilon}}{(4\pi)^{d/2}} \right)^{-1} \mu^{2\epsilon} \displaystyle\int \dfrac{d^d k}{(2\pi)^d} \dfrac{num}{(q_0^2 - m_0^2 + i\varepsilon)(q_1^2 - m_1^2 + i\varepsilon)\cdots}$,

---

Here are some useful points concerning the syntax of **LoopIntegrate**:

- The first argument specifies the numerator, and should be a tensor (k_μ) or scalar-product (*e.g.* $k.p$ or k^2) polynomial in the integration variable.

- The second argument specifies the integration variable k.

- Each argument after the first two must be a **List**, and specifies the denominator factors. Every first element of the list q_i is a linear combination of the integration variable k and external momenta p, and every second element m_i is a mass variable.

It is important to keep in mind how **LoopIntegrate** normalizes the integrals:

> An overall factor $i\, e^{-\gamma_E \epsilon}/(4\pi)^{d/2}$ has been removed from the output of **LoopIntegrate** (and hence also from **LoopRefine**) to maintain readability of the final result. This means that an overall factor of $\frac{i}{16\pi^2}$ is omitted, and the constant $-\gamma_E + \ln(4\pi)$ that accompanies the $1/\epsilon$ poles in UV divergent and mass singular integrals is also missing. *Remember to restore these constants at the end of your computation.*

Start by applying the QED Feynman rules to write down the integral representation of the vacuum polarization function:

$$i\Pi^{\mu\nu}(q) = (-)\mu^{2\epsilon}\int\frac{d^dk}{(2\pi)^d}\text{Tr}\Big[ie\gamma^\nu\frac{i(\slashed{k}+m)}{k^2-m^2}ie\gamma^\mu\frac{i(\slashed{k}-\slashed{q}+m)}{(k-q)^2-m^2}\Big]$$

$$= -e^2\mu^{2\epsilon}\int\frac{d^dk}{(2\pi)^d}\frac{\text{Tr}\big[\gamma^\nu(\slashed{k}+m)\gamma^\mu(\slashed{k}-\slashed{q}+m)\big]}{\big(k^2-m^2\big)\big((k-q)^2-m^2\big)} \quad (5)$$

In the second line, the integral has been brought to a form that is ready for input.

In[1]:= **vacuumPol = LoopIntegrate[Spur[γ$_\nu$, γ.k + m1, γ$_\mu$, γ.(k - q) + m1], k,**
    {k, m}, {k - q, m}]

Out[1]= q$_\mu$q$_\nu$(8 PVB[0, 1, q.q, m, m] + 8 PVB[0, 2, q.q, m, m]) +
    g$_{\mu,\nu}$(-4 PVA[0, m] + 2 q.q PVB[0, 0, q.q, m, m] + 8 PVB[1, 0, q.q, m, m])

Note that the numerator of the integrand is a Dirac trace, so the function **Spur** is used in the first argument of **LoopIntegrate**.

The output is a decomposition of the loop integral in terms of Passarino-Veltman tensor coefficient functions **PVA** and **PVB**. These functions cannot be numerically evaluated, but will be replaced with analytic expressions upon calling **LoopRefine**.

The next step is to supply kinematic conditions. But, since the polarization function is an off shell quantity, there is no information to give; so we'll skip this step. Call **LoopRefine** to obtain the vacuum polarization function:

In[2]:= **LoopRefine[vacuumPol]**

Out[2]= $\Big(-\dfrac{4\,\text{DiscB}[q.q, m, m]\,\big(2\,m^2+q.q\big)}{3\,q.q} - \dfrac{4\,\big(12\,m^2+5\,q.q\big)}{9\,q.q} - \dfrac{4}{3}\Big(\dfrac{1}{\epsilon}+\text{Log}\Big[\dfrac{\mu^2}{m^2}\Big]\Big)\Big)$ q$_\mu$ q$_\nu$ +
    $\Big(\dfrac{4}{3}\text{DiscB}[q.q, m, m]\big(2\,m^2+q.q\big) + \dfrac{4}{9}\big(12\,m^2+5\,q.q\big) + \dfrac{4}{3}q.q\Big(\dfrac{1}{\epsilon}+\text{Log}\Big[\dfrac{\mu^2}{m^2}\Big]\Big)\Big)$ g$_{\mu,\nu}$

__LoopRefine__ always explicitly displays logarithmic UV and/or IR divergences as 1/ϵ poles. If its output is free of ϵ, it is free of these divergences.

Consult the Package-**X** *guide page for more information about* **DiscB**, *and other special functions.*

A few notes concerning this output:

- In *Package*-**X**, UV and IR divergences of loop integrals are regulated by dimensional regularization with $d = 4 - 2\epsilon$. At one-loop, these divergences will appear as 1/$\in$ and 1/$\in^2$ poles near 4 dimensions, and will be accompanied by logarithms of the 't Hooft parameter μ (alias: esc mm esc). In this example, the UV divergence is clearly visible.

- The function **DiscB** is an abbreviation for a more complicated expression that contains the normal threshold discontinuity of the Passarino-Veltman B_0 function. You can call **DiscExpand** to display it in terms of elementary functions. Special simple cases of **DiscB** will be substituted automatically.

- A rank two-tensor can be written as a sum of parts transverse and longitudinal to a given vector v:

$$\Pi^{\mu\nu} = \left(g^{\mu\nu} - \frac{v^\mu v^\nu}{v^2}\right)\Pi_T + \left(\frac{v^\mu v^\nu}{v^2}\right)\Pi_L \tag{6}$$

Applying **Transverse** or **Longitudinal** to a rank-2 tensor will give the corresponding projection. The projection should be made *before* applying any on shell conditions or calling **LoopRefine**.

- Expressions generated by **LoopRefine** are valid for all real external invariants and positive internal masses. When numerically evaluated, the $+i\varepsilon$ prescription is appropriately taken into account and the correct real and imaginary parts are returned.

With these points in mind, let us extract the longitudinal and transverse parts of the vacuum polarization tensor:

In[3]:= **vacuumPol // Longitudinal // LoopRefine**

vacuumPol // Transverse // LoopRefine // DiscExpand

Out[3]= 0

Out[4]= $\frac{4}{9}\left(12\,m^2 + 5\,q.q\right) + \frac{4}{3}\,q.q\left(\frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right]\right) + $

$\dfrac{4\sqrt{q.q\left(-4\,m^2 + q.q\right)}\,\left(2\,m^2 + q.q\right)\,\text{Log}\left[\dfrac{2\,m^2 - q.q + \sqrt{q.q\left(-4\,m^2 + q.q\right)}}{2\,m^2}\right]}{3\,q.q}$

As expected based on the QED Ward identities, the vacuum polarization function has no longitudinal component. We can write down the (bare) vacuum polarization at one-loop:

$$i\Pi^{\mu\nu} = -e^2\frac{i}{16\pi^2}\Big[\text{—computer output—}\Big] \tag{7}$$

$$= \frac{-ie^2}{16\pi^2}\Big[\frac{4}{3}q^2\Big(\frac{1}{\epsilon} - \gamma_E + \ln(4\pi) + \ln\Big(\frac{\mu^2}{m^2}\Big)\Big) + \frac{4}{9}(12m^2 + 5q^2)$$

$$+ \frac{4}{3q^2}\sqrt{q^2(q^2 - 4m^2)}(2m^2 + q^2)\ln\Big(\frac{2m^2 - q^2 + \sqrt{q^2(q^2 - 4m^2)}}{2m^2}\Big)\Big]$$

$$\times\Big(g^{\mu\nu} - \frac{q^\mu q^\nu}{q^2}\Big) \tag{8}$$

5.2 Standard model $H \to gg$ decay width (`Spur`)

In this example, we will compute the partial width of the Higgs boson to two gluons mediated by a top quark loop. The Feynman diagrams along with the momentum routing is shown in Fig. 2. We shall perform the loop integral without anticipating the structure of the final answer based on Ward identities.
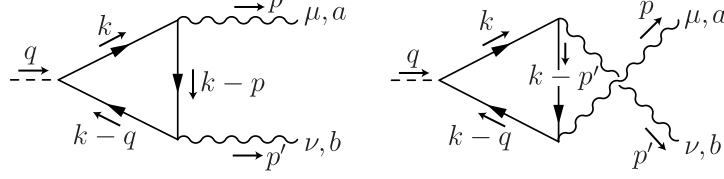
Figure 2: Momentum routing for the one-loop calculation of $H \to gg$ amplitude

Applying the Feynman rules of electroweak interactions, the amplitude is:

$$
i\mathcal{M} = \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} (-1) \mathrm{Tr}_{\substack{\text{Dirac,}\\\text{color}}} \left[\left(\frac{-iy_t}{\sqrt{2}} \right) \frac{i(\slashed{k} - \slashed{q} + m_t)}{(k-q)^2 - m_t^2} \left(-igT^b \gamma^\nu \right) \right.
$$
$$
\left. \times \frac{i(\slashed{k} - \slashed{p} + m_t)}{(k-p)^2 - m_t^2} (-igT^a \gamma^\mu) \frac{i(\slashed{k} + m_t)}{k^2 - m_t^2} \right] \epsilon_\mu^*(\mathbf{p}) \epsilon_\nu^*(\mathbf{p}') + \left(\begin{smallmatrix} p \longleftrightarrow p' \\ \mu \longleftrightarrow \nu \\ a \longleftrightarrow b \end{smallmatrix} \right)
$$

$$
= \frac{-y_t g^2}{2\sqrt{2}} \delta_{\text{color}}^{ab} \epsilon_\mu^*(\mathbf{p}) \epsilon_\nu^*(\mathbf{p}')
$$
$$
\times \left(\mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{\mathrm{Tr}\left[(\slashed{k} - \slashed{q} + m_t)\gamma^\nu (\slashed{k} - \slashed{p} + m_t)\gamma^\mu (\slashed{k} + m_t) \right]}{\left[(k-q)^2 - m_t^2 \right]\left[(k-p)^2 - m_t^2 \right]\left[k^2 - m_t^2 \right]} + \left(\begin{smallmatrix} p \longleftrightarrow p' \\ \mu \longleftrightarrow \nu \end{smallmatrix} \right) \right)
$$

Now we use *Package-**X*** to carry out the integral in round parenthesis. In the code we represent $p =$`p1`, $p' =$`p2` and $m_t =$`m`:

```
In[1]:=  LoopIntegrate[Spur[γ.(k - q) + m1, γᵥ, γ.(k - p1) + m1, γᵤ, γ.k + m1],
           k, {k - q, m}, {k - p1, m}, {k, m}] +
         LoopIntegrate[Spur[γ.(k - q) + m1, γᵤ, γ.(k - p2) + m1, γᵥ, γ.k + m1],
           k, {k - q, m}, {k - p2, m}, {k, m}]

Out[1]=  qᵤqᵥ (8 m PVC[0, 0, 1, p1.p1, p1.p1 - 2 p1.q + q.q, q.q, m, m, m] +
           16 m PVC[0, 0, 2, p1.p1, p1.p1 - 2 p1.q + q.q, q.q, m, m, m]) + ≪ 9 ≫
```

This results in an expression some 20 lines long involving **PVB** and **PVC**. If we apply **LoopRefine** at this stage, we would arrive at a *very complicated* analytic representation of the completely off-shell *Hgg* Green function.

But, by supplying the kinematic conditions relevant for the on shell decay process, we will find a much simpler analytic expression. The kinematic relations for this process are:

Resist the temptation to denote the Higgs mass with a subscripted symbol m_H, as Package-X would erroneously interpret it as a four-vector m with index H.

$p^2 = 0$	`p1.p1 → 0`	$q^\mu = (p + p')^\mu$	`q → p1 + p2`
$p'^2 = 0$	`p2.p2 → 0`	$p \cdot q = \frac{1}{2}m_H^2$	`p1.q → mH^2/2`
$q^2 = m_H^2$	`q.q → mH^2`	$p' \cdot q = \frac{1}{2}m_H^2$	`p2.q → mH^2/2`

There are still two more important conditions we should supply. Remember that external on shell (massive or massless) gauge boson polarization vectors satisfy

---

the transversality condition $p.\epsilon(p) = 0$. For the case at hand, we can impose transversality of the gluon polarization vectors with the following replacements:

$$p.\epsilon(\mathbf{p}) = 0 \quad \Rightarrow \quad \mathbf{p1}_\mu \to 0$$

$$p'.\epsilon(\mathbf{p'}) = 0 \quad \Rightarrow \quad \mathbf{p2}_\nu \to 0$$

After including these eight conditions (which we do in the same input cell), we get a much shorter expression:

In[2]:= `LoopIntegrate[Spur[γ.(k - q) + m𝟙, γᵥ, γ.(k - p1) + m𝟙, γμ, γ.k + m𝟙],`
`  k, {k - q, m}, {k - p1, m}, {k, m}] +`
`  LoopIntegrate[Spur[γ.(k - q) + m𝟙, γμ, γ.(k - p2) + m𝟙, γᵥ, γ.k + m𝟙],`
`  k, {k - q, m}, {k - p2, m}, {k, m}] /.`
`  {p1.p1 → 0, p2.p2 → 0, q.q → mH², q → p1 + p2, p1.q → mH²/2, p2.q → mH²/2} /.`
`  {p1μ → 0, p2ᵥ → 0}`

Out[2]= $2\,\mathrm{p1}_\nu\,\mathrm{p2}_\mu \big(8\,\mathrm{m}\,\mathrm{PVC}[0, 0, 1, 0, 0, \mathrm{mH}^2, \mathrm{m}, \mathrm{m}, \mathrm{m}] + 16\,\mathrm{m}\,\mathrm{PVC}[0, 0, 2, 0, 0, \mathrm{mH}^2, \mathrm{m}, \mathrm{m}, \mathrm{m}]) +$

$2\,\mathrm{p1}_\nu\,\mathrm{p2}_\mu \big(4\,\mathrm{m}\,\mathrm{PVC}[0, 0, 0, 0, 0, \mathrm{mH}^2, \mathrm{m}, \mathrm{m}, \mathrm{m}] +$

$\quad 8\,\mathrm{m}\,\mathrm{PVC}[0, 0, 1, 0, 0, \mathrm{mH}^2, \mathrm{m}, \mathrm{m}, \mathrm{m}] + 16\,\mathrm{m}\,\mathrm{PVC}[0, 1, 1, 0, 0, \mathrm{mH}^2, \mathrm{m}, \mathrm{m}, \mathrm{m}]) +$

$2\,\mathrm{g}_{\mu,\nu} \big(-4\,\mathrm{m}\,\mathrm{PVB}[0, 0, \mathrm{mH}^2, \mathrm{m}, \mathrm{m}] -$

$\quad 2\,\mathrm{m}\,\mathrm{mH}^2\,\mathrm{PVC}[0, 0, 0, 0, 0, \mathrm{mH}^2, \mathrm{m}, \mathrm{m}, \mathrm{m}] + 16\,\mathrm{m}\,\mathrm{PVC}[1, 0, 0, 0, 0, \mathrm{mH}^2, \mathrm{m}, \mathrm{m}, \mathrm{m}])$

The final step of the computation is to call **LoopRefine** (again in the same input cell). Applying **Simplify** should collect and factor out an overall tensor structure that is consistent with the Ward identity on both external gluons:

In[3]:= `LoopIntegrate[Spur[(k - q).γ + m𝟙, γᵥ, (k - p1).γ + m𝟙, γμ, k.γ + m𝟙],`
`  k, {k - q, m}, {k - p1, m}, {k, m}] +`
`  LoopIntegrate[Spur[(k - q).γ + m𝟙, γμ, (k - p2).γ + m𝟙, γᵥ, k.γ + m𝟙],`
`  k, {k - q, m}, {k - p2, m}, {k, m}] /.`
`  {p1.p1 → 0, p2.p2 → 0, q.q → mH², q → p1 + p2, p1.q → mH²/2, p2.q → mH²/2} /.`
`  {p1μ → 0, p2ᵥ → 0} // LoopRefine // Simplify`

Out[3]= $\dfrac{2\,\mathrm{m}\left(4\,\mathrm{mH}^2 + \left(4\,\mathrm{m}^2 - \mathrm{mH}^2\right)\mathrm{Log}\left[\frac{2\mathrm{m}^2 - \mathrm{mH}^2 + \sqrt{-4\mathrm{m}^2\,\mathrm{mH}^2 + \mathrm{mH}^4}}{2\mathrm{m}^2}\right]^2\right)\left(-2\,\mathrm{p1}_\nu\,\mathrm{p2}_\mu + \mathrm{mH}^2\,\mathrm{g}_{\mu,\nu}\right)}{\mathrm{mH}^4}$

Note that a finite result is obtained, independent of the dimensional regulator ϵ and scale μ. With this we may proceed to extract the partial width.

We can parametrize the amplitude as follows:

$$i\mathcal{M} = \left(g^{\mu\nu} - \frac{2p'^\mu p^\nu}{m_H^2}\right)\epsilon_\mu^*(\mathbf{p})\,\epsilon_\nu^*(\mathbf{p'})\,\delta_{\mathrm{color}}^{ab}\,F_{1/2}(m_H, m_t). \tag{9}$$

Here $F_{1/2}(m_H, m_t)$ is the dimensionless loop function, which we can read off the

Mathematica output (remembering the implicit $i/(16\pi^2)$):

$$F_{1/2}(m_H, m_t) = \frac{i}{16\pi^2} \frac{-y_t g^2}{2\sqrt{2}} \times$$

$$\frac{2m_t}{m_H^2}\left[4m_H^2 + \left(4m_t^2 - m_H^2\right)\ln^2\left(\frac{2m_t^2 - m_H^2 + \sqrt{-4m_t^2 m_H^2 + m_H^4}}{2m_t^2}\right)\right]$$

Note that the expression is valid in both domains $m_H > 2m_t$ and $m_H < 2m_t$. As mentioned earlier, the $+i\varepsilon$ prescription is appropriately taken into account by **LoopRefine**. I invite you to make a plot of the real and imaginary parts of the loop function by running the following two lines:

In[4]:=　**f[mH_, m_] = Coefficient[%, g$_{\mu,\nu}$]**

In[5]:=　**Plot[{Re[f[mH, 1]], Im[f[mH, 1]]}, {mH, 0, 5}]**

After squaring the amplitude, summing over the final state gluon spins and colors, and integrating over the gluon momenta, we obtain the partial width:

$$\Gamma_{H\to gg} = \frac{1}{2\pi\, m_H}\left|F_{1/2}(m_H, m_t)\right|^2 . \tag{10}$$

5.3 Electron self-energy function (`DiracMatrix`)

In the next three examples, I will show you how to deal with loop integrals with open fermion lines. Such integrals are analytically cumbersome to work with, and so, I have designed *Package*-**X** to provide a number of ways to approach such integrals:

- Set up the integral and carry out the Dirac algebra by hand, organizing the numerator by structures of Dirac matrices. The coefficients in front of each structure should be in a form that may be input into **LoopIntegrate**. This method is feasible for simple diagrams.

- For more complicated integrals, the Dirac algebra may become too cumbersome to carry out by hand. For two- and three- point integrals, it is possible to use **Projector** to extract the desired form factor.

- Or, use *Package*-**X**'s **FermionLine** (for on shell external fermions) or **DiracMatrix** (for off shell external fermions) objects to compute the integral. This method can largely automate the hand calculations suggested above, but complicated integrals (such as those involving two open fermion lines) may require certain finesse to bring the result into useful forms.
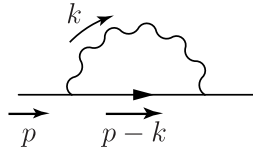

Figure 3: One-loop electron self-energy

We will start with the simpler example of calculating the electron self-energy function in quantum electrodynamics. The Feynman diagram is shown in Figure 3. For this particular example, we will also work in the general class of covariant gauges, where the photon propagator takes the form

$$-i\tilde{D}^{\mu\nu}(k) = \frac{-i}{k^2 + i\varepsilon}\Big[g^{\mu\nu} - (1-\xi)\frac{k^\mu k^\nu}{k^2 + i\varepsilon}\Big] = \frac{-i}{(k^2 + i\varepsilon)^2}\Big[g^{\mu\nu}k^2 - (1-\xi)k^\mu k^\nu\Big].$$

In the second form, the denominator factors have been brought together, so that the one-loop integral representation of the self-energy function can be written more compactly:

$$-i\Sigma(\not{p}) = -e^2\mu^{2\epsilon}\int\frac{d^d k}{(2\pi)^d}\frac{\gamma_\nu(\not{p} - \not{k} + m)\gamma_\mu(g^{\mu\nu}k^2 - (1-\xi)k^\mu k^\nu)}{[(p-k)^2 - m^2](k^2)^2}\,. \tag{11}$$

We will start by working on the numerator, which is a matrix function in spinor space. To enter a product of Dirac matrices without computing its trace, use **DiracMatrix**:

In[1]:= **num = DiracMatrix[γ$_\nu$, γ.(p − k) + m𝟙, γ$_\mu$](g$_{\mu,\nu}$ k.k − (1 − ξ) k$_\mu$ k$_\nu$) // Contract**

Out[1]= −(1 − ξ) DiracMatrix[γ.k, 𝟙m − γ.k + γ.p, γ.k] + DiracMatrix[γ$_\nu$, 𝟙m − γ.k + γ.p, γ$_\nu$]k.k

---

Notice how the syntax of **DiracMatrix** is identical to that of **Spur**. We have also applied **Contract** to contract the tensors into the Dirac matrices. As an aside, we can expand the products of Dirac matrices to bring products such as $k\!\!\!/k\!\!\!/$ together using **FermionLineExpand**:

In[2]:= **num // FermionLineExpand**

Out[2]= DiracMatrix[γ.p] (2 k.k − d k.k + (1 − ξ) k.k) +
          DiracMatrix[] (md k.k + m (−1 + ξ) k.k) +
          DiracMatrix[γ.k]
            (−2 k.k + d k.k + (1 − ξ) k.k + 2 (−1 + ξ) k.p)

TraditionalForm[*expr*] *displays 'expr' in more compact mathematical notation. Try it out!*

The object **DiracMatrix[]** with no arguments stands for the unit Dirac matrix. Note that applying **FermionLineExpand** is not really necessary since **LoopIntegrate** automatically uses an internal version of it to perform the numerator algebra.

Now call **LoopIntegrate** to perform the integral. Notice that in (11) the photon denominator factor k^2 is repeated, *i.e.* raised to the second power. To input this repetition to **LoopIntegrate**, we can supply a third element to the corresponding denominator specifier **{k, 0, 2}** to indicate the power of that denominator factor:

In[3]:= **int = LoopIntegrate[num, k, {k - p, m}, {k, 0, 2}]**

Out[3]= DiracMatrix[]
            (−m PVB[0, 0, p.p, 0, m] + md PVB[0, 0, p.p, 0, m] + m ξ PVB[0, 0, p.p, 0, m]) +
          DiracMatrix[γ.p] (3 PVB[0, 0, p.p, 0, m] − d PVB[0, 0, p.p, 0, m] −
            ξ PVB[0, 0, p.p, 0, m] + 2 PVB[0, 1, p.p, 0, m] −
            d PVB[0, 1, p.p, 0, m] − m² PVB[0, 1, p.p, 0, m, Weights → {2, 1}] +
            m² ξ PVB[0, 1, p.p, 0, m, Weights → {2, 1}] +
            p.p PVB[0, 1, p.p, 0, m, Weights → {2, 1}] −
            ξ p.p PVB[0, 1, p.p, 0, m, Weights → {2, 1}])

Note the appearance of *weighted* Passarino-Veltman **PVB** functions in the output, with the final argument indicating the weight vector $(2, 1)$. Such exotic Passarino-Veltman functions will occur when integrals with repeated propagators are computed. In general, they have better UV behavior (but worse IR behavior) than the corresponding unweighted Passarino-Veltman functions.

Since the self-energy function is an off shell quantity, there are no kinematic relations to include; so now we apply **LoopRefine**:

In[5]:= **LoopRefine[int]**

Out[5]= DiracMatrix[]

$$\left(2\,m\,(2 + \xi) + m\,(3 + \xi)\left(\frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right]\right) + \frac{m\,(3 + \xi)\left(-m^2 + \text{p.p}\right)\text{Log}\left[\frac{m^2}{m^2 - \text{p.p}}\right]}{\text{p.p}} \right) +$$

$$\text{DiracMatrix}[\text{p}.\gamma]\left(-\frac{\xi\left(m^2 + \text{p.p}\right)}{\text{p.p}} - \xi\left(\frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right]\right) - \frac{\xi\left(-m^4 + (\text{p.p})^2\right)\text{Log}\left[\frac{m^2}{m^2 - \text{p.p}}\right]}{(\text{p.p})^2}\right)$$

We can write down the final answer:

$$-i\Sigma(\not{p}) =$$

$$\not{p}\frac{-ie^2}{16\pi^2}\left[-\xi\Big(\frac{1}{\epsilon} - \gamma_{\rm E} + \ln(4\pi) + \ln\Big(\frac{\mu^2}{m^2}\Big)\Big) - \frac{\xi(m^2 + p^2)}{p^2} - \frac{\xi(-m^4 + (p^2)^2)\ln\Big(\frac{m^2}{m^2 - p^2}\Big)}{(p^2)^2}\right]$$

$$+ m\frac{-ie^2}{16\pi^2}\left[(3 + \xi)\Big(\frac{1}{\epsilon} - \gamma_{\rm E} + \ln(4\pi) + \ln\Big(\frac{\mu^2}{m^2}\Big)\Big) + 2(2 + \xi) + \frac{(3 + \xi)(-m^2 + p^2)\ln\Big(\frac{m^2}{m^2 - p^2}\Big)}{p^2}\right]$$

Exercise: The electron self-energy function is gauge dependent. But the position of the pole in the complex p^2 plane of the full propagator $S(\not{p})^{-1} = \not{p} - m - \Sigma(\not{p})$ should be gauge-independent. Compute the one-loop correction to the position of the pole and show that the result is gauge-independent.

5.4 Electron anomalous magnetic moment (`Projector`)

In this example, we will compute the QED contribution to the electron anomalous magnetic moment, which is obtained from the $q^2 \to 0$ limit of the Pauli form factor $F_2(q^2)$. We will perform this calculation by projecting out the appropriate form factor using `Projector`.
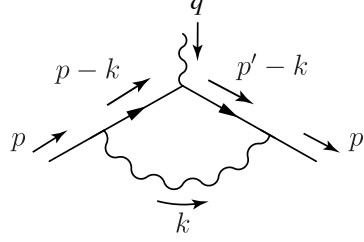

Figure 4: One-loop vacuum polarization in QED

Set up the problem by applying the QED Feynman rules to write down the integral we need to do:

$$\bar{u}(\mathbf{p'})ie\Gamma^\mu u(\mathbf{p})$$
$$= \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \bar{u}(\mathbf{p'}) \Big[ie\gamma^\rho \frac{i(\slashed{p'} - \slashed{k} + m)}{(p'-k)^2 - m^2} ie\gamma^\mu \frac{i(\slashed{p} - \slashed{k} + m)}{(p-k)^2 - m^2} ie\gamma^\nu \Big] u(\mathbf{p}) \frac{-ig_{\rho\nu}}{k^2}$$

$$= e^3 \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \bar{u}(\mathbf{p'}) \Big[\frac{\gamma^\rho(\slashed{p'} - \slashed{k} + m)\gamma^\mu(\slashed{p} - \slashed{k} + m)\gamma^\nu g_{\rho\nu}}{[(p'-k)^2 - m^2][(p-k)^2 - m^2]k^2} \Big] u(\mathbf{p}) \qquad (12)$$

By Lorentz covariance and gauge invariance, we expect the vertex function Γ^μ to take the following form:

$$\bar{u}(\mathbf{p'})\Gamma^\mu u(\mathbf{p}) = \bar{u}(\mathbf{p'}) \Big[F_1(q^2)\gamma^\mu + F_2(q^2)\frac{i}{2m}\sigma^{\mu\nu}q_\nu \Big] u(\mathbf{p})$$

To project out $F_2(q^2)$ from the vertex function, we replace the external spinors u and $\bar{u}$ with the appropriate projector $\mathcal{F}_\mu^{[F2]}(p, p')$, and take the trace:

$$F_2(q^2) = \text{Tr}[\Gamma^\mu \mathcal{F}_\mu^{[F2]}(p, p')]$$

Applying this to both sides of (12), we have

$$ieF_2(q^2) = e^3 \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{\text{Tr}\Big[\gamma^\rho(\slashed{p'} - \slashed{k} + m)\gamma^\mu(\slashed{p} - \slashed{k} + m)\gamma^\nu \mathcal{F}_\mu^{[F2]}(p, p')\Big] g_{\rho\nu}}{[(p'-k)^2 - m^2][(p-k)^2 - m^2]k^2} . \qquad (13)$$

Now we can use *Package*-**X** to compute this integral. We will start with the numerator, with $p =$`p1` and $p' =$`p2`. Although unnecessary at this stage, I have also applied the on shell conditions and simplified the result. Notice, however, that I have not yet made the replacement $q^2 = 0$ for reasons that will soon become clear:

In[1]:= **numerator = Contract[Spur[γ**$_ρ$**, γ.(p2 - k) + m𝟙, γ**$_μ$**, γ.(p1 - k) + m𝟙, γ**$_ν$**,**
Projector["F2", μ][{p1, m}, {p2, m}]] g$_{ρ,ν}$**] /.**
{p1.p1 → m^2**, p2.p2 → m**^{2}**, p1.p2 → -q.q / 2 + m**^{2}**} // Simplify**

Out[1]= $\dfrac{1}{\text{q.q}\left(-4\,\text{m}^2 + \text{q.q}\right)^2}\, 8\,\text{m}^2\,\big(\text{k.k}\,\text{q.q}\left(-4\,\text{m}^2 + \text{q.q}\right) + \text{k.p2}\,\text{q.q}\left(-4\,\text{m}^2 + \text{q.q}\right) +$

$\qquad (\text{k.p1})^2\left(4\,\text{m}^2 + (-2 + d)\,\text{q.q}\right) + (\text{k.p2})^2\left(4\,\text{m}^2 + (-2 + d)\,\text{q.q}\right) +$

$\qquad \text{k.p1}\left(\text{q.q}\left(-4\,\text{m}^2 + \text{q.q}\right) + \text{k.p2}\left(-8\,\text{m}^2 + 2\,d\,\text{q.q}\right)\right)\big)$

Consult the documentation page for **Projector** *for a complete listing of available projectors.*

Note how the function **Projector["F2", μ][{p1, m}, {p2, m}]** is used in the input. The first argument is a String and identifies the projector, and the second argument indicates the vector index. The kinematic information about the external spinors are given in the final arguments; the first momentum and mass always correspond to the unbarred spinor $u(\mathbf{p})$, and the second momentum and mass always correspond to the barred spinor $\bar{u}(\mathbf{p}')$.

Before we proceed, I want to alert you to a potential complication afflicting integrals involving projections. We would eventually like to take the $q^2 \to 0$ limit to obtain the anomalous magnetic moment. But the leading factor $\frac{1}{q^2(-4m^2+q^2)^2}$ in our **numerator** suggests that the integral will be singular at $q^2 = 0$. This singularity is fake (removable): its appearance is an artifact of making a projection, and should disappear upon integrating over the loop momentum k.

I have programmed **LoopIntegrate** to automatically decompose the integral differently than usual when there are kinematic singularities in the numerator so that they may readily be removed algebraically from the output (see the appendix below). With this in mind, we can proceed with computing the integral. Insert **numerator** into **LoopIntegrate**, and replace the scalar products using the on shell conditions:

In[2]:= **LoopIntegrate[numerator, k, {p2 - k, m}, {p1 - k, m}, {k, 0}] /.**
{p1.p1 → m^2**, p2.p2 → m**^{2}**, p1.p2 → -q.q / 2 + m**^{2}**} // Simplify**

Out[2]= $2\,\text{m}^2\,\big(2\,\text{PVC}[0, 0, 1, \text{m}^2, \text{q.q}, \text{m}^2, 0, \text{m}, \text{m}] +$

$\quad (-2 + d)\,\text{PVC}[0, 0, 2, \text{m}^2, \text{q.q}, \text{m}^2, 0, \text{m}, \text{m}] + 2\,\text{PVC}[0, 0, 2, \text{m}^2, \text{q.q}, \text{m}^2, 0, \text{m}, \text{m}] -$

$\quad 4\,\text{PVC}[0, 1, 1, \text{m}^2, \text{q.q}, \text{m}^2, 0, \text{m}, \text{m}] + 2\,d\,\text{PVC}[0, 1, 1, \text{m}^2, \text{q.q}, \text{m}^2, 0, \text{m}, \text{m}] -$

$\quad 2\,\text{PVC}[0, 2, 0, \text{m}^2, \text{q.q}, \text{m}^2, 0, \text{m}, \text{m}] + d\,\text{PVC}[0, 0, 2, \text{m}^2, \text{q.q}, \text{m}^2, 0, \text{m}, \text{m}]\big)$

The final command **Simplify** successfully removes the singularity at $q^2 = 0$. In my experience, *Mathematica*'s algebraic functions **Cancel**, **Factor**, and **FactorSquareFree** all remove the singularity from the output, and may provide faster alternatives.

The replacement **q.q → 0** can now be made immediately.

In[4]:= `% /. q.q → 0 // LoopRefine`

Out[4]= 2

Remembering that there is an implicit $\frac{i}{16\pi^2}$, eqn (13) becomes

$$ieF_2(0) = e^3 \frac{i}{16\pi^2}\Big[\text{—computer output—} \Big]$$

$$= e^3 \frac{i}{16\pi^2} \times 2\,,$$

$$\text{or,} \qquad F_2(0) = \frac{e^2}{8\pi^2} = \frac{\alpha_{\text{em}}}{2\pi}\,. \tag{14}$$

Exercise: Return to the previous example (Electron self-energy function), and use **Projector** to separately extract the coefficients of $\not{p}$ and $\mathbb{1}m$ from $i\Sigma(\not{p})$.

Appendix: Notes on the internals of **LoopIntegrate**

Usually, **LoopIntegrate** tries to cancel scalar products involving the integration variable k common to the numerator and the denominator, controlled by the option **Cancel**, before expressing the result in terms of Passarino-Veltman functions. For example,

$$\mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{k^2}{[k^2 - m^2][(k + p)^2 - m^2]}$$

$$= \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{(k^2 - m^2) + m^2}{[k^2 - m^2][(k + p)^2 - m^2]}$$

$$= \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{1}{(k + p)^2 - m^2} + \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{m^2}{[k^2 - m^2][(k + p)^2 - m^2]}$$

$$\longrightarrow A_0(m) + m^2 B_0(p^2; m, m).$$

Under most circumstances, this leads to substantially increased performance and quality of the output of **LoopRefine**. However this behavior causes removable kinematic singularities generated by projectors to persist. Therefore, with the default setting **Cancel → Automatic**, if the numerator contains kinematic singularities, **LoopIntegrate** will not cancel these scalar products so that it is easier to remove these kinematic singularities.

Exercise: Investigate what happens when the option **Cancel → True** is set to **LoopIntegrate** when trying to make a projection. Observe that it is not possible to make the replacement **q.q → 0** without generating Power::infy errors. Try using **LoopRefineSeries** on the output of **LoopIntegrate** to obtain the leading $q^2 \to 0$ behavior of the integral (feel free to consult the help page for **LoopRefineSeries**).

5.5 One-loop QCD correction to $\gamma^* \to q\bar{q}$ (**FermionLine**, u, v)

In this section, we will compute the one-loop QCD correction to the QED vertex function $\gamma^* \to q\bar{q}$. This calculation is the virtual component of the fully inclusive cross section for the process $e^+e^- \to$ hadrons at $O(\alpha_s)$ shown below.
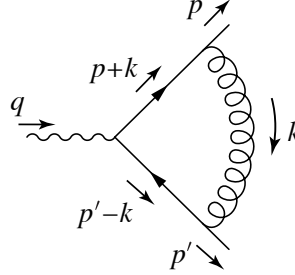

Figure 5: Virtual QCD correction to $\gamma^* \to q\bar{q}$

We will approximate the quarks (electric charge Q_q) as being massless, and compute the vertex function in the Feynman gauge:

$$-iQ_q e\bar{u}(\mathbf{p}')\Gamma^\mu u(\mathbf{p})$$

$$= \mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \bar{u}(\mathbf{p})\Big[(-ig\gamma^\nu T^a)\frac{i(\slashed{p}+\slashed{k})}{(p+k)^2}(-iQ_q e\gamma^\mu)\frac{i(-\slashed{p}'+\slashed{k})}{(-p'+k)^2}(-ig\gamma^\rho T^b)\Big]v(\mathbf{p}')\frac{-i\delta^{ab}g_{\nu\rho}}{k^2}$$

$$= -g^2 C_F Q_q e\mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{\bar{u}(\mathbf{p})\gamma^\nu(\slashed{p}+\slashed{k})\gamma^\mu(-\slashed{p}'+\slashed{k})\gamma^\rho v(\mathbf{p}')\,g_{\nu\rho}}{(p+k)^2(-p'+k)^2 k^2} \tag{15}$$

In the final line, the color factors have been collected $T^a T^b \delta^{ab} = C_F$, and the integral is put in a form that is ready for input.

In *Package*-**X**, the product of Dirac matrices sandwiched between on shell spinors is entered using the **FermionLine** object like this (letting $p =$**p1** and $p' =$**p2**):

In[1]:= **num = ⟨u[p1, 0], γ$_\nu$, γ.(p1 + k), γ$_\mu$, γ.(-p2 + k), γ$_\rho$, v[p2, 0]⟩ g$_{\nu,\rho}$ //Contract**

Out[1]= ⟨u[p1, 0], γ$_\rho$, γ.k + γ.p1, γ$_\mu$, γ.k - γ.p2, γ$_\rho$, v[p2, 0]⟩

Additional input aliases listed in Table 3 have been defined for rapid notebook entry of the on shell spinors at the ends of the string of Dirac matrices. Notice that **u[p, m]** and **v[p, m]** take two mandatory arguments which specify the on shell momentum and mass. The syntax for the intervening Dirac matrices are the same as those for **Spur** and **DiracMatrix**. *Package*-**X** will automatically parse this input as a **FermionLine** object.

Spinor	Alias	StandardForm
$\bar{u}(\mathbf{p})$	esc ub esc	⟨u[p, m]
$\bar{v}(\mathbf{p})$	esc vb esc	⟨v[p, m]
$u(\mathbf{p})$	esc u esc	u[p, m]⟩
$v(\mathbf{p})$	esc v esc	v[p, m]⟩

Table 3: Inputting u and v spinors

As an aside, we can apply **FermionLineExpand** to expand out the product of Dirac matrices:

In[2]:= **num // FermionLineExpand**

Out[2]= $\langle u[\text{p1}, 0], \gamma_\mu, v[\text{p2}, 0]\rangle$

$(-2\,\text{k}.\text{k} + d\,\text{k}.\text{k} + 4\,\text{k}.\text{p1} - 4\,\text{k}.\text{p2} - 4\,\text{p1}.\text{p2}) +$

$(4 - 2d)\,\langle u[\text{p1}, 0], \gamma.\text{k}, v[\text{p2}, 0]\rangle\,\text{k}_\mu -$

$4\,\langle u[\text{p1}, 0], \gamma.\text{k}, v[\text{p2}, 0]\rangle\,\text{p1}_\mu +$

$4\,\langle u[\text{p1}, 0], \gamma.\text{k}, v[\text{p2}, 0]\rangle\,\text{p2}_\nu$

Consult the documentation page for further details on **FermionLineExpand**.

In addition to using the Dirac algebra, **FermionLineExpand** will apply the spinor equations of motion

$$\not{p}\,u(\mathbf{p}) = m\,u(\mathbf{p}), \qquad \not{p}\,v(\mathbf{p}) = -m\,v(\mathbf{p}),$$

express products of Dirac matrices in canonical SVTAP form

$$\gamma^{\mu_1}\gamma^{\mu_2}\gamma^{\mu_3}\cdots = A^{\mu_1\mu_2\mu_3\cdots}\mathbb{1} + (B^{\mu_1\mu_2\mu_3\cdots})_\nu\gamma^\nu + (C^{\mu_1\mu_2\mu_3\cdots})_{\nu\rho}\sigma^{\nu\rho}$$
$$+ (D^{\mu_1\mu_2\mu_3\cdots})_\nu\gamma^\nu\gamma_5 + E^{\mu_1\mu_2\mu_3\cdots}\gamma_5,$$

and if possible, apply the Gordon identities

$$\bar{u}(\mathbf{p}')(p' + p)^\mu u(\mathbf{p}) = 2m\,\bar{u}(\mathbf{p}')\gamma^\mu u(\mathbf{p}) - i\,\bar{u}(\mathbf{p}')\sigma^{\mu\nu}(p' - p)_\nu u(\mathbf{p}). \qquad (16)$$

These manipulations are automatically carried out by **LoopIntegrate** when **FermionLine** appears in the numerator:

In[3]:= **integral = LoopIntegrate[num, k, {p1 + k, 0}, {-p2 + k, 0}, {k, 0}] /.**

{p1.p1 → 0, p2.p2 → 0, p1.p2 → q.q / 2}

Out[3]= $\langle u[\text{p1}, 0], \gamma_\mu, v[\text{p2}, 0]\rangle\,(4\,\text{PVB}[0, 0, 0, 0, 0] - 6\,\text{PVB}[0, 0, \text{q}.\text{q}, 0, 0] +$

$d\,\text{PVB}[0, 0, \text{q}.\text{q}, 0, 0] - 2\,\text{q}.\text{q}\,\text{PVC}[0, 0, 0, 0, \text{q}.\text{q}, 0, 0, 0, 0] +$

$4\,\text{PVC}[1, 0, 0, 0, \text{q}.\text{q}, 0, 0, 0, 0] - 2\,d\,\text{PVC}[1, 0, 0, 0, \text{q}.\text{q}, 0, 0, 0, 0])$

After applying the on shell conditions, call **LoopRefine** to obtain an analytic expression for this integral:

In[3]:= **LoopRefine[integral]**

Out[3]= $\langle u[\text{p1}, 0], \gamma_\mu, v[\text{p2}, 0]\rangle$

$$\left(-8 + \frac{\pi^2}{6} - 3\left(\frac{1}{\epsilon} + \text{Log}\left[-\frac{\mu^2}{\text{q}.\text{q}}\right]\right) - 2\left(\frac{1}{\epsilon^2} + \frac{\text{Log}\left[-\frac{\mu^2}{\text{q}.\text{q}}\right]}{\epsilon} + \frac{1}{2}\,\text{Log}\left[-\frac{\mu^2}{\text{q}.\text{q}}\right]^2\right)\right)$$

From this output, we can write down the result for the virtual QCD correction to $\gamma^* \to \bar{q}q$:

$$-iQ_q e\bar{u}(\mathbf{p}')\Gamma^\mu u(\mathbf{p}) = \bar{u}(\mathbf{p}')\gamma^\mu u(\mathbf{p})\frac{ig^2 C_F Q_q e}{16\pi^2}\left\{-8+\frac{\pi^2}{6}-3\left(\frac{1}{\epsilon}-\gamma_{\mathrm{E}}+\ln(4\pi)+\ln\left(\frac{\mu^2}{-q^2}\right)\right)\right.$$
$$-2\left[\frac{1}{\epsilon^2}+\frac{1}{\epsilon}\left(-\gamma_{\mathrm{E}}+\ln(4\pi)\right)+\frac{\gamma_{\mathrm{E}}^2}{2}-\gamma_{\mathrm{E}}\ln(4\pi)+\frac{1}{2}\ln^2(4\pi)\right.$$
$$\left.\left.+\left(\frac{1}{\epsilon}-\gamma_{\mathrm{E}}+\ln(4\pi)\right)\ln\left(\frac{\mu^2}{-q^2}\right)+\frac{1}{2}\ln^2\left(\frac{\mu^2}{-q^2}\right)\right]\right\}$$

Exercise: The $1/\epsilon$ and $1/\epsilon^2$ poles in the output above arise from both UV and IR divergences. Set the option **Part → UVDivergent** or **Part → IRDivergent** to **LoopRefine** to compute these parts of the integral separately.

Exercise: Compute the vertex function retaining the final state quark masses in the general covariant ξ gauge. Separate the $1/\epsilon$ pole based on their UV or IR origin.

Exercise: Return to the previous example (Electron anomalous magnetic moment) and use the **FermionLine** object to compute the entire QED vertex function $\Gamma^\mu(q^2)$ for massive electrons. Notice that both form factors $F_1(q^2)$ and $F_2(q^2)$ are simultaneously obtained in this way.

5.6 Tree-level $Z^0 \to f\bar{f}$ decay width (`Discontinuity`)

In this section we will re-derive the tree-level $Z^0 \to f\bar{f}$ partial width (4) by calculating the imaginary part of a one-loop integral. The basic idea is that by S-matrix unitarity, the discontinuity across the normal threshold cut (twice the imaginary part) of a one loop integral gives the square of the lower order amplitude, with the phase space integrations already done. Pictorially, this corresponds to cutting the internal lines of a Feynman diagram:

After dividing the discontinuity by the flux factor $2m_Z$ for the decay process, the partial width is obtained:

$$\Gamma = \frac{-i}{\text{Flux}} \text{Disc}\,\mathcal{M}. \tag{17}$$

Using the Feynman rules for electroweak theory, the one-loop amplitude is

$$i\mathcal{M} = (-)\mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} N_c \text{Tr}\Big[\frac{-ie}{\sin^2(2\theta_W)} \gamma^\nu (g_V^f - g_A^f \gamma_5) \frac{i(\slashed{k} + m_f)}{k^2 - m_f^2}$$
$$\times \frac{-ie}{\sin^2(2\theta_W)} \gamma^\mu (g_V^f - g_A^f \gamma_5) \frac{i(\slashed{k} - \slashed{q} + m_f)}{(k-q)^2 - m_f^2} \Big] \epsilon_\nu^*(\mathbf{q}) \epsilon_\mu(\mathbf{q}),$$

or after pulling out the overall constants,

$$i\mathcal{M} = -\frac{e^2 N_c}{\sin^2(2\theta_W)} \epsilon_\nu^*(\mathbf{q}) \epsilon_\mu(\mathbf{q})$$
$$\mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{\text{Tr}\big[\gamma^\nu (g_V^f - g_A^f \gamma_5)(\slashed{k} + m_f)\gamma^\mu (g_V^f - g_A^f \gamma_5)(\slashed{k} - \slashed{q} + m_f)\big]}{\big(k^2 - m_f^2\big)\big((k-q)^2 - m_f^2\big)} \tag{18}$$

Input the one-loop integral, setting aside the prefactor in the first line, and apply the on shell condition $q^2 = m_Z^2$:

In[1]:= **LoopIntegrate[Spur[γ_ν, gV 1 – gA γ5, k.γ + mf 1, γ_μ, gV 1 – gA γ5, (k – q).γ + mf 1],**

k, {k, mf}, {k – q, mf}] /. q.q → mZ^2

Out[1]= $q_\mu q_\nu \Big((8\,\text{gA}^2 + 8\,\text{gV}^2)\,\text{PVB}[0, 1, \text{mZ}^2, \text{mf}, \text{mf}] + (8\,\text{gA}^2 + 8\,\text{gV}^2)\,\text{PVB}[0, 2, \text{mZ}^2, \text{mf}, \text{mf}] \Big) +$

$g_{\mu,\nu} \Big((-4\,\text{gA}^2 - 4\,\text{gV}^2)\,\text{PVA}[0, \text{mf}] + (-8\,\text{gA}^2\,\text{mf}^2 + 2\,\text{gA}^2\,\text{mZ}^2 + 2\,\text{gV}^2\,\text{mZ}^2)$

$\text{PVB}[0, 0, \text{mZ}^2, \text{mf}, \text{mf}] + (8\,\text{gA}^2 + 8\,\text{gV}^2)\,\text{PVB}[1, 0, \text{mZ}^2, \text{mf}, \text{mf}] \Big)$

Now, we would like to obtain the discontinuity across the normal threshold cut of this expression. Apply **LoopRefine** to the expression, but to limit the output to just the discontinuity set the option **Part → Discontinuity[mZ^2]**:

In[2]:= **disc = LoopRefine[%, Part → Discontinuity[mZ²], Organization → Function]**

Out[2]= $2 \, \mathbb{i} \, \pi \, \text{HeavisideTheta}[-4 \, \text{mf}^2 + \text{mZ}^2]$

$$\left(- \frac{4 \left(\text{gA}^2 + \text{gV}^2 \right) \sqrt{\text{mZ}^2 \left(-4 \, \text{mf}^2 + \text{mZ}^2 \right)} \left(2 \, \text{mf}^2 + \text{mZ}^2 \right) q_\mu q_\nu}{3 \, \text{mZ}^4} + \right.$$

$$\left. \frac{4 \sqrt{\text{mZ}^2 \left(-4 \, \text{mf}^2 + \text{mZ}^2 \right)} \left(-4 \, \text{gA}^2 \, \text{mf}^2 + 2 \, \text{gV}^2 \, \text{mf}^2 + \text{gA}^2 \, \text{mZ}^2 + \text{gV}^2 \, \text{mZ}^2 \right) g_{\mu,\nu}}{3 \, \text{mZ}^2} \right)$$

Notice that the argument of **Discontinuity** indicates the channel variable across which the discontinuity should be taken. In this example, a self-energy type integral, there is only one channel $q^2 = m_Z^2$. But problems involving more external lines feature several channel variables.

This result is essentially the square of the tree-level matrix element with the phase space integrals already completed. Notice that the **HeavisideTheta** function has support only when the channel is open, $m_Z^2 \geq 4m_f^2$.

The final step in obtaining the partial width is to perform the polarization average of the initial Z^0 boson. Contract the result with the completeness relation $\left(-g_{\mu\nu} + k_\mu k_\nu / m_Z^2 \right)$, and divide by 3 to account for the averaging. Finally, make the replacement $q^2 = m_Z^2$ and apply **LoopRefine** to take the limit $d \to 4$ and also to organize the result:

In[2]:= **Contract $\left[\text{disc} * \frac{1}{3} \left(-g_{\mu,\nu} + q_\mu q_\nu / mZ^2 \right) \right]$ /. q.q → mZ² // LoopRefine**

Out[2]= $-\dfrac{1}{3 \, \text{mZ}^2} 8 \, \mathbb{i} \, \sqrt{\text{mZ}^2 \left(-4 \, \text{mf}^2 + \text{mZ}^2 \right)}$

$\left(-4 \, \text{gA}^2 \, \text{mf}^2 + 2 \, \text{gV}^2 \, \text{mf}^2 + \text{gA}^2 \, \text{mZ}^2 + \text{gV}^2 \, \text{mZ}^2 \right) \pi \, \text{HeavisideTheta}[-4 \, \text{mf}^2 + \text{mZ}^2]$

It is now a matter of restoring the previously omitted prefactors from (18), (17) and the implicit normalization $\frac{1}{16\pi^2}$ (the i in $\frac{i}{16\pi^2}$ cancels with that in front of $i\mathcal{M}$) to arrive at the final result,

$$\Gamma = -\frac{e^2 N_c}{\sin^2(2\theta_W)} \frac{1}{16\pi^2} \frac{-i}{2m_Z} \Big[\text{---computer output---} \Big]$$

$$= \frac{\alpha \, m_Z}{\sin^2(2\theta_W)} \left(\frac{N_c}{3} \right) \sqrt{1 - \frac{4m_f^2}{m_Z^2}} \Big[\left((g_V^f)^2 + (g_A^f)^2 \right) + 2 \left((g_V^f)^2 - 2 \, (g_A^f)^2 \right) \frac{m_f^2}{m_Z^2} \Big],$$

reproducing eq. (4) at the beginning of this primer.

For another, more involved example of an application of **Discontinuity**, see the tutorial "Tree-Level $e^+ e^- \to \gamma\gamma$ Cross Section by The Optical Theorem" near the bottom of the guide page.

6 Tips for a pleasant computing experience

The documentation pages available within the *Wolfram Documentation Center* provide a lot of details on each symbol defined by *Package-***X**. Please browse through them to familiarize yourself with the available bells and whistles since they may come in handy. In this section, I've gathered supplementary tips and pointers to make your computing experience as pleasant as possible.

Refrain from using special formatting for symbols (like m_H and p')

This is a tip not only for *Package-***X** users, but also for *Mathematica* users in general. Beginners will try to use *Mathematica*'s formatting capabilities to write code using subscripted variables like m_H or primed variables like p'. These are not interpreted by *Mathematica* as symbols, but rather as expressions:

n00b input	internal rep.		better input	internal rep.
m_H	`LTensor[m,H]` in *Package-***X**		**mH**	mH
p'	`Derivative[1][p]`		**p′**	p′

This can cause unexpected behavior, for example, in *Package-***X**

In[1]:= **m_H² // Contract**

Out[1]= m.m

Get in the habit of using unformatted symbols for your variables, opting instead for **mH** and **p′** (alias: ′ = esc ' esc).

Reuse and store expressions generated by **LoopRefine**

When working on extensions of the Standard Model, it could happen that a basic integral can be used to determine contributions from several different particle species. In this case, you can store the general formula as a function of multiple parameters, and then reuse it for the various particles the loop.

For example, you can obtain the general formula for the transverse part of the W-boson vacuum polarization function at zero external momentum $\Pi_\mathrm{T}^{WW}(0)$ as a function of internal fermion masses m_1 and m_2:

In[1]:= **int =**

Transverse$\Big[$LoopIntegrate$\Big[$Spur$\Big[\dfrac{-\mathbb{i}\,g}{\sqrt{2}}\gamma_\mu$, PL,

$\mathbb{i}\,(k.\gamma + m1\,\mathbb{1})$, $\dfrac{-\mathbb{i}\,g}{\sqrt{2}}\gamma_\nu$, PL, $\mathbb{i}\,((k+p).\gamma + m2\,\mathbb{1})\Big]$, k,

$\{k, m1\}, \{k+p, m2\}\Big]\Big] / . p.p \to 0$;

Then store the output of **LoopRefine** as $f[m_1, m_2]$:

In[2]:= **f[m1_, m2_] = LoopRefine[int]**

Out[2]= $-\dfrac{1}{4}\,g^2\left(m1^2 + m2^2\right) + \dfrac{g^2\,m1^4\,\mathrm{Log}\left[\frac{m1^2}{m2^2}\right]}{2\left(m1^2 - m2^2\right)} -$

$\dfrac{1}{2}\,g^2\left(m1^2 + m2^2\right)\left(\dfrac{1}{\epsilon} + \mathrm{Log}\left[\dfrac{\mu^2}{m2^2}\right]\right)$

Notice the use of **Set** (**=**) instead of **SetDelayed** (**:=**) so that the right hand side is evaluated before storing the expression into **f**. You may add a semicolon (**;**) at the end of the line to suppress the output.

If you have a massless particle in the loop (say, $m_2 = 0$), this formula will lead to a ln(0) error. It's easy enough to define a special case for **f**:

In[3]:=   **f[m1_, 0] = LoopRefine[int /. m2 → 0]**

Out[3]= $-\dfrac{1}{4} g^2 m1^2 - \dfrac{1}{2} g^2 m1^2 \left(\dfrac{1}{\epsilon} + \text{Log}\left[\dfrac{\mu^2}{m1^2} \right] \right)$

(*Exercise*: How would you define a special case when $m_1 = m_2$?) Then you can add up the fermionic contributions to $\Pi_T^{WW}(0)$, assuming massless neutrinos, without making multiple calls to **LoopRefine**:

In[4]:=   **3 f[mt, mb] + 3 f[mc, ms] + 3 f[mu, md] +**
           f[mτ, 0] + f[mμ, 0] + f[me, 0]

Apply **LoopRefine** to special sets of Feynman integrals

> In gauge theories, several different topologies of loop integrals usually contribute to a single process (or Green's function), which combine neatly after several cancellations. If you expect such simplifications or cancellations, it may be helpful to apply **LoopRefine** to the sum of **LoopIntegrate** output of the individual integrals.

Pay attention when using **γ5**

> In *Package*-**X**, γ_5 is defined to anticommute with all Dirac matrices in $d = 4 - 2\epsilon$ dimensions. You should be aware that this implementation of γ_5 may lead to algebraic inconsistencies starting at $O(\epsilon)$, which means that finite parts of integrals exhibiting $1/\epsilon$ poles might be incorrect. Here are some examples and suggested solutions to resolve this inconsistency within *Package*-**X**:
>
> - Triangle integrals involving a closed fermion loop with two or three chiral vector vertices are susceptible to inconsistencies.
>
>   *Recommended solution*: Enforce the necessary Ward identities using Adler's method. See the tutorial "Ward Identities and γ5 in Dimensional Regularization" listed near the bottom of the guide page in the documentation center for more information.
>
> - Box integrals involving fermions and internal gauge bosons in the unitary gauge contain UV divergent $1/\epsilon$ poles at intermediate stages. Regardless of whether these poles cancel in the end, finite parts of these integrals are also susceptible to these inconsistencies.
>
>   *Recommended solution*: Perform the computation in a renormalizable gauge such as 't Hooft-Feynman gauge. Note that Landau gauge reintroduces the problem due to mass singular $1/\epsilon$ poles.
>
> Although normally UV divergent, it is unusual for tadpole and bubble integrals to exhibit these inconsistencies due to the lower number of external invariants upon which they depend. For maximum control, you can set the global variable:
>
> In[4]:=   **$DiracAlgebra = False;**
>
> This turns off all Dirac algebraic routines. Then the output of **LoopIntegrate** will be given in terms of unevaluated products of gamma matrices. It will be more verbose, but you can now manipulate the gamma matrices by hand.

Try different evaluation strategies to resolve numerical instabilities

It is quite common to encounter loop integrals with widely separated scales. In these cases, machine precision evaluation usually leads to numerical instabilities. Consider for example this expression obtained after applying **LoopRefine** to an (unspecified) integral:

In[1]:= **f[s_, t_, m_] = LoopRefine[**$example$**, ExplicitC0 → None]**

Out[1]= $-\dfrac{s\,\mathrm{DiscB}[s,m,m]}{s-t} - \dfrac{s\,\mathrm{DiscB}[s,m,m]}{s-t} -$
$2\,m^2\,\mathrm{ScalarC0}[0,s,t,m,m,m]$

For very large **m**, the default machine precision evaluation exhibits numerical instabilities:

In[2]:= **LogLinearPlot[Re[f[21.5, 1.5, m]], {m, 5, 10^8},**
PlotRange → {2.5, 3.5}]

Out[2]=
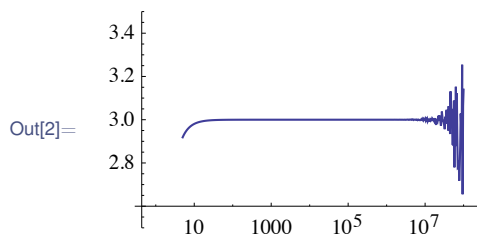
Since all of *Package-***X**'s special functions support *Mathematica*'s arbitrary-precision arithmetic, you can use it (at the expense of computation speed) to achieve numerical stability. Here we supply numbers with a precision level of 20:

In[3]:= **LogLinearPlot[Re[f[21.5`20, 1.5`20, m]], {m, 5, 10^8},**
PlotRange → {2.5, 3.5}, WorkingPrecision → 20]

Out[3]=
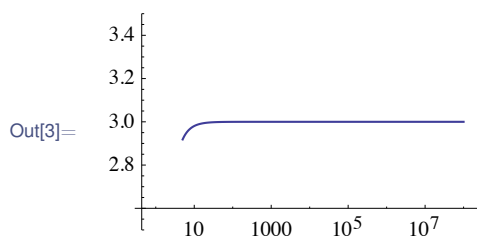
Alternatively, you could obtain an analytic approximation to the integral by using **LoopRefineSeries**:

In[4]:= **fApprox[s_, t_, m_] =**
LoopRefineSeries[$example$**, {s, 0, 1}, {t, 0, 1}] // Normal**

Out[4]= $3 - \dfrac{t}{12\,m^2} + s\left(-\dfrac{t}{12\,m^2} - \dfrac{t}{180\,m^4}\right)$

A plot of this expression using machine precision yields a smooth curve visually identical to the one obtained by using arbitrary-precision above.

Link *Package-X* to third party numerical libraries

> *Package-X* is optimized to give analytic expressions when a compact one is expected. For NLO cross sections with full kinematic dependence, it is known that the formulae are usually not compact. If you insist on using *Package-X* to compute these complicated quantities, I suggest linking *Package-X* to numerical libraries.
>
> - LOOPTOOLS (http://www.feynarts.de/looptools/) by T. Hahn provides the most readily accessible numerical library that can be called within *Mathematica*, but is limited to the lower rank Passarino-Veltman functions that occur in renormalizable gauge theories. See the tutorial "Linking Package-X to LoopTools" listed near the bottom of the guide page in the documentation center for more information.
>
> - COLLIER (https://collier.hepforge.org) by A. Denner, S. Dittmaier, and L. Hofer is a more recent library that is capable of numerically evaluating any Passarino-Veltman coefficient function and features numerous algorithms to avoid numerical instabilities. COLLIERLINK is a recently released expansion pack for *Package-X* which provides an interface to the COLLIER library. See the tutorial "Linking Package-X to COLLIER" for more information.
>
> In both cases, using these library in conjunction with *Package-X* will allow you to directly evaluate the Passarino-Veltman functions **PVA**, **PVB**, **PVC**, **PVD**, without needing to call **LoopRefine**.

Cross check results within *Package-X*

> It is usually possible to obtain equivalent results in several different ways using *Package-X*. This redundancy can be used to cross check your calculation, and to isolate any internal inconsistencies/bugs within *Package-X*. I mention a few here that you may find useful.
>
> For cross-checking analytical integrations, expressions generated by **LoopIntegrate** by toggling the option **Cancel** to **True** and **False** will contain different combinations of Passarino-Veltman functions. Furthermore, if your integrals contain linearly dependent denominator factors (*e.g.* with one or more vanishing external momenta), toggling the option **Apart** to **LoopIntegrate** will also generate different combinations of Passarino-Veltman functions. In all cases, the result of applying **LoopRefine** should *always* generate equivalent results, and can be used for checking the internal consistency of *Package-X*.
>
> For numerical cross-checks of special functions like **ScalarC0**, **ScalarD0**, *etc.*, it is helpful to know that the numerical implementation and the analytic expression generated by **C0Expand** and **D0Expand** are of different origin. Evaluating the analytic representations numerically should return the same result as evaluating the special functions directly.
>
> When projecting form factors with **Projector**, you should be able to read off the same form factors by performing the integral with a **FermionLine** object in the numerator.

A   Space-time conventions

For reference, the conventions for space-time quantities are summarized in Table 4 below.

Quantity	Convention
Metric signature	$g_{\mu\nu} = \text{diag}(+, -, -, -)$
Spacetime dimension	$d = 4 - 2\epsilon$
Dirac matrix commutator	$\sigma_{\mu\nu} = \frac{i}{2}[\gamma_\mu, \gamma_\nu]$
Fifth gamma matrix	$\gamma_5 = i\gamma^0\gamma^1\gamma^2\gamma^3$
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 4: Conventions for space-time quantities