

# Mastering One Loop Feynman Integrals with Package-X

*Hiren Patel*

## Introduction

Package-X is a *Mathematica* package for analytic computations of one loop Feynman integrals. (<http://packagex.hepforge.org>)

**Needs["X`"] (\*requires Package-X 2.0\*)**

$$e.g. \quad \mu^2 \epsilon \int \frac{d^d k}{(2\pi)^d} \frac{k_\mu k_\nu}{[(k+p)^2 - m^2 + i\epsilon][k^2 - m^2 + i\epsilon]}$$

Compute one-loop integrals in three steps.

1. Initiate calculation with **LoopIntegrate**:

```
In[9]:= LoopIntegrate[k_\mu k_\nu, k, {k + p, m}, {k, M}]
```

```
Out[9]:= p_\mu p_\nu PVB[0, 2, p.p, M, m] + g_{\mu,\nu} PVB[1, 0, p.p, M, m]
```

2. Provide on-shell conditions:

```
In[10]:= % /. p.p -> 0
```

```
Out[10]:= p_\mu p_\nu PVB[0, 2, 0, M, m] + g_{\mu,\nu} PVB[1, 0, 0, M, m]
```

3. Apply **LoopRefine**:

```
In[11]:= LoopRefine[%]
```

$$\text{Out[11]} = \left( \frac{2 m^4 - 7 m^2 M^2 + 11 M^4}{18 (m^2 - M^2)^2} + \frac{M^6 \text{Log}\left[\frac{m^2}{M^2}\right]}{3 (-m^2 + M^2)^3} + \frac{1}{3} \left( \frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right] \right) \right) p_\mu p_\nu + \left( \frac{3}{8} (m^2 + M^2) + \frac{M^4 \text{Log}\left[\frac{m^2}{M^2}\right]}{4 (-m^2 + M^2)} + \frac{1}{4} (m^2 + M^2) \left( \frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right] \right) \right) g_{\mu,\nu}$$

## What *really* is Package-X?

Package-X is a general purpose *Mathematica* package to compute relativistic one-loop Feynman integrals in dimensional regularization analytically for any kinematic configuration.

Focus #1: on all applications except fully differential NLO cross section.

- ◆ Shift in vacuum expectation value (tadpoles)
- ◆ (Pole mass)–(mass parameter) relationship
- ◆ Wavefunction renormalization
- ◆ Electroweak oblique parameters
- ◆ Particle electromagnetic multipole moments (EDM/MDM)
- ◆ 2 body decays
- ◆ Cross sections at threshold (DM annihilation/direct detection)
- ◆ UV-divergences (counterterms)
- ◆ Wilson coefficients ...

## Motivation

No comprehensive one-loop package available for **analytic expressions** that can also be **numerically evaluated** at **any kinematic point**.

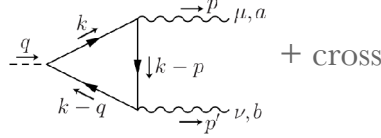
There are analytic packages (like FeynCalc, ...), and numerical packages (like LoopTools, ...)

But...

- ◆ Somewhat complicated to use
- ◆ Does not always get an answer, like at vanishing Gram determinant, etc...  
(except numerical packages like Golem, Collier etc)

## Application (2 body decays)

Electromagnetic disintegration of heavy scalar ( $S(750) \rightarrow \gamma\gamma$ )



$$\mathcal{M}^{\mu\nu}(S \rightarrow \gamma\gamma) \sim \mu^2 \epsilon \int \frac{d^d k}{(2\pi)^d} \frac{\text{Tr}[(k-q+m) \gamma^\nu (k-p+m) \gamma^\mu (k+m)]}{[(k-q)^2 - m^2][(k-p)^2 - m^2][k^2 - m^2]}$$

In[12]= **LoopIntegrate**[

**Spur**[(**k** . **γ** - **q** . **γ** + **m** 1), **γ**<sub>ν</sub>, (**k** . **γ** - **p** . **γ** + **m** 1), **γ**<sub>μ</sub>, (**k** . **γ** + **m** 1)], **k**,  
{**k**, **m**}, {**k** - **p**, **m**}, {**k** - **q**, **m**}

Out[12]= **p**<sub>ν</sub> **q**<sub>μ</sub> (4 m PVC[0, 0, 0, p.p, p.p - 2 p.q + q.q, q.q, m, m, m] +  
8 m PVC[0, 1, 0, q.q, p.p - 2 p.q + q.q, p.p, m, m, m] +  
16 m PVC[0, 1, 1, p.p, p.p - 2 p.q + q.q, q.q, m, m, m]) +  
**p**<sub>μ</sub> **q**<sub>ν</sub> (4 m PVC[0, 0, 0, p.p, p.p - 2 p.q + q.q, q.q, m, m, m] +  
8 m PVC[0, 1, 0, p.p, p.p - 2 p.q + q.q, q.q, m, m, m] +  
8 m PVC[0, 1, 0, q.q, p.p - 2 p.q + q.q, p.p, m, m, m] +  
16 m PVC[0, 1, 1, p.p, p.p - 2 p.q + q.q, q.q, m, m, m]) +  
**p**<sub>μ</sub> **p**<sub>ν</sub> (16 m PVC[0, 1, 0, p.p, p.p - 2 p.q + q.q, q.q, m, m, m] +  
16 m PVC[0, 2, 0, p.p, p.p - 2 p.q + q.q, q.q, m, m, m]) +  
**q**<sub>μ</sub> **q**<sub>ν</sub> (8 m PVC[0, 1, 0, q.q, p.p - 2 p.q + q.q, p.p, m, m, m] +  
16 m PVC[0, 2, 0, q.q, p.p - 2 p.q + q.q, p.p, m, m, m]) +  
**g**<sub>μ,ν</sub>  
(-4 m PVB[0, 0, q.q, m, m] +  
(4 m p.p - 4 m p.q) PVC[0, 0, 0, p.p, p.p - 2 p.q + q.q, q.q, m, m, m] +  
16 m PVC[1, 0, 0, p.p, p.p - 2 p.q + q.q, q.q, m, m, m])

In[13]= % /. {**p** . **p** → 0, **q** . **q** → mS<sup>2</sup>, **q** . **p** →  $\frac{mS^2}{2}$ , **p**<sub>μ</sub> → 0, **p**<sub>ν</sub> → **q**<sub>ν</sub>}

Out[13]= **q**<sub>μ</sub> **q**<sub>ν</sub> (4 m PVC[0, 0, 0, 0, 0, mS<sup>2</sup>, m, m, m] + 8 m PVC[0, 1, 0, mS<sup>2</sup>, 0, 0, m, m, m] +  
16 m PVC[0, 1, 1, 0, 0, mS<sup>2</sup>, m, m, m]) +  
**q**<sub>μ</sub> **q**<sub>ν</sub> (8 m PVC[0, 1, 0, mS<sup>2</sup>, 0, 0, m, m, m] + 16 m PVC[0, 2, 0, mS<sup>2</sup>, 0, 0, m, m, m]) +  
**g**<sub>μ,ν</sub> (-4 m PVB[0, 0, mS<sup>2</sup>, m, m] - 2 m mS<sup>2</sup> PVC[0, 0, 0, 0, 0, mS<sup>2</sup>, m, m, m] +  
16 m PVC[1, 0, 0, 0, 0, mS<sup>2</sup>, m, m, m])

In[14]= **LoopRefine**[%] // **Simplify**

$$\frac{m \left( 4 mS^2 + (4 m^2 - mS^2) \text{Log} \left[ 1 + \frac{-mS^2 + \sqrt{-4 m^2 mS^2 + mS^4}}{2 m^2} \right]^2 \right) (-2 q_\mu q_\nu + mS^2 g_{\mu,\nu})}{mS^4}$$

Output of **LoopRefine** can always be evaluated numerically.

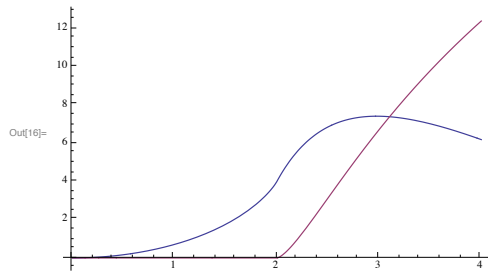
Results always consistent with the  $+i\epsilon$  prescription.

In[15]= **f**[**mS**\_, **m**\_] = **Coefficient**[% , **g**<sub>μ,ν</sub>]

$$\frac{m \left( 4 mS^2 + (4 m^2 - mS^2) \text{Log} \left[ 1 + \frac{-mS^2 + \sqrt{-4 m^2 mS^2 + mS^4}}{2 m^2} \right]^2 \right)}{mS^2}$$

Out[15]=

```
In[16]= Plot[{Re[f[mS, 1]], Im[f[mS, 1]]}, {mS, 0, 4}, PlotRange -> All]
```



## LoopRefine (overview of technical details)

Package-X can give analytic expressions for essentially **all kinematic configurations**.

in[17]: **(\*Usage information\*)**  
**? LoopRefine**

`LoopRefine[expr]` convert the Passarino-Veltman coefficient function in `expr` to simple function that can be numerically evaluated >>

**LoopRefine** is (by far) the biggest and most powerful function in Package-X:

- ◆ **Applies appropriate reduction formula (depending on kinematic configuration), converting tensor coefficient functions to basis functions.**

### 1 and 2 point functions (A, B functions)

- ◆ All  $A_{0\dots 0}(m_0)$  functions are hand-coded explicitly. [Denner, Dittmaier. (2006)]
- ◆ All  $B_{1\dots 1}(s; m_0, m_1)$  functions are hand-coded explicitly. [Denner, Dittmaier. (2006)]
- ◆ Higher rank  $B_{0\dots 01\dots 1}(s; m_0, m_1)$  functions determined iteratively [H.H. Patel, new in v2.0]

### 3 point functions (C functions)

- ◆ Case 1:  $\text{Det}(\mathcal{Z}) \neq 0$  [Passarino, Veltman. (1979)]
- ◆ Case 2a:  $C_{0\dots 01\dots 12\dots 2}(m_0^2, m_2^2, s; m_0, 0, m_2)$  [H.H. Patel, CPC 197 (2016)]
- ◆ Case 2b:  $C_{0\dots 01\dots 12\dots 2}(p_1^2, 0, p_2^2; m_0, 0, 0)$  [H.H. Patel, new in v2.0]
- ◆ Case 3:  $\text{Det}(\mathcal{Z}) = 0$ , but  $\text{Det}(X) \neq 0$  [Denner, Dittmaier. (2006)]
- ◆ Case 4:  $\text{Det}(\mathcal{Z}) = 0$  and  $\text{Det}(X) = 0$  [H.H. Patel, CPC 197 (2016)]
- ◆ Case 5,6:  $\tilde{Z}_{ij} = 0$  all  $i, j$  [Denner, Dittmaier. (2006)]

### 4 point functions (D functions) [New in v2.0]

- ◆ Case 1:  $\text{Det}(\mathcal{Z}) \neq 0$  [Passarino, Veltman. (1979)]
- ◆ Case 2:  $\text{Det}(\mathcal{Z}) = 0$ , but  $\tilde{Z}_{ij} f_j \neq 0$  [Denner, Dittmaier. (2006)]
- ◆ Case 3:  $\text{Det}(\mathcal{Z}) = 0$  and  $\tilde{Z}_{ij} f_j = 0$  [Denner, Dittmaier. (2006)]
- ◆ Case 4: Sporadic configurations [H.H. Patel, new]
- ◆ Case 5,6:  $\tilde{Z}_{ij} = 0$  all  $i, j$  [Denner, Dittmaier. (2006)]

- ◆ **Insert explicit analytic expressions for basis functions**

---

*B-functions* — Section IV A

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

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

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

*Auxiliary  $b^\xi$ -functions* — Section IV B

$b_{1\dots 1}^\xi(0, 0)$	$b_{1\dots 1}^\xi(p^2, 0)$
$b_{1\dots 1}^\xi(0, m)$	$b_{1\dots 1}^\xi(m^2, m)$

*Scalar  $C$ -functions* — Section V

$C_0(0, 0, 0; 0, 0, 0)$	$C_0(0, 0, q^2; 0, m_0, m_0)$	$C_0(m_0^2, 0, q^2; 0, 0, m_0)$	$C_0(0, p_2^2, q^2; m_2, 0, 0)$
$C_0(0, 0, q^2; 0, 0, 0)$	$C_0(0, 0, q^2; 0, m_1, m_0)$	$C_0(0, m_2^2, q^2; m_2, 0, 0)$	$C_0(p_1^2, 0, q^2; m_2, m_1, m_0)$
$C_0(0, 0, m_2^2; m_2, 0, 0)$	$C_0(0, 0, q^2; m_0, m_0, m_0)$	$C_0(m_0^2, 0, m_2^2; m_2, 0, m_0)$	$C_0(m_0^2, m_0^2, q^2; 0, 0, m_0)$
$C_0(0, 0, q^2; m_2, 0, 0)$	$C_0(0, 0, q^2; m_2, m_0, m_0)$	$C_0(p_1^2, p_2^2, q^2; m_2, m_1, m_0)$	$C_0(m_0^2, p_2^2, m_0^2; m_0, 0, m_0)$
$C_0(0, 0, q^2; 0, 0, m_0)$	$C_0(0, 0, q^2; m_2, m_1, m_0)$	$C_0(0, m_0^2, q^2; 0, m_0, m_0)$	$C_0(m_0^2, p_2^2, m_2^2; m_2, 0, m_0)$
$C_0(0, 0, m_0^2; m_0, m_0, m_0)$	$C_0(p_1^2, 0, q^2; 0, 0, 0)$	$C_0(0, p_2^2, q^2; m_0, m_0, m_0)$	$C_0(p_1^2, p_2^2, q^2; 0, 0, 0)$

- ◆ Each one determined by hand (explicit integration), and final result comes from this database:
  - ◇ UV-divergent parts
  - ◇ IR-divergent parts
  - ◇ finite parts
- ◆ **Expand in dimensional regulator  $\epsilon$  [improved in v2.0], retain to  $\mathcal{O}(\epsilon^0)$ .**
- ◆ **Expression optimization (bring results to "beautiful form").**  
 VERY difficult problem because "beautiful form" is not well-defined.
  - ◆ Logs of ratio of masses are brought to canonical form:
  - ◆ Logs of 't Hooft parameter  $\mu$  combined
  - ◆ 't Hooft parameter dependent logs brought together with  $1/\epsilon$

## New in Package-X 2.0

from <http://packagex.hepforge.org>

- `LoopIntegrate`
  - supports integrands with up to *four* distinct propagator factors.
  - accepts new and more natural syntax so that *any* momentum routing is possible.
  - `LoopIntegrate` can now remove spurious kinematic singularities when on-shell vector vertex `Projector` is used.
  - added option `cancel` to cancel common factors in the integrand, which generally leads to *substantially* increased performance and quality of `LoopRefine`.
  - added option `Organization`  $\rightarrow$  `Automatic` to specify how result is to be organized.
  - added support for integrands with propagator factors of arbitrary integer powers.
  - now takes `DiracMatrix`, `FermionLine` (new) and `FermionLineProduct` (new) objects in the numerator.
- `LoopRefine`
  - can now convert four-point coefficient functions `PVD` to elementary functions.
  - can now convert coefficient functions `PVA`, `PVB`, `PVC`, `PVD` of arbitrary integer weights defined in any even number of spacetime dimensions to elementary functions.
  - now tests for convergence, and issues a message if the loop integral is power infrared divergent.
  - added option `Analytic`  $\rightarrow$  `False` (default). Setting `Analytic`  $\rightarrow$  `True` generates a result with dim. reg. parameter  $\epsilon$  analytically continued to large and negative values, rendering power infrared divergences formally convergent.
  - added option `Part`  $\rightarrow$  `All`. Setting `Part`  $\rightarrow$  `UVDivergent` (rapidly) generates only the  $1/\epsilon$  part associated with UV-divergences
  - improved algorithms for expression optimization, including simplification of polylogarithms and their arguments.
- **NEW!** `LoopRefineSeries`  
generates a (multiple) Taylor series expansion of one-loop tensor integrals around any kinematic point away from Landau singularities to arbitrary order. Setting option `Analytic`  $\rightarrow$  `True` enables expansions around Landau singularities, by continuing the dim. reg. parameter  $\epsilon$  to sufficiently large and negative values, rendering the necessary derivatives of tensor integral existent.
- **NEW!** `MandelstamRelations`  
gives a list of replacement rules expressing Lorentz scalar products in terms of Mandelstam invariants and masses.
- **NEW!** Objects `FermionLine` and `FermionLineProduct`, and function `FermionLineExpand` added for rudimentary support of open fermion lines [*experimental*].
- `ScalarC0` (was `pvc0`)
  - improved performance of numerical implementation, and edge cases systematically covered.
  - automatically returns an explicit expression for vanishing external invariants.
- `ScalarC0IR6` (was `pvc0IR6`)
  - redefined so that it no longer exhibits a branch cut along branch cut along  $s < 0$  (which used to yield a non-vanishing imaginary part in that region).
  - improved performance of numerical implementation, and edge cases systematically covered.



## NEW: Four propagator factors

$$\mu^2 \epsilon \int \frac{d^d k}{(2\pi)^d} \frac{1}{[k^2 - M^2][(k+p_1)^2 - m^2][(k+p_1-p_3)^2 - M^2][(k-p_2)^2 - m^2]}$$

```
In[18]= LoopIntegrate[1, k,
  {k, M}, {k + p1, m}, {k + p1 - p3, M}, {k - p2, m}]
```

```
Out[18]= PVD[0, 0, 0, 0, p1.p1, p3.p3, p1.p1 + 2 p1.p2 - 2 p1.p3 + p2.p2 - 2 p2.p3 + p3.p3, p2.p2,
  p1.p1 - 2 p1.p3 + p3.p3, p1.p1 + 2 p1.p2 + p2.p2, M, m, M, m]
```

Generate kinematic conditions for 4 particle processes with **MandelstamRelations** (new):

```
In[19]= kinematic = MandelstamRelations[{p1, p2, p3, p4, M, M, M, M} → {s, t, u}, Eliminate → u]
```

```
Out[19]= {p1.p1 → M^2, p2.p2 → M^2, p3.p3 → M^2, p4.p4 → M^2, p1.p2 → 1/2 (-2 M^2 + s), p3.p4 → 1/2 (-2 M^2 + s),
  p1.p3 → 1/2 (2 M^2 - t), p2.p4 → 1/2 (2 M^2 - t), p1.p4 → 1/2 (-2 M^2 + s + t), p2.p3 → 1/2 (-2 M^2 + s + t)}
```

```
In[20]= loopInt = %% /. kinematic
```

```
Out[20]= PVD[0, 0, 0, 0, M^2, M^2, M^2, M^2, t, s, M, m, M, m]
```

### Analytic results for various kinematic configurations

Reduction at vanishing external momenta,

```
In[21]= loopInt /. {M^2 → 0, s → 0, t → 0}
  % // LoopRefine
```

```
Out[21]= PVD[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, M, m, M, m]
```

```
Out[22]= - 2 / (m^2 - M^2)^2 + (m^2 + M^2) Log[m^2 / M^2] / (m^2 - M^2)^3
```

...in forward scattering limit,

```
In[23]= loopInt /. {t → 0}
  resultForwardScattering[s_, m_, M_] = % // LoopRefine
```

```
Out[23]= PVD[0, 0, 0, 0, M^2, M^2, M^2, M^2, 0, s, M, m, M, m]
```

```
Out[24]= 2 M^2 DiscB[M^2, m, M] / ((-m^2 + 4 M^2) (-m^4 + 4 m^2 M^2 - M^2 s)) + DiscB[s, m, m] / (m^4 - 4 m^2 M^2 + M^2 s)
```

...for infrared divergent cases,

```
In[25]= loopInt /. {m → 0}
  % // LoopRefine
```

```
Out[25]= PVD[0, 0, 0, 0, M^2, M^2, M^2, M^2, t, s, M, 0, M, 0]
```

```
Out[26]= - 2 DiscB[t, M, M] (1/epsilon + Log[-mu^2 / s]) / (s (4 M^2 - t))
```

...most general case

```
In[27]= loopInt
  resultgeneral[s_, t_, m_, M_] =
  % // LoopRefine
```

```
Out[27]= PVD[0, 0, 0, 0, M^2, M^2, M^2, M^2, t, s, M, m, M, m]
```

```
Out[28]= ScalarD0[M^2, M^2, M^2, M^2, s, t, m, M, m, M]
```

## NEW Special functions:

Package-X 1.0 introduced special functions (abbreviations):

**Kallenλ**, **DiscB**, ...

v2.0 brings in new special functions associated with four-point integrals:

**Kibbleφ**, **ScalarD0**, **ScalarD0IR12**, **ScalarD013**, **BardinJ**, ...

*Special functions can be expanded in terms of elementary functions...*

### ◆ DiscB

```
In[29]= resultForwardScattering[s, m, M]
% // DiscExpand
```

$$\text{Out[29]= } \frac{2 M^2 \text{DiscB}[M^2, m, M]}{(-m^2 + 4 M^2) (-m^4 + 4 m^2 M^2 - M^2 s)} + \frac{\text{DiscB}[s, m, m]}{m^4 - 4 m^2 M^2 + M^2 s}$$

$$\text{Out[30]= } \frac{2 \sqrt{m^4 \left(1 - \frac{4 M^2}{m^2}\right)} \text{Log}\left[\frac{m^2 + \sqrt{m^4 \left(1 - \frac{4 M^2}{m^2}\right)}}{2 m M}\right]}{(-m^2 + 4 M^2) (-m^4 + 4 m^2 M^2 - M^2 s)} + \frac{\sqrt{s (-4 m^2 + s)} \text{Log}\left[\frac{2 m^2 - s + \sqrt{s (-4 m^2 + s)}}{2 m^2}\right]}{s (m^4 - 4 m^2 M^2 + M^2 s)}$$

◆ Compact analytic results for scalar D functions are rare. For example, this seemingly simple case [Davydychev (arXiv:9307323)]:

```
In[31]= ScalarD0[0, 0, 0, 0, s, t, m, m, m, m] // D0Expand
```

$$\text{Out[31]= } \text{ConditionalExpression}\left[-\frac{\pi^2}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}} - \frac{2 \text{Log}\left[\frac{\left(1 + \sqrt{1 - \frac{4 m^2}{s}}\right) t \left(\sqrt{1 - \frac{4 m^2}{s}} - \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)}{4 m^2}\right]^2}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}} - \frac{2 \text{Log}\left[\frac{s \left(1 + \sqrt{1 - \frac{4 m^2}{t}}\right) \left(\sqrt{1 - \frac{4 m^2}{t}} - \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)}{4 m^2}\right]^2}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}}\right] +$$

$$\frac{2 \text{Log}\left[-\frac{t \left(\sqrt{1 - \frac{4 m^2}{s}} - \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)^2}{4 m^2}\right] \text{Log}\left[-\frac{s \left(\sqrt{1 - \frac{4 m^2}{t}} - \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)^2}{4 m^2}\right]}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}} + \frac{4 \text{Log}\left[-\frac{s \left(\sqrt{1 - \frac{4 m^2}{s}} + \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right) \left(-\sqrt{1 - \frac{4 m^2}{t}} + \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)}{4 m^2}\right]^2}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}}$$

$$\frac{4 \text{PolyLog}\left[2, -\frac{\left(-1 + \sqrt{1 - \frac{4 m^2}{s}}\right) s \left(\sqrt{1 - \frac{4 m^2}{s}} - \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)}{4 m^2}\right]}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}} + \frac{4 \text{PolyLog}\left[2, \frac{\left(-1 + \sqrt{1 - \frac{4 m^2}{s}}\right) t \left(\sqrt{1 - \frac{4 m^2}{s}} - \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)}{4 m^2}\right]}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}}\right] +$$

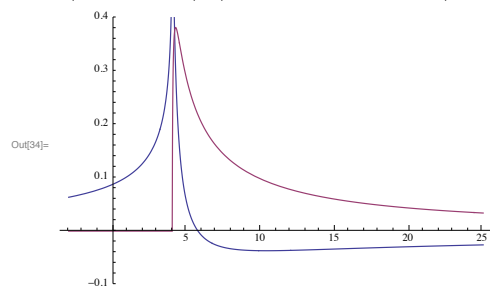
$$\frac{4 \text{PolyLog}\left[2, \frac{s \left(-1 + \sqrt{1 - \frac{4 m^2}{t}}\right) \left(\sqrt{1 - \frac{4 m^2}{t}} - \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)}{4 m^2}\right]}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}} - \frac{4 \text{PolyLog}\left[2, -\frac{\left(-1 + \sqrt{1 - \frac{4 m^2}{t}}\right) t \left(\sqrt{1 - \frac{4 m^2}{t}} - \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}\right)}{4 m^2}\right]}{s t \sqrt{1 - \frac{4 m^2 (s+t)}{s t}}}, s t (s t - 4 m^2 (s + t)) > 0]$$

∴ Package-X simply outputs these functions without inserting the analytic expressions.

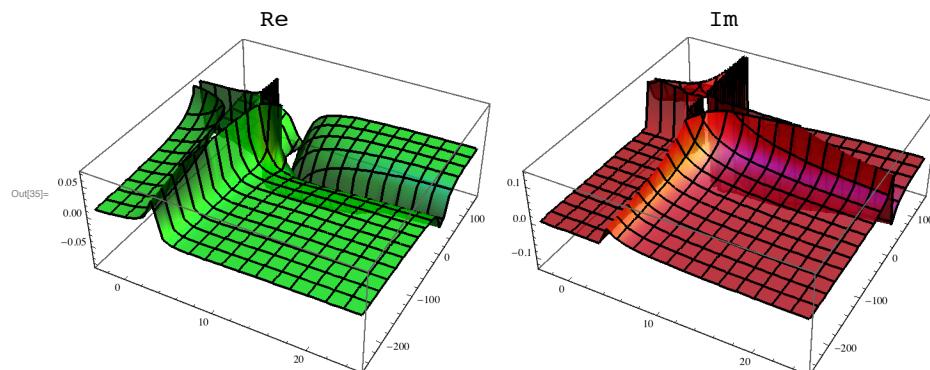
*Special functions can be evaluated numerically...*

```
In[33]= resultForwardScattering[s, m, M]
Plot[{Re[resultForwardScattering[s, 1, 2]], Im[resultForwardScattering[s, 1, 2]]},
{s, -3, 25}, PlotRange -> {-0.1, .4}]
```

$$\text{Out[33]= } \frac{2 M^2 \text{DiscB}[M^2, m, M]}{(-m^2 + 4 M^2) (-m^4 + 4 m^2 M^2 - M^2 s)} + \frac{\text{DiscB}[s, m, m]}{m^4 - 4 m^2 M^2 + M^2 s}$$



```
In[35]= Grid[{{"Re", "Im"}, {Plot3D[Re[resultgeneral[s, t, 1, 2]], {s, -3, 25},
{t, -250, 150}, PlotPoints -> 30, ClippingStyle -> None, PlotStyle ->
Directive[Green, Specularity[White, 20], Opacity[0.8]], ImageSize -> Medium],
Plot3D[Im[resultgeneral[s, t, 1, 2]], {s, -3, 25},
{t, -250, 150}, PlotPoints -> 30, ClippingStyle -> None, PlotStyle ->
Directive[Red, Specularity[White, 20], Opacity[0.8]], ImageSize -> Medium]}}
```



## Special functions I -- machine precision evaluation:

Implementation is compiled to the Wolfram Virtual Machine (using `Compile[]`)

Increased evaluation speed of `ScalarC0` and `ScalarC0IR6`

Lot of thought and time went into refining the underlying algorithms... (see Package-X paper for details)

```
Manipulate[Plot[{Re[ScalarC0[qSq, m1^2, m1^2, m2, m2, m2]], Im[ScalarC0[qSq, m1^2, m1^2, m2, m2, m2]]}, {qSq, -.05, 5}, PlotRange -> {-20, 12},
  Epilog -> {
    Text["anomalous\ntreshold", {m1^2 (4 - m1^2/m2^2), 6}], Arrow[{{m1^2 (4 - m1^2/m2^2), 4}, {m1^2 (4 - m1^2/m2^2), .2}}],
    Text["normal\ntreshold", {4 m2^2, -12}], Arrow[{{4 m2^2, -10}, {4 m2^2, -5}}]
  }],
  {{m1, 0.2}, .1, 2}, {{m2, .95}, .8, 1.0}]
```

Compare evaluation speeds with Fortran implementation of LoopTools:

	CompilationTarget	AbsoluteTiming*
LoopTools		1.17 sec
Package-X	"WVM"	0.96 sec
	"C"	0.19 sec

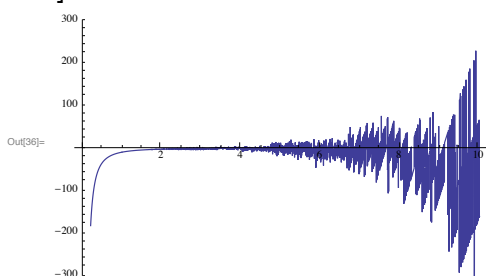
\*10,000 evaluations with all arguments non-zero above normal threshold

To be fair, LoopTools does a better job handling large cancellations, and can handle complex masses.

## NEW: Special Functions II -- arbitrary precision evaluation:

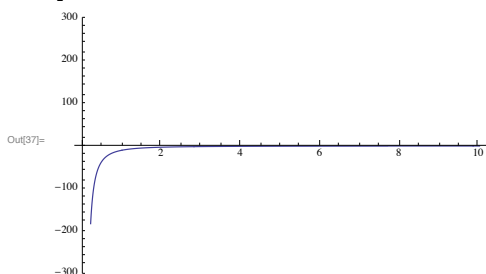
For theories with wide separation of scales, straightforward evaluation can lead to numerical noise:

```
In[36]= With[{me = 0.511*^-3, mμ = 105.7*^-3, mq = 4.8*^-3},
  Plot[Re@ScalarC0[0, me^2, mμ^2, mq, mq, mLQ], {mLQ, 0.2, 10}, PlotRange -> {-300, 300}]
]
```



New in v2.0: Special functions of Package-X now takes advantage of *Mathematica's* arbitrary precision arithmetic (no more noise, slower to calculate):

```
In[37]= With[{me = 0.511`20*^-3, mμ = 105.7`20*^-3, mq = 4.8`20*^-3},
  Plot[Re@ScalarC0[0, me^2, mμ^2, mq, mq, mLQ],
  {mLQ, 0.2, 10}, PlotRange -> {-300, 300}, WorkingPrecision -> 20]
]
```



## NEW: Taylor series expansions

A second (smarter) option to handle wide separation of scales is to make a Taylor expansion:

New in Package-X v2.0: `LoopRefineSeries`

Can construct a multivariate Taylor series expansion of loop integrals to any order around any kinematic point (where they exist).

$$\mu^{2\epsilon} \int \frac{d^d k}{(2\pi)^d} \frac{1}{[k^2 - m^2 + i\epsilon][(k+p)^2 - M^2 + i\epsilon]} = B_0(p^2; m, M)$$

In[38]= `LoopIntegrate[1, k, {k, m}, {k+p, M}] /. p.p -> s`

Out[38]= `PVB[0, 0, s, m, M]`

In[39]= `(*Small momentum expansion s=0 *)`

`LoopRefineSeries[PVB[0, 0, s, m, M], {s, 0, 2}]`

$$\text{Out[39]=} \left( 1 + \frac{1}{\epsilon} - \frac{m^2 \text{Log}\left[\frac{m^2}{M^2}\right]}{m^2 - M^2} + \text{Log}\left[\frac{\mu^2}{M^2}\right] \right) + \left( \frac{m^2 + M^2}{2(m^2 - M^2)^2} - \frac{m^2 M^2 \text{Log}\left[\frac{m^2}{M^2}\right]}{(m^2 - M^2)^3} \right) s +$$

$$\left( \frac{m^4 + 10 m^2 M^2 + M^4}{6(m^2 - M^2)^4} - \frac{m^2 M^2 (m^2 + M^2) \text{Log}\left[\frac{m^2}{M^2}\right]}{(m^2 - M^2)^5} \right) s^2 + \mathcal{O}[s]^3$$

In[40]= `(*Degenerate mass expansion M~m*)`

`LoopRefineSeries[PVB[0, 0, s, m, M], {M, m, 2}]`

$$\text{Out[40]=} \left( 2 + \frac{1}{\epsilon} + \text{DiscB}[s, m, m] + \text{Log}\left[\frac{\mu^2}{m^2}\right] \right) + \frac{2 m \text{DiscB}[s, m, m] (M - m)}{4 m^2 - s} +$$

$$\left( -\frac{2(2 m^2 - s)}{(4 m^2 - s) s} + \frac{(-8 m^4 + 4 m^2 s - s^2) \text{DiscB}[s, m, m]}{s(-4 m^2 + s)^2} \right) (M - m)^2 + \mathcal{O}[M - m]^3$$

In[41]= `(*Double expansion*)`

`LoopRefineSeries[PVB[0, 0, s, m, M], {s, 0, 2}, {M, m, 2}]`

$$\text{Out[41]=} \left( \left( \frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right] \right) - \frac{M - m}{m} + \frac{(M - m)^2}{6 m^2} + \mathcal{O}[M - m]^3 \right) + \left( \frac{1}{6 m^2} - \frac{M - m}{6 m^3} + \frac{7 (M - m)^2}{60 m^4} + \mathcal{O}[M - m]^3 \right) s +$$

$$\left( \frac{1}{60 m^4} - \frac{M - m}{30 m^5} + \frac{17 (M - m)^2}{420 m^6} + \mathcal{O}[M - m]^3 \right) s^2 + \mathcal{O}[s]^3$$

### Truly annoying limitation of LoopRefineSeries:

Can only expand where Taylor expansion exists, away from Landau singularities (no asymptotic expansions):

#### ◆ No small mass expansion

```
In[42]= LoopRefineSeries[PVB[0, 0, s, m, m], {m, 0, 3}]
```

LoopRefineSeries::notaylor: TayloseriesofPVBdoesnotexisthearm=0. >>

```
Out[42]= $Failed
```

#### ◆ No threshold expansion

```
In[43]= LoopRefineSeries[PVB[0, 0, s, m, m], {s, 4 m^2, 2}]
```

LoopRefineSeries::notaylor: TayloseriesofPVBdoesnotexisthears=4m^2. >>

```
Out[43]= $Failed
```

### Workaround

Convert everything to expressions *Mathematica* knows about, and use built-in `Series`

```
In[44]= LoopRefine[PVB[0, 0, s, m, m]] // DiscExpand
```

$$\text{Out[44]= } 2 + \frac{1}{\epsilon} + \frac{\sqrt{s(-4m^2 + s)} \operatorname{Log}\left[\frac{2m^2 - s + \sqrt{s(-4m^2 + s)}}{2m^2}\right]}{s} + \operatorname{Log}\left[\frac{\mu^2}{m^2}\right]$$

```
In[45]= Assuming[s < 0, Simplify[Series[%, {m, 0, 4}, Assumptions -> s < 0]]]
```

$$\text{Out[45]= } \left(2 + \frac{1}{\epsilon} + \operatorname{Log}\left[-\frac{\mu^2}{s}\right]\right) + \frac{(2 - 4 \operatorname{Log}[m] + 2 \operatorname{Log}[-s]) m^2}{s} + \frac{(-1 - 4 \operatorname{Log}[m] + 2 \operatorname{Log}[-s]) m^4}{s^2} + O[m]^5$$

```
In[46]= Assuming[s < 0, Simplify[Series[%%, {s, 4 m^2, 2}, Assumptions -> s < 0]]]
```

$$\text{Out[46]= } \left(2 + \frac{1}{\epsilon} + \operatorname{Log}\left[\frac{\mu^2}{m^2}\right]\right) + \frac{i \pi \sqrt{s - 4 m^2}}{2 \sqrt{m^2}} - \frac{s - 4 m^2}{2 m^2} - \frac{i \sqrt{m^2} \pi (s - 4 m^2)^{3/2}}{16 m^4} + \frac{(s - 4 m^2)^2}{12 m^4} + O[s - 4 m^2]^{5/2}$$

Drawback: for more complicated cases, `Series` can be slow. R&D underway...

## NEW: Tests for power IR divergences

Certain Feynman integrals can have IR divergences worse than logarithmic. Typically, these are log-IR divergent integrals evaluated at threshold.

*e.g. 1* Ellis-Zanderighi triangle 5:

$$C_0(0, m^2, m^2; 0, 0, m) \sim \int_0^1 dz \frac{1-z}{(z^2-i\epsilon)^{1-\epsilon}} \leftarrow \text{non-convergent}$$

```
In[47]= LoopRefine[PVC[0, 0, 0, 0, m^2, m^2, 0, 0, m]]
```

```
LoopRefining: Non-logarithmic power infrared divergence encountered from moving away from threshold and inspect behaviors kinematic points reached or set option Analytic->True >>
```

```
Out[47]= ComplexInfinity
```

*e.g. 2* Forward Bhabha scattering ( $e^+ e^- \rightarrow e^+ e^-$ ) at NLO

```
In[48]= LoopRefine[PVD[0, 0, 0, 0, m^2, m^2, m^2, m^2, s, 0, m, 0, m, 0]]
```

```
LoopRefining: Non-logarithmic power infrared divergence encountered from moving away from threshold and inspect behaviors kinematic points reached or set option Analytic->True >>
```

```
Out[48]= ComplexInfinity
```

New option to **LoopRefine**: obtain an analytically continued result:

```
In[49]= LoopRefine[PVC[0, 0, 0, 0, m^2, m^2, 0, 0, m], Analytic->True]
```

```
Out[49]=
```

$$\frac{1}{m^2} - \frac{\frac{1}{\epsilon} + \text{Log}\left[\frac{s}{m^2}\right]}{2 m^2}$$

```
In[50]= LoopRefine[PVD[0, 0, 0, 0, m^2, m^2, m^2, m^2, s, 0, m, 0, m, 0], Analytic->True]
```

```
Out[50]=
```

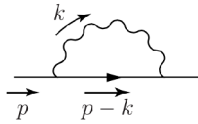
$$\frac{1}{m^2 (4 m^2 - s)} - \frac{2 \text{DiscB}[s, m, m]}{(4 m^2 - s)^2}$$



## NEW: Support for integrals with open fermion lines

In v1.0, no support for open fermion lines. Needed to use `Projector` to extract form factors out of one loop integrals:

vis, QED electron self energy



$$\sim A(p^2) p \cdot \gamma + B(p^2) m$$

```
In[51]:= LoopIntegrate[Spur[γμ, p·γ - k·γ + 1 m, γμ, Projector["A"][p, m]], k, {k, 0}, {k - p, m}];
% // LoopRefine
LoopIntegrate[Spur[γμ, p·γ - k·γ + 1 m, γμ, Projector["B"][p, m]], k, {k, 0}, {k - p, m}];
% // LoopRefine
```

```
Out[52]=
```

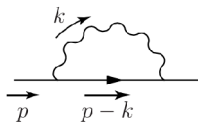
$$-\frac{1}{\epsilon} + \frac{-m^2 - p \cdot p}{p \cdot p} - \text{Log}\left[\frac{\mu^2}{m^2}\right] + \frac{(m^4 - (p \cdot p)^2) \text{Log}\left[\frac{m^2}{m^2 - p \cdot p}\right]}{(p \cdot p)^2}$$

```
Out[54]=
```

$$6 + 4 \left( \frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right] \right) - \frac{4 (m^2 - p \cdot p) \text{Log}\left[\frac{m^2}{m^2 - p \cdot p}\right]}{p \cdot p}$$

### Now in v2.0

Off shell fermion lines (no  $u$  and  $v$  spinors) represented by `DiracMatrix`:



$$\sim A(p^2) p \cdot \gamma + B(p^2) m$$

```
In[55]:= LoopIntegrate[DiracMatrix[γμ, p·γ - k·γ + 1 m, γμ], k, {k, 0}, {k - p, m}];
% // LoopRefine
```

```
Out[56]=
```

$$\text{DiracMatrix}[] \left( 6 m + 4 m \left( \frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right] \right) - \frac{4 (m^3 - m p \cdot p) \text{Log}\left[\frac{m^2}{m^2 - p \cdot p}\right]}{p \cdot p} \right) +$$

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

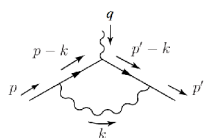
On shell fermion lines represented with `FermionLine`

```
In[57]:= FermionLine[{1, p2, m2}, {1, p1, m1}, DiracMatrix[γμ, (p1 + k)·γ + m 1, γν]]
```

```
Out[57]= <u[p2, m2], γμ, m 1 + k·γ + p1·γ, γν, u[p1, m1]>
```

Now it is possible to get  $F_1, F_2, \dots$  in one go.

QED vertex function:



$$\sim F_1(q^2) \gamma^\mu + F_2(q^2) \sigma^{\mu\nu} q_\nu / (2 m)$$

```

In[58]= LoopIntegrate[⟨u[p', m], γν, (p'·γ - k·γ + m 1), γμ, (p·γ - k·γ + m 1), γν, u[p, m]⟩,
  k, {k - p', m}, {k - p, m}, {k, 0, 1}] /.
  {p·p → m2, p'·p' → m2, p·p' → -q·q / 2 + m2} // LoopRefine
Out[58]= 
$$\begin{aligned}
& - \frac{2 \operatorname{Im} \operatorname{DiscB}[q \cdot q, m, m] \langle u[p', m], \sigma_{\mu, \{-p+p'\}}, u[p, m] \rangle}{4 m^2 - q \cdot q} + \\
& \langle u[p', m], \gamma_{\mu}, u[p, m] \rangle \\
& \left( \frac{\operatorname{DiscB}[q \cdot q, m, m] (-8 m^2 + 3 q \cdot q)}{4 m^2 - q \cdot q} + \left( 1 - \frac{2 \operatorname{DiscB}[q \cdot q, m, m] (2 m^2 - q \cdot q)}{4 m^2 - q \cdot q} \right) \left( \frac{1}{\epsilon} + \operatorname{Log} \left[ \frac{\mu^2}{m^2} \right] \right) + \right. \\
& \left. 2 (2 m^2 - q \cdot q) \operatorname{ScalarC0IR6}[q \cdot q, m, m] \right)
\end{aligned}$$


```

Fully automated application of Dirac algebra, spinor on shell conditions, Chisholm identities, Sirlin identities, Gordon identities...

## Summary

### *What does Package-X do?*

Package-X 2.0 can now generate analytic expressions for arbitrarily high rank dimensionally regulated tensor integrals with up to four distinct propagators, each with arbitrary integer weight, giving UV divergent, IR divergent, and finite parts at arbitrary (real-valued) kinematic points. Package-X 2.0 can test for power infrared divergences, and can give expressions continued to large (negative) dim. reg. epsilon if needed. Furthermore, Package-X 2.0 can generate multivariable Taylor series expansions of these integrals near any (non-singular) kinematic point to arbitrary order. All output can be numerically evaluated or symbolically manipulated natively inside *Mathematica*.

### *Wishlist (stuff I need help with)*

- ◆ Enable expansions near Landau singularities
- ◆ For more general kinematics, use unitarity based methods
- ◆ Extend to two loop integrals (vacuum bubbles/self energy/form factors...)
- ◆ Add support for non-covariant propagator factors (HQET/Coulomb gauge prop/SCET,...)
- ◆ Link to other well-established packages