Package-X 2.1 and CollierLink

August 22, 2017

ACAT Track 3

Hiren Patel
hhpatel@umass.edu

UMASS AMHERST

Amherst Center for Fundamental Interactions

What is *Package*-X ?

Released in 2015 :
Package-X is a *Mathematica* package to (mainly) calculate one-loop integrals analytically.

Calculations carried out in 3 steps:

$$\int \frac{d^d k}{(2\pi)^d} \frac{k^\mu k^\nu}{[(k+p)^2 - m^2]\, k^2} \text{ at } p^2 = 0$$

1.
```
LoopIntegrate[kμ kν, k, {k + p, m}, {k, 0}]
```
$p_\mu\, p_\nu\, \text{PVB}[0, 2, p.p, 0, m] + g_{\mu,\nu}\, \text{PVB}[1, 0, p.p, 0, m]$

2.
```
% /. p.p → 0
```
$p_\mu\, p_\nu\, \text{PVB}[0, 2, 0, 0, m] + g_{\mu,\nu}\, \text{PVB}[1, 0, 0, 0, m]$

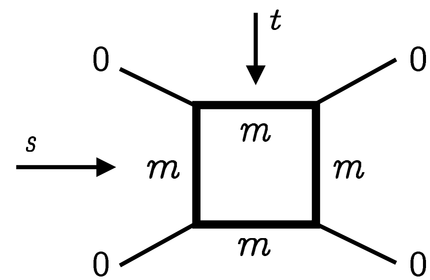
3.
```
% // LoopRefine
```
$\left(\frac{1}{9} + \frac{1}{3}\left(\frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right]\right)\right) p_\mu\, p_\nu + \left(\frac{3\, m^2}{8} + \frac{1}{4} m^2 \left(\frac{1}{\epsilon} + \text{Log}\left[\frac{\mu^2}{m^2}\right]\right)\right) g_{\mu,\nu}$

Important goal: ***EASY TO USE***

What is *Package*-X ?

Version 2 Released in 2016 :

A) boxes




```
LoopIntegrate[1, k, {k, m}, {k + p1, m},
 {k - p3 + p1, m}, {k - p2, m}]
```

```
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]
```

```
LoopRefine[%] // DOExpand
```

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

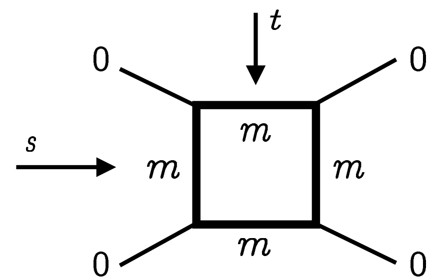
A.I.Davydychev
arXiv: 9307323

UMASS AMHERST | ACFI

What is *Package*-X ?

Version 2 Released in 2016 :

B) series expansions



Large mass expansion ($m \sim \infty$)

```
LoopRefineSeries[%%, {m, Infinity, 12}]
```

$$\frac{1}{6\,m^4}+\frac{s+t}{60\,m^6}+\frac{2\,s^2+s\,t+2\,t^2}{840\,m^8}+\frac{3\,s^3+s^2\,t+s\,t^2+3\,t^3}{7560\,m^{10}}+\frac{12\,s^4+3\,s^3\,t+2\,s^2\,t^2+3\,s\,t^3+12\,t^4}{166\,320\,m^{12}}+O\left[\frac{1}{m}\right]^{13}$$

Forward limit ($t \sim 0$)

```
LoopRefineSeries[%%%, {t, 0, 2}]
```

$$\left(\frac{2}{m^2\,s}+\frac{\text{DiscB}[s,m,m]}{m^2\,s}\right)+\left(-\frac{4\,(3\,m^2-s)}{9\,m^4\,s^2}-\frac{(4\,m^2-s)\,\text{DiscB}[s,m,m]}{6\,m^4\,s^2}\right)t+$$
$$\left(\frac{240\,m^4-140\,m^2\,s+23\,s^2}{225\,m^6\,s^3}+\frac{(4\,m^2-s)^2\,\text{DiscB}[s,m,m]}{30\,m^6\,s^3}\right)t^2+O[t]^3$$

C) improved numerics

machine precision

```
ScalarD0[5.4, 1.2, 5.4, 1.2, 54.1, -12, 1.1, 1.1, 1.1, 2.3]
-0.0174954 + 0.0101058 i
```

arbitrary precision

```
ScalarD0[5.4`20, 1.2`20, 5.4`20, 1.2`20, 54.1`20, -12`20, 1.1`20, 1.1`20, 1.1`20, 2.3`20]
-0.017495380335162055127905 8 + 0.010105816736175921756070 57 i
```

What is *Package*-X ?

Version 2 Released in 2016 :

D) Dirac algebra

Calculate traces (already in v1.0)

```
Spur[k.γ + m 1, γ_μ, (k + p).γ + m 1, γ_ν]
```

$8\,k_\mu\,k_\nu + 4\,k_\nu\,p_\mu + 4\,k_\mu\,p_\nu + 4\,m^2\,g_{\mu,\nu} - 4\,k.k\,g_{\mu,\nu} - 4\,k.p\,g_{\mu,\nu}$

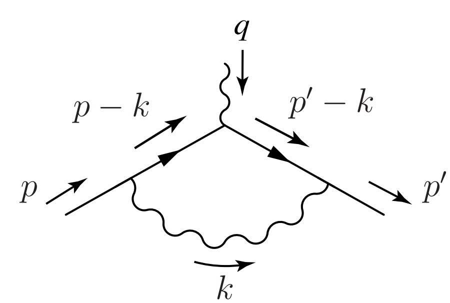
Put open fermion lines in canonical form:

```
⟨u[p2, m], p1.γ, γ_μ, p2.γ, u[p1, m]⟩ // FermionLineExpand
```

$-2\,i\,m\,\langle u[p2,m],\ \sigma_{\mu,\{-p1+p2\}},\ u[p1,m]\rangle + \langle u[p2,m],\ \gamma_\mu,\ u[p1,m]\rangle\,\left(3\,m^2 - 2\,p1.p2\right)$

Compute a Feynman diagram (QED vertex function):

```
LoopIntegrate[⟨u[p′, m], γ_ν, (p′.γ - k.γ + m 1), γ_μ,
    (p.γ - k.γ + m 1), γ_ν, 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
```



$$-\frac{2\,i\,m\,\mathrm{DiscB}[q.q,m,m]\,\langle u[p',m],\ \sigma_{\mu,\{-p+p'\}},\ u[p,m]\rangle}{4\,m^2 - q.q} +$$

$$\langle u[p',m],\ \gamma_\mu,\ u[p,m]\rangle\left(-\frac{\mathrm{DiscB}[q.q,m,m]\,\left(8\,m^2 - 3\,q.q\right)}{4\,m^2 - q.q} + \right.$$

$$\left.\left(1 - \frac{2\,\mathrm{DiscB}[q.q,m,m]\,\left(2\,m^2 - q.q\right)}{4\,m^2 - q.q}\right)\left(\frac{1}{\epsilon} + \mathrm{Log}\left[\frac{\mu^2}{m^2}\right]\right) + 2\,\left(2\,m^2 - q.q\right)\,\mathrm{ScalarC0IR6}[q.q,m,m]\right)$$

What is *Package*-X ?

Version 2 Released in 2016 :

E) Intuitive and readable user interface

Input and Output in StandardForm

```
In[5]:= LDot[a, b]

Out[5]= a.b
```

```
In[4]:= ⟨u[p, m], γμ, v[p, m]⟩ // InputForm

     FermionLine[{1, p, m}, {-1, p, m},
      DiracMatrix[LTensor[DiracG, μ]]]
```

Traditional form also available!

```
DiracMatrix[γμ, (k + p).γ, γν, PL] //
 TraditionalForm
```

$\gamma^{\mu}\,(\not{k}+\not{p})\,\gamma^{\nu}\,\mathrm{P_L}$

$\gamma.p$

F) Integrated documentation

General Tensor Operations and Symbols

- **LTensor** ($\square_\square$), **LDot** ($\square.\square$), **LScalarQ** — Lorentz tensors and scalar products
- **Contract** — contract tensors with repeated indices
- **Transverse**, **Longitudinal** — project 2nd or higher rank Lorentz tensors
- **MandelstamRelations** — express scalar products in terms of Mandelstam invariants
- $g_{\square,\square}$ · $\varepsilon_{\square,\square,\square,\square}$ — metric tensor, and Levi-Civita symbol

Computing One-Loop Integrals

- **LoopIntegrate**, **LoopRefine** — routines for one loop integrals
- **LoopRefineSeries** — compute series expansions of loop integrals
- d · ϵ · μ — dimensional regularization symbols

Dirac Algebra and Fermion Spinors

- **Spur** — compute traces over product of Dirac matrices
- **Projector** — project fermion self energy and vertex functions onto form factors
- $\mathbb{1}$ · $\gamma_\square$ · $\gamma5$ · $\mathbb{P}\mathrm{L}$ · $\mathbb{P}\mathrm{R}$ · $\sigma_{\square,\square}$ — objects in spinor space
- **DiracMatrix**, **FermionLine** (u, v), **FermionLineProduct**, **FermionLineExpand** — open fermion chains

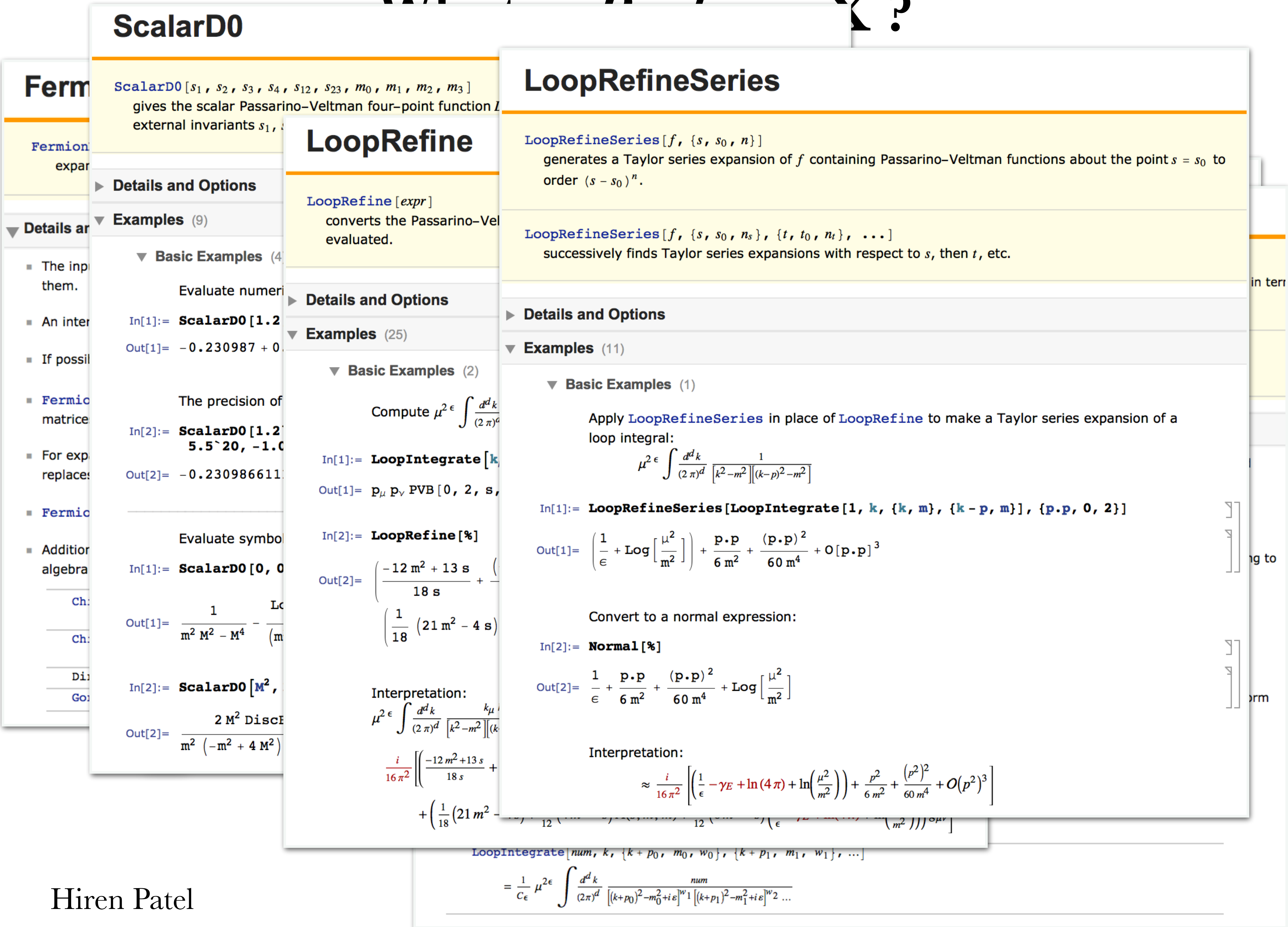
---

Generated Symbols

- **PVA** · **PVB** · **PVC** · **PVD** · **PVX** — symbolic Passarino-Veltman functions

Special Functions and Abbreviations »

- **Kallenλ** · **DiscB** · **ScalarC0** · **ScalarD0** · ...



ScalarD0
ScalarD0[s1, s2, s3, s4, s12, s23, m0, m1, m2, m3]
gives the scalar Passarino–Veltman four–point function
Details and Options
Examples (9)
Basic Examples
LoopRefine
LoopRefine[expr]
converts the Passarino–Vel
evaluated.
Details and Options
Examples (25)
Basic Examples (2)
Interpretation:
LoopRefineSeries
LoopRefineSeries[f, {s, s0, n}]
generates a Taylor series expansion of f containing Passarino–Veltman functions about the point s = s0 to order (s − s0)^n.
LoopRefineSeries[f, {s, s0, ns}, {t, t0, nt}, ...]
successively finds Taylor series expansions with respect to s, then t, etc.
Details and Options
Examples (11)
Basic Examples (1)
Apply LoopRefineSeries in place of LoopRefine to make a Taylor series expansion of a loop integral:
In[1]:= LoopRefineSeries[LoopIntegrate[1, k, {k, m}, {k - p, m}], {p.p, 0, 2}]
Out[1]=
Convert to a normal expression:
In[2]:= Normal[%]
Out[2]=
Interpretation:

Hiren Patel

What is *Package*-X ?

skip

Selected features of *Package*-X

Package-X can generate analytic expressions for **arbitrarily high rank** tensor integrals **with up to four distinct propagators**, each with **arbitrary integer weight**, giving UV and IR divergent, and finite parts at **arbitrary** (**real-valued**) **kinematic points**, and can can construct multivariable Taylor series expansions near **arbitrary** (**non-singular**) **kinematic point** to **arbitrary order**.

Package-X can calculate traces of products of gamma matrices, and perform tensor algebraic operations on open fermion lines.

Package-X can numerically evaluate scalar basis functions with either machine precision (fast) and arbitrary precision (slow) at any kinematic point.

What is *Package*-X ?

Applications: anything… except fully differential cross sections

- shift in vacuum expectation values
- shift in pole mass
- wavefunction renormalization
- electroweak oblique parameters
- particle electromagnetic moments (EDMs, polarizability, …)
- 2 body decays, 3 body decays
- counterterms
- wilson coefficients
- cross sections at threshold

…

Motivation:

Until *Package*-X, there was no comprehensive one-loop package available.
There are semi-analytic packages (like FeynCalc,…) and numerical ones (like LoopTools,…)

- Sometimes difficult to use
- Does not always get an answer, like at vanishing Gram determinant, etc…

New in *Package*-X 2.1

A) Command-line readiness

Before, names of keywords were the symbols

$\varepsilon_{\mu,\nu,\rho,\varsigma}\, \mathfrak{g}_{\rho,\nu}$ **DiracMatrix**[γ_ν, $\sigma_{\rho,\varsigma}$] **// FullForm**

```
Times[DiracMatrix[LTensor[\[Gamma], \[Nu]],
  LTensor[\[Sigma], \[Rho], \[FinalSigma]]],
 LTensor[\[DoubleStruckG], \[Nu], \[Rho]],
 LTensor[\[CurlyEpsilon], \[Mu],
  \[Nu], \[Rho], \[FinalSigma]]]
```

Now, keywords have proper full names, and parse into full names by front-end.

$\varepsilon_{\mu,\nu,\rho,\varsigma}\, \mathfrak{g}_{\rho,\nu}$ **DiracMatrix**[γ_ν, $\sigma_{\rho,\varsigma}$] **// FullForm**

```
Times[DiracMatrix[LTensor[DiracG, \[Nu]],
  LTensor[DiracS, \[Rho], \[FinalSigma]]],
 LTensor[MetricG, \[Nu], \[Rho]], LTensor[
  LeviCivitaE, \[Mu], \[Nu], \[Rho], \[FinalSigma]]]
```

All old front-end notebooks continue to work,
but now also convenient to use command-line and edit .m (.wl) files:

Square amplitude

```
applyPolarizationCompleteness[expr_, List[p_,0,idx_]]:=
 Replace[expr,{
   HoldPattern[rest_. Power[LTensor[MetricG, PatternSequence[idx,a_]|Pat
   HoldPattern[rest_. LTensor[v_,idx]^2] :> (-LDot[v,v]) rest,
   HoldPattern[rest_. LTensor[MetricG, PatternSequence[idx,a_]|PatternSe
   HoldPattern[rest_. LTensor[MetricG, PatternSequence[idx,a_]|PatternSe
   HoldPattern[rest_. LTensor[MetricG, idx, idx]] :> rest (-Dim),
   HoldPattern[rest_. LTensor[v_,idx]*LTensor[w_,idx]] :> (-LDot[v,w])

   expr :> ReplaceOnce[expr,LTensor[vec_,l___,idx,r___]:>-LTensor[vec,l,
 }];
```

```
Mathematica 10.3.1 for Mac OS X x86 (64-bit)
Copyright 1988-2015 Wolfram Research, Inc.

In[1]:= <<X`
Package-X v2.1.0 [developer version], by Hiren H. Patel

In[2]:= FermionLine[{-1,p,m},{1,p,m},DiracMatrix[LTensor[DiracG,mu]]]*LTensor[q,
mu]

Out[2]= 〈 v[p, m], γ(mu), u[p, m] 〉 q(mu)

In[3]:= %//Contract

Out[3]= 〈 v[p, m], γ.q, u[p, m] 〉

In[4]:=
```

New in *Package*-X 2.1

B) Calculate the discontinuity across normal threshold cut for any one-loop integral:

Set option `Part→Discontinuity[s]` *to LoopRefine*

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

1. `LoopIntegrate[k_μ k_ν, k, {k + p, m}, {k, 0}]`

   $p_\mu\, p_\nu\, \mathrm{PVB}[0, 2, p.p, 0, m] + g_{\mu,\nu}\, \mathrm{PVB}[1, 0, p.p, 0, m]$

2. `% /. p.p → t`

   $p_\mu\, p_\nu\, \mathrm{PVB}[0, 2, t, 0, m] + g_{\mu,\nu}\, \mathrm{PVB}[1, 0, t, 0, m]$

3. `LoopRefine[%, Part → Discontinuity[t]]`

   $$\frac{2\, i\, \pi\, (-m^2 + t)^3\, \mathrm{HeavisideTheta}[-m^2 + t]\, p_\mu\, p_\nu}{3\, t^3} - \frac{i\, \pi\, (-m^2 + t)^3\, \mathrm{HeavisideTheta}[-m^2 + t]\, g_{\mu,\nu}}{6\, t^2}$$

Applications:

- Compute tree-level cross sections (optical theorem)
- Leptogenesis: only need imaginary parts of 1-loop diagrams

Important!

Discontinuity

`Discontinuity[s]`
is a setting for `Part` to `LoopRefine` to compute the discontinuity across the normal threshold c
s.

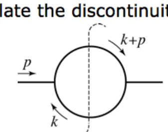
`Discontinuity[s, t]`
computes the Mandelstam double spectral function in overlapping channels *s* and *t*.

▸ Details and Options

▾ Examples (7)

▾ Basic Examples (1)

Calculate the discontinuity across the normal threshold cut of the scalar two-point functio




```
In[1]:= LoopIntegrate[1, k, {k + p, m}, {k, m}]
        LoopRefine[%, Part → Discontinuity[p.p]]
Out[1]= PVB[0, 0, p.p, m, m]
```

COLLIER

April 2016, COLLIER released!

arXiv.org > hep-ph > arXiv:1604.06792

Search

(Help | Adv

High Energy Physics – Phenomenology

Collier: a fortran-based Complex One-Loop LIbrary in Extended Regularizations

Ansgar Denner, Stefan Dittmaier, Lars Hofer

- Numerical library of Passarino-Veltman functions (scalar *and* tensor integrals)
- Arbitrarily high rank *and* arbitrarily many legs
- Dimensional regularization *and* mass regularization
- Complex internal masses
- Dedicated expansions near vanishing Gram/Cayley determinants
- **Two** independent implementations (and nothing to do with FF library)

Bring to *Mathematica*! $\Longrightarrow$ CollierLink

CollierLink

Three features -

1. *Direct numerical evaluation of Passarino-Veltman functions*

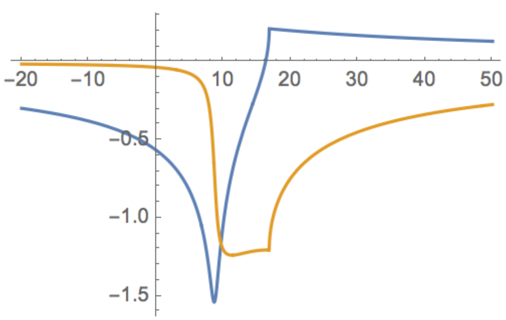
```
<< X`
<< CollierLink`
Package-X v2.1.0 [ALPHA], by Hiren H. Patel
For more information, see the guide
CollierLink v1.0.0 [ALPHA], by Hiren H. Patel
```

C_{0011}

```
PVC[1, 2, 0, 0, 0, 3.2, 1.0 - .2 I, 1.0 - .2 I, 2.0]
-0.0155049 + 0.010052 i
```

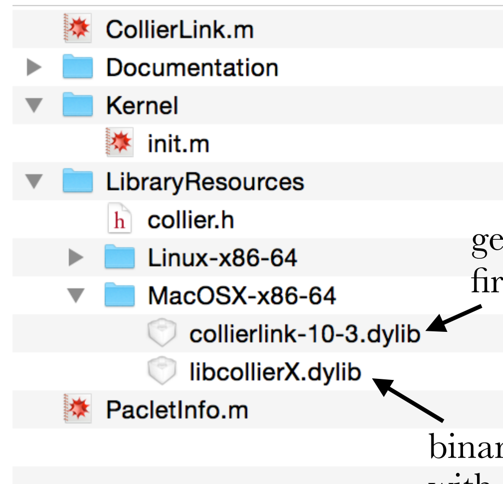
complex masses

```
Plot[{Re[PVC[0, 0, 0, 5.3, 12.1, s, 1.1, 1.2, 3.0]],
  Im[PVC[0, 0, 0, 5.3, 12.1, s, 1.1, 1.2, 3.0]]}, {s, -20, 50}]
```



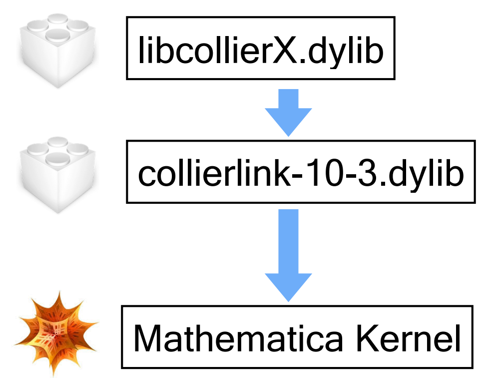

```
PVD[0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.0]
COLLIER::nocase : Error flag -10: Unable to
    evaluate PVD[0, 0, 0, 0, 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.];
    case is not supported or implemented.
Indeterminate
```

under the hood…



generated when first loaded

binary bundled with CollierLink



in Fortran: COLLIER

in C: LibraryLink

CollierLink

Three features -

2. Automatic code generation, compilation, and relinking

$$\int \frac{d^dk}{(2\pi)^d} \frac{(k^2+k.p+k.q+p.q)^2}{(k+p)^2\,[(k+q)^2-m^2]\,k^2}$$

```
myIntegral = LoopIntegrate[(k.k + k.p + k.q + p.q)^2, k, {k + p, 0}, {k + q, m}, {k, 0}] /.
  {p.p → m^2, q.q → 0, p.q → 1/2 (m^2 - t)}
```

$$\frac{3}{4}\,\text{PVA}[0,0] + \frac{1}{4}\,\text{PVA}[0,m] + \left(\frac{m^2}{4} + \frac{1}{2}(m^2-t)\right)\text{PVB}[0,0,0,0,m] +$$
$$\left(-\frac{m^2}{4} + \frac{1}{2}(m^2-t)\right)\text{PVB}[0,0,m^2,0,0] + \frac{1}{4}(m^2-t)\,\text{PVB}[0,1,0,0,m] +$$
$$\frac{1}{4}(m^2-t)\,\text{PVB}[0,1,m^2,0,0] + \frac{1}{4}(m^2-t)^2\,\text{PVC}[0,0,0,m^2,t,0,0,0,m]$$

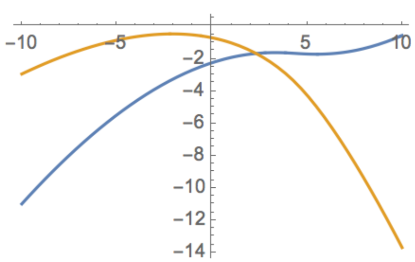
pretend this is many pages long…

```
f = CollierCompile[{t, m}, Evaluate[myIntegral]]
```

CollierCompiledFunction[Variables (2): {t, m} Denom: 3 (triangle) Rank: 1]

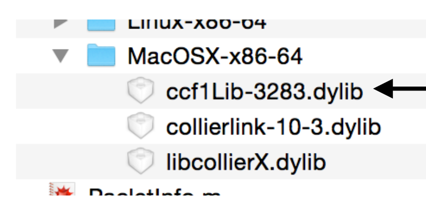
Evaluates with machine-code speed!

```
Plot[{Re[f[t, 2.1]], Im[f[t, 2.1]]}, {t, -10, 10}]
```



under the hood…

- **Optimize expression** (Experimental`OptimizeExpression) — makes abbreviations for common subexpressions
- **Convert expression to pseudocode**
- **Translate pseudocode to C code**
- **Compile with available compiler (clang or gcc)** — creates temporary ccf1Lib-xxxx.dylib
- **Relink**
- **Prime COLLIER cache system** — evaluate at few random points



temporary dylib

CollierLink

Three features -

2. Automatic code generation, compilation, and relinking

uses COLLIER's cache system —
Each CollierCompiledFunction gets its own cache

```
CollierCompile[{s, m}, PVA[0, m]]
```

CollierCompiledFunction[Variables (2): {s, m}
Denom: 1 (tadpole) Rank: 0
TN_cll calls: 1 cache_no: 1
Code lines: 2 File size: 14.80 kB]

```
CollierCompile[{s, m}, PVC[1, 0, 0, 0, 0, s, m, m, m]]
```

CollierCompiledFunction[Variables (2): {s, m}
Denom: 3 (triangle) Rank: 2
TN_cll calls: 1 cache_no: 2
Code lines: 2 File size: 14.84 kB]

```
CollierCompile[{s, m}, PVB[0, 0, s, m, m]]
```

CollierCompiledFunction[Variables (2): {s, m}
Denom: 2 (bubble) Rank: 0
TN_cll calls: 1 cache_no: 3
Code lines: 2 File size: 14.84 kB]

No conflicts

And CollierCompiledFunction can be parallelized!

CollierCompiledFunction

`CollierCompiledFunction[`*args*`, …]`
represents compiled code for evaluating a compiled function using the COLLIER library.

Compile the infrared divergent scalar triangle function:

```
In[1]:= myfun = CollierCompile[{qSq}, PVC[0, 0, 0, 0, 0, qSq, 0, 0, 0]]

Out[1]= CollierCompiledFunction[ Variables (1): {qSq}
                                 Denom: 3 (triangle)  Rank: 0 ]
```

Load CollierLink on all parallel kernels with `ParallelNeeds`, and distribute the definition of `myFun` to parallel kernels:

```
In[2]:= ParallelNeeds["CollierLink`"]
        DistributeDefinitions[myfun]

Out[2]= {myfun}
```

Compute a table of values in parallel:

```
In[3]:= ParallelTable[myfun[qSq], {qSq, 5.1, 10.3}]

Out[3]= {-0.868639 - 1.00361 i, -0.675789 - 0.931296 i, -0.540322 - 0.867299 i,
         -0.440657 - 0.811331 i, -0.364729 - 0.762362 i, -0.305283 - 0.719311 i}
```

Each parallel kernel gets its own copy of dylib

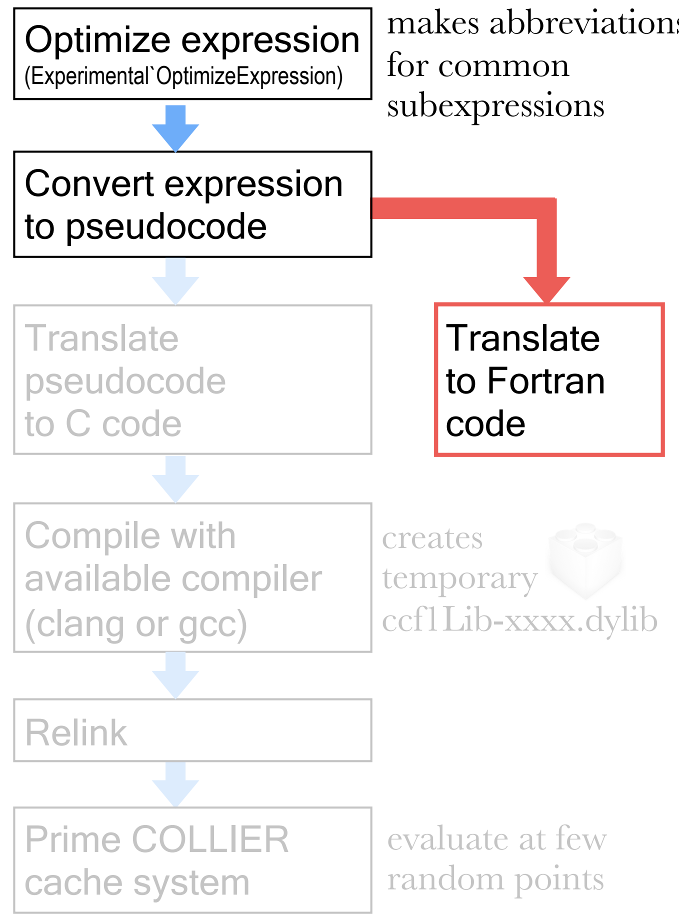
CollierLink

Three features -

3. Automatic Fortran code generation

```
CollierCodeGenerate[{t, m}, Evaluate[myIntegral]]

!!*****************************************************************************!!
!! This code was automatically created using Package-X v2.1.0 CollierLink     !!
!!  routine CollierCodeGenerate[] by Hiren H. Patel, and requires the        !!
!!  Fortran library COLLIER v1.1 by A. Denner, S. Dittmaier, L. Hofer         !!
!!  for numerical evaluation.                                                !!
!!-----------------------------------------------------------------------------!!
!!  Language: Fortran                  Creation date: Aug-21-2017 21:15      !!
!!  Function name: generatedFunction   Arguments (2): {t,m}                  !!
!!  Author: hhpatel                                                          !!
!!  Initialization requirement: Init_cll(3,1,'')                             !!
!!*****************************************************************************!!

  DOUBLE COMPLEX FUNCTION generatedFunction(inputVar1, inputVar2)

    USE COLLIER
    IMPLICIT NONE

    DOUBLE COMPLEX, INTENT(IN) :: inputVar1  ! t
    DOUBLE COMPLEX, INTENT(IN) :: inputVar2  ! m

    !Allocate memory for calculated results of Passarino-Veltman,
    !   coefficient functions, and their UV divergent parts
    DOUBLE COMPLEX :: pvx1(1), pvx1uv(1)   ! PVA[_,0]
    DOUBLE COMPLEX :: pvx2(1), pvx2uv(1)   ! PVA[_,m]
    DOUBLE COMPLEX :: pvx3(2), pvx3uv(2)   ! PVB[_,_,0,0,m]
    DOUBLE COMPLEX :: pvx4(2), pvx4uv(2)   ! PVB[_,_,Compile`optVar5,0,0]
    DOUBLE COMPLEX :: pvx5(3), pvx5uv(3)   ! PVC[_,_,_,Compile`optVar5,t,0,0,0,m]

    !Local variables
    DOUBLE COMPLEX :: localVar1  ! optVar5
    DOUBLE COMPLEX :: localVar2  ! optVar7
    DOUBLE COMPLEX :: localVar3  ! optVar8
    DOUBLE COMPLEX :: localVar4  ! optVar9
    DOUBLE COMPLEX :: localVar5  ! optVar21

    !Evaluation code
    localVar1 = (inputVar2**2)
    localVar2 = (-inputVar1)
    localVar3 = (localVar1+localVar2)
    localVar4 = ((1.d0/2.d0)*localVar3)
    localVar5 = (localVar3**2)
```

under the hood…



Idea is to be able to easily plug this code into popular event generators MCFM, MadGraph, …

Summary

★ Introduced *Package*-X, described its salient features

★ Described updates in upcoming release *Package*-X v2.1

★ Introduced new package, CollierLink

1. *Direct numerical evaluation of Passarino-Veltman functions*
2. *Automatic code generation, compilation, and relinking*
3. *Automatic Fortran code generation*

Conclusions:

Together with CollierLink, *Package*-X provides a user-friendly yet *truly comprehensive tool* for the calculation of one loop integrals.

Applications: anything… ~~except fully differential cross sections~~